

ITEM NO: 1929414



**FOR
REFERENCE ONLY**



THRUSTER FAULT DIAGNOSIS AND ACCOMMODATION FOR OVERACTUATED OPEN-FRAME UNDERWATER VEHICLES

Thesis submitted to the University of Wales for the Degree of
Doctor of Philosophy

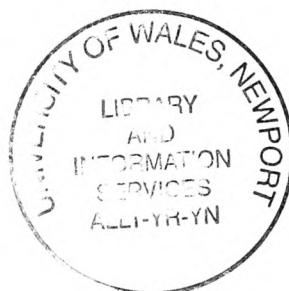
by

Edin Omerdic, BSc., MSc.

Mechatronics Research Centre

University of Wales College, Newport

February 2004



Trademarks

Windows 98TM, Windows 2000TM and Windows XPTM are trademarks of Microsoft Corporation.

MATLABTM, SimulinkTM, Dials & Gauges BlocksetTM, Aerospace BlocksetTM, Fuzzy Logic ToolboxTM,

Neural Networks ToolboxTM and Virtual Reality ToolboxTM are trademarks of The MathWorks Inc.

Declaration / Statements

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed Omerdic Edm (candidate)

Date 10/06/2004

STATEMENT 1

This thesis is a result of my own investigations, except where otherwise is stated.

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed Omerdic Edm (candidate)

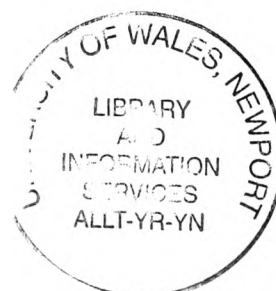
Date 10/06/2004

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed Omerdic Edm (candidate)

Date 10/06/2004



In memory of my father

Acknowledgements

In retrospect, my journey to the wonderful world of control of marine vessels began in November 1995 – a discussion with Professor Ljubomir Kuljača and Professor Zoran Vukić (Faculty of Electrical Engineering and Computing, Zagreb, Croatia) about the subject for my undergraduate thesis opened the door of this mystic world for me. They told me that my work will be concentrated on the application of fuzzy logic in ship control. I still remember the astonishment in my reply: "What? Ship? Fuzzy logic?" I simply did not know anything about ships and fuzzy logic at that time.

Over the past eight years, I resolved many secrets on this voyage. Now it is time to acknowledge all of those who helped me to finish the important stage of this journey.

First of all, I would like to thank my wonderful mother Hadžira, younger brother Emir and his wife Nermina for providing inspiration, love and support throughout my education.

Next, I would like to thank my supervisor, Professor Geoff Roberts, for improving the thesis by reading early drafts and supplying corrections and critical advices.

Thanks to Professor Tor A. Johansen, Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, for fruitful discussion about limits of the pseudoinverse method.

I am grateful to Professor Marc Bodson, Electrical and Computer Engineering, University of Utah, and Adjunct Associate Professor Dale Frederick Enns, Aerospace Engineering & Mechanics, University of Minnesota, for sending copies of their papers about control allocation to me.

Special thanks to Alex for his encouragement and enthusiasm. After my coming to Newport, Alex and his girlfriend, Mandi, helped me to understand and adapt to a new way of life, quite different than my previous experience.

Thanks to Nathan, Tristan and Gurvinder for supplying corrections and constructive suggestions.

I would also like to thank the other members of the Mechatronics Research Centre (especially to Mokhtar, Tasos, Fangmin, Eric and Chinese friends) for creating an enjoyable working environment.

Thanks to Jane, our secretary, for her readiness to help me whenever I needed it.

I would like to thank my housemates (Alfredo, Pernille, Michalis, Karen, Dave and Luca) who lived with me for the last three years and who were forced sometimes to share with me sadness when something goes wrong and the joy of new discovery.

I would like to acknowledge the help and support that Pere, Marc, Bianca, Carlos, Joan and other researchers and friends from Girona provided to me during my stays in this beautiful town.

Special thanks to Sylvia and Merce, my housemates in Girona, for their patience and support.

Finally, I would like to thank to Isela (my Bonnie) for providing constant source of love, support and inspiration. She was my lighthouse on the journey, which inspired me to persevere in sailing and successfully completing the most important stage of the journey.



Summary

The work presented in the thesis concerns the design and development of a novel thruster fault diagnosis and accommodation system (FDAS) for overactuated, open-frame underwater vehicles. The remotely operated vehicles (ROVs) considered in this thesis have four thrusters for motion in the horizontal plane with three controllable degrees of freedom (DoF). Due to the redundancy resulting from this configuration, for the case of a partial fault or a total fault in a single thruster it is possible to reallocate control among operable thrusters in order that the ROV pilot is able to maintain control of the faulty ROV and to continue with missions.

The proposed FDAS consists of two subsystems: a fault diagnosis subsystem (FDS) and a fault accommodation subsystem (FAS). The FDS uses fault detector units to monitor thruster states. Robust and reliable interrogation of thruster states, and subsequent identification of faults, is accomplished using methods based on the integration of self-organising maps and fuzzy logic clustering. The FAS uses information provided by the FDS to perform an appropriate redistribution of thruster demands in order to accommodate faults. The FAS uses a hybrid approach for control allocation, which integrates the pseudoinverse method and the fixed-point iterations method. A control energy cost function is used as the optimisation criteria. In fault-free and faulty cases the FAS finds the optimal solution, which minimises this criteria. The concept of feasible region is developed in order to visualise thruster velocity saturation bounds. The FDAS provides a dynamic update of saturation bounds using a complex three-dimensional visualisation of the feasible region (attainable command set), such that the ROV pilot is informed with the effects of thruster fault accommodation, incorporated in the new shape of the attainable command set. In this way the ROV pilot can easily adapt to newly created changes and continue the mission in the presence of a fault.

The prototype of the FDAS was developed in the MATLAB environment as a Simulink model, which includes a nonlinear model of an ROV with 6 DOF, propulsion system and a hand control unit. The hand control unit was simulated in hardware using a joystick as input device to generate command signals. Different fault conditions are simulated in order to investigate the performance of the FDAS. A virtual underwater world was developed, which enabled tuning, testing and evaluation of the FDAS using simulations of two underwater vehicles (FALCON, Seaeye Marine Ltd. and URIS, University of Girona) in a 'realistic' underwater environment.

The performance of the FDAS was demonstrated and evaluated via tank trials of the FALCON ROV in QinetiQ Ocean Basin Tank at Haslar, where the existing control software was enhanced with the FDAS algorithm. The results of real-world experiments confirmed the effectiveness of the FDAS in maintaining vehicle manoeuvrability and in preserving the vehicle mission in the presence of thruster faults.

Nomenclature

Symbols

Neural networks:

$\mathbf{u}(k)$	– Input vector
$\mathbf{y}(k)$	– Actual output vector
$\mathbf{F}(\cdot, \dots, \cdot)$	– General non-linear function
\mathbf{W}	– Weight matrix
n	– System order
$NN(\cdot, \dots, \cdot)$	– Neural network-based non-linear functional mapping
$\hat{\mathbf{y}}(k)$	– Predicted output vector
$\mathbf{r}(k)$	– Residual vector

Direct control allocation:

Φ	– Attainable Moment Set
\mathbf{m}	– Moment vector
Ω	– Set of control constraints
\mathbf{m}_d	– Desired moment vector
a	– Scaling factor
\mathbf{u}	– True control vector
\mathbf{B}	– Control effectiveness matrix
k	– Number of rows of \mathbf{B}

Generalised inverse:

\mathbf{B}	– Control effectiveness matrix
\mathbf{P}	– Right generalised inverse
\mathbf{B}^+	– Moore-Penrose pseudoinverse of \mathbf{B}
\mathbf{W}	– Positive definite, weighting matrix
\mathbf{B}_w^+	– Weighted pseudoinverse of \mathbf{B}

- a – Scaling factor
- y_d – Desired virtual control input

Optimisation based methods:

- y – Virtual control input
- u_{\min}, u_{\max} – Control constraints
- u_p – Preferred true control vector
- B – Control effectiveness matrix
- J – Criteria
- ε – Adjustable parameter

Fault-tolerant system design of AUV:

- $\tau = \begin{bmatrix} X \\ Y \\ Z \\ K \\ M \\ N \end{bmatrix}$ – Generalised vector of total forces and moments exerted by thrusters

- TCM** – Thruster control matrix
- f – Vector of individual thruster forces
- s – $\sin 45^\circ$
- R_v – Radial distance from centre to the centre of the vertical thruster
- R_h – Radial distance from centre to the centre of the horizontal thruster
- HT_i – i^{th} horizontal thruster
- VT_i – i^{th} vertical thruster
- x – Surge direction in body-fixed frame
- y – Sway direction in body-fixed frame
- V – Input voltage
- F_T – Thruster force

Fault tolerant control of an AUV under thruster redundancy:

\mathbf{q}	– Generalised position and orientation vector in the Earth-fixed frame
\mathbf{w}	– Linear and angular velocity vector in the body-fixed frame
\mathbf{M}	– Inertia matrix, including the rigid body and the added mass terms
\mathbf{C}	– Matrix of centrifugal and Coriolis terms, including the rigid body and the added mass terms
\mathbf{D}	– Matrix of hydrodynamic drag terms
\mathbf{G}	– Vector of restoring forces (gravity and buoyancy)
$\boldsymbol{\tau}$	– Vector of generalised forces, exerted by thrusters
n	– Number of thrusters
\mathbf{F}_i	– Vector of thruster forces
\mathbf{E}	– Thruster configuration matrix
$\boldsymbol{\zeta}$	– Substitution for $\mathbf{C}(\mathbf{w})\mathbf{w} + \mathbf{D}(\mathbf{w})\mathbf{w} + \mathbf{G}(\mathbf{q})$
\mathbf{B}	– Transformation matrix
\mathbf{x}	– Position and orientation vector in task space
m	– Dimension of task space
\mathbf{J}	– Jacobian matrix
$\boldsymbol{\mu}$	– Thruster control matrix
$\boldsymbol{\beta}$	– Vector of non-linear terms
$\boldsymbol{\mu}^+$	– Moore-Penrose pseudoinverse of $\boldsymbol{\mu}$
$\boldsymbol{\mu}_w^+$	– Weighted pseudoinverse of $\boldsymbol{\mu}$
\mathbf{W}	– Weighting matrix
$\boldsymbol{\Psi}$	– Thruster fault matrix

Optimal distribution of propulsion and control forces:

p	– Number of control inputs (thrusters)
n	– Number of controllable DOF
\mathbf{u}	– Control vector
n	– Propeller angular velocity
$\boldsymbol{\tau}$	– Vector of forces and moments exerted by thrusters

\mathbf{B}	– Thruster control matrix
\mathbf{W}	– Weighting matrix
J	– Criteria (Control energy cost function)
L	– Lagrangian
λ	– Lagrange multipliers
\mathbf{B}_w^+	– Weighted pseudoinverse of \mathbf{B}
$\boldsymbol{\tau}_d$	– Desired vector of forces and moments exerted by thrusters

ROV model:

$\{B\}$	– Body-fixed frame
$\{E\}$	– Earth-fixed frame
O	– Origin of body-fixed frame
CG	– Centre of gravity
${}^E\boldsymbol{\eta} = \begin{bmatrix} {}^E\boldsymbol{\eta}_1 \\ {}^E\boldsymbol{\eta}_2 \end{bmatrix}$	– Position and orientation vector
${}^E\boldsymbol{\eta}_1 = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$	– Position vector
${}^E\boldsymbol{\eta}_2 = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}$	– Orientation vector
${}^B\mathbf{v} = \begin{bmatrix} {}^B\mathbf{v}_1 \\ {}^B\mathbf{v}_2 \end{bmatrix}$	– Linear and angular velocity vector
${}^B\mathbf{v}_1 = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$	– Linear velocity vector
${}^B\mathbf{v}_2 = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$	– Angular velocity vector
${}^E\mathbf{e}$	– Vector for attitude representation using Euler parameters

${}^E\boldsymbol{\sigma}$	– Vector for attitude representation using Modified Rodrigues parameters
$\mathbf{J}_1({}^E\boldsymbol{\eta}_2)$	– Transformation matrix ${}^B\mathbf{v}_1 \rightarrow {}^E\dot{\boldsymbol{\eta}}_1$
$\mathbf{J}_2({}^E\boldsymbol{\eta}_2)$	– Transformation matrix ${}^B\mathbf{v}_2 \rightarrow {}^E\dot{\boldsymbol{\eta}}_2$
\mathbf{q}	– Quaternion
w, x, y, z	– Real parameters
$\varepsilon_1, \varepsilon_2, \varepsilon_3, \eta$	– Real parameters
$\mathbf{i}, \mathbf{j}, \mathbf{k}$	– Imaginary units
$\boldsymbol{\lambda}$	– Principal unit vector
β	– Principal angle
\mathbf{p}	– Point in 3D space
$\mathbf{E}_1({}^E\mathbf{e})$	– Transformation matrix ${}^B\mathbf{v}_1 \rightarrow {}^E\dot{\boldsymbol{\eta}}_1$
$\mathbf{E}_2({}^E\mathbf{e})$	– Transformation matrix ${}^B\mathbf{v}_2 \rightarrow {}^E\dot{\mathbf{e}}$
$\boldsymbol{\rho} = \begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \end{bmatrix}$	– Vector of Cayley-Rodrigues parameters
$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{bmatrix}$	– Vector of Modified Rodrigues parameters
$\mathbf{J}_1({}^E\boldsymbol{\sigma})$	– Transformation matrix ${}^B\mathbf{v}_1 \rightarrow {}^E\dot{\boldsymbol{\eta}}_1$
$\mathbf{J}_2({}^E\boldsymbol{\sigma})$	– Transformation matrix ${}^B\mathbf{v}_2 \rightarrow {}^E\dot{\boldsymbol{\sigma}}$
\mathbf{M}_{RB}	– Rigid-body inertia matrix
${}^B\mathbf{r}_G$	– Position of CG
${}^B\mathbf{r}_O$	– Position of O
\mathbf{I}_0	– Inertia tensor
I_x, I_y, I_z	– Moments of inertia about X_B, Y_B and Z_B -axes
I_{xy}, I_{yz}, I_{zx}	– Products of inertia
ρ_A	– Mass density
\mathbf{I}_C	– Inertia tensor about CG

$\mathbf{C}_{RB}({}^B\mathbf{v})$	– Rigid-body Coriolis and centripetal matrix
m	– Mass of the ROV
${}^B\boldsymbol{\tau}_{RB}$	– Generalised vector of external forces and moments (including control and hydrodynamic forces and moments)
\mathbf{M}_A	– Added-mass matrix
\mathbf{M}	– Inertia matrix (including added mass)
$\mathbf{C}_A({}^B\mathbf{v})$	– Added-mass Coriolis and centripetal matrix
$\mathbf{C}({}^B\mathbf{v})$	– Coriolis and centripetal matrix (including added mass)
$\mathbf{D}({}^B\mathbf{v})$	– Total hydrodynamic damping matrix
${}^B\mathbf{g}({}^E\boldsymbol{\eta})$	– Vector of restoring (gravitational and buoyant) forces and moments
${}^B\boldsymbol{\tau}_E$	– Vector of environmental forces and moments
${}^B\boldsymbol{\tau}$	– Vector of propulsion forces and moments (exerted by the thrusters)
g	– Acceleration due to gravity
ρ	– Fluid (water) density
∇	– Volume of fluid (water) displaced by the ROV
W	– Weight of the ROV
B	– Buoyant force
p	– Number of thrusters in the general case
iTh	– General thruster
${}^i\mathbf{T}$	– Propeller thrust (force)
${}^i\mathbf{Q}_e$	– Propeller torque (moment), generated by rotation
${}^i\mathbf{Q}_r$	– Propeller torque (moment), generated by ${}^i\mathbf{T}$
${}^i\mathbf{Q}$	– Total torque (moment), exerted by thruster
iHT	– i^{th} horizontal thruster
iVT	– i^{th} vertical thruster
${}^i\mathbf{r}$	– Position vector of the thruster iTh relative to O
${}^i\mathbf{e}$	– Orientation vector of the thruster iTh
i_c	– Spin direction coefficient
a	– Distance between thrusters 1HT & 2HT (4HT & 3HT) (X-shaped thruster configuration)

b	– Distance between thrusters 1HT & 4HT (2HT & 3HT) (X-shaped thruster configuration)
α	– Angle between horizontal thruster and positive direction of x_B axis (X-shaped thruster configuration)
R	– Radial distance between horizontal thruster and O (cross-shaped thruster configuration)

Propeller shaft speed models:

n	– Propeller shaft speed
T	– Propeller thrust
u_p	– Axial flow velocity
τ	– Control input (shaft torque)
u	– Forward speed
Q_e	– Propeller torque
X_u	– Coefficient of linear laminar skin friction
$X_{u u }$	– Coefficient of non-linear quadratic drag
d_{f0}	– Coefficient of linear damping
d_f	– Coefficient of quadratic damping
u_a	– Ambient water velocity
w	– Wake fraction number
K_T, K_{Q_e}	– Non-dimensional thrust and torque coefficients
D	– Propeller diameter
ρ	– Water density
J_0	– Advance ratio
η_0	– Open water propeller efficiency
$\alpha_1, \alpha_2, \beta_1, \beta_2$	– Positive non-dimensional coefficients
$T_{n n }, T_{n u_a}, Q_{n n }$	– Positive coefficients of the bilinear thruster model
$Q_{ n u_a}$	
$T_{n n }^+$	– Coefficient $T_{n n }$ for $n > 0$

$T_{n n }^-$	– Coefficient $T_{n n }$ for $n < 0$
u	– Control variable
u'	– Auxiliary control variable
K	– Average coefficient $T_{n n }$

Full thruster model:

\bar{n}_d	– Desired angular velocity
n_d	– Desired angular velocity (reduced by GR)
\bar{n}	– Actual angular velocity
n	– Actual angular velocity (reduced by GR)
GR	– Gear ratio
L_a	– Armature inductance
R_a	– Armature resistance
U_a	– Armature voltage
K_M	– Motor torque constant
J_m	– Moment of inertia of motor and thruster
ω	– Angular velocity if the motor
Q_e	– Load from the propeller

General control allocation:

\mathbf{v}	– Virtual control input
\mathbf{v}_d	– Desired virtual control input
\mathbf{u}	– True control input
\mathbf{v}_{sys}	– Total control effect
k	– Dimension of the virtual control space
m	– Dimension of the true control space
\mathbf{g}	– Mapping from the true to the virtual control input, performed by the actuators
\mathbf{B}	– Control effectiveness matrix

$\mathbf{u}_{\min}, \mathbf{u}_{\max}$	– Limits for actuator position constraints
ρ_{\min}, ρ_{\max}	– Limits for actuator rate constraints
T	– Sampling time
\mathbb{K}	– Intersection of hyperplanes defined by $\mathbf{B}\mathbf{u} = \mathbf{v}$
Ω	– Constrained control subset
$\partial(\Omega)$	– Boundary of Ω
$\underline{\Omega}$	– Normalised constrained control subset
$\partial(\underline{\Omega})$	– Boundary of $\underline{\Omega}$
Φ	– Attainable command set
$\partial(\Phi)$	– Boundary of Φ
$\underline{\Phi}$	– Normalised attainable command set
$\partial(\underline{\Phi})$	– Boundary of $\underline{\Phi}$
\mathcal{S}	– Solution set (intersection of \mathbb{K} and Ω)
\mathbf{N}	– Normal vector to the plane
l	– Line
t	– Parameter of the line
\mathbf{u}_p	– Preferred position of the actuators
$\mathbf{W}_u, \mathbf{W}_v$	– Weighting matrices
Ψ	– Set of feasible control inputs that minimise $\mathbf{B}\mathbf{u} - \mathbf{v}$
\mathbf{B}^+	– Pseudoinverse of \mathbf{B}
r	– Radius of sphere
l_p	– Norm
b	– Element of \mathbf{B}
J	– Criteria
ε	– Parameter used in J
$\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{H},$	– Temporary parameters used in the fixed-point iterations method
η	
\mathbf{u}_0	– Initial iteration the fixed-point iterations method
tol	– Threshold for stopping the fixed-point iterations method
\mathbf{G}	– Substitution for $\mathbf{W}_u^{-1}(\mathbf{B}\mathbf{W}_u^{-1})^+$

Φ_v	– Virtual control space
Ω_v	– Image of Φ_v , obtained by applying \mathbf{B}^+
Ω_e	– Image of Φ , obtained by applying \mathbf{B}^+
Φ_p	– Feasible region for pseudoinverse
Ω_p	– Image of Φ_p , obtained by applying \mathbf{B}^+
$\mathbf{B}_{w_u}^+$	– Weighted pseudoinverse of \mathbf{B}
\mathbf{u}^*	– Approximation of unfeasible $\mathbf{u} = \mathbf{B}^+ \mathbf{v}$
\mathbf{v}^*	– Approximation of \mathbf{v}
\mathbf{e}	– Approximation error
\mathbf{u}_t^*	– T -approximation of unfeasible $\mathbf{u} = \mathbf{B}^+ \mathbf{v}$
\mathbf{u}_s^*	– S -approximation of unfeasible $\mathbf{u} = \mathbf{B}^+ \mathbf{v}$
\mathbf{u}_f^*	– Approximation of unfeasible $\mathbf{u} = \mathbf{B}^+ \mathbf{v}$ obtained by fixed-point iterations
θ	– Direction error of approximation
$\ \mathbf{e}\ _2$	– Magnitude error of approximation
f	– Scaling factor for S -approximation
\mathbf{P}	– Generalised inverse of \mathbf{B}

Direct control allocation:

\mathbf{v}	– Virtual control input
\mathbf{v}_d	– Desired virtual control input
$\hat{\mathbf{v}}_d$	– Unit vector in the direction of \mathbf{v}_d
\mathbf{u}	– True control input
a	– Scaling factor for direct control allocation

Daisy chain control allocation:

\mathbf{v}	– Virtual control input
\mathbf{v}_d	– Desired virtual control input

m	– Number of true control inputs
\mathbf{u}	– True control input
$\mathbf{u}^1, \dots, \mathbf{u}^M$	– Partitions of \mathbf{u}
$\mathbf{B}_1, \dots, \mathbf{B}_M$	– Partitions of \mathbf{B}
k	– Dimension of \mathbf{v}

Control allocation for underwater vehicles:

P	– Number of thrusters
\mathbf{T}	– Thruster configuration matrix
\mathbf{f}	– Vector of control forces
\mathbf{K}	– Force coefficient matrix
\mathbf{u}'	– Control vector
\mathbf{B}	– Thruster control matrix
$\boldsymbol{\tau}$	– Total vector of propulsion forces and moments
A	– Substitution for $\frac{b}{2}\sin\alpha + \frac{a}{2}\cos\alpha$
τ_x, τ_y, τ_z	– Surge, sway and yaw force
τ_K, τ_M, τ_N	– Roll, pitch and yaw moment
T_m	– Maximum thruster force
u_m'	– Maximum auxiliary control variable
τ_{xm}	– Maximum surge force
τ_{ym}	– Maximum sway force
τ_{Nm}	– Maximum yaw moment
$\underline{\mathbf{u}}^{HT}$	– Normalised true control input (horizontal plane)
$\underline{\boldsymbol{\tau}}^{HT}$	– Normalised virtual control input (horizontal plane)
$\underline{\mathbf{B}}^{HT}$	– Control effectiveness matrix (horizontal plane)
$\mathbf{N}_x, \mathbf{N}_y, \mathbf{N}_N$	– Normal vector to the planes π_x, π_y and π_N , respectively
$\underline{\mathbf{K}}^{HT}$	– Intersection of the planes π_x, π_y and π_N (horizontal plane)
$\underline{\boldsymbol{\Omega}}^{HT}$	– Normalised constrained control subset (horizontal plane)

$\underline{\mathcal{S}}^{HT}$	– Solution set (intersection of $\underline{\mathcal{K}}^{HT}$ and $\underline{\Omega}^{HT}$, horizontal plane)
$\underline{\Phi}^{HT}$	– Attainable command set (horizontal plane)
$\underline{\Phi}_v^{HT}$	– Virtual control space (horizontal plane)
$\underline{\Phi}_p^{HT}$	– Feasible region for pseudoinverse (horizontal plane)
\mathbf{W}_u^{HT}	– Weighting matrix (horizontal plane)
s_i^{HT}	– Normalised saturation (constraint) bound for iHT (horizontal plane)
w_i^{HT}	– Weight for iHT (horizontal plane)
$\underline{\mathbf{B}}_{W_u^{HT}}^{HT}$	– Weighted pseudoinverse of $\underline{\mathbf{B}}^{HT}$ (horizontal plane)
τ_{Zm}	– Maximum heave force
\mathbf{W}_u^{VT}	– Weighting matrix (vertical plane)
w_j^{VT}	– Weight for jVT (vertical plane)
$\underline{\mathbf{B}}^{VT}$	– Control effectiveness matrix (vertical plane)
$\underline{\mathbf{B}}_{W_u^{VT}}^{+VT}$	– Weighted pseudoinverse of $\underline{\mathbf{B}}^{VT}$ (vertical plane)

Fault diagnosis and accommodation system:

$\underline{\tau}_d^*$	– Desired vector of propulsion forces and moments, generated by the HCU
$\underline{\tau}_d$	– Filtered $\underline{\tau}_d^*$
\mathbf{f}	– Total fault indicator vector
f_i	– Fault indicator (output of the FDU for iTh)
$\underline{\tau}_d^{HT}$	– Partition of $\underline{\tau}_d$ (horizontal plane)
$\underline{\tau}_d^{VT}$	– Partition of $\underline{\tau}_d$ (vertical plane)
$\underline{\mathbf{u}}^{HT}$	– Pseudoinverse solution (horizontal plane)
$\underline{\mathbf{u}}^{VT}$	– Pseudoinverse solution (vertical plane)
$\underline{\mathbf{u}}^{*HT}$	– Approximation of $\underline{\mathbf{u}}^{HT}$
$\underline{\mathbf{u}}^{*VT}$	– Approximation of $\underline{\mathbf{u}}^{VT}$
$\underline{\mathbf{u}}^*$	– Composition of $\underline{\mathbf{u}}^{*HT}$ and $\underline{\mathbf{u}}^{*VT}$

$\underline{\mathbf{n}}$	– Normalised vector of desired thruster velocities
\mathbf{n}	– Transformed $\underline{\mathbf{n}}$
n	– Actual velocity of the motor shaft
n_d	– Desired velocity of the motor shaft
I	– Motor current
\mathbf{x}	– Feature vector
h	– Logical function
q	– Number of closest BMUs to \mathbf{x} from each SOM prototype
${}^k\mathbf{BMU}_j$	– j^{th} BMU in SOM k
${}^k d_j$	– Euclidian distance between \mathbf{x} and ${}^k\mathbf{BMU}_j$
\mathbf{M}	– Matrix of distances ${}^k d_j$
\mathbf{m}	– Vector of minimum values in columns of \mathbf{M}
\overline{m}	– Mean of \mathbf{m}
\mathbf{b}	– Vector of indices of minimum values in columns of \mathbf{M}
\mathbf{B}	– Buffer
w	– Substitution for $\sum_{i=1}^4 w_i^{HT}$
$\underline{\mathcal{L}}_X, \underline{\mathcal{L}}_Y, \underline{\mathcal{L}}_Z, \underline{\mathcal{L}}_N$	– Normalised surge, sway and yaw force and yaw moment
V	– Volume

Acronyms & abbreviations

AI	– Artificial Intelligence
AMS	– Attainable Moment Set
ANN	– Artificial Neural Network
ATC	– Advanced Thruster Control
AUV	– Autonomous Underwater Vehicle
CG	– Centre of Gravity
DP	– Diagnosis Part
DOF	– Degree of Freedom
EKF	– Extended Kalman Filter
EPSRC	– Engineering Physical Science Research Council
FAS	– Fault Accommodation Subsystem
FCM	– Fuzzy C-Means clustering
FCT	– Fault Code Table
FDAS	– Fault Diagnosis and Accommodation System
FDI	– Fault Detection and Isolation
FDS	– Fault Diagnosis Subsystem
FDU	– Fault Detector Unit
FPI	– Fixed-Point Iterations
FTC	– Fault Tolerant Control
GI	– Generalised Inverse
HCU	– Hand Control Unit
HT	– Horizontal Thruster
IFAC	– International Federation of Automatic Control
IMPROVES	– IMproving the Performance of Remotely Operated VehicleS
IMU	– Inertial Measurement Unit
LDV	– Laser-Doppler Velocimeter
MMP	– Model Matching Part
NEROV	– Norwegian Experimental Remotely Operated Vehicle
ODIN	– Omni Directional Intelligent Navigator
PCA	– Principal Component Analysis

PIV	– Particle Image Velocimeter
ROV	– Remotely Operated Vehicle
RPI	– Redistributed Pseudo Inverse
SISO	– Single Input Single Output
SNAME	– Society of Naval Architects and Marine Engineers
SOM	– Self-Organising Map
SUT	– Society for Underwater Technology
TCM	– Thruster Control Matrix
TCU	– Thruster Control Unit
UUV	– Unmanned Underwater Vehicle
URV	– Unmanned Robotic Vehicle
VRML	– Virtual Reality Modelling Language
VT	– Vertical Thruster

Table of Contents

Declaration / Statements	ii
Acknowledgements	iv
Summary	vi
Nomenclature	viii
Symbols	viii
Acronyms & abbreviations	xxi
Table of Contents	xxiii
Table of Figures	xxix
List of Tables	xxxv
List of Examples	xxxvii
List of Algorithms	xxxviii
CHAPTER 1: INTRODUCTION	1-1
1.1 BACKGROUND	1-2
1.2 MOTIVATION	1-4
1.3 AIMS AND OBJECTIVES	1-5
1.4 OVERVIEW OF CHAPTER CONTENTS	1-6
1.5 LIST OF MAIN CONTRIBUTIONS	1-8
CHAPTER 2: LITERATURE REVIEW	2-1
2.1 INTRODUCTION	2-2
2.2 BASIC CONCEPTS OF FAULT-TOLERANT CONTROL	2-3
2.3 FAULT DIAGNOSIS TERMINOLOGY	2-3
2.4 FAULT DIAGNOSIS METHODOLOGY	2-6
2.5 CLASSIFICATION OF FAULT DIAGNOSIS METHODS	2-7
2.5.1 Model-free methods	2-8
2.5.2 Model-based methods	2-10
2.6 SOME OF RECENT FAULT DIAGNOSIS APPROACHES	2-15
2.7 FUZZY LOGIC IN FAULT DIAGNOSIS	2-18
2.7.1 Fuzzy filter for residual evaluation	2-18
2.7.2 Fuzzy model-based parity equations for fault isolation	2-20
2.8 PATTERN RECOGNITION IN FAULT DIAGNOSIS	2-20
2.8.1 Neural networks	2-21
Neural networks as models of non-linear dynamic systems	2-21
Neural networks as classifiers	2-22
Fault diagnosis scheme based on neural networks	2-23
2.8.2 Self-organising maps	2-25

Process monitoring using SOM	2-25
Process analysis using SOM	2-26
Fault diagnosis using SOM	2-26
2.9 CONTROL ALLOCATION TECHNIQUES FOR AIRCRAFT	2-28
2.9.1 Ad hoc methods	2-29
2.9.2 Direct control allocation	2-29
2.9.3 Generalised inverse	2-30
2.9.4 Daisy chain method	2-32
2.9.5 Optimisation based methods	2-33
2.10 FAULT DIAGNOSIS AND ACCOMMODATION FOR UNDERWATER VEHICLES	2-35
Model-based approach for self-diagnosis of AUV (Takai and Ura, 1999)	2-36
On-line damage detection for AUV (Rae and Dunn, 1994)	2-38
Fault detection of actuator faults in UUV using bank of estimators (Alessandri, <i>et al.</i> , 1999)	2-39
Fault-tolerant system design of AUV (Yang, <i>et al.</i> , 1999; 1998)	2-41
ROV actuator fault diagnosis through servo-amplifiers' monitoring (Bono, <i>et al.</i> , 1999)	2-47
Fault tolerant control of an AUV under thruster redundancy (Podder, <i>et al.</i> , 2000)	2-47
Optimal distribution of propulsion and control forces (Fossen, 1995)	2-52
2.11 CONCLUDING REMARKS	2-53
2.12 REFERENCES	2-54
CHAPTER 3: MODELLING OF ROV & PROPULSION SYSTEM	3-1
3.1 INTRODUCTION	3-2
3.2 COORDINATE FRAMES	3-2
3.3 KINEMATIC EQUATIONS OF MOTION	3-3
3.3.1 Euler angles representation	3-4
3.3.2 Euler parameters (Unit quaternion) representation	3-6
3.3.3 Modified Rodrigues parameters	3-10
3.3.4 Comments on representation alternatives	3-12
3.4 DYNAMIC EQUATIONS OF MOTION	3-13
3.5 FORCES AND MOMENTS ACTING ON ROV	3-17
3.6 SIMULATION DIAGRAMS FOR ROV DYNAMICS AND KINEMATICS	3-23
3.7 PROPULSION SYSTEM	3-25
3.7.1 Thruster configuration	3-25
3.7.2 Propeller shaft speed models	3-29
One-state model	3-29
Two-state model	3-29
Three-state model	3-30
3.7.3 Propeller thrust and torque modelling	3-31
Quasi-steady thrust and torque	3-31
3.7.4 Thruster test rig (IMPROVES project)	3-39
3.7.5 Vectorisation	3-39

Bilinear thruster model	3-40
Affine thruster model	3-40
3.7.6 Full thruster model	3-40
3.8 CONCLUDING REMARKS	3-42
3.9 REFERENCES	3-43
CHAPTER 4: CONTROL ALLOCATION	4-1
4.1 INTRODUCTION	4-2
4.2 CONTROL SYSTEM ARCHITECTURE	4-2
4.3 THE CONTROL ALLOCATION PROBLEM	4-8
4.3.1 Problem formulation	4-8
4.3.2 Nomenclature for constrained control subset Ω	4-12
4.3.3 Nomenclature for attainable command set Φ	4-13
4.3.4 Remarks	4-15
Applicability	4-15
Difficulties	4-16
4.4 CONTROL ALLOCATION METHODS	4-17
4.4.1 Optimisation based methods	4-17
Problem statement	4-17
Choice of norm	4-18
Choice of the weighting matrix \mathbf{W}_u	4-21
Fixed-point method	4-24
Pseudoinverse methods	4-26
4.4.2 Direct control allocation	4-46
Problem statement	4-46
Description	4-46
4.4.3 Daisy chain control allocation	4-50
Description	4-50
4.5 CONCLUDING REMARKS	4-55
4.6 REFERENCES	4-56
CHAPTER 5: FAULT DIAGNOSIS AND ACCOMMODATION SYSTEM	5-1
5.1 INTRODUCTION	5-2
5.1.1 Conceptual elements	5-3
5.1.2 Requirements	5-7
5.1.3 Implementation issues	5-7
5.2 CONTROL ALLOCATION FOR UNDERWATER VEHICLES	5-9
5.2.1 Background	5-9
5.2.2 Normalisation	5-16
5.2.3 Problem formulation	5-20

5.2.4 Nomenclature	5-23
5.2.5 Introducing criteria in problem formulation	5-27
Weighting matrix \mathbf{W}_u^{HT} for fault-free case	5-28
Weighting matrix \mathbf{W}_u^{HT} for faulty situations	5-28
Geometric interpretation	5-29
5.2.6 Remarks on vertical thrusters	5-30
5.3 ARCHITECTURE OF THE FDAS	5-31
5.4 FAULT DIAGNOSIS SUBSYSTEM	5-35
5.4.1 Fault classification	5-35
5.4.2 Fault code table	5-36
5.4.3 Fault detection unit	5-36
5.4.4 Integration of fault indicators	5-50
5.5 FAULT ACCOMMODATION SUBSYSTEM	5-51
5.5.1 Description	5-51
5.5.2 Hybrid approach for control allocation	5-51
Pseudoinverse	5-51
Feasibility of pseudoinverse solution	5-53
Fixed-point iteration method	5-55
Co-ordinator	5-55
Feasible region for pseudoinverse	5-57
5.5.3 FAS algorithm	5-64
5.6 REMARKS ON IMPLEMENTATION ISSUES	5-65
5.7 CONCLUDING REMARKS	5-67
CHAPTER 6: TESTING AND EVALUATION OF THE FDAS	6-1
6.1 INTRODUCTION	6-2
6.2 EVALUATION OF THE FDU	6-2
6.3 SIMULATION RESULTS	6-4
6.3.1 (A) Fault-free case	6-7
(A1) Feasible pseudoinverse solution	6-8
(A2) Unfeasible pseudoinverse solution – activation of the fixed-point iterations	6-8
(A3) Approximation of an unattainable command input	6-10
(A4) Feasible trajectory – the trajectory lies inside the feasible region for pseudoinverse	6-12
(A5) Feasible trajectory – the trajectory lies inside the attainable command set	6-13
(A6) Partially unfeasible trajectory – the trajectory partially lies outside the attainable command set	6-14
(A7) Difference between affine and bilinear thruster model	6-16
(A8) Choice of propeller spin direction	6-17
(A9) Motion in vertical plane	6-19
(A10) Difference between symmetrical and non-symmetrical T-curve	6-20
6.3.2 (B) Faulty situations	6-22

(B1) Partial fault - "Jammed propeller"	6-24
(B2) Partial fault - "Heavy jammed propeller"	6-26
(B3) Partial fault - "Unknown state"	6-26
(B4) Total fault - "Broken propeller"	6-27
(B5) Consecutive faults – passing through the pipe	6-28
(B6) Consecutive faults – passing through the hole in the rock	6-29
6.4 REAL-TIME APPLICATION	6-59
6.4.1 Introduction	6-59
6.4.2 Experiment set-up	6-59
6.4.3 Day one	6-60
6.4.4 Day two	6-63
6.5 CONCLUDING REMARKS	6-75
CHAPTER 7: CONCLUSIONS AND FURTHER WORK	7-1
7.1 INTRODUCTION	7-2
7.2 REVIEW OF THE THESIS	7-2
7.3 REALISATION OF AIMS AND OBJECTIVES	7-7
7.3.1 Aims	7-7
7.3.2 Objectives	7-7
7.4 MAIN CONTRIBUTIONS	7-9
7.5 FURTHER WORK	7-12
APPENDICES	
APPENDIX A: ROV MODELS – TECHNICAL DETAILS	A-1
Introduction	A-1
Underwater Robotic Intelligent System (URIS)	A-1
FALCON	A-18
APPENDIX B: SOME RESULTS FROM OPTIMAL CONTROL THEORY	B-1
References	B-4
APPENDIX C: SOME RESULTS FROM 3D GEOMETRY	C-1
Introduction	C-1
The intersection of two planes	C-1
The intersection of three planes	C-2
APPENDIX D: ROV SIMULATOR	D-1
Introduction	D-1
Requirements	D-1
Quick start	D-2
ROV simulator history	D-3
Description	D-7
Hand Control Unit & Joystick	D-7
Signal conditioning	D-9

Fault diagnosis subsystem	D-10
Fault accommodation subsystem	D-12
Propulsion system	D-14
ROV model	D-15
Display (User interface)	D-17
Virtual Reality Display	D-22
Run-time behaviour	D-34
References	D-35
APPENDIX E: PUBLISHED PAPERS & AWARDS	E-1
List of published and submitted papers	E-1
List of awards	E-1

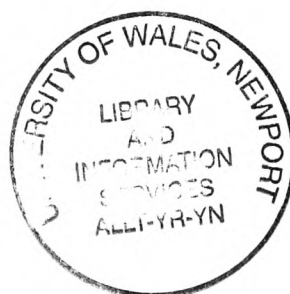


Table of Figures

Figure 1.1 Pictures captured by ROV on-board cameras showing the harsh nature of the operational environment. Courtesy of Seaeye Marine Ltd.	1-4
Figure 2.1 Classification of fault diagnosis methods.	2-8
Figure 2.2 Hardware vs analytical redundancy.	2-11
Figure 2.3 Model-based fault diagnosis scheme (analytical redundancy).	2-11
Figure 2.4 Classification of different residual generation concepts.	2-13
Figure 2.5 Classification of different residual evaluation concepts.	2-14
Figure 2.6 General structure of the fuzzy-filter-based diagnostic concept.	2-19
Figure 2.7 The neural network model of a non-linear dynamic system.	2-22
Figure 2.8 Decision-making mechanism in fault diagnosis system using pattern classifier.	2-22
Figure 2.9 Fault detection and isolation scheme using two neural networks.	2-24
Figure 2.10 Application of the SOM in industrial process monitoring.	2-26
Figure 2.11 Daisy chain allocation method.	2-33
Figure 2.12 Self-diagnosis system for AUV, proposed in Takai and Ura (1999).	2-37
Figure 2.13 Types of damage.	2-39
Figure 2.14 An observer-based fault-diagnostic scheme.	2-40
Figure 2.15 ODIN in the underwater environment.	2-42
Figure 3.1 Body-fixed and Earth-fixed coordinate frames for ROV.	3-3
Figure 3.2 General case: centre of gravity CG does not coincide with the origin O.	3-13
Figure 3.3 Thrust and torque, exerted by a thruster, for two possible propeller spin directions.	3-22
Figure 3.4 Simulation diagram for ROV dynamics and kinematics, where the attitude is represented with Euler angles.	3-23
Figure 3.5 Simulation diagram for ROV dynamics and kinematics, where the attitude is represented with Euler parameters.	3-24
Figure 3.6 Simulation diagram for ROV dynamics and kinematics, where the attitude is represented with Modified Rodrigues parameters.	3-24
Figure 3.7 Two common configurations of the horizontal thrusters.	3-26
Figure 3.8 Relationship between propeller spin direction and direction of thrust and torque vector.	3-28
Figure 3.9 Non-dimensional coefficients K_T , K_{Q_e} and η_0 as a function of positive advance ratio J_0 .	3-33
Figure 3.10 Propeller thrust (force) T as a function of propeller revolution n and ambient water velocity u_a (bilinear thruster model).	3-34
Figure 3.11 Propeller thrust (force) T as a function of propeller revolution n (affine thruster model).	3-35
Figure 3.12 Propeller thrust (force) T as a function of new control variable u (affine thruster model).	3-36
Figure 3.13 Relationship between control variable u and auxiliary control variable u' .	3-37
Figure 3.14 Propeller thrust (force) T as odd function of auxiliary control variable u' .	3-38
Figure 3.15 Thrust and torque as vector variables.	3-39

Figure 3.16 Block diagram showing full thruster model, including thruster control loop dynamics.	3-41
Figure 3.17 Linear dynamic model of the speed-controlled DC motor.	3-41
Figure 4.1 The overall control system architecture (Härkegård, 2003).	4-3
Figure 4.2 Modular flight control structure (Beck, 2002).	4-4
Figure 4.3 Typical open-loop ROV control structure.	4-5
Figure 4.4 Typical closed-loop AUV control structure.	4-6
Figure 4.5 Relationship between FDAS and typical control structure for open-frame underwater vehicle.	4-6
Figure 4.6 Constrained (admissible) control subset Ω .	4-12
Figure 4.7 Attainable command set Φ .	4-14
Figure 4.8 Position of the desired virtual control input \mathbf{v}_d in Φ .	4-19
Figure 4.9 Family of spheres for l_1 norm and the solution segment P_1P_2 .	4-22
Figure 4.10 Family of spheres for l_2 norm and the solution segment P_1P_2 for case $\mathbf{W}_u = \mathbf{I}_3$.	4-22
Figure 4.11 Family of spheres for l_2 norm and the solution segment P_1P_2 for case $\mathbf{W}_u = \text{diag}(1,1,2)$.	4-23
Figure 4.12 The solution obtained by the fixed-point method.	4-25
Figure 4.13 Partition of the virtual control space $\Phi_v: \Phi_p$ (feasible region for pseudoinverse) and Φ (attainable command set).	4-31
Figure 4.14 Images of partitions in the true control space: $\Omega_v = \mathbf{B}^+(\Phi_v)$ (image of Φ_v), $\Omega_p = \mathbf{B}^+(\Phi_p)$ (image of Φ_p) and $\Omega_e = \mathbf{B}^+(\Phi)$ (image of Φ).	4-31
Figure 4.15 Three typical cases for position of virtual control inputs relative to Φ_p and Φ .	4-34
Figure 4.16 Case $S_1 \in \Phi_p$ yields to the pseudoinverse solution $T_1 \in \Omega_p \subset \Omega$ that is optimal in l_2 sense.	4-36
Figure 4.17 Case $S_2 \in \Phi \setminus \Phi_p$ yields to the unfeasible pseudoinverse solution $T_2 \in \Omega_e \setminus \Omega_p$ that lies outside Ω .	4-36
Figure 4.18 Case $S_3 \in \Phi_v \setminus \Phi$ yields to the unfeasible pseudoinverse solution $T_3 \in \Omega_v \setminus \Omega_e$ that lies outside Ω .	4-37
Figure 4.19 Approximation error $\mathbf{e} = \mathbf{v} - \mathbf{v}^* = \mathbf{v} - \mathbf{B}\mathbf{u}^*$.	4-38
Figure 4.20 Approximation of unfeasible pseudoinverse solutions in the true control space.	4-41
Figure 4.21 Results of approximations in the virtual control space.	4-41
Figure 4.22 Improving of T - and S -approximation by the fixed-point method.	4-45
Figure 4.23 The direct control allocation method searches for $\mathbf{v}_d^* = a\mathbf{v}_d$ on $\partial(\Phi)$.	4-48
Figure 4.24 The direct control allocation method finds the limit solution \mathbf{u}^* on $\partial(\Omega)$, such that $\mathbf{B}\mathbf{u}^* = \mathbf{v}_d^*$, and performs inverse scaling to obtain the solution $\mathbf{u} = \frac{1}{a}\mathbf{u}^*$.	4-48
Figure 4.25 Daisy chain control allocation for $M = 3$.	4-52

Figure 4.26 Daisy chain solution for $M = 3$ groups of actuators.	4-54
Figure 5.1 Open-loop ROV control structure.	5-3
Figure 5.2 Partitions of the normalised virtual control space for motion in the horizontal plane.	5-5
Figure 5.3 Thruster model used in control allocation for underwater vehicles.	5-11
Figure 5.4 Relationship between thruster configuration and controllable DOF.	5-16
Figure 5.5 Three cases for finding the maximum modules of force and moment vectors (X-shaped thruster configuration).	5-17
Figure 5.6 Three cases for finding the maximum modules of force and moment vectors (cross-shaped thruster configuration).	5-19
Figure 5.7 Attainable command set $\underline{\Phi}^{HT}$ for the X-shaped thruster configuration.	5-25
Figure 5.8 Attainable command set $\underline{\Phi}^{HT}$ for the cross-shaped thruster configuration.	5-25
Figure 5.9 Overall functional architecture of the proposed FDAS.	5-32
Figure 5.10 Block diagram showing connections between FDU and TCU for thruster Th .	5-37
Figure 5.11 The first stage in off-line training: test trials for acquisition of training data.	5-39
Figure 5.12 Raw training data in the 3D space.	5-39
Figure 5.13 Time diagrams of raw training data.	5-40
Figure 5.14 Pre-processed training data in the 3D space.	5-41
Figure 5.15 Time diagrams of pre-processed training data. One of the zero-velocity segments is highlighted for illustration. These segments are excluded from the training process.	5-41
Figure 5.16 Time diagrams of pre-processed training data, together with fault code.	5-43
Figure 5.17 SOM trajectories were initially used for fault detection purpose.	5-44
Figure 5.18 The second stage in off-line training phase: different fault types (patterns) are replaced by SOM prototypes.	5-45
Figure 5.19 On-line fault detection phase: position of the feature vector is determined relative to SOM prototypes by finding $q = 3$ closest BMUs in each SOM.	5-48
Figure 5.20 Buffer $\mathbf{B}_{s \times q}$ with present and past values of vector \mathbf{b} .	5-50
Figure 5.21 Partitions of the virtual control space $\underline{\Phi}_v^{HT}$.	5-54
Figure 5.22 Feasible region $\underline{\Phi}_p^{HT}$ for different thruster configurations.	5-60
Figure 5.23 Partitions of the virtual control space $\underline{\Phi}_v^{HT}$ for faulty state "Heavy jammed propeller" in 2HT ($s_2^{HT} = 0.5$).	5-61
Figure 5.24 Partitions of the virtual control space $\underline{\Phi}_v^{HT}$ for faulty state "Broken propeller" in 2HT ($s_2^{HT} = 0$).	5-63
Figure 6.1 Simulink model for evaluation of the FDU.	6-3
Figure 6.2 Evaluation is performed by comparing the actual fault indicator and FDU output.	6-3
Figure 6.3 Virtual underwater world.	6-6
Figure 6.4 (A1) Feasible pseudoinverse solution $(\underline{\tau}_d^{HT} \in \underline{\Phi}_p^{HT} \Rightarrow \underline{\mathbf{u}}^{HT} \in \underline{\Omega}^{HT})$.	6-31

Figure 6.5 (A2) Unfeasible pseudoinverse solution $(\underline{\tau}_d^{HT} \notin \underline{\Phi}_p^{HT} \Rightarrow \underline{u}^{HT} \notin \underline{\Omega}^{HT})$.	6-32
Figure 6.6 (A3) Approximation of an unattainable command input $(\underline{\tau}_d^{HT} \notin \underline{\Phi}^{HT})$.	6-33
Figure 6.7 (A4) Feasible trajectory – FALCON $(\underline{\tau}_d^{HT}(t) \subset \underline{\Phi}_p^{HT})$.	6-34
Figure 6.8 (A4) Feasible trajectory – URIS $(\underline{\tau}_d^{HT}(t) \subset \underline{\Phi}_p^{HT})$.	6-35
Figure 6.9 (A5) Feasible trajectory – FALCON $(\underline{\tau}_d^{HT}(t) \subset \underline{\Phi}^{HT})$.	6-36
Figure 6.10 (A5) Feasible trajectory – URIS $(\underline{\tau}_d^{HT}(t) \subset \underline{\Phi}^{HT})$.	6-37
Figure 6.11 (A6) Partially unfeasible trajectory – FALCON (parts of $\underline{\tau}_d^{HT}(t)$ lie outside $\underline{\Phi}^{HT}$).	6-38
Figure 6.12 (A6) Partially unfeasible trajectory – URIS (parts of $\underline{\tau}_d^{HT}(t)$ lie outside $\underline{\Phi}^{HT}$).	6-39
Figure 6.13 (A7) Difference between affine and bilinear thruster model (FALCON).	6-40
Figure 6.14 (A7) Difference between affine and bilinear thruster model (URIS).	6-41
Figure 6.15 (A8) Choice of propeller spin direction (FALCON).	6-42
Figure 6.16 (A8) Choice of propeller spin direction (URIS).	6-43
Figure 6.17 (A9) Motion in vertical plane (FALCON).	6-44
Figure 6.18 (A10) Difference between symmetrical and non-symmetrical T -curve (FALCON).	6-45
Figure 6.19 (A10) Difference between symmetrical and non-symmetrical T -curve (URIS).	6-46
Figure 6.20 (B1) Partial fault - "Jammed propeller" (FALCON).	6-47
Figure 6.21 (B1) Partial fault - "Jammed propeller" (URIS).	6-48
Figure 6.22 (B2) Partial fault - "Heavy jammed propeller" (FALCON).	6-49
Figure 6.23 (B2) Partial fault - "Heavy jammed propeller" (URIS).	6-50
Figure 6.24 (B3) Partial fault - "Unknown state" (FALCON).	6-51
Figure 6.25 (B3) Partial fault - "Unknown state" (URIS).	6-52
Figure 6.26 (B4) Total fault - "Broken propeller" (FALCON).	6-53
Figure 6.27 (B4) Total fault - "Broken propeller" (URIS).	6-54
Figure 6.28 (B5) Consecutive faults – passing through the pipe (FALCON).	6-55
Figure 6.29 (B5) Consecutive faults – passing through the pipe (URIS).	6-56
Figure 6.30 (B6) Consecutive faults – passing through the hole in the rock (FALCON).	6-57
Figure 6.31 (B6) Consecutive faults – passing through the hole in the rock (URIS).	6-58
Figure 6.32 Experiment set-up.	6-60
Figure 6.33 Tank trials with FALCON – Day one (the first experiment).	6-62
Figure 6.34 Tank trials with FALCON – Day one (the second experiment).	6-62
Figure 6.35 Inertial Measurement Unit (IMU).	6-63
Figure 6.36 FALCON and COUGAR in the tank.	6-64
Figure 6.37 Test FM1 (free motion in 4 DOF, fault-free state in all thrusters, FDAS active).	6-66
Figure 6.38 Test FM2 (free motion in 4 DOF, total fault in 2HT (disabled), FDAS not active). Segments with $n_d^{HT_2} = 0$ and $n^{HT_2} \neq 0$ are highlighted.	6-66

Figure 6.39 Test FM3 (free motion in 4 DOF, total fault in 2HT (disabled), FDAS active). Segments with $n_d^{HT_1} = 0$ and $n^{HT_2} \neq 0$ are highlighted.	6-67
Figure 6.40 Test FM4 (free motion in 4 DOF, partial fault in 2HT ($s_2^{HT} = 0.50$), FDAS active).	6-67
Figure 6.41 Test FM5 (free motion in 4 DOF, partial fault in 2HT ($s_2^{HT} = 0.20$), FDAS active).	6-67
Figure 6.42 Test FM6 (free motion in 4 DOF, partial fault in 2HT ($s_2^{HT} = 0.80$), FDAS active).	6-68
Figure 6.43 Test PF1 (path following, fault-free state in all thrusters, FDAS active).	6-70
Figure 6.44 Test PF2 (path following, total fault in 2HT (disabled), FDAS not active).	6-71
Figure 6.45 Test PF3 (path following, total fault in 2HT (disabled), FDAS active).	6-72
Figure 6.46 Test PF4 (path following, partial fault in 2HT ($s_2^{HT} = 0.50$), FDAS active).	6-73
Figure 6.47 Test PF5 (path following, partial fault in 2HT ($s_2^{HT} = 0.20$), FDAS active).	6-74
Figure C.1 The intersection of two non-parallel planes.	C-2
Figure C.2 The intersection of three planes, where no two of them are parallel.	C-3
Figure D.1 Screenshot showing a possible arrangement of windows and displays.	D-2
Figure D.2 The first version of the ROV simulator.	D-3
Figure D.3 Block diagram showing the second version of the ROV simulator.	D-4
Figure D.4 Block diagram showing the third version of the ROV simulator.	D-5
Figure D.5 Block diagram showing the fourth version of the ROV simulator.	D-6
Figure D.6 Implementation of the HCU.	D-7
Figure D.7 Block parameters dialog box "Signal Conditioning".	D-9
Figure D.8 Selection of the command input source.	D-10
Figure D.9 Implementation of the FDS.	D-12
Figure D.10 Implementation of the FAS.	D-13
Figure D.11 Implementation of the propulsion system.	D-14
Figure D.12 Implementation of the ROV model.	D-16
Figure D.13 Displays "Fault states" and " τ_d & τ ".	D-17
Figure D.14 Display Panel.	D-18
Figure D.15 Display "Velocity & Heading".	D-19
Figure D.16 Display "Feasible Region".	D-20
Figure D.17 Display "Horizontal Thrusters".	D-20
Figure D.18 Connection between Simulink model and virtual underwater world.	D-23
Figure D.19 Block parameters dialog box "Underwater World".	D-23
Figure D.20 Virtual Reality Display.	D-25
Figure D.21 Coordinate frames $\{E\}$ and $\{VR\}$ in virtual underwater world.	D-26
Figure D.22 Block parameters dialog box "Propeller's rotation".	D-27
Figure D.23 Viewpoint <i>Camera Side View</i> .	D-28
Figure D.24 Hat switch is used to choose which point V_i is active.	D-29
Figure D.25 Block parameters dialog box "Camera Side View".	D-29

- Figure D.26** Different perspectives obtained from the viewpoint V_1 by varying r and FoV . D-30
- Figure D.27** Scene seen from different viewpoints V_i , $i = \overline{1,8}$ with fixed parameters $r = 3m$, $\varphi = 10^\circ$ and $FoV = 45^\circ$. D-30
- Figure D.28** Viewpoint *Camera Front View*. D-31
- Figure D.29** Relative position between objects in the virtual underwater world. D-32
- Figure D.30** Initial positions of the vehicle, close to the pipe and the rock with the hole. D-33
- Figure D.31** Block parameters dialog box "Testing objects". D-33
- Figure D.32** Change of pipe's orientation: $Rotation = [0 \ 0 \ 1 \ -80^\circ]$. D-33
- Figure D.33** Block parameters dialog box "Lights". D-34

List of Tables

Table 2.1 Example of a pattern table.	2-37
Table 2.2 Damage types.	2-38
Table 3.1 Restoring forces in the Earth-fixed frame.	3-20
Table 3.2 Restoring forces and moments in the body-fixed frame for the attitude representation with Euler angles (see section 3.3.1).	3-21
Table 3.3 Restoring forces and moments in the body-fixed frame for the attitude representation with Euler parameters (see section 3.3.2).	3-21
Table 3.4 Restoring forces and moments in the body-fixed frame for the attitude representation with Modified Rodrigues parameters (see section 3.3.3).	3-21
Table 3.5 Position vectors for different thruster configurations.	3-28
Table 3.6 Orientation vectors for different thruster configurations.	3-28
Table 4.1 Iterations of the fixed-point method.	4-26
Table 4.2 Vertices of Φ_p .	4-33
Table 4.3 Vertices of $\Omega_p = \mathbf{B}^+(\Phi_p)$.	4-33
Table 4.4 Cross-relation between the virtual and the true control space.	4-33
Table 4.5 Iterations of the fixed-point method for $\mathbf{v}_2 \in \Phi \setminus \Phi_p$.	4-44
Table 4.6 Iterations of the fixed-point method for $\mathbf{v}_3 \in \Phi_v \setminus \Phi$.	4-44
Table 5.1 Thruster control matrix for different thruster configurations.	5-14
Table 5.2 Decomposition of the the FALCON motion.	5-15
Table 5.3 Decomposition of the URIS motion.	5-15
Table 5.4 Coordinates of vertices of $\underline{\Omega}^{HT}$ and $\underline{\Phi}^{HT}$.	5-24
Table 5.5 Nomenclature for $\underline{\Omega}^{HT}$.	5-26
Table 5.6 Nomenclature for $\underline{\Phi}^{HT}$ (X-shaped thruster configuration).	5-26
Table 5.7 Nomenclature for $\underline{\Phi}^{HT}$ (cross-shaped thruster configuration).	5-26
Table 5.8 Fault code table.	5-36
Table 5.9 "Co-ordinator" actions for fault-free case.	5-56
Table 5.10 "Co-ordinator" actions for a partial fault in 1HT .	5-56
Table 5.11 "Co-ordinator" actions for a total fault in 1HT .	5-56
Table 5.12 "Co-ordinator" actions for a partial fault in 1VT .	5-57
Table 5.13 "Co-ordinator" actions for a total fault in 1VT .	5-57
Table 5.14 Normal vectors for different thruster configurations.	5-59
Table 5.15 Facets of $\underline{\Phi}_p^{HT}$ and colour codes.	5-59
Table 6.1 Navigation table for test cases in the group A (fault-free cases).	6-4

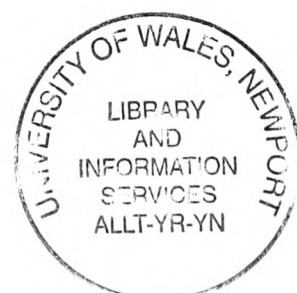
Table 6.2 Navigation table for test cases in the group B (faulty situations).	6-5
Table 6.3 Abbreviated symbols used in diagrams in this chapter.	6-5
Table 6.4 Simulation settings for group A tests.	6-7
Table 6.5 Hybrid approach for unfeasible pseudoinverse solution.	6-9
Table 6.6 Hybrid approach for unattainable command input.	6-11
Table 6.7 Simulation settings for test cases in group B.	6-23
Table 6.8 Description of tests in the first experiment (Day two).	6-64
Table 6.9 Description of tests in the second experiment (Day two).	6-68
Table D.1 HCU controls.	D-8

List of Examples

Example 3.1 (Propeller spin direction)	3-27
Example 4.1 (Linear control allocation problem)	4-10
Example 4.2 (Choice of norm)	4-19
Example 4.3 (Choice of the weighting matrix \mathbf{W}_u)	4-21
Example 4.4 (Fixed-point method)	4-25
Example 4.5 (Redistributed pseudoinverse method)	4-27
Example 4.6 (Pseudoinverse method)	4-28
Example 4.7 (Pseudoinverse – partitioning of the virtual control space)	4-29
Example 4.8 (Pseudoinverse – geometric interpretation of solution)	4-33
Example 4.9 (Pseudoinverse – approximation of unfeasible solutions)	4-38
Example 4.10 (Hybrid approach for control allocation)	4-44
Example 4.11 (Direct control allocation)	4-47
Example 4.12 (Daisy chain control allocation)	4-52

List of Algorithms

Algorithm 5.1 (FDU - On-line fault detection)	5-46
Algorithm 5.2 (FAS – Hybrid approach for control allocation)	5-65



Chapter 1: Introduction

"I begin my story for nothing, without benefit for myself or anyone else, from a need stronger than benefit or reason. I must leave a record of myself, the chronicled anguish of my inner conversations, in the vague hope that a solution will be found when all accounts have been settled (if they may ever be), when I have left my trail of ink on this paper, which lies in front of me like a challenge."

Meša Selimović, *Death and the Dervish*

1.1 Background

This thesis is a partial contribution to the *IMproving the Performance of Remotely Operated VEHicleS* (IMPROVES) project. The IMPROVES project is an *Engineering Physical Science Research Council* (EPSRC) funded collaboration between three UK universities and Seaeye Marine Ltd. of Fareham, UK. The Universities involved are: University of Wales College, Newport, University of Southampton and University of Plymouth. IMPROVES intends to improve the dynamic performance of advanced, multi-mission remotely operated vehicles (ROVs) used for submarine tasks by the offshore industry. Improvements are made through the design and development of a new and robust predictive control system, enhanced with the on-line fault diagnosis and accommodation features. Performance is currently limited by the harsh nature of the environment, the dynamic properties of the vehicles and the delay introduced by the distance between the vehicle and operator. The part of the IMPROVES project undertaken by the research fellowships in Newport addresses the following issues:

- modelling the underwater vehicle,
- design of the fault diagnosis and accommodation system.
- development of the thruster test rig.

This thesis is focused on the design of thruster fault diagnosis and accommodation system for underwater vehicles. In order to avoid any misunderstanding with the terminology, it is necessary to give some basic definitions. In the *Encyclopaedia Britannica*, submerged vehicles are defined as following:

Submarine: "*any naval vessel that is capable of propelling itself beneath the water as well as on the water's surface. This is a unique capability among warships, and submarines are quite different in design and appearance from surface ships.*"

Underwater Vehicle: "small vehicle that is capable of propelling itself beneath the water surface as well as on the water's surface. This includes unmanned underwater vehicles (UUV), remotely operated vehicles (ROV), autonomous underwater vehicles (AUV) and underwater robotic vehicles (URV). Underwater vehicles are used both commercially and by the navy. "

Hence, submarines are clearly distinguished from underwater vehicles. The same separation is kept throughout this thesis: when the term underwater vehicle(s) is used, it excludes submarine(s).

A large number of open-frame underwater vehicles have no other actuators except for thrusters. This thesis introduces a new approach associated with thruster fault diagnosis and accommodation for this class of underwater vehicles. Torpedo-shaped vehicles are not covered in this thesis, since they use control surfaces as well as thrusters, but the same fault diagnosis and accommodation concept can be extended to cover this class of underwater vehicles by including new actuators into control architecture and reformulating the control allocation problem.

Underwater vehicles are liable to faults or failures during underwater missions. Thrusters are one of the most common and most important sources of faults. In all but the most trivial cases the existence of a fault may lead to cancelling the mission. The implication of small faults can be very expensive and time consuming. Although good design practice tries to minimize the occurrence of faults and failures, there is a certain probability that faults will occur. Recognition that such events do occur enables system designers to develop strategies by which the effect they exert is minimised. A large number of open-frame underwater vehicles represent overactuated control systems, i.e. they have four or more horizontal thrusters for the motion in the horizontal plane in three DOF (surge, sway and yaw). This thesis demonstrates that, for this class of vehicle, in the case of a partial or

total fault in a horizontal thruster, it is possible to reconfigure the control system in an optimal manner, in order to maintain a high level of manoeuvrability of the faulty vehicle and complete the mission.

Two ROVs (FALCON, SeaEye Marine Ltd. and URIS, University of Girona) with different thruster configurations are used throughout the thesis to test the algorithms presented in Chapter 5. Technical specifications of these two vehicles are given in Appendix A.

1.2 Motivation

Underwater vehicles are used for commercial purposes (marine off-shore inspections, surveying, repairs, etc.) or by research groups to investigate navigation guidance and new control techniques. Figure 1.1 displays pictures captured by ROV on-board camera during typical underwater mission. Close proximity of pipes and underwater structures demonstrate the harsh nature of the environment in which underwater vehicles operate. Despite the preventive measures undertaken by manufacturers to protect the on-board equipment, components like actuators, sensors, etc. are liable to faults, mainly due to inhospitable operational environment.



Figure 1.1 Pictures captured by ROV on-board cameras showing the harsh nature of the operational environment. Courtesy of Seaeye Marine Ltd.

Conversations with researchers from other universities and representatives of Seaeye Marine Ltd. confirmed that they *all* experienced some kind of thruster fault during

operations. In a few cases the propeller was jammed by seaweed, kelp or rope. In other cases the water penetrated inside the thruster control electronics. In the last case, thruster fault occurred during the operation with FALCON (spring 2003, South Africa), when a defect in manufacturing of the gearbox produced a failure that damaged the Hall sensors. In most cases the mission was aborted and the vehicle was recovered for repair. In many cases spare parts were not available on the mothership, leading to a total abortion of the mission. In some cases the ROV pilot tried (unsuccessfully) to navigate the faulty vehicle and continue the mission.

The main motivation for this thesis is to overcome these problems by designing a system that is able to detect and isolate thruster faults, automatically redistribute the control energy among operable thrusters in the case of a fault in a thruster and inform the ROV pilot or the main controller about changes and their effects. In such a way, the risk of more serious damage is minimised and the framework for mission continuation and completion is provided, with a minimal loss of control performance.

1.3 Aims and objectives

The aim of the research is development of thruster fault diagnosis and accommodation system for overactuated, open-frame underwater vehicles, which fulfil the following requirements:

- In fault-free case, optimal control allocation must be guaranteed for all possible command inputs, which minimises a control energy cost function, the most suitable criteria for underwater applications. Minimising control energy means maximising operational battery life, which is very important issue for future development of autonomous underwater vehicles.
- In faulty situations, any malfunction of a thruster must be immediately recognised and remedial actions must be performed to isolate the fault and to prevent further

damage. At the same time, the control allocation must be automatically updated to accommodate the fault and to provide a framework for continuation of the mission, with a minimal loss of control performance.

The specific objectives of the thesis are:

- Explore the existing methods for fault diagnosis and accommodation in dynamic systems.
- Identify which of these methods are applicable to meet requirements and specific implementation issues, defined in the IMPROVES project.
- Develop module for detection of thruster faults.
- Formulate and solve the control allocation problem for fault-free case.
- Extend the algorithm to cover faulty situations.
- Develop the simulation model to test the algorithm.
- Verify the performance of the algorithm with real-world applications.

1.4 Overview of chapter contents

Chapter 2, "Literature Review", provides an overview of traditional and modern approaches to fault diagnosis and accommodation of dynamic systems. Different approaches for fault diagnosis and accommodation are presented in a systematic way. Due to similarity between the control allocation problem for underwater vehicle and aircraft, this chapter includes recent advances in the field of control allocation for aircraft. Previous work on fault diagnosis and accommodation for underwater vehicles is discussed in more detail at the end of the chapter.

Chapter 3, "Models for ROV & Propulsion System", provides background into modelling and simulation of ROVs. The development of a non-linear ROV dynamic model in 6 DOF is described, including three different attitude representations. This chapter ends with the discussion of different models for thrusters and thruster control units.

Chapter 4, "Control Allocation", is devoted to gaining insight into the geometry of the general control allocation problem. The general formulation of the control allocation problem is used to establish the criteria for separation of the system control architecture into two independent tasks (control law and control allocation), thereby allowing the control allocation to be considered separately from the control law. Existing methods for solution are presented and their performance is compared using the same example, where the control allocation problem is formulated for the two-dimensional virtual control space and the three-dimensional true control space, enabling easy visualisation and geometric interpretation. The hybrid approach for control allocation, based on the integration of the pseudoinverse and the fixed-point iteration method, is gradually introduced, providing an easy extension of the concept to higher-dimensional cases.

Chapter 5, "Fault Diagnosis and Accommodation System", proposes thruster fault diagnosis and accommodation system (FDAS) for overactuated, open-frame underwater vehicles. The hybrid approach for control allocation, introduced in Chapter 4, is extended for the case of the three-dimensional virtual control space and the four-dimensional true control space and used as a foundation to build an enhanced control allocator, with fault detection and accommodation capabilities. The feasible region concept is developed in order to visualise thruster velocity saturation bounds. Implementation issues are discussed at the end of the chapter.

Chapter 6, "Testing and Evaluation of the FDAS", evaluates the performance of the FDAS and highlights its key features using simulation and real-world application. In essence, this chapter effectively combines the material presented in previous chapters into a collection of representative examples, in order to examine the behaviour of the FDAS in fault-free and faulty conditions. The FDAS was used in a real-world application, where the performance of the FALCON ROV was examined in different, artificially generated

fault conditions in the thrusters. Preliminary results from these tests are presented at the end of the chapter.

Chapter 7, "Conclusions and Further Work", reviews the thesis, lists and describes the contributions and makes suggestions for further work.

The approach adopted in most chapters is to present "overview-type" material in the main body and to cover more detailed explanations and analysis of particular aspects through the use of appropriate appendices, which are included at the end of the thesis. Description of individual appendices is given in the following.

Appendix A contains technical details and specifications for FALCON and URIS.

Appendix B provides some results from optimal control theory.

Appendix C presents some results from 3D geometry.

Appendix D describes the ROV simulator.

Appendix E contains a list of published and submitted papers produced during the course of the work described in the thesis and a list of awards received in international conferences. Copies of published papers are also included.

1.5 List of main contributions

The main contributions of the work presented in this thesis are summarised as follows:

- Development of the on-line fault detector units, able to detect external and internal thruster faults.
- Development of the hybrid approach for control allocation, able to allocate optimal solutions of the control allocation problem in fault-free and faulty situations.
- Visualisation of thruster velocity saturation bounds using the feasible region concept.
- Formulation of the control problem in normalised form.

- Development of a method to compensate for non-symmetrical propeller T -curve.
- Design of a simulation model with virtual reality display.

These contributions are discussed throughout the thesis and summarised in section 7.3.

Chapter 2: Literature Review

“With a shriek birds flee across the black sky, people are silent, my blood aches from waiting...”

Meša Selimović

2.1 *Introduction*

This chapter provides an overview of classic and modern approaches to fault diagnosis and accommodation of dynamic systems. The purpose is to present a subset of different approaches for fault diagnosis in a systematic way, such that the reader could obtain a global picture about a palette of possible methods. The problem of fault accommodation for underwater vehicles is closely related to the control allocation problem for aircrafts. The latter problem has gained much wider interest in the research community than the former. Because both problems can be solved using the same techniques, outlines of the main methods and recent advances in the field of control allocation for aircraft are given in this chapter, while a full mathematical description of these methods, with numerical examples, can be found in Chapter 4. Previous work on fault diagnosis and accommodation for underwater vehicles is presented at the end of the chapter.

The chapter is organised as follows: section 2.2 introduces a basic concepts of fault diagnosis. Section 2.3 describes fault diagnosis terminology, and fault diagnosis methodology is described in section 2.4. General classification of the fault diagnosis methods is given in section 2.5. A short overview of recent fault diagnosis approaches is given in section 2.6. Section 2.7 addresses the usage of fuzzy logic in fault diagnosis. Pattern recognition methods used in fault diagnosis are presented in section 2.8. Control allocation techniques for aircraft are described in section 2.9. Special attention is devoted to fault diagnosis approaches applied for underwater vehicles, which are described in section 2.10. The concluding remarks are presented in section 2.11 and a list of references, used for the compilation of the chapter and cited in the text, is provided in section 2.12.

2.2 Basic concepts of fault-tolerant control

Faults in dynamic systems can cause undesired reactions and behaviour of a plant, and the consequences could be damage to technical parts of the plant, to personnel and/or the environment (Blanke, *et al.*, 2000b). One way to cope with faulty situations is to use *Fault-Tolerant Control* (FTC). Blanke (2001) defines FTC as a set of techniques developed to handle faults autonomously, prevent that simple faults develop into serious failure, increase plant availability and reduce the risk of safety hazards. The FTC combines fault detection, isolation and identification (*fault diagnosis*) with control methods (*fault accommodation*) to handle faults in an intelligent way. The first stage in this process is fault diagnosis. The early detection of the occurrence of a fault and its isolation and identification is critical in avoiding product deterioration, performance degradation, major damage to the machinery itself and damage to human health or even loss of lives (Gertler, 1998). Fault diagnosis is followed by fault accommodation, which includes automatic condition assessment and calculation of appropriate remedial actions to avoid certain consequences of a fault. Fault diagnosis and accommodation techniques have been the subject of research over the last two decades, and this field has gained wide interest in the research community (Patton and Chen, 1999; Gertler, 1998; Isermann, 1997; Köppen-Seliger and Frank, 1996; Pouliezios and Stavrakakis, 1989).

2.3 Fault diagnosis terminology

The terminology in the literature of the field of fault diagnosis is not consistent. This makes it difficult to understand the goals of the particular contributions and to compare the different approaches. The terminology used in this thesis is consistent with SAFEPROCESS terminology, established by the IFAC Technical Committee: SAFEPROCESS (Blanke, *et al.*, 2000b; Patton and Chen, 1999).

States and signals

- **Constraint:** A functional relation between variables and parameters of a system. Constraints may be specified in different forms, including linear and non-linear differential equations, and tabular relations with logic conditions between variables.
- **Fault:** An unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable (usual, standard) condition.
- **Failure:** A permanent interruption of a system's ability to perform a required function under specified operating conditions. The term "failure" suggests complete breakdown of a system component or function, whilst the term "fault" may be used to indicate that a malfunction may be tolerable at its present stage.
- **Residual:** A fault indicator, based on a deviation between measurements and model-equation-based computations.
- **Symptom:** A change of an observable quantity from normal behaviour.

Systems

- **Controlled system:** A physical plant under consideration with sensors and actuators used for control.
- **Fail-operational system:** A system, which is able to operate with no change in objectives or performance, despite of any single failure.
- **Fail-safe system:** A system, which fails to a state that is considered safe in the particular context.

Functions

- **Fault accommodation:** Change in controller parameters or structure to avoid the consequences of a fault. The input-output between controller and plant is

unchanged. The original control objective is achieved although performance may degrade.

- **Fault detection:** Determination of the faults present in a system and the time of detection. Produces a binary decision – either that something has gone wrong or that everything is fine.
- **Fault isolation:** Determination of the kind, location and time of detection of a fault. Follows fault detection.
- **Fault identification:** Determination of the size and time-variant behaviour of a fault. Follows fault isolation.
- **Fault diagnosis:** Determination of the kind, size, location and time of detection of a fault. Includes fault detection, isolation and identification.
- **Fault tolerance:** The ability of a controlled system to maintain control objectives, despite the occurrence of a fault. A degradation of control performance may be accepted. Fault-tolerance can be obtained through fault accommodation or through system and /or controller reconfiguration.
- **Reconfiguration:** Change in input-output between the controller and plant through change of controller structure and parameters. The original control objective is achieved although performance may degrade.
- **Supervision:** The ability to monitor whether control objectives are met. If not, obtain/calculate a revised control objective and a new control structure and parameters that make a faulty closed-loop system meet the new modified objective. Supervision should take effect if faults occur and it is not possible to meet the original control objective within the fault-tolerant scheme.

2.4 *Fault diagnosis methodology*

As mentioned in section 2.2, a wide range of fault diagnosis approaches have been proposed in the literature. Patton, *et al.* (2000a) suggest that these approaches can be divided into model-based techniques, knowledge-based methods and signal processing techniques.

There are two main classes of model-based approaches. In the first class, quantitative models are used, such as differential equations, transfer functions, state-space methods, etc. These methods are based upon parameter estimation, state estimation or parity space concepts. The core of the approach is that a fault will cause changes to certain physical parameters and measurements, which will lead to change in some model parameters or states. Fault detection and isolation is then possible by monitoring the estimated parameters or states. In order to apply this approach, it is essential to have *a priori* knowledge about the relationships between the system, faults and model parameters/states. This is not easy task, since comprehensive theoretical models for complex systems (e.g., chemical processes) are difficult to obtain and in some situations impossible to derive. *Artificial Intelligence* (AI) methods are used in the second class of the model-based techniques. Some methods use qualitative reasoning and qualitative modelling. Essentially, qualitative models of the process are used to predict the behaviour of the process under normal operating conditions and also during various faulty conditions. Fault detection is then performed by comparing the predicted behaviour with the actual observations. Other methods within the AI domain, applicable to dynamic systems, use neural networks, fuzzy decision-making and neuro-fuzzy methods. These methods are attractive, since explicit mathematical model of the monitored plant is not required to be known in advance. Implicit models of the plant ("data-based models") are provided by applying soft-computing techniques (the neural network training and fuzzy

rule design) to raw plant data. The relationship between faults and their causes can be identified and stored as network weights during training phase of the neural network. After training, the network can be used to diagnose faults by associating the observed malfunctions with the corresponding fault. Fuzzy-logic methods, which belong to AI rule-based approaches, extract diagnostic rules from process structure and unit functions (Patton, *et al.*, 2000a).

Knowledge about the process structure, functions of the process units and their qualitative models under various faulty conditions are required for knowledge-based methods. A disadvantage of this approach is that the development of a knowledge-based diagnostic system demands considerable time and effort to be really effective. To reduce the development time, a large amount of research effort has been dedicated to integrate knowledge-based and neural networks-based approaches.

Signal processing methods belong to model-free methods, i.e. methods that do not require a process model. Different tests on the statistical properties of signals are applied in order to detect faulty situations. In practical applications, process control charts are used for monitoring the statistical state of a process.

2.5 Classification of fault diagnosis methods

The methods of fault diagnosis may be classified into two major groups, as shown in Figure 2.1 (Patton and Chen, 1999; Gertler, 1998; Gertler, 1997; Frank 1990):

- those which do not utilise the mathematical model of the plant (*Model-Free Methods*),
- those which utilise the mathematical model (*Model-Based Methods*).

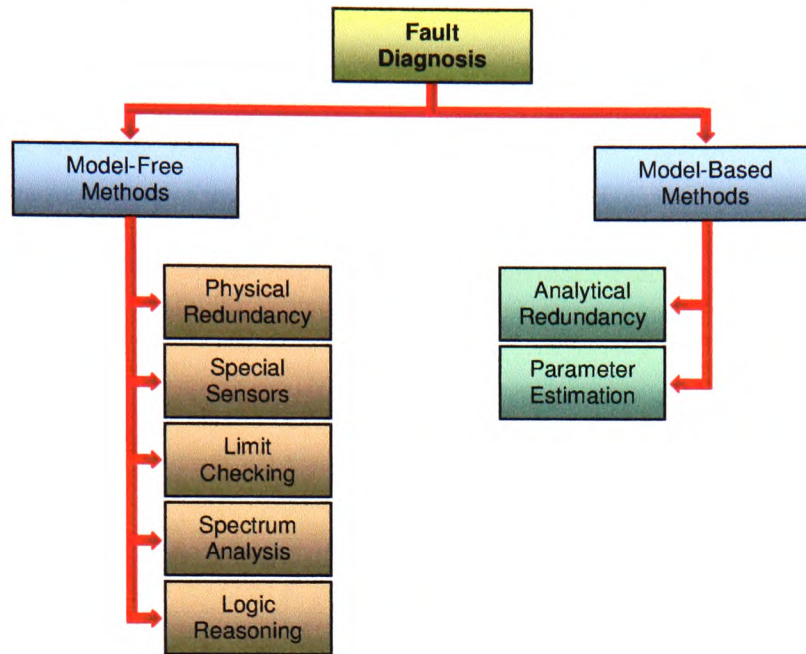


Figure 2.1 Classification of fault diagnosis methods.

2.5.1 Model-free methods

Model-free methods range from physical redundancy and special sensors through limit-checking and spectrum analysis to logical reasoning (Gertler, 1998).

Physical redundancy. In this approach, the same physical quantity is measured by multiple sensors. Any serious discrepancy between the measurements indicates a sensor fault. With only two parallel sensors, fault isolation is not possible. With three sensors, a voting scheme can be utilised, which isolates the faulty sensor. Disadvantages of the approach are that physical redundancy involves extra hardware cost and extra weight, the latter representing a serious concern in aerospace and underwater applications.

Special sensors. Special sensors can be installed explicitly for fault detection and diagnosis. These may be limit sensors (measuring e.g., temperature or pressure), which

perform limit checking (see below) in hardware. Other special sensors may measure some fault-indicating physical quantity, such as sound, vibration, elongation, etc.

Limit checking. This approach is widely used in practice. The first step consists in establishing thresholds for the plant variables. In the next step, plant measurements are compared to preset thresholds. Exceeding the threshold indicates a fault situation. In many systems, there are two levels of alarms; the first serves for a pre-warning purpose, while the second is for triggering an emergency reaction. The limit checking approach is simple and straightforward, but it suffers from two serious drawbacks:

- Since the plant variables may vary widely due to normal input variations, the test thresholds need to be set quite conservatively. Also, noisy data or change of operating point may trigger false alarms.
- The effect of a single component fault may propagate to many plant variables and cause many system signals to exceed their limits and appear as multiple faults, making isolation extremely difficult.

Spectrum analysis. Another method that is applicable for fault detection and isolation is spectrum analysis of plant measurements. Most plant variables exhibit a typical frequency spectrum under normal operating conditions. Any deviation from this can be interpreted as abnormality. Certain types of faults may even have their characteristic signature in the spectrum, facilitating fault isolation.

Logic reasoning. These techniques are complementary to the methods outlined above, in the sense that they use detection hardware or software to evaluate the symptoms. The simplest techniques consist of trees of logical rules of the "*IF-symptom-AND-symptom-THEN-conclusion*" type. Each rule produces the conclusion, which can serve as a symptom in the next rule, until the final conclusion is reached. The system may be configured to process the information presented by the detection hardware/software



standalone or may interact with a human operator in such a way that operator supervise the entire logical process and make decision how to cope with particular symptoms.

2.5.2 Model-based methods

Model-based fault diagnosis methods utilise an explicit mathematical model of the monitored plant. There are two main trends of quantitative model-based fault detection and diagnosis methods, namely *analytical redundancy* and *parameter estimation*.

Analytical redundancy. Most of the model-based fault detection and diagnosis methods rely on the concept of *analytical redundancy*. These methods share the common characteristic that determination of faults is obtained from the comparison of available system measurements with *a priori* information represented by the system's mathematical model, through generation of residual quantities and their analysis. Plant measurements, provided by sensors, are compared to analytically computed values of the corresponding variables. This is in contrast to physical redundancy, where measurements from parallel sensors are compared to each other.

Figure 2.2 illustrates the hardware and analytical redundancy concepts (Patton and Chen, 1999). The major problems encountered with hardware redundancy are the extra equipment, weight, maintenance cost and the additional space required to accommodate the equipment. The concept of analytical redundancy uses redundant analytical (or functional) relationships between various measured variables. In analytical redundancy scheme extra hardware is not required and, therefore, additional hardware faults are avoided.

Figure 2.3 shows a schematic description of the model-based fault diagnosis scheme (Patton and Chen, 1999). The process model is a quantitative or a qualitative description of the normal (fault-free) process dynamic and steady behaviour, which is obtained using well-established process modelling techniques.

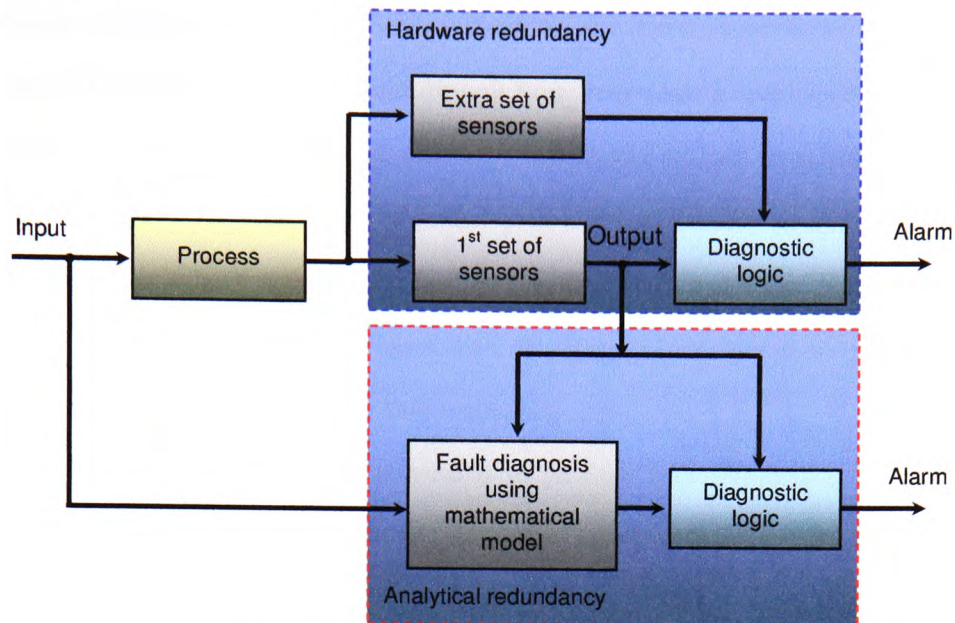


Figure 2.2 Hardware vs analytical redundancy.

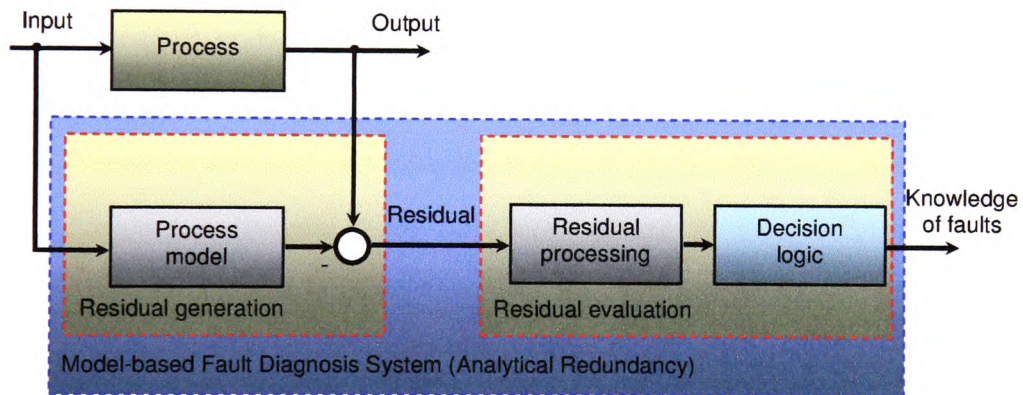


Figure 2.3 Model-based fault diagnosis scheme (analytical redundancy).

The process and process model are driven by the same inputs. The process model delivers an estimation of the measured process variables and comparing the estimated one with the measured one yields the *residual*. Residual quantities represent the difference between the measured signals and signals generated by the mathematical model. A residual is a fault indicator, which reflects the faulty situation of the monitored system. In the ideal case where there are no unknown inputs and an exact

process modelling is possible, the residual should be zero-valued when the system is in normal (fault-free) state, and should diverge from zero when a fault occurs in the system. No one technical process can be modelled exactly and there often exist unknown inputs. This means that a further processing of residuals is necessary to distinguish the faults from the model uncertainty and unknown inputs.

The general structure of a model-based fault diagnosis system consists of two main stages: *residual generation* and *residual evaluation*.

- **Residual generation:** Its purpose is to generate a fault-indicating signal – residual, using available input and output information from the monitored system. The algorithm used to generate residuals is called a residual generator. Hence, residual generation is a procedure for extracting fault symptoms from the system, with the fault symptom represented by the residual signal.
- **Residual evaluation:** The generation of residuals is followed by residual evaluation, with the goal of fault detection and, if possible, fault isolation. The residuals are examined for the likelihood of faults, and a decision rule is then applied to determine if any faults have occurred. A decision process may consist of a simple threshold test on the instantaneous values or moving averages of the residuals or it may consist of methods of statistical decision theory.

The commonly known approaches for residual generation can basically be divided into two categories of signal-based and model-based concepts with a further subdivision as shown in Figure 2.4 (Köppen-Seliger and Frank, 1999). The main research emphasis of the last two decades has been placed on the development of model-based approaches starting from analytical models and leading to the recently employed data-based models, such as neural networks.

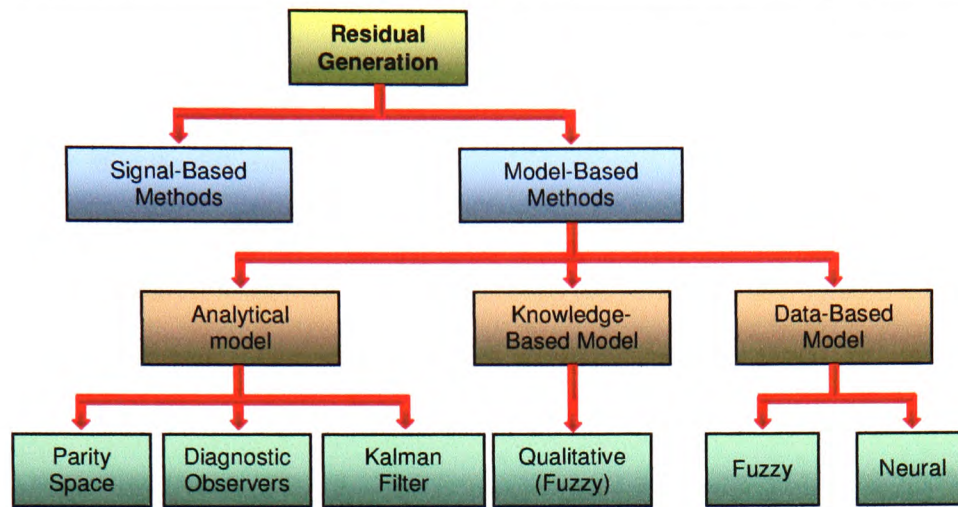


Figure 2.4 Classification of different residual generation concepts.

The concepts of Parity space, Diagnostic observers and Kalman filter are described in the following, while fuzzy and neural concepts are covered in sections 2.7 and 2.8.1, respectively.

- *Parity space.* Parity relations are rearranged direct input-output model equations, subjected to a linear dynamic transformation. The transformed residuals serve for detection and isolation. The design freedom provided by the transformation can be used for disturbance decoupling and fault isolation enhancement. Also, the dynamics of the response can be assigned, within the limits dictated by the requirements of causality and stability.
- *Diagnostic observers.* Different types of diagnostic observers are developed for residual generation. "Unknown input" design techniques may be used to decouple the residuals from (a limited number of) disturbances. The freedom in the design of the observer can be utilised to enhance the residuals for isolation. The dynamics of the fault response can be controlled, within certain limits, by placing the poles of the observer.

- *Kalman filter*. The innovation (prediction error) of the Kalman filter can be used as fault detection residual; its mean is zero if there is no fault (and disturbance) and becomes non-zero in the presence of faults. Since the innovation sequence is white, statistical tests are relatively easy to construct. However, fault isolation is somewhat awkward with the Kalman filter: it is necessary to run a bank of "matched filters", one for each suspected fault and for each possible arrival time, and check which filter output can be matched with the actual observations.

Residual evaluation techniques can be principally divided into threshold decisions, statistical methods and classification approaches (Figure 2.5).

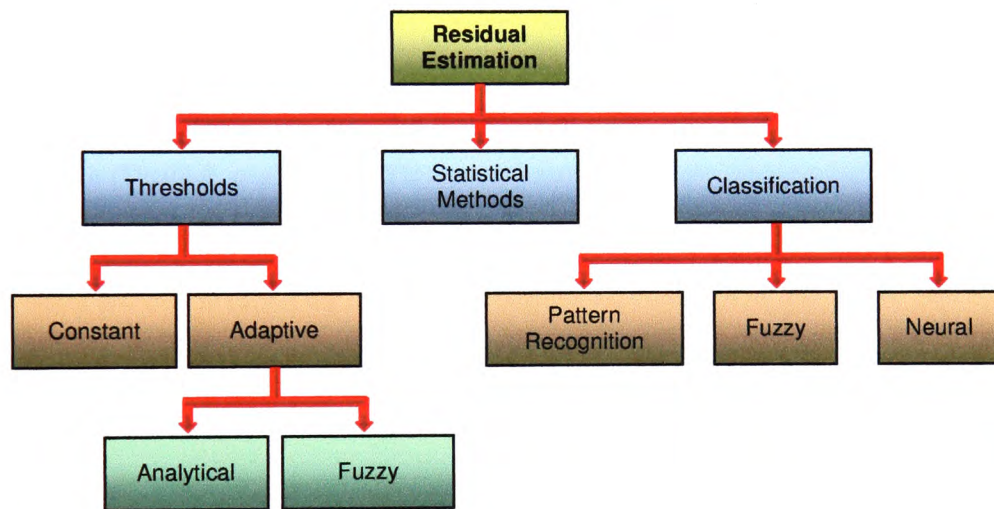


Figure 2.5 Classification of different residual evaluation concepts.

Parameter estimation. Parameter estimation is a natural approach to the detection and isolation of parametric (multiplicative) faults. The first step is to identify the plant in a fault-free situation and obtain reference model. Then the parameters are repeatedly recalculated on-line. Deviations from the reference model serve as a basis for detection and isolation. Parameter estimation may be more reliable than the analytical

redundancy methods, but it is also more demanding in terms of on-line computation and input excitation requirements (Gertler, 1998).

2.6 Some of recent fault diagnosis approaches

Fault diagnosis has received great attention in the research community during the last two decades. Recent developments in communication and instrumentation technologies have made possible to collect a large amount of real-time plant data. The existing control systems can be enhanced with fault diagnosis and accommodation modules, able to utilize this wealth of information in an intelligent way, to minimise plant downtime and to optimise plant operations. This section provides an overview of the recent approaches and new concepts in fault diagnosis. The main ideas are given herein, while more details can be found in referenced papers.

A sensor fault diagnosis and accommodation method, based on analytical redundancy, is proposed in (Theilliol, *et al.*, 2000). This method makes it possible the compensation of additive or multiplicative sensor faults in closed-loop control. When a sensor fault occurs, the control law tries to cancel the static error created by the corrupted output. Sensor fault tolerance control is achieved by computing a new control law using a fault-free estimation of the faulty element. In addition, a robust residual generation using unknown input observer is used for fault isolation. The performance of the approach was tested using simulated non-linear process (a three-tank system). Results showed that the proposed method was successful in detection, isolation and compensation of sensor faults. An approach to sensor fault detection and isolation, based on using *Principal Component Analysis* (PCA), is proposed in (Harkat, *et al.*, 2000). PCA is a statistical modelling technique, which finds the directions of significant variability in the data by analyzing the eigen vectors of the correlation matrix. PCA is used to model normal process behaviour and faults are then detected by referencing the observed behaviour against this model.

Process fault detection using PCA is performed by monitoring the residuals. An abnormal situation is flagged whenever this residual is statistically significant. A sensor validity index is calculated for isolation purpose. A test on the last principal components is proposed for the detection and the localisation of sensor failure, in order to overcome sensitivity to model errors. The proposed approach was successfully applied to air monitoring networks in Lorraine, France.

A model-based method for the detection and isolation of faults in an industrial gas turbine system is presented in (Patton, *et al.*, 2000b). The diagnosis system uses an output observers designed in both deterministic and stochastic environments. Identification procedures are used to obtain a model of the process under investigation. Residual analysis and statistical tests are used for fault detection and isolation, respectively. The proposed designs have been evaluated using non-linear simulation, based on gas turbine data.

Jakubek and Jörgl (2000) proposed an observer-based approach for sensor fault diagnosis that utilizes only one observer (Kalman filter). A parity check is performed on the observation errors such that even in the case of multiple simultaneous sensor faults correct fault detection, isolation and identification can be achieved. The method has been evaluated on industrial turbo-charged combustion engine power plant.

A combined qualitative and quantitative approach for fault isolation in continuous dynamic systems is proposed in (Manders, *et al.*, 2000). This scheme uses qualitative fault isolation to narrow down possible fault hypothesis and then uses a focused quantitative parameter estimation scheme to identify the true fault. The authors claimed that this approach provides a number of advantages over purely quantitative *Fault Detection and Isolation* (FDI) scheme.

The method for process condition monitoring, based on integration of fuzzy inference system and *Self-Organising Map* (SOM), is proposed in (Cuadrado, *et al.*, 2001). The method identifies regions in the SOM visualisation space, corresponding to different conditions of a monitored process by means of a fuzzy rule system, which incorporates expert knowledge about the process in the procedure of region identification.

The majority of papers in the literature, related with the problem of fault diagnosis, concern linear systems and far fewer non-linear dynamic systems (Zhirabok, 1997; Seliger and Frank, 1991; Zhirabok and Shumsky, 1987). The FDI of bilinear systems (a special class of non-linear systems) is considered in (Shields, 1996). The author proposes two main methods: *Bilinear Fault Detection Observer* (focused on the problem of decoupling the unknown inputs from the residuals) and *Parity Space Method for Bilinear Systems* (robust, but computationally expensive).

An interesting approach to observer-based fault diagnosis of a certain class of non-linear dynamic systems is proposed in (Zhirabok and Usoltsev, 2002). The main idea is replacing the initial non-linear system by certain linear logic-dynamic system, obtaining the bank of linear logic-dynamic observers, and transforming these observers into the non-linear ones. The authors developed the procedure of the linear logic-dynamic observer synthesis.

Another recent paper that addresses the fault diagnosis problem of non-linear systems is (Szigeti, *et al.*, 2002). The main contribution of the paper is an algorithm, which can be used for the calculation of the system inverse. Using the idea of input reconstruction by means of dynamic inversion, the authors first discuss the properties of fault observability in linear systems. The results are extended to non-linear systems, together with a mathematical framework, which provides calculation of the inverse system in finite algorithmic steps.

2.7 *Fuzzy logic in fault diagnosis*

A disadvantage of the analytical approach to fault diagnosis is that under real conditions no accurate mathematical models of the system of interest can be obtained. The robust analytical techniques described, for example, in (Patton and Chen, 1999) can overcome this deficiency only to a certain degree and with great effort. This consideration, together with the evolution of fuzzy and neural techniques, has led to the development of knowledge models and data-based models. In both approaches fuzzy logic can be integrated as depicted in Figure 2.4. In contrast to the qualitative approach, which utilise a rule-based model, a data-based fuzzy approach consist of a fuzzy relational module. The parameters of this module are trained by input-output data following a given performance criterion. Fuzzy logic tools can also be applied for residual evaluation in the form of a classifier as shown in Figure 2.5. One possibility is the combination of this qualitative approach with a quantitative residual generating algorithm. In the following section two different approaches using fuzzy logic in a fault diagnosis system are described.

2.7.1 A Fuzzy filter for residual evaluation

In practice, analytical models often exist only for parts of the plant - submodels. In general, the connections between the submodels are not specified analytically, and the analytical model-based methods cannot be used as a fault diagnosis tools for the entire plant. However, there always exists some useful qualitative or heuristic knowledge of the plant, which may not be very detailed but suitable to characterise, in linguistic terms, the connections between the existing analytical submodels. This means that, for the submodels, quantitative model-based techniques can be used, while the qualitative and heuristic knowledge of the connections can be used for the fault symptom generation of

the complete system. The advantages of using such a combined quantitative/knowledge-based approach can be summarised as follows (Köppen-Seliger and Frank, 1999):

- It is not necessary to build an analytical model of the complete process; it is sufficient to have only analytical submodels.
- The connections between the submodels can be described by qualitative or heuristic knowledge. Some qualitative or heuristic description of the plant or the interconnections between the submodels is usually available.
- The mathematical effort, compared to using the analytical model of the complete plant, is significantly reduced.
- The causes and effects of the faults can be transferred more easily into the fault diagnosis concept.

Fuzzy residual evaluation is a process that transforms quantitative knowledge (residuals) into qualitative knowledge (fault indications). Residuals, generated by analytical submodels, represent the inputs of the Fuzzy Filter (Figure 2.6), which consists of the three basic components: *fuzzification*, *inference* and *presentation of the fault indication*.

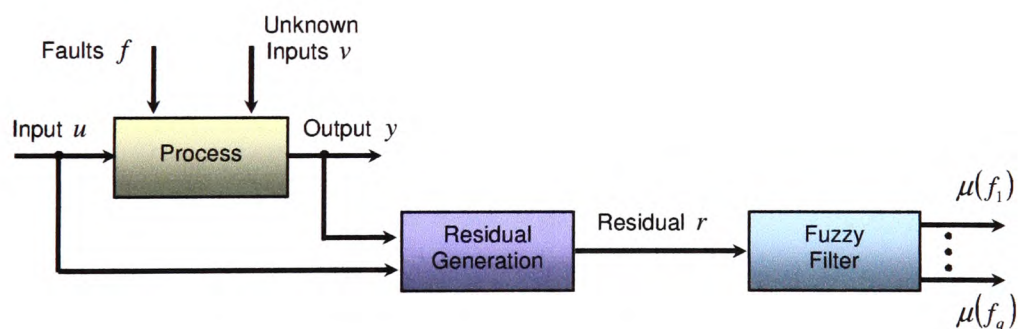


Figure 2.6 General structure of the fuzzy-filter-based diagnostic concept.

- **Fuzzification:** In this step, a knowledge base has to be built, which includes the definition of the faults of interest, the measurable residuals (symptoms), the relations between the residuals and the faults in terms of IF THEN rules and the

representation of the residuals in terms of fuzzy sets, for example, "normal" and "not normal".

- **Inference:** The process of inference includes determination of indication signals for the faults from the given rule base, with the aid of an inference mechanism. The result of inference is a fault indication signal found from a corresponding combination of residuals as characterised by the rules. This fault indication signal is called *Fuzzy Fault Indication Signal* (FFIS) and is in a fuzzyfied format.
- **Presentation of the fault indication:** The final task of the proposed FDI concept is the proper presentation of the fault situation to the operator, who has to make the final decision about the appropriate fault handling. Each FFIS is, by its nature, a singleton, the amplitude of which characterises the degree of membership to only one, preassigned fuzzy set " $fault_m$ ". This degree is characterised by the FFIS, i.e., the signal obtained as a result of the inference.

2.7.2 Fuzzy model-based parity equations for fault isolation

In this approach, a local linear fuzzy model of the process is used for the generation of structured parity equations (Ballé, 1999). The model is run both in parallel and in series-parallel to the process, which leads to residuals with different sensitivities. The sensitivities of the parallel and series-parallel residuals are compared, and the most sensitive residuals are selected for fault detection and isolation.

2.8 Pattern recognition in fault diagnosis

Many data-driven, analytical and knowledge-based fault diagnosis methods incorporate pattern recognition techniques to some extent (Chiang, *et al.*, 2001; Russel, *et al.*, 2000). Some pattern recognition methods use the relationship between the data patterns and fault classes without modelling the internal process states or structure explicitly. These

approaches include *Artificial Neural Networks* (ANN) and *Self-Organising Maps* (SOM). Reviews of pattern recognition approaches are given in (Micheli, 1999; Schalkoff, 1992). Two of the most popular pattern recognition approaches (artificial neural networks and self-organising maps) are described in more detail in the following subsections.

2.8.1 Neural networks

This section discusses how artificial neural networks can be used in FDI. A neural network is used to model a multi-input and multi-output non-linear dynamic system. After training, the neural network can give very accurate estimation of the system output. Using the residual generation concept developed in model-based fault diagnosis, the weighted difference between actual and estimated output is used as a residual to detect faults. When the magnitude of this residual exceeds a pre-defined threshold, it is likely the system is faulty. In order to locate faults in the system (fault isolation) reliably, a secondary neural network is used to examine features in the residual. A particular feature would correspond to a specific fault location. Based on feature extraction and classification principles, the second neural network can locate (or isolate) faults reliably.

Neural networks as models of non-linear dynamic systems

The feed-forward neural network is a static non-linear mapping from input to output space. Without modification, the feed-forward network cannot be used to represent dynamic systems. The simplest approach in representing non-linear dynamic systems is to use a combination of a feed-forward network with some time delay units (Patton and Chen, 1999). Assume that a non-linear dynamic system is described as:

$$\mathbf{y}(k) = \mathbf{F}(\mathbf{y}(k-1), \dots, \mathbf{y}(k-n), \mathbf{u}(k), \dots, \mathbf{u}(k-n)) \quad (2.1)$$

where $\mathbf{u}(k) \in \mathcal{R}^r$ is the input vector, $\mathbf{y}(k) \in \mathcal{R}^m$ is the output vector and $\mathbf{F}(\dots)$ represents a general non-linear function. A feed-forward network with weight matrix \mathbf{W}

can now be used to represent this static non-linear function, with output

$$\hat{y}(k) = NN(\mathbf{W}, \mathbf{y}(k-1), \dots, \mathbf{y}(k-n), \mathbf{u}(k), \dots, \mathbf{u}(k-n)) \quad (2.2)$$

where n is the system order and $NN(\cdot, \dots, \cdot)$ denotes a neural network-based non-linear functional mapping. This model, illustrated in Figure 2.7, is called the *one step prediction model*.

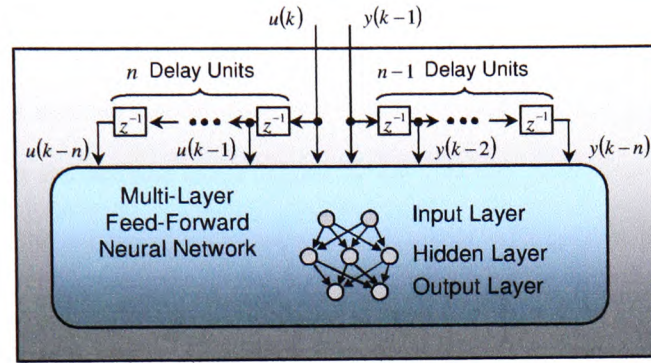


Figure 2.7 The neural network model of a non-linear dynamic system.

Neural networks as classifiers

After the residual has been generated, a decision-making mechanism must be used to determine fault occurrence and location. Traditionally, decision-making is implemented via threshold logic using either fixed or adaptive thresholds or statistical testing methods. The main task in decision-making is to classify the residuals into a number of distinguishable patterns corresponding to different faulty situations. Hence, decision-making can be based on the pattern recognition principle. Pattern recognition implies initiating certain actions based on the observation of input data.

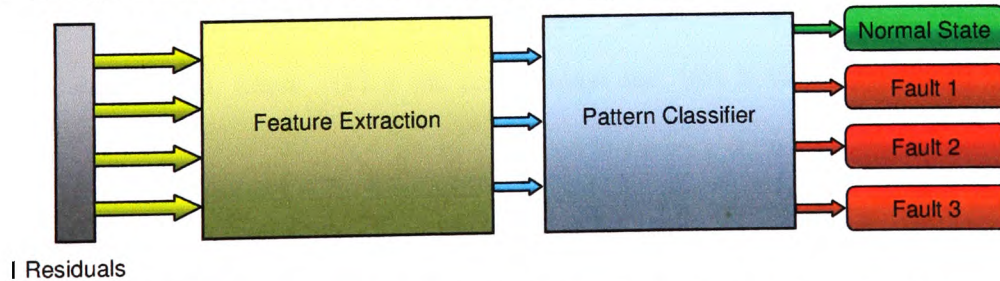


Figure 2.8 Decision-making mechanism in fault diagnosis system using pattern classifier.

Figure 2.8 shows decision-making mechanism, based on pattern classification. The input representing a pattern is known as the measurement or feature vector. The function performed by a pattern recognition system is the mapping of the input feature vector into one of the various decision classes. In fault diagnosis, these decision classes are the different types (or locations) of faults occurring in the system. One of the advantages of neural networks is their ability to partition the pattern space for classification problems. Hence, a neural network can be used as a classifier (or pattern recogniser) to partition residual patterns and activate alarm signals. It can therefore detect and isolate the faults accordingly. In the training of neural networks to classify faults, output node values of 0.1 and 0.9 are typically used to indicate fault-free and faulty cases, respectively. In the application to fault diagnosis, output values above 0.5 indicate a fault. If fault patterns are known to occur for specific faults, this information could be stored in the neural network by choosing the training set of the neural network to coordinate with known faults.

Fault diagnosis scheme based on neural networks

The neural networks-based FDI scheme, taken from (Patton and Chen, 1999), is illustrated in Figure 2.9. This scheme comprises two stages: *residual generation* and *decision-making*. The residual generation scheme described here is based on the comparison of actual and anticipated system responses. The anticipated system response is generated by a neural network-based prediction model, shown in Figure 2.7. The difference between actual and predicted outputs gives rise to a residual vector $r(k) = y(k) - \hat{y}(k)$, where $y(k)$ is the actual output and $\hat{y}(k)$ is the predicted output, defined by equation (2.2). The residuals generated in this way should be independent of the system operating state under nominal plant operating conditions. In the absence of faults, the residual is only due to unmodelled noise and disturbance. When a fault occurs in the system, the residual deviates from zero in characteristic ways.

- powerful non-linear mapping properties,
- noise tolerance,
- self-learning,
- parallel processing capabilities.

2.8.2 Self-organising maps

Neural network models can also be used for unsupervised learning using a SOM, in which the neural network learns some internal features of the input vectors. A SOM maps the non-linear statistical dependencies between high-dimensional data into simple geometric relationships, which preserve the most important topological and metric relationships of the original data. This allows the data to be clustered without knowing the class memberships of the input data.

Process monitoring using SOM

Figure 2.10 shows how the input vectors of the SOM are formed and manipulated when the monitored process is an industrial process (Alhoniemi, *et al.*, 1998; Simula and Kangas, 1995). Input and output measurements as well as process parameters are collected into a data buffer, where data is processed. Inputs, outputs and process parameters are concatenated to form a feature vector, which is used as an input to the SOM. Due to the topology preserving property of the map, similar features corresponding to similar states of the process are mapped close to each other resulting in clusters on the map.

In process monitoring, two different approaches can be distinguished:

1. The SOM may be applied in on- or off- line process analysis. In this case, the SOM provides analysis of normal operation of the process, without fault diagnosis capabilities.

- The SOM may be used to detect (and possibly identify) faults occurred in the process. Now the situation is opposite to one above: effect of faults is emphasised and variations in the normal operation are less important.

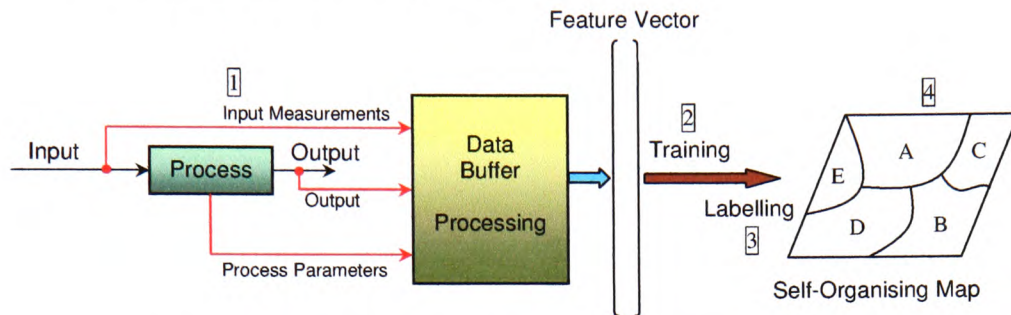


Figure 2.10 Application of the SOM in industrial process monitoring:

(1) Data processing, (2) Map training, (3) Validation, (4) Visualisation.

Process analysis using SOM

In on-line use, the SOM is used to form a display of the operational states of the process. The operation point (i.e., the current process state) and its history in time can be visualised as a trajectory on the map, which makes it possible to track the process dynamics. The SOM facilitates understanding of processes so that several variables and their interactions may be inspected simultaneously. In off-line analysis, the SOM is also a highly visual data exploration tool. In (Kasslin, *et al.*, 1992) a SOM was used to monitor the state of a power transformer and to indicate when the process was entering a non-desired state represented by a "forbidden" area on the map. In (Tryba and Gosser, 1991) a SOM was applied for monitoring of a distillation process and discussed its use in chemical process control in general.

Fault diagnosis using SOM

Another important application of the SOM is in fault diagnosis. The map can be used in two ways:

- to detect the fault,

2. to isolate the fault.

In practical applications, it is possible to distinguish between two different situations: whether measurements of the faulty situations are *a priori* available or not. In the learning space, the map is trained to recognise only those states of the process that are covered by the measurements. Thus, the state space is divided into two parts:

- the possible operation space,
- its complementary space.

Therefore, only the situations included into the training data can be recognised by the labelled map. In the case when the training data contains no measurements from faulty situations, the operation space on the map covers only normal situations. *Fault detection* can now be based on the quantisation error. A faulty situation can be detected by monitoring the *quantisation error* (distance between actual feature vector and its *Best Matching Unit* (BMU)). If the quantisation error is greater than a predetermined threshold the process is in a faulty situation i.e. the operating point belongs to the complementary space, not covered by the training data. Therefore, the situation has not occurred before and something is possibly going wrong.

The problem of *fault detection and isolation* is more difficult. The SOM should be trained using all possible data describing the process: both normal and abnormal situations should be present in the training data set. If necessary, measurements describing simulated faults may be added. Map units representing faulty states of the process may be marked (labelled) according to known samples. In these cases, clusters corresponding to certain faults are created on the map and these clusters can be considered as "forbidden" areas. The fault can now be easily identified by following the trajectory of the operating point. If the trajectory moves to a forbidden area the fault will be identified. Hence, location of the operating point on the map indicates the process state and the possible fault type.

2.9 Control allocation techniques for aircraft

Significant efforts have been undertaken in research community over last two decades to solve the control allocation problem for modern aircraft. Different methods for problem solution are briefly described here, while more details with precise mathematical formulation and examples can be found in Chapter 4. These methods are important, because the control allocation problems for an aircraft and an underwater vehicle are very similar. In both cases, the control allocation problem can be defined as the determination of the actuator control values that generate a given set of desired or commanded forces and moments. Publications on control allocation problem are almost exclusively application driven, which results in different notion and terminology, depending on particular application. In aerospace applications, the term *control effector* is used as a common name for control surface (elevator, ailerons, rudder, etc.) or force/moment generator (for example, thrust vectoring vane) of the aircraft. In order to reflect the ability of the control effectors to generate effects in addition to moments, a set of desired (commanded) forces and moments is referred as a *set of objectives* (Beck, 2002). A typical modern aircraft has many more effectors than objectives and the control allocation problem has many solutions. Multiple solutions provide the extra freedom and redundancy in the case of damage in one or more effectors. The similar case appears in solving the control allocation problem for underwater vehicles with overactuated control system in the horizontal plane, where the number of effectors (thrusters) is four and the number of objectives (desired surge and sway forces and yaw moment) is three. Some kind of criteria must be employed to extract the best solution from the infinite set of solutions. Bordignon (1996) summarised much of the history and early work on the control allocation problem in his dissertation. He described several methods, including various *ad hoc* schemes, direct control allocation methods, methods that belong to the

general group of generalised inverse methods and methods which daisy chain groups of controls. In the following, these methods are briefly reviewed.

2.9.1 Ad hoc methods

The common characteristic of the *ad hoc* methods, cited by Bordinon (1996), is that the control designer uses engineering judgment to assign individual control effectors to specific moment commands. The serious drawback of these methods is that these systems are unable to make use of the individual effector's capabilities to generate moments in axes other than that chosen by the designer.

2.9.2 Direct control allocation

Direct control allocation, proposed in (Durham, 1994a; 1993), is an approach based on the concept of the *Attainable Moment Set*¹ (AMS). The AMS (denoted by Φ) is the set of all moment vectors, \mathbf{m} , that are achievable within a set of control constraints (denoted by Ω). The motivation behind this method was the recognition that current solution methods to the control allocation problem, although sometimes computationally simple, were restrictive, i.e. the full set of moment-generating capability of the aircraft controls was not realisable by existing schemes. In contrast, direct control allocation, while computationally expensive, allows the entire attainable moment set to be used. The vector \mathbf{u} is called a control vector and its components are called controls. Direct control allocation solves the problem employing the following steps (Leedy, 1998):

1. Determine the actual AMS,
2. Search and solve for the intersection $\mathbf{m}^* = \alpha \mathbf{m}_d$ of the AMS and the half-line in the direction of \mathbf{m}_d ,

¹ Durham uses the term *moment* because he discusses the problem from the aircraft control perspective.

3. Find unique vector of controls \mathbf{u}^* , which produce the vector \mathbf{m}^* ,
4. Scale the result by the inverse scaling factor from step 2. i.e. compute

$$\mathbf{u} = \begin{cases} \frac{1}{a}\mathbf{u}^*, & a > 1 \\ \mathbf{u}^*, & a \leq 1 \end{cases}.$$

The method is based on the fact that, for overactuated systems, the linear mapping $\mathbf{m} = \mathbf{B}\mathbf{u}, \mathbf{m} \in \Phi, \mathbf{u} \in \Omega$ is many-to-one on the interior of Φ and one-to-one on the boundary of Φ , under assumption that any k columns of the *control effectiveness* matrix \mathbf{B} are linearly independent (linear independency condition), where k is the number of rows of \mathbf{B} . The equivalent interpretation of this condition is that no k actuators produce coplanar vectors. The direct control allocation concept can be extended to cover the systems that do not satisfy the linear independence condition (see Petersen and Bodson, 2000). Another difficulty of the direct allocation formulation is that the construction of solution requires that the lower and higher constraints of the control vector have opposite signs. This condition could be violated in the case of rate-limited actuators (see section 4.3.1). The original algorithm (Durham, 1993) was slow and difficult to implement. An elegant approach, proposed in (Durham, 1994b) reduced considerably the number of computations. Petersen and Bodson (1999) proposed a fast implementation using spherical coordinates and look-up tables.

2.9.3 Generalised inverse

For systems with equal number of effectors and objectives, the obvious method to solve the control allocation problem is to invert the control effectiveness matrix, \mathbf{B} . The extension of this approach for overactuated (under-determined) systems is to use a *Generalised Inverse* (GI) matrix. A right generalised inverse of a matrix \mathbf{B} is any matrix \mathbf{P} satisfying

$$\mathbf{BP} = \mathbf{I} \quad (2.3)$$

A solution of the control allocation problem $\mathbf{m} = \mathbf{Bu}$ can be found using \mathbf{P} as follows:

$$\mathbf{m} = \mathbf{Bu} \Rightarrow \mathbf{BPm} = \mathbf{Bu} \Rightarrow \mathbf{u} = \mathbf{Pm} \quad (2.4)$$

The most common choice for a GI is the Moore-Penrose pseudoinverse, $\mathbf{P} = \mathbf{B}^+ = \mathbf{B}^T (\mathbf{BB}^T)^{-1}$ (see Appendix B). In the absence of constraints on the control vector, this inverse minimises the l_2 norm of \mathbf{u} , i.e. $\|\mathbf{u}\|_2$. Control efforts of individual effectors can be weighted by minimising the weighted norm, $\|\mathbf{u}\|_w = \sqrt{\mathbf{u}^T \mathbf{W} \mathbf{u}}$, where \mathbf{W} is a positive definite, weighting matrix, usually diagonal. The resulting pseudoinverse matrix has the form $\mathbf{B}_w^+ = \mathbf{W}^{-1} \mathbf{B}^T (\mathbf{B} \mathbf{W}^{-1} \mathbf{B}^T)^{-1}$ (see Appendix B). Fossen (1995) used this approach to determine optimal distribution of propulsion forces with the minimal control effort (see page 2-52). GI solutions have the advantages of being relatively simple to compute and allowing some control in distribution of control energy among available effectors. However, handling of constrained controls is the most difficult problem for GI approach. In some cases, the solution obtained by generalised inverse approach is not feasible, i.e. it does not belong to Ω . Durham (1993) demonstrated that, except in certain degenerate cases, a general inverse cannot allocate controls inside Ω that will map to all of Φ , i.e. only subset of Φ can be covered. Bordinon (1996) suggested two methods to handle unfeasible solutions, i.e. cases where attainable objectives cannot be allocated². The first approach calculates a GI solution and truncates any controls which exceed their limits. The second approach maintains the direction of the objective command by finding the largest scaling factor, a ($0 \leq a \leq 1$), which satisfies $\mathbf{u} = a \mathbf{B}^+ \mathbf{y}_d$ without violating the

² The FDAS uses terms *truncation* and *scaling* for the first and the second type of approximation, respectively.

control constraints. Even if the controls do not saturate, care must be taken in choosing the GI. When weighted pseudo-inverse solutions are used for problems where the control effectors are measured in different physical dimensions, the elements of the weighting matrix must be chosen carefully if the resulting solution is desired to be invariant to changes in units and coordinate systems (Doty, *et al.*, 1993). The FDAS overcomes this problem by performing normalisation, such that all physical parameters are removed from the \mathbf{B} matrix and included in limit constraints, which are used during normalisation process to scale individual components of vectors on interval $[-1,1]$.

2.9.4 Daisy chain method

In the general case the standard GI approach is not able to yield admissible control for all AMS. This means that there is some control power available to improve the accuracy of solution, even when some of the effectors are saturated, i.e. instead of using any of two GI approaches mentioned in the previous section, it is possible to find even better solution, which uses this additional power (Beck, 2002; Bordignon, 1996). A *daisy chain* allocator partitions the control effectors into two sets, so that objectives unattainable by the first set are allocated with the second set. The method can be used for any type of control allocation scheme, although it was firstly proposed as improvement of GI method. The main idea of the daisy chain method is partitioning of control effectors into two groups using prioritising scheme, where a second set of effectors is used only when the first set is unable to meet the demands. Figure 2.11 illustrates the daisy chain approach. Primary set of controls is allocated first, and then the secondary set is (optionally) allocated using the residual objective $\mathbf{y}_{d2} = \mathbf{y}_d - \mathbf{B}\mathbf{u}_1$. Daisy chaining control allocation enables limiting the usage of certain control effectors. The most important drawback is the inability to allocate controls for some portions of Φ (Bordignon, 1996; Durham, 1993).

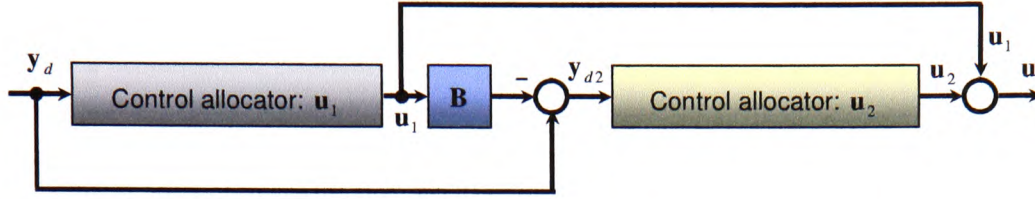


Figure 2.11 Daisy chain allocation method.

Another drawback is demonstrated in (Berg, *et al.*, 1996). The authors showed that there is a phase delay in the output of a daisy chain, in response to inputs that rate saturate the individual sets of controls in rate-limited systems.

2.9.5 Optimisation based methods

In the past, many optimisation based methods were not attractive for solution of the control allocation problem, because of their high demand for computational power. But, fast development of computer hardware and increasing of computational performance made these methods more suitable for control allocation. The control allocation problem can be pragmatically seen as a determination of feasible control vector \mathbf{u} for a given vector of objectives \mathbf{y} , such that $\mathbf{Bu} = \mathbf{y}$. If the solution is not unique, the best one must be found. If the solution does not exist, vector \mathbf{u} has to be determined such that \mathbf{Bu} approximates \mathbf{y} as well as possible. In light of this pragmatic interpretation, three mathematical formulations of the control allocation problem are proposed in (Bodson, 2002):

Error Minimisation Problem

Given a matrix \mathbf{B} and a vector \mathbf{y} , find a vector \mathbf{u} , which minimise

$$J = \|\mathbf{Bu} - \mathbf{y}\| \quad (2.5)$$

subject to

$$\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}^3 \quad (2.6)$$

Control Minimisation Problem

Given a matrix \mathbf{B} , a vector \mathbf{u}_p and a vector \mathbf{u}_1 such that $\mathbf{u}_{\min} \leq \mathbf{u}_1 \leq \mathbf{u}_{\max}$, find a vector \mathbf{u} which minimise

$$J = \|\mathbf{u} - \mathbf{u}_p\| \quad (2.7)$$

subject to

$$\mathbf{B}\mathbf{u} = \mathbf{B}\mathbf{u}_1 \quad (2.8)$$

and (2.6).

Mixed Optimisation Problem

Given a matrix \mathbf{B} , a vector \mathbf{y} and a vector \mathbf{u}_p , find a vector \mathbf{u} which minimise

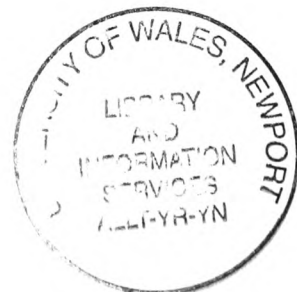
$$J = \|\mathbf{B}\mathbf{u} - \mathbf{y}\| + \varepsilon \|\mathbf{u} - \mathbf{u}_p\| \quad (2.9)$$

subject to (2.6).

The error minimisation problem is the most commonly encountered formulation of the control allocation problem. The l_2 norm is typically used in (2.5), although the l_1 norm has also been proposed in order to use linear programming techniques (Ikeda and Hood, 2000; Enns, 1998; Lindfors, 1993). In most cases, a weighting matrix is inserted in the norm to prioritise the axes.

The control minimisation problem is a secondary optimisation objective to be satisfied if the solution of the primary objective is not unique. The vector \mathbf{u}_p represents preferred position of the effectors (for example, zero deflections). The primary objective is to find the minimum error solution (2.5). If this solution is not unique, the secondary objective is

³ Vectors are compared on component-to-component basis.



to pick, among all these solutions, the solution with minimum deviation from the preferred position.

The mixed optimisation problem combines the error and control minimisation problems into a single problem through the use of a small, weighting parameter ε . If the parameter ε is small, priority is given to error minimisation over control minimisation, as desired. Often, the combined problem can be solved faster than the error and control minimisation problem solved separately and with better numerical properties.

Recently, some authors reformulated the constrained control allocation problem as a *Quadratic Programming* (QP) problem. QP generally refers to the numerical solution of the optimisation problems with an l_2 norm. An explicit solution approach is developed by Tøndel, *et al.* (2001). An on-line algorithm is presented in (Tøndel, *et al.* 2003), while the application to marine vessels is given in (Johansen, *et al.*, 2002). An alternative to the explicit solution is to use an iterative solution to solve the QP problem. The drawback with the iterative solution is that several iterations may have to be performed at each sample in order to find the optimal solution. An advantage of the iterative approach is that there is more flexibility for on-line reconfiguration. Computational complexity is also greatly reduced by a "warm start", i.e. the numerical solver is initialised with the solution of the optimisation problem from the previous sample (Fossen, 2002).

2.10 Fault diagnosis and accommodation for underwater vehicles

Recent advances and approaches in fault diagnosis and accommodation for underwater vehicles are given in this section. The approach adopted is to use the key words from the title of the paper as subsection heading. Approaches proposed by Yang, *et al.*, (1999; 1998), Podder, *et al.*, (2000) and Fossen (1995) are described in more detail, because the main ideas of these approaches were used as a foundation for development of the novel thruster fault diagnosis and accommodation system, described in Chapter 5.

Model-based approach for self-diagnosis of AUV (Takai and Ura, 1999)

Takai and Ura (1999) proposed a model-based approach for self-diagnosis of an AUV. The performance of the proposed system was examined through tank tests using a test-bed AUV called a "Twin-Burger", with two horizontal thrusters (HT1 and HT2) and one vertical thruster (VT). A key element of the self-diagnosis scheme was a recurrent neural network representation of the dynamics of the AUV. The proposed self-diagnosis system consists of two subsystems (see Figure 2.12): *Model Matching Part* (MMP) and *Diagnosis Part* (DP). The MMP includes a dynamic model that represents the characteristics of the vehicle's motion. The MMP produces index values that result from the comparison between sensor and model outputs. In the DP, the index values from the MMP are used to identify the defective component of the vehicle. When the information for accurate identification of the defective component is insufficient, the DP selects an appropriate predefined control sequence for the vehicle ("Active Diagnosis"), in order to acquire more information for identification. When a sensor failure is identified, then the outputs of the dynamics model are used instead of sensor outputs for control purpose ("Substituting Control").

As shown in Figure 2.12, the proposed self-diagnosis system introduces two diagnosis procedures ("Routine Diagnosis" and "Mission Diagnosis") in accordance with the situation when the diagnosis is carried out. The vehicle executes the routine diagnosis before starting a mission in order to check the hardware. A predefined control sequence, called "Diagnosis Motion Sequence", is applied to drive the actuators. The mission diagnosis is introduced to supervise the hardware condition of the AUV during the mission. The MMP continuously compares the model outputs with the sensor outputs during the mission and the DP verifies whether a fault exists or not. The example of a pattern table, used in the Diagnosis Part, is given in Table 2.1.

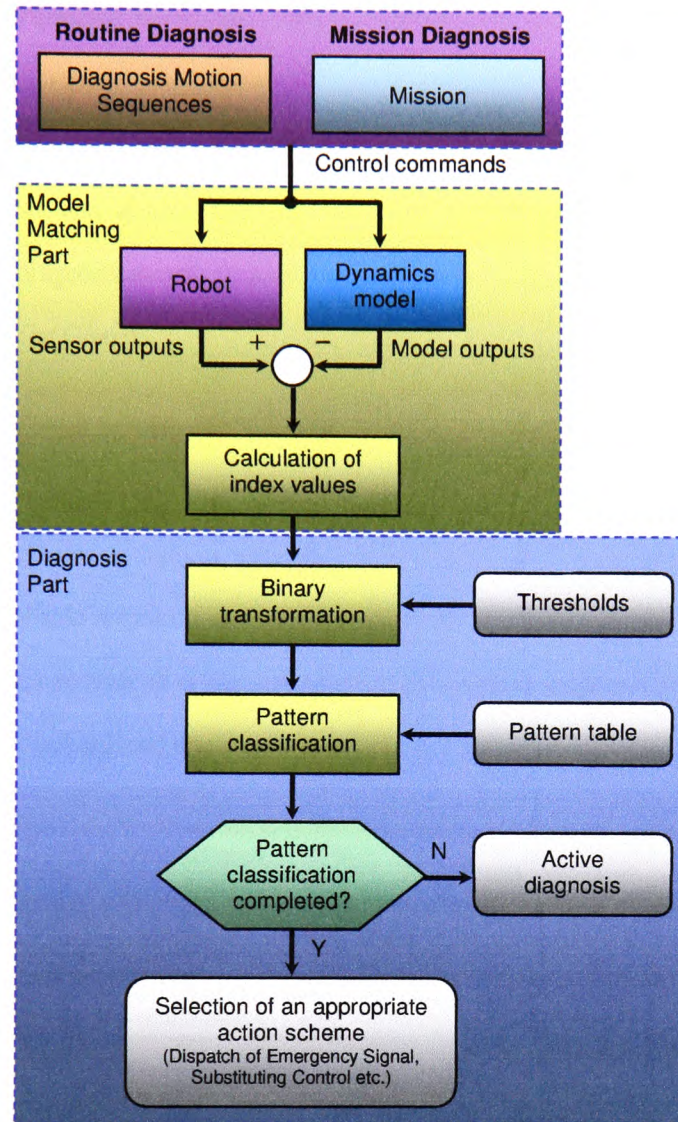


Figure 2.12 Self-diagnosis system for AUV, proposed in Takai and Ura (1999).

Case	State 0 - normal, 1 - abnormal			Defective component
	Surge velocity	Yaw rate	Heave velocity	
0	0	0	0	Nothing
1	1	1	0	HT1 or HT2
2	1	0	0	Sensor (Surge velocity)
3	0	1	0	Sensor (Yaw rate)
4	0	0	1	VT or Sensor (Heave velocity)

Table 2.1 Example of a pattern table.

When a fault is detected, the system selects an appropriate action in order to cope with the fault e.g. interrupt the mission or carry out the routine diagnosis to identify the fault. The proposed scheme was implemented on a test bed AUV, and results showed its ability to cope with sensor and actuator failures. The results of the experiment showed that the proposed self-diagnosis system was able to identify the fault and complete the mission, using Substituting Control.

On-line damage detection for AUV (Rae and Dunn, 1994)

The problem of on-line damage detection for AUV is addressed in (Rae and Dunn, 1994). Several expected system failures, examples of their occurrence, categories to which they belong and probable responses are listed in Table 2.2. For example, the creeping failure could be a motor seizing up or the loss of pressure in a ballast system, leading to slower response times, and reduced response magnitudes.

Damage	Type	Time of event	Result
Breaking	Instantaneous	Instant	Zero response
Jamming	Instantaneous	Instant	Permanent offset
Slipping	Intermittent	Intermittent	Intermittent
Creeping failure	Gradual	Persistent	Decreasing response
Control failure	Instantaneous	Any	Random

Table 2.2 Damage types.

The authors introduced the concept of "damage level", the amount of system degradation. A damage level of 0.0 indicates no damage, while 1.0 means total failure. Figure 2.13 displays three typical types of damage (Instantaneous, Gradual and Intermittent Failure). Damage to the system must be associated with the level over which the problem occurred, because of the existing discrepancy in time lengths of different activity levels (mission, sub-mission, task, manoeuvre). For example, consider the case where an instantaneous failure occurs during a task (typical length 10 minutes). This is equivalent to a gradual failure during a manoeuvre (typical length 1 minute).

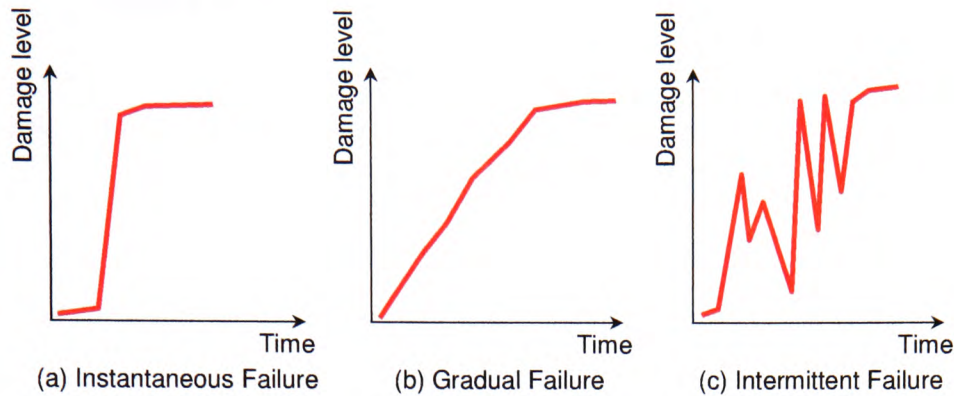


Figure 2.13 Types of damage.

Damage detection has been implemented and tested for three *Single Input Single Output* (SISO) systems (Rudder – Yaw, Sternplane – Depth and RPM – Speed). The non-linear simulation was developed on a Silicon Graphics IRIS. Recent changes in vehicle performance are determined by comparing two constantly updated models. These changes are indicated by differences in the coefficients of the two ("fast" and "slow") models. The information is extracted by performing several levels of manipulation on the coefficient difference time history. The authors proposed a hierarchical approach in order to monitor systems for slowly changing problems: slow models are used in a low level, while fast models are used in a higher level. Simulation results demonstrated that the proposed damage detection system was able to detect four typical failures in an AUV (slip of speed sensor, jam in the paddlewheel sensor, total depth-system failure, wind-up error in the compass).

Fault detection of actuator faults in UUV using bank of estimators (Alessandri, *et al.*, 1999)

A fault-diagnostic system for unmanned underwater vehicles was designed and tested in real operating conditions by (Alessandri, *et al.*, 1999). They considered total and partial actuator faults. An approximate model of the vehicle was used. A simple PID controller

has been designed to perform auto heading. A fault in a thruster acts on the dynamics of the vehicle as a virtual disturbance, for which the controller tries to compensate. Fault detection and diagnosis was accomplished by evaluating any change in the normal behaviour of the system by comparing the state, the parameters and other related quantities of the observed process with those of the normal and faulty processes. On the basis of the healthy and faulty models, a bank of estimators was used for the nominal plant, the left and the right actuator fault (see Figure 2.14).

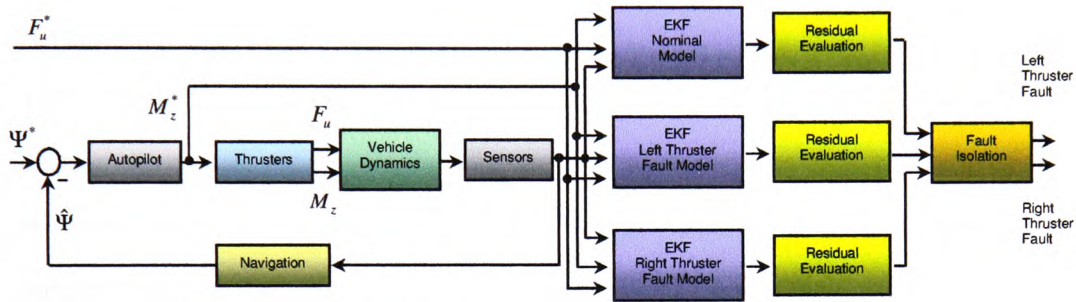


Figure 2.14 An observer-based fault-diagnostic scheme.

Extended Kalman filters (EKF) were implemented in the process of residual generation for each actuator fault type, including the no-fault case. This scheme showed effective isolation, at the cost of greater computational efforts. Experimental results proved the effectiveness of the proposed approach. Test trials were undertaken in pool with the Roby 2 vehicle. A right thruster fault was introduced by switching off the right propeller. The residuals of the EKF based on the model of the right propeller fault became smaller, while the residuals of the other two EKF became bigger. The residual evaluation involves three steps: (i) collect the residuals, (ii) compute the required quantities as functions of the residuals, (iii) choose the thresholds for each filter. It was found that unprocessed (non-filtered) residuals produced false fault alarms, which were eliminated by low-pass filtering. The authors mentioned that the evaluation of the residuals can be improved by

integrating all the available information, such as the external conditions of the environment or information related to the dynamics but not included in the model of the residual generator (Zhuang and Frank, 1997). For instance, the revolution rate and the current absorption of the propulsors are useful in monitoring the performance of the vehicle in cases of thruster malfunctions. If a leakage occurs in the canister containing one of the DC motors, an abnormal increase in the current is measured.

Fault-tolerant system design of AUV (Yang, *et al.*, 1999; 1998)

A fault-tolerant system for use in an experimental AUV was outlined in (Yang, *et al.*, 1999; 1998). The system was subdivided into individual fault-tolerant subsystems for dealing with thruster and sensor failures separately. The thruster subsystem consisted of a rule base for detection and isolation purposes, and an algorithm for reconfiguring the thruster control matrix by eliminating the corresponding column to accommodate the failure. Only a total fault (failure) of the thruster was considered. The authors used a constraint-based method instead of the pseudo-inverse method to compute the inverse of the thruster configuration matrix. An experimental investigation was conducted on a 6 DOF AUV, *Omni Directional Intelligent Navigator* (ODIN) at the University of Hawaii to evaluate the performance of the proposed approaches and experimental results showed that the overall system was capable of performing effectively. More details about these interesting works are given in the following.

ODIN is 6 DOF spherically shaped, underwater vehicle with four horizontal thrusters (HT1, HT2, HT3 and HT4) and four vertical thrusters (VT1, VT2, VT3 and VT4, see Figure 2.15). The vertical thrusters allow instantaneous coupled motions of pitch, roll and heave. The four horizontal thrusters allow instantaneous coupled motions of sway, surge and yaw. From the eight-thruster configuration, the vehicle possesses inherent thruster

redundancy, because the 6 DOF motion is possible with only six thrusters (three horizontal and three vertical thrusters).

It is assumed that the origin of the body-fixed frame $\{B\}$ is located in the centre of the sphere. The relationship between the vector of individual thruster forces \mathbf{f} and vector of total forces and moments $\boldsymbol{\tau}$, exerted by thrusters, is given by:

$$\boldsymbol{\tau} = \mathbf{TCM} \cdot \mathbf{f} \quad (2.10)$$

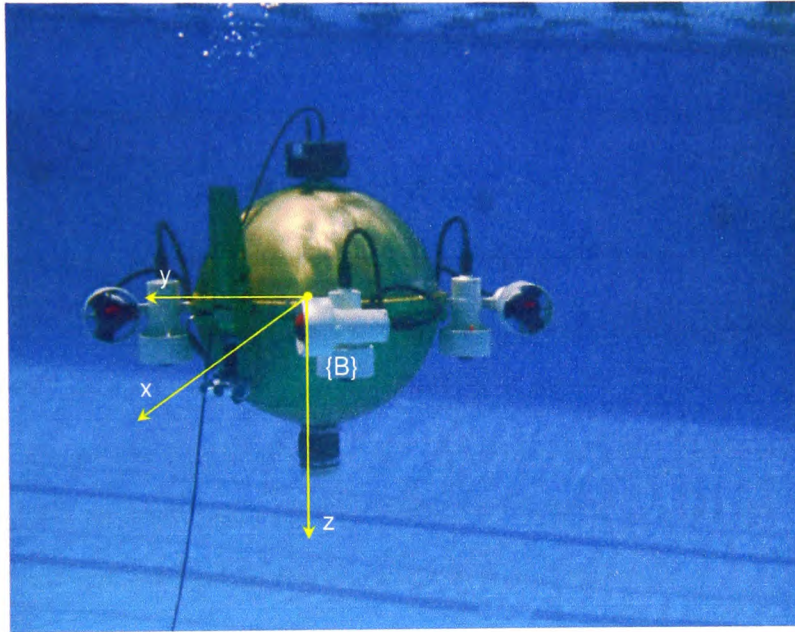


Figure 2.15 ODIN in the underwater environment.

where

$\boldsymbol{\tau} = [X \ Y \ Z \ K \ M \ N]^T$ is a generalised vector of total forces and moments, exerted by thrusters,

$\mathbf{f} = [HT_1 \ HT_2 \ HT_3 \ HT_4 \ VT_1 \ VT_2 \ VT_3 \ VT_4]^T$ is vector of individual thruster forces,

$$\mathbf{TCM} = \begin{bmatrix} s & -s & -s & s & 0 & 0 & 0 & 0 \\ s & s & -s & -s & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & R_v s & R_v s & -R_v s & -R_v s \\ 0 & 0 & 0 & 0 & R_v s & -R_v s & -R_v s & R_v s \\ R_H & -R_H & R_H & -R_H & 0 & 0 & 0 & 0 \end{bmatrix} \text{ is thruster control}$$

matrix,

$$s = \sin 45^\circ,$$

R_v is the radial distance from centre to the centre of the vertical thruster,

R_H is the radial distance from centre to the centre of the horizontal thruster.

To calculate the required input voltages for the thrusters, the force requirement for individual thruster must be obtained by inverting thruster control matrix

$$\mathbf{f} = \mathbf{TCM}^{-1} \boldsymbol{\tau} \quad (2.11)$$

Matrix \mathbf{TCM} is non-square 6×8 matrix and cannot be inverted in standard way. Instead of using the pseudo-inverse method for a non-square matrix \mathbf{TCM} , the authors introduced the constraint-based method. Firstly, the \mathbf{TCM} is separated into two 3×4 matrices (separated relationships for horizontal and vertical thrusters):

$$\text{horizontal TCM: } \begin{bmatrix} X \\ Y \\ N \end{bmatrix} = \begin{bmatrix} s & -s & -s & s \\ s & s & -s & -s \\ R_H & -R_H & R_H & -R_H \end{bmatrix} \begin{bmatrix} HT_1 \\ HT_2 \\ HT_3 \\ HT_4 \end{bmatrix} \quad (2.12)$$

$$\text{vertical TCM: } \begin{bmatrix} Z \\ K \\ M \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 & -1 \\ R_v s & R_v s & -R_v s & -R_v s \\ R_v s & -R_v s & -R_v s & R_v s \end{bmatrix} \begin{bmatrix} VT_1 \\ VT_2 \\ VT_3 \\ VT_4 \end{bmatrix} \quad (2.13)$$

During normal operation, without any thruster failure, each of 3×4 matrices can be modified to a 3×3 matrix by applying a proper constraint to one of the thrusters (one horizontal and one vertical thruster).

$$\text{constraint on horizontal thrusters (normal operation):} \quad HT_2 = -HT_4 \quad (2.14)$$

$$\text{constraint on vertical thrusters (normal operation):} \quad VT_2 = VT_3 \quad (2.15)$$

These constraints were chosen based on the fact that minimising the yaw capability by allowing only two thruster involvement is more efficient than minimising thruster involvement for the local x (surge) or local y (sway) directions. After introducing the constraints (2.14) & (2.15) in (2.12) & (2.13) respectively, modified relationships are:

$$\text{modified horizontal TCM:} \quad \begin{bmatrix} X \\ Y \\ N \end{bmatrix} = \begin{bmatrix} s & -2s & -s & 0 \\ s & 2s & -s & 0 \\ R_H & 0 & R_H & 0 \end{bmatrix} \begin{bmatrix} HT_1 \\ HT_2 \\ HT_3 \\ 0 \end{bmatrix} \quad (2.16)$$

$$\text{modified vertical TCM:} \quad \begin{bmatrix} Z \\ K \\ M \end{bmatrix} = \begin{bmatrix} -1 & -2 & 0 & -1 \\ R_V s & 0 & 0 & -R_V s \\ R_V s & -2R_V s & 0 & R_V s \end{bmatrix} \begin{bmatrix} VT_1 \\ VT_2 \\ 0 \\ VT_4 \end{bmatrix} \quad (2.17)$$

The final relationships for the force requirements are

$$\begin{bmatrix} HT_1 \\ HT_2 \\ HT_3 \end{bmatrix} = \begin{bmatrix} s & -2s & -s \\ s & 2s & -s \\ R_H & 0 & R_H \end{bmatrix}^{-1} \begin{bmatrix} X \\ Y \\ N \end{bmatrix}, \quad HT_4 = -HT_2 \quad (2.18)$$

$$\begin{bmatrix} VT_1 \\ VT_2 \\ VT_4 \end{bmatrix} = \begin{bmatrix} -1 & -2 & -1 \\ R_V s & 0 & -R_V s \\ R_V s & -2R_V s & R_V s \end{bmatrix}^{-1} \begin{bmatrix} Z \\ K \\ M \end{bmatrix}, \quad VT_3 = VT_2 \quad (2.19)$$

To obtain the relationship between the input voltage and the output force of the thruster, an experiment using the single-thruster system set-up was conducted (Tsukamoto, *et al.*, 1999) and the following relationship between input voltage V and thruster force F_T was obtained (for both rotation directions):

$$V = \begin{cases} \frac{453.6F_T + 18.978}{505.05}, & F_T \geq 0 \\ \frac{453.6F_T - 48.312}{624.7}, & F_T < 0 \end{cases} \quad (2.20)$$

The thruster fault detection subsystem was built with the following assumptions:

- The maximum number of failed thrusters during the mission is two: one horizontal and one vertical thruster.
- Once a faulty thruster is detected, the thruster is assumed to be completely out of service throughout the operation.

The first assumption guarantees that fault accommodation will not produce the loss in controllable DOF i.e. after accommodation the vehicle will still possess 6 controllable DOF. Each thruster is equipped with a Hall-effect sensor for measurement of shaft velocity. The main premise for the fault detection is that if errors between the input voltage (desired voltage calculated by the controller and **TCM**) and output voltage (measurement of Hall-effect sensor) do not stay within a tolerable limit inside predefined tolerance time, the thruster is concluded to be faulty. Values for tolerance 0.06 and tolerance time 3.0s were obtained by experiments. After the first stage, where the fault detection-isolation scheme detects and isolates a fault, the next stage is thruster fault accommodation, based on reconfiguration of the **TCM**. Because of inherent thruster redundancy, accommodation is performed by eliminating the corresponding column of the **TCM**. For example, if VT_1 is detected to be faulty, the control program reconfigures the vertical **TCM** (2.13) by eliminating corresponding (first) column in the matrix and reducing it to a 3×3 matrix. The reduced matrix is non-singular and can be inverted in the standard way. Hence, in the case of fault in a thruster, it is not needed to apply any constraints to obtain an invertible matrix, as in the normal (fault-free) case (2.18) & (2.19). Hence, proposed thruster fault-tolerant algorithm involves these steps:

1. Obtain fault signal from detection scheme,
2. Eliminate the corresponding column in the **TCM**,
3. Invert the resulting matrix to obtain the required thrust for each thruster,
4. Calculate the voltage input signal (2.20).

In order to evaluate proposed thruster fault-tolerant system, test trials were undertaken at University of Hawaii. Fault in a vertical thruster 2 was realised with a pre-programmed Hall-effect sensor to generate zero output in the case of fault. When the output reads 0V and the error residual exceeds its tolerance limit 0.06 for tolerant time 3.0s, the detection scheme decides that the thruster is faulty. The fault signal is sent to the control program and the proper accommodation automatically takes place. In the first experiment the fault was injected during the steady-state and after some fluctuation the control system stabilized the vehicle with two thrusters and maintained the desired depth. Vertical thruster 4, symmetrical to faulty thruster 2, is immediately switched off after the accommodation period and voltage inputs to thrusters 1 and 3 become doubled compared to time before fault occurred, to compensate the loss of force of thrusters 2 and 4. A similar experiment was repeated but this time a fault was injected in vertical thruster 1 during a depth-changing manoeuvre. The behaviour of the proposed thruster detection, isolation and accommodation system was similar as in steady-state case, except for the larger oscillations in roll and pitch motions that caused a large deviation from the desired trajectory. This was expected since a thruster failure during the diving motion would result in increased roll and pitch motions due to the moments created by the water turbulence; when two thrusters are switched off, the external moments cannot be immediately counterbalanced, resulting in oscillations. However, at steady state, ODIN retained the desired depth. The authors developed a voting technique with redundant sensors for accommodation of sensor faults, which compares the signals of two actual

sensors and signal of a virtual sensor. Experimental results demonstrated that the presented approaches could effectively detect, isolate and accommodate thruster and sensor failures, thereby allowing the vehicle to accomplish the initially given task.

ROV actuator fault diagnosis through servo-amplifiers' monitoring (Bono, *et al.*, 1999)

A fault detection, isolation and accommodation system, based on operationally experienced faults in ROV actuators, is proposed in (Bono, *et al.*, 1999). The authors designed a fault management system for underwater vehicles, able to satisfy the basic requirement of handling experienced faults (e.g. flooded thruster) and conventional zero output failures treated in the literature. In addition, the fault management system had to be easily integrated within the hierarchical control architectures. The authors published experience from the sea trials, when the water penetrated inside the thruster and modified the internal electrical connections in such a way that the actual angular speed was higher than the desired one, and current consumption was higher than normal. In particular, the salt water caused a dispersion, which reduced the feedback signal of the motor revolution rate from the tachometer to the servo-amplifier. Fault detection was performed by monitoring the servo-amplifiers residuals, while fault isolation required the vehicle to execute steady-state manoeuvres. Actuator fault accommodation was performed by inhibiting the faulty thruster and by reconfiguring the distribution of the control actions cancelling the corresponding column in the *Thruster Control Matrix* (TCM).

Fault tolerant control of an AUV under thruster redundancy (Podder, *et al.*, 2000)

The problem of optimal distribution of propulsion forces for overactuated underwater vehicles is addressed in (Podder, *et al.*, 2000). The authors investigate how to exploit the excess number of thrusters to accommodate thruster faults. First, a redundancy resolution

scheme is presented, which takes into account the presence of an excess number of thrusters along with any thruster faults and determines the reference thruster forces to produce the desired motion. In the next step, these reference thruster forces are utilized in the thruster controller to generate the required motion. This approach resolves the thruster redundancy in the Cartesian space and allows the AUV to track the task-space trajectories with asymptotic reduction of the task-space errors. Results from both computer simulations and experiments were provided to demonstrate the viability of the proposed scheme. The paper is a development of the preliminary concept proposed in (Podder and Sarkar, 1999). This concept is described below in more detail.

The dynamic equation of motion of an AUV can be written in the following form:

$$\mathbf{M}\dot{\mathbf{w}} + \mathbf{C}(\mathbf{w})\mathbf{w} + \mathbf{D}(\mathbf{w})\mathbf{w} + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau} \quad (2.21)$$

where

$\mathbf{q}_{6 \times 1}$ is the generalised position and orientation vector in the Earth-fixed frame,

$\mathbf{w}_{6 \times 1}$ is the linear and angular velocity vector in the body-fixed frame,

$\mathbf{M}_{6 \times 6}$ is the inertia matrix of the AUV which includes both the rigid body and the added mass terms,

$\mathbf{C}_{6 \times 6}$ is the matrix of centrifugal and Coriolis terms which includes both the rigid body and added mass terms,

$\mathbf{D}_{6 \times 6}$ is the matrix of hydrodynamic drag terms,

$\mathbf{G}_{6 \times 1}$ is the vector of restoring forces (gravity and buoyancy),

$\boldsymbol{\tau}_{6 \times 1}$ is the vector of generalised forces, exerted by thrusters.

It is assumed that n (number of thrusters) is greater than 6. The relationship between $\boldsymbol{\tau}$ and \mathbf{F}_i is given by

$$\boldsymbol{\tau} = \mathbf{E}\mathbf{F}_t \quad (2.22)$$

where

$\mathbf{E}_{6 \times n}$ is the *thruster configuration matrix*,

$\mathbf{F}_{t \times 1}$ is the vector of thruster forces.

The matrix \mathbf{E} captures the geometry of the AUV and its thruster positions and orientations to transform the individual thruster force into generalised forces and moments. Substituting (2.22) in (2.21) yields

$$\dot{\mathbf{w}} = \mathbf{M}^{-1}(\mathbf{E}\mathbf{F}_t - \boldsymbol{\zeta}) \quad (2.23)$$

where

$$\boldsymbol{\zeta} = \mathbf{C}(\mathbf{w})\mathbf{w} + \mathbf{D}(\mathbf{w})\mathbf{w} + \mathbf{G}(\mathbf{q}) \quad (2.24)$$

The relationship between $\dot{\mathbf{q}}$ and \mathbf{w} is given by linear transformation

$$\dot{\mathbf{q}} = \mathbf{B}\mathbf{w} \quad (2.25)$$

where $\mathbf{B}_{6 \times 6}$ is the transformation matrix. Differentiation of (2.25) leads to

$$\ddot{\mathbf{q}} = \mathbf{B}\dot{\mathbf{w}} + \dot{\mathbf{B}}\mathbf{w} \quad (2.26)$$

The task space (i.e. the Cartesian space) velocity and the derivative of the generalised coordinates are related by the following relation:

$$\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}} \quad (2.27)$$

where

$\mathbf{x}_{m \times 1}$ is position and orientation vector in the task space ($m \leq 6$),

$\mathbf{J}_{m \times 6}$ is the Jacobian matrix.

Differentiating (2.27) and combining with (2.25) & (2.26) leads to

$$\ddot{\mathbf{x}} = \mathbf{J}\mathbf{B}\dot{\mathbf{w}} + (\dot{\mathbf{J}}\mathbf{B} + \mathbf{J}\dot{\mathbf{B}})\mathbf{w} \quad (2.28)$$

Finally, substituting $\dot{\mathbf{w}}$ from (2.23) into (2.28) gives the following relationship between the task-space acceleration and the thruster forces

$$\ddot{\mathbf{x}} = \boldsymbol{\mu} \mathbf{F}_t + \boldsymbol{\beta} \quad (2.29)$$

where

$\boldsymbol{\mu}_{m \times n} = \mathbf{J} \mathbf{B} \mathbf{M}^{-1} \mathbf{E}$ is the *thruster control matrix*,

$\boldsymbol{\beta}_{m \times 1} = (\mathbf{J} \dot{\mathbf{B}} + \dot{\mathbf{J}} \mathbf{B}) \mathbf{w} - \mathbf{J} \mathbf{B} \mathbf{M}^{-1} \boldsymbol{\zeta}$ is the vector of nonlinear terms.

The Moore-Penrose Generalised Inverse of the non-square matrix $\boldsymbol{\mu}$ was employed to obtain the unique least-norm solution for the thruster force

$$\mathbf{F}_t = \boldsymbol{\mu}^+ (\ddot{\mathbf{x}} - \boldsymbol{\beta}) \quad (2.30)$$

where $\boldsymbol{\mu}^+ = \boldsymbol{\mu}^T (\boldsymbol{\mu} \boldsymbol{\mu}^T)^{-1}$ is the pseudoinverse of $\boldsymbol{\mu}$ (see Appendix B).

The author developed a method of thruster forces redistribution (reallocation) in the presence of a fault in one or more thrusters, without going into details of the possible nature of thruster faults and how they can be detected. The objective is that, when a fault in a thruster occurs, thruster forces should be redistributed among the functioning thrusters in such a way that the AUV still follows the desired task-space trajectories. The weighted pseudoinverse $\boldsymbol{\mu}_w^+$ of the thruster control matrix $\boldsymbol{\mu}$ is introduced, in order to incorporate the thruster faults into the mathematical formulation of the problem (Ben-Israel and Greville, 1974):

$$\boldsymbol{\mu}_w^+ = \mathbf{W}^{-1} \boldsymbol{\mu}^T (\boldsymbol{\mu} \mathbf{W}^{-1} \boldsymbol{\mu}^T)^{-1} \quad (2.31)$$

where $\mathbf{W}_{n \times n}$ is a symmetric and positive definite weight matrix.

The weighted least-norm solution for the thruster force is defined by

$$\mathbf{F}_t = \boldsymbol{\mu}_w^+ (\ddot{\mathbf{x}} - \boldsymbol{\beta}) \quad (2.32)$$

The individual components of the thruster force vector \mathbf{F}_i can be modified by changing the weights in the weighting matrix \mathbf{W} . The authors introduced *Thruster Fault Matrix* $\Psi_{n \times n}$, defined as

$$\Psi = \mathbf{W}^{-1} \quad (2.33)$$

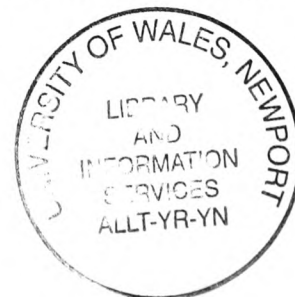
where $\mathbf{W} = \mathbf{I}_{n \times n}$ is identity matrix. In the normal case, without any faults, $\Psi = \mathbf{W}^{-1} = \mathbf{I}^{-1} = \mathbf{I}$ and equation (2.32) becomes equation (2.30). In the case of failure in a thruster T_i , the associated weight w_i in the matrix \mathbf{W} becomes equal to infinity and corresponding diagonal element ψ_i of the matrix Ψ becomes equal to zero and equation (2.32) becomes different than equation (2.30).

Rewriting (2.32) in terms of thruster fault matrix it is possible to obtain an expression for the allocation of thruster forces subject to thruster faults:

$$\mathbf{F}_i = \Psi \boldsymbol{\mu}^T (\boldsymbol{\mu} \Psi \boldsymbol{\mu}^T)^{-1} (\ddot{\mathbf{x}} - \boldsymbol{\beta}) \quad (2.34)$$

Equation (2.34) allows determining a thruster force allocation in the presence of thruster faults.

Computer simulations and underwater experiments with ODIN, University of Hawaii, were undertaken in order to evaluate the proposed approach. The thruster faults were simulated by imposing zero voltages to the relevant thrusters. Results from two cases were presented. The first case was trajectory following task without any fault in thrusters. Both (simulation and experimental) results for all six trajectories matched corresponding desired trajectories within reasonable limits. In the second case, the same trajectory following task was performed, but this time with thruster faults. Firstly, a fault was injected in a horizontal thruster and then, after some time, in a vertical thruster. Both faulty thrusters were located at the same thruster bracket of ODIN. Thus this situation is one of the worst fault conditions. The only visible consequence of the faults was in yaw



response, where a small error was visible at the last part of the trajectory. Beside that, there was small perturbation after the second fault, from which the controller quickly recovered. The authors observed that, besides a lot of similarity, there was discrepancy between simulation and experimental results. In particular, discrepancy was visible for responses of vertical thrusters. Several reasons for explanation of these differences were highlighted (ODIN is not perfectly spherical, thruster model is not perfect, presence of unexpected disturbance from water current in swimming pool, presence of the noise in the sonar navigation system, effects of the cable are neglected).

Optimal distribution of propulsion and control forces (Fossen, 1995)

If the number of control inputs p is equal to or more than the number of controllable DOF n , it is possible to find an "optimal" distribution of the control energy, which minimises the quadratic energy cost function i.e. a measure of the control effort (Fossen, 1995). The quadratic energy cost function is defined as

$$J = \frac{1}{2} \mathbf{u}^T \mathbf{W} \mathbf{u} \quad (2.35)$$

where $\mathbf{u}_{p \times 1}$ is control vector defined as

$$u_i = |n_i| n_i, \quad i = \overline{1, p} \quad (2.36)$$

and n_i is angular velocity (propeller revolution) of the thruster T_i . The relation between force vector $\boldsymbol{\tau}$ and control vector \mathbf{u} is determined by affine thruster model

$$\boldsymbol{\tau} = \mathbf{B} \mathbf{u} \quad (2.37)$$

where $\mathbf{B}_{n \times p}$ is thruster control matrix. The problem can be formulated as follows: find \mathbf{u} , which minimise criteria (2.35) subject to constraint (2.37). \mathbf{W} is a positive definite matrix, usually diagonal, weighting the control energy. The solution of the problem is given in Lema B.2 (Appendix B).

Hence, if desired force vector τ_d is known, then required control vector \mathbf{u} , which should be applied to actuate thrusters can be computed from

$$\mathbf{u} = \mathbf{B}_w^+ \tau_d \quad (2.38)$$

If control vector \mathbf{u} (2.38) is applied to actuate the thrusters, the force vector $\tau = \mathbf{B}\mathbf{u}$, exerted by thrusters, is equal to desired force vector τ_d in the ideal case, neglecting possible thruster velocity saturation. However, in real applications saturation always exists and must be taken into account.

2.11 Concluding remarks

The intention of the chapter was to undertake a literature survey to explain fundamental approaches to fault diagnosis and accommodation of dynamic systems. Special attention was devoted to the control allocation techniques for aircraft and fault diagnosis approaches applied to underwater vehicles. It is possible to summarise the weak points of latter approaches as follows:

- The assumption that a faulty sensor generates a zero output is not realistic in practical applications,
- The problem of the accommodation of a partial thruster fault is mentioned, but not fully solved.

The work presented in this thesis addresses these issues and provides an original, practical solution to the problem, which is integrated into a novel thruster fault diagnosis and accommodation system, described in Chapter 5.

2.12 References

- ALESSANDRI, A., CACCIA, M. and VERUGGIO, G. (1999). Fault detection of actuator faults in unmanned underwater vehicles. *Control Engineering Practice*, 7, pp. 357-368.
- ALHONIEMI, E., HOLLMEN, J., SIMULA, O. and VESANTO, J. (1998). *Process monitoring and modelling using the self-organising map*. Helsinki: Helsinki University of Technology.
- BALLÉ, P. (1999). Fuzzy model-based parity equations for fault isolation. *Control Engineering Practice*, 7, pp. 261-270.
- BECK, R.E. (2002). *Application of Control Allocation Methods to Linear Systems with Four or More Objectives*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.
- BEN-ISRAEL and GREVILLE, T.N.E. (1974). *Generalised Inverse: Theory and applications*. Wiley, New York.
- BERG, J.M., HAMMETT, K.D., SCHWARTZ, C.A. and BANDA, S.S. (1996) An analysis of the destabilizing effect of daisy chained rate-limited actuators. *IEEE Transactions on Control Systems Technology*, 4(2), pp. 171-176.
- BLANKE, M. (2001). Enhanced maritime safety through diagnosis and fault tolerant control. *IFAC Conference on Control Applications and Marine Systems, CAMS 2001*, Glasgow. (Invited plenary)
- BLANKE, M., STAROSWIECKI, M. and EVA WU, N. (2000b). Concepts and methods in fault-tolerant control. *Proc. American Control Conference*, Washington. (Invited tutorial)
- BONO, R., BRUZZONE, GA., BRUZZONE, GI. and CACCIA, M. (1999). ROV actuator fault diagnosis through servo-amplifiers' monitoring: an operational experience. *MTS/IEEE Oceans'99*, Vol. 3, pp. 1318-1324, Seattle, USA.
- BORDIGNON, K.A. (1996). *Constrained Control Allocation for Systems with Redundant Control Effectors*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.
- CHIANG, L.H., RUSSEL, E.L. and BRAATZ, R.D. (2001). *Fault Detection and Diagnosis in*

Industrial Systems. London: Springer-Verlag.

CUADRADO, A.A., DÍAZ, I., DÍEZ, A., OBESO, F. and GONZÁLEZ, J.A. (2001). Fuzzy inference maps for condition monitoring with self-organising maps. *An International Conference in Fuzzy Logic and Technology (EUSFLAT 2001)*, pp. 55-58, Leicester, UK.

DOTY, K.L., MELCHIORRI, C. and BONIVENTO, C. (1993) A theory of generalised inverses applied to robotics. *The International Journal of Robotics Research*, **12**(1), pp. 1-19.

DURHAM, W.C. (1993) Constrained Control Allocation. *Journal of Guidance, Control, and Dynamics*, Vol. **16**, No. 4, pp. 717-725.

DURHAM, W.C. (1994a) Constrained Control Allocation: Three Moment Problem. *Journal of Guidance, Control, and Dynamics*, Vol. **17**, No. 2, pp. 330-336.

DURHAM, W. (1994b) Attainable Moments for the Constrained Control Allocation Problem. *Journal of Guidance, Control, and Dynamics*, Vol. **17**, No. 6, pp. 1371-1373.

ENNS, D. (1998) Control allocation approaches. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Boston, pp. 98-108.

FOSSSEN, T.I. (1995). *Guidance and Control of Ocean Vehicles*. Chichester: John Wiley & Sons.

FOSSSEN, T.I. (2002). *Marine Control Systems*. Marine Cybernetics AS, Trondheim, Norway.

FRANK, P.M. (1990). Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy – a survey and some new results. *Automatica*, **26**(3). pp. 459-474.

GERTLER, J. (1997). Fault detection and isolation using parity relations. *Control Engineering Practice*, **5**(5). pp. 653-661.

GERTLER, J. (1998). *Fault detection and diagnosis in engineering systems*. New York: Marcel Dekker.

HARKAT, M.F., MOUROT, G. and RAGOT, J. (2000). Sensor Failure Detection of Air Quality Monitoring Network. *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pp. 529-534, Budapest, Hungary.

IKEDA, Y. and Hood, M. (2000). An application of L1 optimisation to control allocation. In *AIAA*

Guidance, Navigation, and Control Conference and Exhibit, Denver.

ISERMANN, R. (1997). Supervision, fault-detection and fault-diagnosis methods – an introduction. *Control Engineering Practice*, 5(5). pp. 639-652.

JAKUBEK, S. and JÖRGL, H.P. (2000). Sensor Fault-diagnosis in a Turbo-charged Combustion Engine. *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pp. 547-552, Budapest, Hungary.

JOHANSEN, T.A., FOSSEN, T.I. and TØNDEL, P. (2002). Efficient Optimal Constrained Control Allocation via Multi-Parametric Programming. *AIAA Journal of Guidance, Control and Dynamics*, submitted.

KASSLIN, M., KANGAS, J. and SIMULA, O. (1992). Process state monitoring using self-organising maps. In: ALEKSANDER and J. TEYLOR, ed. *Artificial Neural Networks 2*. Volume II. pp. 1531-1534. Amsterdam: North-Holland.

KÖPPEN-SELIGER, B. and FRANK, P.M. (1996). Neural network in model-based fault diagnosis. *13th IFAC World Congress*, 7f-02 5, pp. 67-72.

LEEDY, J.Q. (1997). *Real-Time Moment Rate Constrained Control Allocation for Aircraft with a Multiply-Redundant Control Suite*. MSc thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

LINDFORS, I. (1993). Thrust allocation method for the dynamic positioning system. In *10th International Ship Control Systems Symposium (SCSS'93)*, Ottawa, pp. 3.93-3.106.

MANDERS, E.J., NARASIMHAN, S., BISWAS, G. and MOSTERMAN, P.J. (2000). Sensor Fault-diagnosis in a Turbo-charged Combustion Engine. *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pp. 505-510, Budapest, Hungary.

MICHELI, E. (1999). *Supervised and Unsupervised Pattern Recognition*. New York: CRC Press (Tzanakou, editor).

PATTON, R.J. and CHEN, J. (1999). *Robust model-based fault diagnosis for dynamic systems*. Boston: Kluwer Academic Publishers.

PATTON, R.J., FRANK, P.M. and CLARK, R.N. (2000a). *Issues of Fault Diagnosis for Dynamic*

Systems. Great Britain: Springer-Verlag London Limited.

PATTON, R.J., SIMANI, S., DALEY, S. and PIKE, A. (2000b). Fault Diagnosis of a Simulated Model of an Industrial Gas Turbine Prototype Using Identification Techniques. *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pp. 511-516, Budapest, Hungary

PETERSEN, J. and BODSON, M. (1999) Fast Control Allocation Using Spherical Coordinates. *Proceedings of the AIAA Guidance, Control, and Control Conference*, Vol. 2, AIAA, Reston, VA, pp. 1321-1330.

PETERSEN, J. and BODSON, M. (2000) Control Allocation for Systems with Coplanar Controls. *Proceedings of the AIAA Guidance, Control, and Control Conference*, [CD-ROM], AIAA, Reston, VA.

PODDER, T.K. and SARKAR, N. (1999). Fault Tolerant Decomposition of Thruster Forces of an Autonomous Underwater Vehicle. *Proceeding of the 1999 IEEE International Conference on Robotics & Automation*, Detroit, USA.

PODDER, T.K., ANTONELLI, G. and SARKAR, N. (2000). Fault Tolerant Control of an Autonomous Underwater Vehicle Under Thruster Redundancy: Simulations and Experiments. *Proceeding of the 2000 IEEE International Conference on Robotics & Automation*, San Francisco, USA.

POULIEZOS, A.D. and STAVRAKAKIS, G.S. (1989). *Real Time Fault Monitoring of Industrial Processes*. Netherlands: Kluwer Academic Publishers.

RAE, G.J.S. and DUNN, S.E. (1994). On-Line Damage Detection for Autonomous Underwater Vehicles. *IEEE AUV'94*, pp. 383-392.

RUSSEL, E.L., CHIANG, L.H. and BRAATZ, R.D. (2000). *Data-driven Techniques for Fault Detection and Diagnosis in Chemical Processes*. London: Springer-Verlag.

SCHALKOFF, R.J. (1992). *Pattern Recognition: Statistical, Structural and Neural Approaches*. New York: John Wiley & Sons.

SELIGER R. and FRANK, P.M. (1991). Fault Diagnosis by Disturbance Decoupling Nonlinear Observers. *Proc. 30th Conf. on Decision and Control*, pp. 2248-2253, Brighton.

- SHIELDS D.N. (1996). Qualitative Approaches for Fault Diagnosis based on Bilinear Systems. *Proc. 13th World Congress IFAC*, San Francisco, N, pp. 151-156.
- SIMULA O. and KANGAS, J. (1995). Neural Networks for Chemical Engineers. Vol. 6, Computer-Aided Chemical Engineering, Chapter 14, *Process Monitoring and Visualization Using Self-Organising Maps*. Amsterdam: Elsevier.
- SZIGETI, F., BOKOR, J. and EDELMAYER, A. (2002). Input reconstruction by means of system inversion: application to fault detection and isolation. *Prepr. 15th Triennial World Congress of IFAC b'02*, 21st-26th July, Barcelona, Spain.
- TAKAI, M. and URA, T. (1999). Development of a system to diagnose autonomous underwater vehicle. *International Journal of Systems Science*, **30**(9). pp. 981-988.
- THEILLIOL, D., PONSART, J.C. and NOURA, H. (2000). Sensor Fault Diagnosis and Accommodation based on Analytical Redundancy: Application to a Three-Tank System. *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pp. 535-540, Budapest, Hungary.
- TØNDEL, P., JOHANSEN, T.A. and BEMPORAD, A. (2001). An Algorithm for Multi-Parameter Quadratic Programming and Explicit MPC Solutions. In *Proceedings of the IEEE Conference on Decision and Control (CDC'01)*. pp. TuP11-4.
- TØNDEL, P., JOHANSEN, T.A. and BEMPORAD, A. (2003). Evaluation of Piecewise Affine Control via Binary Search Tree, *Automatica*, Vol. 39, pp. 743-749.
- TRYBA, V. and GOSER, K. (1991). Self-organising feature maps for process control in chemistry. In: T. KOHONEN, K. MAKISARA, O. SIMULA and J. KANGAS, ed. *Artificial Neural Networks*. pp. I-847-852. Amsterdam: North-Holland.
- TSUKAMOTO, C.L., YUH, J., CHOI, S.K., LEE, W. and LORENTZ, J. (1999). Experimental study of advanced controllers for a underwater robotic vehicle thruster system. *International Journal of Intelligent Automation and Soft Computing*, **5**, No 1.
- YANG, K.C., YUH, J. and CHOI, S.K. (1998). Experimental Study of Fault-Tolerant System Design for Underwater Robots. *Proceeding of the 1998 IEEE International Conference on Robotics & Automation*, Leuven, Belgium.

YANG, K.C.H., YUH, J. and CHOI, S. K. (1999). Fault-tolerant system design of an autonomous underwater vehicle – ODIN: an experimental study. *International Journal of Systems Science*, **30**(9), pp. 1011-1019.

ZHIRABOK, A.N. (1997). Fault detection and isolation: linear and non-linear systems. *Prepr. IFAC Symp. SAFEPROCESS'97, Hall*, pp. 903-908.

ZHIRABOK, A.N. and SHUMSKY, A.YE. (1987). Functional diagnosis of continuous dynamic systems described by equations whose right-hand side is polynomial. *Automation and Remote Control*, **N8**, pp. 154-164.

ZHIRABOK, A.N. and USOLTSEV, S.A. (2002). Fault diagnosis in non-linear dynamic systems via linear methods. *Prepr. 15th Triennial World Congress of IFAC b'02, 21st-26th July, Barcelona, Spain*.

ZHUANG, Z. and FRANK, P. (1997). Fault detection and isolation with qualitative models. *Safeprocess '97, 2*, pp. 675-680.

Chapter 3: Modelling of ROV & Propulsion System

3.1 Introduction

Experimenting with a real ROV is time consuming and expensive. A dynamic model is useful for simulation purposes and investigation of different control algorithms. In order to investigate different approaches for the design of the fault diagnosis and accommodation system, it is necessary to use realistic models of the vehicle and the propulsion system. In this chapter the development of non-linear dynamic models of thrusters and an ROV in 6 degrees of freedom (DOF) is described.

The chapter is organised as follows: reference frames for the description of the ROV motion are defined in section 3.2, and kinematic equations are given in section 3.3. Dynamic equations of motion are given in section 3.4. Forces and moments acting on the ROV are described in section 3.5, and diagrams for simulation of ROV dynamics and kinematics are given in section 3.6. Section 3.7 discusses the propulsion system and, particularly, thruster models and the thruster control unit. Section 3.8 summarizes concluding remarks, and a list of references, cited in text, is provided in section 3.9.

3.2 Coordinate frames

Considering a ROV as a rigid body with six DOF, two approaches can be used in deriving the equations of motion:

- *The Newton-Euler approach*, related to the forces and moments acting on the body,
- *The Lagrangian approach*, related to the vehicle's total energy.

When analysing the motion of ROV in 6 DOF it is convenient to define two coordinate frames as indicated in Figure 3.1 (Ridao *et al*, 2001; Fossen, 1995). The moving coordinate frame $\{B\}$ is body-fixed and its origin O is usually chosen to coincide with the *centre of gravity* (CG) of the ROV. The motion of the body-fixed frame is described

relative to the Earth-fixed frame $\{E\}$, which can be considered as *inertial*¹, as the effect of the Earth's motion on the low speed ROV is negligible. This suggests that the position and orientation of the ROV should be described relative to the $\{E\}$, while the linear and angular velocities of the ROV should be expressed in $\{B\}$.

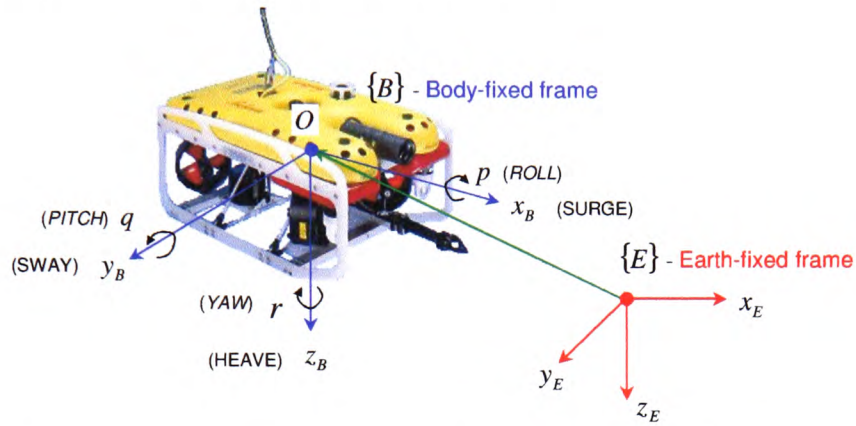


Figure 3.1 Body-fixed and Earth-fixed coordinate frames for ROV.

3.3 Kinematic equations of motion

The kinematic model describes the geometric relationship between Earth-fixed and body-fixed reference frames. These equations are described by the vectors ${}^E\boldsymbol{\eta}$ and ${}^B\mathbf{v}$:

$${}^E\boldsymbol{\eta} = \begin{bmatrix} {}^E\boldsymbol{\eta}_1 \\ {}^E\boldsymbol{\eta}_2 \end{bmatrix} \text{ - position and orientation vector relative to } \{E\}$$

$${}^E\boldsymbol{\eta}_1 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ - position vector relative to } \{E\}$$

¹ An inertial frame of reference is any set of coordinates at rest or moving with constant velocity. The laws of physics are unchanged in any one of the inertial frame of reference. For most purpose, Earth is a good inertial frame of reference in spite of its rotation and revolution. (<http://www.atlans.org/elements/force>)

$${}^E \boldsymbol{\eta}_2 = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} - \text{orientation vector relative to } \{E\}. \text{ This nomenclature is used in the case of}$$

Euler angle attitude representation; for *Euler parameters (Unit quaternion)*

representation vector ${}^E \boldsymbol{\eta}_2$ is replaced by vector ${}^E \mathbf{e}$ (see section 3.3.2); for

Modified Rodrigues parameters representation vector ${}^E \boldsymbol{\eta}_2$ is replaced by

vector ${}^E \boldsymbol{\sigma}$ (see section 3.3.3);

$${}^B \mathbf{v} = \begin{bmatrix} {}^B \mathbf{v}_1 \\ {}^B \mathbf{v}_2 \end{bmatrix} - \text{linear and angular velocity vector relative to } \{B\}$$

$${}^B \mathbf{v}_1 = \begin{bmatrix} u \\ v \\ w \end{bmatrix} - \text{linear velocity vector relative to } \{B\}$$

$${}^B \mathbf{v}_2 = \begin{bmatrix} p \\ q \\ r \end{bmatrix} - \text{angular velocity vector relative to } \{B\}$$

3.3.1 Euler angles representation

For rigid-bodies in 6 DOF the non-linear dynamic equations of motion have a systematic structure that becomes apparent when applying vector notation (Fjellstad and Fossen, 1994b). These equations are usually separated into translational and rotational motion. Position is specified by a 3×1 vector, while various representations of the attitude have been discussed in the literature. The most frequently applied representation is Euler angle representation. The popularity of the Euler angle representation can be explained by its easily understood physical interpretation and the fact that the Euler angles can be measured directly with sensors (gyros). There are also some disadvantages, as mentioned below. Euler angle representation belongs to the family of 3-parameter representations and therefore it must contain singular points (Stuelpnagel, 1964). The orientation of a

rigid-body is usually represented by means of the Euler angles: *roll* (ϕ), *pitch* (θ) and *yaw* (ψ). If vectors ${}^B \mathbf{v}_1$ and ${}^B \mathbf{v}_2$ are known, then it is possible to find time derivatives of vectors ${}^E \dot{\boldsymbol{\eta}}_1$ and ${}^E \dot{\boldsymbol{\eta}}_2$ using linear transformations:

$${}^E \dot{\boldsymbol{\eta}}_1 = \mathbf{J}_1({}^E \boldsymbol{\eta}_2) {}^B \mathbf{v}_1 \quad (3.1)$$

$${}^E \dot{\boldsymbol{\eta}}_2 = \mathbf{J}_2({}^E \boldsymbol{\eta}_2) {}^B \mathbf{v}_2 \quad (3.2)$$

where $\mathbf{J}_1({}^E \boldsymbol{\eta}_2)$ and $\mathbf{J}_2({}^E \boldsymbol{\eta}_2)$ are transformation matrices, defined as:

$$\mathbf{J}_1({}^E \boldsymbol{\eta}_2) = \begin{bmatrix} \cos \psi \cos \theta & -\sin \psi \cos \phi + \cos \psi \sin \theta \sin \phi & \sin \psi \sin \phi + \cos \psi \cos \phi \sin \theta \\ \sin \psi \cos \theta & \cos \psi \cos \phi + \sin \psi \sin \theta \sin \phi & -\cos \psi \sin \phi + \sin \psi \cos \phi \sin \theta \\ -\sin \theta & \cos \theta \sin \phi & \cos \theta \cos \phi \end{bmatrix} \quad (3.3)$$

$$\mathbf{J}_2({}^E \boldsymbol{\eta}_2) = \begin{bmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \frac{\sin \phi}{\cos \theta} & \frac{\cos \phi}{\cos \theta} \end{bmatrix} \quad (3.4)$$

Notice that $\mathbf{J}_2({}^E \boldsymbol{\eta}_2)$ is undefined for a pitch angle $\theta = \pm 90^\circ$ and becomes ill conditioned when the pitch angle approaches $\pm 90^\circ$ (Silpa-Anan, 2001). For ROVs, which operate close to this singularity, it can be a problem. Other disadvantages of the Euler angles representation are extensive computations with trigonometric functions. One solution is to use Euler parameters (see section 3.3.2) or Modified Rodrigues parameters (see section 3.3.3). Another solution is to use the kinematics equations described by two Euler angle representations with different singularities (Fossen, 1995; Fjellstad and Fossen, 1994b). Relations (3.1) & (3.2) can be represented in a compact form:

$$\underbrace{\begin{bmatrix} {}^E \dot{\boldsymbol{\eta}}_1 \\ {}^E \dot{\boldsymbol{\eta}}_2 \end{bmatrix}}_{{}^E \dot{\boldsymbol{\eta}}} = \underbrace{\begin{bmatrix} \mathbf{J}_1({}^E \boldsymbol{\eta}_2) & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{J}_2({}^E \boldsymbol{\eta}_2) \end{bmatrix}}_{\mathbf{J}({}^E \boldsymbol{\eta})} \underbrace{\begin{bmatrix} {}^B \mathbf{v}_1 \\ {}^B \mathbf{v}_2 \end{bmatrix}}_{{}^B \mathbf{v}} \Leftrightarrow {}^E \dot{\boldsymbol{\eta}} = \mathbf{J}({}^E \boldsymbol{\eta}) {}^B \mathbf{v} \quad (3.5)$$

3.3.2 Euler parameters (Unit quaternion) representation

An alternative to the Euler angle representation is a 4-parameter method based on unit quaternions (Fossen, 1995; Fjellstad and Fossen, 1994a; 1994b; 1994c; Chou, 1992).

Definition 3.1 (Quaternion)

A quaternion \mathbf{q} is defined as a complex number

$$\mathbf{q} = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \quad (3.6)$$

where w , x , y and z are real parameters, and \mathbf{i} , \mathbf{j} and \mathbf{k} mutually orthogonal imaginary units.

Hence, quaternions are an extension of complex numbers. Instead of just one imaginary unit \mathbf{i} , there are three different, mutually orthogonal, imaginary units, labelled as \mathbf{i} , \mathbf{j} and \mathbf{k} , defined as

$$\begin{aligned} \mathbf{i} \cdot \mathbf{i} &= -1 \\ \mathbf{j} \cdot \mathbf{j} &= -1 \\ \mathbf{k} \cdot \mathbf{k} &= -1 \end{aligned} \quad (3.7)$$

When two of these units are multiplied together, they behave similarly to cross products of the unit basis vectors:

$$\begin{aligned} \mathbf{i} \cdot \mathbf{j} &= -\mathbf{j} \cdot \mathbf{i} = \mathbf{k} \\ \mathbf{j} \cdot \mathbf{k} &= -\mathbf{k} \cdot \mathbf{j} = \mathbf{i} \\ \mathbf{k} \cdot \mathbf{i} &= -\mathbf{i} \cdot \mathbf{k} = \mathbf{j} \end{aligned} \quad (3.8)$$

The conjugate and magnitude of the quaternion $\mathbf{q} = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ are found in much the same way as complex conjugate and magnitude:

$$\text{Conjugate:} \quad \mathbf{q}' = w - x\mathbf{i} - y\mathbf{j} - z\mathbf{k} \quad (3.9)$$

$$\text{Magnitude:} \quad \|\mathbf{q}\| = \sqrt{\mathbf{q} \cdot \mathbf{q}'} = \sqrt{w^2 + x^2 + y^2 + z^2} \quad (3.10)$$

Quaternions are associative $((\mathbf{q}_1 \cdot \mathbf{q}_2) \cdot \mathbf{q}_3 = \mathbf{q}_1 \cdot (\mathbf{q}_2 \cdot \mathbf{q}_3))$, but not commutative $(\mathbf{q}_1 \cdot \mathbf{q}_2 \neq \mathbf{q}_2 \cdot \mathbf{q}_1)$.

Definition 3.2 (Unit Quaternion)

A quaternion \mathbf{q} is called unit quaternion if $\|\mathbf{q}\| = 1$.

The *inverse* of a quaternion refers to the multiplicative inverse (or $1/\mathbf{q}$) and can be computed by

$$\text{Inverse:} \quad \mathbf{q}^{-1} = \frac{\mathbf{q}'}{\mathbf{q} \cdot \mathbf{q}'} \quad (3.11)$$

For unit quaternion $\|\mathbf{q}\| = 1 \Rightarrow \mathbf{q}^{-1} = \mathbf{q}'$.

Rotation about unit vector $\boldsymbol{\lambda} = [\lambda_1 \ \lambda_2 \ \lambda_3]^T$ by an angle β can be computed using the unit quaternion

$$\mathbf{q} = \begin{bmatrix} \cos \frac{\beta}{2} \\ \boldsymbol{\lambda} \sin \frac{\beta}{2} \end{bmatrix} = \begin{bmatrix} \cos \frac{\beta}{2} \\ \lambda_1 \sin \frac{\beta}{2} \\ \lambda_2 \sin \frac{\beta}{2} \\ \lambda_3 \sin \frac{\beta}{2} \end{bmatrix} \quad (3.12)$$

If position of the point \mathbf{p} in the 3D space is represented by quaternion $\mathbf{P} = [0 \ \mathbf{p}]^T$, then position of the same point after rotation about unit vector $\boldsymbol{\lambda}$ by an angle β is given by quaternion

$$\mathbf{P}_{rot} = \mathbf{q} \cdot \mathbf{P} \cdot \mathbf{q}^{-1} \quad (3.13)$$

Quaternions are very powerful in the case when two consecutive rotations of the object are required. Suppose \mathbf{q}_1 and \mathbf{q}_2 are unit quaternions representing two rotations. Rotation \mathbf{q}_1 should be performed first and then \mathbf{q}_2 . To achieve this, \mathbf{q}_2 is first applied to the result of \mathbf{q}_1 . Then the product is regrouped using associativity and the composite rotation is written in a standard way:

$$\mathbf{q}_2 \cdot (\mathbf{q}_1 \cdot \mathbf{P} \cdot \mathbf{q}_1^{-1}) \cdot \mathbf{q}_2^{-1} = (\mathbf{q}_2 \cdot \mathbf{q}_1) \cdot \mathbf{P} \cdot (\mathbf{q}_1^{-1} \cdot \mathbf{q}_2^{-1}) = (\mathbf{q}_2 \cdot \mathbf{q}_1) \cdot \mathbf{P} \cdot (\mathbf{q}_2 \cdot \mathbf{q}_1)^{-1} \quad (3.14)$$

A matrix product, typical for the Euler angles representation, requires many more operations than a quaternion product. Therefore, considerable computational time can be saved and numerical accuracy can be preserved by using quaternions instead of matrices.

Nomenclature for the quaternions is not consistent in the literature. For example, the unit quaternion in (Fjellstad and Fossen, 1994a; 1994b; 1994c) is defined as $\mathbf{q} = [\eta \ \varepsilon_1 \ \varepsilon_2 \ \varepsilon_3]^T$, while it is defined as $\mathbf{q} = [\varepsilon_1 \ \varepsilon_2 \ \varepsilon_3 \ \eta]^T$ in (Fossen, 1995). In the following the nomenclature proposed in (Fossen, 1995) will be used.

Let β denote the principal angle and let λ denote the principal axis associated with Euler's Theorem (Fossen, 1995), which states that *several rotations about different axis passing through a fixed point are equivalent to a single rotation about an axis passing through this fixed point* (tools.ecn.purdue.edu/~me597k/Homework/hwk03.pdf). The Euler parameters representing the attitude are defined as:

$$\mathbf{e} = \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \eta \end{bmatrix} \quad (3.15)$$

where

$$\boldsymbol{\varepsilon} = [\varepsilon_1 \ \varepsilon_2 \ \varepsilon_3]^T = \lambda \sin \frac{\beta}{2} \quad (3.16)$$

$$\eta = \cos \frac{\beta}{2}, \quad 0 \leq \beta < 2\pi \quad (3.17)$$

The Euler parameters satisfy the unit quaternion constraint, that is:

$$\|\mathbf{e}\| = 1 \Leftrightarrow \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \eta^2 = 1 \quad (3.18)$$

The transformation relating the linear velocity vector ${}^E \dot{\mathbf{q}}_1$ to the velocity ${}^B \mathbf{v}_1$ can be expressed as:

$${}^E \dot{\mathbf{q}}_1 = \mathbf{E}_1 ({}^E \mathbf{e})^B \mathbf{v}_1 \quad (3.19)$$

where

$$\mathbf{E}_1({}^E \mathbf{e}) = \left[(\eta^2 - \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon}) \mathbf{I} + 2\boldsymbol{\varepsilon} \boldsymbol{\varepsilon}^T - 2\eta \mathbf{S}(\boldsymbol{\varepsilon}) \right]^T = \begin{bmatrix} 1 - 2(\varepsilon_1^2 + \varepsilon_3^2) & 2(\varepsilon_1 \varepsilon_2 - \varepsilon_3 \eta) & 2(\varepsilon_1 \varepsilon_3 + \varepsilon_2 \eta) \\ 2(\varepsilon_1 \varepsilon_2 + \varepsilon_3 \eta) & 1 - 2(\varepsilon_1^2 + \varepsilon_3^2) & 2(\varepsilon_2 \varepsilon_3 - \varepsilon_1 \eta) \\ 2(\varepsilon_1 \varepsilon_3 - \varepsilon_2 \eta) & 2(\varepsilon_2 \varepsilon_3 + \varepsilon_1 \eta) & 1 - 2(\varepsilon_1^2 + \varepsilon_2^2) \end{bmatrix} \quad (3.20)$$

Definition 3.3 (Skew-Symmetrical Operator)

Operator $\mathbf{S} : \mathcal{R}^3 \rightarrow \mathcal{R}^3 \times \mathcal{R}^3$ is skew-symmetrical operator defined as

$$\mathbf{S} \left(\begin{bmatrix} a \\ b \\ c \end{bmatrix} \right) \triangleq \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}.$$

An important property of the operator \mathbf{S} is that if $\mathbf{v} = [a \ b \ c]^T$ and $\mathbf{w} = [d \ e \ f]^T$ are two vectors from \mathcal{R}^3 , then the cross product $\mathbf{v} \times \mathbf{w}$ can be expressed as

$$\mathbf{v} \times \mathbf{w} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \times \begin{bmatrix} d \\ e \\ f \end{bmatrix} \triangleq \begin{bmatrix} bf - ce \\ cd - af \\ ae - db \end{bmatrix} = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix} \cdot \begin{bmatrix} d \\ e \\ f \end{bmatrix} = \mathbf{S}(\mathbf{v})\mathbf{w} \quad (3.21)$$

The angular velocity transformation (attitude representation) can be expressed as:

$${}^E \dot{\mathbf{e}} = \mathbf{E}_2({}^E \mathbf{e})^\beta \mathbf{v}_2 \quad (3.22)$$

where

$$\mathbf{E}_2({}^E \mathbf{e}) = \frac{1}{2} \begin{bmatrix} \eta \mathbf{I}_{3 \times 3} + \mathbf{S}(\boldsymbol{\varepsilon}) \\ -\boldsymbol{\varepsilon}^T \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \eta & -\varepsilon_3 & \varepsilon_2 \\ \varepsilon_3 & \eta & -\varepsilon_1 \\ -\varepsilon_2 & \varepsilon_1 & \eta \\ -\varepsilon_1 & -\varepsilon_2 & -\varepsilon_3 \end{bmatrix} \quad (3.23)$$

Hence, the attitude representation with Euler parameters (3.23) is defined for any valid unit quaternion \mathbf{e} and the singularity, typical for any 3-parameter representation, is avoided. Finally, the kinematic equations of motion can be expressed by means of Euler parameters (unit quaternions) as follows

$$\underbrace{\begin{bmatrix} {}^E \dot{\eta}_1 \\ {}^E \dot{\mathbf{e}} \end{bmatrix}}_{{}^E \dot{\eta}} = \underbrace{\begin{bmatrix} \mathbf{E}_1({}^E \mathbf{e}) & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{4 \times 3} & \mathbf{E}_2({}^E \mathbf{e}) \end{bmatrix}}_{\mathbf{J}({}^E \eta)} \underbrace{\begin{bmatrix} {}^B \mathbf{v}_1 \\ {}^B \mathbf{v}_2 \end{bmatrix}}_{{}^B \mathbf{v}} \Leftrightarrow {}^E \dot{\eta} = \mathbf{J}({}^E \eta) {}^B \mathbf{v} \quad (3.24)$$

3.3.3 Modified Rodrigues parameters

The attitude representation in unit quaternions is a 4-parameter representation and, therefore, it is non-minimal (Bošković and Krstić, 1999). Introducing a new set of coordinates (Cayley-Rodrigues parameters) defined as ratio of unit quaternions

$$\rho_i = \frac{\epsilon_i}{\eta}, \quad i = \overline{1,3} \quad (3.25)$$

the constraint (3.18) is eliminated. The vector $\boldsymbol{\rho} = [\rho_1 \ \rho_2 \ \rho_3]^T$ is related to the principal vector $\boldsymbol{\lambda}$ and the principal angle β as

$$\boldsymbol{\rho} = \boldsymbol{\lambda} \tan \frac{\beta}{2} \quad (3.26)$$

The introduction of minimal 3-parameter representations is mainly motivated by their potential advantages in stabilisation and control related problems. Unfortunately, as mentioned in section 3.3.1, all 3-parameter representations contain singularity points. One can see from equation (3.26) that this representation has a singularity at $\beta = \pi$, i.e. the classical Cayley-Rodrigues parameters cannot be used for describing eigenaxis rotations of more than 180° . If (3.25) is replaced by

$$\sigma_i = \frac{\epsilon_i}{1 + \eta}, \quad i = \overline{1,3} \quad (3.27)$$

it can be easily verified that the Modified Rodrigues parameter vector $\boldsymbol{\sigma} = [\sigma_1 \ \sigma_2 \ \sigma_3]$ is related to the principal vector and principal angle as

$$\boldsymbol{\sigma} = \boldsymbol{\lambda} \tan \frac{\beta}{4} \quad (3.28)$$

It is obvious from equation (3.28) that the Modified Rodrigues parameters are superior to any other 3-parameter representation. Firstly, all eigenaxis rotations in the range $0 \leq \beta < 360^\circ$ are well defined. Secondly, unlike other 3-parameter representations (Euler angles or Cayley-Rodrigues parameters), which eliminate an infinity number of possible orientation configurations due to singularity, this parameterisation eliminates only one attitude configuration being singular (namely, $\beta = 360^\circ$ implies $\eta = -1$ and (3.18) implies $\varepsilon = [0 \ 0 \ 0]^T$).

The transformation relating the linear velocity vector ${}^E \dot{\eta}_1$ to the velocity ${}^B \mathbf{v}_1$ is given by

$${}^E \dot{\eta}_1 = \mathbf{J}_1({}^E \boldsymbol{\sigma}) {}^B \mathbf{v}_1 \quad (3.29)$$

where

$$\mathbf{J}_1(\boldsymbol{\sigma}) = \mathbf{I}_{3 \times 3} + \frac{8}{(1 + \|\boldsymbol{\sigma}\|^2)^2} \mathbf{S}(\boldsymbol{\sigma}) \left(\mathbf{S}(\boldsymbol{\sigma}) - \frac{1 - \|\boldsymbol{\sigma}\|^2}{2} \mathbf{I}_{3 \times 3} \right) \quad (3.30)$$

The angular velocity transformation is given by

$${}^E \dot{\boldsymbol{\sigma}} = \mathbf{J}_2({}^E \boldsymbol{\sigma}) {}^B \mathbf{v}_2 \quad (3.31)$$

where

$$\mathbf{J}_2(\boldsymbol{\sigma}) = \frac{1}{2} \left[\mathbf{I}_{3 \times 3} + \boldsymbol{\sigma} \boldsymbol{\sigma}^T - \mathbf{S}(\boldsymbol{\sigma}) - \frac{1 + \boldsymbol{\sigma}^T \boldsymbol{\sigma}}{2} \mathbf{I}_{3 \times 3} \right] \quad (3.32)$$

Finally, the compact representation of the kinematics described with the Modified Rodrigues parameters is given by

$$\underbrace{\begin{bmatrix} {}^E \dot{\eta}_1 \\ {}^E \dot{\boldsymbol{\sigma}} \end{bmatrix}}_{{}^E \dot{\boldsymbol{\eta}}} = \underbrace{\begin{bmatrix} \mathbf{J}_1({}^E \boldsymbol{\sigma}) & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{J}_2({}^E \boldsymbol{\sigma}) \end{bmatrix}}_{\mathbf{J}({}^E \boldsymbol{\eta})} \underbrace{\begin{bmatrix} {}^B \mathbf{v}_1 \\ {}^B \mathbf{v}_2 \end{bmatrix}}_{{}^B \mathbf{v}} \Leftrightarrow {}^E \dot{\boldsymbol{\eta}} = \mathbf{J}({}^E \boldsymbol{\eta}) {}^B \mathbf{v} \quad (3.33)$$

3.3.4 Comments on representation alternatives

In the previous sections Euler angles, Euler parameters and Modified Rodrigues parameters have been suggested as candidates to describe the orientation (attitude) of the ROV. The Euler angle representation is attractive to use, since it is a 3-parameter set corresponding to well-known quantities like the roll, pitch and yaw angle of the vehicle. However, the roll-pitch-yaw representation, like any other 3-parameter representation, has a singularity. In particular, it is not defined for a pitch angle $\theta = \pm 90^\circ$. However, during practical operations with ROVs, the vehicle's orientation of $\theta = \pm 90^\circ$ is not likely to be obtained. This is due to the metacentric restoring forces (Fossen, 1995). Another problem with the Euler angle representation is the so-called "wraparound" problem, which implies that the Euler angles may be integrated up to values outside the normal $\pm 90^\circ$ range of pitch and $\pm 180^\circ$ range of roll and yaw. In order to avoid discontinuities, some normalisation procedure must be performed. One way to avoid singularities and "wraparound" problems is by applying a 4-parameter representation based on Euler parameters. Another advantage of Euler parameters is computational efficiency. The Euler angles are computed by numerical integration of a set of non-linear differential equations. This procedure involves computation of a large number of trigonometric functions, which can be time-consuming. The Modified Rodrigues parameters representation is also computationally effective, but still has one singularity. Although it is not easy to generalise, computational efficiency and accuracy suggests that Euler parameters are the best choice. However, Euler angles are more intuitive and therefore more used.

3.4 Dynamic equations of motion

In the general case, when the origin O of the body-fixed frame does not coincide with CG (see Figure 3.2), general 6 DOF equations of motion can be written in a compact form as (Silpa-Anan, 2001; Bošković and Krstić, 1999; Fossen, 1995):

$$\mathbf{M}_{RB} {}^B \dot{\mathbf{v}} + \mathbf{C}_{RB} ({}^B \mathbf{v}) {}^B \mathbf{v} = {}^B \boldsymbol{\tau}_{RB} \quad (3.34)$$

or in the regular form as

$$m {}^B \dot{\mathbf{v}}_1 + m {}^B \dot{\mathbf{v}}_2 \times {}^B \mathbf{r}_G + m {}^B \mathbf{v}_2 \times {}^B \mathbf{v}_1 + m {}^B \mathbf{v}_2 \times ({}^B \mathbf{v}_2 \times {}^B \mathbf{r}_G) = {}^B \boldsymbol{\tau}_1 \quad (3.35)$$

$$\mathbf{I}_0 {}^B \dot{\mathbf{v}}_2 + m {}^B \mathbf{r}_G \times {}^B \dot{\mathbf{v}}_1 + {}^B \mathbf{v}_2 \times (\mathbf{I}_0 {}^B \mathbf{v}_2) + m {}^B \mathbf{r}_G \times ({}^B \mathbf{v}_2 \times {}^B \mathbf{v}_1) = {}^B \boldsymbol{\tau}_2 \quad (3.36)$$

where

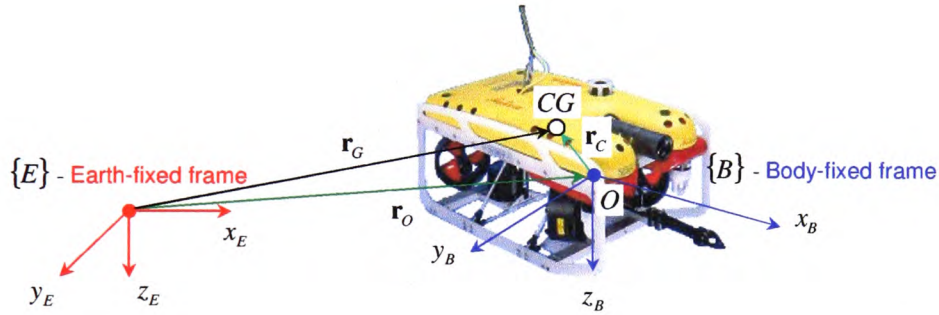


Figure 3.2 General case: centre of gravity CG does not coincide with the origin O .

\mathbf{M}_{RB} is a rigid-body inertia matrix ($\mathbf{M}_{RB} = \mathbf{M}_{RB}^T > 0$, $\dot{\mathbf{M}}_{RB} = 0$)

$$\mathbf{M}_{RB} {}^B \dot{\mathbf{v}} = \begin{bmatrix} m {}^B \dot{\mathbf{v}}_1 + m {}^B \dot{\mathbf{v}}_2 \times {}^B \mathbf{r}_G \\ \mathbf{I}_0 {}^B \dot{\mathbf{v}}_2 + m {}^B \mathbf{r}_G \times {}^B \dot{\mathbf{v}}_1 \end{bmatrix} \quad (3.37)$$

$$\mathbf{M}_{RB} = \begin{bmatrix} m \mathbf{I}_{3 \times 3} & -m \mathbf{S}({}^B \mathbf{r}_G) \\ m \mathbf{S}({}^B \mathbf{r}_G) & \mathbf{I}_0 \end{bmatrix} = \begin{bmatrix} m & 0 & 0 & 0 & mz_G & -my_G \\ 0 & m & 0 & -mz_G & 0 & mx_G \\ 0 & 0 & m & my_G & -mx_G & 0 \\ 0 & -mz_G & my_G & I_x & -I_{xy} & -I_{xz} \\ mz_G & 0 & -mx_G & -I_{yx} & I_y & -I_{yz} \\ -my_G & mx_G & 0 & -I_{zx} & -I_{zy} & I_z \end{bmatrix} \quad (3.38)$$

m is mass of ROV,

${}^B \mathbf{r}_G$ is *centre of gravity CG*, with respect to the body-fixed frame $\{B\}$

$${}^B \mathbf{r}_G = \begin{bmatrix} x_G \\ y_G \\ z_G \end{bmatrix} \quad (3.39)$$

\mathbf{I}_0 is *inertia tensor* of the ROV, with respect to the body-fixed frame $\{B\}$

$$\mathbf{I}_0 = \Delta \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{yx} & I_y & -I_{yz} \\ -I_{zx} & -I_{zy} & I_z \end{bmatrix}, \quad \mathbf{I}_0 = \mathbf{I}_0^T > \mathbf{0} \quad (3.40)$$

I_x , I_y and I_z are the *moments of inertia* about the X_B , Y_B and Z_B -axes

$$I_x = \int_V (y^2 + z^2) \rho_A dV \quad (3.41)$$

$$I_y = \int_V (x^2 + z^2) \rho_A dV \quad (3.42)$$

$$I_z = \int_V (x^2 + y^2) \rho_A dV \quad (3.43)$$

$I_{xy} = I_{yx}$, $I_{xz} = I_{zx}$ and $I_{yz} = I_{zy}$ are the *products of inertia*

$$I_{xy} = \int_V xy \rho_A dV = \int_V yx \rho_A dV = I_{yx} \quad (3.44)$$

$$I_{xz} = \int_V xz \rho_A dV = \int_V zx \rho_A dV = I_{zx} \quad (3.45)$$

$$I_{yz} = \int_V yz \rho_A dV = \int_V zy \rho_A dV = I_{zy} \quad (3.46)$$

where ρ_A is the mass density of the body.

It is possible to simplify the general dynamic equations of motion by applying the following theorem (Fossen, 1995):

Theorem 3.1 (Parallel Axes Theorem)

The inertia tensor \mathbf{I}_0 about an arbitrary origin O can be calculated from the relation

$$\mathbf{I}_0 = \mathbf{I}_C - m \mathbf{S}({}^B \mathbf{r}_G) \mathbf{S}({}^B \mathbf{r}_G) = \mathbf{I}_C - m ({}^B \mathbf{r}_G {}^B \mathbf{r}_G^T - {}^B \mathbf{r}_G^T {}^B \mathbf{r}_G \mathbf{I}_{3 \times 3}) \quad (3.47)$$

where $\mathbf{I}_{3 \times 3}$ is the identity matrix and \mathbf{I}_C is the inertia tensor about the centre of gravity CG .

The first simplification can be obtained if origin O coincides with the CG . In that case ${}^B \mathbf{r}_G = [0 \ 0 \ 0]^T$ and $\mathbf{I}_0 = \mathbf{I}_C$. The second simplification can be obtained if the body axes coincide with the principal axes of inertia or the longitudinal, lateral and normal symmetry axes of the ROV. In that case the origin O can be chosen such that the inertia tensor \mathbf{I}_0 is a diagonal matrix.

$\mathbf{C}_{RB}({}^B \mathbf{v})$ is a skew-symmetrical parameterisation of the rigid-body Coriolis and centripetal matrix ($\mathbf{C}_{RB}({}^B \mathbf{v}) = -\mathbf{C}_{RB}^T({}^B \mathbf{v}) > \mathbf{0}$)

$$\mathbf{C}_{RB}({}^B \mathbf{v}) {}^B \mathbf{v} = \begin{bmatrix} m {}^B \mathbf{v}_2 \times {}^B \mathbf{v}_1 + m {}^B \mathbf{v}_2 \times ({}^B \mathbf{v}_2 \times {}^B \mathbf{r}_G) \\ {}^B \mathbf{v}_2 \times (\mathbf{I}_0 {}^B \mathbf{v}_2) + m {}^B \mathbf{r}_G \times ({}^B \mathbf{v}_2 \times {}^B \mathbf{v}_1) \end{bmatrix} \quad (3.48)$$

There are a large numbers of parameterisations for the $\mathbf{C}_{RB}({}^B \mathbf{v})$ matrix, which satisfy condition (3.48). Two common used parameterisations, proposed in (Fossen, 1995) are given below:

$$\mathbf{C}_{RB_1}({}^B \mathbf{v}) = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{3 \times 3} & -m \mathbf{S}({}^B \mathbf{v}_1) - m \mathbf{S}({}^B \mathbf{v}_2) {}^B \mathbf{r}_G \\ -m \mathbf{S}({}^B \mathbf{v}_1) - m \mathbf{S}({}^B \mathbf{v}_2) {}^B \mathbf{r}_G & m \mathbf{S}({}^B \mathbf{v}_1) {}^B \mathbf{r}_G - \mathbf{S}(\mathbf{I}_0 {}^B \mathbf{v}_2) \end{bmatrix} \quad (3.49)$$

$$\mathbf{A}_{11} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{A}_{12} = \mathbf{A}_{21} = \begin{bmatrix} 0 & m(w + y_G p - x_G q) & m(-v + z_G p - x_G r) \\ -m(w + y_G p - x_G q) & 0 & m(u + z_G q - y_G r) \\ -m(-v + z_G p - x_G r) & -m(u + z_G q - y_G r) & 0 \end{bmatrix}$$

$$\mathbf{A}_{22} = \begin{bmatrix} 0 & -my_G u + mx_G v - I_{xx} p - I_{xy} q + I_z r & -mz_G u + mx_G w + I_{yx} p - I_y q + I_{yz} r \\ -(-my_G u + mx_G v - I_{xx} p - I_{xy} q + I_z r) & 0 & -mz_G v + my_G w + I_{xp} - I_{xy} q - I_{xz} r \\ -(-mz_G u + mx_G w + I_{yx} p - I_y q + I_{yz} r) & -(-mz_G v + my_G w + I_{xp} - I_{xy} q - I_{xz} r) & 0 \end{bmatrix}$$

$$\mathbf{C}_{RB_1}({}^B \mathbf{v}) = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{3 \times 3} & -m\mathbf{S}({}^B \mathbf{v}_1) - m\mathbf{S}({}^B \mathbf{v}_2)\mathbf{S}({}^B \mathbf{r}_G) \\ -m\mathbf{S}({}^B \mathbf{v}_1) + m\mathbf{S}({}^B \mathbf{r}_G)\mathbf{S}({}^B \mathbf{v}_2) & -\mathbf{S}(\mathbf{I}_0 {}^B \mathbf{v}_2) \end{bmatrix} \quad (3.50)$$

$$\mathbf{B}_{11} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{B}_{12} = \begin{bmatrix} m(z_G r + y_G q) & m(w - x_G q) & m(-v - x_G r) \\ -m(w + y_G p) & m(z_G r + x_G p) & m(u - y_G r) \\ -m(-v + z_G p) & -m(u + z_G q) & m(x_G p + y_G q) \end{bmatrix}$$

$$\mathbf{B}_{21} = -\mathbf{B}_{12}^T = \begin{bmatrix} -m(z_G r + y_G q) & m(w + y_G p) & m(-v + z_G p) \\ -m(w - x_G q) & -m(z_G r + x_G p) & m(u + z_G q) \\ -m(-v - x_G r) & -m(u - y_G r) & -m(x_G p + y_G q) \end{bmatrix}$$

$$\mathbf{B}_{22} = \begin{bmatrix} 0 & -I_{xx} p - I_{xy} q + I_z r & I_{yx} p - I_y q + I_{yz} r \\ -(-I_{xx} p - I_{xy} q + I_z r) & 0 & I_{xp} - I_{xy} q - I_{xz} r \\ -(I_{yx} p - I_y q + I_{yz} r) & -(I_{xp} - I_{xy} q - I_{xz} r) & 0 \end{bmatrix}$$

$$\begin{aligned} \mathbf{C}_{RB_1}({}^B \mathbf{v}) {}^B \mathbf{v} &= \mathbf{C}_{RB_2}({}^B \mathbf{v}) {}^B \mathbf{v} = \begin{bmatrix} m {}^B \mathbf{v}_2 \times {}^B \mathbf{v}_1 + m {}^B \mathbf{v}_2 \times ({}^B \mathbf{v}_2 \times {}^B \mathbf{r}_G) \\ {}^B \mathbf{v}_2 \times (\mathbf{I}_0 {}^B \mathbf{v}_2) + m {}^B \mathbf{r}_G \times ({}^B \mathbf{v}_2 \times {}^B \mathbf{v}_1) \end{bmatrix} = \\ &= \begin{bmatrix} mq(w + y_G p - x_G q) + mr(-v + z_G p - x_G r) \\ mp(-w - y_G p + x_G q) + mr(u + z_G q - y_G r) \\ mp(v - z_G p - x_G r) + mq(-u - z_G q + y_G r) \\ m(vy_G p + wz_G p - qy_G u - rz_G u) - qI_{xx} p - I_{xy} q^2 + qI_z r + rI_{yx} p - rI_y q + I_{yz} r^2 \\ m(ux_G q + wz_G q - px_G v - rz_G v) + I_{xx} p^2 + pI_{xy} q - pI_z r + rI_{xp} - rI_{xy} q - I_{xz} r^2 \\ m(ux_G r + vy_G r - px_G w - qy_G w) - I_{yx} p^2 + pI_y q - pI_{yz} r - qI_{xp} + qI_{xz} r + I_{xy} q^2 \end{bmatrix} \end{aligned}$$

${}^B \boldsymbol{\tau}_{RB}$ is a generalised vector of external forces and moments (including control and hydrodynamic forces and moments)

This vector will be described in more detail in the following section.

3.5 Forces and moments acting on ROV

The general dynamic equation of motion of the ROV in 6 DOF in the body-fixed frame is given in compact form by

$$\underbrace{(\mathbf{M}_{RB} + \mathbf{M}_A)}_{\mathbf{M}} \dot{\mathbf{v}} + \underbrace{(\mathbf{C}_{RB}(\mathbf{v}) + \mathbf{C}_A(\mathbf{v}))}_{\mathbf{C}(\mathbf{v})} \mathbf{v} + \mathbf{D}(\mathbf{v}) \mathbf{v} + \mathbf{g}^E(\boldsymbol{\eta}) = \mathbf{\tau}_E + \mathbf{\tau} \quad (3.51)$$

where

\mathbf{M} is inertia matrix (including added mass)

\mathbf{M}_{RB} is rigid-body inertia matrix

\mathbf{M}_A is added-mass matrix

$\mathbf{C}(\mathbf{v})$ is Coriolis and centripetal matrix (including added mass)

$\mathbf{C}_{RB}(\mathbf{v})$ is rigid body Coriolis and centripetal matrix

$\mathbf{C}_A(\mathbf{v})$ is added-mass Coriolis and centripetal matrix

$\mathbf{D}(\mathbf{v})$ is total hydrodynamic damping matrix

$\mathbf{g}^E(\boldsymbol{\eta})$ is vector of restoring (gravitational and buoyant) forces and moments

$\mathbf{\tau}_E$ is vector of environmental forces and moments

$\mathbf{\tau}$ is vector of propulsion forces and moments (exerted by the thrusters)

From (3.34) & (3.51) the following expression for the generalised vector of external forces and moments can be found:

$$\mathbf{\tau}_{RB} = \mathbf{\tau}_E + \mathbf{\tau} - \mathbf{M}_A \dot{\mathbf{v}} - \mathbf{C}_A(\mathbf{v}) \mathbf{v} - \mathbf{D}(\mathbf{v}) \mathbf{v} - \mathbf{g}^E(\boldsymbol{\eta}) \quad (3.52)$$

Various forces and moments act on the ROV whilst moving through a fluid (water):

1. Hydrodynamic rigid-body-like added mass forces and moments $-\mathbf{M}_A \dot{\mathbf{v}}$,
2. Hydrodynamic Coriolis-like added mass forces and moments $-\mathbf{C}_A(\mathbf{v}) \mathbf{v}$,

3. Hydrodynamic damping and lift forces and moments $-\mathbf{D}({}^B\mathbf{v})^B\mathbf{v}$,
4. Restoring (gravitational and buoyant) forces and moments $-{}^B\mathbf{g}({}^E\boldsymbol{\eta})$,
5. Environmental forces and moments ${}^B\boldsymbol{\tau}_E$.
6. Propulsion forces and moments (exerted by the thrusters) ${}^B\boldsymbol{\tau}$,

\mathbf{M}_A (added-mass matrix) and $\mathbf{C}_A({}^B\mathbf{v})$ (added-mass Coriolis and centripetal matrix)

The concept of added mass is usually misunderstood to be a finite amount of water connected to the vehicle such that the vehicle and the fluid represents a new system with mass larger than the original vehicle. This is *not* true, since the vehicle motion will force the whole fluid to oscillate with different fluid particle amplitudes in phase with the forced harmonic motion of the vehicle (Fossen, 1995). However, the amplitudes will decay far away from the body and may therefore be negligible. Added (virtual) mass should be understood as pressure-induced forces and moments due to a forced harmonic motion of the body, which are proportional to the acceleration of the body. Consequently, the added mass forces and the acceleration will be 180° out of phase to the forced harmonic motion. If the ROV moves at low-speed and has (almost) three planes of symmetry, then the following expressions for \mathbf{M}_A and $\mathbf{C}_A({}^B\mathbf{v})$ are obtained (Fossen, 1995):

$$\mathbf{M}_A = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} = - \left[\begin{array}{ccc|ccc} \frac{\partial X}{\partial \ddot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\partial Y}{\partial \ddot{v}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\partial Z}{\partial \ddot{w}} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & \frac{\partial K}{\partial \dot{p}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\partial M}{\partial \dot{q}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\partial N}{\partial \dot{r}} \end{array} \right] = - \left[\begin{array}{ccc|ccc} X_i & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_i & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_i & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & K_p & 0 & 0 \\ 0 & 0 & 0 & 0 & M_q & 0 \\ 0 & 0 & 0 & 0 & 0 & N_r \end{array} \right] \quad (3.53)$$

$$\begin{aligned}
C_A({}^B \mathbf{v}) &= \begin{bmatrix} \mathbf{0}_{3 \times 3} & -\mathbf{S}(\mathbf{M}_{11}{}^B \mathbf{v}_1 + \mathbf{M}_{12}{}^B \mathbf{v}_2) \\ -\mathbf{S}(\mathbf{M}_{11}{}^B \mathbf{v}_1 + \mathbf{M}_{21}{}^B \mathbf{v}_2) & -\mathbf{S}(\mathbf{M}_{21}{}^B \mathbf{v}_1 + \mathbf{M}_{22}{}^B \mathbf{v}_2) \end{bmatrix} = \\
&= \begin{bmatrix} 0 & 0 & 0 & 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v \\ 0 & 0 & 0 & Z_{\dot{w}}w & 0 & -X_{\dot{u}}u \\ 0 & 0 & 0 & -Y_{\dot{v}}v & X_{\dot{u}}u & 0 \\ 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v & 0 & -N_{\dot{r}}r & M_{\dot{q}}q \\ Z_{\dot{w}}w & 0 & -X_{\dot{u}}u & N_{\dot{r}}r & 0 & K_{\dot{p}}p \\ -Y_{\dot{v}}v & X_{\dot{u}}u & 0 & -M_{\dot{q}}q & -K_{\dot{p}}p & 0 \end{bmatrix} \quad (3.54)
\end{aligned}$$

The notation of SNAME (the *Society of Naval Architects and Marine Engineers*, 1950) is used in expressions (3.53) & (3.54). For instance, the hydrodynamic added mass force Y_A along the y -axis due to an acceleration \ddot{u} in the x -direction is written as

$$Y_A = Y_{\ddot{u}}\ddot{u} \quad (3.55)$$

where

$$Y_{\ddot{u}} = \frac{\Delta Y}{\Delta \ddot{u}} \quad (3.56)$$

The diagonal structure is highly attractive since off-diagonal elements are difficult to determine from experiments as well as theory. In practice, the diagonal approximation is found to be quite good for many applications.

$\mathbf{D}({}^B \mathbf{v})$ (total hydrodynamic damping matrix)

In the general case, the damping of a ROV moving in 6 DOF at high-speed is highly non-linear and coupled. One rough approximation considers the ROV is performing a non-coupled motion, where terms higher than second order are negligible (Fossen, 1995). This suggests a diagonal structure of $\mathbf{D}({}^B \mathbf{v})$ with only linear and quadratic damping terms on the diagonal:

$$\mathbf{D}({}^B \mathbf{v}) = - \begin{bmatrix} X_u + X_{u|u}|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_v + Y_{v|v}|v| & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_w + Z_{w|w}|w| & 0 & 0 & 0 \\ 0 & 0 & 0 & K_p + K_{p|p}|p| & 0 & 0 \\ 0 & 0 & 0 & 0 & M_q + M_{q|q}|q| & 0 \\ 0 & 0 & 0 & 0 & 0 & N_r + N_{r|r}|r| \end{bmatrix} \quad (3.57)$$

${}^B \mathbf{g}({}^E \boldsymbol{\eta})$ (restoring (gravitational and buoyant) forces and moments)

The gravitational force \mathbf{f}_G is induced by the weight, W of the ROV and acts through the centre of gravity ${}^B \mathbf{r}_G = {}^B [x_G \ y_G \ z_G]^T$ of the ROV. The buoyant force \mathbf{f}_B is induced by the buoyancy, B and acts through the centre of buoyancy ${}^B \mathbf{r}_B = {}^B [x_B \ y_B \ z_B]^T$ of the ROV. Restoring forces in the Earth-fixed frame $\{E\}$ are shown in Table 3.1, where

$W = mg$ is weight of the ROV,

m is mass of the ROV,

g is the acceleration due to gravity,

$B = \rho g \nabla$ is buoyant force,

ρ is the fluid (water) density,

∇ is the volume of fluid (water) displaced by the ROV.

Force	Earth-fixed frame $\{E\}$
Gravitational force	${}^E \mathbf{f}_G = \begin{bmatrix} 0 \\ 0 \\ W \end{bmatrix}$
Buoyant force	${}^E \mathbf{f}_B = \begin{bmatrix} 0 \\ 0 \\ -B \end{bmatrix}$

Table 3.1 Restoring forces in the Earth-fixed frame.

Vector of restoring forces and moments can be transformed to the body-fixed frame representation using equations given in Table 3.2 (Euler angles representation), Table 3.3 (Euler parameters representation) and Table 3.4 (representation with Modified Rodrigues parameters).

Force	Body-fixed frame {B}
Gravitational force	${}^B \mathbf{f}_G({}^E \boldsymbol{\eta}_2) = \mathbf{J}_1^{-1}({}^E \boldsymbol{\eta}_2) {}^E \mathbf{f}_G$
Buoyant force	${}^B \mathbf{f}_B({}^E \boldsymbol{\eta}_2) = \mathbf{J}_1^{-1}({}^E \boldsymbol{\eta}_2) {}^E \mathbf{f}_B$
Restoring forces and moments	${}^B \mathbf{g}({}^E \boldsymbol{\eta}) = - \begin{bmatrix} {}^B \mathbf{f}_G({}^E \boldsymbol{\eta}_2) + {}^B \mathbf{f}_B({}^E \boldsymbol{\eta}_2) \\ {}^B \mathbf{r}_G \times {}^B \mathbf{f}_G({}^E \boldsymbol{\eta}_2) + {}^B \mathbf{r}_B \times {}^B \mathbf{f}_B({}^E \boldsymbol{\eta}_2) \end{bmatrix} =$ $= \begin{bmatrix} (W - B) \sin \theta \\ -(W - B) \cos \theta \sin \phi \\ -(W - B) \cos \theta \cos \phi \\ -(y_G W - y_B B) \cos \theta \cos \phi + (z_G W - z_B B) \cos \theta \sin \phi \\ (z_G W - z_B B) \sin \theta + (x_G W - x_B B) \cos \theta \cos \phi \\ -(x_G W - x_B B) \cos \theta \sin \phi - (y_G W - y_B B) \sin \theta \end{bmatrix}$

Table 3.2 Restoring forces and moments in the body-fixed frame for the attitude representation with Euler angles (see section 3.3.1).

Force	Body-fixed frame {B}
Gravitational force	${}^B \mathbf{f}_G({}^E \mathbf{e}) = \mathbf{E}_1^{-1}({}^E \mathbf{e}) {}^E \mathbf{f}_G$
Buoyant force	${}^B \mathbf{f}_B({}^E \mathbf{e}) = \mathbf{E}_1^{-1}({}^E \mathbf{e}) {}^E \mathbf{f}_B$
Restoring forces and moments	${}^B \mathbf{g}({}^E \boldsymbol{\eta}) = - \begin{bmatrix} {}^B \mathbf{f}_G({}^E \mathbf{e}) + {}^B \mathbf{f}_B({}^E \mathbf{e}) \\ {}^B \mathbf{r}_G \times {}^B \mathbf{f}_G({}^E \mathbf{e}) + {}^B \mathbf{r}_B \times {}^B \mathbf{f}_B({}^E \mathbf{e}) \end{bmatrix} =$ $= \begin{bmatrix} 2(\eta \varepsilon_2 - \varepsilon_1 \varepsilon_3)(W - B) \\ -2(\eta \varepsilon_1 + \varepsilon_2 \varepsilon_3)(W - B) \\ (-\eta^2 + \varepsilon_1^2 + \varepsilon_2^2 - \varepsilon_3^2)(W - B) \\ (-\eta^2 + \varepsilon_1^2 + \varepsilon_2^2 - \varepsilon_3^2)(y_G W - y_B B) + 2(\eta \varepsilon_1 + \varepsilon_2 \varepsilon_3)(z_G W - z_B B) \\ -(-\eta^2 + \varepsilon_1^2 + \varepsilon_2^2 - \varepsilon_3^2)(x_G W - x_B B) + 2(\eta \varepsilon_2 - \varepsilon_1 \varepsilon_3)(z_G W - z_B B) \\ -2(\eta \varepsilon_1 + \varepsilon_2 \varepsilon_3)(x_G W - x_B B) - 2(\eta \varepsilon_2 - \varepsilon_1 \varepsilon_3)(y_G W - y_B B) \end{bmatrix}$

Table 3.3 Restoring forces and moments in the body-fixed frame for the attitude representation with Euler parameters (see section 3.3.2).

Force	Body-fixed frame {B}
Gravitational force	${}^B \mathbf{f}_G({}^E \boldsymbol{\sigma}) = \mathbf{J}_1^{-1}({}^E \boldsymbol{\sigma}) {}^E \mathbf{f}_G$
Buoyant force	${}^B \mathbf{f}_B({}^E \boldsymbol{\sigma}) = \mathbf{J}_1^{-1}({}^E \boldsymbol{\sigma}) {}^E \mathbf{f}_B$
Restoring forces and moments	${}^B \mathbf{g}({}^E \boldsymbol{\eta}) = - \begin{bmatrix} {}^B \mathbf{f}_G({}^E \boldsymbol{\sigma}) + {}^B \mathbf{f}_B({}^E \boldsymbol{\sigma}) \\ {}^B \mathbf{r}_G \times {}^B \mathbf{f}_G({}^E \boldsymbol{\sigma}) + {}^B \mathbf{r}_B \times {}^B \mathbf{f}_B({}^E \boldsymbol{\sigma}) \end{bmatrix}$

Table 3.4 Restoring forces and moments in the body-fixed frame for the attitude representation with Modified Rodrigues parameters (see section 3.3.3).

A neutrally buoyant ROV satisfies the condition:

$$W = B \quad (3.58)$$

${}^B\tau_E$ (environmental forces and moments)

The main environmental forces to be considered for ocean vehicles are:

- surface waves,
- wind,
- ocean currents.

For the ROV moving at a depth more than 20 meters, the effects of surface waves can be neglected. Moreover, the wind has an effect only when the ROV is moving on the surface. Therefore, the only perturbations to be considered are those due to ocean currents.

${}^B\tau$ (propulsion forces and moments (exerted by the thrusters))

In the general case an ROV has p thrusters ${}^1Th, {}^2Th, \dots, {}^pTh$. Each thruster ${}^iTh, i = \overline{1, p}$ exerts thrust (force) iT and torque (moment) iQ_e (Figure 3.3). Depending on propeller spin direction, vectors iT and iQ_e have the same direction (for clockwise rotation looking from the back of the propellers) or opposite direction (for counter clockwise rotation). The thrust iT also generates moment ${}^iQ_r = {}^i\mathbf{r} \times {}^iT$, so that the total moment vector exerted by the thruster is given by ${}^iQ = {}^iQ_e + {}^iQ_r$. Contributions of each thruster are summed together to form vector of propulsion forces and moments ${}^B\tau$:

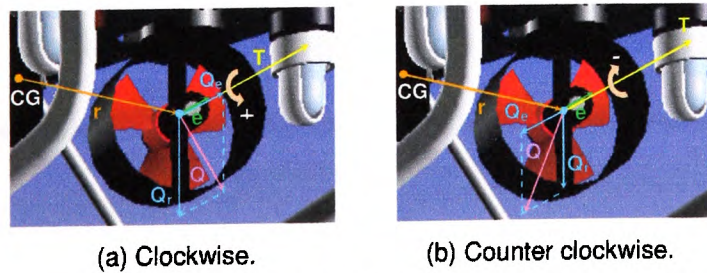


Figure 3.3 Thrust and torque, exerted by a thruster, for two possible propeller spin directions.

$${}^B \boldsymbol{\tau} = \begin{bmatrix} \mathbf{T} \\ \mathbf{Q} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^p {}^i \mathbf{T} \\ \sum_{i=1}^p {}^i \mathbf{Q} \end{bmatrix} \quad (3.59)$$

More information about thruster configuration and propulsion system can be found in section 3.7.

3.6 Simulation diagrams for ROV dynamics and kinematics

In order to construct the simulation diagram, the first step is to find the expression for ${}^B \dot{\mathbf{v}}$ from (3.51):

$${}^B \dot{\mathbf{v}} = \mathbf{M}^{-1} ({}^B \boldsymbol{\tau} + {}^B \boldsymbol{\tau}_E - (\mathbf{C}({}^B \mathbf{v}) + \mathbf{D}({}^B \mathbf{v})) {}^B \mathbf{v} - {}^B \mathbf{g}({}^E \boldsymbol{\eta})) \quad (3.60)$$

Depending on the choice of the attitude representations, there are three alternatives to represent ROV kinematics. The first alternative is shown in Figure 3.4 (Euler angles attitude representation). The second alternative (Euler parameters attitude representation) is shown in Figure 3.5. Finally, Figure 3.6 displays a simulation diagram for attitude representation with Modified Rodrigues parameters.

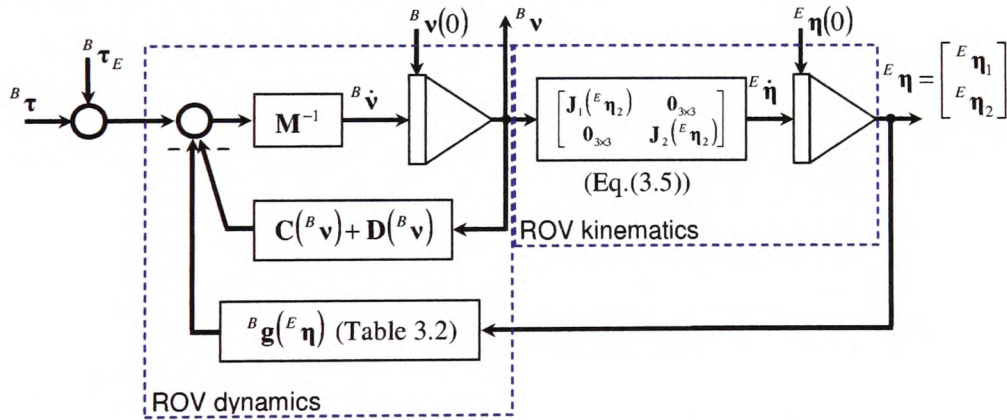


Figure 3.4 Simulation diagram for ROV dynamics and kinematics, where the attitude is represented with Euler angles.

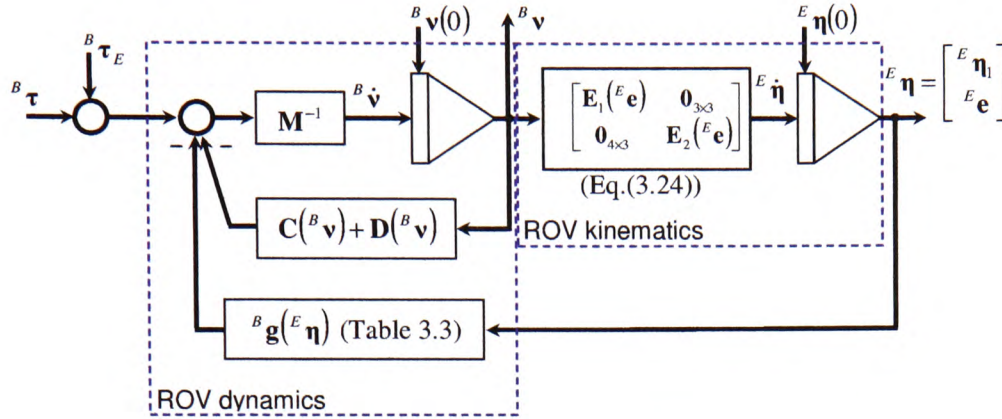


Figure 3.5 Simulation diagram for ROV dynamics and kinematics, where the attitude is represented with Euler parameters.

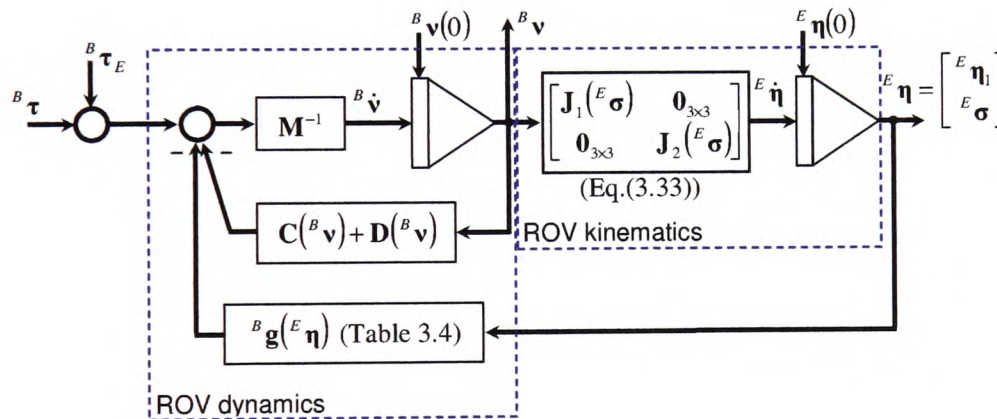


Figure 3.6 Simulation diagram for ROV dynamics and kinematics, where the attitude is represented with Modified Rodrigues parameters.

These simulation diagrams can be used to simulate ROV dynamics using MATLAB & Simulink. In particular, the diagram shown in Figure 3.4 is transformed to functionally equivalent *S*-function and implemented in the ROV simulator as a Simulink block with an associated dialog box for setting of model parameters.

3.7 *Propulsion system*

This section discusses propulsion system of the open-frame underwater vehicle. Since the majority of open-frame underwater vehicles use thrusters for propulsion, the focus will be on propeller thrust and torque modelling.

3.7.1 Thruster configuration

As stated in Chapter 1, two underwater vehicles (FALCON, SeaEye Marine Ltd. and URIS, University of Girona, see Figure 3.7.) with different thruster configurations are used to demonstrate the performance of the FDAS, described in Chapter 5. More technical details about the vehicles can be found in Appendix A. In order to examine the performance of the FDAS, which require that control system for motion in horizontal plane is overactuated, the original thruster configuration of URIS, with two horizontal and two vertical thrusters, is transformed into a configuration with four horizontal thrusters, without any vertical thruster. This modification was possible, because the tank for test trials at University of Girona was very shallow and there was no space and need for motion in vertical plane. A three-dimensional view of the FALCON moving in a virtual underwater world is shown in Figure 3.7 (a) – top. This picture is taken from the ROV simulator, as well as the picture of URIS, shown in Figure 3.7 (b) – top. Plan view of the vehicles with corresponding configuration of the horizontal thrusters is shown in bottom part of Figure 3.7. The origin of the body-fixed reference frame $\{B\}$ is chosen to coincide with the CG. The axes are chosen to coincide with the principal axes of inertia and they are defined as:

- x_B - *longitudinal* axes (directed to front side),
- y_B - *transversal* axes (directed to starboard),
- z_B - *normal* axes (directed from top to bottom, perpendicular to the paper).

The FALCON has four horizontal thrusters, denoted as ${}^iHT, i = \overline{1,4}$ and one vertical 1VT (not shown in Figure 3.7 (a) - bottom). The URIS has only four horizontal thrusters, denoted in the same way. In the following, the discussion is concentrated on horizontal thrusters, although the same principle is valid for vertical thrusters.

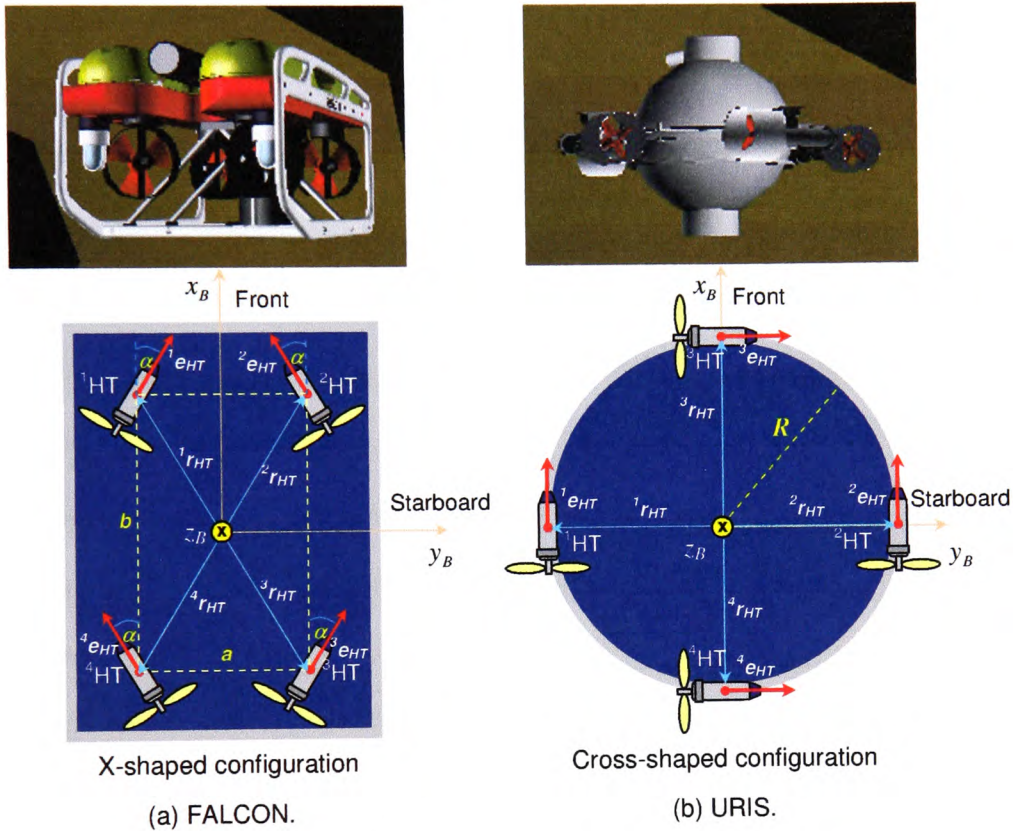


Figure 3.7 Two common configurations of the horizontal thrusters.

The nomenclature for vertical thrusters can be derived by replacing HT with VT in corresponding symbols for horizontal thrusters. In addition, the nomenclature iTh refers to any thruster, while iHT (iVT) refers to a horizontal (vertical) thruster.

The thruster iHT exerts thrust (force) ${}^i T_{HT}$ and torque (moment) ${}^i Q_{eHT}$ (see Figure 3.3).

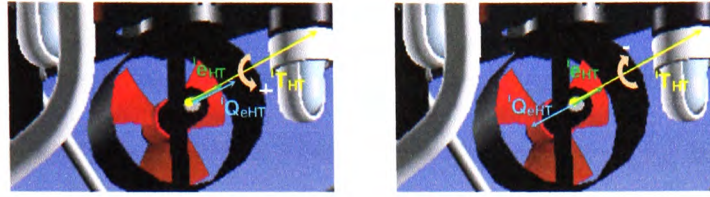
The position vector ${}^i r_{HT} = [{}^i r_x \quad {}^i r_y \quad {}^i r_z]^T$ determines the position of the point of attack

of the force ${}^i\mathbf{T}_{HT}$, relative to the $\{B\}$. The force ${}^i\mathbf{T}_{HT}$ also generates the moment ${}^i\mathbf{Q}_{HT} = {}^i\mathbf{r}_{HT} \times {}^i\mathbf{T}_{HT}$, so the total moment vector exerted by the thruster iHT is given by ${}^i\mathbf{Q}_{HT} = {}^i\mathbf{Q}_{eHT} + {}^i\mathbf{Q}_{rHT}$. The orientation of the thruster iHT relative to the $\{B\}$ is defined by the unit vector ${}^i\mathbf{e}_{HT} = [{}^ie_x \quad {}^ie_y \quad {}^ie_z]_{HT}^T$. The vector ${}^i\mathbf{e}_{HT}$ shows the positive direction of the force ${}^i\mathbf{T}_{HT}$. This means that, if the propeller angular velocity is positive, it will exert the force ${}^i\mathbf{T}_{HT}$ in the direction of ${}^i\mathbf{e}_{HT}$. Otherwise, the force ${}^i\mathbf{T}_{HT}$ is opposite to ${}^i\mathbf{e}_{HT}$. The relationship between propeller spin direction and direction of the thrust and the torque vector is described in previous section (see page 3-22). An elegant way to describe this relationship is to introduce a *spin direction coefficient* ${}^ic_{HT}$: the value ${}^ic_{HT} = +1(-1)$ means that the force vector ${}^i\mathbf{T}_{HT}$ and the torque vector ${}^i\mathbf{Q}_{eHT}$ have the same (opposite) direction.

Example 3.1 (Propeller spin direction)

Assume that propeller angular velocity is positive and that blades are chosen such that propeller spin direction is clockwise (looking from the back of the propellers, see Figure 3.8(a)). The direction of the torque vector ${}^i\mathbf{Q}_{eHT}$ is determined using the right hand rule. In this case the force vector ${}^i\mathbf{T}_{HT}$ and the torque vector ${}^i\mathbf{Q}_{eHT}$ have the same direction as the vector ${}^i\mathbf{e}_{HT}$ and ${}^ic_{HT} = +1$. If the propeller spin direction is counter clockwise (Figure 3.8(b)), then vectors ${}^i\mathbf{T}_{HT}$ and ${}^i\mathbf{e}_{HT}$ have the same direction, while vectors ${}^i\mathbf{Q}_{eHT}$ and ${}^i\mathbf{e}_{HT}$ have the opposite direction and ${}^ic_{HT} = -1$.

Position vectors for different configurations are given in Table 3.5, while Table 3.6 shows orientation vectors. Parameters a , b , α and R can be obtained from the technical specifications of the vehicles.

(a) Clockwise ($i_{c_{HT}} = +1$).(b) Counter clockwise ($i_{c_{HT}} = -1$).**Figure 3.8** Relationship between propeller spin direction and direction of the thrust and the torque vector².

	Horizontal thrusters				Vertical thruster
	${}^1\mathbf{r}_{HT}$	${}^2\mathbf{r}_{HT}$	${}^3\mathbf{r}_{HT}$	${}^4\mathbf{r}_{HT}$	${}^1\mathbf{r}_{VT}$
X-shaped configuration (FALCON)	$\begin{bmatrix} b/2 \\ -a/2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} b/2 \\ a/2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -b/2 \\ a/2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -b/2 \\ -a/2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ L_z \end{bmatrix}$
Cross-shaped configuration (URIS)	$\begin{bmatrix} 0 \\ -R \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ R \\ 0 \end{bmatrix}$	$\begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -R \\ 0 \\ 0 \end{bmatrix}$	-

Table 3.5 Position vectors for different thruster configurations.

	Horizontal thrusters				Vertical thruster
	${}^1\mathbf{e}_{HT}$	${}^2\mathbf{e}_{HT}$	${}^3\mathbf{e}_{HT}$	${}^4\mathbf{e}_{HT}$	${}^1\mathbf{e}_{VT}$
X-shaped configuration (FALCON)	$\begin{bmatrix} \cos \alpha \\ \sin \alpha \\ 0 \end{bmatrix}$	$\begin{bmatrix} \cos \alpha \\ -\sin \alpha \\ 0 \end{bmatrix}$	$\begin{bmatrix} \cos \alpha \\ \sin \alpha \\ 0 \end{bmatrix}$	$\begin{bmatrix} \cos \alpha \\ -\sin \alpha \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
Cross-shaped configuration (URIS)	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$	-

Table 3.6 Orientation vectors for different thruster configurations.² Figure 3.8 displays front side of propellers.

3.7.2 Propeller shaft speed models

Short descriptions of one-, two- and three-state dynamic models for propeller shaft speed are given in this section, while more information can be found in cited papers.

One-state model

A *one-state model* with propeller shaft speed n as state and propeller thrust T as output was proposed in (Yoerger, *et al.*, 1991). The model can be written as:

$$J_m \dot{n} + K_n |n|n = \tau \quad (3.61)$$

$$T = T(n, u_p) \quad (3.62)$$

where u_p is the axial flow velocity in the propeller disc and τ is the control input (shaft torque). It is common to assume that $u_p = 0$ when computing T . However, u_p can be measured by using special devices, such as a *Laser-Doppler Velocimeter* (LDV) system or a *Particle Image Velocimeter* (PIV) system. Fossen and Blanke (2000) designed a state observer for reconstruction of u_p , where u_p is treated as an unmeasured state. The main drawback of the one-state model is its inability to describe overshoots in thrust, which are observed in experimental data.

Two-state model

Healey, *et al.* (1995) have modified the one-state model to include possibility to describe overshoots in thrust. They proposed a *two-state model* with n and u_p as states:

$$J_m \dot{n} + K_n n = \tau - Q_e \quad (3.63)$$

$$m_f \dot{u}_p + d_f (u_p - u) u_p - u = T \quad (3.64)$$

$$T = T(n, u_p) \quad (3.65)$$

$$Q_e = Q_e(n, u_p) \quad (3.66)$$

where u is the forward speed of the vehicle and Q_e is the propeller torque. The model was obtained by modelling a control volume of water around the propeller as a mass-damper system. Experimental verifications of the one- and two-state models are given in (Whitcomb and Yoerger, 1999).

Three-state model

A more general model is the three-state model proposed by Blanke, *et al.* (2000), with n , u_p and u as states:

$$J_m \dot{n} + K_n n = \tau - Q_e \quad (3.67)$$

$$m_f \dot{u}_p + d_{f0} u_p + d_f |u_p| (u_p - u_a) = T \quad (3.68)$$

$$(m - X_u) \dot{u} - X_u u - X_{u|u}|u| = (1 - t)T \quad (3.69)$$

$$T = T(n, u_p) \quad (3.70)$$

$$Q_e = Q_e(n, u_p) \quad (3.71)$$

where damping in surge is modelled as the sum of *linear laminar skin friction*, $-X_u u$ and *non-linear quadratic drag*, $-X_{u|u}|u|$. Similarly, linear damping, $d_{f0} u_p$, is included in the axial flow model, since quadratic damping, $d_f |u_p| u_p$, alone would give an unrealistic response at low speeds. Linear skin friction gives exponential convergence to zero at low speeds (Fossen, 2002).

The ambient water velocity u_a is computed by using the steady-state condition:

$$u_a = (1 - w)u \quad (3.72)$$

where $w, 0 < w < 1$ is the *wake fraction number*.

3.7.3 Propeller thrust and torque modelling

For a fixed pitch propeller the shaft torque Q_s and force (thrust) T depend on the forward speed u of the vehicle, the advance speed u_a (ambient water speed) and the propeller rate n (Fossen, 2002). In addition, other dynamic effects, due to *unsteady flows*, influence the propeller thrust and torque. The following unsteady flow effects are significant (Carlton, 1994; Newman, 1977):

- air suction,
- cavitation,
- in-and-out-of-water effects (Wagner's effect),
- wave influenced boundary layer effect,
- Kuessner effect (gust).

For a deeply submerged vehicle, the first four effects can be neglected. The Kuessner effect, caused by a propeller in gust, appears as a rapid oscillating thrust component. These fluctuations are usually small compared to the total thrust in dynamic regime. Under these assumptions, the thrust and torque models can be modelled using a *quasi-steady* representation.

Quasi-steady thrust and torque

Quasi-steady modelling of thrust and torque is usually performed in terms of *lift* and *drag* curves, which are transformed to thrust and torque by using the angle of incidence. The lift and drag are usually represented as non-dimensional thrust and torque coefficients K_T and K_Q , computed from self-propulsion tests (Fossen, 1995; Fossen and Sagatun, 1994):

$$K_T(J_0) = \frac{T}{\rho D^4 n |n|} \quad (3.73)$$

$$K_{Q_e}(J_0) = \frac{Q_e}{\rho D^5 n |n|} \quad (3.74)$$

where D is the propeller diameter, ρ is the water density and

$$J_0 = \frac{u_a}{nD} \quad (3.75)$$

is the *advance ratio*. The numerical expressions for K_T and K_{Q_e} can be found by open water tests, usually performed in a cavitation tunnel or a towing tank. In this tests unsteady flow effects are neglected and steady-state values of T , Q_e and n are used.

From (3.73) & (3.74) the thrust T and the torque Q_e can be expressed as

$$T = \rho D^4 K_T(J_0) n |n| \quad (3.76)$$

$$Q_e = \rho D^5 K_{Q_e}(J_0) n |n| \quad (3.77)$$

The open water propeller efficiency in undisturbed water is given as the ratio of the work done by the propeller in producing a thrust force to the work required to overcome the shaft torque (Fossen, 2002). Consequently:

$$\eta_0 = \frac{u_p T}{2\pi n Q_e} = \frac{J_0}{2\pi} \cdot \frac{K_T}{K_{Q_e}} \quad (3.78)$$

Typical curves for K_T , K_{Q_e} and η_0 are shown in Figure 3.9. They are taken from (Fossen and Sagatun, 1991) and represent results obtained by open water test, performed with the *Norwegian Experimental Remotely Operated Vehicle* (NEROV) thruster in the towing tank at the Norwegian Marine Technology Research Institute in Trondheim. For positive values of J_0 experiments showed that K_T and K_{Q_e} are linear in J_0 , while the results for negative values of J_0 show a non-linear behaviour. Linear approximation of experimental curves for K_T and K_{Q_e} for positive J_0 yields

$$K_T(J_0) = \alpha_2 - \alpha_1 J_0 \quad (3.79)$$

$$K_{Q_e}(J_0) = \beta_2 - \beta_1 J_0 \quad (3.80)$$

where α_i and β_i ($i=1,2$) are four positive non-dimensional coefficients.

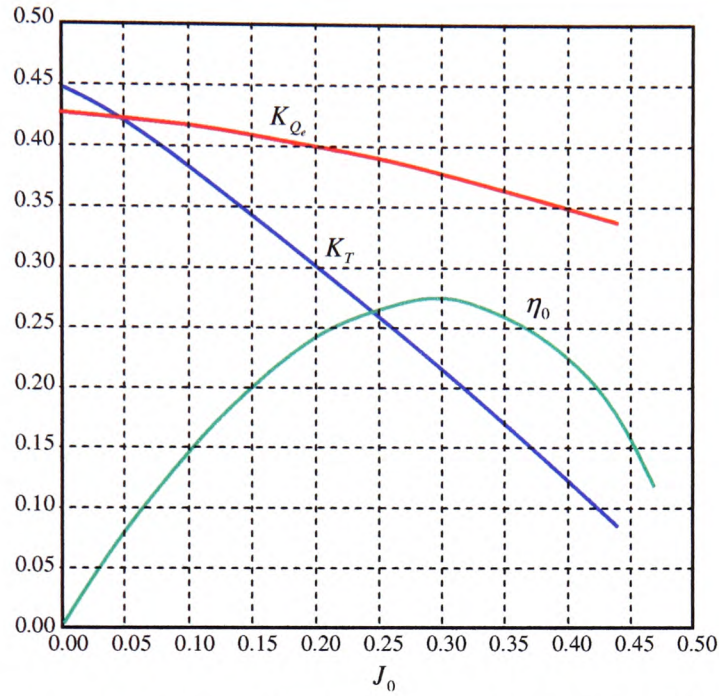


Figure 3.9 Non-dimensional coefficients K_T , K_{Q_e} and η_0 as a function of the positive advance ratio J_0 .

Substituting (3.79) & (3.80) into (3.76) & (3.77) yields

$$T(n, u_a) = T_{n|n}|n| - T_{n|u_a}|n|u_a \quad (3.81)$$

$$Q_e(n, u_a) = Q_{n|n}|n| - Q_{n|u_a}|n|u_a \quad (3.82)$$

where

$$\begin{aligned} T_{n|n} &= \rho D^4 \alpha_2, & T_{n|u_a} &= \rho D^3 \alpha_1 \\ Q_{n|n} &= \rho D^5 \beta_2, & Q_{n|u_a} &= \rho D^4 \beta_1 \end{aligned} \quad (3.83)$$

are positive propeller coefficients, given by the propeller characteristics. Expressions (3.81) & (3.82) are known as a *bilinear thruster model* (Fossen, 1995).

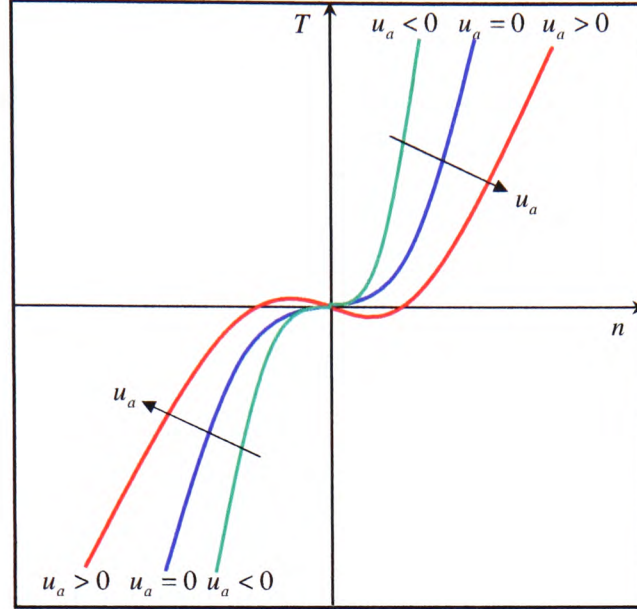


Figure 3.10 Propeller thrust (force) T as a function of propeller revolution n and ambient water velocity u_a (bilinear thruster model).

Figure 3.10 displays typical T -curves as a function of propeller angular velocity n and the ambient water velocity u_a . Typical parabolic shape is deformed, depending on the size and sign of u_a . For example, for the same angular velocity n , the propeller will generate a larger force in the case $u_a = 0$ than for $u_a > 0$. This can be confirmed in the ROV simulator (Appendix D) by changing the thruster model (bilinear/affine) and monitoring force responses.

However, in practical applications, the bilinear thruster model (3.81) - (3.82) can be approximated by an *affine thruster model* (3.84) - (3.85), assuming $u_a = 0$:

$$T(n) = T_{n|n} n |n| \quad (3.84)$$

$$Q_e(n) = Q_{n|n} n |n| \quad (3.85)$$

It is important to note that the efficiency of the propeller may not be the same for both possible spin directions, i.e. for the same propeller revolution ($|n_1| = |n_2|$) the

corresponding thrusts (forces) may not be equal ($|T_1| > |T_2|$) (Figure 3.11). This means that, in real applications, the T -curve is not symmetrical, i.e. $T(n)$ is not a odd function of n . In order to include non-symmetry in the model, the function $T(n)$ can be rewritten as

$$T(n) = \begin{cases} T_{n|n|}^+ n|n|, & n > 0 \\ T_{n|n|}^- n|n|, & n < 0 \end{cases} \quad (3.86)$$

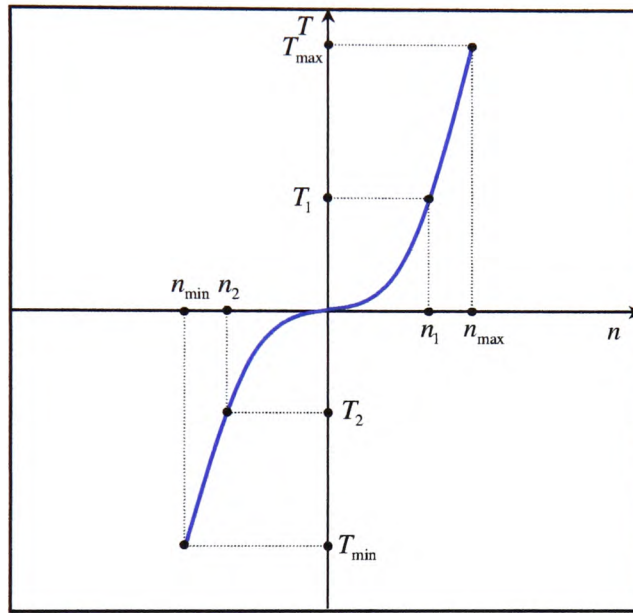


Figure 3.11 Propeller thrust (force) T as a function of propeller revolution n (affine thruster model).

where $T_{n|n|}^+$ and $T_{n|n|}^-$ ($T_{n|n|}^+ > T_{n|n|}^- > 0$) are curve coefficients in the first and the third quadrant, respectively. In addition, it can be observed from Figure 3.11 that $|T_{\max}| > |T_{\min}|$, although $|n_{\max}| = |n_{\min}|$. Commands, generated by an ROV pilot, are interpreted as desired forces and moments and it would be very useful, from pilot's point of view, to have symmetrical command space, where $|T_{\max}| = |T_{\min}|$.

The original solution for this problem is presented in the following. The first step is to introduce a new control variable u ³

$$u = \eta|\eta| \quad (3.87)$$

Equation (3.86) can be rewritten as

$$T(u) = \begin{cases} T_{\eta|\eta|}^+ u, & u > 0 \\ T_{\eta|\eta|}^- u, & u < 0 \end{cases} \quad (3.88)$$

In this way, the quadratic relationship (3.86) is replaced by the linear relationship (3.88) (see Figure 3.12).

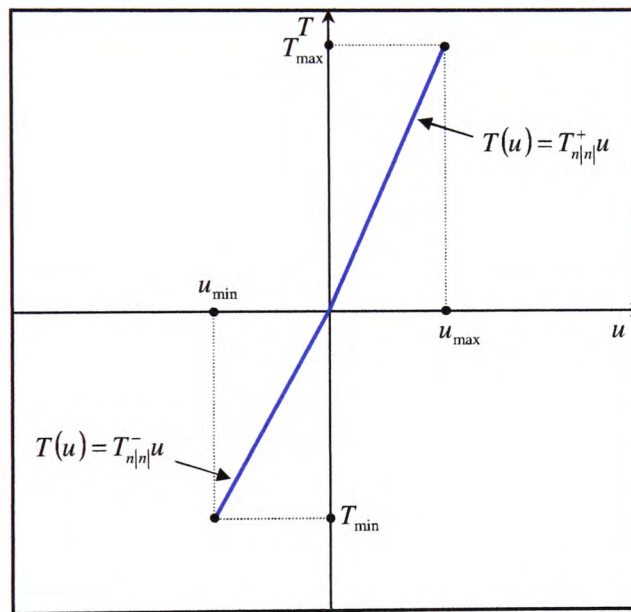


Figure 3.12 Propeller thrust (force) T as a function of new control variable u (affine thruster model).

³ Unfortunately, nomenclature for marine vessels is not consistent. The symbol u is also used to represent the forward speed of the vehicle. This confusion will be resolved later by vectorisation.

The next step is to introduce a new coefficient K and an auxiliary variable u' , defined as:

$$K = \frac{1}{2}(T_{n|n|}^+ + T_{n|n|}^-) \quad (3.89)$$

$$T(u') = Ku' \quad (3.90)$$

The relationship between u and u' can be found in the following way (see Figure 3.13):

$$Ku' = T_{n|n|}^+ u \Rightarrow u = \frac{K}{T_{n|n|}^+} u', \quad u' > 0 \quad (3.91)$$

$$Ku' = T_{n|n|}^- u \Rightarrow u = \frac{K}{T_{n|n|}^-} u', \quad u' < 0 \quad (3.92)$$

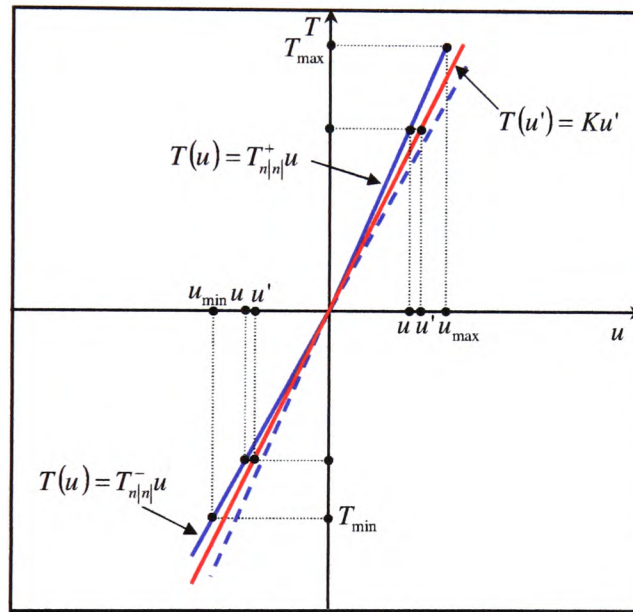


Figure 3.13 Relationship between control variable u and auxiliary control variable u' .

Finally, the last step is to make a range of force T symmetrical by introducing new limits (see Figure 3.14):

$$-T_m \leq T \leq T_m \quad (3.93)$$

where

$$T_m = |T_{min}| \quad (3.94)$$

New range for u' is given by

$$-u_m' \leq u' \leq u_m' \quad (3.95)$$

where

$$u_m' = \frac{|T_{\min}|}{K} \quad (3.96)$$

In this way, the non-symmetrical relationship is transformed to symmetrical.

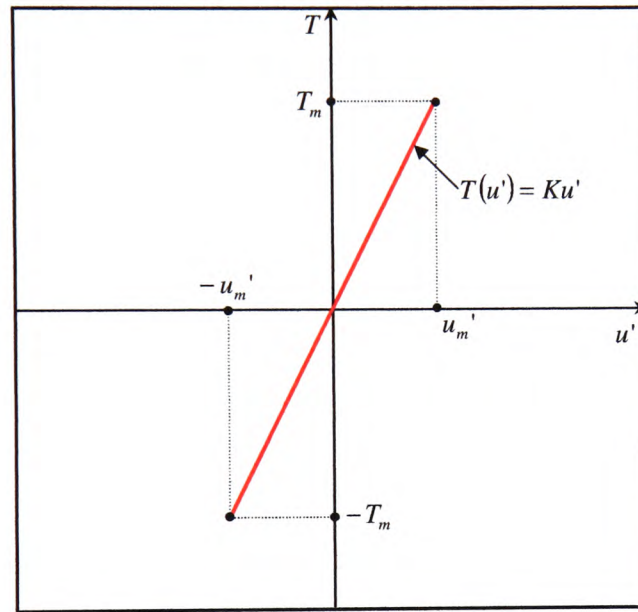


Figure 3.14 Propeller thrust (force) T as odd function of auxiliary control variable u' .

Comments

- if the auxiliary control variable u' is used, relationship between thrust (force) T and u' is linear and ranges of variables are symmetrical. Because of this reason, it is easy to normalise control constraints and the control allocation problem can be formulated in the normalised form, which is more understandable and easier to solve.
- Transformation from u' to u is easy: after u' is calculated, u can be calculated from (3.91) or (3.92), depending on the sign of u' .

- Transformation from u to n is achieved using equation

$$n = \sqrt{|u|} \cdot \text{sgn } u \quad (3.97)$$

- the "price" paid for achieving a symmetrical relationship between T and u is that a part of achievable force range $[T_{\min}, T_{\max}]$ is lost, because $u_m' < u_{\max}$ and interval $[u_m', u_{\max}]$ is never used.

3.7.4 Thruster test rig (IMPROVES project)

A part of the IMPROVES project is development of a thruster test rig. The aim is to develop facilities for performing experiments with the FALCON thruster, in order to obtain experimental curves similar to those shown in Figure 3.9. This is an ongoing project, expected to be finished in May 2004. When the project is completed, the thruster model used in the ROV simulator can be upgraded to represent real FALCON thruster dynamics.

3.7.5 Vectorisation

Previous discussion about thruster models assumed scalar variables. In order to develop the Simulink model of the thrusters, it is necessary to use vector variables. In addition, vectorisation resolves the confusion related with the nomenclature (see page 3-39). In the following it is assumed that the position and the orientation of the thruster is determined by vectors \mathbf{r} and \mathbf{e} , respectively (see Figure 3.15).

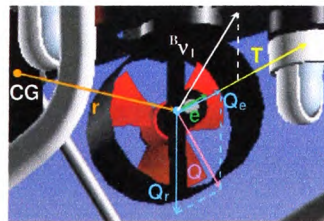


Figure 3.15 Thrust and torque as vector variables.

Bilinear thruster model

The forward speed u in (3.72) should be interpreted as a projection of the linear velocity vector ${}^B \mathbf{v}_1 = [u \ v \ w]^T$ on the orientation vector \mathbf{e} . Consequently, the bilinear thruster model can be represented in vector form as

$$\begin{aligned}
 u_a &= (1-w)^B \mathbf{v}_1^T \cdot \mathbf{e} \\
 T(n, u_a) &= T_{n|n}|n| - T_{n|u_a}|n|u_a, \quad T_{n|n|} = \begin{cases} T_{n|n|}^+, & n > 0 \\ T_{n|n|}^-, & n < 0 \end{cases} \\
 \mathbf{T} &= T\mathbf{e} \\
 Q_e(n, u_a) &= Q_{n|n}|n| - Q_{n|u_a}|n|u_a \\
 \mathbf{Q}_e &= cQ_e\mathbf{e}, \quad \mathbf{Q}_r = \mathbf{r} \times \mathbf{T} \\
 \mathbf{Q} &= \mathbf{Q}_e + \mathbf{Q}_r
 \end{aligned} \tag{3.98}$$

Affine thruster model

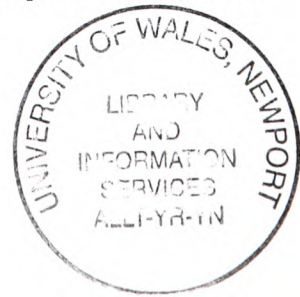
The vector form of the affine thruster model is obtained from the bilinear thruster model (3.98), assuming $u_a = 0$:

$$\begin{aligned}
 T(n) &= T_{n|n}|n|, \quad T_{n|n|} = \begin{cases} T_{n|n|}^+, & n > 0 \\ T_{n|n|}^-, & n < 0 \end{cases} \\
 \mathbf{T} &= T\mathbf{e} \\
 Q_e(n) &= Q_{n|n}|n| \\
 \mathbf{Q}_e &= cQ_e\mathbf{e}, \quad \mathbf{Q}_r = \mathbf{r} \times \mathbf{T} \\
 \mathbf{Q} &= \mathbf{Q}_e + \mathbf{Q}_r
 \end{aligned} \tag{3.99}$$

Both vector forms are implemented in the ROV simulator.

3.7.6 Full thruster model

A full thruster model, including dynamics of the thruster control loop, is shown in Figure 3.16. A DC motor, designed for underwater operating conditions, drives a thruster. The input to the thruster control loop is a control variable n_d (desired angular velocity). The motor is equipped with a tachometer, which measures actual angular velocity \bar{n} . A



gearbox with the gear ratio $GR > 1$ is used to reduce the output angular velocity \bar{n} ($n = (1/GR)\bar{n}$) and to enhance the output torque. Because of this reason, the input n_d must be multiplied by GR ($\bar{n}_d = GR \cdot n_d$). In this case, the thruster control loop looks like a box, with n_d as the input and n as the output, although inside the motor the shaft rotates much faster. A typical thruster control loop is implemented as independent device called *Thruster Control Unit* (TCU) with integrated power amplifiers and controlled by a microcontroller. More information about the TCU for FALCON and URIS can be found in Appendix A. The velocity controller is usually implemented as a digital PID controller, although the other designs are possible.

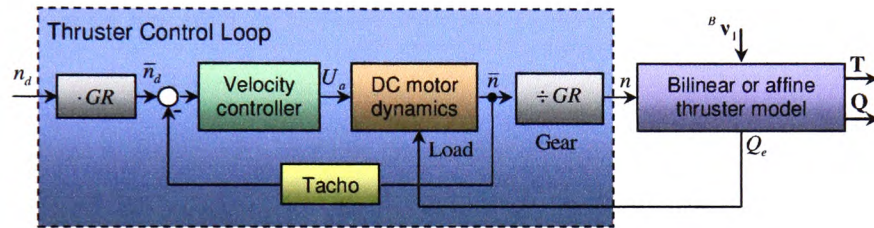


Figure 3.16 Block diagram showing full thruster model, including thruster control loop dynamics.

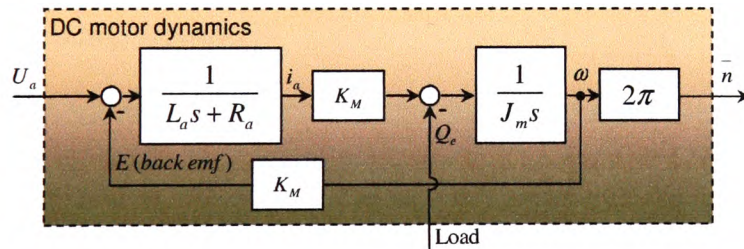


Figure 3.17 Linear dynamic model of the speed-controlled DC motor.

Most thruster systems are driven by DC motors designed for underwater operating conditions. The dynamic model of a speed-controlled DC motor can be written as

$$L_a \frac{di_a}{dt} = -R_a i_a - K_M \omega + U_a \quad (3.100)$$

$$J_m \frac{d\omega}{dt} = K_M i_a - Q_e \quad (3.101)$$

where

- L_a is the armature inductance $[H]$,
- R_a is the armature resistance $[\Omega]$,
- U_a is the armature voltage $[V]$,
- K_M is the motor torque constant $[Nm/A]$,
- J_m is the moment of inertia of motor and thruster $[kgm^2]$,
- ω is the angular velocity of the motor $[rad/s]$,
- Q_e is the load from the propeller, defined in (3.82) $[Nm]$.

Figure 3.17 displays internal structure of the block "DC motor dynamics", shown in the block diagram in Figure 3.16, based on equations (3.100) & (3.101). In order to obtain a more realistic model, it would be necessary to include hard non-linearities like actuator saturation, Coulomb friction, dead-zones and hysteresis into the model.

3.8 Concluding remarks

The purpose of the chapter was to describe non-linear dynamic models of thrusters and an ROV in 6 DOF. These mathematical models are used in the ROV simulator to investigate the performance of the FDAS, described in Chapter 5. Discussion was concentrated on ROVs, although the same models can be used to describe AUV dynamics. In this way, the ROV simulator can be augmented with the ability to compare different control architectures for AUVs.

A novel approach has been proposed whereby the affine thruster model is transformed into symmetrical form, enabling normalisation and easier visualisation of the control allocation problem.

3.9 References

- BLANKE, M., LINDEGAARD, K.P. and FOSSEN, T.I. (2000). Dynamic Model for Thrust Generation of Marine Propellers. In *Proceedings of the IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC 2000)*, Aalborg, Denmark.
- BOŠKOVIĆ, D.M. and KRSTIĆ, M. (1999). Global attitude/position regulation for underwater vehicles. *International Journal of Systems Science*, **30**(9). pp. 939-946.
- CARLTON, J.S. (1994). *Marine Propellers and Propulsion*. Oxford: Butterworth-Heinemann.
- CHOU, J.C.K. (1992). Quaternion Kinematic and Dynamic Differential Equations. *IEEE Transactions on Robotics and Automation*, **RA-8**(1): 53-64.
- FJELLSTAD, O.-E. and FOSSEN, T.I. (1994a). Singularity-Free Tracking of Unmanned Underwater Vehicles in 6 DOF. *Proceedings of the 33rd Conference on Decision and Control*, Lake Buena Vista, FL.
- FJELLSTAD, O.-E. and FOSSEN, T.I. (1994b). Quaternion Feedback Regulation of Underwater Vehicles, *Proceedings of the 3rd IEEE Conference on Control Applications (CCA'94)*, Glasgow, pp. 857-862.
- FJELLSTAD, O.-E. and FOSSEN, T.I. (1994c). Position and Attitude Tracking of AUVs: A Quaternion Feedback Approach, *IEEE Journal of Oceanic Engineering*, **OE-19**(4), pp. 512-518.
- FOSSEN, T.I. and SAGATUN, S.I. (1991). . Adaptive Control of Nonlinear Underwater Robotic Systems. *Proceedings of the IEEE International Conference on Robotics and Automation*, Sacramento, CA, pp. 1687-1694.
- FOSSEN, T.I. (1995). *Guidance and Control of Ocean Vehicles*. Chichester: John Wiley & Sons.
- FOSSEN, T.I. and BLANKE, M (2000). Nonlinear Output Feedback Control of Underwater Vehicle Propellers Using Feedback From Estimated Axial Flow Velocity. *IEEE Journal of Oceanic Engineering*, Vol. **25**, No. 2, pp. 241-255.
- FOSSEN, T.I. (2002). *Marine Control Systems*. Marine Cybernetics AS, Trondheim, Norway.
- HEALEY, A.J., ROCK, S.M., CODY, S., MILES, D. and BROWN, J.P. (1995). Toward an

improved understanding of thruster dynamics for underwater vehicles. *IEEE Journal of Oceanic Engineering*, **JOE-29**(4), pp. 354-361.

LEWIS, A.J., ROCK, S.M., CODY, S., MILES, D. and BROWN, J.P. (1995). Toward an improved understanding of thruster dynamics for underwater vehicles. *IEEE Journal of Oceanic Engineering*, **JOE-29**(4), pp. 354-361.

NEWMAN, J.N. (1977). *Marine Hydrodynamics*. MIT Press. Cambridge, MA.

RIDAO, P., BATLLE, J. and CARRERAS, M. (2001). Dynamic model of an underwater robotic vehicle. In: P. RIDAO, *A hybrid control architecture for an AUV*. Ph.D. thesis. Girona: University of Girona.

SILPA-ANAN, C. (2001). *Autonomous Underwater Robot: Vision and Control*. MSc thesis. The Australian National University.

SNAME (1950). The Society of Naval Architects and Marine Engineers. Nomenclature for Treating the Motion of a Submerged Body Through a Fluid. *Technical and Research Bulletin No. 1-5*.

STUELPNAGEL, P. (1964). On the Parametrisation of the Three-Dimensional Rotation Group. *SIAM Review*, **6**(4), pp. 422-430.

WHITCOMB, L.L. and YOERGER, D.R. (1999). Development, Comparison and Preliminary Experimental Validation of Nonlinear Dynamic Thruster Models. *IEEE Journal of Oceanic Engineering*, **JOE-24**(4), pp. 481-494.

YOERGER, D.R., COOKE, J.G. and SLOTINE, J.J.E. (1991). The Influence of Thruster Dynamics on Underwater Vehicle Behaviour and Their Incorporation into Control Systems Design. *IEEE Journal of Oceanic Engineering*, **JOE-15**(3), pp. 167-179.

Chapter 4: Control Allocation

4.1 Introduction

The control allocation problem plays a vital role in the accommodation part of the FDAS. Typically, open-frame underwater vehicles have $p \geq 4$ actuators (thrusters) for the motion in the horizontal plane and the control allocation problem in this case is very complex and hard to visualise, because the normalised constrained control subset Ω is p -dimensional unit cube. The aim of this chapter is to give a clear picture and a geometric interpretation of the problem, to present existing methods for its solution and to introduce a hybrid approach, based on the integration of a pseudoinverse and the fixed-point iteration method, which is able to allocate the entire attainable command set and finds the solution optimal in l_2 sense, i.e. which minimises the control energy cost function. The performance of presented methods is compared using the same example in low-dimensional control spaces, where the main idea of the method can be easily visualised and geometrically interpreted. However, the same concepts can be extended for higher dimensional cases, which are explored further in Chapter 5.

The chapter is organised as follows. The general control architecture, as well as modification of this architecture for aerospace and underwater applications, is introduced in section 4.2. In section 4.3 the control allocation problem is formulated, and notation and terminology are introduced. A survey of existing control allocation methods is given in section 4.4. Each method is used to solve the same sample problem, with a clear geometric interpretation of the method. The concluding remarks are given in section 4.5 and a list of references, cited in the text, is provided in section 4.6.

4.2 Control system architecture

The control allocation problem is related with the determination of a set of actuator commands, which produce a given set of desired (commanded) controls. The task of

finding control combinations to achieve a specific objective is only a part of the larger control problem. Using control allocation, the actuator selection task is separated from the regulation task in the control design. The majority of modern aircrafts and marine vessels represent overactuated systems, for which it is possible to split the control design into the following steps (Härkegård, 2003):

1. **REGULATION TASK:** Design a control law, which specifies the total control effort to be produced (net force, moment, etc.),
2. **ACTUATOR SELECTION TASK:** Design a control allocator, which maps the total control effort (demand) onto individual actuator settings (thrust forces, control surface deflections, etc.).

Figure 4.1 illustrates the configuration of the overall control system. The control system consists of a control law (specifying the total control effect, \mathbf{v} , that should be produced) and a control allocator (allocating control vector, \mathbf{u} , which distributes this control demand among the individual actuators). In the system, the actuators generate a total control effect, \mathbf{v}_{sys} , which is applied as the input to system dynamics block and which determines the system behaviour. The main objective of the control allocation is to ensure that condition $\mathbf{v}_{\text{sys}} = \mathbf{v}$ is satisfied for all attainable \mathbf{v} .

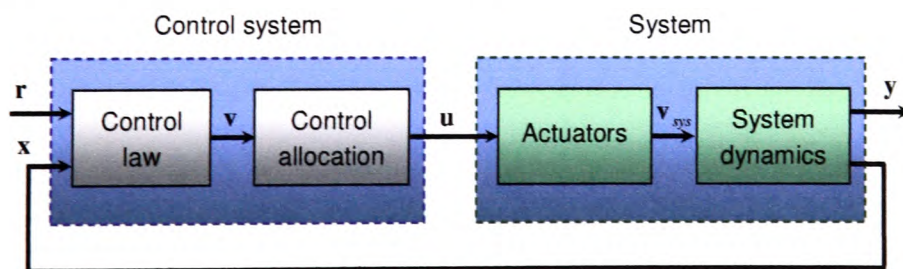


Figure 4.1 The overall control system architecture (Härkegård, 2003).

In aerospace applications, many authors have suggested a modular flight control structure similar to that presented in Figure 4.1. For example, Figure 4.2 displays a control

structure where the part of the overall system responsible for following the desired response is conceptually separated from the part that is responsible for handling control redundancy (Beck, 2002). The control law, which maps the desired response to a set of commands (objectives), is not dependent on the design of the control allocation system, which relates these commands with settings and positions of individual effectors.

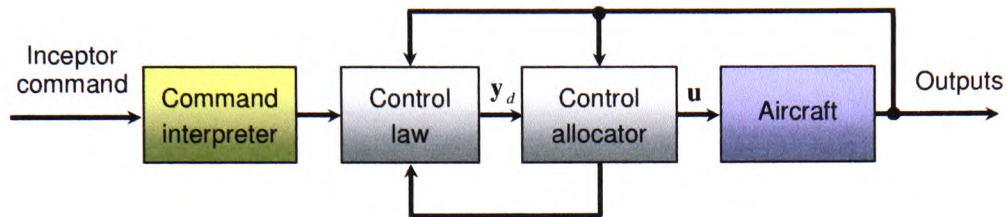


Figure 4.2 Modular flight control structure (Beck, 2002).

A similar control structure exists in underwater applications. Figure 4.3 shows a typical, open-loop ROV control structure. An ROV pilot uses *Hand Control Unit* (HCU) and information from sensors and real-time video from the on-board camera to generate commands, which cause the vehicle to follow the desired trajectory according to the mission objective. These commands can be interpreted as a desired forces and moments among axes in the body-fixed frame. Raw signals from the HCU pass through the low-pass pre-filter to smooth out the commanded input, in order to prevent abrupt changes in set points and to protect the thrusters from damage. Beside that, additional restrictions¹ can be included in the pre-filter to prevent undesired behaviour of the vehicle. The output of the pre-filter is the vector of desired forces and moments (virtual control input) τ_d . The task of the control allocator is to find control settings for individual actuators (true

¹ FALCON has so powerful thrusters that it would be able to perform a full loop in vertical plane, if full freedom for commanded inputs is allowed. Because of this reason, restrictions are built in the control software to restrict the size of the attainable command set.

control input) \mathbf{u} such that $\boldsymbol{\tau} = \boldsymbol{\tau}_d$, where $\boldsymbol{\tau}$ is vector of propulsion forces and moments, exerted by the actuators after actuation with the control vector \mathbf{u} . For an ROV the most common actuators are thrusters and control surfaces, such as fins for diving, rolling and pitching, rudders for steering, etc.

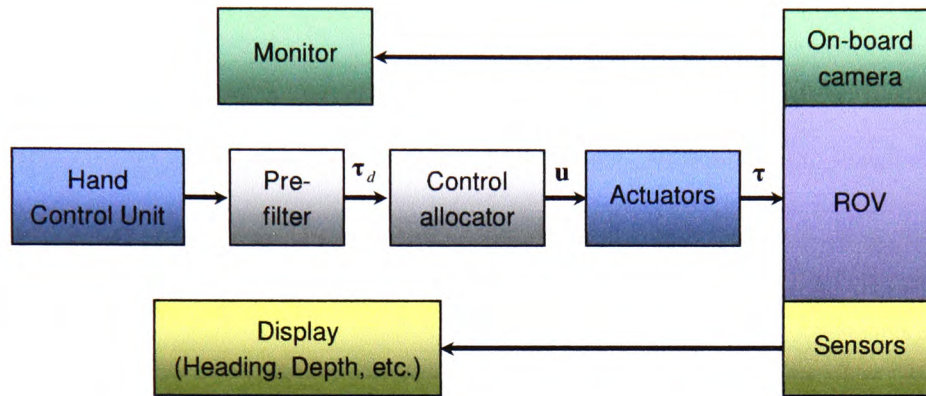


Figure 4.3 Typical open-loop ROV control structure.

In contrast to ROV control architecture, Figure 4.4 displays a typical, closed-loop AUV control structure. In order to achieve mission objective, the control law uses actual knowledge about the environment and sensors' measurements to find a reference inputs for a set of controllers (heading controller, depth controller, etc.). The outputs of the controllers are integrated into a vector that is similar to the output of the HCU in Figure 4.3. The control allocator performs in exactly the same way as in the previous case. Hence, from the control allocator point of view, it does not matter how the virtual control input $\boldsymbol{\tau}_d$ is generated (by the ROV pilot or the control law). The control allocation algorithm is the same for both structures. The task of the control allocator in both cases is to determine appropriate control settings for individual actuators, which produce the desired set of forces and moments.

It is useful at this point to consider the role of the FDAS in the control structures shown in Figure 4.3 and Figure 4.4. Basically, the FDAS performs the control allocation task, but

this primary task is enhanced with the ability to monitor the state of the thrusters and, if necessary, perform automatic reconfiguration, i.e. redistribution of propulsion forces among the operable thrusters (Figure 4.5).

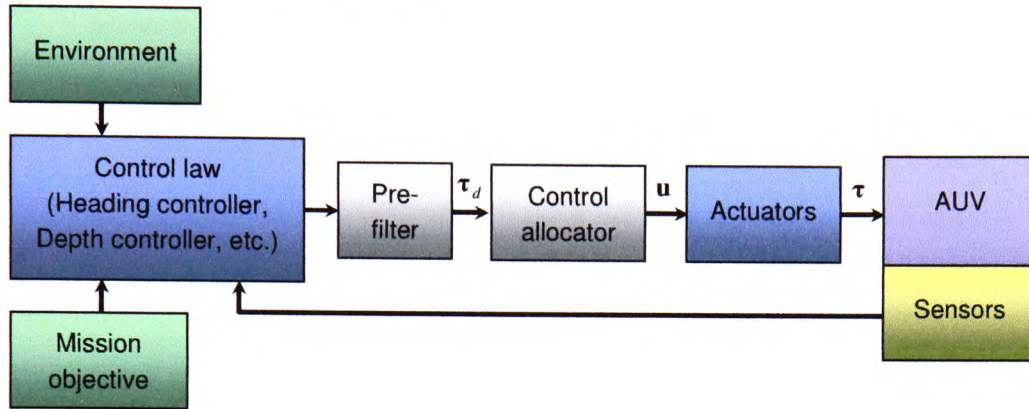


Figure 4.4 Typical closed-loop AUV control structure.

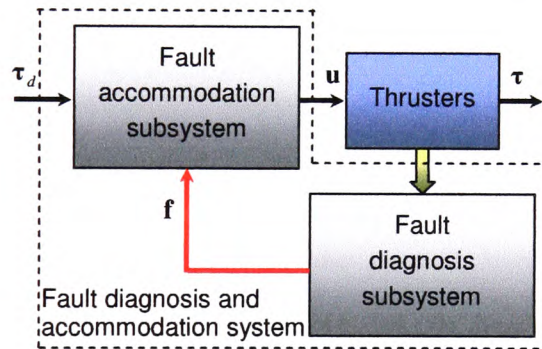


Figure 4.5 Relationship between the FDAS and a typical control structure for open-frame underwater vehicle.

In particular, the FDAS consists of two subsystems: a *Fault Diagnosis Subsystem* (FDS) and a *Fault Accommodation Subsystem* (FAS). The FDS uses *Fault Detector Units* (FDUs), associated with each thruster, to monitor their state. The output of the FDS is the total fault indicator vector f , carrying the codes of faulty states for each thruster. The FAS uses information provided by the FDS to accommodate faults and perform an appropriate reconfiguration, i.e. to reallocate control energy among operable thrusters.

Treating control allocation independently of the control law is convenient because of the following:

- *Actuator constraints can be taken into account.* In real applications actuator saturation always exists. If one actuator saturates, some of methods for control allocation are able to redistribute control energy among other available actuators to compensate for the inability of a saturated thruster to produce its nominal control effect. In this way, available control resources are fully exploited before the closed-loop is degraded (Durham, 1993).
- *Reconfiguration can be performed.* If the effectiveness of the actuators change over time, or in the case of an actuator total or partial fault, reconfiguration i.e. redistribution of control energy among a set of available actuators can be performed, without having to redesign the control law (Eberhardt and Ward, 1999; Wise, *et al.*, 1999).
- *Adaptation to a particular application.* The actuator utilisation can be optimised for the application considered, due to separation from the regulation problem. The actuator redundancy can be used for different purposes. In most cases, the extra freedom for control distribution, which is available for overactuated systems, is used to optimise some secondary objective, like total aerosurface deflections, drag or wing load in aerospace applications (Eberhardt and Ward, 1999; Wise, *et al.*, 1999), or total thrust in ship control applications (Sørdalen, 1997; Lindfors, 1993). Another possibility is to include filtering in the control allocation process in order to obtain different control distribution among the actuators for different frequencies (Davidson, *et al.*, 2001).

4.3 The control allocation problem

As stated in section 2.9.2, the notation and terminology used in publications on control allocation are application driven and not consistent. This makes it difficult to understand the main ideas and compare the different approaches. In this section, an effort is made to develop a generic formulation of the control allocation problem, which provides a unified framework which will be used in the following chapters.

4.3.1 Problem formulation

The task of the *control allocator* is to solve underdetermined, typically constrained, system of equations (Härkegård, 2003). The input to the control allocator is the total control effect to be produced, the *virtual control input* $\mathbf{v}(t) \in \mathbb{R}^k$ (see Figure 4.1). The output of the control allocator is the *true control input* $\mathbf{u}(t) \in \mathbb{R}^m$, where $m > k$. When a set of actuators is actuated by vector \mathbf{u} , it generates the *total control effect* $\mathbf{v}_{sys}(t) \in \mathbb{R}^k$. If the control allocation is successful, $\mathbf{v}_{sys} = \mathbf{v}$.

Mathematically, for a given vector $\mathbf{v}(t) \in \mathbb{R}^k$ the vector $\mathbf{u}(t) \in \mathbb{R}^m$ must be found such that

$$\mathbf{g}(\mathbf{u}(t)) = \mathbf{v}(t) \quad (4.1)$$

where $\mathbf{g}: \mathbb{R}^m \rightarrow \mathbb{R}^k$ is the mapping from the true to the virtual control input, performed by the actuators and $\text{rank}(\text{Jacobian}(\mathbf{g})) = k$. The majority of publications in the field of control allocation consider a linear case, for which (4.1) has the form

$$\mathbf{B}\mathbf{u}(t) = \mathbf{v}(t) \quad (4.2)$$

where the *control effectiveness* matrix \mathbf{B} is a $k \times m$ matrix with rank k .

Actuator position constraints are given as set of inequalities

$$\mathbf{u}_{\min} \leq \mathbf{u}(t) \leq \mathbf{u}_{\max} \quad (4.3)$$

where the inequalities apply componentwise.

In addition, if actuator rate constraints exist, another set of inequalities need to be satisfied:

$$\rho_{\min} \leq \dot{\mathbf{u}}(t) \leq \rho_{\max} \quad (4.4)$$

Since the control allocator is part of the digital control system, the time derivative in (4.4) can be approximated as

$$\dot{\mathbf{u}}(t) \approx \frac{\mathbf{u}(t) - \mathbf{u}(t-T)}{T} \quad (4.5)$$

where T is the sampling time (Durham and Bordignon, 1996). Substituting (4.5) in (4.4) the rate constraints are transformed into set of an additional position constraints

$$T\rho_{\min} + \mathbf{u}(t-T) \leq \mathbf{u}(t) \leq T\rho_{\max} + \mathbf{u}(t-T) \quad (4.6)$$

Finally, combining (4.3) & (4.6) yields

$$\underline{\mathbf{u}}(t) \leq \mathbf{u}(t) \leq \bar{\mathbf{u}}(t) \quad (4.7)$$

where

$$\begin{aligned} \underline{\mathbf{u}}(t) &= \max\{\mathbf{u}_{\min}, T\rho_{\min} + \mathbf{u}(t-T)\} \\ \bar{\mathbf{u}}(t) &= \min\{\mathbf{u}_{\max}, T\rho_{\max} + \mathbf{u}(t-T)\} \end{aligned} \quad (4.8)$$

Dropping the time dependence, the standard constrained linear control allocation problem can be formulated as:

For given \mathbf{v} , find \mathbf{u} such that $\mathbf{B}\mathbf{u} = \mathbf{v}$ and $\underline{\mathbf{u}} \leq \mathbf{u} \leq \bar{\mathbf{u}}$.

The equation $\mathbf{B}\mathbf{u} = \mathbf{v}$ defines the set of hyperplanes in the true control space \mathbb{R}^m . The intersection of these hyperplanes is a convex set, denoted by \mathbb{K} . The set of inequalities $\underline{\mathbf{u}} \leq \mathbf{u} \leq \bar{\mathbf{u}}$ represent the hyperbox in the same space. This hyperbox is called *constrained (admissible) control subset* and denoted by Ω . The solution set \mathfrak{S} is given by the intersection of \mathbb{K} and Ω . Three cases are possible:

- \mathfrak{S} is empty (i.e. no solution exists),
- \mathfrak{S} has exactly one element (i.e. there is one unique solution),
- \mathfrak{S} has more than one element (i.e. there are many solutions).

Example 4.1 (Linear control allocation problem)

Consider the control allocation problem $\mathbf{B}\mathbf{u} = \mathbf{v}$, where

- True control input is $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \in \mathbb{R}^3$ ($m = 3$),
- Virtual control input is $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \in \mathbb{R}^2$ ($k = 2$),
- Control effectiveness matrix is $\mathbf{B} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{4} & -\frac{1}{4} \\ 0 & \frac{3}{5} & -\frac{2}{5} \end{bmatrix}$,
- Actuator position constraints are $\underline{\mathbf{u}} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \leq \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \leq \bar{\mathbf{u}} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

Equation $\mathbf{B}\mathbf{u} = \mathbf{v}$ represents system of equations

$$\begin{aligned} \frac{1}{2}u_1 - \frac{1}{4}u_2 - \frac{1}{4}u_3 &= v_1 \\ \frac{3}{5}u_2 - \frac{2}{5}u_3 &= v_2 \end{aligned} \tag{4.9}$$

Each equation in (4.9) represents a plane in \mathbb{R}^3 . Consequently, (4.9) can be rewritten as

$$\begin{aligned} \pi_1: \quad \mathbf{N}_1^T \cdot \mathbf{u} &= v_1 \\ \pi_2: \quad \mathbf{N}_2^T \cdot \mathbf{u} &= v_2 \end{aligned} \tag{4.10}$$

where $\mathbf{N}_1 = \begin{bmatrix} \frac{1}{2} & -\frac{1}{4} & -\frac{1}{4} \end{bmatrix}^T$ and $\mathbf{N}_2 = \begin{bmatrix} 0 & \frac{3}{5} & -\frac{2}{5} \end{bmatrix}^T$ are normal vectors, orthogonal on planes π_1 and π_2 , respectively. Since $\Delta = 0.1925 \neq 0$ (see Appendix C, eq. (C.9)) planes are not parallel and their intersection is a line l (see (C.4)):

$$l: \quad \mathbf{p} = \begin{bmatrix} \frac{104}{77}v_1 + \frac{10}{77}v_2 + \frac{1}{4}t \\ -\frac{40}{77}v_1 + \frac{85}{77}v_2 + \frac{1}{5}t \\ -\frac{60}{77}v_1 - \frac{65}{77}v_2 + \frac{3}{10}t \end{bmatrix} \quad (4.11)$$

where t is the parameter of the line. The line l is a convex set, denoted by \mathfrak{L} in previous discussion. The constrained control subset Ω , which satisfies actuator position constraints, is a unit cube in \mathfrak{R}^3 (Figure 4.6):

$$\Omega = \{\mathbf{u} \in \mathfrak{R}^3 \mid \|\mathbf{u}\|_{\infty} \leq 1\} \subset \mathfrak{R}^3 \quad (4.12)$$

Geometric interpretation of the control allocation problem can be obtained by reformulating the problem as follows:

For a given \mathbf{v} , find intersection \mathfrak{S} of l (4.11) and Ω (4.12).

Three cases are possible:

- If the intersection is a segment, there is infinite number of solutions (each point that belongs to the segment is solution),
- If the intersection is a point, there is only one solution,
- If the intersection is an empty set, no solution exists.

The control allocation problem, formulated in Example 4.1, will be used throughout the chapter to introduce the terminology and to demonstrate different approaches to the control allocation problem.

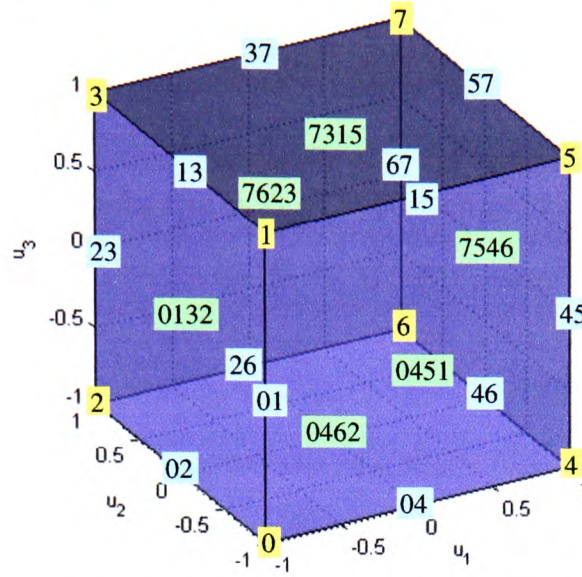


Figure 4.6 Constrained (admissible) control subset Ω .

4.3.2 Nomenclature for constrained control subset Ω

The following nomenclature is adopted for referring to Ω (Durham, 1993):

Boundary of Ω is denoted by $\partial(\Omega)$. A control vector belongs to $\partial(\Omega)$ if and only if at least one of its components is at a limit. *Vertices* are the points on $\partial(\Omega)$ where each control receives a limit (min or max). In Figure 4.6 vertices are denoted as **0**, **1**, ..., **7**. In the general case, the number of vertices is equal to 2^m . Vertices are numerated using the following rule: if the vertex is represented in a binary form, then "0" in the k^{th} position of this representation indicates that the corresponding control u_k is at a minimum \underline{u}_k , while "1" indicates it is at a maximum \bar{u}_k . For example, binary representation for the vertex **1** is 001. Using the rule it can be decoded as $\underline{u}_1\underline{u}_2\bar{u}_3$, which refers to the vertex generated by $u_1 = -1, u_2 = -1, u_3 = 1$. *Edges* are lines that connect vertices and that lie on $\partial(\Omega)$. In Figure 4.6 edges are denoted as **01**, **02**, ..., **67**. They are generated by varying only one of the m controls, while the remaining $m-1$ are at their limits, associated with

the two connected vertices. In the general case, the number of edges is equal to $2^{m-1} \binom{m}{1}$.

Two vertices are connected by an edge if and only if their binary representations differ in only one bit. For example, vertices **0** and **1** are connected by edge **01**, because their binary representations (000 for **0**, 001 for **1**) differ in only one (last) bit. In contrast, vertices **0** and **3** are not connected by edge, because their binary representations (000 for **0**, 011 for **3**) differ in more than one bit. *Facets* are plane surface on $\partial(\Omega)$ that contain two adjacent edges, i.e. two edges that have a common vertex. In Figure 4.6 facets are denoted as **0132**, **0451**, ..., **7623**. In the general case, the number of facets is equal to $2^{m-2} \binom{m}{2}$. A facet is defined as the set in the control space obtained by taking all

but two controls at their limits and varying the two free controls within their limits. For example, for facet **0132** binary representations of its vertices are: 000 for **0**, 001 for **1**, 011 for **3** and 010 for **2**. It can be seen that the first digit in these representation is fixed, while the other digits are not fixed, indicating that on this facet the first control is fixed at limit $u_1 = -1$, while the other two controls u_2 and u_3 are free to vary between their limits $[-1, 1]$. For higher dimension m it is possible to define d -dimensional facets in a similar way, as a subset of $\partial(\Omega)$ where $m - d$ controls are fixed at their limits and d controls are free to vary within their limits. A two-dimensional facet is a rectangular.

4.3.3 Nomenclature for attainable command set Φ

The control effectiveness matrix \mathbf{B} performs a linear transformation from the true control space \mathfrak{R}^m to the virtual control space \mathfrak{R}^k . The image of $\Omega \subset \mathfrak{R}^m$ is called the *attainable command set* and denoted by Φ . The attainable command set Φ is a convex polyhedron, whose boundary $\partial(\Phi)$ is the image of the facets of Ω . It is important to emphasize that

not all facets of Ω are mapped on the boundary $\partial(\Phi)$; most of these facets are mapped to the interior of Φ . If any k columns of \mathbf{B} are linearly independent (non-coplanar controls), then mapping \mathbf{B} is one-to-one on $\partial(\Phi)$, i.e. $(\forall \mathbf{v}^* \in \partial(\Phi))(\exists! \mathbf{u}^* \in \partial(\Omega)) \mathbf{v}^* = \mathbf{B}\mathbf{u}^*$. The attainable command set Φ for the control allocation problem in Example 4.1 is shown in Figure 4.7. Images of vertices from $\partial(\Omega)$ are called *vertices* (if they lie on $\partial(\Phi)$) or *nodes* (if they lie in the interior of Φ). In Figure 4.7, 1, 2, 3, 4, 5 and 6 are vertices, while 0 and 7 are nodes. In a similar way, images of edges from $\partial(\Omega)$ are called *edges* (if they lie on $\partial(\Phi)$) or *connections* (if they lie in the interior of Φ). In Figure 4.7, 13, 23, 26, 46, 45 and 15 are edges, while 01, 02, 04, 37, 57 and 67 are connections. Images of facets that lie on $\partial(\Omega)$ are called *facets* (if they lie on $\partial(\Phi)$) or *faces* (if they lie in the interior of Φ). In Example 4.1, Φ is two-dimensional and there are no faces or facets. If Φ is three-dimensional, facets or faces are parallelograms.

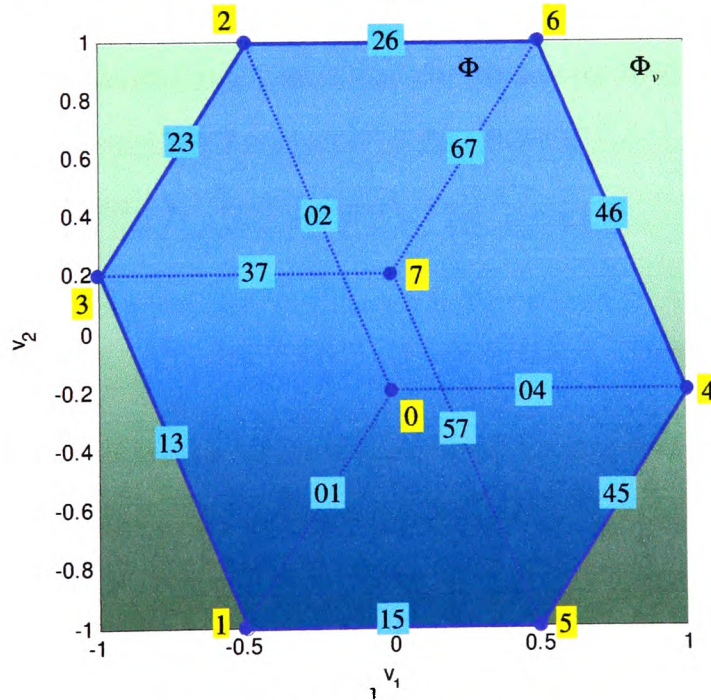


Figure 4.7 Attainable command set Φ .

4.3.4 Remarks

Two important issues are addressed in the following (Härkegård, 2003):

- *applicability* of the control allocation,
- *difficulties* for using the control allocation in real applications.

Applicability

The system needs to be separable as explained in section 4.2 in order to use control allocation. Applicability of the control allocation is bounded to the classes of linear and non-linear systems, which satisfy a criteria that will be determined in the following.

Consider first a *linear system*, described in state-space form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}_u\mathbf{u} \quad (4.13)$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $\mathbf{u} \in \mathbb{R}^m$ is the control input, $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{B}_u \in \mathbb{R}^{n \times m}$.

Assume that \mathbf{B}_u has rank $k < m$. Then \mathbf{B}_u has a nullspace (see Appendix B, Definition B.3) of dimension $m - k$ in which the control input can be perturbed without affecting $\dot{\mathbf{x}}$, i.e. there are several choices of control input that produces the same system dynamics. This type of redundancy can be resolved by control allocation.

Since \mathbf{B}_u is rank deficient, it can be factorised as

$$\mathbf{B}_u = \mathbf{B}_v\mathbf{B} \quad (4.14)$$

where matrices $\mathbf{B}_v \in \mathbb{R}^{n \times k}$ and $\mathbf{B} \in \mathbb{R}^{k \times m}$ both have rank k . Introducing the virtual control input

$$\mathbf{v} = \mathbf{B}\mathbf{u} \quad (4.15)$$

where $\mathbf{v} \in \mathbb{R}^k$, the system (4.13) can be rewritten as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}_v\mathbf{v} \quad (4.16)$$

Now, control design can be performed in two steps, as outlined in section 4.2.

The same idea can be used for *non-linear systems* of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{g}(\mathbf{x}, \mathbf{u})) \quad (4.17)$$

where $\mathbf{f} : \mathcal{R}^n \times \mathcal{R}^k \rightarrow \mathcal{R}^n$ and $\mathbf{g} : \mathcal{R}^n \times \mathcal{R}^m \rightarrow \mathcal{R}^k$, where $k < m$. Introducing the virtual control input

$$\mathbf{v} = \mathbf{g}(\mathbf{x}, \mathbf{u}) \quad (4.18)$$

where $\mathbf{v} \in \mathcal{R}^k$, the system (4.17) can be rewritten as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{v}) \quad (4.19)$$

Again, a two-step control design can be used, as described in section 4.2.

In order to solve (4.18), subject to actuator position constraints (4.7), a constrained non-linear programming method must be used at each time step. Since the control allocation has to be performed in real time, this may not be computationally feasible. One way to resolve this problem is to approximate (4.18) on a local basis with an affine mapping (Härkegård, 2003), which leads to the (locally) linear control allocation problem.

Difficulties

The most important difficulties for using the control allocation in real applications are *actuator dynamics* and *non-monotonic nonlinearities*.

Each actuator is a system with its own dynamics. The change of actuator's input will not be immediately reflected on its output, i.e. the output needs some time to achieve new steady-state value. For example, change of the set point (desired propeller velocity) of the thruster will generate a chain of changes inside the DC motor, which will lead to the acceleration or deceleration of the output (actual propeller velocity), until new steady-state is reached. Fortunately, in many cases actuator dynamics is much faster than the dynamics of the other parts of the system. In these cases the most common solution is that actuator dynamics are simply neglected. This will work as long as the closed-loop system is designed to be substantially slower than the actuator servo systems.

Non-monotonic nonlinearities in the mapping (4.18) constitute another important difficulty. If the mapping is not monotonic, the linearization can not be performed. One solution to the problem could be to limit controls and restrict the mapping, such that only monotonic part is used (Doman and Oppenheimer, 2002).

4.4 Control allocation methods

This section represents a survey of the most popular methods for control allocation appearing in the literature. Many of these methods correspond to different ways of computing the solution for a certain control allocation objective, rather than for different objectives. In this representation, the aim is to make a clear distinction between the main idea of the solution and how the solution can be computed numerically. All these methods will be demonstrated on the control allocation problem described in Example 4.1.

4.4.1 Optimisation based methods

Optimisation based methods rely on the following geometric interpretation of the control allocation problem (see page 4-9): find the intersection of Ω and \mathfrak{K} , where Ω is constrained control subset and \mathfrak{K} is a convex set, defined as intersection of hyperplanes $\mathbf{B}\mathbf{u} = \mathbf{v}$. If there are many solutions, choose the best one. If no solution exists, determine \mathbf{u} such that $\mathbf{B}\mathbf{u}$ is the best approximation of \mathbf{v} .

Problem statement

The l_p norm is used as a measure how good a solution (or approximation) is. In the general case, the optimal control input is given by the solution to a two-step optimisation problem (Härkegård, 2003):

$$\mathbf{u} = \arg \min_{\mathbf{u} \in \Psi} \|\mathbf{W}_u (\mathbf{u} - \mathbf{u}_p)\|_p \quad (4.20)$$

$$\Psi = \arg \min_{\mathbf{u} \leq \mathbf{u} \leq \mathbf{u}} \|\mathbf{W}_v (\mathbf{B}\mathbf{u} - \mathbf{v})\|_p \quad (4.21)$$

where \mathbf{u}_p represents preferred position of the actuators (preferred control input) and \mathbf{W}_u and \mathbf{W}_v are weighting matrices. The problem (4.20) - (4.21) can be interpreted as follows: Given Ψ , the set of feasible control inputs that minimise $\mathbf{B}\mathbf{u} - \mathbf{v}$ (weighted by \mathbf{W}_v), find the control input \mathbf{u} that minimizes $\mathbf{u} - \mathbf{u}_p$ (weighted by \mathbf{W}_u). In (4.20) - (4.21), \mathbf{u}_p , \mathbf{W}_u and \mathbf{W}_v are design parameters. The choice of \mathbf{u}_p may correspond, for example, to minimum control deflections in aerospace applications. \mathbf{W}_u can be used for actuator prioritisation, i.e. which actuator should be used primarily. In a similar way, \mathbf{W}_v allows for prioritisation among the virtual control inputs when the problem (4.20) - (4.21) has no exact solution.

Choice of norm

The l_2 norm is the most frequently used (Härkegård, 2003; Eberhardt and Ward, 1999; Enns, 1998; Snell, *et al.*, 1992). The reason is that the unconstrained minimum norm allocation problem

$$\min_{\mathbf{u}} \|\mathbf{u}\|_2 \quad (4.22)$$

subject to

$$\mathbf{B}\mathbf{u} = \mathbf{v} \quad (4.23)$$

has an explicit solution given by

$$\mathbf{u} = \mathbf{B}^+ \mathbf{v} \quad (4.24)$$

where $\mathbf{B}^+ = \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1}$ is the pseudoinverse of \mathbf{B} (see Lema B.1, Appendix B). A similar result can be derived for the general case ($\mathbf{u}_p \neq \mathbf{0}, \mathbf{W}_u \neq \mathbf{I}$), see Lema B.3. This fact is exploited by many authors, who developed different numerical schemes to compute the solution of (4.20) - (4.21).

Some authors used l_1 norm instead of l_2 in (4.20) - (4.21) (see, for example, Ikeda and Hood, 2000; Enns, 1998; Lindfors, 1993). A motivation for this choice is that, in general, a linear program can be solved faster than a quadratic one.

The following example illustrates the differences obtained by choosing different norms for solving the control allocation problem.

Example 4.2 (Choice of norm)

Consider the control allocation problem in Example 4.1 and let $\mathbf{v}_d = [-0.5 \ 0.6]^T$ is the desired virtual control input (Figure 4.8). Let the optimisation objective be given by (4.20) - (4.21), with $\mathbf{u}_p = \mathbf{0}$, $\mathbf{W}_u = \mathbf{I}_3$, $\mathbf{W}_v = \mathbf{I}_2$. Since $\mathbf{v}_d \in \Phi$, i.e. \mathbf{v}_d is attainable, the problem (4.20) - (4.21) can be reduced to

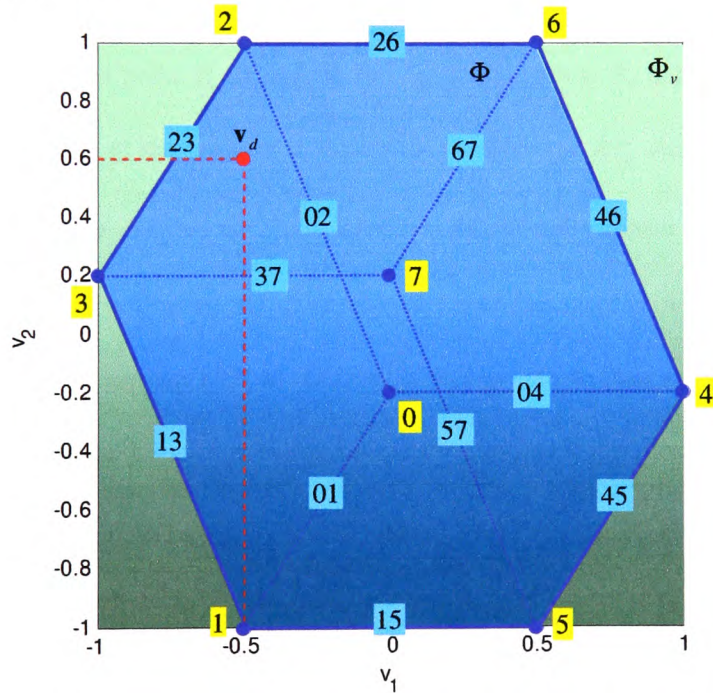


Figure 4.8 Position of the desired virtual control input \mathbf{v}_d in Φ .

$$\min_{\mathbf{u}} \|\mathbf{u}\|_p \quad (4.25)$$

subject to

$$\mathbf{B}\mathbf{u} = \begin{bmatrix} -0.5 \\ 0.6 \end{bmatrix} \quad (4.26)$$

$$\begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \leq \mathbf{u} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (4.27)$$

Substituting $v_1 = -0.5$ and $v_2 = 0.6$ in (4.11) yields

$$l: \quad \mathbf{p} = \begin{bmatrix} -\frac{46}{77} + \frac{1}{4}t \\ \frac{71}{77} + \frac{1}{5}t \\ -\frac{9}{77} + \frac{3}{10}t \end{bmatrix} \quad (4.28)$$

The intersection (solution set) \mathfrak{S} of l and Ω is a segment $\mathfrak{S} = P_1P_2$, where

$$P_1 = \begin{bmatrix} -1 & 3/5 & -3/5 \end{bmatrix}^T \quad (\text{for } t = t_1 = -123/77) \quad \text{and} \quad P_2 = \begin{bmatrix} -1/2 & 1 & 0 \end{bmatrix}^T \quad (\text{for } t = t_2 = 30/77).$$

The point P_1 belongs to the 0132 facet, while P_2 belongs to the 7623 facet. The solution of the problem is a point on the segment P_1P_2 that minimises $\|\mathbf{u}\|_p$.

Hence, the solution depends on choice of norm. Recall that a sphere $S(\mathbf{0}, r)_p$ is set of vectors $\mathbf{u} \in \mathbb{R}^m$ for which $\|\mathbf{u}\|_p \leq r$ (see Appendix B, Definition B.2). A family of spheres can be obtained by varying r . Now, the problem can be reformulated as follows:

Find r for which the segment P_1P_2 is tangent to the sphere $S(\mathbf{0}, r)_p$.

The solution set is given as set of point(s) where the segment touches the sphere. Figure 4.9 illustrates the situation for $p = 1$. The family of spheres is represented by concentric diamond-shaped bodies. If the radius of spheres is increased, for certain value r_1 the sphere $S(\mathbf{0}, r_1)_1$ will touch the segment. The touching point(s) is a solution, which minimises $\|\mathbf{u}\|_1$ and $\min_{\mathbf{u}} \|\mathbf{u}\|_1 = r_1$.

Case $p = 2$ is illustrated in Figure 4.10. Now the shape of the sphere is familiar from Euclidian metric. The same procedure, as described above, can be applied again: if the radius of spheres is increased, for certain value r_2 the sphere $S(\mathbf{0}, r_2)_2$ will touch the segment. The touching point is a solution, which minimise $\|\mathbf{u}\|_2$ and $\min_{\mathbf{u}} \|\mathbf{u}\|_2 = r_2$.

Choice of the weighting matrix \mathbf{W}_u

The weighting matrix \mathbf{W}_u is a design parameter typically used for actuator prioritisation. If all actuators have the same priority, then \mathbf{W}_u is equal to unity matrix. Otherwise, the weight of the actuator with less priority is increased. In this way it is possible to accommodate actuator faults by changing weighting matrix \mathbf{W}_u . This topic is discussed in section 5.2.5. The influence of the choice of the weighting matrix \mathbf{W}_u on the solution of (4.20) - (4.21) is illustrated in the following example.

Example 4.3 (Choice of the weighting matrix \mathbf{W}_u)

Consider the same control allocation problem as in Example 4.1 and Example 4.2. Let the optimisation objective be given by (4.20) - (4.21), with l_2 norm, $\mathbf{u}_p = \mathbf{0}$, $\mathbf{W}_v = \mathbf{I}_2$ and $\mathbf{W}_u = \text{diag}(w_1, w_2, w_3) = \text{diag}(1, 1, 2)$. Since $w_3 = 2 > w_1 = w_2 = 1$, the third actuator has a lower priority than other two. Change of the weight w_3 deforms the shape of spheres, shown in Figure 4.10, in such a way that they become flattened (compressed) in the u_3 direction, as indicated in Figure 4.11. In this way, the solution (touching point of a sphere with the segment \mathfrak{S}) exhibits lower contribution of the third control and higher participation of the other two controls, compared to previous case, when all actuators had the same priority. This property is used to accommodate partial thruster faults in section 5.2.5.

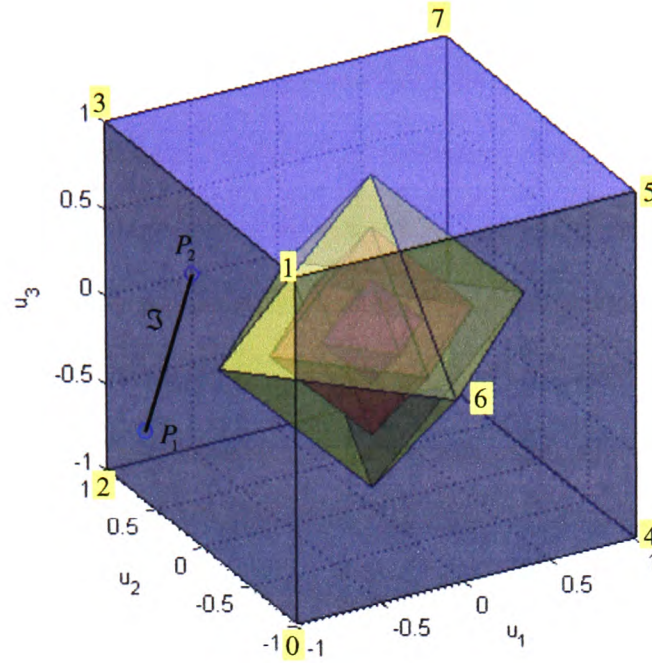


Figure 4.9 Family of spheres for l_1 norm and the solution segment P_1P_2 .

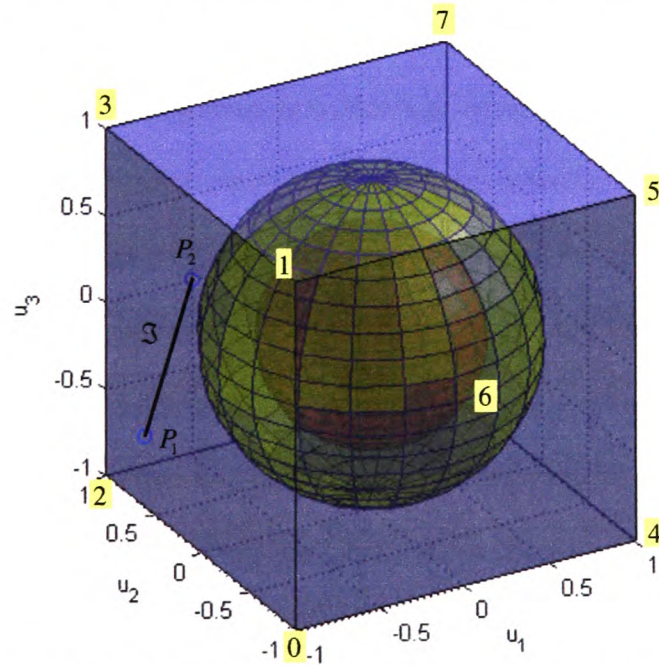


Figure 4.10 Family of spheres for l_2 norm and the solution segment P_1P_2 for the case $W_u = I_3$.

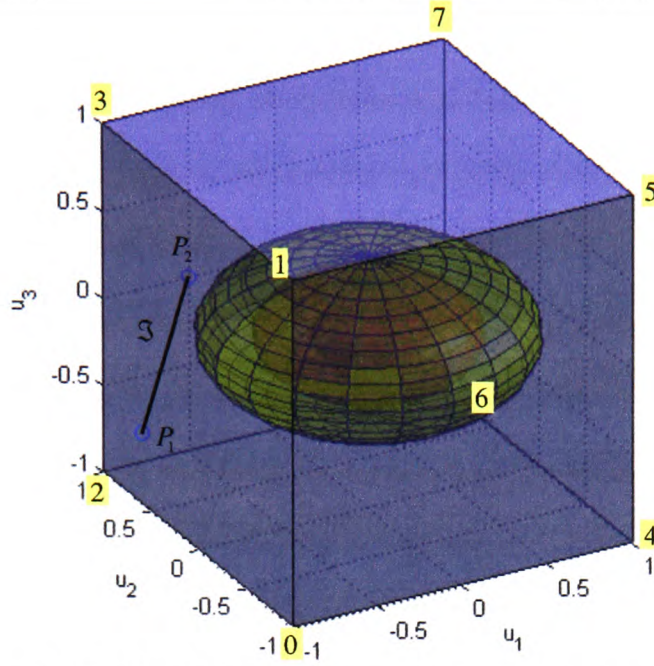


Figure 4.11 Family of spheres for l_2 norm and the solution segment P_1P_2 for the case $\mathbf{W}_u = \text{diag}(1,1,2)$.

Remarks

- In the case when \mathbf{W}_u is the unity matrix, the l_2 norm distributes the virtual control demand among the control inputs in uniform way, while the l_1 solution utilises as few control inputs as possible to satisfy the virtual control demand.
- The l_2 solution varies continuously with the parameters (elements) of \mathbf{B} , while the l_1 solution does not. Change in a parameter (element) b of \mathbf{B} will produce the change in slope of l . The l_2 solution will vary continuously with b , while it can be shown that the l_1 solution will have discontinuity for some value of $b = b^*$ and the solution in the break point b^* is not unique.
- If \mathbf{W}_u is non-singular, the problem $\min_{\mathbf{u}} \|\mathbf{W}_u \mathbf{u}\|_p$ has a unique solution for $p = 2$. For $p = 1$, this is not always the case, as discussed above. The reason lies in the fact that the sphere $S(\mathbf{0}, r)_2$ is strictly a convex set, while this is not the case for $S(\mathbf{0}, r)_1$.

Fixed-point method

One of methods for solving the problem with l_2 norm is the fixed-point method (Härkegård, 2003; Bodson; 2002; Burken, *et al.*, 2001; 1999). This method finds the control vector \mathbf{u} that minimises

$$J(\mathbf{u}) = (1 - \varepsilon) \|\mathbf{W}_v(\mathbf{B}\mathbf{u} - \mathbf{v})\|_2^2 + \varepsilon \|\mathbf{W}_u \mathbf{u}\|_2^2 \quad (4.29)$$

subject to

$$\underline{\mathbf{u}} \leq \mathbf{u} \leq \bar{\mathbf{u}} \quad (4.30)$$

where $|\varepsilon| < 1$. This problem is a special case of the weighted, mixed optimisation problem (2.9) where an l_2 norm is used, $\mathbf{y} = \mathbf{v}$ and $\mathbf{u}_p = \mathbf{0}$. The algorithm proceeds by iterating on the equation

$$\mathbf{u}_{k+1} = \text{sat}[(1 - \varepsilon)\eta \mathbf{B}^T \mathbf{Q}_1 \mathbf{v} - (\eta \mathbf{H} - \mathbf{I})\mathbf{u}_k] \quad (4.31)$$

where

$$\mathbf{Q}_1 = \mathbf{W}_v^T \mathbf{W}_v \quad (4.32)$$

$$\mathbf{Q}_2 = \mathbf{W}_u^T \mathbf{W}_u \quad (4.33)$$

$$\mathbf{H} = (1 - \varepsilon) \mathbf{B}^T \mathbf{Q}_1 \mathbf{B} + \varepsilon \mathbf{Q}_2 \quad (4.34)$$

$$\eta = 1 / \|\mathbf{H}\|_2 \quad (4.35)$$

and $\text{sat}(\mathbf{u})$ is the saturation function that clips the components of the vector \mathbf{u} to their limits. The condition for stopping the iteration process could be, for example, $|J(\mathbf{u}_{k+1}) - J(\mathbf{u}_k)| < \text{tol}$.

The fixed-point algorithm is very simple, and the most computations need to be performed only once before iterations start. Remarkably, the algorithm also provides an exact solution to the optimisation problem and it is guaranteed to converge. To improve the efficiency, Burken, *et al.* (2001; 1999) suggest selecting the initial point \mathbf{u}_0 as the true

Example 4.4 (Fixed-point method)

The individual iterations are shown in Table 4.1. The approximate solution is $\mathbf{u}_7 = [-0.5972 \quad 0.9221 \quad -0.1170]^T$ and $\mathbf{B}\mathbf{u}_7 = [-0.4999 \quad 0.6000]^T \approx \mathbf{v}_d = [-0.5 \quad 0.6]^T$.

In the light of the previous discussion, the norm $\|\mathbf{u}_7\|_2 = 1.1048$ can be interpreted as (approximate) radius of sphere (Figure 4.10) which touches the segment P_1P_2 .

The fixed-point method is used later as part of the accommodation process in the FDAS to find solutions that lie outside the feasible region for the pseudoinverse method i.e. the fixed-point method is activated only in case that the solution is not feasible using pseudoinverse (see Chapter 5 for more details; see also Example 4.10).

k	0	1	2	3	4	5	6	7
\mathbf{u}_k	$\begin{bmatrix} 0.0000 \\ 0.0000 \\ 0.0000 \end{bmatrix}$	$\begin{bmatrix} -0.4668 \\ 0.9056 \\ -0.2147 \end{bmatrix}$	$\begin{bmatrix} -0.5544 \\ 0.9166 \\ -0.1491 \end{bmatrix}$	$\begin{bmatrix} -0.5833 \\ 0.9203 \\ -0.1275 \end{bmatrix}$	$\begin{bmatrix} -0.5928 \\ 0.9215 \\ -0.1204 \end{bmatrix}$	$\begin{bmatrix} -0.5959 \\ 0.9219 \\ -0.1180 \end{bmatrix}$	$\begin{bmatrix} -0.5969 \\ 0.9220 \\ -0.1173 \end{bmatrix}$	$\begin{bmatrix} -0.5972 \\ 0.9221 \\ -0.1170 \end{bmatrix}$
$\ \mathbf{u}_k\ $	0.0000	1.0412	1.0816	1.0970	1.1023	1.1040	1.1046	1.1048
$J(\mathbf{u}_k)$	0.60999939	0.00967132	0.00104717	0.00011435	0.00001346	0.00000254	0.00000136	0.00000124

Table 4.1 Iterations of the fixed-point method.

Pseudoinverse methods

Most existing methods for l_2 -optimal control allocation can be classified as pseudoinverse methods. These methods exploit the fact that, if the actuator constraints are neglected, the general problem (4.20) - (4.21) reduces to

$$\min_{\mathbf{u}} \|\mathbf{W}_u(\mathbf{u} - \mathbf{u}_p)\|_2 \quad (4.36)$$

subject to

$$\mathbf{B}\mathbf{u} = \mathbf{v} \quad (4.37)$$

which, using Lema B.3 and assuming \mathbf{W}_u is non-singular, has the explicit solution

$$\begin{aligned} \mathbf{u} &= (\mathbf{I} - \mathbf{G}\mathbf{B})\mathbf{u}_p + \mathbf{G}\mathbf{v} \\ \mathbf{G} &= \mathbf{W}_u^{-1}(\mathbf{B}\mathbf{W}_u^{-1})^+ \end{aligned} \quad (4.38)$$

where $^+$ is pseudoinverse operator, see Appendix B.

Durham (1993) considered the case $\mathbf{u}_p = \mathbf{0}$ and showed that, in general, \mathbf{G} does not exist such that the pseudoinverse solution (4.38) is feasible for all attainable \mathbf{v} . That is, a set of commands attainable by pseudoinverse solution (4.38) is a subset of Φ .

Various ways to accommodate the pseudoinverse solution (4.38) to the actuator constraints have been proposed in the literature. The simplest alternative is to truncate the solution (4.38) by clipping those components that violate some constraints. Virnig and Bodden (1994) proposed a *Redistributed Pseudo Inverse* (RPI) scheme, where all control inputs that violate their limits in (4.38) are saturated and removed from the optimisation. Then, the control allocation problem is resolved with only the remaining control inputs as free variables. The RPI method is very simple and effective. However, it does not guarantee full utilisation of the actuators' capabilities and some bad choices made early in the iterations cannot be recovered later, as shown in the following example (Bodson, 2002):

Example 4.5 (Redistributed pseudoinverse method)

Consider the control allocation problem

- True control input is $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \in \mathbb{R}^4$ ($m = 4$),
- Desired virtual control input is $\mathbf{v}_d = \begin{bmatrix} 0 \\ 9 \\ 0 \end{bmatrix} \in \mathbb{R}^3$ ($k = 3$),
- Control effectiveness matrix is $\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$,

- Actuator position constraints are $\underline{\mathbf{u}} = \begin{bmatrix} -5 \\ -10 \\ -2 \\ -1 \end{bmatrix} \leq \mathbf{u} \leq \bar{\mathbf{u}} = \begin{bmatrix} 5 \\ 10 \\ 2 \\ 1 \end{bmatrix}$.

The unconstrained pseudoinverse solution (4.38) in the first iteration is $\mathbf{u}_1 = [0 \ 6 \ -3 \ 3]^T$. The control inputs that exceed their limits are u_3 and u_4 , since $u_3 = -3 < \underline{u}_3 = -2$ and $u_4 = 3 > \bar{u}_4 = 1$. The clipped solution is $\mathbf{u}_1^* = [0 \ 6 \ -2 \ 1]^T$. The control inputs u_3 and u_4 are saturated to their limits -2 and 1 , respectively, while u_1 and u_2 are free to vary. From $\mathbf{B}[u_1 \ u_2 \ -2 \ 1]^T = [u_1 \ 1+u_2 \ -1]^T \approx \mathbf{v}_d = [0 \ 9 \ 0]^T$ it can be easily found that $u_1 = 0$ and $u_2 = 8$. Hence, at the second and final iteration the control vector $\mathbf{u}_2 = [0 \ 8 \ -2 \ 1]^T$ is obtained, which satisfies the actuator constraints. But, this solution is just approximate, since $\mathbf{B}\mathbf{u}_2 = [0 \ 9 \ -1]^T \neq \mathbf{v}_d = [0 \ 9 \ 0]^T$. However, the desired vector \mathbf{v}_d can be attained using only the second control variable, i.e. the exact solution is $\mathbf{u} = [0 \ 9 \ 0 \ 0]^T$.

Example 4.6 (Pseudoinverse method)

The pseudoinverse solution (4.38) will be used for the control allocation problem, described in Example 4.1 and Example 4.2. Design parameters are $\mathbf{u}_p = [0 \ 0 \ 0]^T$ and $\mathbf{W}_u = \mathbf{I}_3$. Substituting design parameters in (4.38) yields $\mathbf{u} = [-0.5974 \ 0.9221 \ -0.1169]^T$. It can be easily verified that $\mathbf{u} \in P_1 P_2$, and, therefore, satisfy the actuator constraints. The solution \mathbf{u} is a limit of iterations \mathbf{u}_k (obtained in Example 4.4) when $k \rightarrow \infty$ (see Figure 4.12). The norm $\|\mathbf{u}\|_2 = 1.1049$ can be interpreted as the exact radius of sphere (Figure 4.10) which touches the segment $P_1 P_2$.

The pseudoinverse solution is a fast and efficient method to find the solution of the control allocation problem. The pseudoinverse is a member of a family of generalised inverses and is the one that yields minimum control energy. However, Durham (1993) showed that a general inverse is able to allocate controls only on a subset of Φ , i.e. there exist some points inside Φ (and on the boundary $\partial(\Phi)$) that are not feasible by general inverse. This important fact is demonstrated for pseudoinverse by the following example.

Example 4.7 (Pseudoinverse – partitioning of the virtual control space)

Pseudoinverse of \mathbf{B} from Example 4.1 is given by²

$$\mathbf{B}^+ = \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} = \begin{bmatrix} b_{11}^+ & b_{12}^+ \\ b_{21}^+ & b_{22}^+ \\ b_{31}^+ & b_{32}^+ \end{bmatrix} = \begin{bmatrix} 1.3506 & 0.1299 \\ -0.5195 & 1.1039 \\ -0.7792 & -0.8442 \end{bmatrix} \quad (4.39)$$

Recall that the attainable command set Φ is found in section 4.3.3 and shown in Figure 4.7. The pseudoinverse is mapping $\mathbf{u} = \mathbf{B}^+ \mathbf{v}$ from $k=2$ -dimensional virtual control space to $m=3$ -dimensional true control space. The virtual control space (square $\Phi_v = V_0 V_1 V_3 V_2$, Figure 4.13) is mapped by the pseudoinverse to a parallelogram $\Omega_v = U_0 U_1 U_3 U_2$ (Figure 4.14). The intersection of the parallelogram Ω_v with the cube Ω is a convex polygon $\Omega_p = R_{13} R_{15} R_{45} R_{46} R_{26} R_{23}$, where the vertex R_{ij} lies on the edge ij of Ω .

The following discussion will address these issues:

- Find $\Phi_p \subset \Phi_v$ such that $\mathbf{B}^+(\Phi_p) = \Omega_p$,

² If $\mathbf{W}_u \neq \mathbf{I}_3$, the pseudoinverse matrix is given by $\mathbf{B}_{\mathbf{W}_u}^+ = \mathbf{W}_u^{-1} \mathbf{B}^T (\mathbf{B} \mathbf{W}_u^{-1} \mathbf{B}^T)^{-1}$ (Lema B.2, Appendix

B). The change of weights in \mathbf{W}_u produce the change in slope of the parallelogram Ω_v and, consequently, the change in shape of Ω_p .



- Find $\mathbf{B}^+(\Phi)$.

In order to find Φ_p it is sufficient to find points P_{ij} such that

$$R_{ij} = \mathbf{B}^+(P_{ij}) \quad (4.40)$$

Let $P_{ij} = [v_1^{ij} \ v_2^{ij}]^T$ and $R_{ij} = [u_1^{ij} \ u_2^{ij} \ u_3^{ij}]^T$. Then (4.40) can be rewritten as

$$\begin{aligned} b_{11}^+ v_1^{ij} + b_{12}^+ v_2^{ij} &= u_1^{ij} \\ b_{21}^+ v_1^{ij} + b_{22}^+ v_2^{ij} &= u_2^{ij} \\ b_{31}^+ v_1^{ij} + b_{32}^+ v_2^{ij} &= u_3^{ij} \end{aligned} \quad (4.41)$$

The fact that $R_{ij} \in ij$ of Ω means that R_{ij} is bounded to the edge ij defined by vertices i and j , i.e. two coordinates (controls) of R_{ij} are fixed to their limits, while one is free to vary. Recall that the nomenclature for edges, introduced in section 4.3.2, enables easy detection of free and fixed controls for ij : binary representations of vertices i and j differ in only one bit and the position of this bit indicates a free control for ij . For example, the edge 13 is determined by vertices 1 and 3 . The binary representations of 1 (001) and 3 (011) differ in the second bit, which means that the free control is u_2^{13} , while fixed controls are $u_1^{13} = \underline{u}_1 = -1$ and $u_3^{13} = \bar{u}_3 = +1$. Once the fixed and free controls for ij are obtained, it is easy to find coordinates v_1^{ij} and v_2^{ij} of P_{ij} from (4.41) by removing the equation that corresponds to free control and replacing the right hand sides of other equations with corresponding limits for fixed controls. For example, for R_{13} system (4.41) can be rewritten as

$$\begin{aligned} b_{11}^+ v_1^{13} + b_{12}^+ v_2^{13} &= u_1^{13} \\ b_{21}^+ v_1^{13} + b_{22}^+ v_2^{13} &= u_2^{13} \\ b_{31}^+ v_1^{13} + b_{32}^+ v_2^{13} &= u_3^{13} \end{aligned} \quad (4.42)$$

A free control is u_2^{13} so the second equation is removed, while the right hand sides of the first and the third equation are replaced with $u_1^{13} = -1$ and $u_3^{13} = +1$, respectively:

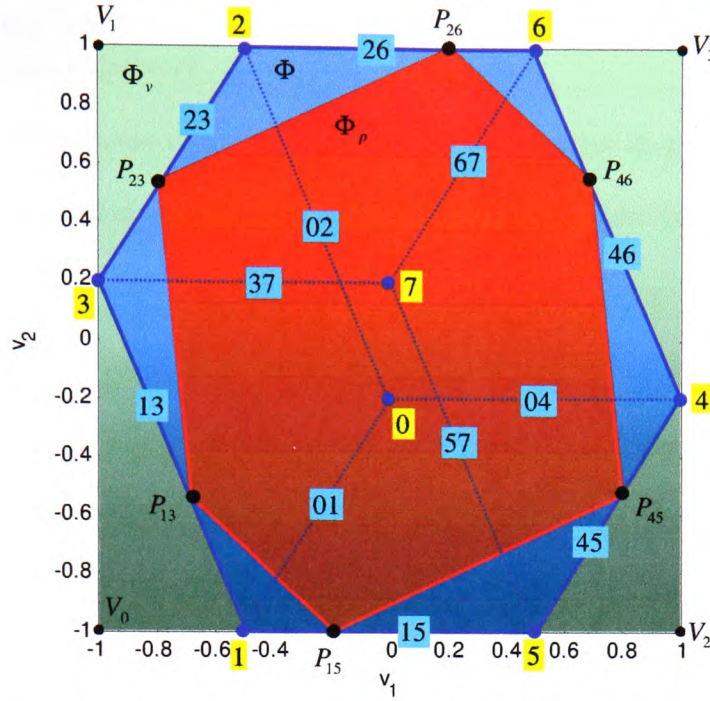


Figure 4.13 Partition of the virtual control space Φ_v : Φ_p (feasible region for pseudoinverse) and Φ (attainable command set).

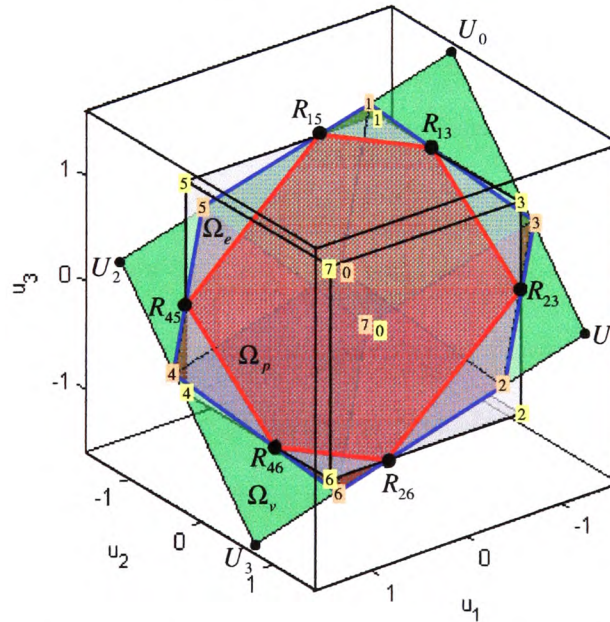


Figure 4.14 Images of partitions in the true control space: $\Omega_v = \mathbf{B}^+(\Phi_v)$ (image of Φ_v), $\Omega_p = \mathbf{B}^+(\Phi_p)$ (image of Φ_p) and $\Omega_e = \mathbf{B}^+(\Phi)$ (image of Φ).

$$\begin{aligned} b_{11}^+ v_1^{13} + b_{12}^+ v_2^{13} &= -1 \\ b_{31}^+ v_1^{13} + b_{32}^+ v_2^{13} &= +1 \end{aligned} \quad (4.43)$$

which yields the solution $v_1^{13} = -0.6875$ and $v_2^{13} = -0.5500$. Other vertices can be found in a similar way and results are shown in Table 4.2 (vertices P_{ij}) and Table 4.3 (vertices R_{ij}). Hence, the subset $\Phi_p \subset \Phi_v$ such that $\mathbf{B}^+(\Phi_p) = \Omega_p$ is a convex polygon $P_{13}P_{15}P_{45}P_{46}P_{26}P_{23}$, whose vertex P_{ij} lies on the edge ij of Φ (Figure 4.13).

The pseudoinverse image of Φ is a convex polygon $\Omega_e = \mathbf{B}^+(\Phi)$. Vertices 1, 2, 3, 4, 5 and 6 of Φ are mapped to vertices 1, 2, 3, 4, 5, 6 of Ω_e that lie outside Ω , while nodes 0 and 7 of Φ are mapped to nodes 0 and 7 of Ω_e that lie inside Ω . The virtual control space can be partitioned into three characteristic regions (Figure 4.13): Φ_p (polygon $P_{13}P_{15}P_{45}P_{46}P_{26}P_{23}$), $\Phi \setminus \Phi_p$ (union of triangles 1 $P_{15}P_{13}$, 3 $P_{13}P_{23}$, ..., 5 $P_{45}P_{15}$) and $\Phi_v \setminus \Phi$ (union of triangles V_0 1 3, ..., V_2 4 5). The pseudoinverse image of each partition lies inside parallelogram Ω_v (Figure 4.14). Table 4.4 displays cross-relation between these polygons in the virtual and the true control space.

Subset Φ_p represents a part of the virtual control space Φ_v attainable (feasible) by pseudoinverse. That is, if $\mathbf{v}' \in \Phi_p$, then $\mathbf{u}' = \mathbf{B}^+ \mathbf{v}' \in \Omega$ and $\|\mathbf{u}'\|_2 = \arg \min_{\mathbf{B}\mathbf{u}=\mathbf{v}'} \|\mathbf{u}\|$. In other words, if the virtual control input lies in Φ_p , then the pseudoinverse solution is feasible and has a minimal l_2 norm of all other solutions. Otherwise, if $\mathbf{v}' \notin \Phi_p$, then $\mathbf{u}' = \mathbf{B}^+ \mathbf{v}' \notin \Omega$, which means that \mathbf{v}' is not feasible with pseudoinverse, although some other choice of general inverse or application of other methods, like direct allocation, can make it feasible. This aspect of the relationship between the position of the virtual control input and the feasibility of the solution is important and discussed in the following example.

	P_{23}	P_{45}	P_{13}	P_{46}	P_{15}	P_{26}
v_1	-0.7917	0.7917	-0.6875	0.6875	-0.2000	0.2000
v_2	0.5333	-0.5333	-0.5500	0.5500	-1.0000	1.0000

Table 4.2 Vertices of Φ_p .

	R_{23}	R_{45}	R_{13}	R_{46}	R_{15}	R_{26}
u_1	-1.0000	1.0000	-1.0000	1.0000	-0.4000	0.4000
u_2	1.0000	-1.0000	-0.2500	0.2500	-1.0000	1.0000
u_3	0.1667	-0.1667	1.0000	-1.0000	1.0000	-1.0000

Table 4.3 Vertices of $\Omega_p = \mathbf{B}^+(\Phi_p)$.

Virtual Control Space		True Control Space	
Partition	Polygon	Partition	Polygon
Φ_p	$P_{13}P_{15}P_{45}P_{46}P_{26}P_{23}$	Ω_p	$R_{13}R_{15}R_{45}R_{46}R_{26}R_{23}$
$\Phi \setminus \Phi_p$	1 $P_{15}P_{13}$	$\Omega_e \setminus \Omega_p$	1 $R_{15}R_{13}$
	3 $P_{13}P_{23}$		3 $R_{13}R_{23}$
	2 $P_{23}P_{26}$		2 $R_{23}R_{26}$
	6 $P_{26}P_{46}$		6 $R_{26}R_{46}$
	4 $P_{46}P_{45}$		4 $R_{46}R_{45}$
	5 $P_{45}P_{15}$		5 $R_{45}R_{15}$
$\Phi_v \setminus \Phi$	V_0 1 3	$\Omega_v \setminus \Omega_e$	U_0 1 3
	V_1 3 2		U_1 3 2
	V_3 6 4		U_3 6 4
	V_2 4 5		U_2 4 5

Table 4.4 Cross-relation between the virtual and the true control space.

Example 4.8 (Pseudoinverse – geometric interpretation of solution)

Consider again the control allocation problem from Example 4.1 and let $S = [v_1 \ v_2]^T$ denotes an arbitrary point from the virtual control space Φ_v . Recall that a total solution set \mathfrak{S} is given by the intersection of the line l (4.11) and Ω . When a point S moves inside Φ_v , the corresponding line l moves in the true control space. For a given S ,

pseudoinverse will select the solution from \mathfrak{S} where the line l intersects the parallelogram Ω_v . Three characteristic cases are possible, regarding the position of S relative to the partitions of Φ_v (Figure 4.15):

1. $S = S_1 \in \Phi_p$,
2. $S = S_2 \in \Phi \setminus \Phi_p$,
3. $S = S_3 \in \Phi_v \setminus \Phi$.

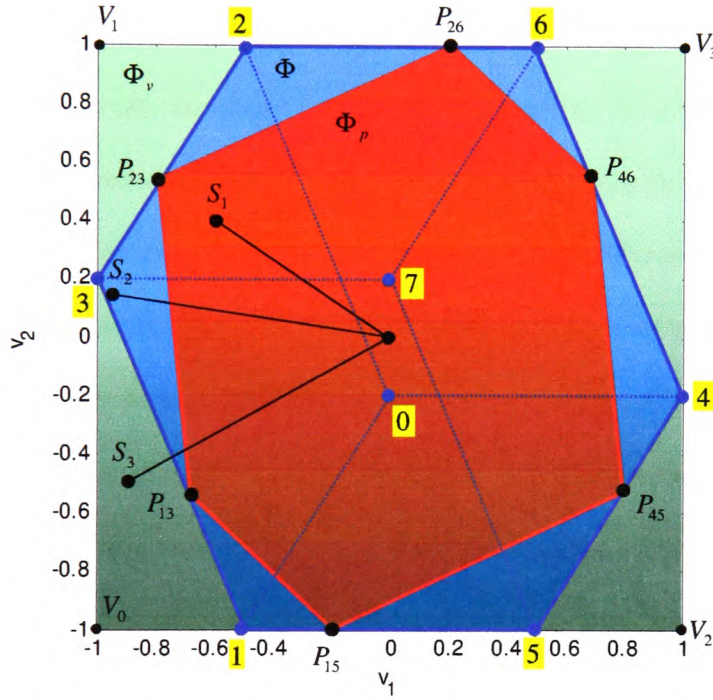


Figure 4.15 Three typical cases for position of virtual control inputs relative to Φ_p and Φ .

Case 1: In this case, point S_1 lies inside Φ_p and the pseudoinverse solution is $T_1 = \mathbf{B}^+(S_1)$ that lies inside Ω_p (Figure 4.16). A solution set is a segment $\mathfrak{S}_1 = l_1 \cap \Omega$. This segment intersects the parallelogram Ω_v in the point T_1 , $T_1 = \mathfrak{S}_1 \cap \Omega_v$. From all solutions in \mathfrak{S} , the solution T_1 , selected by pseudoinverse, is optimal in l_2 sense.

For example, if $S_1 = [-0.6 \ 0.4]^T$, then the solution set is a segment $\mathfrak{S}_1 = P_1 P_2 = l_1 \cap \Omega$, $P_1 = [-1 \ 14/25 \ -4/25]^T$, $P_2 = [-9/20 \ 1 \ 1/2]^T$ (Figure 4.16). The pseudoinverse solution ($T_1 = \mathbf{B}^+(S_1) = [-0.7584 \ 0.7532 \ 0.1299]^T$) represents the point where the segment $P_1 P_2$ intersects the parallelogram Ω_v . This solution is feasible, since it belongs to Ω .

Case 2: In this case, point S_2 lies outside Φ_p , but inside Φ . The image $T_2 = \mathbf{B}^+(S_2)$ lies outside Ω_p , but inside Ω_e (Figure 4.17). Geometrically, a solution set is segment $\mathfrak{S}_2 = l_2 \cap \Omega$ that does not intersect with Ω_v , which means that the pseudoinverse solution T_2 lies on l_2 but outside \mathfrak{S}_2 , i.e. $T_2 \notin \Omega$. Hence, the virtual control input S_2 is unfeasible (unattainable) by pseudoinverse, but, because $\mathfrak{S}_2 \neq \emptyset$, some other methods (like direct allocation) are able to allocate solution from \mathfrak{S}_2 .

For example, if $S_2 = [-0.9375 \ 0.1600]^T$, then the solution set is a segment $\mathfrak{S}_2 = P_1 P_2 = l_2 \cap \Omega$, $P_1 = [-1 \ 43/50 \ 89/100]^T$, $P_2 = [-9/20 \ 1 \ 1/2]^T$ (Figure 4.17). The pseudoinverse solution ($T_2 = \mathbf{B}^+(S_2) = [-1.2455 \ 0.6636 \ 0.5955]^T$) represents the point where the line l_2 intersects the parallelogram Ω_v . This solution is unfeasible, since it lies on the line l_2 outside \mathfrak{S}_2 and does not belong to Ω .

Case 3: Finally, in the last case point S_3 lies outside Φ , but inside Φ_v (Figure 4.15). The image $T_3 = \mathbf{B}^+(S_3)$ lies outside Ω_e , but inside Ω_v (Figure 4.18). In this case line l_3 does not intersect with Ω , i.e. $\mathfrak{S}_3 = l_3 \cap \Omega = \emptyset$ and the exact solution of the problem does not exist. The pseudoinverse solution T_3 is unfeasible, since it does not belong to Ω .

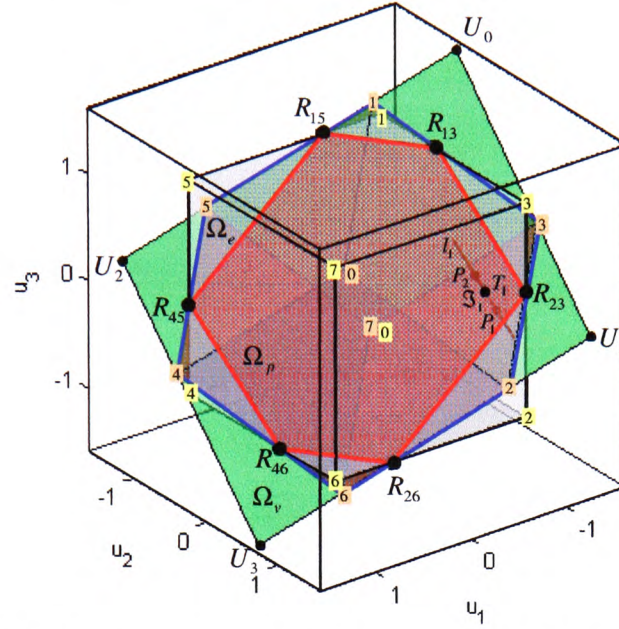


Figure 4.16 Case $S_1 \in \Phi_p$ yields to the pseudoinverse solution $T_1 \in \Omega_p \subset \Omega$ that is optimal in

l_2 sense.

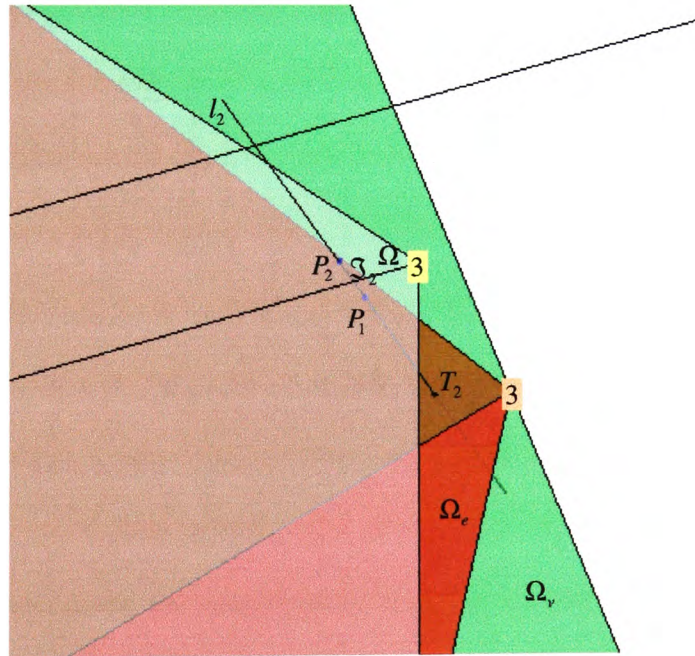


Figure 4.17 Case $S_2 \in \Phi \setminus \Phi_p$ yields to the unfeasible pseudoinverse solution $T_2 \in \Omega_e \setminus \Omega_p$ that

lies outside Ω .

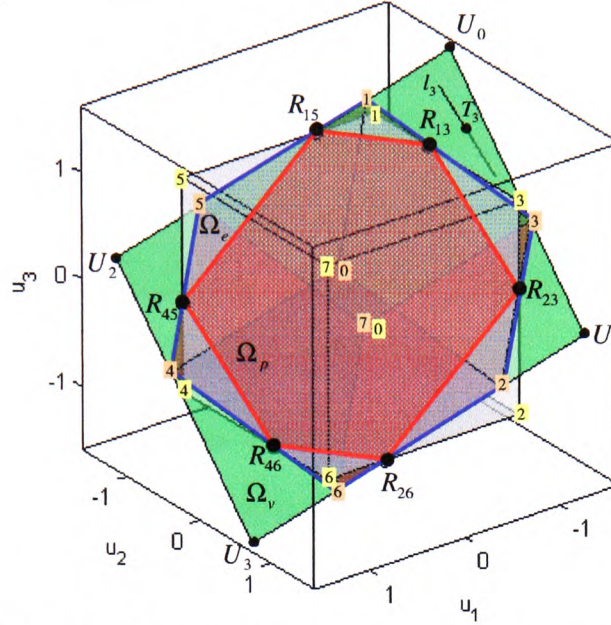


Figure 4.18 Case $S_3 \in \Phi_v \setminus \Phi$ yields to the unfeasible pseudoinverse solution $T_3 \in \Omega_v \setminus \Omega_e$ that lies outside Ω .

For example, if $S_3 = [-0.9000 \quad -0.5000]^T$, then line $l = l_3$ (4.11) is given by

$$l_3: \quad \mathbf{p} = \begin{bmatrix} -\frac{493}{385} + \frac{1}{4}t \\ \frac{13}{154} + \frac{1}{5}t \\ \frac{173}{154} + \frac{3}{10}t \end{bmatrix} \quad (4.44)$$

The line l_3 does not intersect Ω and the exact solution of the control allocation problem does not exist (Figure 4.18). However, intersection of l_3 and Ω_v is the pseudoinverse solution $T_3 = \mathbf{B}^+(S_3) = [-1.2805 \quad -0.0844 \quad 1.1234]^T$. This solution lies outside Ω and is not feasible.

This example demonstrated that pseudoinverse is able to allocate the exact solution (optimal in l_2 sense) only if the virtual control input \mathbf{v} lies in Φ_p . Otherwise, if

$\mathbf{v} \in \Phi \setminus \Phi_p$, the pseudoinverse finds solutions that lie outside Ω , i.e. that violate control constraints. The approximation of these solutions is the topic of the following example.

Example 4.9 (Pseudoinverse – approximation of unfeasible solutions)

This example is a continuation of discussion from Example 4.8. If a virtual control input \mathbf{v} lies outside Φ_p (for example, S_2 and S_3 in Figure 4.15), then the pseudoinverse solution $\mathbf{u} = \mathbf{B}^+ \mathbf{v}$ is unfeasible and lies outside Ω , i.e. it violates control constraints (T_2 in Figure 4.17 and T_3 in Figure 4.18). In this case it is necessary to approximate unfeasible $\mathbf{u} \notin \Omega$ with feasible $\mathbf{u}^ \in \Omega$ such that $\mathbf{B}\mathbf{u}^* \approx \mathbf{v}$.*

Definition 4.1 (Approximation error)

The approximation error is defined as $\mathbf{e} = \mathbf{v} - \mathbf{v}^$, where \mathbf{v} is the virtual control input, $\mathbf{v}^* = \mathbf{B}\mathbf{u}^*$ is an approximation of \mathbf{v} and \mathbf{u}^* is an approximation of $\mathbf{u} = \mathbf{B}^+ \mathbf{v}$. In order to be able to compare different approximations, two scalar errors are introduced: direction error $\theta = \arccos \frac{\mathbf{v}^T \cdot \mathbf{v}^*}{\|\mathbf{v}\|_2 \|\mathbf{v}^*\|_2}$ and magnitude error $\|\mathbf{e}\|_2 = \|\mathbf{v} - \mathbf{v}^*\|_2$. The direction error represents the angle between \mathbf{v} and \mathbf{v}^* , while the magnitude error represents the module of the approximation error vector \mathbf{e} (Figure 4.19). In the case when $\theta = 0$, the approximation \mathbf{v}^* preserves the direction of the original vector \mathbf{v} .*

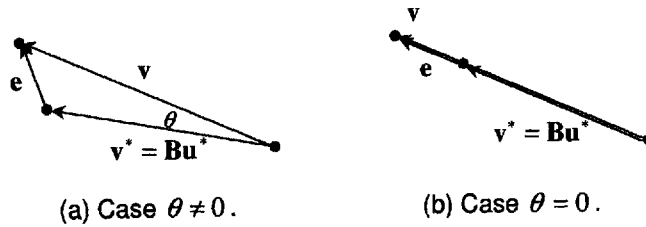


Figure 4.19 Approximation error $\mathbf{e} = \mathbf{v} - \mathbf{v}^* = \mathbf{v} - \mathbf{B}\mathbf{u}^*$.

Two common approximations (truncation and scaling) are explained in the following:

Truncation (T-approximation): In this case approximation \mathbf{u}_i^* is obtained from \mathbf{u} by truncating (clipping) all controls that exceed their control constraints. For example,

the first control u_{21} of the pseudoinverse solution $\mathbf{u}_2 = T_2 = \begin{bmatrix} \underbrace{-1.2455}_{u_{21}} & \underbrace{0.6636}_{u_{22}} & \underbrace{0.5955}_{u_{23}} \end{bmatrix}^T$

for the virtual control input $\mathbf{v}_2 = S_2 = [-0.9375 \ 0.1600]^T \in \Phi \setminus \Phi_\rho$ (see Example 4.8)

violates the constraint \underline{u}_1 ($u_{21} < \underline{u}_1 = -1$) and must be clipped, which leads to T-

approximation $\mathbf{u}_{2t}^* = T_{2t}^* = [-1 \ 0.6636 \ 0.5955]^T$ (Figure 4.20). This approximation

generates $\mathbf{v}_{2t}^* = S_{2t}^* = \mathbf{B}\mathbf{u}_{2t}^* = [-0.8148 \ 0.1600]^T$ (Figure 4.21). Vector \mathbf{v}_{2t}^* is not

colinear with \mathbf{v}_2 and the direction error is $\theta_{2t} = \arccos \frac{\mathbf{v}_2^T \cdot \mathbf{v}_{2t}^*}{\|\mathbf{v}_2\|_2 \|\mathbf{v}_{2t}^*\|_2} = 1.4249^\circ$, while the

magnitude error is $\|\mathbf{e}_{2t}\|_2 = \|\mathbf{v}_2 - \mathbf{v}_{2t}^*\|_2 = 0.1227$.

The same procedure can be applied for the case when the virtual control input lies outside Φ . For example, let $\mathbf{v}_3 = S_3 = [-0.9000 \ -0.5000]^T \in \Phi_v \setminus \Phi$. Controls u_{31} and

u_{33} of the unfeasible pseudoinverse solution $\mathbf{u}_3 = T_3 = \begin{bmatrix} \underbrace{-1.2805}_{u_{31}} & \underbrace{-0.0844}_{u_{32}} & \underbrace{1.1234}_{u_{33}} \end{bmatrix}^T$,

found in Example 4.8, violates their constraints and must be clipped ($u_{31} = \underline{u}_1 = -1$ and

$u_{33} = \bar{u}_3 = 1$), which leads to T-approximation $\mathbf{u}_{3t}^* = T_{3t}^* = [-1 \ -0.0844 \ 1]^T$.

This approximation produces $\mathbf{v}_{3t}^* = S_{3t}^* = \mathbf{B}\mathbf{u}_{3t}^* = [-0.7289 \ -0.4506]^T$ (Figure 4.21),

which lies on the boundary $\partial(\Phi)$, since \mathbf{u}_{3t}^* lies on the boundary $\partial(\Omega)^3$. Again, the

³ Since every $k \times k = 2 \times 2$ partition of \mathbf{B} is non-singular, equation $\mathbf{v}' = \mathbf{B}\mathbf{u}'$, $\mathbf{u}' \in \partial(\Phi)$ has unique solution $\mathbf{v}' \in \partial(\Omega)$. In this case controls are said to be independent (see section 4.4.2).

approximation \mathbf{v}_{3t}^* is not colinear with \mathbf{v}_3 and the direction error is

$$\theta_{3t} = \arccos \frac{\mathbf{v}_3^T \cdot \mathbf{v}_{3t}^*}{\|\mathbf{v}_3\|_2 \|\mathbf{v}_{3t}^*\|_2} = 2.6724^\circ, \text{ while the magnitude error is } \|\mathbf{e}_{3t}\|_2 = \|\mathbf{v}_3 - \mathbf{v}_{3t}^*\|_2 = 0.1781.$$

Scaling (S-approximation): In this case approximation \mathbf{u}_s^* is obtained from \mathbf{u} by scaling all controls by factor f such that $\mathbf{u}_s^* = f\mathbf{u} \in \partial(\Omega_p)$. Geometrically, point $T_s^* = \mathbf{u}_s^*$ is the intersection of \mathbf{u} and $\partial(\Omega_p)$ (Figure 4.20). But $T_s^* \in \partial(\Omega_p) \Rightarrow T_s^* \in \partial(\Omega)$, since $\partial(\Omega_p) \subset \partial(\Omega)$. In the general case, the scaling factor f is calculated from⁴

$$f = \min \left(\frac{1}{\max \left[\max_{i \in \{1, \dots, m\}} \left(\frac{u_i}{\underline{u}_i} \right), \max_{j \in \{1, \dots, m\}} \left(\frac{u_j}{\overline{u}_j} \right) \right]}, 1 \right) \quad (4.45)$$

If $\mathbf{u} \in \Omega$ then $f = 1$. Otherwise, $f < 1$. For example, since Ω is symmetrical about $\mathbf{u} = \mathbf{0}$, the scaling factor f_2 for $\mathbf{u}_2 = T_2 = [u_{21} \ u_{22} \ u_{23}]^T \in \Omega_e \setminus \Omega$ is found using simplified (4.45):

$$\begin{aligned} f_2 &= \min \left(\frac{1}{\max \left[\left| \frac{u_1}{u_{1m}} \right|, \left| \frac{u_2}{u_{2m}} \right|, \left| \frac{u_3}{u_{3m}} \right| \right]}, 1 \right) = \min \left(\frac{1}{\max \left[\left| \frac{-1.2455}{+1} \right|, \left| \frac{0.6636}{+1} \right|, \left| \frac{0.5955}{+1} \right| \right]}, 1 \right) \\ &= \min \left(\frac{1}{1.2455}, 1 \right) = 0.8029 \end{aligned} \quad (4.46)$$

⁴ It is assumed that lower and upper limits have opposite signs, i.e. that the origin $\mathbf{u} = \mathbf{0}$ belongs to Ω . In special case, when Ω is symmetrical about the origin ($\overline{u}_i = -\underline{u}_i = u_{mi}$), (4.45) is simplified to

$$f = \min \left(\frac{1}{\max_{i \in \{1, 2, \dots, m\}} |u_i / u_{mi}|}, 1 \right).$$

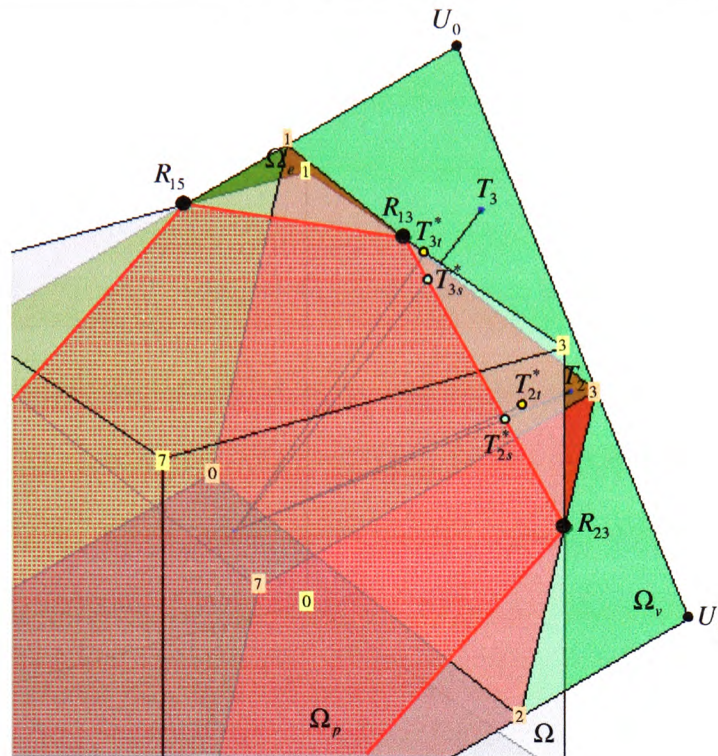


Figure 4.20 Approximation of unfeasible pseudoinverse solutions in the true control space.

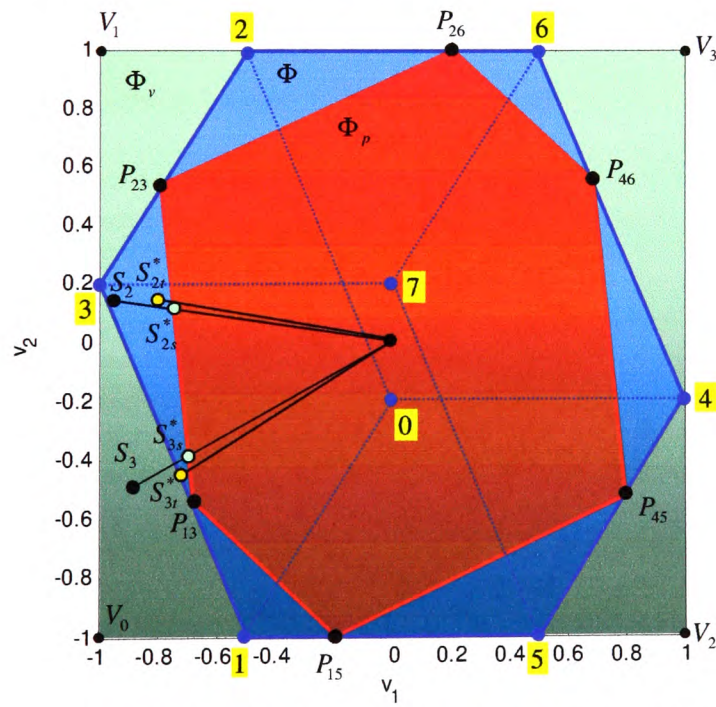


Figure 4.21 Results of approximations in the virtual control space.

which leads to a S -approximation $\mathbf{u}_{2s}^* = T_{2s}^* = f\mathbf{u}_2 = [-1 \ 0.5328 \ 0.4781]^T \in \partial(\Omega_p)$ (Figure 4.20). This approximation generates $\mathbf{v}_{2s}^* = S_{2s}^* = \mathbf{B}\mathbf{u}_{2s}^* = [-0.7527 \ 0.1285]^T \in \partial(\Phi_p)$

(Figure 4.21). Vector \mathbf{v}_{2s}^* is colinear with \mathbf{v}_2 and the direction error is

$$\theta_{2s} = \arccos \frac{\mathbf{v}_2^T \cdot \mathbf{v}_{2s}^*}{\|\mathbf{v}_2\|_2 \|\mathbf{v}_{2s}^*\|_2} = 0.0^\circ, \text{ while the magnitude error is } \|\mathbf{e}_{2s}\|_2 = \|\mathbf{v}_2 - \mathbf{v}_{2s}^*\|_2 = 0.1874.$$

The same procedure can be applied for the case when the virtual control input lies outside Φ , like $\mathbf{v}_3 = S_3 = [-0.9000 \ -0.5000]^T \in \Phi_v \setminus \Phi$ in Figure 4.21. The S -approximation of the unfeasible pseudoinverse solution $\mathbf{u}_3 = T_3 \in \Omega_v \setminus \Omega_e$ is

$$\mathbf{u}_{3s}^* = T_{3s}^* = f_3 \mathbf{u}_3 = 0.7809 [-1.2805 \ -0.0844 \ 1.1234]^T = [-1 \ -0.0659 \ 0.8773]^T \in \partial(\Omega_p)$$

(Figure 4.20), which yields $\mathbf{v}_{3s}^* = S_{3s}^* = \mathbf{B}\mathbf{u}_{3s}^* = [-0.7028 \ -0.3905]^T$ (Figure 4.21),

which lies on the boundary $\partial(\Phi_p)$, since \mathbf{u}_{3s}^* lies on the boundary $\partial(\Omega_p)$. As in the previous case, the approximation \mathbf{v}_{3s}^* is colinear with \mathbf{v}_3 and the direction error is

$$\theta_{3s} = \arccos \frac{\mathbf{v}_3^T \cdot \mathbf{v}_{3s}^*}{\|\mathbf{v}_3\|_2 \|\mathbf{v}_{3s}^*\|_2} = 0.0^\circ, \text{ while the magnitude error is } \|\mathbf{e}_{3s}\|_2 = \|\mathbf{v}_3 - \mathbf{v}_{3s}^*\|_2 = 0.2255.$$

Remarks

- Pseudoinverse \mathbf{B}^+ is a special case of generalised inverse \mathbf{P} (section 2.9.3). The shape of Φ_p depends on the selection of \mathbf{P} . Durham (1993) showed that the equation $\mathbf{u} = \mathbf{P}\mathbf{v}$ can be satisfied at no more than $(m-k)k$ pre-selected points on the boundary $\mathbf{v} \in \partial(\Phi)$. This means that from all vertices of Φ_p no more than $(m-k)k$ can be selected arbitrarily, while other vertices can be obtained by some linear combination of controls that correspond to the selected vertices.

-
- Each selection of no more than $(m-k)k$ points on the boundary $\partial(\Phi)$ yields to unique partition of matrix \mathbf{P} . Durham (1993) demonstrated one way to find partitions of \mathbf{P} using nullspace $\mathcal{N}[\mathbf{PB-I}]$.
 - Durham (1993) also demonstrated a method to find the "best" generalised inverse, that is, the generalised inverse which maximise the area or volume of Φ_p inside Φ , without violating any control constraints. This is clearly the same as minimising the difference between the area or volume of the boundary $\partial(\Phi)$ and Φ_p within constraints. The "price" that is paid for maximising Φ_p is that the solutions are not optimal in l_2 sense.
 - T -approximation \mathbf{u}_t^* of the unfeasible pseudoinverse solution \mathbf{u} lies in $(\Phi \setminus \Phi_p) \cup \partial(\Phi)$ and introduces direction error $\theta_t \neq 0$, i.e. vectors \mathbf{v} and \mathbf{v}_t^* have not the same direction. At the same time, the S -approximation \mathbf{u}_s^* lies on $\partial(\Phi_p)$ and the direction error θ_s is always zero, i.e. vectors \mathbf{v} and \mathbf{v}_s^* always have the same direction, but the magnitude error $\|\mathbf{e}_s\|_2$ is greater than $\|\mathbf{e}_t\|_2$.
 - The fixed-point method (page 4-24) is able to improve the T - or S -approximation of the unfeasible pseudoinverse solution \mathbf{u} . Approximations \mathbf{u}_t^* or \mathbf{u}_s^* can be used as the initial iteration \mathbf{u}_0 and the algorithm will find the solution \mathbf{u}_f^* such that $\mathbf{v}_f^* = \mathbf{B}\mathbf{u}_f^*$ is a better approximation of \mathbf{v} than \mathbf{v}_t^* or \mathbf{v}_s^* . This is the main idea of the hybrid approach for control allocation used in the FDAS:

If the virtual control input \mathbf{v} lies inside Φ_p , use the pseudoinverse solution $\mathbf{u} = \mathbf{B}^+ \mathbf{v}$ that is optimal in l_2 sense. Otherwise, if $\mathbf{v} \in \Phi_v \setminus \Phi_p$ activate the fixed-point iterations to find \mathbf{u}_f^* , using the T -approximation \mathbf{u}_t^* , S -approximation \mathbf{u}_s^* or control vector from previous sample $\mathbf{u}(t-T)$ as an initial point for iteration. Depending on design parameters of the fixed-point method, the solution $\mathbf{v}_f^* = \mathbf{B}\mathbf{u}_f^*$ obtained in this way is (almost) exact (if $\mathbf{v} \in \Phi \setminus \Phi_p$) or approximate (if $\mathbf{v} \in \Phi_v \setminus \Phi$).

Example 4.10 (Hybrid approach for control allocation)

In this example, the fixed-point method is used to improve T - and S -approximation of unfeasible pseudoinverse solutions from Example 4.9. Design parameters are $\mathbf{W}_u = \mathbf{I}_3$, $\mathbf{W}_v = \mathbf{I}_2$, $\varepsilon = 10^{-6}$ and $\text{tol} = 10^{-6}$. Results are shown in Table 4.5 and Table 4.6.

Initial point	# of iterations	Last iteration	Limit	Obtained virtual control input	Desired virtual control input	Direction error	Magnitude error
\mathbf{u}_0	k	\mathbf{u}_{fk}	\mathbf{u}_f	$\mathbf{v}_{fk} = \mathbf{B}\mathbf{u}_{fk}$	\mathbf{v}	θ_k	$\ \mathbf{e}_k\ _2$
\mathbf{u}_{2t}^*	19	$\begin{bmatrix} -1.0000 \\ 0.8585 \\ 0.8874 \end{bmatrix}$	$\begin{bmatrix} -1.0000 \\ 0.8600 \\ 0.8900 \end{bmatrix}$	$\begin{bmatrix} -0.9365 \\ 0.1601 \end{bmatrix}$	$\begin{bmatrix} -0.9375 \\ 0.1600 \end{bmatrix}$	0.0181°	0.0010
\mathbf{u}_{2s}^*	20	$\begin{bmatrix} -1.0000 \\ 0.8582 \\ 0.8870 \end{bmatrix}$	$\begin{bmatrix} -1.0000 \\ 0.8600 \\ 0.8900 \end{bmatrix}$	$\begin{bmatrix} -0.9363 \\ 0.1601 \end{bmatrix}$	$\begin{bmatrix} -0.9375 \\ 0.1600 \end{bmatrix}$	0.0208°	0.0012

Table 4.5 Iterations of the fixed-point method for $\mathbf{v}_2 \in \Phi \setminus \Phi_p$.

Initial point	# of iterations	Last iteration	Limit	Obtained virtual control input	Desired virtual control input	Direction error	Magnitude error
\mathbf{u}_0	k	\mathbf{u}_{fk}	\mathbf{u}_f	$\mathbf{v}_{fk} = \mathbf{B}\mathbf{u}_{fk}$	\mathbf{v}	θ_k	$\ \mathbf{e}_k\ _2$
\mathbf{u}_{3t}^*	3	$\begin{bmatrix} -1.0000 \\ -0.0535 \\ 1.0000 \end{bmatrix}$	$\begin{bmatrix} -1.0000 \\ -0.0533 \\ 1.0000 \end{bmatrix}$	$\begin{bmatrix} -0.7366 \\ -0.4321 \end{bmatrix}$	$\begin{bmatrix} -0.9000 \\ -0.5000 \end{bmatrix}$	1.3431°	0.1769
\mathbf{u}_{3s}^*	5	$\begin{bmatrix} -1.0000 \\ -0.0533 \\ 1.0000 \end{bmatrix}$	$\begin{bmatrix} -1.0000 \\ -0.0533 \\ 1.0000 \end{bmatrix}$	$\begin{bmatrix} -0.7367 \\ -0.4320 \end{bmatrix}$	$\begin{bmatrix} -0.9000 \\ -0.5000 \end{bmatrix}$	1.3341°	0.1769

Table 4.6 Iterations of the fixed-point method for $\mathbf{v}_3 \in \Phi_v \setminus \Phi$.

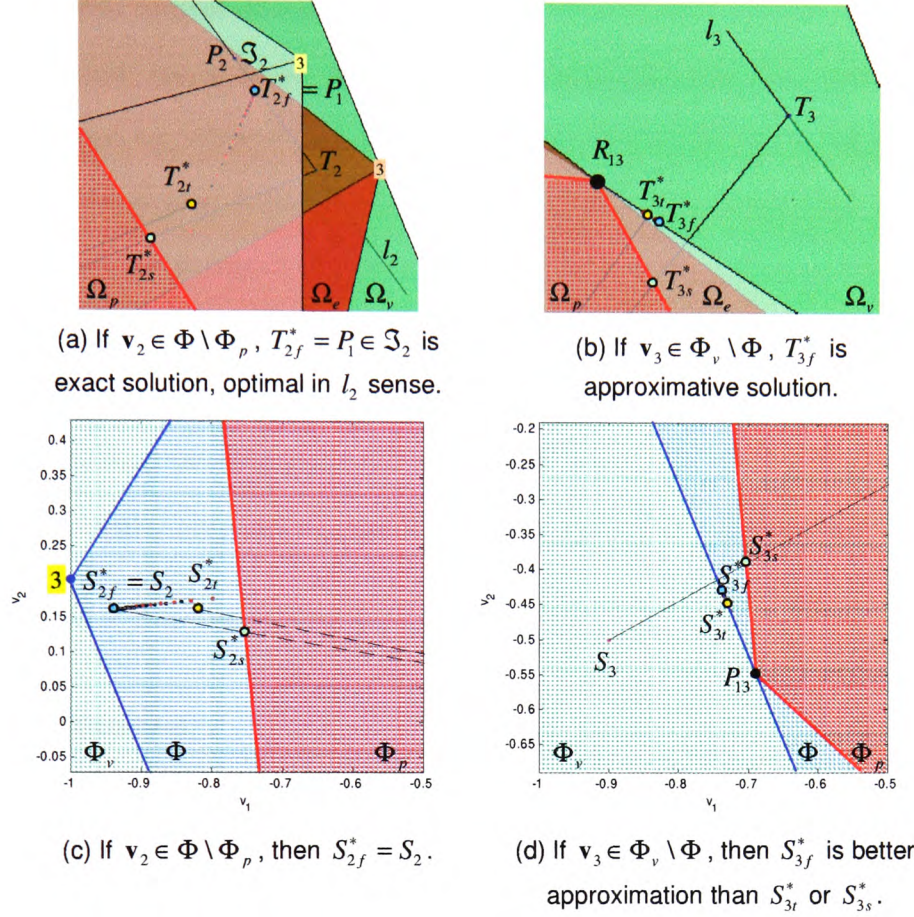


Figure 4.22 Improving of T - and S -approximation by the fixed-point method.

Individual iterations are shown in Figure 4.22. In particular, iterations that start from the T -approximation are shown as black dots, and as red dots, if they start from the S -approximation. If the desired virtual control input \mathbf{v}_2 lies in $\Phi \setminus \Phi_p$, the fixed-point algorithm converges toward the exact solution $T_{2f}^* = P_1$, which lies in the solution set \mathfrak{S}_2 and has lower l_2 norm than any other point in \mathfrak{S}_2 . However, if the desired virtual control input \mathbf{v}_3 lies outside Φ , the fixed point algorithm finds an approximate solution T_{3f}^* that is better than solutions obtained by T - or S -approximation, since it has lower direction and magnitude error.

4.4.2 Direct control allocation

In the original paper (Durham, 1993), the author uses nomenclature $\mathbf{v} = \mathbf{m}$ (the virtual control inputs are moments) and the term attainable moment set for Φ , because he discusses the problem from the aerospace perspective. In the following, the general nomenclature will be kept. It is assumed that the controls are independent (non-coplanar controls), i.e. that every $k \times k$ partition of the control effectiveness matrix $\mathbf{B}_{k \times m}$ is non-singular. This assumption yields that the control allocation problem has a unique solution for the case when virtual control input lies on the boundary of Φ . Vectors that lie on the boundary $\partial(\Omega)$ ($\partial(\Phi)$) are denoted as \mathbf{u}^* (\mathbf{v}^*), respectively. A unit vector in the direction of \mathbf{v} is denoted by $\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|_2}$.

Problem statement

The control allocation problem is defined as follows: given \mathbf{B} , Ω and some desired virtual control input \mathbf{v}_d , determine the true control input $\mathbf{u} \in \Omega$ that will generate a virtual control input for the largest possible magnitude of \mathbf{v} in the direction $\hat{\mathbf{v}}_d$. That is, the objective is to find a control vector $\mathbf{u} \in \Omega$ that results in the best approximation of the vector \mathbf{v}_d in the given direction, i.e. such that direction error is zero. The implicit assumption is that directionality is an important characteristic of flight control. Hence, the direct allocation method moves the focus from the interior or exterior of Φ to its boundary and searches for a solution there, employing the fact that the solution is unique on the boundary, under assumption of non-coplanar controls.

Description

Given a virtual control demand \mathbf{v}_d , the direct control allocation method involves the following steps:

1. determine Φ ,
2. find the boundary $\partial(\Phi)$,
3. find the intersection \mathbf{v}_d^* of the half-line p (in the direction of $\hat{\mathbf{v}}_d$) and $\partial(\Phi)$,
4. calculate the scaling factor

$$a = \frac{\|\mathbf{v}_d^*\|_2}{\|\mathbf{v}_d\|_2} \quad (4.47)$$

5. determine the unique control vector $\mathbf{u}^* \in \partial(\Omega)$ such that $\mathbf{B}\mathbf{u}^* = \mathbf{v}_d^*$,
6. select the true control input $\mathbf{u} \in \Omega$ according to

$$\mathbf{u} = \begin{cases} \frac{1}{a}\mathbf{u}^*, & \text{if } a > 1 \\ \mathbf{u}^*, & \text{if } a \leq 1 \end{cases} \quad (4.48)$$

Bodson (2002) condensed this verbal description into the following optimisation problem:

Given \mathbf{B} and \mathbf{v}_d , find a real number a and a vector \mathbf{u}^* that solve

$$\max_{a, \mathbf{u}} a \quad (4.49)$$

subject to

$$\begin{aligned} \mathbf{B}\mathbf{u} &= a\mathbf{v}_d \\ \underline{\mathbf{u}} &\leq \mathbf{u} \leq \bar{\mathbf{u}} \end{aligned} \quad (4.50)$$

Then assign \mathbf{u} as in (4.48).

Example 4.11 (Direct control allocation)

Consider the control allocation problem in Example 4.1 and let $\mathbf{v}_d = [-0.5 \ 0.6]^T$, as in Example 4.2. The attainable command set Φ and its boundary $\partial(\Phi)$ are already found in section 4.3.3 and shown again in Figure 4.23.

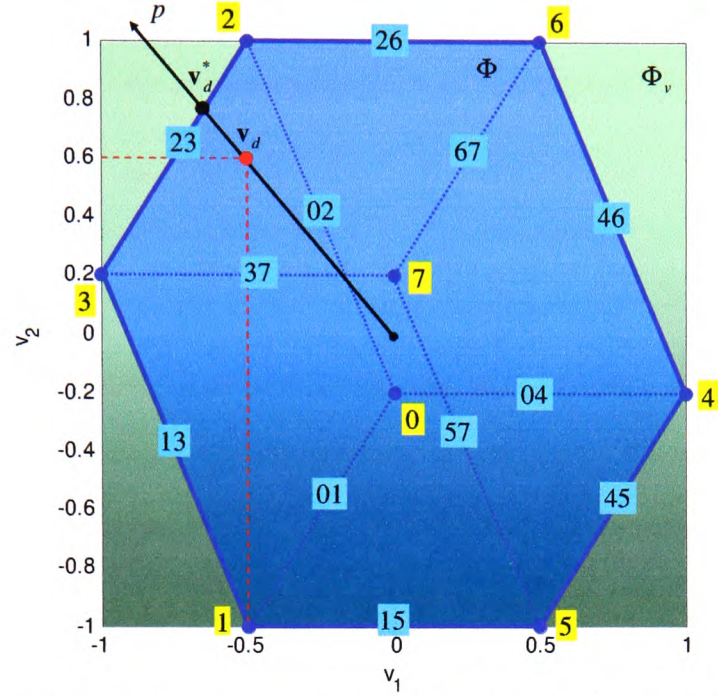


Figure 4.23 The direct control allocation method searches for $\mathbf{v}_d^* = a\mathbf{v}_d$ on $\partial(\Phi)$.

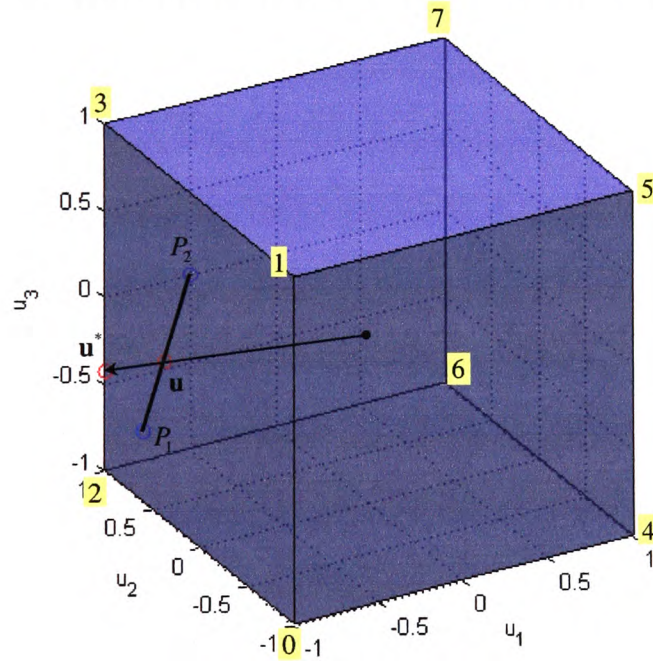


Figure 4.24 The direct control allocation method finds the limit solution \mathbf{u}^* on $\partial(\Omega)$, such that

$$\mathbf{B}\mathbf{u}^* = \mathbf{v}_d^*, \text{ and performs inverse scaling to obtain the solution } \mathbf{u} = \frac{1}{a}\mathbf{u}^*.$$

The intersection of the half-line p in the direction of \mathbf{v}_d and $\partial(\Phi)$ (edge 23, see Figure 4.24) is denoted as \mathbf{v}_d^* and can be found as the intersection of the two lines:

$$p: -\frac{3}{5}v_1 - \frac{1}{2}v_2 = 0 \quad \text{and} \quad 23: \frac{4}{5}v_1 - \frac{1}{2}v_2 = -\frac{9}{10}. \quad \text{Solving the system yields}$$

$$\mathbf{v}_d^* = \begin{bmatrix} -\frac{9}{14} & \frac{27}{35} \end{bmatrix}^T. \quad \text{The scaling factor is } a = \frac{\|\mathbf{v}_d^*\|_2}{\|\mathbf{v}_d\|_2} = \frac{1.0042}{0.7810} = 1.2857. \quad \text{Now the focus is}$$

moved to Ω : a vector \mathbf{u}^* must be found such that $\mathbf{B}\mathbf{u}^* = \mathbf{v}_d^*$. Since \mathbf{v}_d^* lies on edge 23 $\in \partial(\Phi)$, vector \mathbf{u}^* must lie on edge 23 $\in \partial(\Omega)$. Every vector on this edge has the first coordinate equal to -1 , the second equal to 1 and the third free to vary between -1 and 1 .

Now, the condition $\mathbf{B}\mathbf{u}^* = \mathbf{v}_d^*$ yields $\mathbf{u}^* = \begin{bmatrix} -1 & 1 & -\frac{6}{14} \end{bmatrix}^T$. Finally, since $a > 1$, the

solution \mathbf{u} that satisfies $\mathbf{B}\mathbf{u} = \mathbf{v}_d$ is found by performing inverse scaling

$$\mathbf{u} = \frac{1}{a}\mathbf{u}^* = \begin{bmatrix} -0.7778 & 0.7778 & -0.3333 \end{bmatrix}^T. \quad \text{It is easy to verify that } \mathbf{u} \in P_1P_2.$$

Remarks

- The solution obtained by direct control allocation has l_2 norm $\|\mathbf{u}\|_2 = 1.1493$, while the pseudoinverse method found the solution with l_2 norm $\|\mathbf{u}\|_2 = 1.1049$. Comparing these two norms, it is possible to conclude that the solution obtained by the direct allocation method is not optimal in l_2 sense.
- Direct control allocation is trying to allocate the actuators by direction preservation. If \mathbf{v} is not feasible, i.e. if \mathbf{v} lies outside Φ , the method will find \mathbf{u}^* such that $\mathbf{B}\mathbf{u}^* = \mathbf{v}^*$, where $\mathbf{v}^* \in \partial(\Phi)$ is the best approximation of \mathbf{v} in the direction of \mathbf{v} (\mathbf{v}^* and \mathbf{v} have the same angle, but different magnitude, i.e. direction error is always zero).

- There are no design variables to be selected. That is, the solution is completely determined by the control effectiveness matrix \mathbf{B} , and the control constraints $\underline{\mathbf{u}}$ and $\bar{\mathbf{u}}$.
- For $a > \mathbf{I}$, no one element in \mathbf{u} is saturated (Durham, 1994). That is, if \mathbf{v}_d lies strictly inside Φ , then \mathbf{u} lies strictly inside Ω .
- The control constraints $\underline{\mathbf{u}} \leq \mathbf{u} \leq \bar{\mathbf{u}}$ must include the origin (Bodson, 2002). That is, the method requires $\underline{\mathbf{u}} \leq \mathbf{0}$, $\bar{\mathbf{u}} \geq \mathbf{0}$, i.e. the origin $\mathbf{u} = \mathbf{0}$ must be a feasible true control input. This stems from the way how \mathbf{u} is constructed in (4.48). When rate constraints are included, as in (4.8), the origin $\mathbf{u} = \mathbf{0}$ is often outside the control constraints. A solution to this problem consists in applying the technique to increments of the vector \mathbf{u} . However, this solution introduces a wind-up problem, which must be resolved using "restoring" techniques.
- Another drawback is that the direct control allocation method does not enable axis prioritisation.

4.4.3 Daisy chain control allocation

In daisy chain control allocation (Härkegård, 2003; Bordignon, 1996), the actuators are divided into groups which are successively employed to generate the total control effort.

Description

The m true control inputs are first divided into M groups,

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}^1 \\ \vdots \\ \mathbf{u}^M \end{bmatrix}$$

after possibly reordering the control inputs. Then, the control effectiveness matrix is partitioned accordingly,

$$\mathbf{B} = [\mathbf{B}_1 \quad \dots \quad \mathbf{B}_M]$$

The control allocation problem $\mathbf{B}\mathbf{u} = \mathbf{v}$ can be rewritten as

$$\mathbf{B}_1 \mathbf{u}^1 + \dots + \mathbf{B}_M \mathbf{u}^M = \mathbf{v} \quad (4.51)$$

The main idea of daisy chain is to first find the best possible contribution of group \mathbf{u}^1 by solving

$$\mathbf{B}_1 \mathbf{u}^1 = \mathbf{v} \quad (4.52)$$

for \mathbf{u}^1 .

Two cases are possible:

1. if $\text{rank } \mathbf{B}_1 \geq \dim \mathbf{v} = k$, (4.52) is solved by

$$\mathbf{u}^1 = \mathbf{P}_1 \mathbf{v} \quad (4.53)$$

where \mathbf{P}_1 is any right inverse of \mathbf{B}_1 (see (2.3)).

2. if $\text{rank } \mathbf{B}_1 < k$, \mathbf{v} can not, in general, be generated using only \mathbf{u}^1 . In this case, an approximate solution of (4.52) is assumed to have the same form (4.53).

Definition 4.2 (Saturation function)

Saturation function is defined as $\text{sat}_{\mathbf{u}}(\mathbf{x}) = \mathbf{y}$, such that $y_k = \begin{cases} \bar{u}_k, & \text{if } x_k > \bar{u}_k \\ x_k, & \text{if } \underline{u}_k \leq x_k \leq \bar{u}_k \\ \underline{u}_k, & \text{if } x_k < \underline{u}_k \end{cases}$.

If \mathbf{u}^1 satisfies (4.52), as well as the actuator constraints, the allocation was successful and the procedure halts. Otherwise, \mathbf{u}^1 is saturated according to its constraints

$$\mathbf{u}^1 = \text{sat}_{\mathbf{u}^1}(\mathbf{P}_1 \mathbf{v}) \quad (4.54)$$

and the second group of actuators \mathbf{u}^2 is employed to solve

$$\mathbf{B}_2 \mathbf{u}^2 = \mathbf{v} - \mathbf{B}_1 \mathbf{u}^1 \quad (4.55)$$

for \mathbf{u}^2 . The principle is the same as for \mathbf{u}^1 : the solution (possibly approximate) is given by $\mathbf{u}^2 = \mathbf{P}_2(\mathbf{v} - \mathbf{B}_1\mathbf{u}^1)$. Again, if \mathbf{u}^2 satisfies (4.55) and the actuator constraints, the procedure halts. Otherwise, \mathbf{u}^2 is saturated and the procedure is repeated until either \mathbf{v} is met or all actuator groups have been employed.

The daisy chain control allocation procedure can be summarised as

$$\begin{aligned}
 \mathbf{u}^1 &= \text{sat}_{\mathbf{u}^1}(\mathbf{P}_1\mathbf{v}) \\
 \mathbf{u}^2 &= \text{sat}_{\mathbf{u}^2}(\mathbf{P}_2(\mathbf{v} - \mathbf{B}_1\mathbf{u}^1)) \\
 &\vdots \\
 \mathbf{u}^M &= \text{sat}_{\mathbf{u}^M}\left(\mathbf{P}_M\left(\mathbf{v} - \sum_{i=1}^{M-1} \mathbf{B}_i\mathbf{u}^i\right)\right)
 \end{aligned} \tag{4.56}$$

where $\mathbf{B}_i\mathbf{P}_i = \mathbf{I}$ (if possible). Figure 4.25 illustrates the procedure for $M = 3$.

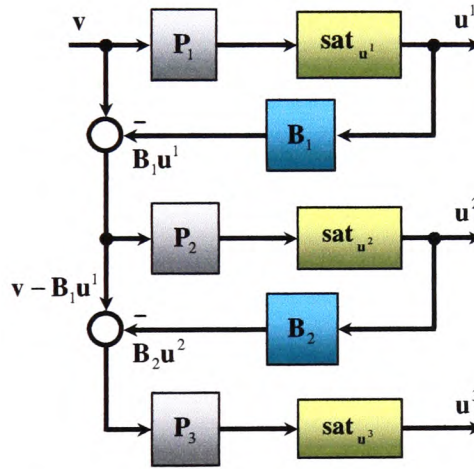


Figure 4.25 Daisy chain control allocation for $M = 3$.

Example 4.12 (Daisy chain control allocation)

Consider the same control allocation problem as in previous examples. The method will be demonstrated for the case when a set of actuators is divided in $M = 3$ groups:

$$\begin{aligned}
u^1 &= u_1 \\
u^2 &= u_2 \\
u^3 &= u_3
\end{aligned} \tag{4.57}$$

This grouping yields to the following partition of \mathbf{B} :

$$\mathbf{B} = [\mathbf{B}_1 \quad \mathbf{B}_2 \quad \mathbf{B}_3] = \left[\underbrace{\begin{bmatrix} 0.5 \\ 0 \end{bmatrix}}_{\mathbf{B}_1} \underbrace{\begin{bmatrix} -0.25 \\ 0.6 \end{bmatrix}}_{\mathbf{B}_2} \underbrace{\begin{bmatrix} -0.25 \\ -0.4 \end{bmatrix}}_{\mathbf{B}_3} \right] \tag{4.58}$$

The approximate solution of the first equation $\mathbf{B}_1 u^1 = \mathbf{v}_d$ is given as

$$u^1 = \text{sat}_{u^1}(\mathbf{P}_1 \mathbf{v}_d) = \text{sat}_{u^1} \left(\underbrace{\begin{bmatrix} 2 & 0 \end{bmatrix}}_{\mathbf{P}_1} \underbrace{\begin{bmatrix} -0.5 \\ 0.6 \end{bmatrix}}_{\mathbf{v}_d} \right) = \text{sat}_{u^1}(-1) = -1 \tag{4.59}$$

where \mathbf{P}_1 is found from⁵

$$\underbrace{\begin{bmatrix} 0.5 \\ 0 \end{bmatrix}}_{\mathbf{B}_1} \underbrace{\begin{bmatrix} p_{11} & p_{12} \end{bmatrix}}_{\mathbf{P}_1} \approx \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.60}$$

Since $\mathbf{B}_1 u^1 \neq \mathbf{v}_d$, the procedure is continued for u^2 . The approximate solution of the second equation $\mathbf{B}_2 u^2 = \mathbf{v}_d - \mathbf{B}_1 u^1$ is given by

$$u^2 = \text{sat}_{u^2}(\mathbf{P}_2(\mathbf{v}_d - \mathbf{B}_1 u^1)) = \text{sat}_{u^2} \left(\underbrace{\begin{bmatrix} -0.5917 & 1.4201 \end{bmatrix}}_{\mathbf{P}_2} \underbrace{\begin{bmatrix} 0 \\ 0.6 \end{bmatrix}}_{\mathbf{v}_d - \mathbf{B}_1 u^1} \right) = \text{sat}_{u^2}(0.8521) = 0.8521 \tag{4.61}$$

where \mathbf{P}_2 is found from

$$\underbrace{\begin{bmatrix} -0.25 \\ 0.6 \end{bmatrix}}_{\mathbf{B}_2} \underbrace{\begin{bmatrix} p_{21} & p_{22} \end{bmatrix}}_{\mathbf{P}_2} \approx \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.62}$$

⁵ MATLAB operator "\" is used to find \mathbf{P}_1 (MATLAB code: $\mathbf{P}_1 = \mathbf{B}_1 \backslash \text{eye}(2)$).

Since $\mathbf{B}_2 \mathbf{u}^2 \neq \mathbf{v}_d - \mathbf{B}_1 \mathbf{u}^1$, the procedure is continued for \mathbf{u}^3 . The approximate solution of the third equation $\mathbf{B}_3 \mathbf{u}^3 = \mathbf{v}_d - (\mathbf{B}_1 \mathbf{u}^1 + \mathbf{B}_2 \mathbf{u}^2)$ is given by

$$\begin{aligned} \mathbf{u}^3 &= \text{sat}_{\mathbf{u}^3} \left(\mathbf{P}_3 \left(\mathbf{v}_d - (\mathbf{B}_1 \mathbf{u}^1 + \mathbf{B}_2 \mathbf{u}^2) \right) \right) = \text{sat}_{\mathbf{u}^3} \left(\underbrace{\begin{bmatrix} -1.1236 & -1.7978 \end{bmatrix}}_{\mathbf{P}_3} \underbrace{\begin{bmatrix} 0.2130 \\ 0.0888 \end{bmatrix}}_{\mathbf{v}_d - (\mathbf{B}_1 \mathbf{u}^1 + \mathbf{B}_2 \mathbf{u}^2)} \right) \\ &= \text{sat}_{\mathbf{u}^3} (-0.3989) = -0.3989 \end{aligned} \quad (4.63)$$

Hence, the daisy chain method found the solution $\mathbf{u} = [\mathbf{1} \quad 0.8521 \quad -0.3989]^T$ (see Figure

4.26), which is approximate, since $\mathbf{B}\mathbf{u} = \begin{bmatrix} -0.6133 \\ 0.6708 \end{bmatrix} \approx \begin{bmatrix} -0.5 \\ 0.6 \end{bmatrix} = \mathbf{v}_d$.

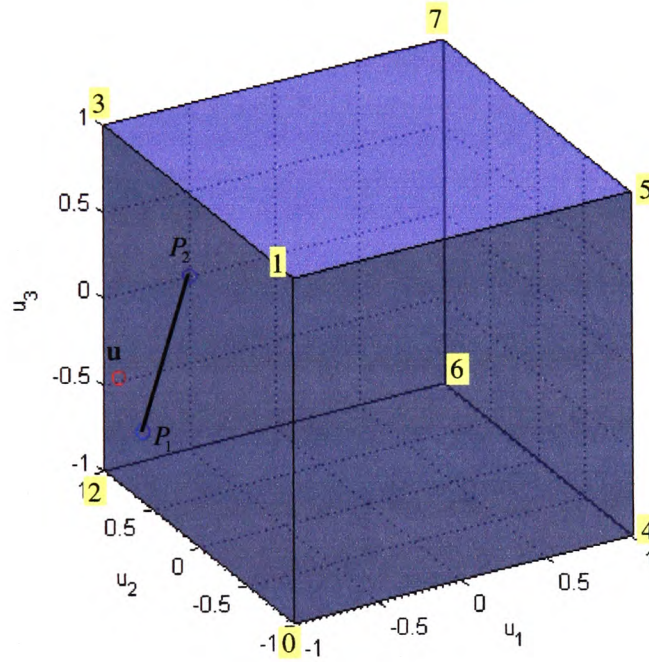


Figure 4.26 Daisy chain solution for $M = 3$ groups of actuators.

Remarks

- The design choices consist of the actuator groupings and the \mathbf{P}_i matrices. Note that in the special case, when each group consist of k actuators and each of square matrices \mathbf{B}_i has a full rank, each matrix \mathbf{P}_i is uniquely determined from $\mathbf{B}_i \mathbf{P}_i = \mathbf{I}$.

- The method is very dependent on design choices and may fail to produce feasible virtual control inputs, like in Example 4.12. Similar cases are given in (Bordignon, 1996).
- In aircraft applications, daisy chain control allocation is often used when thrust vectoring is available (Enns, *et al.*, 1994). Conventional control surfaces, such as elevator, aileron and rudder, are then primarily used for control, and the thrust vectoring vanes are used for auxiliary control.

4.5 Concluding remarks

In this chapter the control allocation problem, in the general case, has been formulated. This formulation was used to establish which class of system control architecture can be separated into two independent tasks (control law and control allocation), thereby allowing the control allocation to be considered separately from the control law, which is the norm for ROVs and AUVs.

A number of methods for the solution of the general control allocation problem have been presented. In order to compare their performance the solutions were tested using the same example and it was shown that the solution is dependant upon the method and/or criteria used. A new hybrid approach, which integrates pseudoinverse and fixed point methods, was introduced and shown to be able to allocate the exact solution, optimal in an l_2 sense, inside the entire attainable command set Φ . This solution minimises the control energy cost function, which is the most suitable criteria for underwater applications.

Clear geometric interpretations were given for each method, which enabled visualisation and better understanding of the complex underlying mathematical relationships. For easy geometric interpretation, the control allocation problem in selected example had the two-dimensional virtual control space and the three-dimensional true control space. However, the concepts can easily be extended to ROV applications which have three-dimensional

virtual space (three DOF in horizontal plane) and four-dimensional true control space (4 horizontal thrusters). This is explored further in the development of the FDAS, which is described in the next chapter.

4.6 References

- BECK, R.E. (2002). *Application of Control Allocation Methods to Linear Systems with Four or More Objectives*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.
- BODSON, M. (2002) Evaluation of Optimisation Methods for Control Allocation. In *Journal of Guidance, Control and Dynamics*, **25**(4), pp. 703-711.
- BORDIGNON, K.A. (1996). *Constrained Control Allocation for Systems with Redundant Control Effectors*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.
- BURKEN, J., LU, P. and WU, Z. (1999) Reconfigurable Flight Control Designs With Applications to the X-33 Vehicle. NASA/TM-1999-206582.
- BURKEN, J., LU, P., WU, Z. and BAHM, C. (2001) Two Reconfigurable Flight-Control Design Methods: Robust Servomechanism and Control Allocation. In *Journal of Guidance, Control and Dynamics*, **24**(3), pp. 482-493.
- DAVIDSON, J., LALLMAN, F.J. and BUNDICK, W.T. (2001) Integrated reconfigurable control allocation. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Montreal.
- DOMAN, D.B. and OPPENHEIMER, M.W. (2002) Improving control allocation accuracy for non-linear aircraft dynamics. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Monterey, CA.
- DURHAM, W.C. (1993) Constrained Control Allocation. *Journal of Guidance, Control, and Dynamics*, Vol. **16**, No. 4, pp. 717-725.
- DURHAM, W.C. (1994) Constrained Control Allocation: Three Moment Problem. *Journal of Guidance, Control, and Dynamics*, Vol. **17**, No. 2, pp. 330-336.
- DURHAM, W.C. and BORDIGNON, K.A. (1996) Multiple control effector rate limiting. *Journal*

of Guidance, Control, and Dynamics, **19**(1), pp. 30-37.

EBERHARDT, R.L. and WARD, D.G. (1999) Indirect adaptive flight control of a tailless fighter aircraft. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Portland, pp. 466-476.

ENNS, D., BUGAJSKI, D., HENDRICK, R. and STEIN, G. (1994) Dynamic inversion: an evolving methodology for flight control design. *International Journal of Control*, **59**(1), pp. 71-91.

ENNS, D. (1998) Control allocation approaches. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Boston, pp. 98-108.

HÄRKEGÅRD, O. (2003). *Backstepping and Control Allocation with Application to Flight Control*. PhD thesis, Department of Electrical Engineering, Linköping University, Sweden.

IKEDA, Y. and HOOD, M. (2000). An application of l_1 optimisation to control allocation. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Denver.

LINDFORS, I. (1993). Thrust allocation method for the dynamic positioning system. In *10th International Ship Control Systems Symposium (SCSS'93)*, Ottawa, pp. 3.93-3.106.

SNELL, S.A., ENNS, D.F. and GARRARD, W.L. (1992). Nonlinear inversion flight control for a supermaneuverable aircraft. *Journal of Guidance, Control and Dynamics*, **15**(4), pp. 976-984.

SØRDALEN, O.J. (1997). Optimal thrust allocation for marine vessels. *Control Engineering Practice*, **5**(9), pp. 1223-1231.

VIRNIG, J.C. and BODDEN, D.S. (1994). Multivariable control allocation and control law conditioning when control effectors limit. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Scottsdale.

WISE, K.A., BRINKER, J.S., CALISE, A.J., ENNS, D.F., ELGERSMA, M.R. and VOULGARIS, P. (1999). Direct adaptive reconfigurable flight control for a tailless advanced fighter aircraft. *International Journal of Robust and Nonlinear Control*, **9**(14), pp. 999-1012.

Chapter 5: Fault Diagnosis and Accommodation System

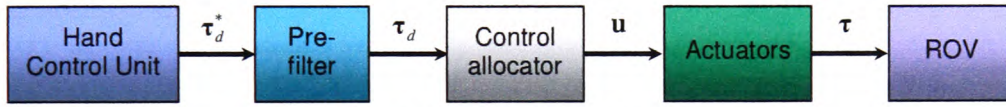
5.1 Introduction

A novel thruster fault diagnosis and accommodation system (FDAS) for underwater vehicles is proposed in this chapter. Basically, the FDAS is a control allocator, but this primary function is enhanced with the ability of automatic thruster fault detection and accommodation. Material presented in this chapter is closely related with the basic concepts about propulsion system (section 3.7) and control allocation (section 4.3). The hybrid approach for control allocation (Example 4.10) is extended for the case of the three-dimensional virtual control space and the four-dimensional true control space in this chapter and used as a foundation to build an enhanced control allocator, with fault detection and accommodation capabilities.

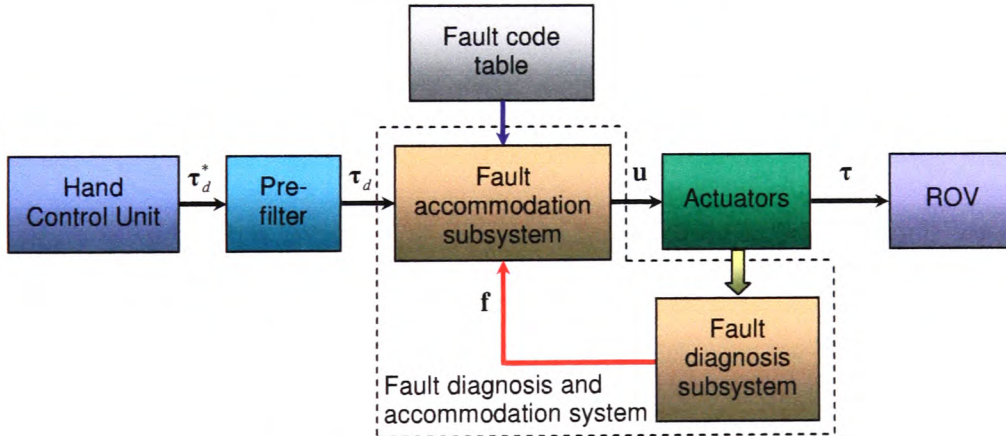
This chapter is organised as follows: the rest of this section provides conceptual elements and reveals the main idea of the FDAS, displays a list of requirements and addresses some implementation issues. Section 5.2 discusses the control allocation problem for underwater vehicles. Topics include introducing basic concepts, nomenclature and terminology, description of normalisation procedure and choice of optimisation criteria and weighting matrices. The architecture of the FDAS is described in section 5.3. A detailed description of the Fault Diagnosis System (FDS) is given in section 5.4. Topics include fault classification and description of the fault code table, Fault Detection Unit (FDU) and the FDU algorithm. Section 5.5 presents the structure of the Fault Accommodation System (FAS), describes the hybrid approach for control allocation and discusses the feasibility of pseudoinverse solution. In addition, the feasible region concept is described for different faulty situations. The FAS algorithm is presented at the end of section. Remarks on implementation issues are given in section 5.6. Section 5.7 summarises concluding remarks.

5.1.1 Conceptual elements

A standard open-loop ROV control structure is shown in Figure 5.1 (a). The ROV pilot uses the Hand Control Unit (HCU) to generate vector τ_d^* , which can be interpreted as a desired vector of propulsion forces and moments among axes in the body-fixed frame. Raw signals from the HCU, packed in vector τ_d^* , pass through the low-pass pre-filter to smooth out the commanded input and to protect the actuators from damage caused by abrupt changes of set points. The output of the pre-filter is the desired vector of propulsion forces and moments (virtual control input) τ_d . The control allocator maps the vector τ_d into the vector (true control input) u , representing control settings for individual actuators.



(a) Standard architecture, without FDAS.



(b) Improved architecture, with FDAS.

Figure 5.1 Open-loop ROV control structure.

After actuation with u , the actuators generate a vector of propulsion forces and moments (total control effect) τ , which is applied as input ${}^B\tau$ to the ROV dynamics block (for

example, simulation diagram for ROV dynamics and kinematics shown in Figure 3.4), and determine the behaviour of the vehicle. The main objective of the control allocation is to ensure that the condition $\tau = \tau_d$ is satisfied for all attainable τ_d .

The thruster configuration of the FALCON (Figure 3.7 (a)) enables direct control of 4 DOF: surge, sway and yaw in the horizontal plane and heave in the vertical plane. In a similar manner, the modified thruster configuration of the URIS (Figure 3.7 (b)) allows direct control of only 3 DOF (surge, sway and yaw in the horizontal plane). In both cases, control system for motion in the horizontal plane is overactuated, since there are three controllable DOF and four horizontal thrusters. Vector τ_d can be decomposed into two

parts as $\tau_d = \begin{bmatrix} \tau_d^{HT} \\ \tau_d^{VT} \end{bmatrix}$, where τ_d^{HT} represents desired surge force τ_x , sway force τ_y and

yaw moment τ_N for motion in the horizontal plane, and τ_d^{VT} is equal to heave force τ_z for motion in the vertical plane. The control allocation problem for motion of the FALCON in the vertical plane is straightforward, since the vector τ_d^{VT} has only one component, i.e. there is one-to-one correspondence between the controllable DOF (heave) and the vertical thruster. However, in the general case the vector τ_d^{VT} can have three components (heave force τ_z , roll moment τ_K and pitch moment τ_M). Typical example is ODIN (see page 2-41), with four horizontal and four vertical thrusters, where each of vectors τ_d^{HT} and τ_d^{VT} have three components. Since the work described herein is related with improvements of two particular vehicles (FALCON and URIS), the following discussion will concentrate on horizontal thrusters, although the same principles are valid for vertical thrusters. Following on from the discussion in Example 4.7, the 3D virtual

control space $\underline{\Phi}_v^{HT}$ for *normalised*¹ $\underline{\tau}_d^{HT}$ is represented as unit cube in Figure 5.2, and the two characteristic regions inside $\underline{\Phi}_v^{HT}$ are $\underline{\Phi}_p^{HT}$ (feasible region for pseudoinverse) and $\underline{\Phi}^{HT} \supset \underline{\Phi}_p^{HT}$ (attainable command set). It should be emphasized that there is an infinite number of exact solutions for $\underline{\tau}_d^{HT} \in \underline{\Phi}^{HT}$, while no exact solution exists for $\underline{\tau}_d^{HT} \in \underline{\Phi}_v^{HT} \setminus \underline{\Phi}^{HT}$. The standard control structure for the FALCON, developed by Seaeye Marine Ltd., uses pseudoinverse approach to solve the control allocation problem, which is able to find the exact solution, optimal in the l_2 sense, only if $\underline{\tau}_d^{HT} \in \underline{\Phi}_p^{HT}$. In order to extend the size of the region with the exact solution, the control allocator in the standard structure shown in Figure 5.1 (a) is replaced by the FAS in the improved control structure, shown in Figure 5.1 (b).

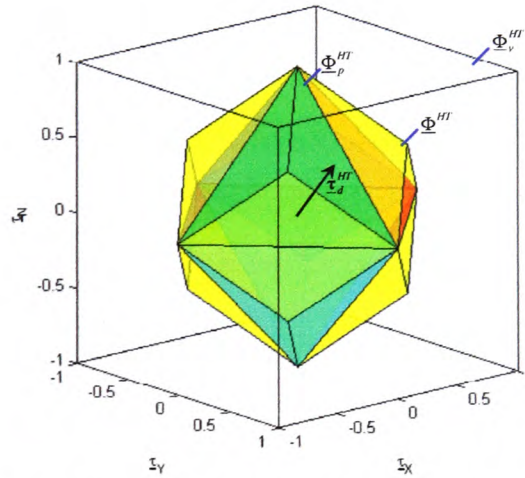


Figure 5.2 Partitions of the normalised virtual control space for motion in the horizontal plane².

¹ The normalisation process is described in section 5.2.2. Normalised variables are denoted by an underline.

² Partitions, shown in Figure 5.2, illustrate the main idea of the FDAS concept and assume X-shaped configuration of horizontal thrusters (FALCON). Full description of these partitions for X-shaped and cross-shaped configurations is given in section 5.5.2.

The FAS performs a hybrid approach for control allocation, such that it finds the exact solution of the control allocation problem, optimal in l_2 sense, for $\underline{\tau}_d^{HT} \in \underline{\Phi}^{HT}$. Hybrid approach (Example 4.10) uses pseudoinverse for $\underline{\tau}_d^{HT} \in \underline{\Phi}_p^{HT}$ and fixed-points iterations³ for $\underline{\tau}_d^{HT} \in \underline{\Phi}^{HT} \setminus \underline{\Phi}_p^{HT}$. In addition, the primary task of control allocation is enhanced with the fault diagnosis module performed by FDS, able to monitor state of the thrusters and inform the FAS about any malfunctions using the total fault indicator vector \mathbf{f} , carrying the codes of faulty states for each thruster. The FAS uses information provided by the FDS to accommodate faults and perform appropriate reconfiguration reallocating control energy among operable thrusters. The overall fault diagnosis and accommodation process is very fast, despite the fact that in some cases it is necessary to perform iterations, due to the computational efficiency of the FDAS algorithm, where the heaviest numerical calculations are performed off-line, in advance.

The thruster fault diagnosis and accommodation process is summarised as follows:

1. The FDS detects fault (type and degree of damage in each horizontal thruster) and generates the total fault indicator vector,
2. Using the fault code table (see section 5.4.2), the FAS penalises the faulty thruster by increasing the corresponding weight in weighting matrix, updating the criteria and restricting the saturation bounds,
3. The FAS finds a new, feasible control vector for the given input vector using the hybrid approach, which minimises a new criteria,
4. New control vector is denormalised and used to actuate the thrusters.

³ The fixed-point iterations can also be used to find the approximate solution for cases $\underline{\tau}_d^{HT} \in \underline{\Phi}_v^{HT} \setminus \underline{\Phi}^{HT}$,

when the exact solution does not exist.

5.1.2 Requirements

The problem of thruster fault detection and accommodation for underwater vehicle has special features, due to specific environmental conditions in which the vehicle operates.

At the beginning of the IMPROVES project, Seaeye Marine Ltd. specified a list of requirements that should be taken into account before choosing a fault detection method.

The most important requirements that the FDAS should fulfil are:

- Reliable and fast fault detection, without false alarms,
- Easy integration with the existing control system,
- On-line learning and adaptation to new types of faults,
- Cost efficiency i.e. the FDAS should use resources already available, without introducing new hardware,
- Easy transfer to and implementation in other vehicles.

These requirements limit the choice of available approaches and restrict the designer freedom.

5.1.3 Implementation issues

There are certain implementation issues that must be taken into account during the design process. These issues, described below, are addressed in section 5.6 in relation to implementation of the FDAS in real applications.

Timing issues: The FALCON uses a distributed intelligence control system (see Appendix A), where each device (thruster, light, compass etc.) is controlled by a microcontroller and represents a "slave node" with the unique ID. Each slave node is connected to the network, supervised by "master node" (processor), located in the Surface Unit. The master node uses a loop technique to control the behaviour of the slave nodes. At each control cycle, the master node reads the states of the HCU controls and on-board



sensors, performs mathematical calculations to find the new set points and sends updated values to the slave nodes. Typical execution time of one cycle is about 50 *ms*. It is required that the FDAS algorithm does not increase this time significantly. Hence, the FDAS algorithm must be computationally very efficient.

Memory issues: The master node has a limited storage capacity, which must be used in an efficient way. It is desirable that the FDAS performs as many calculations as possible off-line and stores the results (i.e. knowledge about faulty situations) in a compact form in memory (hard disk). In this way, the FDAS is able to take advantage of saved results to make easier on-line processing and to perform the fault diagnosis and accommodation tasks very fast. However, there is a trade-off between the amount of data that can be saved and available memory.

Accuracy issues: The master node controls the velocities of each thruster such that at each control cycle it sends updated set points (desired velocities) to slave nodes that represent Thruster Control Units (TCU). Each TCU uses the digital PID algorithm to control the propeller angular velocity. The FALCON control protocol uses the normalised form for the desired thruster velocities, where each demand is represented as an integer number between -100 and $+100$. Positive values correspond to positive spin direction of the propeller. In this way, at the last stage of the control allocation, the existing control software must round the desired velocities to the nearest integer before transmitting them to the TCUs, introducing rounding error. In addition, signals from the HCU are converted into digital form using A/D converter, introducing quantisation error. Hence, the architecture of the FALCON control protocol and A/D conversion process introduce errors that must be taken into account for design of the FDAS.

5.2 *Control allocation for underwater vehicles*

5.2.1 Background

For underwater vehicles the most common actuators are (Fossen, 2002):

- **Azimuth thrusters:** thruster units that can be rotated an angle α about the z -axis during the mission and produce two force components (F_x, F_y) in the horizontal plane. They are attractive in dynamic positioning systems, since they can produce forces in different directions, leading to an overactuated control problem that can be optimised with respect to power and possible faulty situations.
- **Fixed direction (non-rotatable) thrusters:** In contrast to azimuth thrusters, where an angle α can vary with time, fixed direction thrusters are characterised with a fixed angle $\alpha = \alpha_0$, i.e. orientation of these thrusters is fixed in advance and cannot be changed during the mission.
- **Control surfaces:** control surfaces can be mounted at different locations to produce lift and drag forces, like fins for diving, rolling and pitching, rudders for steering, etc.

The FALCON and URIS have no other actuators except fixed direction thrusters, and the following discussion will concentrate on this type of actuators, while more information about other types can be found in (Fossen, 2002). Initially, the control allocation problem will be formulated and solved under the following assumptions:

1. the dynamics of thruster control loop are neglected (Figure 3.16),
2. the relationship between propeller thrust/torque and the control variable is given by a modified version of affine thruster model (3.99), as described in the following.

The first assumption is realistic, since the time constants of DC motors used to drive thrusters of the FALCON and the URIS are very small and dynamics of the thruster control loop is much faster than dynamics of the rest of the system (see discussion about difficulties for using control allocation on page 4-16).

The second assumption means that:

- shaft torque Q_s is neglected, since it is small compared to Q_r , $\mathbf{Q}_r = \mathbf{r} \times \mathbf{T}$,
- the effect of the ambient water velocity \mathbf{u}_a on propeller thrust T is neglected,
- the symmetrical relationship between thrust T and the auxiliary control variable \mathbf{u}' is used (see Figure 3.14).

Under these assumptions, the control allocation problem for open-frame underwater vehicles (in particular, FALCON and URIS) is linear and can be solved using techniques described in Chapter 4. The influence of neglected factors on the performance of the ROV control system can be investigated using the ROV simulator incorporating the bilinear thruster model (3.98) and the dynamics of the thruster control loop. Test cases (A7), (A8) and (A10) in section 6.3.1 addresses these topics. In addition, experimental results, presented in section 6.4.4, reveal the effects of neglected thruster control loop dynamics. Hence, in the following it is assumed that the fixed direction thrusters are the only available actuators for control allocation. Under the assumptions discussed above, a general thruster iTh , $i = \overline{1, p}$ is modelled by a modified affine model shown in Figure 5.3.

Vector of forces and moments, exerted by thruster iTh , can be written as:

$${}^i\boldsymbol{\tau} = \begin{bmatrix} {}^i\mathbf{T} \\ {}^i\mathbf{Q} \end{bmatrix} = \begin{bmatrix} T^i \mathbf{e} \\ T^i (\mathbf{r} \times \mathbf{e}) \end{bmatrix} = \begin{bmatrix} {}^i e_x \\ {}^i e_y \\ {}^i e_z \\ (\mathbf{r} \times \mathbf{e})_x \\ (\mathbf{r} \times \mathbf{e})_y \\ (\mathbf{r} \times \mathbf{e})_z \end{bmatrix} {}^iT \quad (5.1)$$

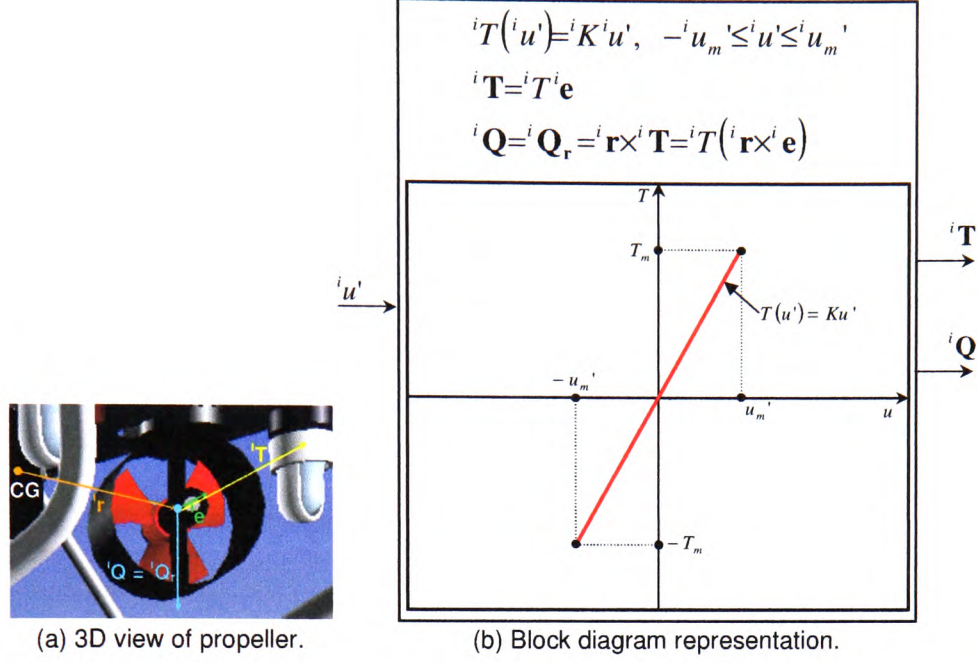


Figure 5.3 Thruster model used in control allocation for underwater vehicles.

Superposition of the individual contributions ${}^i \tau$, $i = \overline{1, p}$ leads to total vector of propulsion forces and moments τ^4 :

$$\tau = \begin{bmatrix} \tau_X \\ \tau_Y \\ \tau_Z \\ \tau_K \\ \tau_M \\ \tau_N \end{bmatrix} = \sum_{i=1}^p {}^i \tau = \sum_{i=1}^p \begin{bmatrix} {}^i e \\ ({}^i r \times {}^i e) \end{bmatrix} {}^i T = \underbrace{\begin{bmatrix} {}^1 e_x & \dots & {}^i e_x & \dots & {}^p e_x \\ {}^1 e_y & \dots & {}^i e_y & \dots & {}^p e_y \\ {}^1 e_z & \dots & {}^i e_z & \dots & {}^p e_z \\ ({}^1 r \times {}^1 e)_x & \dots & ({}^i r \times {}^i e)_x & \dots & ({}^p r \times {}^p e)_x \\ ({}^1 r \times {}^1 e)_y & \dots & ({}^i r \times {}^i e)_y & \dots & ({}^p r \times {}^p e)_y \\ ({}^1 r \times {}^1 e)_z & \dots & ({}^i r \times {}^i e)_z & \dots & ({}^p r \times {}^p e)_z \end{bmatrix}}_{\mathbf{T}} \underbrace{\begin{bmatrix} {}^1 T \\ \vdots \\ {}^i T \\ \vdots \\ {}^p T \end{bmatrix}}_{\mathbf{f}} = \mathbf{T} \mathbf{f} \quad (5.2)$$

where $\mathbf{T} \in \mathbb{R}^{6 \times p}$ is the *thruster configuration matrix*⁵ and $\mathbf{f} \in \mathbb{R}^p$ is vector of control forces. For azimuth thrusters ${}^i e = {}^i e(\alpha)$ and ${}^i r = {}^i r(\alpha)$, which means that ${}^i \tau = {}^i \tau(\alpha)$ and

⁴ Vector τ is equal to ${}^B \tau$ from (3.59), but superscript B is removed in order to improve readability.

⁵ In the general case, when the vehicle is equipped with different types of actuators, the matrix \mathbf{T} is called the *actuator configuration matrix* (Fossen, 2002).

$\mathbf{T} = \mathbf{T}(\alpha)$. However, for fixed direction thrusters $\alpha = \alpha_0 = \text{const.}$ and $\mathbf{T} = \mathbf{T}(\alpha_0) = \text{const.}$

Substituting ${}^i\mathbf{T} = {}^i\mathbf{K}{}^i\mathbf{u}'$ in (5.2) yields

$$\boldsymbol{\tau} = \underbrace{\begin{bmatrix} {}^1\mathbf{e}_x & & {}^i\mathbf{e}_x & & {}^p\mathbf{e}_x \\ {}^1\mathbf{e}_y & \dots & {}^i\mathbf{e}_y & \dots & {}^p\mathbf{e}_y \\ {}^1\mathbf{e}_z & & {}^i\mathbf{e}_z & & {}^p\mathbf{e}_z \\ ({}^1\mathbf{r} \times {}^1\mathbf{e})_x & & ({}^i\mathbf{r} \times {}^i\mathbf{e})_x & & ({}^p\mathbf{r} \times {}^p\mathbf{e})_x \\ ({}^1\mathbf{r} \times {}^1\mathbf{e})_y & \dots & ({}^i\mathbf{r} \times {}^i\mathbf{e})_y & \dots & ({}^p\mathbf{r} \times {}^p\mathbf{e})_y \\ ({}^1\mathbf{r} \times {}^1\mathbf{e})_z & & ({}^i\mathbf{r} \times {}^i\mathbf{e})_z & & ({}^p\mathbf{r} \times {}^p\mathbf{e})_z \end{bmatrix}}_{\mathbf{T}} \underbrace{\begin{bmatrix} {}^1K & 0 & 0 \\ & \ddots & \\ 0 & {}^iK & 0 \\ & \dots & \ddots \\ 0 & 0 & {}^pK \end{bmatrix}}_{\mathbf{K}} \underbrace{\begin{bmatrix} {}^1u' \\ \vdots \\ {}^iu' \\ \vdots \\ {}^pu' \end{bmatrix}}_{\mathbf{u}'} = \mathbf{TKu}' \quad (5.3)$$

where $\mathbf{K} \in \mathfrak{R}^{p \times p}$ is the *force coefficient matrix* and $\mathbf{u}' \in \mathfrak{R}^p$ is the *control vector*.

Introducing

$$\mathbf{B} = \mathbf{TK} \quad (5.4)$$

where $\mathbf{B} \in \mathfrak{R}^{6 \times p}$ is the *thruster control matrix*, (5.3) can be rewritten as

$$\boldsymbol{\tau} = \mathbf{Bu}' \quad (5.5)$$

A zero-row in \mathbf{B} means that the corresponding DOF is not directly controllable with the particular thruster configuration. For example, if the third row of \mathbf{B} is a zero-row, this means that the *heave* DOF is uncontrollable, i.e. the particular thruster configuration is not able to produce non-zero heave force τ_z .

Assuming that thrusters are identical, the coefficients iK are the same for all thrusters,⁶

$${}^1K = \dots = {}^pK = K \quad (5.6)$$

and equation (5.4) can be simplified as

$$\mathbf{B} = \mathbf{TK} = \mathbf{T}(\mathbf{KI}_p) = K(\mathbf{TI}_p) = \mathbf{KT} \quad (5.7)$$

Recall from (3.95) that each component ${}^iu'$ of the control vector \mathbf{u}' is limited by constraint

⁶ This assumption is realistic, since in most cases the open-frame underwater vehicles are equipped with the identical thrusters.

$$-{}^i u_m' \leq {}^i u' \leq {}^i u_m', \quad i = \overline{1, p} \quad (5.8)$$

Constraint (5.8) represents thruster velocity saturation, i.e. the physical construction of the thruster ${}^i Th$ imposes velocity limitations and the thruster cannot rotate faster than the maximum velocity. For the control vector \mathbf{u}' set of constraints (5.8) can be written in compact vector form as

$$-\mathbf{u}_m' \leq \mathbf{u}' \leq \mathbf{u}_m' \quad (5.9)$$

where

$$\mathbf{u}_m' = [{}^1 u_m' \quad \dots \quad {}^i u_m' \quad \dots \quad {}^p u_m']^T \quad (5.10)$$

For identical thrusters, ${}^1 u_m' = \dots = {}^p u_m' = u_m'$ and $\mathbf{u}_m' = u_m' \left[\underbrace{1 \quad \dots \quad 1 \quad \dots \quad 1}_p \right]^T$. Recall that

the constrained control subset Ω is defined as a set of all control vectors \mathbf{u}' which satisfy (5.9). Finally, the general control allocation problem for the open-frame underwater vehicles can be formulated as:

For given $\boldsymbol{\tau}$, find $\mathbf{u}' \in \Omega$ such that $\mathbf{B}\mathbf{u}' = \boldsymbol{\tau}$.

Analogous to the general problem formulation in section 4.3.1, the equation $\mathbf{B}\mathbf{u}' = \boldsymbol{\tau}$ defines the set of hyperplanes in the true control space \mathfrak{R}^p . The intersection of these hyperplanes is a convex set, denoted by \mathfrak{K} . The solution set \mathfrak{S} is given by the intersection of \mathfrak{K} and Ω . From many solutions in \mathfrak{S} it is necessary to select the one that minimises the chosen criteria. The most suitable criteria for open-frame underwater vehicles is control energy cost function, since minimising this criteria means maximising operational battery life, which is very important issue for future underwater vehicles.

The position and orientation vectors ${}^i \mathbf{r}$ and ${}^i \mathbf{e}$ for FALCON and URIS are given in Table 3.5 and Table 3.6, respectively. Thruster configuration matrices for these vehicles, shown

in Table 5.1⁷, are obtained from (5.7), assuming that all thrusters are identical. Parameter

A is defined by $A = \frac{b}{2}\sin\alpha + \frac{a}{2}\cos\alpha$.

	FALCON	URIS
B	$K \begin{bmatrix} \overbrace{\cos\alpha \quad \cos\alpha \quad \cos\alpha \quad \cos\alpha}^{HT} & \underbrace{0}_{VT} \\ \sin\alpha \quad -\sin\alpha \quad \sin\alpha \quad -\sin\alpha & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ A & -A & -A & A & 0 \end{bmatrix}$	$K \begin{bmatrix} \overbrace{1 \quad 1 \quad 0 \quad 0}^{HT} \\ 0 \quad 0 \quad 1 \quad 1 \\ 0 \quad 0 \quad 0 \quad 0 \\ 0 \quad 0 \quad 0 \quad 0 \\ 0 \quad 0 \quad 0 \quad 0 \\ R \quad -R \quad R \quad -R \end{bmatrix}$
	$\underbrace{\hspace{10em}}_T$	$\underbrace{\hspace{10em}}_T$

Table 5.1 Thruster control matrix for different thruster configurations.

It can be seen that the uncontrollable DOF for the FALCON are *roll* and *pitch*, since the fourth and the fifth row of **B** are zero-rows. In a similar way, uncontrollable DOF for the URIS are *heave*, *roll* and *pitch*.

In the following, the general control allocation problem will be separated into two subproblems, which will be treated individually. The first subproblem is related with the motion in the horizontal plane, and the second with the motion in the vertical plane. Decomposition of the motion is given in Table 5.2 for the FALCON and in Table 5.3 for the URIS. Visual interpretation of these decompositions is illustrated in Figure 5.4, where it can be seen that the motion of the FALCON in the vertical plane is determined by the vertical thruster and the heave force is directly proportional to the value of control signal. In the case of a partial fault in a vertical thruster, the only available solution is to limit

⁷ In order to be consistent, the same nomenclature K is used for the force coefficients for both vehicles, although in reality these coefficients have different numerical values.

angular velocity of the thruster. In the case of total fault (failure), the thruster must be switched off and the vehicle must be recovered for repair.

	FALCON
Horizontal thrusters (motion in the horizontal plane)	Controllable DOF: Surge, Sway, Yaw $\tau^{HT} = \begin{bmatrix} \tau_X \\ \tau_Y \\ \tau_N \end{bmatrix} = K \underbrace{\begin{bmatrix} \overbrace{\cos \alpha}^{HT_1} & \overbrace{\cos \alpha}^{HT_2} & \overbrace{\cos \alpha}^{HT_3} & \overbrace{\cos \alpha}^{HT_4} \\ \overbrace{\sin \alpha}^{HT_1} & \overbrace{-\sin \alpha}^{HT_2} & \overbrace{\sin \alpha}^{HT_3} & \overbrace{-\sin \alpha}^{HT_4} \\ A & -A & -A & A \end{bmatrix}}_{B^{HT}} \underbrace{\begin{bmatrix} u^{1,HT} \\ u^{2,HT} \\ u^{3,HT} \\ u^{4,HT} \end{bmatrix}}_{u^{HT}}$
Vertical thruster (motion in the vertical plane)	Controllable DOF: Heave $\tau^{VT} = [\tau_Z] = \underbrace{K}_{B^{VT}} \underbrace{u^{VT}}_{u^{VT}}$

Table 5.2 Decomposition of the the FALCON motion.

	URIS
Horizontal thrusters (motion in the horizontal plane)	Controllable DOF: Surge, Sway, Yaw $\tau^{HT} = \begin{bmatrix} \tau_X \\ \tau_Y \\ \tau_N \end{bmatrix} = K \underbrace{\begin{bmatrix} \overbrace{1}^{HT_1} & \overbrace{1}^{HT_2} & \overbrace{0}^{HT_3} & \overbrace{0}^{HT_4} \\ 0 & 0 & 1 & 1 \\ R & -R & R & -R \end{bmatrix}}_{B^{HT}} \underbrace{\begin{bmatrix} u^{1,HT} \\ u^{2,HT} \\ u^{3,HT} \\ u^{4,HT} \end{bmatrix}}_{u^{HT}}$
Vertical thruster (motion in the vertical plane)	Controllable DOF: -

Table 5.3 Decomposition of the URIS motion.

The situation is different for motion in the horizontal plane, where the number of horizontal thrusters is four and the number of controllable DOF is three. In this case inherent redundancy in thruster configuration enables successful control allocation in the case of partial or even total fault in a horizontal thruster.

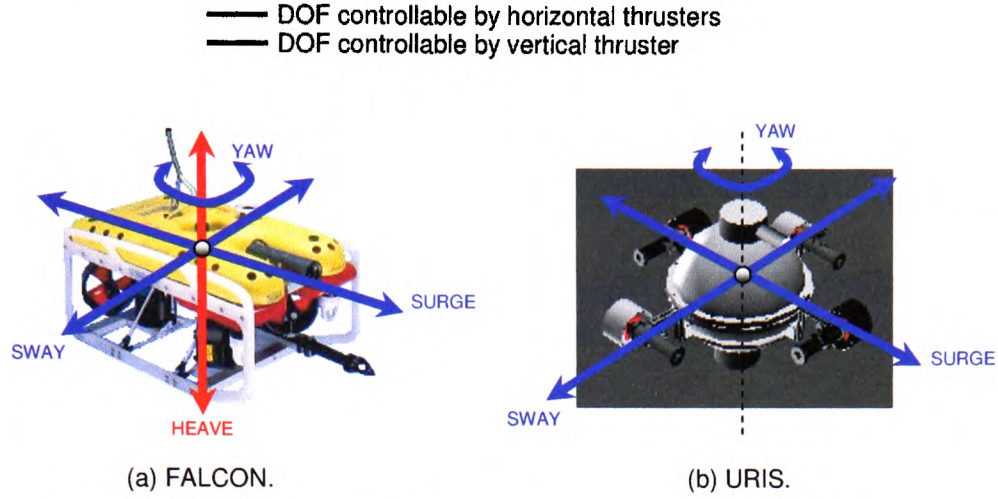


Figure 5.4 Relationship between thruster configuration and controllable DOF.

The control allocation of horizontal thrusters is the topic of the discussion in the following sections. Before the full problem is formulated, relevant vectors and matrices will be normalised, in order to make problem more understandable and easier to visualize and solve.

5.2.2 Normalisation

Normalisation means that vector components are divided by their maximum values, such that each component is dimensionless number that lies between -1 and $+1$. Normalised vectors and matrices are underlined, in order to distinguish them from the standard nomenclature. The normalisation procedure will be explained in details for the X-shaped thruster configuration (FALCON). Recall from Figure 5.3 (b) that maximum thruster force is given by

$$\underline{T}_m = K \underline{u}_m \quad (5.11)$$

The first step is to find maximum values (modules) of the surge and sway forces and the yaw moment. Three characteristic cases are indicated in Figure 5.5. It can be seen that

$$\tau_{x_m} = 4T_m \cos \alpha = 4Ku_m' \cos \alpha \Rightarrow K \cos \alpha = \frac{\tau_{x_m}}{4u_m'} \quad (5.12)$$

$$\tau_{y_m} = 4T_m \sin \alpha = 4Ku_m' \sin \alpha \Rightarrow K \sin \alpha = \frac{\tau_{y_m}}{4u_m'} \quad (5.13)$$

$$\tau_{N_m} = 4T_m A = 4Ku_m' A \Rightarrow KA = \frac{\tau_{N_m}}{4u_m'} \quad (5.14)$$

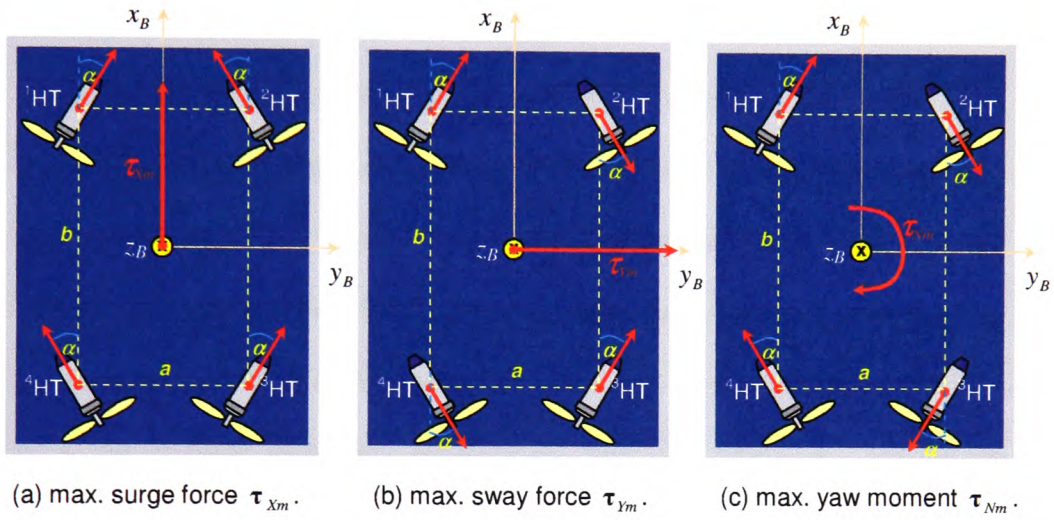


Figure 5.5 Three cases for finding the maximum modules of force and moment vectors (X-shaped thruster configuration).

The second step is to substitute expressions (5.12) - (5.14) in the standard relation

$\tau^{HT} = \mathbf{B}^{HT} \mathbf{u}^{HT}$ as follows:

$$\begin{aligned} \tau^{HT} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_N \end{bmatrix} &= \underbrace{\begin{bmatrix} K \cos \alpha & K \cos \alpha & K \cos \alpha & K \cos \alpha \\ K \sin \alpha & -K \sin \alpha & K \sin \alpha & -K \sin \alpha \\ KA & -KA & -KA & KA \end{bmatrix}}_{\mathbf{B}^{HT}} \underbrace{\begin{bmatrix} u^{HT}_1 \\ u^{HT}_2 \\ u^{HT}_3 \\ u^{HT}_4 \end{bmatrix}}_{\mathbf{u}^{HT}} = \\ &= \begin{bmatrix} \frac{\tau_{x_m}}{4u_m'} & \frac{\tau_{x_m}}{4u_m'} & \frac{\tau_{x_m}}{4u_m'} & \frac{\tau_{x_m}}{4u_m'} \\ \frac{\tau_{y_m}}{4u_m'} & -\frac{\tau_{y_m}}{4u_m'} & \frac{\tau_{y_m}}{4u_m'} & -\frac{\tau_{y_m}}{4u_m'} \\ \frac{\tau_{N_m}}{4u_m'} & -\frac{\tau_{N_m}}{4u_m'} & -\frac{\tau_{N_m}}{4u_m'} & \frac{\tau_{N_m}}{4u_m'} \end{bmatrix} \begin{bmatrix} u^{HT}_1 \\ u^{HT}_2 \\ u^{HT}_3 \\ u^{HT}_4 \end{bmatrix} \end{aligned} \quad (5.15)$$

The final step is to rewrite (5.15) in the normalised form as follows:

$$\underbrace{\begin{bmatrix} \tau_x \\ \tau_{xm} \\ \tau_y \\ \tau_{ym} \\ \tau_N \\ \tau_{Nm} \end{bmatrix}}_{\underline{\tau}^{HT}} = \underbrace{\begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \end{bmatrix}}_{\underline{\mathbf{B}}^{HT}} \underbrace{\begin{bmatrix} u_m' \\ u_m' \\ u_m' \\ u_m' \end{bmatrix}}_{\underline{u}^{HT}} \Leftrightarrow \underline{\tau}^{HT} = \underline{\mathbf{B}}^{HT} \underline{u}^{HT} \quad (5.16)$$

Formulation of the control allocation problem using a normalised form (5.16) has a number of advantages compared to the standard form $\underline{\tau}^{HT} = \underline{\mathbf{B}}^{HT} \underline{u}^{HT}$ as follows:

1. Components of the vectors $\underline{\tau}^{HT}$ and \underline{u}^{HT} are dimensionless number, restricted to the standard interval $[-1, +1]$. This enables better understanding and easier visualisation of the problem.
2. All physical parameters are removed from the matrix $\underline{\mathbf{B}}^{HT}$ during the normalisation process. The compact form of $\underline{\mathbf{B}}^{HT}$ simplifies calculations and, as will be shown later, leads to the very simple representation of the weighted pseudoinverse solution and a clear geometric interpretation of the control allocation problem.

Normalisation for the cross-shaped thruster configuration (URIS) can be performed in a similar way. As indicated in Figure 5.6, the maximum surge and sway forces and yaw moment are given by

$$\tau_{xm} = 2T_m = 2Ku_m' \Rightarrow K = \frac{\tau_{xm}}{2u_m'} \quad (5.17)$$

$$\tau_{ym} = 2T_m = 2Ku_m' \Rightarrow K = \frac{\tau_{ym}}{2u_m'} \quad (5.18)$$

$$\tau_{Nm} = 4T_m R = 4Ku_m' R \Rightarrow KR = \frac{\tau_{Nm}}{4u_m'} \quad (5.19)$$

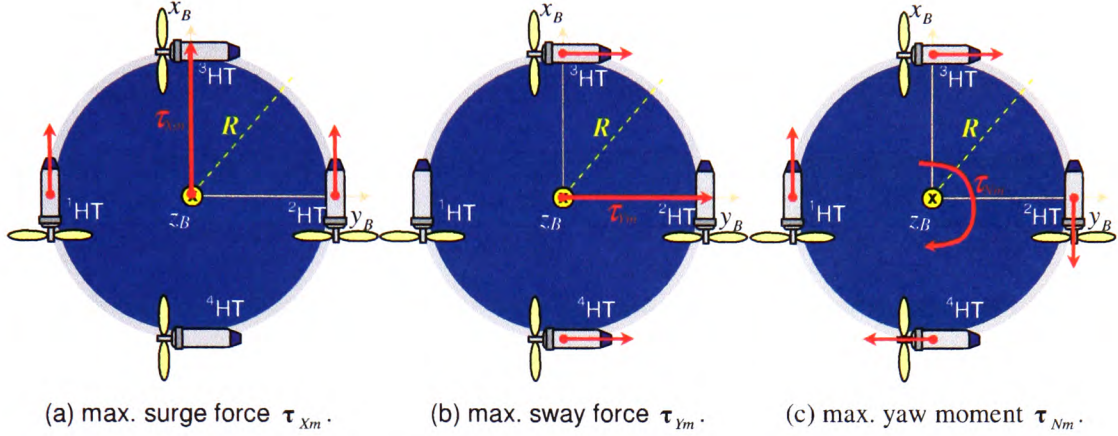


Figure 5.6 Three cases for finding the maximum modules of force and moment vectors (cross-shaped thruster configuration).

Substituting these expressions in the standard relation $\tau^{HT} = \mathbf{B}^{HT} \mathbf{u}^{HT}$ yields

$$\begin{aligned} \tau^{HT} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_N \end{bmatrix} &= \underbrace{\begin{bmatrix} K & K & 0 & 0 \\ 0 & 0 & K & K \\ KR & -KR & KR & -KR \end{bmatrix}}_{\mathbf{B}^{HT}} \underbrace{\begin{bmatrix} u^{1,HT} \\ u^{2,HT} \\ u^{3,HT} \\ u^{4,HT} \end{bmatrix}}_{\mathbf{u}^{HT}} = \\ &= \begin{bmatrix} \frac{\tau_{xm}}{2u_m'} & \frac{\tau_{xm}}{2u_m'} & 0 & 0 \\ 0 & 0 & \frac{\tau_{ym}}{2u_m'} & \frac{\tau_{ym}}{2u_m'} \\ \frac{\tau_{Nm}}{4u_m'} & -\frac{\tau_{Nm}}{4u_m'} & \frac{\tau_{Nm}}{4u_m'} & -\frac{\tau_{Nm}}{4u_m'} \end{bmatrix} \begin{bmatrix} u^{1,HT} \\ u^{2,HT} \\ u^{3,HT} \\ u^{4,HT} \end{bmatrix} \end{aligned} \quad (5.20)$$

Finally, the normalised form for the cross-shaped thruster configuration is given by

$$\underbrace{\begin{bmatrix} \tau_x \\ \tau_{xm} \\ \tau_y \\ \tau_{ym} \\ \tau_N \\ \tau_{Nm} \end{bmatrix}}_{\tau^{HT}} = \underbrace{\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \end{bmatrix}}_{\mathbf{B}^{HT}} \underbrace{\begin{bmatrix} u_m' \\ u^{1,HT} \\ u_m' \\ u^{2,HT} \\ u_m' \\ u^{3,HT} \\ u_m' \\ u^{4,HT} \\ u_m' \\ u^{HT} \end{bmatrix}}_{\mathbf{u}^{HT}} \Leftrightarrow \tau^{HT} = \mathbf{B}^{HT} \mathbf{u}^{HT} \quad (5.21)$$

5.2.3 Problem formulation

The control allocation problem for the motion in the horizontal plane can be formulated using normalised variables as follows:

For given $\underline{\tau}^{HT}$, find $\underline{u}^{HT} \in \underline{\Omega}^{HT}$ such that $\underline{B}^{HT} \underline{u}^{HT} = \underline{\tau}^{HT}$.

In the following, the problem is analysed in more detail from the general control allocation perspective.

- The true control input is $\underline{u}^{HT} = \begin{bmatrix} \underline{u}_1^{HT} \\ \underline{u}_2^{HT} \\ \underline{u}_3^{HT} \\ \underline{u}_4^{HT} \end{bmatrix} = \begin{bmatrix} \frac{1}{u_m^{HT}} \\ \frac{2}{u_m^{HT}} \\ \frac{3}{u_m^{HT}} \\ \frac{4}{u_m^{HT}} \end{bmatrix} \in \mathbb{R}^4 \quad (m=4),$

- The virtual control input is $\underline{\tau}^{HT} = \begin{bmatrix} \tau_X \\ \tau_Y \\ \tau_N \end{bmatrix} = \begin{bmatrix} \frac{\tau_X}{\tau_{Xm}} \\ \frac{\tau_Y}{\tau_{Ym}} \\ \frac{\tau_N}{\tau_{Nm}} \end{bmatrix} \in \mathbb{R}^3 \quad (k=3),$

- The control effectiveness matrix \underline{B}^{HT} is given by

$$\text{FALCON: } \underline{B}^{HT} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \end{bmatrix} \quad (5.22)$$

$$\text{URIS:} \quad \underline{\mathbf{B}}^{HT} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \end{bmatrix} \quad (5.23)$$

$$\bullet \text{ Actuator position constraints are } \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \leq \begin{bmatrix} \underline{u}_1^{HT} \\ \underline{u}_2^{HT} \\ \underline{u}_3^{HT} \\ \underline{u}_4^{HT} \end{bmatrix} \leq \begin{bmatrix} +1 \\ +1 \\ +1 \\ +1 \end{bmatrix}.$$

Equation $\underline{\mathbf{B}}^{HT} \underline{\mathbf{u}}^{HT} = \underline{\boldsymbol{\tau}}^{HT}$ represents the system of equations

$$\begin{aligned} \text{FALCON:} \quad & \frac{1}{4}\underline{u}_1^{HT} + \frac{1}{4}\underline{u}_2^{HT} + \frac{1}{4}\underline{u}_3^{HT} + \frac{1}{4}\underline{u}_4^{HT} = \underline{\tau}_X \\ & \frac{1}{4}\underline{u}_1^{HT} - \frac{1}{4}\underline{u}_2^{HT} + \frac{1}{4}\underline{u}_3^{HT} - \frac{1}{4}\underline{u}_4^{HT} = \underline{\tau}_Y \\ & \frac{1}{4}\underline{u}_1^{HT} - \frac{1}{4}\underline{u}_2^{HT} - \frac{1}{4}\underline{u}_3^{HT} + \frac{1}{4}\underline{u}_4^{HT} = \underline{\tau}_N \end{aligned} \quad (5.24)$$

$$\begin{aligned} \text{URIS:} \quad & \frac{1}{2}\underline{u}_1^{HT} + \frac{1}{2}\underline{u}_2^{HT} = \underline{\tau}_X \\ & \frac{1}{2}\underline{u}_3^{HT} + \frac{1}{2}\underline{u}_4^{HT} = \underline{\tau}_Y \\ & \frac{1}{4}\underline{u}_1^{HT} - \frac{1}{4}\underline{u}_2^{HT} + \frac{1}{4}\underline{u}_3^{HT} - \frac{1}{4}\underline{u}_4^{HT} = \underline{\tau}_N \end{aligned} \quad (5.25)$$

Each equation in (5.24) & (5.25) represents a hyperplane in \mathcal{R}^4 . Consequently, (5.24) &

(5.25) can be rewritten as

$$\begin{aligned} \pi_X: \quad & \mathbf{N}_X^T \cdot \underline{\mathbf{u}}^{HT} = \underline{\tau}_X \\ \pi_Y: \quad & \mathbf{N}_Y^T \cdot \underline{\mathbf{u}}^{HT} = \underline{\tau}_Y \\ \pi_N: \quad & \mathbf{N}_N^T \cdot \underline{\mathbf{u}}^{HT} = \underline{\tau}_N \end{aligned} \quad (5.26)$$

where normal vectors \mathbf{N}_X , \mathbf{N}_Y and \mathbf{N}_N , defined as

$$\begin{aligned}
 & \mathbf{N}_x = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}^T \\
 \text{FALCON: } & \mathbf{N}_y = \begin{bmatrix} \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \end{bmatrix}^T \\
 & \mathbf{N}_z = \begin{bmatrix} \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \end{bmatrix}^T
 \end{aligned} \tag{5.27}$$

$$\begin{aligned}
 & \mathbf{N}_x = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{bmatrix}^T \\
 \text{URIS: } & \mathbf{N}_y = \begin{bmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}^T \\
 & \mathbf{N}_z = \begin{bmatrix} \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \end{bmatrix}^T
 \end{aligned} \tag{5.28}$$

are orthogonal on the hyperplanes π_x , π_y and π_z , respectively. The intersection of these hyperplanes is a convex set, denoted by $\underline{\mathbf{x}}^{HT}$, which represents the set of all points $\underline{\mathbf{u}}^{HT}$ that satisfy $\underline{\mathbf{B}}^{HT} \underline{\mathbf{u}}^{HT} = \underline{\boldsymbol{\tau}}^{HT}$.

The actuator position constraints determine the constrained control subset $\underline{\Omega}^{HT}$, that is, the unit four-dimensional hypercube in \mathbb{R}^4 :

$$\underline{\Omega}^{HT} = \left\{ \underline{\mathbf{u}}^{HT} \in \mathbb{R}^4 \mid \|\underline{\mathbf{u}}^{HT}\|_{\infty} \leq 1 \right\} \subset \mathbb{R}^4 \tag{5.29}$$

Intersection of $\underline{\mathbf{x}}^{HT}$ and $\underline{\Omega}^{HT}$ is a solution set, denoted by $\underline{\mathfrak{S}}^{HT}$. The geometric interpretation of the control allocation problem for the motion in the horizontal plane using normalised variables is given by:

For a given $\underline{\boldsymbol{\tau}}^{HT}$, find intersection $\underline{\mathfrak{S}}^{HT} = \underline{\mathbf{x}}^{HT} \cap \underline{\Omega}^{HT}$.

Recall from section 4.3.1 that the control effectiveness matrix $\underline{\mathbf{B}}^{HT}$ performs a linear transformation from the true control space \mathbb{R}^m to the virtual control space \mathbb{R}^k and that the image of $\underline{\Omega}^{HT} \subset \mathbb{R}^k$ is called the attainable command set and denoted by $\underline{\Phi}^{HT}$.

5.2.4 Nomenclature

The nomenclature for the constrained control subset $\underline{\Omega}^{HT}$ and the attainable command set $\underline{\Phi}^{HT}$ was introduced in sections 4.3.2 and 4.3.3, respectively. The shape of $\underline{\Omega}^{HT}$ is the same for both configurations, while the shape of $\underline{\Phi}^{HT}$ varies with configuration. Unfortunately, $\underline{\Omega}^{HT}$ is the four-dimensional hypercube and cannot be easily visualised in contrast to $\underline{\Phi}^{HT}$ that is shown in Figure 5.7 (for the X-shaped thruster configuration) and Figure 5.8 (for the cross-shaped thruster configuration). The coordinates of vertices of $\underline{\Omega}^{HT}$ and $\underline{\Phi}^{HT}$ are given in Table 5.4. Using the MATLAB command `convhulln`, it was found that a convex hull of $\underline{\Phi}^{HT}$ consists of all but **3** and **C** vertices for X-shaped configuration, and **6** and **9** vertices for cross-shaped configuration. Hence, nodes **3** and **C** (**6** and **9**) lie inside $\underline{\Phi}^{HT}$ for the X-shaped (cross-shaped) configuration.

The linear independency condition (section 2.9.2) is satisfied for both configurations, since every 3×3 partition of $\underline{\mathbf{B}}^{HT}$ is non-singular. This means that the equation $\underline{\mathbf{B}}^{HT} \underline{\mathbf{u}}^{HT} = \underline{\boldsymbol{\tau}}^{HT}$ has a unique solution $\underline{\mathbf{u}}^{HT} \in \partial(\underline{\Omega}^{HT})$ on the boundary $\underline{\boldsymbol{\tau}}^{HT} \in \partial(\underline{\Phi}^{HT})$.

In the general case, the character of the solution depends on the position of the vector $\underline{\boldsymbol{\tau}}^{HT}$ relative to $\underline{\Phi}^{HT}$. Three cases are possible:

1. If $\underline{\boldsymbol{\tau}}^{HT}$ lies inside $\underline{\Phi}^{HT}$, then the solution set $\underline{\mathcal{S}}^{HT}$ has infinite number of points and the control allocation problem has an infinite number of solutions.
2. If $\underline{\boldsymbol{\tau}}^{HT}$ lies on the boundary $\partial(\underline{\Phi}^{HT})$, then solution set $\underline{\mathcal{S}}^{HT}$ is a single point, which is a unique solution of the control allocation problem.
3. Finally, if $\underline{\boldsymbol{\tau}}^{HT}$ lies outside $\underline{\Phi}^{HT}$, i.e. if $\underline{\boldsymbol{\tau}}^{HT} \in \underline{\Phi}_v^{HT} \setminus \underline{\Phi}^{HT}$, then solution set $\underline{\mathcal{S}}^{HT}$ is an empty set, i.e. no exact solution exist.

Using the hybrid approach, described in Example 4.10, the FDAS is able to find the exact solution of the problem for the cases 1. and 2., optimal in the l_2 sense, and a good approximate solution for the case 3.

$\underline{\Omega}^{HT}$	$\underline{\Phi}^{HT}$	
	X-shaped configuration	Cross-shaped configuration
0 -1 -1 -1 -1	0 -1.0 0.0 0.0	0 -1.0 -1.0 0.0
1 -1 -1 -1 1	1 -0.5 -0.5 0.5	1 -1.0 0.0 -0.5
2 -1 -1 1 -1	2 -0.5 0.5 -0.5	2 -1.0 0.0 0.5
3 -1 -1 1 1	3 0.0 0.0 0.0	3 -1.0 1.0 0.0
4 -1 1 -1 -1	4 -0.5 -0.5 -0.5	4 0.0 -1.0 -0.5
5 -1 1 -1 1	5 0.0 -1.0 0.0	5 0.0 0.0 -1.0
6 -1 1 1 -1	6 0.0 0.0 -1.0	6 0.0 0.0 0.0
7 -1 1 1 1	7 0.5 -0.5 -0.5	7 0.0 1.0 -0.5
8 1 -1 -1 -1	8 -0.5 0.5 0.5	8 0.0 -1.0 0.5
9 1 -1 -1 1	9 0.0 0.0 1.0	9 0.0 0.0 0.0
A 1 -1 1 -1	A 0.0 1.0 0.0	A 0.0 0.0 1.0
B 1 -1 1 1	B 0.5 0.5 0.5	B 0.0 1.0 0.5
C 1 1 -1 -1	C 0.0 0.0 0.0	C 1.0 -1.0 0.0
D 1 1 -1 1	D 0.5 -0.5 0.5	D 1.0 0.0 -0.5
E 1 1 1 -1	E 0.5 0.5 -0.5	E 1.0 0.0 0.5
F 1 1 1 1	F 1.0 0.0 0.0	F 1.0 1.0 0.0

Table 5.4 Coordinates of vertices of $\underline{\Omega}^{HT}$ and $\underline{\Phi}^{HT}$.

The nomenclature for $\underline{\Omega}^{HT}$ is given in Table 5.5. Recall from section 4.3.2 that the component values (controls) of \underline{u}^{HT} in a vertex can be decoded from its binary representation. For example, decoding the binary representation 1000 of the vertex **8** yields $\underline{u}_1^{HT} = +1$, $\underline{u}_2^{HT} = -1$, $\underline{u}_3^{HT} = -1$ and $\underline{u}_4^{HT} = -1$. Vertices **i** and **j** are connected by edge **ij** if and only if their binary representations differ in only one bit. For example, vertices **1** (0001) and **5** (0101) are connected by edge **15**, since their binary representations differ in the second bit.

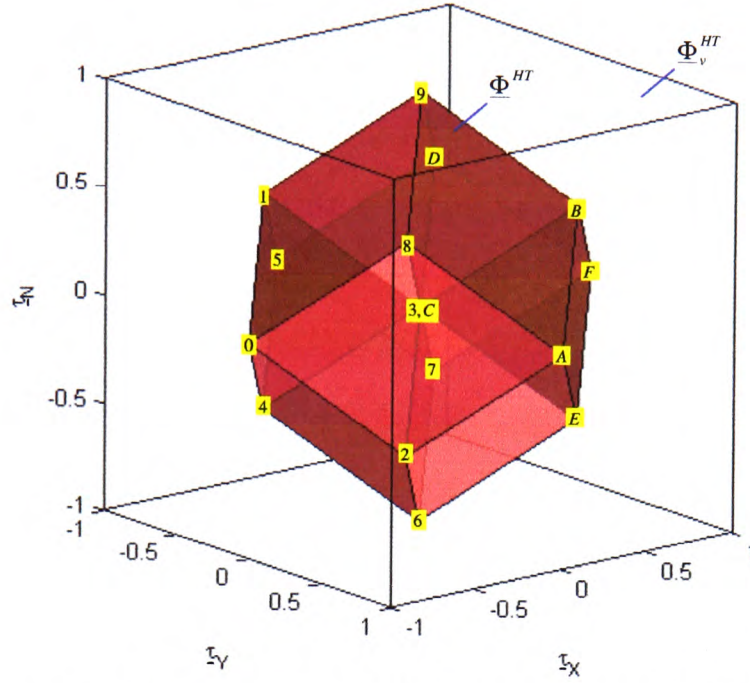


Figure 5.7 Attainable command set Φ^{HT} for the X-shaped thruster configuration.

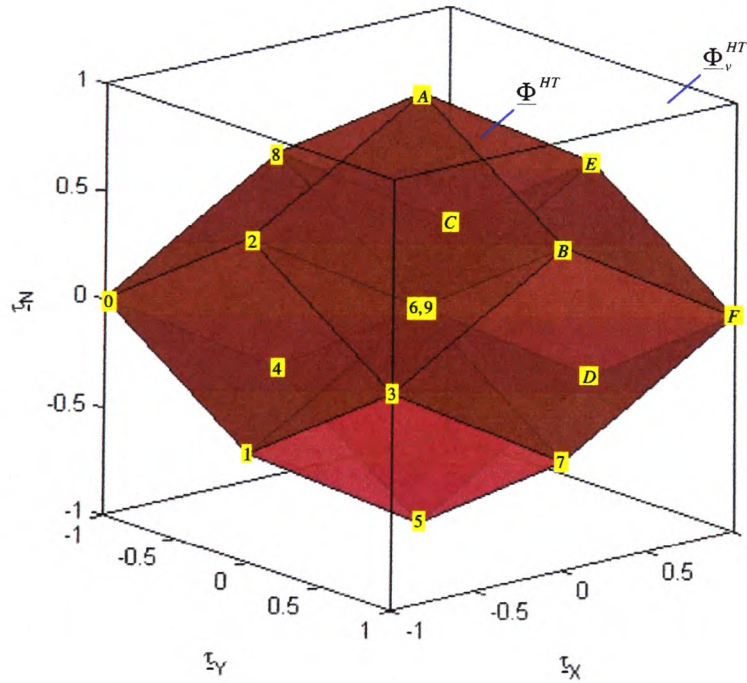


Figure 5.8 Attainable command set Φ^{HT} for the cross-shaped thruster configuration.

A facet is defined as a set of points on $\partial(\underline{\Omega}^{HT})$ obtained by taking $m - 2 = 2$ controls at their limits and varying the two free controls within their limits. For example, the facet 0132 is a rectangle, determined by vertices 0 (0000), 1 (0001), 3 (0011) and 2 (0010). It can be seen that the first two controls are fixed to limit values $\underline{u}_1^{HT} = -1$ and $\underline{u}_2^{HT} = -1$, and two free controls \underline{u}_3^{HT} and \underline{u}_4^{HT} are free to vary, i.e. coordinates of any point which lie on the facet 0132 have the form $[-1 \ -1 \ q_3 \ q_4]^T$, where $q_3, q_4 \in [-1, 1]$.

Vertices	Edges	Facets
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	01, 02, 04, 08, 13, 15, 19, 23, 26, 2A, 37, 3B, 45, 46, 4C, 57, 5D, 67, 6E, 7F, 89, 8A, 8C, 9B, 9D, AB, AE, BF, CD, CE, DF, EF	0132, 89BA, 4576, CDFE, 0198, 23BA, 45DC, 67FE, 04C8, 15D9, 26EA, 37FB, 0154, 2376, 89DC, ABFE, 0264, 1375, 8AEC, 9BFD, 02A8, 13B9, 46EC, 57FD
Total: $2^4 = 16$ vertices	Total: $2^{4-1} \binom{4}{1} = 32$ edges	Total: $2^{4-2} \binom{4}{2} = 24$ facets

Table 5.5 Nomenclature for $\underline{\Omega}^{HT}$.

Vertices	Nodes	Edges	Connections	Facets	Faces
0, 1, 2, 4, 5, 6, 7, 8, 9, A, B, D, E, F	3, C	01, 02, 04, 08, 15, 19, 26, 2A, 45, 46, 57, 5D, 67, 6E, 7F, 89, 8A, 9B, 9D, AB, AE, BF, DF, EF	13, 23, 37, 3B, 4C, 8C, CD, CE	89BA, 4576, 0198, 67FE, 15D9, 26EA, 0154, ABFE, 0264, 9BFD, 02A8, 57FD	0132, CDFE, 23BA, 45DC, 04C8, 37FB, 2376, 89DC, 1375, 8AEC, 13B9, 46EC

Table 5.6 Nomenclature for $\underline{\Phi}^{HT}$ (X-shaped thruster configuration).

Vertices	Nodes	Edges	Connections	Facets	Faces
0, 1, 2, 3, 4, 5, 7, 8, A, B, C, D, E, F	6, 9	01, 02, 04, 08, 13, 15, 23, 2A, 37, 3B, 45, 4C, 57, 5D, 7F, 8A, 8C, AB, AE, BF, CD, CE, DF, EF	19, 26, 46, 67, 6E, 89, 9B, 9D	0132, CDFE, 23BA, 45DC, 04C8, 37FB, 0154, ABFE, 1375, 8AEC, 02A8, 57FD	89BA, 4576, 0198, 67FE, 15D9, 26EA, 2376, 89DC, 0264, 9BFD, 13B9, 46EC

Table 5.7 Nomenclature for $\underline{\Phi}^{HT}$ (cross-shaped thruster configuration).

The nomenclature for Φ^{HT} is given in Table 5.6 (for X-shaped thruster configuration) and Table 5.7 (for cross-shaped thruster configuration).

5.2.5 Introducing criteria in problem formulation

It was mentioned in section 5.2.3 that the character of the solution is closely related with the position of the vector $\underline{\tau}^{HT}$ relative to Φ^{HT} . In order to extract a unique, "best" solution from a solution set, it is necessary to introduce criteria, which is minimised by the chosen solution. The most suitable criteria for underwater applications is a control energy cost function, as stated previously.

In the general case, the optimal control input \underline{u}^{HT} is given as a solution to a two-step optimisation problem

$$\underline{u}^{HT} = \arg \min_{\underline{u}^{HT} \in \Psi^{HT}} \|\mathbf{W}_u^{HT} \underline{u}^{HT}\|_2 \quad (5.30)$$

$$\underline{\Psi}^{HT} = \arg \min_{\underline{u}^{HT} \in \Omega^{HT}} \|\mathbf{B}^{HT} \underline{u}^{HT} - \underline{\tau}^{HT}\|_2 \quad (5.31)$$

The general formulation (5.30) - (5.31) of the control allocation problem for motion in the horizontal plane is obtained from (4.20) – (4.21) assuming $p = 2$, $\mathbf{u}_p = \mathbf{0}$ and $\mathbf{W}_v = \mathbf{I}_3$.

The first assumption $p = 2$ means that the l_2 norm is used as a measure of how good a solution (or approximation) is. This norm represents a measure of control effort and (5.30) can be interpreted as control energy cost function. The second assumption $\mathbf{u}_p = \mathbf{0}$ means that the non-actuated state of the horizontal thrusters is a preferred state (preferred virtual control input). The third assumption $\mathbf{W}_v = \mathbf{I}_3$ means that the same priority is given to all horizontal thrusters in the case when the problem (5.30) - (5.31) has no exact solution. This is reasonable, since all horizontal thrusters of the FALCON and the URIS are identical and have the same priority level in a fault-free case.

The problem (5.30) - (5.31) can be interpreted as follows:

Given $\underline{\Psi}^{HT}$, the set of feasible control inputs that minimise $\|\underline{\mathbf{B}}^{HT} \underline{\mathbf{u}}^{HT} - \underline{\mathbf{r}}^{HT}\|_2$, find the control input $\underline{\mathbf{u}}^{HT} \in \underline{\Psi}^{HT}$ that minimises $\|\underline{\mathbf{W}}_u^{HT} \underline{\mathbf{u}}^{HT}\|_2$.

The design parameter $\underline{\mathbf{W}}_u^{HT}$ is a positive definite weighting matrix, weighting the control energy, and can be used for thruster prioritisation, i.e. to decide which thruster should be used primarily. The weighting matrix $\underline{\mathbf{W}}_u^{HT}$ is usually chosen to be a diagonal matrix

$$\underline{\mathbf{W}}_u^{HT} = \begin{bmatrix} w_1^{HT} & 0 & 0 & 0 \\ 0 & w_2^{HT} & 0 & 0 \\ 0 & 0 & w_3^{HT} & 0 \\ 0 & 0 & 0 & w_4^{HT} \end{bmatrix} \quad (5.32)$$

where $w_i^{HT} > 0$ is the weight associated with the thruster i^{HT} , $i = \overline{1,4}$. Using $\underline{\mathbf{W}}_u^{HT}$, a faulty thruster is penalised by increasing its weight, as explained in the following.

Weighting matrix $\underline{\mathbf{W}}_u^{HT}$ for fault-free case

In the fault-free case, all horizontal thrusters have the same priority and $\underline{\mathbf{W}}_u^{HT}$ is chosen to be equal to identity matrix

$$\underline{\mathbf{W}}_u^{HT} = \underline{\mathbf{I}}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.33)$$

Attainable command sets, shown in Figure 5.7 and Figure 5.8, are found assuming (5.33).

Weighting matrix $\underline{\mathbf{W}}_u^{HT}$ for faulty situations

Two faulty situations are possible: a *partial* fault and a *total* fault (failure).

In the case of a partial fault in i^{HT} , the thruster is typically allowed to continue operation with the restricted usage, i.e. the new constraint (saturation) bounds are:

$$-s_i^{HT} \leq \underline{u}_i^{HT} \leq s_i^{HT} \quad (5.34)$$

where $0 < s_i^{HT} < 1$. The numerical value of s_i^{HT} depends on the type of the fault and is selected in advance for each particular fault type. For example, restricted constraint bound $s_i^{HT} = 0.75$ can be selected for a faulty state "Jammed propeller" of i^{HT} , which means that the thruster's operating range is restricted to 75% of its nominal range. In addition to the change of the constraint bounds, the weight w_i^{HT} of the faulty thruster is increased using

$$w_i^{HT} = 1 + \Delta w_i^{HT} \quad (5.35)$$

where

$$\Delta w_i^{HT} = 2 \cdot \left(\frac{1}{s_i^{HT}} - 1 \right) \quad (5.36)$$

The weight update (5.35) is introduced to penalise the faulty thruster, prioritise healthy thrusters and to compensate for restricted usage of the faulty thruster in an optimal way.

Formula (5.36) is obtained from the weighted pseudoinverse solution $\underline{\mathbf{B}}_{w_i^{HT}}^{HT}$, analysing the components of $\underline{\mathbf{u}}^{HT}$ and the desired restriction of saturation bounds in faulty situations.

In the case of a total fault in i^{HT} , the thruster is switched off and removed from the allocation process. The same effect can be achieved using formulation (5.30) by allowing $w_i^{HT} \rightarrow \infty$. In this way, the redundancy is eliminated from the system of equation $\underline{\mathbf{B}}^{HT} \underline{\mathbf{u}}^{HT} = \underline{\mathbf{r}}^{HT}$, which can now be solved in a standard way.

Geometric interpretation

It is useful to give a geometric interpretation of the relationship between the choice of the weighting matrix \mathbf{W}_u^{HT} and the solution of the control allocation problem for fault-free and faulty situations. Recall that a set of points $\underline{\mathbf{u}}^{HT}$ that satisfy $\|\mathbf{W}_u^{HT} \underline{\mathbf{u}}^{HT}\|_2 \leq r$ is called

a weighted sphere $S_{\mathbf{W}_u^{HT}}(\mathbf{0}, r)_2$ with the centre at $\mathbf{0}$ and the radius r (Definition B.2, Appendix B).

In a fault-free case, the choice $\mathbf{W}_u^{HT} = \mathbf{I}_4$ means that $S_{\mathbf{W}_u^{HT}}(\mathbf{0}, r)_2$ is the standard sphere with regular shape, similar to those shown in Figure 4.10. The touching point of a sphere with the solution set $\underline{\mathcal{S}}^{HT}$ is a solution that minimises $\|\underline{\mathbf{u}}^{HT}\|_2$.

In a faulty situation, the faulty thruster is penalised with weight increase and restriction of constraint bounds. This means that $S_{\mathbf{W}_u^{HT}}(\mathbf{0}, r)_2$ is compressed in the direction of penalised control, like the sphere shown in Figure 4.11. The restriction of constraint bounds reduces the size of the constrained control set $\underline{\Omega}^{HT}$. The touching point of a deformed (compressed) sphere with the solution set $\underline{\mathcal{S}}^{HT}$ that lies inside restricted $\underline{\Omega}^{HT}$ is a solution that minimises $\|\mathbf{W}_u^{HT} \underline{\mathbf{u}}^{HT}\|_2$.

5.2.6 Remarks on vertical thrusters

For the motion of FALCON in the vertical plane normalisation yields

$$\left. \begin{aligned} \boldsymbol{\tau}^{VT} = [\tau_z] &= \underbrace{K}_{\mathbf{B}^{VT} \mathbf{u}^{VT}} \underbrace{\mathbf{u}^{VT}}_{\mathbf{u}_m^{VT}} \\ K &= \frac{\tau_{zm}}{\mathbf{u}_m^{VT}} \end{aligned} \right\} \Rightarrow \underbrace{\begin{bmatrix} \tau_z \\ \tau_{zm} \end{bmatrix}}_{\boldsymbol{\tau}^{VT}} = \underbrace{\begin{bmatrix} 1 \\ \mathbf{u}_m^{VT} \end{bmatrix}}_{\mathbf{B}^{VT}} \underbrace{\left[\frac{\mathbf{u}^{VT}}{\mathbf{u}_m^{VT}} \right]}_{\mathbf{u}^{VT}} \Leftrightarrow \boldsymbol{\tau}^{VT} = \mathbf{B}^{VT} \mathbf{u}^{VT} \quad (5.37)$$

The weighting matrix for vertical thruster \mathbf{W}_u^{VT} becomes a scalar, defined as

$$\mathbf{W}_u^{VT} = w_1^{VT} \quad (5.38)$$

where $w_1^{VT} > 0$ is the weight associated with the thruster 1^{VT} .

In the fault-free case, $w_1^{VT} = 1$.

In the case of a partial fault, the new restricted constraint (saturation) bounds are:

$$-s_1^{VT} \leq \underline{\mathbf{u}}_1^{VT} \leq s_1^{VT} \quad (5.39)$$

where $0 < s_1^{VT} < 1$. Since $\underline{\mathbf{B}}^{VT} = 1$, the weighted pseudoinverse solution $\underline{\mathbf{B}}_{w_1^{VT}}^{+VT}$ does not depend on w_1^{VT} . Hence, restriction of the constraint bounds (5.39) is the only action that is undertaken in the case of a partial fault in the vertical thruster.

In the case of a total fault, the thruster VT must be switched off and removed from the allocation process. In this case, heave (depth) becomes uncontrollable DOF.

5.3 Architecture of the FDAS

The overall functional architecture of the proposed FDAS, shown in Figure 5.9, represents the expanded version of the improved architecture shown in Figure 5.1 (b). The description of the architecture will be given in a hierarchical way, such that the general description and the main idea of the method are presented in this section, while more details about individual components of the overall system can be found in the following sections.

The FDAS consists of two subsystems: the FAS and the FDS. Essentially, the FAS performs the control allocation task, but this primary task is enhanced with the ability to perform automatic reconfiguration in the case of a partial or total fault in a single thruster. The thruster states are obtained from the FDS.

The FDS uses FDUs, associated with each thruster, to monitor their state. The outputs of the FDUs are integrated inside the FDS into the total fault indicator vector, carrying the codes of faulty states for each thruster. Relationships between fault types and remedial actions are stored in the fault code table. The architecture shown in Figure 5.9 can be easily expanded for vehicles with more vertical thrusters.

The input to the FDAS is the vector $\underline{\tau}_d$, obtained by filtering (smoothing) the desired vector of propulsion forces and moments $\underline{\tau}_d^*$, generated by the HCU (see discussion in section 5.1).

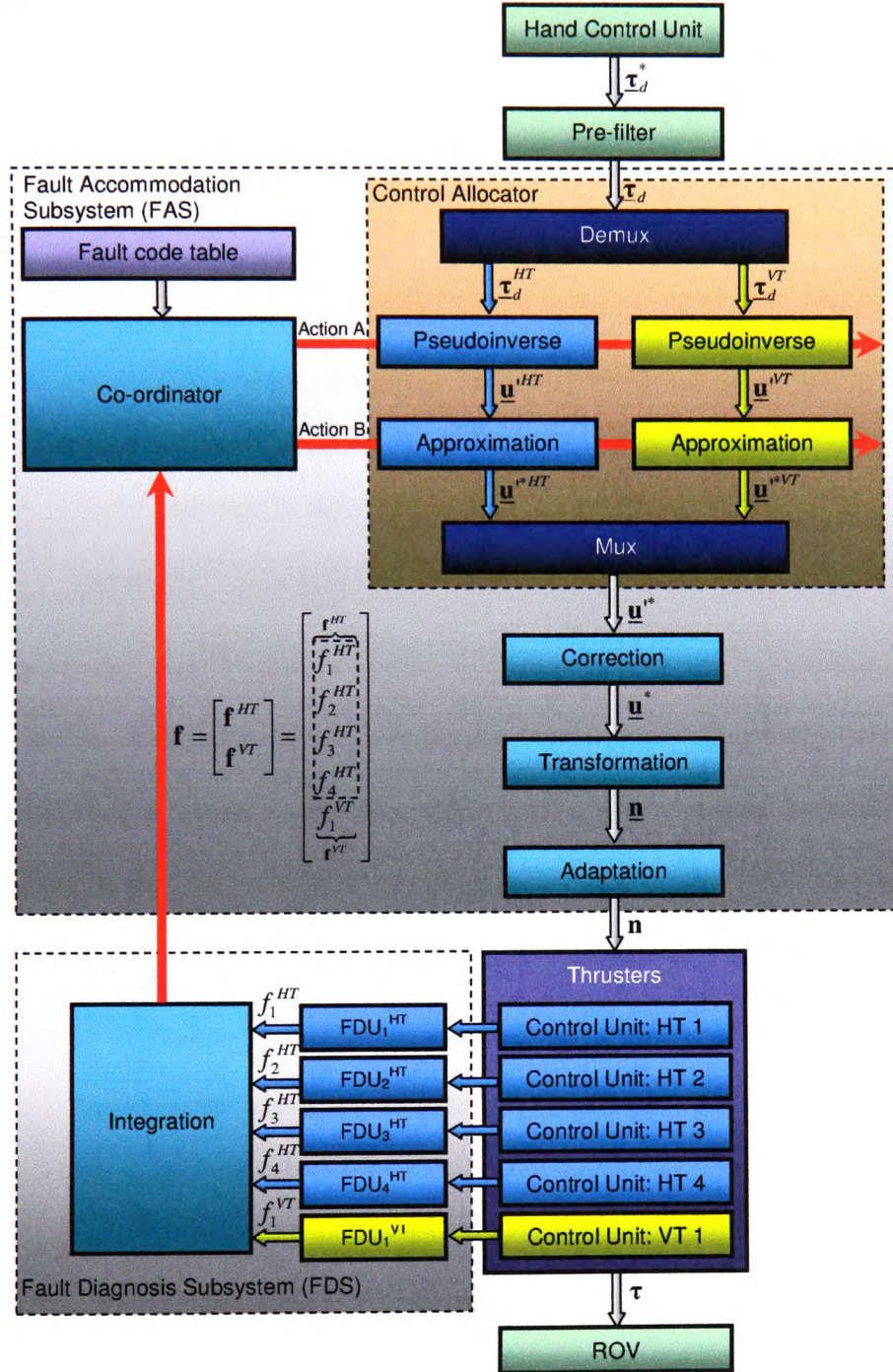


Figure 5.9 Overall functional architecture of the proposed FDAS.

The output of the FDAS is the vector of desired thruster velocities \mathbf{n} , transformed into the form that is compatible (acceptable) by the TCUs. A brief description of the individual components of the overall FDAS architecture is given in the following.

FDS

FDU: The FDS uses FDUs to monitor the state of the thrusters. The FDU is a software module associated with the thruster, able to detect internal faults (for example, temperature of the windings exceeds limits) and external faults (for example, jammed propeller). The output of the FDU is a fault indicator f_i , the code of the fault.

Integration: The fault indicators f_i are integrated into the total fault indicator vector \mathbf{f} inside this block. The vector \mathbf{f} is a carrier of thrusters' states.

FAS

Demux: In accordance with the decomposition of the motion, shown in Table 5.2 and Table 5.3, this block symbolically indicates separation of the vector $\mathbf{\tau}_d$ into two parts: $\mathbf{\tau}_d^{HT}$, representing the DOF (surge, sway and yaw) controllable by horizontal thrusters, and $\mathbf{\tau}_d^{VT}$, representing the DOF (heave) controllable by vertical thruster.

Pseudoinverse: This block finds the weighted pseudoinverse solution of the control allocation problem, separately for horizontal and vertical thrusters. For horizontal thrusters, the solution \mathbf{u}^{HT} is given by (5.46) & (5.48) for FALCON and (5.47) & (5.48) for URIS. For vertical thruster, the solution \mathbf{u}^{VT} is given by (5.50) for FALCON.

Approximation: In contrast to \mathbf{u}^{VT} , which is always feasible, the weighted pseudoinverse solution \mathbf{u}^{HT} can be feasible (satisfies all constraints) or

unfeasible (violates some constraint(s)). The outputs \underline{u}^{*HT} and $\underline{u}^{*VT} = \underline{u}^{VT}$ of this block must be feasible solutions for all time. Hence, if \underline{u}^{HT} is feasible, then $\underline{u}^{*HT} = \underline{u}^{HT}$. Otherwise, fixed-point iterations (FPI) are activated that lead to the feasible solution \underline{u}^{*HT} .

Co-ordinator: The role of this block is to undertake remedial actions in accordance to the context of the total fault indicator vector \mathbf{f} and the instructions, stored in the fault code table. For each possible fault type the fault code table has stored corresponding actions A and B. The action A is related to the weight updates of weighting matrices, used to find the weighted pseudoinverse solution. The Action B is related to changes of constraint bounds, in accordance to fault type.

Mux: This block performs an opposite role to that of the Demux block, i.e. it merges feasible solutions \underline{u}^{*HT} and \underline{u}^{*VT} into composite solution vector \underline{u}^* .

Correction: Vector \underline{u}^* is found using the assumption of a symmetrical relationship between thrust T and the auxiliary control variable u' for a thruster (Figure 5.3 (b)). Since in most real applications this relationship is not symmetrical (Figure 3.12), it is necessary to correct each component of \underline{u}^* in accordance to (3.91) & (3.92). The output of this block is corrected vector \underline{u}^* .

Transformation: The vector \underline{u}^* cannot be directly applied to drive the thrusters. It must be transformed into the vector of desired thruster velocities \underline{n} . This

block performs this function, using transformation (3.97) for each component.

Adaptation: Different TCUs accept data in different format. For example, the desired angular velocity for the TCU of FALCON must be represented as an integer number between -100 and $+100$ (see section 5.1.3). In contrast, the same variable must be converted into the voltage in order to be applied to drive the thruster of URIS. This block transforms the vector \underline{n} into the vector \mathbf{n} , which has the format adapted for the particular TCU.

Vector \mathbf{n} is used to drive the thrusters, which generate a vector of propulsion forces and moments $\boldsymbol{\tau}$. The proposed FDAS guarantees that the condition $\boldsymbol{\tau} = \boldsymbol{\tau}_d$ is satisfied for all attainable $\boldsymbol{\tau}_d$. That is, if $\boldsymbol{\tau}_d$ is attainable, the FDAS will find the exact solution of the control allocation problem, optimal in the l_2 sense. Otherwise, the solution obtained by the FDAS is a very good approximation, which depends on design parameters of the fixed-point iteration method.

5.4 *Fault diagnosis subsystem*

5.4.1 Fault classification

Thrusters are liable to different fault types during the underwater mission e.g. propellers can be jammed or broken, water can penetrate inside the TCU, communication between TCU and the master node can be lost, temperature of the winding can exceed the threshold etc. Some of these faults (partial faults) are not critical and the thruster is able to continue operation in the presence of a fault with the restricted usage, i.e. reduced maximum velocity. In other cases (total faults – failures) the thruster must be switched off and mission has to be continued with remaining operable thrusters. Thruster faults are classified into two main classes:

- *Internal faults* (e.g. temperature of the windings is out of range, lost communication between the TCU and master node, water penetration inside the TCU, drop in bus voltage etc.)
- *External faults* (e.g. jammed or broken propeller).

5.4.2 Fault code table

Relationships between thruster states, fault types and remedial actions are stored in the fault code table (Table 5.8). It must be emphasized, at this point, that this fault code table is just a suggestion, intended to reveal the main ideas of the proposed FDAS. New states (rows) can be added, and the existing relationships can be changed, in order to accommodate specific requirements and available thruster data.

Thruster state	Class	Type	Indicator f_i	Constraint bound s_i
Normal (Fault-free)	-	-	1	1.00
Jammed propeller	Ext.	Partial	2	0.75
Heavy jammed propeller	Ext.	Partial	3	0.50
Broken propeller	Ext.	Total	4	0.00
Unknown fault	Ext.	Partial	5	0.25
Internal fault	Int.	Total	6	0.00

Table 5.8 Fault code table.

5.4.3 Fault detection unit

Description

The FDU is used for monitoring the thruster states and reporting any faulty situation. The FDU is a software module, able to detect internal and external faults. Connections between the FDU and the TCU for arbitrary thruster iTh are indicated in Figure 5.10.

Signals Int_1 , Int_2 , ... for detection of internal faults are already available in existing TCUs for both vehicles. In particular, the TCU for URIS, based on Maxon Servoamplifier

ADS 50/5 (see Appendix A), has a status-reading signal $Int_1 = \text{"Ready"}$, which can be used to report internal faults (i.e. excess temperature or excess current). Similarly, the communication protocol for the FALCON provides monitoring of the winding temperature (Int_1) and bus voltage (Int_2) of each thruster. In order to build a universal FDU, able to detect both internal and external faults, it is necessary to augment the existing internal protection with a software module for fast and reliable detection of external faults.

For detection of external faults available signals are actual velocity of the motor shaft n and current consumption I of the thruster. For URIS, these signals are called "Monitor n " and "Monitor I ", respectively (Appendix A); for FALCON, the communication protocol enables output speed and winding current to be read. By monitoring n and I , together with desired speed n_d obtained as output of the FDAS, the FDU is designed to detect and categorise external thruster faults.

Finally, the universal FDU integrates both parts (internal and external) into one unit, which is able to detect internal and external faults (Figure 5.10). Integration includes a priority scheme, where total faults have higher priority than partial faults. Indicator f_i , the output of the FDU, is the code of the fault.

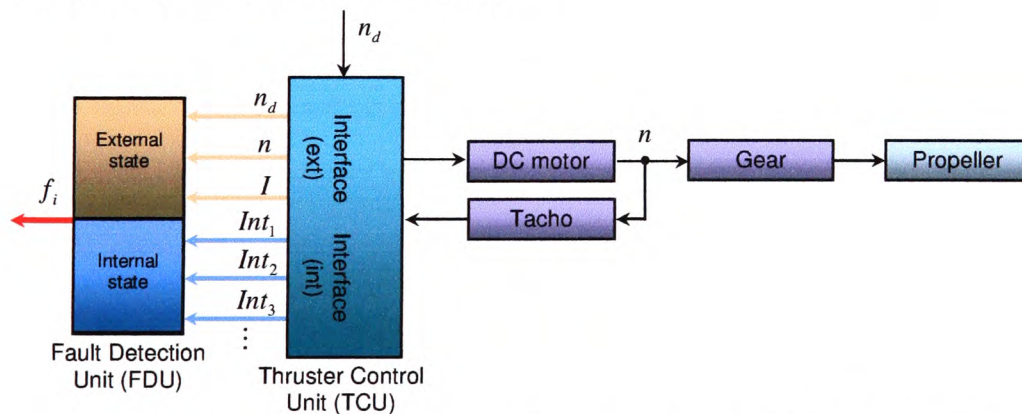


Figure 5.10 Block diagram showing connections between the FDU and the TCU for thruster iTh .

Implementation

Implementation involves two phases: *off-line training* and *on-line fault detection*.

Off-line training phase

The first stage in the training phase is acquisition of training data. Test trials were performed with URIS at University of Girona in July 2002 (Figure 5.11), and training data were saved in files. Normal state and three different fault cases were considered (jammed, heavy jammed and broken propeller)⁸. Jammed propeller was simulated such that an object was attached to the propeller. When the thruster is actuated, the propeller and the object rotate together, representing additional load for the motor. Heavy jammed propeller was simulated with two objects attached. In order to simulate broken propeller, all blades were removed from the shaft. Signals n and I , delivered by the TCU, were acquired by onboard A/D card. Since "Monitor n " and "Monitor I " outputs of the TCU have output voltage range $-10\dots+10\text{ VDC}$ and output resistance $10\text{ k}\Omega$, a signal conditioning circuit⁹ had to be used to convert signals into the form acceptable by the A/D card. The drawback was a small drop in voltage during conversion. Each record in file consists of acquired data from the TCU (n_d , n and I) and associated fault code f_i . Sampling time was 0.1 s , long enough to ensure that all transient responses decay between samples. The motion of URIS was controlled by a joystick, such that all range of possible thruster velocities was covered with enough data points. Figure 5.12 displays raw training data in the 3D space $I - n_d - n$, where each fault type is represented by a different colour.

⁸ However, in real applications the number of faulty cases can be higher. In addition, partially broken or damaged propeller blades can be used to cover intermediate cases.

⁹ The circuit was a voltage follower, with low output resistance and voltage gain slightly less than 1.

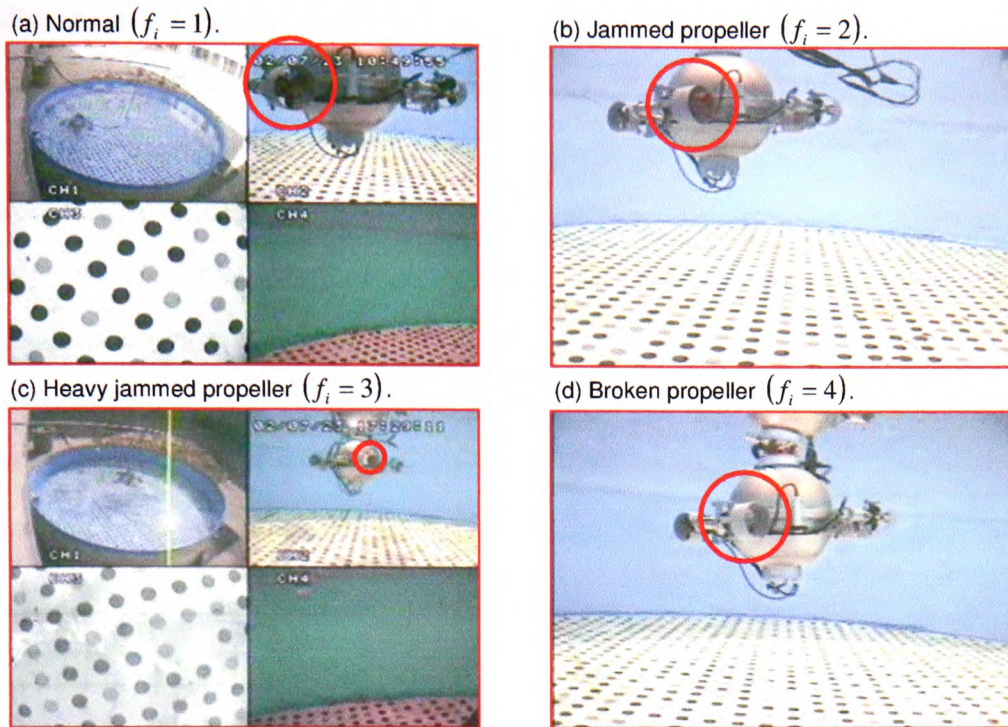


Figure 5.11 The first stage in off-line training: test trials for acquisition of training data.

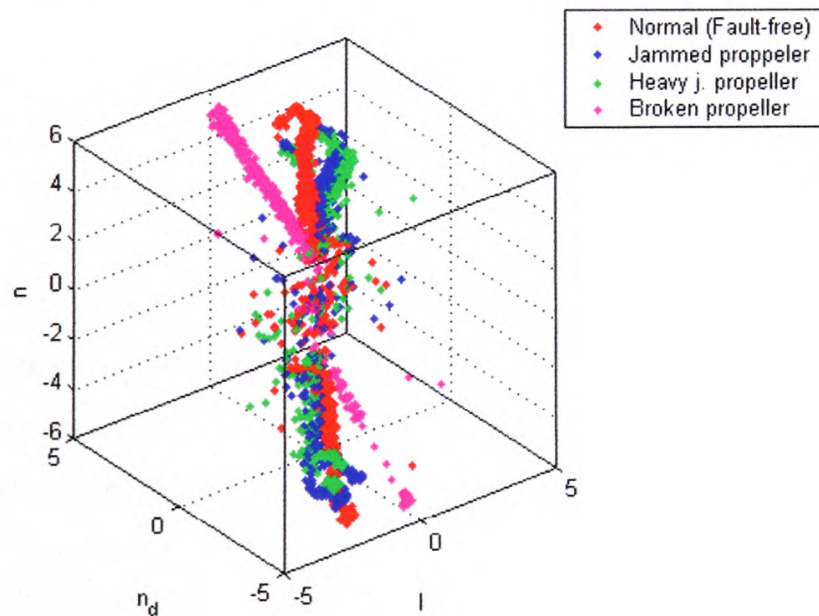


Figure 5.12 Raw training data in the 3D space.

Discretisation levels can be observed in the training data, which are caused by discretisation of the joystick output. It should be noted that the real-time experiments were undertaken during the development stage of URIS, and inadequate signal conditioning, wiring and shielding resulted in noisy data. Nevertheless, design of robust FDU, able to cope with poor data quality, was a real challenge.

The time diagrams of the raw training data are shown in Figure 5.13. Signals from different fault types are connected next to each other in order to make easier their comparison. Noise and *outliers* (data items that lie very far from the main body of the data) are particularly noticeable in the motor current waveform.

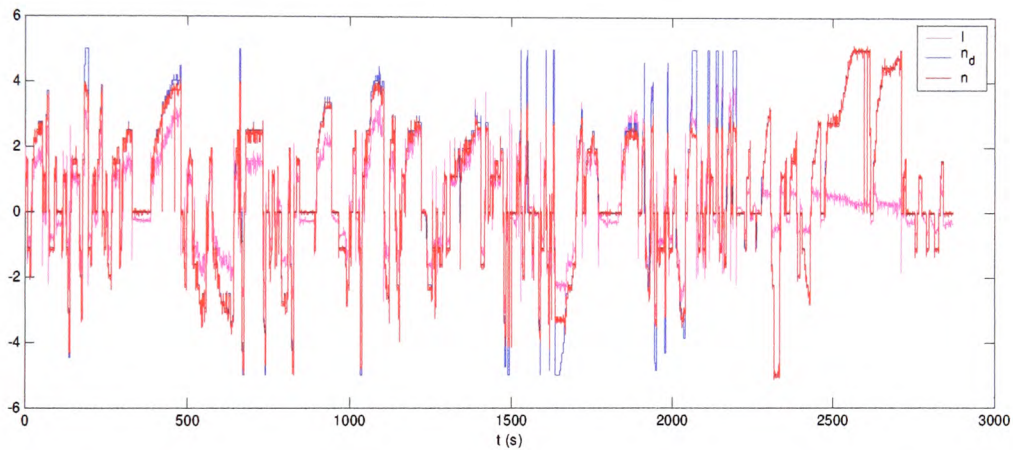


Figure 5.13 Time diagrams of raw training data.

Data pre-processing filters the raw training data in order to remove outliers and reject noise. Pre-processed training data in 3D space are shown in Figure 5.14¹⁰. Figure 5.15 displays time diagrams of pre-processed data. Normalisation step in pre-processing is optional, since all variables have the same range and no one is dominant.

¹⁰ MATLAB function `medfilt1` was used to filter data and to remove outliers.

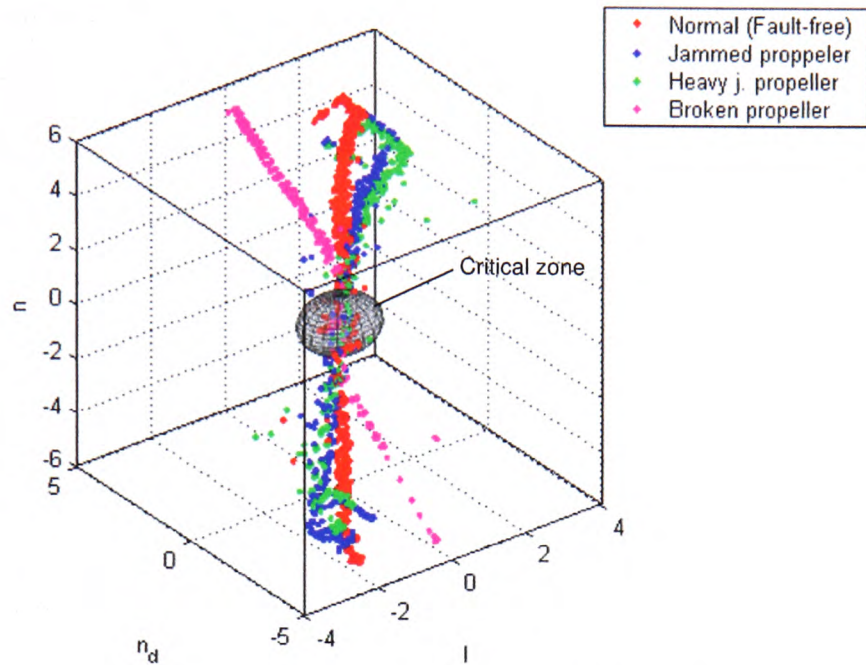


Figure 5.14 Pre-processed training data in the 3D space.

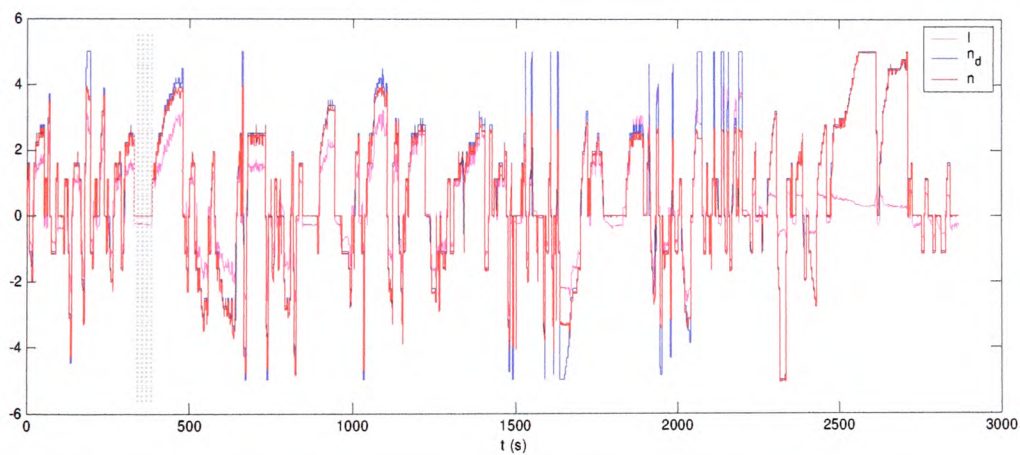


Figure 5.15 Time diagrams of pre-processed training data. One of the zero-velocity segments is highlighted for illustration. These segments are excluded from the training process.

Analysing the distribution of the training data in Figure 5.14, the *first* feature that can be noticed is that each fault type creates certain pattern. In the ideal case, these patterns should be well-defined curves. However, the presence of the noise and outliers in the

training data results that patterns shown in Figure 5.14 exhibit a fuzzy ("cloudy") look. The *second* feature is that the zone around $n_d \approx 0$ (called the *critical zone*) is filled with data from different fault types in such a way that it is very hard to distinguish individual fault types. Geometrically, the critical zone represents intersection of different fault type patterns. This makes successful fault detection and isolation in the critical zone difficult to achieve. In particular, for zero-velocity case $n_d = 0$ the thruster does not rotate and successful and reliable fault detection is impossible, since external faults cannot be detected without shaft rotation. In other words, thruster must operate (rotate) in order to detect external fault. The solution for this problem is the exclusion of the zero-velocity segments from the training process, i.e. the training is performed considering data records with $n_d \neq 0$, as shown in Figure 5.15. However, the same exclusion is performed during the on-line fault detection phase (see page 5-46).

Time diagrams of the pre-processed training data are shown in Figure 5.16, together with associated fault codes. Each fault type is characterised by specific features, which makes them different from the other types. These features are discussed in the following.

In general, all three variables (I , n_d and n) are correlated, i.e. they tend to rise and fall together in a non-linear way. The degree of nonlinearity depends on n_d and motor load.

For normal (fault-free) thruster state, n is close to n_d for approximately $|n_d| \leq 4V$. Disagreement between n and n_d becomes higher with increasing $|n_d|$. Eventually, for cases when $|n_d| \rightarrow 5V$, n is not able to follow n_d and $|n|$ is saturated to approximately $4V$. This means that the thruster is not able to utilise the full operating range even in a fault-free case, under standard load conditions. It is believed that this performance can be improved by re-tuning the controller parameters inside the TCU. For jammed propeller, an object attached to the blades generates an additional load for the motor and leads to

higher current I than for the fault-free case. In addition, the saturation of n is reached earlier, i.e. for smaller n_d . A similar conclusion can be drawn for the heavy jammed propeller, where two objects attached to the blades result in an even higher load for the motor. Saturation is achieved for smaller n_d than in previous cases and current I is much higher. Finally, in the case of broken propeller, the absence of the blades means that the load for the motor is much smaller than in the other cases and n is able to follow n_d over all operating range, with reduced current I . However, the thruster does not generate any propulsion force in this case.

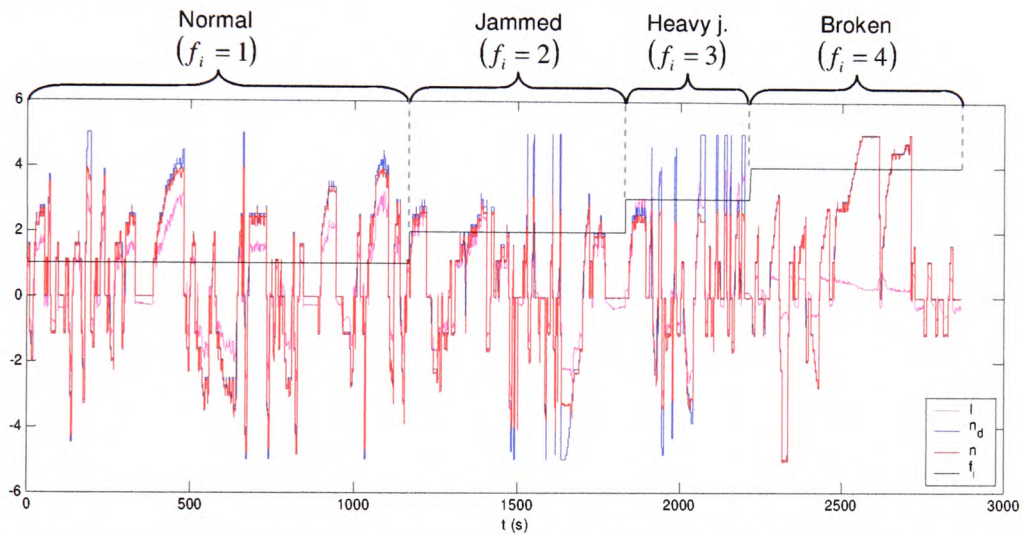


Figure 5.16 Time diagrams of pre-processed training data, together with fault code.

Since the thruster fault detection problem is time-dependent, the initial idea for the second stage in the training phase was to use trajectories of thruster states on the map grid of the single, large SOM for all fault types. The SOM was trained by pre-processed training data using the MATLAB SOM Toolbox (<http://www.cis.hut.fi/projects/somtoolbox/>). Clusters of the map, represented as a collection of map units with the same label (fault type), correspond to different fault types. Different training parameters (e.g. map shape, map size, neighbourhood function etc.) and learning methods (supervised and unsupervised:

sequential and batch) were combined. Figure 5.17 displays typical U -matrices obtained during training. In particular, Figure 5.17 (a) displays a U -matrix obtained using unsupervised learning (batch algorithm), while the same matrix for supervised learning is shown in Figure 5.17 (b). The U -matrix uses colour code to visualise the relative distance between adjacent map units on the whole map. The clusters are highlighted by plotting appropriate labels on the map. Labels N , A , B and C are used for fault codes 1, 2, 3 and 4, respectively. The input to the map was a feature vector \mathbf{x} , created from the pre-processed n_d , n and I . Trajectories (paths created on the map grid by the BMUs of consecutive input vectors) were used for fault detection purpose.

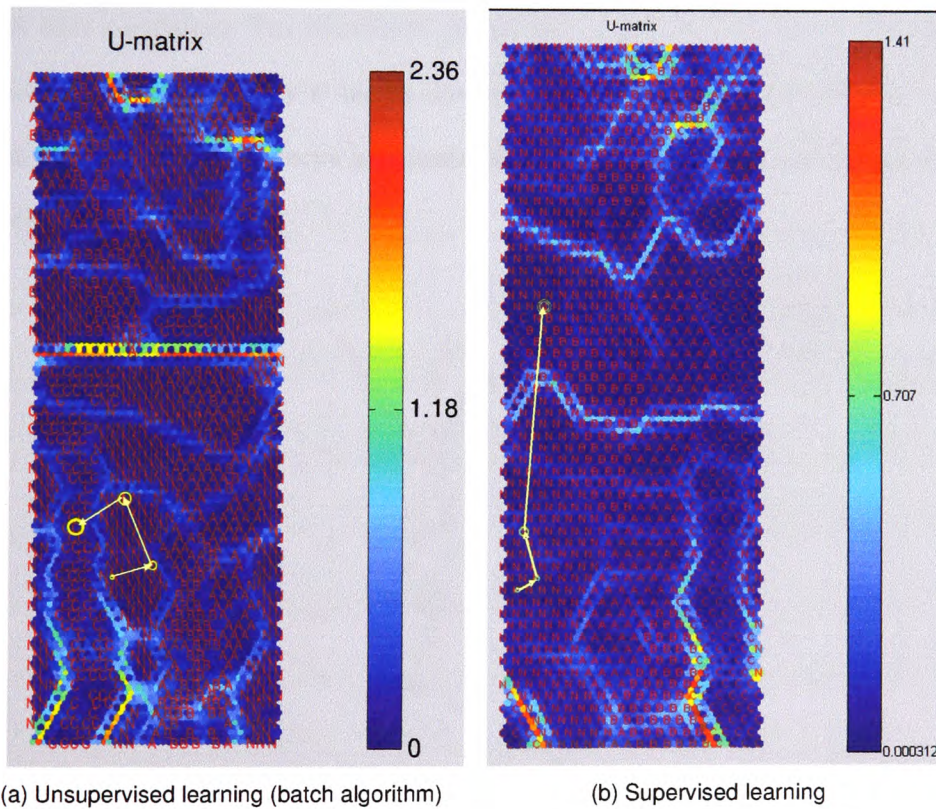


Figure 5.17 SOM trajectories were initially used for fault detection purpose.

Unfortunately, each attempt resulted in the FDU producing in certain cases wrong fault detection when thruster operates in the critical zone. Although the large number of detection errors in the initial design stage was later significantly reduced by fine tuning of the training parameters, it was not possible to eliminate these errors. The initial idea about a single, large SOM for all fault types was abandoned and replaced with representation of each fault type by a single, one-dimensional SOM. Hence, the main idea of the second stage in the training phase is to replace each fault type (pattern) in Figure 5.14 with a SOM prototype as shown in Figure 5.18, which serves as a representative of the particular fault type. A fault type with code $f_i = k$ is replaced with SOM k . Each SOM k is one-dimensional array of 100 neurons. Each of these neurons has associated prototype vector with three coordinates. The distribution of prototype vectors (Figure 5.18) in the input space was found using fuzzy C -means clustering and approximately 80% data from each fault type. Each prototype vector is a cluster centre and the representative of all data from its cluster.

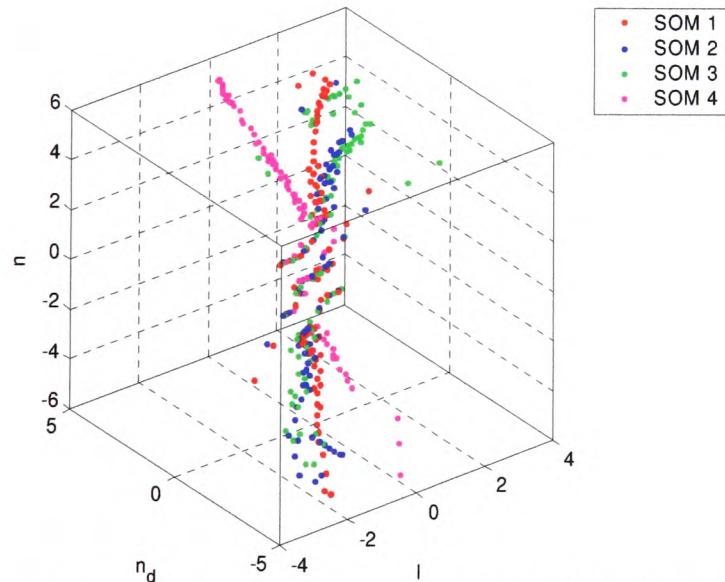


Figure 5.18 The second stage in off-line training phase: different fault types (patterns) are replaced by SOM prototypes.

In the third and the last stage in the training phase, the structure of the SOM prototypes is saved on hard disk for future use. In this way, heavy demanding calculations are performed off-line, during the training phase, which enables fast and efficient fault detection during the on-line phase.

On-line fault detection phase

From the preceding discussion, the problem of thruster fault detection can therefore be interpreted as a pattern recognition problem. An original method for on-line fault detection, adapted to the specific features of the underlying pattern recognition problem, is described in the following.

During the initialisation stage of the on-line fault detection phase, the master node (the main processor) reads the structure of the SOM prototypes, saved on hard disk during the training phase, and stores this data in the working memory for fast access.

After the initialisation is finished, the fault detection is performed by repeating the following steps at each program cycle:

Algorithm 5.1 (FDU - On-line fault detection)

1. Read inputs for internal faults (Int_1, Int_2, \dots) and external faults (I, n_d and n).
2. If any of Int_k is on, set $f_i = 6$ (Internal fault) and go to 13.
3. Perform pre-processing on I, n_d and n .
4. Create feature vector \mathbf{x} .
5. Find q closest prototype vectors (BMUs) to \mathbf{x} in each SOM.
6. Create matrix $\mathbf{M}_{4 \times q}$ with distances between \mathbf{x} and q BMUs in each SOM.
7. Find minimal values and corresponding indices for each column of \mathbf{M} .
8. Store minimum values in row vector \mathbf{m} .

9. If characteristic property of \mathbf{m} exceeds its limit, set $f_i = 5$ (Unknown fault) and go to 13.
10. Store indices in row vector b .
11. Add vector b to the buffer $\mathbf{B}_{s \times q}$.
12. Analyse buffer elements and find the fault code f_i .
13. Deliver f_i as the output of the FDU.

At step 1, the algorithm reads actual values of inputs Int_1, Int_2, \dots for detection of internal faults and I , n_d and n for detection of external faults. It is assumed that digital signals Int_k are active (on) in the case of corresponding internal faults. The logical function $h = \text{OR}_k Int_k$ is evaluated in step 2. If h is equal to "1", then the thruster exhibits internal fault and the output of the FDU is set to $f_i = 6$ (code for Internal fault, as defined in the fault code table shown in Table 5.8). The sequential execution of the algorithm is terminated and redirected to the end (step 13).

Otherwise, if no Int_k is on, pre-processing of I , n_d and n is performed in step 3, as discussed on page 5-40.

Numerical values of pre-processed I , n_d and n are chosen to create a *feature vector* \mathbf{x} in step 4, where $\mathbf{x} = [I \quad n_d \quad n]^T$ is the three-dimensional column vector. In this way, the actual measurement is represented as a point in the 3D space $I - n_d - n$, which can be considered as a *feature space*.

In step 5, the q closest prototype vectors (BMUs) from each map to the feature vector \mathbf{x} are found¹¹, together with corresponding distances. Figure 5.19 illustrates the situation for $q = 3$. Nomenclature ${}^k\mathbf{BMU}_j$ means j^{th} BMU in SOM k , while ${}^k d_j$ means Euclidian distance between \mathbf{x} and ${}^k\mathbf{BMU}_j$, $k = \overline{1,4}$, $j = \overline{1,3}$. It can be seen from definition that ${}^k d_1 \leq {}^k d_2 \leq {}^k d_3$, $k = \overline{1,4}$.

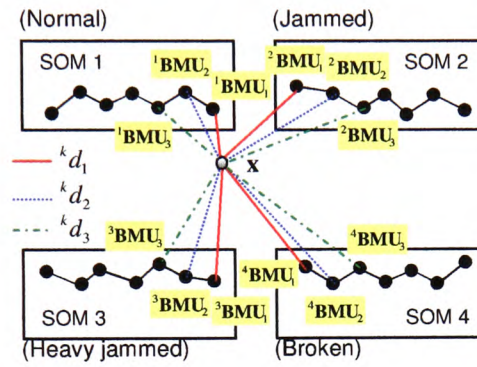


Figure 5.19 On-line fault detection phase: position of the feature vector is determined relative to SOM prototypes by finding $q = 3$ closest BMUs in each SOM.

In step 6 the matrix $\mathbf{M} = [{}^k d_j]_{4 \times 3}$ is created. In order to make the following discussion more understandable assume that

$$\mathbf{M} = \begin{bmatrix} {}^1 d_1 & {}^1 d_2 & {}^1 d_3 \\ {}^2 d_1 & {}^2 d_2 & {}^2 d_3 \\ {}^3 d_1 & {}^3 d_2 & {}^3 d_3 \\ {}^4 d_1 & {}^4 d_2 & {}^4 d_3 \end{bmatrix} = \begin{bmatrix} 0.2 & 0.7 & 0.9 \\ 0.4 & 0.5 & 0.6 \\ 0.3 & 0.4 & 0.7 \\ 0.5 & 0.6 & 0.8 \end{bmatrix} \quad (5.40)$$

Minimum values $\min_k {}^k d_j$ of each column of \mathbf{M} and corresponding indices (values of k that correspond to minimum values) are found in step 7:

¹¹ Function som_bmus from SOM Toolbox (Appendix A, page A-15) was used.

$$\mathbf{M} = \begin{bmatrix} 0.2 & 0.7 & 0.9 \\ 0.4 & 0.5 & 0.6 \\ 0.3 & 0.4 & 0.7 \\ \underline{0.5} & \underline{0.6} & \underline{0.8} \end{bmatrix} \quad (5.41)$$

minimums \longrightarrow $\begin{matrix} 0.2 & 0.4 & 0.6 \\ (1) & (3) & (2) \end{matrix}$
 indices \longrightarrow

Minimum values are stored in row vector \mathbf{m} in step 8:

$$\mathbf{m} = [0.2 \quad 0.4 \quad 0.6] \quad (5.42)$$

In order to determine if \mathbf{x} is too far from all maps, some property of \mathbf{m} must be evaluated and compared with standard thresholds (step 9). If the property is out of range it means that the thruster state is unknown (not experienced before). In order to avoid wrong detection, past values of \mathbf{m} could also be included in the decision process. Another option is to utilise fuzzy inference system to fuzzify thresholds and to improve robustness. One of the simplest properties is average (mean) value \bar{m} . If $\bar{m} > \text{Threshold}$, then the output of the FDU is set to $f_i = 5$ (code for Unknown fault in the fault code table). The sequential execution of the algorithm is terminated and redirected to the end (step 13).

Otherwise, indices of minimum values of each column are stored in row vector \mathbf{b} in step 10. For example, $\mathbf{b} = [1 \quad 3 \quad 2]$ means that the closest first BMU is in SOM 1, second – in SOM 3 and third – in SOM 2.

In order to avoid false detection, the final decision about thruster state is accomplished using present vector $\mathbf{b}(t)$ and past vectors $\mathbf{b}(t-1)$, $\mathbf{b}(t-2)$, ..., $\mathbf{b}(t-(s-1))$, which are stored in the buffer $\mathbf{B}_{s \times q}$. This buffer operates similar to shift register: when the new vector is added to the buffer (step 11), the other vectors are pushed down and the "oldest" vector is shifted out (Figure 5.20).

Elements of the buffer are compared to each other in step 12. If all buffer elements have the same value, then the fault indicator f_i is set to this value. Otherwise, the previous value of f_i is kept.

Finally, the fault indicator f_i is delivered as the output of the FDU in step 13.

Although at first sight the Algorithm 5.1 might seem relatively complicated, its MATLAB implementation is very efficient, due to advanced MATLAB programming capabilities.

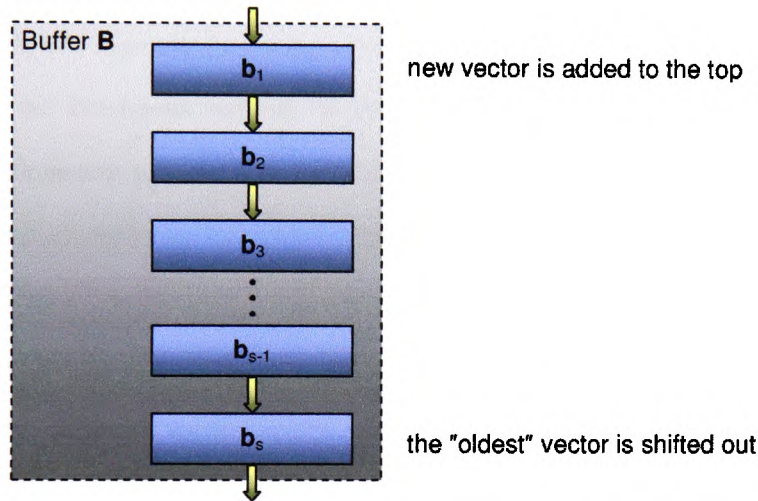


Figure 5.20 Buffer $B_{s \times q}$ with present and past values of vector b .

5.4.4 Integration of fault indicators

Integration of individual fault indicators is performed by the "Integration" block (Figure 5.9). The composite vector

$$\mathbf{f} = \left[\underbrace{f_1^{HT} \quad f_2^{HT} \quad f_3^{HT} \quad f_4^{HT}}_{\mathbf{f}^{HT}} \quad \underbrace{f_1^{VT}}_{\mathbf{f}^{VT}} \right]^T \quad (5.43)$$

is called a total fault indicator vector. This vector carries the fault state codes for each thruster.

5.5 *Fault accommodation subsystem*

5.5.1 Description

As stated previously, the ROV pilot uses the HCU (Figure 5.9) to generate the input command vector τ_d . The FDS finds the total fault indicator vector \mathbf{f} with information about the state of each thruster. The FAS uses these two vectors and relationships in the fault code table to solve the control allocation problem separately for motion in the horizontal and vertical plane.

The hybrid approach for control allocation, based on the integration of the pseudoinverse and the fixed-point method, is implemented as a two-step process. The weighted pseudoinverse solution is found in the first step. Then the feasibility of the solution is examined analysing individual components of the solution. If violation of actuator constraint(s) is detected, the fixed-point iteration method is activated in the second step, which results in guaranteed feasible solution. In this way the hybrid approach is able to allocate the exact solution, optimal in the l_2 sense, inside the entire attainable command set. This solution minimises the control energy cost function, which is the most suitable criteria for underwater applications.

5.5.2 Hybrid approach for control allocation

Pseudoinverse

The solution method adopted in the FAS relies on the fact that an explicit solution to the unconstrained control allocation problem:

$$\begin{aligned} \min_{\mathbf{u}} \|\mathbf{W}\mathbf{u}\|_2 \\ \text{subject to } \mathbf{B}\mathbf{u} = \tau_d \end{aligned}$$

is given by (Lema B.2, Appendix B)

$$\mathbf{u} = \mathbf{B}_w^+ \boldsymbol{\tau}_d \quad (5.44)$$

where the matrix \mathbf{B}_w^+

$$\mathbf{B}_w^+ = \mathbf{W}^{-1} (\mathbf{B} \mathbf{W}^{-1})^+ = \mathbf{W}^{-1} \mathbf{B}^T (\mathbf{B} \mathbf{W}^{-1} \mathbf{B}^T)^{-1} \quad (5.45)$$

is the weighted pseudoinverse of \mathbf{B} .

Horizontal plane

For motion in the horizontal plane the weighted pseudoinverse matrix $\underline{\mathbf{B}}_{w_u}^{+HT}$ is given by

$$\text{FALCON: } \underline{\mathbf{B}}_{w_u}^{+HT} = \frac{1}{\sum_{i=1}^4 w_i^{HT}} \begin{bmatrix} 2(w_3^{HT} + w_4^{HT}) & 2(w_2^{HT} + w_3^{HT}) & 2(w_2^{HT} + w_4^{HT}) \\ 2(w_3^{HT} + w_4^{HT}) & -2(w_1^{HT} + w_4^{HT}) & -2(w_1^{HT} + w_3^{HT}) \\ 2(w_1^{HT} + w_2^{HT}) & 2(w_1^{HT} + w_4^{HT}) & -2(w_2^{HT} + w_4^{HT}) \\ 2(w_1^{HT} + w_2^{HT}) & -2(w_2^{HT} + w_3^{HT}) & 2(w_1^{HT} + w_3^{HT}) \end{bmatrix} \quad (5.46)$$

$$\text{URIS: } \underline{\mathbf{B}}_{w_u}^{+HT} = \frac{1}{\sum_{i=1}^4 w_i^{HT}} \begin{bmatrix} (2w_2^{HT} + w_3^{HT} + w_4^{HT}) & (w_3^{HT} - w_4^{HT}) & 2(w_3^{HT} + w_4^{HT}) \\ (2w_1^{HT} + w_3^{HT} + w_4^{HT}) & (w_4^{HT} - w_3^{HT}) & -2(w_3^{HT} + w_4^{HT}) \\ (w_1^{HT} - w_2^{HT}) & (w_1^{HT} + w_2^{HT} + 2w_4^{HT}) & 2(w_1^{HT} + w_2^{HT}) \\ (w_2^{HT} - w_1^{HT}) & (w_1^{HT} + w_2^{HT} + 2w_3^{HT}) & -2(w_1^{HT} + w_2^{HT}) \end{bmatrix} \quad (5.47)$$

These expressions are obtained by combining (5.22), (5.23) and (5.45).

The weighted pseudoinverse solution of the unconstrained control problem for motion in the horizontal plane is given by

$$\underline{\mathbf{u}}^{HT} = \underline{\mathbf{B}}_{w_u}^{+HT} \boldsymbol{\tau}_d^{HT} \quad (5.48)$$

Vertical plane

Combining (5.37), (5.38) and (5.45), the weighted pseudoinverse matrix $\underline{\mathbf{B}}_{w_v}^{+VT}$ for motion in the vertical plane becomes scalar

$$\text{FALCON: } \underline{\mathbf{B}}_{w_v}^{+VT} = 1 \quad (5.49)$$

The weighted pseudoinverse solution of the unconstrained control problem for motion in the vertical plane is given by

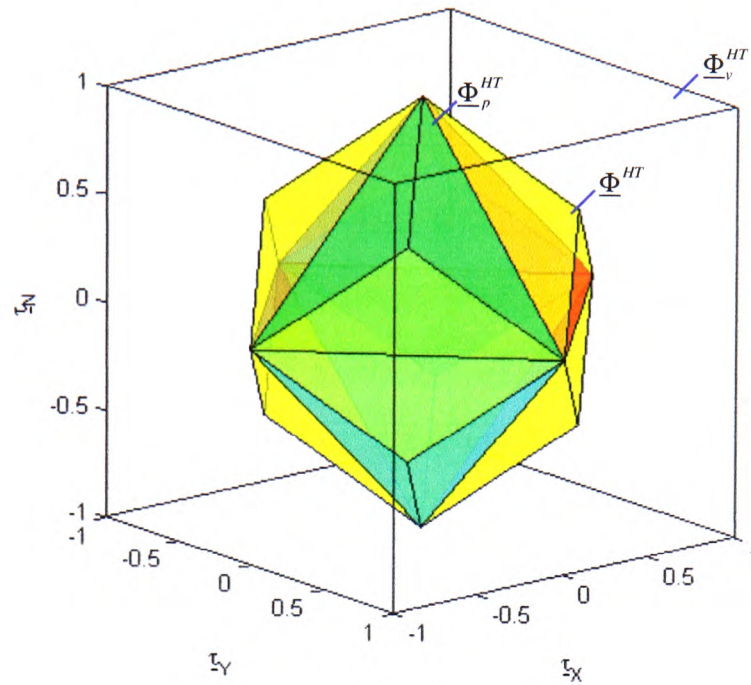
$$\underline{\mathbf{u}}^{VT} = \underline{\mathbf{B}}_{w_v}^{+VT} \boldsymbol{\tau}_d^{VT} = \boldsymbol{\tau}_d^{VT} \quad (5.50)$$

Feasibility of pseudoinverse solution

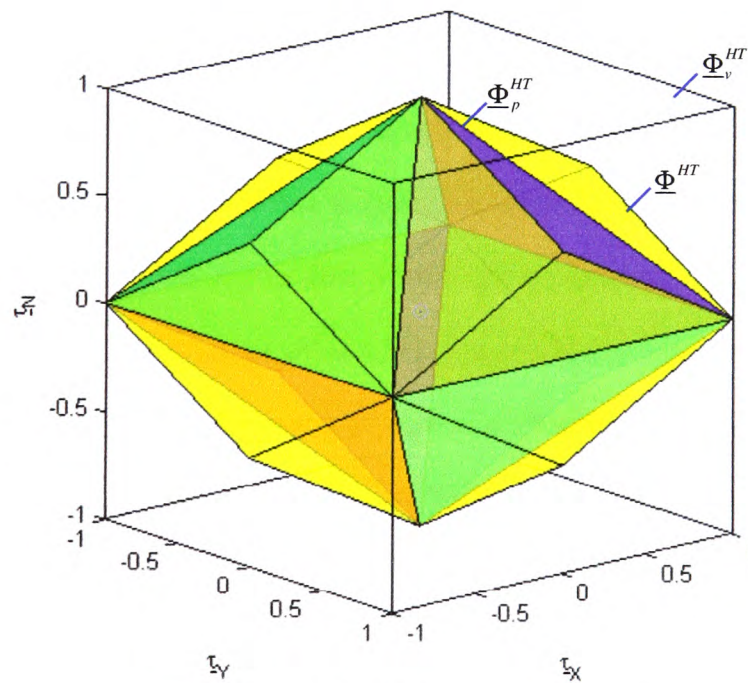
The weighted pseudoinverse solution for motion in the vertical plane (5.50) is always feasible. However, it is not always the case with the solution (5.48) for motion in the horizontal plane. The input command vector $\underline{\tau}_d^{HT} = [\tau_x \ \tau_y \ \tau_N]^T$ lies in the virtual control space $\underline{\Phi}_v^{HT}$, that is, the unit cube in \mathcal{R}^3 :

$$\underline{\Phi}_v^{HT} = \left\{ \underline{\tau}^{HT} \in \mathcal{R}^3 \mid \|\underline{\tau}^{HT}\|_\infty \leq 1 \right\} \subset \mathcal{R}^3 \quad (5.51)$$

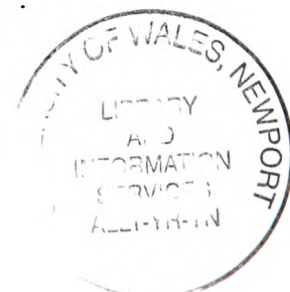
For the constrained control allocation problem, where the constraint $\underline{u}^{HT} \in \underline{\Omega}^{HT}$ is required to be satisfied, the solution (5.48) may become unfeasible, depending on the position of $\underline{\tau}_d^{HT}$ inside $\underline{\Phi}_v^{HT}$. The virtual control space $\underline{\Phi}_v^{HT}$ can be partitioned into characteristic regions, as indicated in Figure 5.21. The two characteristic regions inside $\underline{\Phi}_v^{HT}$ are $\underline{\Phi}_p^{HT}$ (the feasible region for pseudoinverse) and $\underline{\Phi}^{HT} \supset \underline{\Phi}_p^{HT}$ (the attainable command set). The shape of $\underline{\Phi}^{HT}$ is already given for the fault-free case in Figure 5.7 for FALCON and Figure 5.8 for URIS. The shape of $\underline{\Phi}_p^{HT}$ is found from the condition $\underline{u}^{HT} = \underline{B}_v^{+HT} \underline{\tau}_d^{HT} \in \underline{\Omega}^{HT}$ (see page 5-57). It should be emphasized that, for the general constrained control allocation problem, there is an infinite number of exact solutions for $\underline{\tau}_d^{HT} \in \underline{\Phi}_p^{HT}$, while no exact solution exists for $\underline{\tau}_d^{HT} \in \underline{\Phi}_v^{HT} \setminus \underline{\Phi}_p^{HT}$. The weighted pseudoinverse is able to find the exact feasible solution of the control allocation problem, optimal in the l_2 sense, only if $\underline{\tau}_d^{HT} \in \underline{\Phi}_p^{HT}$. Otherwise, for $\underline{\tau}_d^{HT} \in \underline{\Phi}_v^{HT} \setminus \underline{\Phi}_p^{HT}$, the solution obtained by pseudoinverse is unfeasible. However, as demonstrated in Example 4.10, the fixed-point iteration method is able to find the exact solution, optimal in the l_2 sense, for cases $\underline{\tau}_d^{HT} \in \underline{\Phi}^{HT} \setminus \underline{\Phi}_p^{HT}$. This idea is explored further in the following section.



(a) FALCON.



(b) URIS.

Figure 5.21 Partitions of the virtual control space Φ_v^{HT} .

Fixed-point iteration method

In the case when the pseudoinverse solution $\underline{u}^{HT} = \underline{B}_{W_\mu}^{*HT} \underline{\tau}_d^{HT}$ is unfeasible, the fixed-point iteration method is triggered, which is able to find the exact solution for $\underline{\tau}_d^{HT} \in \underline{\Phi}^{HT} \setminus \underline{\Phi}_\rho^{HT}$ or approximate solution for $\underline{\tau}_d^{HT} \in \underline{\Phi}_v^{HT} \setminus \underline{\Phi}^{HT}$. Three choices are available as the initial point for iterations: T -approximation \underline{u}_t^{*HT} or S -approximation \underline{u}_s^{*HT} of the unfeasible pseudoinverse solution \underline{u}^{HT} , or the output at previous time sample $\underline{u}^{*HT}(t-T)$, where T is sampling time. As stated in Example 4.10, design parameters of the fixed-point iteration method are \underline{W}_μ , \underline{W}_v , ε and tol .

Co-ordinator

In accordance with previous discussions, the actions undertaken by "Co-ordinator" are summarised in Table 5.9 - Table 5.13 for different faulty situations. In particular, Table 5.9 displays actions for fault-free case. Weights and constraint bounds are all set to unity. Actions undertaken in the case of a partial fault in 1HT with code f_k , $k \in \{2,3,5\}$ (see the fault code table on page 5-36) are shown in Table 5.10. The constraint bound $0 < s_k < 1$ is associated with the fault code f_k . The faulty thruster 1HT is penalised by increasing w_1^{HT} and restricting the constraint bound s_1^{HT} to $s_1^{HT} = s_k$. In the case of a total fault in 1HT ($k \in \{4,6\}$), the faulty thruster is switched off ($w_1^{HT} \rightarrow \infty, s_1^{HT} = 0$) and excluded from the allocation process, as indicated in Table 5.11. The control allocation is solved with remaining operable thrusters. Table 5.12 shows actions in the case of a partial fault in 1VT with code f_k . In contrast to similar case shown in Table 5.10, the constraint bound s_1^{VT} is restricted to $s_1^{VT} = s_k$, but the weight w_1^{VT} is not updated, since the pseudoinverse solution (5.50) does not depend on this weight.

Action	Scope	Fault-free case
A	"Pseudoinverse"	$w_i^{HT} = 1, i = \overline{1,4}$ $w_1^{VT} = 1$
B	"Approximation"	$s_i^{HT} = 1, i = \overline{1,4}$ $s_1^{VT} = 1$

Table 5.9 "Co-ordinator" actions for fault-free case.

Action	Scope	Partial fault (f_k) in 1HT
A	"Pseudoinverse"	$w_i^{HT} = \begin{cases} 1 + 2 \cdot \left(\frac{1}{s_k} - 1 \right), & i = 1 \\ 1, & i = 2,3,4 \end{cases}$ $w_1^{VT} = 1$
B	"Approximation"	$s_i^{HT} = \begin{cases} s_k, & i = 1 \\ 1, & i = 2,3,4 \end{cases}$ $w_1^{VT} = 1$

Table 5.10 "Co-ordinator" actions for a partial fault in 1HT .

Action	Scope	Total fault in 1HT
A	"Pseudoinverse"	$w_i^{HT} = \begin{cases} \infty, & i = 1 \\ 1, & i = 2,3,4 \end{cases}$ $w_1^{VT} = 1$
B	"Approximation"	$s_i^{HT} = \begin{cases} 0, & i = 1 \\ 1, & i = 2,3,4 \end{cases}$ $w_1^{VT} = 1$

Table 5.11 "Co-ordinator" actions for a total fault in 1HT .

Action	Scope	Partial fault (f_k) in 1VT
A	"Pseudoinverse"	$w_i^{HT} = 1, i = \overline{1,4}$ $w_1^{VT} = 1$
B	"Approximation"	$s_i^{HT} = 1, i = \overline{1,4}$ $s_1^{VT} = s_k$

Table 5.12 "Co-ordinator" actions for a partial fault in 1VT .

Action	Scope	Total fault in 1VT
A	"Pseudoinverse"	$w_i^{HT} = 1, i = \overline{1,4}$ $w_1^{VT} = 1$
B	"Approximation"	$s_i^{HT} = 1, i = \overline{1,4}$ $s_1^{VT} = 0$

Table 5.13 "Co-ordinator" actions for a total fault in 1VT .

Table 5.13 displays actions in the case of a total fault in 1VT . The faulty thruster is switched off as before, but in this case the heave DOF becomes uncontrollable, since the remaining operable horizontal thrusters cannot produce force in z direction.

Feasible region for pseudoinverse

The problem of finding the shape of $\underline{\Phi}_p^{HT}$ for different thruster configurations is addressed in this section. In the general case $\underline{\mathbf{B}}_{\mathbf{w}_u}^{+HT}$ can be partitioned as

$$\underline{\mathbf{B}}_{\mathbf{w}_u}^{+HT} = \begin{bmatrix} \mathbf{N}_1^T \\ \mathbf{N}_2^T \\ \mathbf{N}_3^T \\ \mathbf{N}_4^T \end{bmatrix} \quad (5.52)$$

Equation $\underline{u}^{HT} = \underline{B}_{w_d^{HT}}^{+HT} \underline{\tau}^{HT}$ can be rewritten as

$$\underbrace{\begin{bmatrix} \underline{u}_1^{HT} \\ \underline{u}_2^{HT} \\ \underline{u}_3^{HT} \\ \underline{u}_4^{HT} \end{bmatrix}}_{\underline{u}^{HT}} = \underbrace{\begin{bmatrix} \underline{N}_1^T \\ \underline{N}_2^T \\ \underline{N}_3^T \\ \underline{N}_4^T \end{bmatrix}}_{\underline{B}_{w_d^{HT}}^{+HT}} \underline{\tau}^{HT} \quad (5.53)$$

or in component form

$$\begin{aligned} \pi_1 : \quad \underline{N}_1^T \cdot \underline{\tau}^{HT} &= \underline{u}_1^{HT} \\ \pi_2 : \quad \underline{N}_2^T \cdot \underline{\tau}^{HT} &= \underline{u}_2^{HT} \\ \pi_3 : \quad \underline{N}_3^T \cdot \underline{\tau}^{HT} &= \underline{u}_3^{HT} \\ \pi_4 : \quad \underline{N}_4^T \cdot \underline{\tau}^{HT} &= \underline{u}_4^{HT} \end{aligned} \quad (5.54)$$

Each equation in (5.54) represents a plane π_i in \mathcal{R}^3 , and $\underline{N}_i \perp \pi_i$ are normal vectors, orthogonal to planes π_i , $i = \overline{1,4}$.

Condition $\underline{u}^{HT} \in \underline{\Omega}^{HT}$ can be rewritten as

$$\begin{aligned} -s_1^{HT} \leq \underline{u}_1^{HT} \leq s_1^{HT} & \quad -s_1^{HT} \leq \underline{N}_1^T \cdot \underline{\tau}^{HT} \leq s_1^{HT} \\ -s_2^{HT} \leq \underline{u}_2^{HT} \leq s_2^{HT} & \quad -s_2^{HT} \leq \underline{N}_2^T \cdot \underline{\tau}^{HT} \leq s_2^{HT} \\ -s_3^{HT} \leq \underline{u}_3^{HT} \leq s_3^{HT} & \quad -s_3^{HT} \leq \underline{N}_3^T \cdot \underline{\tau}^{HT} \leq s_3^{HT} \\ -s_4^{HT} \leq \underline{u}_4^{HT} \leq s_4^{HT} & \quad -s_4^{HT} \leq \underline{N}_4^T \cdot \underline{\tau}^{HT} \leq s_4^{HT} \end{aligned} \quad (5.55)$$

A set of equations (5.55) determine the shape of a convex body $\underline{\Phi}_p^{HT}$. The boundary

$\partial(\underline{\Phi}_p^{HT})$ represents a set of all $\underline{\tau}^{HT}$ for which at least one component of the pseudoinverse solution (5.48) receives extreme value. This boundary can be determined by solving the following set of equations:

$$\begin{aligned} \pi_1^- : \quad \underline{N}_1^T \cdot \underline{\tau}^{HT} &= -s_1^{HT} & \pi_1^+ : \quad \underline{N}_1^T \cdot \underline{\tau}^{HT} &= s_1^{HT} \\ \pi_2^- : \quad \underline{N}_2^T \cdot \underline{\tau}^{HT} &= -s_2^{HT} & \pi_2^+ : \quad \underline{N}_2^T \cdot \underline{\tau}^{HT} &= s_2^{HT} \\ \pi_3^- : \quad \underline{N}_3^T \cdot \underline{\tau}^{HT} &= -s_3^{HT} & \pi_3^+ : \quad \underline{N}_3^T \cdot \underline{\tau}^{HT} &= s_3^{HT} \\ \pi_4^- : \quad \underline{N}_4^T \cdot \underline{\tau}^{HT} &= -s_4^{HT} & \pi_4^+ : \quad \underline{N}_4^T \cdot \underline{\tau}^{HT} &= s_4^{HT} \end{aligned} \quad (5.56)$$

The plane π_i^- (π_i^+) represents a set of all $\underline{\tau}^{HT}$ for which $\underline{u}_i^{HT} = -s_i^{HT}$ ($\underline{u}_i^{HT} = s_i^{HT}$). Planes π_i^- and π_i^+ are parallel. The feasible region Φ_p^{HT} is a convex polyhedron inside Φ_v^{HT} determined by intersection of the four pairs of parallel planes defined by (5.56). Vertices of Φ_p^{HT} can be found as the intersection of each group of a three non-parallel planes, which lie inside the virtual control space Φ_v^{HT} or on its boundary.

Normal vectors for different thruster configurations, shown in Table 5.14, are obtained

from (5.46) & (5.47). The parameter w is defined as $w = \sum_{i=1}^4 w_i^{HT}$.

	X-shaped configuration	Cross-shaped configuration
\mathbf{N}_1^T	$\frac{1}{w} [2(w_3^{HT} + w_4^{HT}) \quad 2(w_2^{HT} + w_3^{HT}) \quad 2(w_2^{HT} + w_4^{HT})]$	$\frac{1}{w} [(2w_2^{HT} + w_3^{HT} + w_4^{HT}) \quad (w_3^{HT} - w_4^{HT}) \quad 2(w_3^{HT} + w_4^{HT})]$
\mathbf{N}_2^T	$\frac{1}{w} [2(w_3^{HT} + w_4^{HT}) \quad -2(w_1^{HT} + w_4^{HT}) \quad -2(w_1^{HT} + w_3^{HT})]$	$\frac{1}{w} [(2w_1^{HT} + w_3^{HT} + w_4^{HT}) \quad (w_4^{HT} - w_3^{HT}) \quad -2(w_3^{HT} + w_4^{HT})]$
\mathbf{N}_3^T	$\frac{1}{w} [2(w_1^{HT} + w_2^{HT}) \quad 2(w_1^{HT} + w_4^{HT}) \quad -2(w_2^{HT} + w_4^{HT})]$	$\frac{1}{w} [(w_1^{HT} - w_2^{HT}) \quad (w_1^{HT} + w_2^{HT} + 2w_4^{HT}) \quad 2(w_1^{HT} + w_2^{HT})]$
\mathbf{N}_4^T	$\frac{1}{w} [2(w_1^{HT} + w_2^{HT}) \quad -2(w_2^{HT} + w_3^{HT}) \quad 2(w_1^{HT} + w_3^{HT})]$	$\frac{1}{w} [(w_2^{HT} - w_1^{HT}) \quad (w_1^{HT} + w_2^{HT} + 2w_3^{HT}) \quad -2(w_1^{HT} + w_2^{HT})]$

Table 5.14 Normal vectors for different thruster configurations.

Figure 5.22 displays the shape of Φ_p^{HT} for different thruster configurations. Each facet is denoted by a label showing the corresponding plane and component of \underline{u}^{HT} that is saturated on the facet. Facets are colour-coded, in order to make their recognition easier.

Table 5.15 displays facets and their colour codes.

Facet	π_1^+, π_1^-	π_2^+, π_2^-	π_3^+, π_3^-	π_4^+, π_4^-
Colour	red	green	blue	cyan

Table 5.15 Facets of Φ_p^{HT} and colour codes.

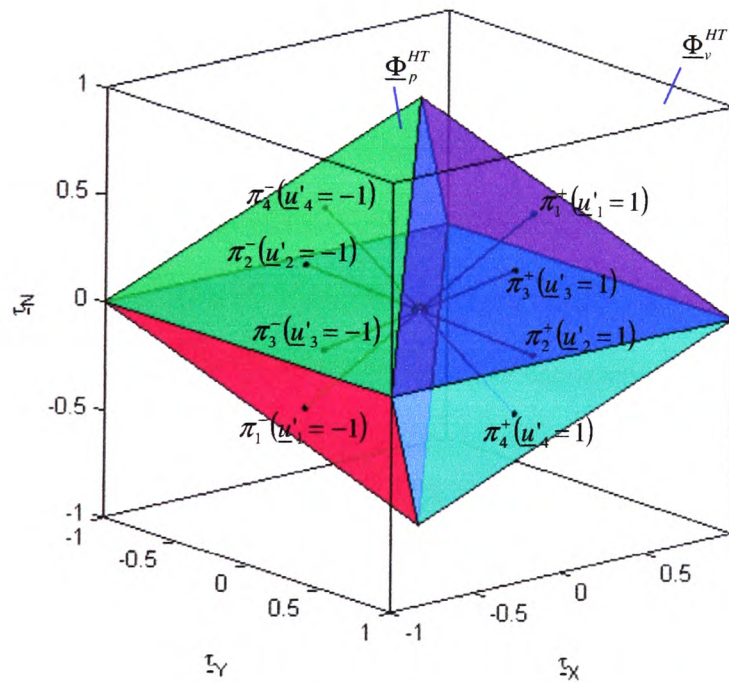
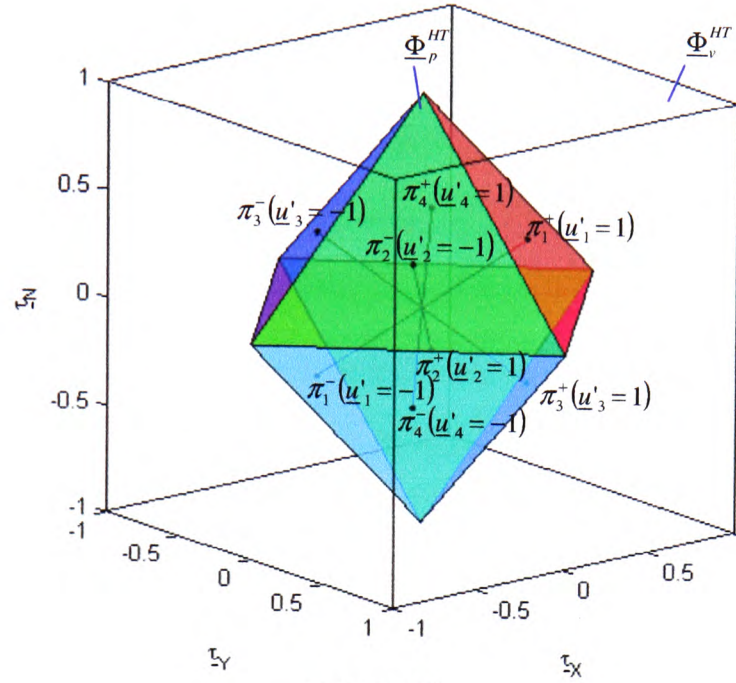


Figure 5.22 Feasible region Φ_p^{HT} for different thruster configurations.

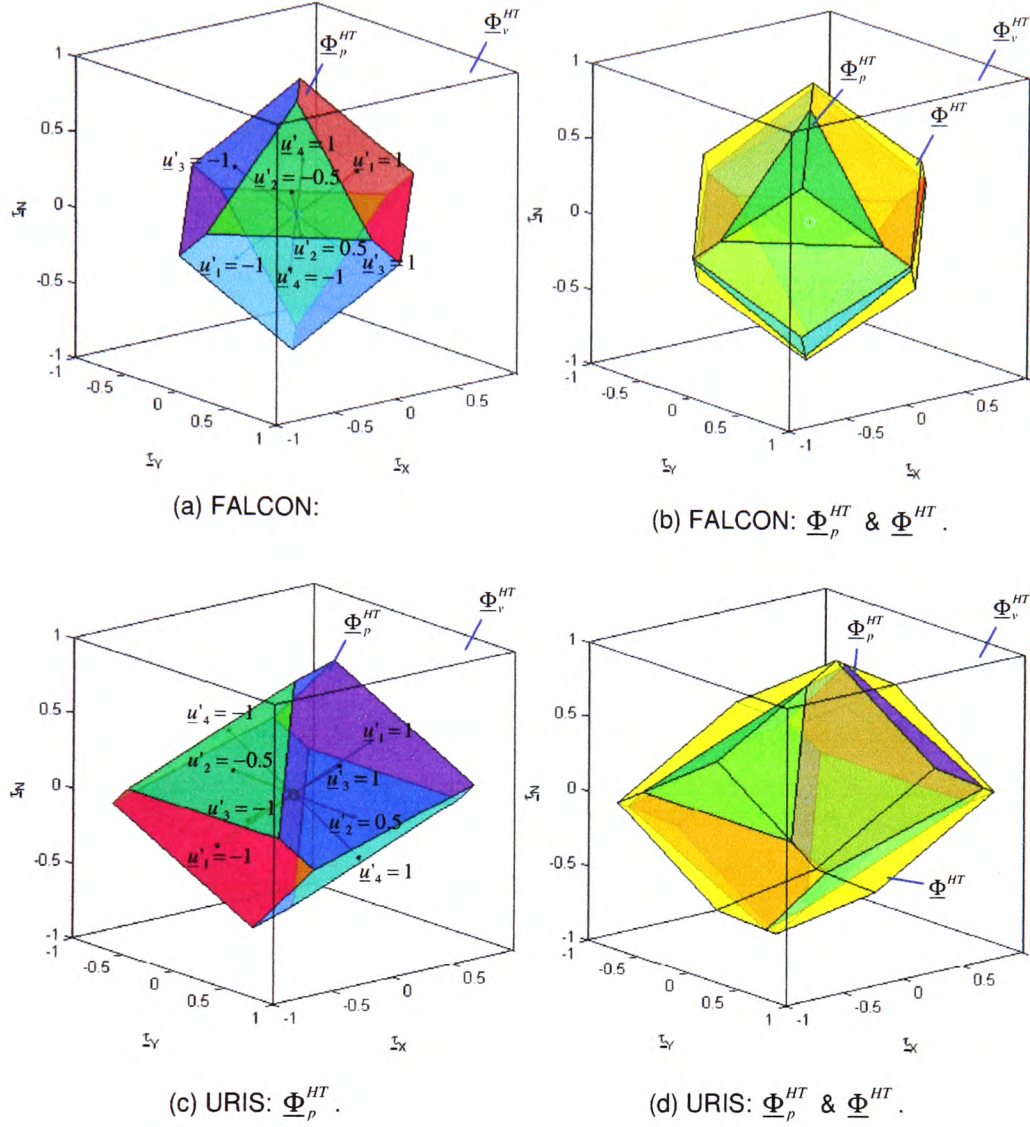


Figure 5.23 Partitions of the virtual control space Φ_v^{HT} for faulty state "Heavy jammed propeller" in 2HT ($s_2^{HT} = 0.5$).

In the case of a fault in a single thruster, the feasible region for pseudoinverse Φ_p^{HT} and attainable command set Φ_v^{HT} shrink, as shown in Figure 5.23. In particular, Figure 5.23 (a) & (c) display the region Φ_p^{HT} for the case of a partial fault ("Heavy jammed

propeller") in 2HT . In accordance to the fault code table, "Co-ordinator" penalises 2HT by increasing its weight ($w_2^{HT} = 3$, equations (5.35) & (5.36)) and changes the corresponding saturation bound to $s_2^{HT} = 0.5$. The geometrical interpretation can be obtained by observing that the change of weight w_2^{HT} produces changes in \mathbf{N}_1^T , \mathbf{N}_3^T and \mathbf{N}_4^T , but no change in \mathbf{N}_2^T . This means that the planes π_i^- and π_i^+ , $i \in \{1,3,4\}$ change their slopes, while the planes π_2^- and π_2^+ move closer to the origin, staying parallel i.e. without changing their slopes. The pseudoinverse guarantees equality between the desired vector $\underline{\mathbf{t}}_d^{HT}$ and the actual vector $\underline{\mathbf{t}}^{HT}$ inside $\underline{\Phi}_p^{HT}$, and, at the same time, the value of $|\underline{\mathbf{u}}_2^{HT}|$ will never be greater than $s_2^{HT} = 0.5$. The shape of the constrained control subset $\underline{\Omega}^{HT}$ is changed to a compressed 4D cube, in accordance with the change in the saturation bound s_2^{HT} . This change produces a change in the shape of the attainable command set $\underline{\Phi}^{HT}$, as indicated in Figure 5.23 (b) & (d), where $\underline{\Phi}_p^{HT}$ and $\underline{\Phi}^{HT}$ are shown together. It can be seen that $\underline{\Phi}_p^{HT} \subset \underline{\Phi}^{HT}$, i.e. the pseudoinverse is not able to find a feasible solution in the region $\underline{\Phi}^{HT} \setminus \underline{\Phi}_p^{HT}$. In this case the fixed-point iteration method, able to find a feasible solution optimal in the l_2 sense, is activated. In this way, hybrid approach allocates the entire attainable command set in an optimal way, despite the limited usage of a faulty thruster.

The extreme (worst) case (total breakdown in 2HT) is shown in Figure 5.24. In particular, Figure 5.24 (a) & (c) display the region $\underline{\Phi}_p^{HT}$ for the case of a total fault ("Broken propeller") in 2HT . In accordance with the fault code table, the "Co-ordinator" penalises 2HT by increasing its weight ($w_2^{HT} \rightarrow \infty$) and changes the corresponding saturation bound to $s_2^{HT} = 0$ (see page 5-29).

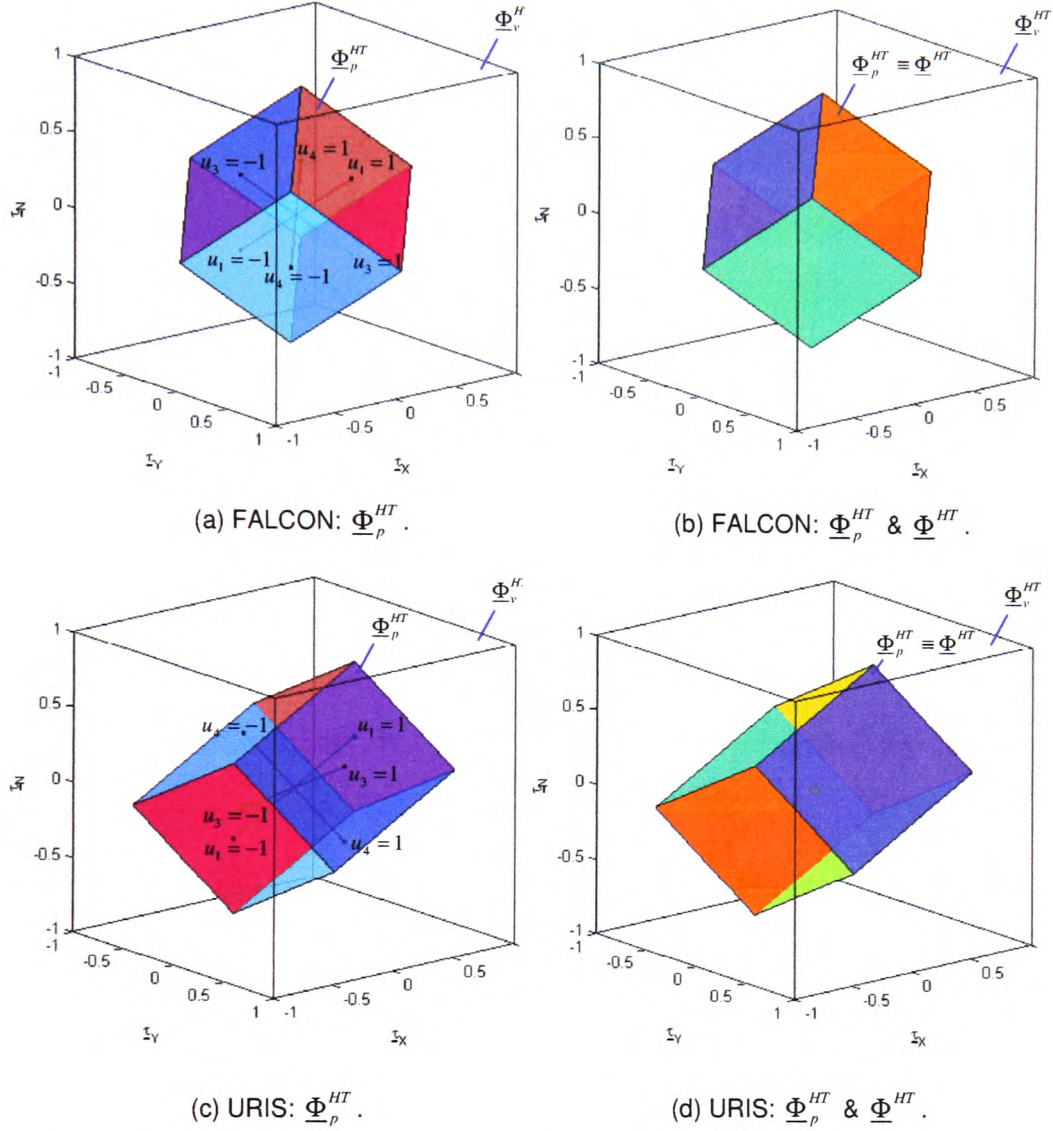


Figure 5.24 Partitions of the virtual control space Φ_v^{HT} for faulty state "Broken propeller" in 2HT ($s_2^{HT} = 0$).

This means that the thruster 2HT is switched off and the redundancy is eliminated from the control allocation problem by removing the variable \underline{u}_2^{HT} from the system of equations (5.24) & (5.25), which is simplified to

$$\begin{aligned}
 \text{FALCON:} \quad & \frac{1}{4}\underline{u}_1^{HT} + \frac{1}{4}\underline{u}_3^{HT} + \frac{1}{4}\underline{u}_4^{HT} = \underline{t}_x \\
 & \frac{1}{4}\underline{u}_1^{HT} + \frac{1}{4}\underline{u}_3^{HT} - \frac{1}{4}\underline{u}_4^{HT} = \underline{t}_y \\
 & \frac{1}{4}\underline{u}_1^{HT} - \frac{1}{4}\underline{u}_3^{HT} + \frac{1}{4}\underline{u}_4^{HT} = \underline{t}_N
 \end{aligned} \tag{5.57}$$

$$\begin{aligned}
 \text{URIS:} \quad & \frac{1}{2}\underline{u}_3^{HT} = \underline{t}_x \\
 & \frac{1}{2}\underline{u}_3^{HT} + \frac{1}{2}\underline{u}_4^{HT} = \underline{t}_y \\
 & \frac{1}{4}\underline{u}_1^{HT} + \frac{1}{4}\underline{u}_3^{HT} - \frac{1}{4}\underline{u}_4^{HT} = \underline{t}_N
 \end{aligned} \tag{5.58}$$

The modified thruster control matrix $\underline{\mathbf{B}}^{HT}$ in (5.57) & (5.58) is a non-singular 3×3 square matrix, and the problem can be solved in a standard way. The constrained control subset $\underline{\Omega}^{HT}$ is transformed to the 3D unit cube, defined as

$$\underline{\Omega}^{HT} = \left\{ \begin{pmatrix} \underline{u}_1^{HT} & \underline{u}_3^{HT} & \underline{u}_4^{HT} \end{pmatrix} \in \mathbb{R}^3 : \left| \underline{u}_1^{HT} \right| \leq 1, \left| \underline{u}_3^{HT} \right| \leq 1, \left| \underline{u}_4^{HT} \right| \leq 1 \right\} \tag{5.59}$$

$\underline{\Omega}^{HT}$ is mapped by modified $\underline{\mathbf{B}}^{HT}$ to $\underline{\Phi}^{HT}$, which coincides with $\underline{\Phi}_p^{HT}$, as indicated in Figure 5.24 (b) & (d). This means that the pseudoinverse solution for $w_2^{HT} \rightarrow \infty$ is equivalent to the exact solution of (5.57) & (5.58).

The ratio of volumes $\frac{V(\underline{\Phi}_p^{HT})_{\text{faulty situation}}}{V(\underline{\Phi}_p^{HT})_{\text{fault-free}}}$ or $\frac{V(\underline{\Phi}^{HT})_{\text{faulty situation}}}{V(\underline{\Phi}^{HT})_{\text{fault-free}}}$ can be used as a measure of

loss in manoeuvring capabilities of the vehicle produced by limiting the usage of a faulty thruster.

5.5.3 FAS algorithm

In this section, the FAS algorithm is presented, which summarises signal flow and processing inside the FAS, discussed in previous sections. The inputs to the algorithm are the total fault indicator vector \mathbf{f} (the output of the FDS, Figure 5.9) and the desired vector of propulsion forces and moments $\underline{\mathbf{t}}_d$ (the output of the pre-filter). The output is

the vector of desired thruster velocities \mathbf{n} , transformed into the form that is acceptable by the TCUs. Design parameters are \mathbf{W}_u , \mathbf{W}_v , ε , tol and initial iteration for the fixed-point iteration method. It is assumed that the remedial actions are stored in the fault code table (Table 5.8), covering possible faulty situations.

Algorithm 5.2 (FAS – Hybrid approach for control allocation)

1. Read inputs \mathbf{f} and τ_d .
2. Perform Actions A and B in accordance to Table 5.9 - Table 5.13.
3. Find the pseudoinverse solution using (5.48) for horizontal and (5.50) for vertical thrusters.
4. If the solution is feasible, go to step 6.
5. Otherwise, use the fixed-point iterations (4.29) to find feasible solution.
6. Use (3.91) & (3.92) to correct for not symmetrical T -curves.
7. Transform the control vector \mathbf{u}^* into the vector of desired thruster velocities \mathbf{n} , using (3.97) for each component.
8. Transform \mathbf{n} into \mathbf{n} , using TCU-dependent format.
9. Deliver \mathbf{n} as the output of the FAS.

5.6 Remarks on implementation issues

Timing issues: The heaviest computations in the FDAS are performed off-line, during the training process. Implementation of the FAS and FDU algorithms utilises the matrix representation of the control allocation problem, which can be very efficient, if high-quality numerical libraries are used during the program compilation stage. It is important to note that the existing FALCON control software for control allocation is not optimised for efficiency and can be improved using the matrix formulation of the control



allocation problem, which can partially compensate for the additional processing time needed for fault detection and accommodation.

Memory issues: Knowledge about faulty situations is encoded into the SOM prototypes, which are saved on the hard disk during the training phase. Each fault type is represented by a SOM with 100 neurons (prototype vectors) and associated labels (fault indicators). Each prototype vector has three coordinates (real numbers) and associated label (integer number). If two (five) bytes are needed to represent an integer (real) number, then the total memory space occupied by one SOM is approximately $100 \cdot (3 \cdot 5 + 1 \cdot 2) = 1000 \text{ byte} \approx 1 \text{ kB}$. For the fault code table shown in Table 5.8, it is necessary to save SOM prototypes of four fault types (with indicators 1, 2, 3 and 4). Hence, a memory space needed to store a knowledge about faulty situations for one thruster is approximately 4 kB . FALCON has five thrusters and total memory space that needs to be allocated is about 20 kB . This amount of memory is reasonable, compared to the size of modern hard disks and memory modules.

Accuracy issues: The number of fixed-point iterations, performed to find the feasible solution for cases when the pseudoinverse solution is unfeasible, depends on the desired accuracy and the choice of the design parameters. The level of accuracy is limited by the rounding error, since the FALCON control protocol requires desired velocities to be presented as integer numbers between -100 and $+100$. This means that the true control space for motion in the horizontal plane is discretised by uniform grid of $201^4 = 1.632240801 \cdot 10^9$ discrete control vectors and each solution must be rounded to the closest point in the grid. Design parameters of the FDAS must be chosen taking into account these issues. This topic is explored further in Chapter 6.

5.7 *Concluding remarks*

A novel thruster fault detection and accommodation system for overactuated open-frame underwater vehicles is presented in this chapter. The FDAS includes two subsystems: FAS and FDS. The FAS performs a novel hybrid approach for control allocation. The primary task of control allocation is enhanced with the FDS, able to monitor state of the thrusters and inform the FAS about any malfunctions using the total fault indicator vector, carrying the codes of faulty states for each thruster. The FDS is a hybrid, on-line, model-free approach, based on the integration of SOM and fuzzy C -means clustering methods. In the training phase the FDS uses data obtained during test trial to find SOM prototypes for each fault type. In the detection phase the FDS categorises the fault type by comparing the position of feature vector relative to these maps. The FAS uses information provided by the FDS to accommodate faults by performing an appropriate reconfiguration, i.e. to reallocate control energy among operable thrusters.

Despite the fact that in some cases it is necessary to perform iterations, the overall fault diagnosis and accommodation process is very fast, due to the computational efficiency of the FDAS algorithm, where the heaviest numerical calculations are performed off-line. This aspect of computational efficiency, combined with the adoption of a matrix formulation of the control allocation problem, means that the addition of the FDAS can be accomplished without the need to extend the cycle time.

The hybrid approach for control allocation is based on the integration of the pseudoinverse and the fixed-point iteration method. It is implemented as a two-step process. The pseudoinverse solution is found in the first step. Then the feasibility of the solution is examined analysing its individual components. If violation of actuator constraint(s) is detected, the fixed-point iteration method is activated in the second step. In this way, the hybrid approach is able to allocate the exact solution, optimal in the l_2

sense, inside the entire attainable command set. This solution minimises a control energy cost function, the most suitable criteria for underwater applications.

The FDS and FAS are presented at an algorithmic level. Evaluation of the proposed algorithms is performed in Chapter 6, where a number of representative test cases demonstrate key features of the proposed FDAS.

Chapter 6: Testing and Evaluation of the FDAS

6.1 Introduction

A new thruster fault diagnosis and accommodation system (FDAS) has been proposed in Chapter 5. The performance of the FDAS is evaluated and its key features highlighted in this chapter. The FDAS has been implemented as a Simulink model (ROV simulator, Appendix D), which was used to simulate a number of representative test cases, also presented in this chapter. These test cases were chosen to examine the behaviour of the FDAS in different situations. In order to make easier comparison, simulation results for different thruster configurations are shown next to each other. The FAS was tested using FALCON at the QinetiQ Ocean Basin Tank at Haslar, UK. Here the FAS was used in a real-world situation, where the motion of the vehicle was controlled with different, artificially generated fault conditions in the thrusters. Preliminary results from these experiments are presented at the end of chapter.

This chapter is organised as follows. The performance of the FDU, described in section 5.4.3, is evaluated using the full set of data acquired during experiments with URIS and results are presented in section 6.2. Simulation results, presented in section 6.3, evaluate the performance of the FDAS through the series of test cases for fault-free and faulty situations. The results from the FALCON trials at Haslar (described above) are presented in Section 6.4. Finally, concluding remarks are given in section 6.5.

6.2 Evaluation of the FDU

As stated in section 5.4.3, a large data set was acquired during test trials and only a part of this data was used for training. The capability of the proposed FDU to detect external faults is evaluated using the entire data set. The FDU algorithm (Algorithm 5.1, page 5-46) is implemented as a Simulink model, shown in Figure 6.1. Signals I , n_d and n are presented as inputs to the FDU, which must estimate the state of the thruster using only

these inputs. Figure 6.2 displays the actual fault indicator and FDU output, together with training data. It can be seen that the FDU identifies the new thruster state correctly in a short time after the change in state (circled regions in Figure 6.2). These delays are unavoidable, because the thruster must spend some time in a faulty state before the faults can be identified. The delays are proportional to the buffer size s . A conservative value $s = 25$ was used in Figure 6.2, in order to prevent a wrong detection and false alarms in the critical zone (see Figure 5.14). It is expected that the buffer size and delay will be reduced in future similar trials with FALCON, due to advanced signal conditioning and better quality of measured signals.

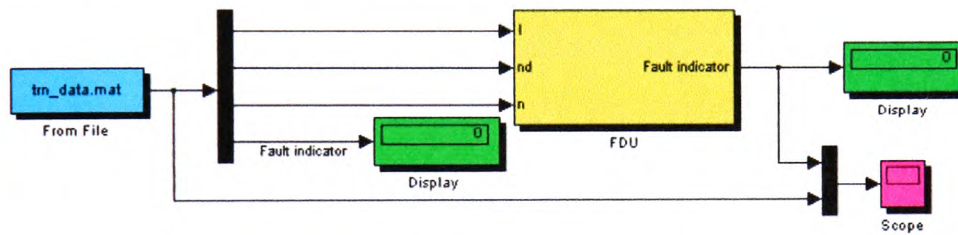


Figure 6.1 Simulink model for evaluation of the FDU.

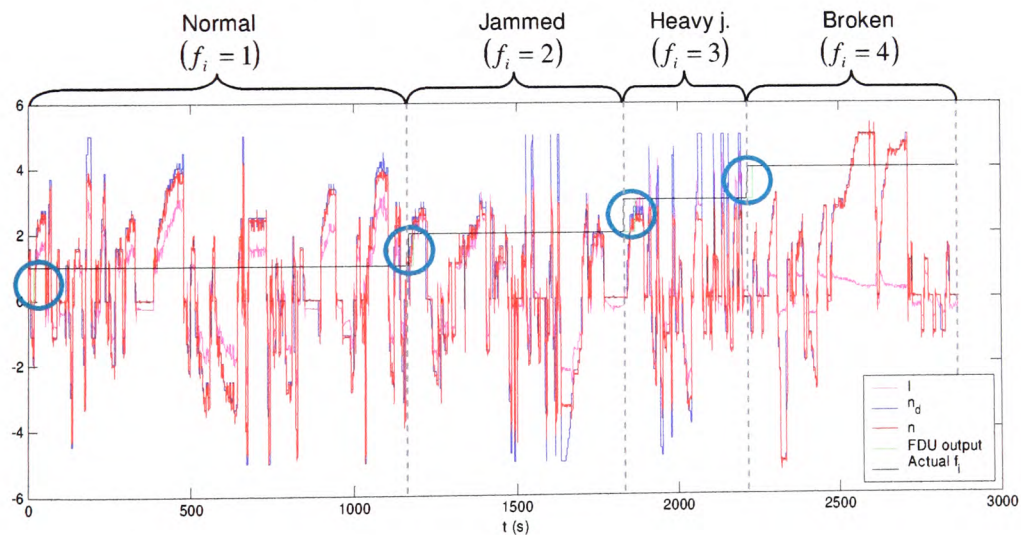


Figure 6.2 Evaluation is performed by comparing the actual fault indicator and FDU output.

Another improvement of the FDU can be obtained by using Mahalanobis metric for finding the closest BMUs from the feature vector in the on-line fault detection phase. The Mahalanobis metric automatically accounts for the scaling of the coordinate axes, corrects for correlation between the different features and can provide curved as well as linear decision boundaries. However, the price that must be paid is higher memory and time requirements. In addition, the covariance matrix is typically hard to determine accurately, which can reduce the quality of improvements.

6.3 Simulation results

Simulation results, presented in this section, are divided into two groups: A and B. Group A is a collection of test cases, which assume fault-free thruster states. The test cases from group B demonstrate the behaviour of the vehicle in different faulty situations. Table 6.1 and Table 6.2 enable easy navigation through these test cases.

Test case	Description	Diagrams
A1: Feasible pseudoinverse solution	pg. 6-8	FALCON: pg. 6-31 URIS: pg. 6-31
A2: Unfeasible pseudoinverse solution – activation of the fixed-point iterations	pg. 6-8	FALCON: pg. 6-32 URIS: pg. 6-32
A3: Approximation of an unattainable command input	pg. 6-10	FALCON: pg. 6-33 URIS: pg. 6-33
A4: Feasible trajectory – the trajectory lies inside the feasible region for pseudoinverse	pg. 6-12	FALCON: pg. 6-34 URIS: pg. 6-35
A5: Feasible trajectory – the trajectory lies inside the attainable command set	pg. 6-13	FALCON: pg. 6-36 URIS: pg. 6-37
A6: Partially unfeasible trajectory – the trajectory partially lies outside the attainable command set	pg. 6-14	FALCON: pg. 6-38 URIS: pg. 6-39
A7: Difference between affine and bilinear thruster model	pg. 6-16	FALCON: pg. 6-40 URIS: pg. 6-41
A8: Choice of propeller spin direction	pg. 6-17	FALCON: pg. 6-42 URIS: pg. 6-43
A9: Motion in vertical plane	pg. 6-19	FALCON: pg. 6-44
A10: Difference between symmetrical and non-symmetrical T-curve	pg. 6-20	FALCON: pg. 6-45 URIS: pg. 6-46

Table 6.1 Navigation table for test cases in the group A (fault-free cases).

Test case	Description	Diagrams
B1: Partial fault – “Jammed propeller”	pg. 6-24	FALCON: pg. 6-47 URIS: pg. 6-48
B2: Partial fault – “Heavy jammed propeller”	pg. 6-26	FALCON: pg. 6-49 URIS: pg. 6-50
B3: Partial fault – “Unknown state”	pg. 6-26	FALCON: pg. 6-51 URIS: pg. 6-52
B4: Total fault – “Broken propeller”	pg. 6-27	FALCON: pg. 6-53 URIS: pg. 6-54
B5: Consecutive faults – passing through the pipe	pg. 6-28	FALCON: pg. 6-55 URIS: pg. 6-56
B6: Consecutive faults – passing through the hole in the rock	pg. 6-29	FALCON: pg. 6-57 URIS: pg. 6-58

Table 6.2 Navigation table for test cases in the group B (faulty situations).

In order to improve the clarity and readability of the results presented, some symbols defined in previous chapters are replaced with a simplified version, as shown in Table 6.3. Description of the display showing the distribution of propulsion forces can be found in Appendix D. Dynamics of the thruster control loop (Figure 3.16) and umbilical cable are neglected in all test cases.

Symbol	Type	Definition	Description
$\underline{\mathbf{T}}_d^{HT}$	Vector (\mathfrak{R}^2)	$\underline{\mathbf{T}}_d^{HT} = [\underline{\tau}_{xd} \quad \underline{\tau}_{yd}]$	Projection of $\underline{\mathbf{T}}_d^{HT}$ in the $\tau_x - \tau_y$ plane
$\underline{\mathbf{T}}_f^{HT}$	Vector (\mathfrak{R}^2)	$\underline{\mathbf{T}}_f^{HT} = [\underline{\tau}_{xf} \quad \underline{\tau}_{yf}]$	Projection of $\underline{\mathbf{T}}_f^{HT}$ in the $\tau_x - \tau_y$ plane
v	Scalar	$v = \ \mathbf{v}_1\ _2$	Module of the linear velocity vector \mathbf{v}_1
ψ	Scalar	$\psi = {}^E\eta_2(3)$	Heading (yaw angle) of the vehicle in $\{E\}$
x_E, y_E, z_E	Scalars	${}^E\eta_1 = [x_E \quad y_E \quad z_E]$	Coordinates of the vehicle in $\{E\}$
$\underline{u}_i, i = \overline{1,4}$	Scalars	$\underline{\mathbf{u}}^{HT} = [\underline{u}_1 \quad \underline{u}_2 \quad \underline{u}_3 \quad \underline{u}_4]$	Components of the control vector $\underline{\mathbf{u}}^{HT}$
$\underline{u}'_i, i = \overline{1,4}$	Scalars	$\underline{\mathbf{u}}'^{HT} = [\underline{u}'_1 \quad \underline{u}'_2 \quad \underline{u}'_3 \quad \underline{u}'_4]$	Components of the control vector $\underline{\mathbf{u}}'^{HT}$
$\underline{T}_i, i = \overline{1,4}$	Scalars	$\underline{T}_i = \text{sgn}(\underline{u}_i) \frac{\ \mathbf{T}_{HT}\ _2}{T_m}$	Normalised force exerted by thruster iHT

Table 6.3 Abbreviated symbols used in diagrams in this chapter.

In order to enhance the graphical presentation of simulation results and to improve understanding of the underlying fault detection and accommodation approach, a virtual underwater world has been developed with two ROV models (FALCON and URIS) in a realistic underwater environment. The relative position of different objects in the underwater world is shown in Figure 6.3. Three particular objects (the rock with a hole in the middle, long pipe and "Stonehenge"-like group of rocks) are used throughout the test cases in this section to evaluate the manoeuvring capabilities and performance of the ROV, when equipped with the FDAS. Appendix D provides more information about the virtual underwater world. It is important to emphasize that the results presented in this section should be considered from the qualitative point of view, i.e. they serve only to highlight the key features and principles of the FDAS and to outline the main enhancements obtained by introducing the FDAS concepts into the ROV control architecture. However, the actual dynamics of FALCON and URIS are faster than dynamics of the vehicle used in simulations (see Appendix D for more information about dynamic models of FALCON and URIS). More accurate simulations will be attainable when the work on modelling and identification of FALCON and URIS is completed and corresponding dynamic models become available.

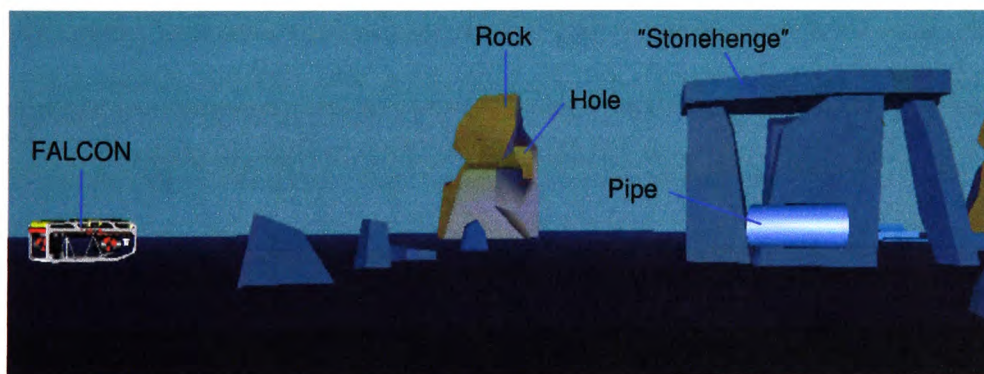


Figure 6.3 Virtual underwater world.

6.3.1 (A) Fault-free case

For all test cases in this subsection it is assumed that each thruster is in a fault-free state, i.e. $w_i^{HT} = 1$, $s_i^{HT} = 1$, $i = \overline{1,4}$ and $w_i^{VT} = 1$, $s_i^{VT} = 1$. Design parameters of the fixed-point method are $\mathbf{W}_u^{HT} = \mathbf{I}_4$, $\mathbf{W}_v^{HT} = \mathbf{I}_3$, $\varepsilon = 10^{-6}$ and $tol = 10^{-6}$. Other simulation settings are given in Table 6.4. The ROV simulator was used for simulation and results (diagrams and movies) are saved on disk for post-simulation analysis.

Test case	T-curve	Thruster model	Propeller spin direction ¹	
			FALCON	URIS
(A1)	Symmetrical ($T_{n n }^+ = T_{n n }^-$)	Affine (3.99)	${}^1C_{HT} = {}^2C_{HT} = 1$ (CW) ${}^3C_{HT} = {}^4C_{HT} = -1$ (CCW)	${}^1C_{HT} = {}^3C_{HT} = 1$ (CW) ${}^2C_{HT} = {}^4C_{HT} = -1$ (CCW)
(A2)	Symmetrical ($T_{n n }^+ = T_{n n }^-$)	Affine (3.99)	${}^1C_{HT} = {}^2C_{HT} = 1$ (CW) ${}^3C_{HT} = {}^4C_{HT} = -1$ (CCW)	${}^1C_{HT} = {}^3C_{HT} = 1$ (CW) ${}^2C_{HT} = {}^4C_{HT} = -1$ (CCW)
(A3)	Symmetrical ($T_{n n }^+ = T_{n n }^-$)	Affine (3.99)	${}^1C_{HT} = {}^2C_{HT} = 1$ (CW) ${}^3C_{HT} = {}^4C_{HT} = -1$ (CCW)	${}^1C_{HT} = {}^3C_{HT} = 1$ (CW) ${}^2C_{HT} = {}^4C_{HT} = -1$ (CCW)
(A4)	Symmetrical ($T_{n n }^+ = T_{n n }^-$)	Affine (3.99)	${}^1C_{HT} = {}^2C_{HT} = 1$ (CW) ${}^3C_{HT} = {}^4C_{HT} = -1$ (CCW)	${}^1C_{HT} = {}^3C_{HT} = 1$ (CW) ${}^2C_{HT} = {}^4C_{HT} = -1$ (CCW)
(A5)	Symmetrical ($T_{n n }^+ = T_{n n }^-$)	Affine (3.99)	${}^1C_{HT} = {}^2C_{HT} = 1$ (CW) ${}^3C_{HT} = {}^4C_{HT} = -1$ (CCW)	${}^1C_{HT} = {}^3C_{HT} = 1$ (CW) ${}^2C_{HT} = {}^4C_{HT} = -1$ (CCW)
(A6)	Symmetrical ($T_{n n }^+ = T_{n n }^-$)	Affine (3.99)	${}^1C_{HT} = {}^2C_{HT} = 1$ (CW) ${}^3C_{HT} = {}^4C_{HT} = -1$ (CCW)	${}^1C_{HT} = {}^3C_{HT} = 1$ (CW) ${}^2C_{HT} = {}^4C_{HT} = -1$ (CCW)
(A7)	Symmetrical ($T_{n n }^+ = T_{n n }^-$)	Affine (3.99) & Bilinear (3.98)	${}^1C_{HT} = {}^2C_{HT} = 1$ (CW) ${}^3C_{HT} = {}^4C_{HT} = -1$ (CCW)	${}^1C_{HT} = {}^3C_{HT} = 1$ (CW) ${}^2C_{HT} = {}^4C_{HT} = -1$ (CCW)
(A8)	Symmetrical ($T_{n n }^+ = T_{n n }^-$)	Affine (3.99)	Different combinations	Different combinations
(A9)	Symmetrical ($T_{n n }^+ = T_{n n }^-$)	Affine (3.99)	${}^1C_{VT} = 1$ (CW)	-
(A10)	Symmetrical ($T_{n n }^+ = T_{n n }^-$) & Nonsymmetrical ($T_{n n }^+ \neq T_{n n }^-$)	Affine (3.99)	${}^1C_{HT} = {}^2C_{HT} = 1$ (CW) ${}^3C_{HT} = {}^4C_{HT} = -1$ (CCW)	${}^1C_{HT} = {}^3C_{HT} = 1$ (CW) ${}^2C_{HT} = {}^4C_{HT} = -1$ (CCW)

Table 6.4 Simulation settings for group A tests.

¹ See Example 3.1 on page 3-27 for description of propeller spin direction.

(A1) Feasible pseudoinverse solution

Let $\underline{\tau}_d^{HT} = [0.18 \ 0.28 \ 0.34]^T$ for X-shaped thruster configuration in fault-free case. The pseudoinverse solution $\underline{u}^{HT} = [0.80 \ -0.44 \ 0.12 \ 0.24]^T$ is feasible and belongs to $\underline{\Omega}^{HT}$. Hence, $\underline{\tau}_d^{HT} \in \underline{\Phi}_p^{HT}$ so the fixed-point iterations don't need to be activated. The solution \underline{u}^{HT} leads to $\underline{\tau}^{HT} = \underline{B}^{HT} \underline{u}^{HT} = [0.18 \ 0.28 \ 0.34]^T$, i.e. $\underline{\tau}^{HT} \equiv \underline{\tau}_d^{HT}$, direction error $\theta = 0^\circ$ and magnitude error $\|\underline{\tau}_d^{HT} - \underline{\tau}^{HT}\|_2 = 0$ (Figure 6.4 (a)). Since $\underline{\tau}_d^{HT}$ lies inside $\underline{\Phi}_p^{HT}$, there is an infinite number of feasible solutions², but the pseudoinverse solution \underline{u}^{HT} is optimal in the l_2 sense. Distribution of propulsion forces among horizontal thrusters is shown in Figure 6.4 (b). Similar case for the cross-shaped thruster configuration is shown in Figure 6.4 (c) & (d). If $\underline{\tau}_d^{HT} = [0.42 \ 0.28 \ 0.32]^T \in \underline{\Phi}_p^{HT}$, then $\underline{u}^{HT} = [0.74 \ 0.10 \ 0.60 \ -0.04]^T \in \underline{\Omega}^{HT}$.

The main conclusion of this test case is that, while $\underline{\tau}_d^{HT}$ lies inside $\underline{\Phi}_p^{HT}$, the pseudoinverse solution is always feasible and optimal in the l_2 sense.

(A2) Unfeasible pseudoinverse solution – activation of the fixed-point iterations

In this test case desired virtual control input $\underline{\tau}_d^{HT}$ is chosen to lie in $\underline{\Phi}^{HT} \setminus \underline{\Phi}_p^{HT}$. Table 6.5 summarises the steps undertaken by the FAS to solve control allocation problem in this case. Figure 6.5 (a) & (c) display the virtual control inputs $\underline{\tau}_i^{*HT}$ and $\underline{\tau}_s^{*HT}$ (obtained by the T - and S -approximation) and $\underline{\tau}_f^{*HT}$ (obtained by the fixed-point iterations). Individual

² The FALCON control protocol discretises the true control space (see section 5.6), which makes the number of feasible solutions finite.

fixed-point iterations converge in the true control space towards the exact solution \underline{u}_f^{*HT} , optimal in the l_2 sense. The control variable \underline{u} and force (thrust) \underline{T} , exerted by a thruster, have the same numerical value in a normalised form for the affine thruster model and symmetrical T -curve (see test case (A7)), and the visualisation of \underline{u}_f^{*HT} can be obtained by showing the distribution of propulsion forces for horizontal thrusters (Figure 6.5 (b)).

	FALCON	URIS
Desired virtual control input $\underline{\tau}_d^{HT}$	$[0.70 \ 0.20 \ 0.25]^T$	$[0.975 \ -0.025 \ 0.475]^T$
Pseudoinverse solution \underline{u}^{HT}	$[1.15 \ 0.25 \ 0.65 \ 0.75]^T$	$[1.45 \ 0.50 \ 0.45 \ -0.50]^T$
Feasibility of \underline{u}^{HT}	Unfeasible, since $u_1 > 1$	Unfeasible, since $u_1 > 1$
T-approximation \underline{u}_t^{*HT}	$[1.00 \ 0.25 \ 0.65 \ 0.75]^T$	$[1.00 \ 0.50 \ 0.45 \ -0.50]^T$
Obtained virtual control input $\underline{\tau}_t^{*HT}$	$[0.6625 \ 0.1625 \ 0.2125]^T$	$[0.7500 \ -0.0250 \ 0.3625]^T$
Direction error θ_t	2.6362°	0.4366°
Magnitude error $\ \underline{\tau}_d^{HT} - \underline{\tau}_t^{*HT}\ _2$	0.0650	0.2516
S-approximation \underline{u}_s^{*HT}	$[1 \ 0.2174 \ 0.5652 \ 0.6522]^T$	$[1 \ 0.3448 \ 0.3103 \ -0.3448]^T$
Obtained virtual control input $\underline{\tau}_s^{*HT}$	$[0.6087 \ 0.1739 \ 0.2174]^T$	$[0.6724 \ -0.0172 \ 0.3276]^T$
Direction error θ_s	0°	0°
Magnitude error $\ \underline{\tau}_d^{HT} - \underline{\tau}_s^{*HT}\ _2$	0.1004	0.3367
FPI solution \underline{u}_f^{*HT}	$[1.00 \ 0.50 \ 0.45 \ -0.50]^T$	$[1.00 \ 0.95 \ 0.90 \ -0.95]^T$
Obtained virtual control input $\underline{\tau}_f^{*HT}$	$[0.70 \ 0.20 \ 0.25]^T$	$[0.975 \ -0.025 \ 0.475]^T$
Direction error θ_f	0°	0°
Magnitude error $\ \underline{\tau}_d^{HT} - \underline{\tau}_f^{*HT}\ _2$	0	0

Table 6.5 Hybrid approach for unfeasible pseudoinverse solution.

Corresponding iterations in the virtual control space are shown in Figure 6.5 (a) & (c) as red dots (for the initial iteration $\underline{\mathbf{u}}_{f0}^{*HT} = \underline{\mathbf{u}}_s^{*HT}$) and black dots (for $\underline{\mathbf{u}}_{f0}^{*HT} = \underline{\mathbf{u}}_f^{*HT}$). It is interesting to note that, in the case of FALCON, iterations starting from $\underline{\mathbf{u}}_s^{*HT}$ coincide with iterations starting from $\underline{\mathbf{u}}_f^{*HT}$ after the second iteration.

The main conclusion of the test case (A2) is that the fixed-point iterations can find the exact, l_2 -optimal solution for $\underline{\boldsymbol{\tau}}_d^{HT} \in \Phi^{HT} \setminus \Phi_p^{HT}$. Test cases (A1) and (A2) demonstrates that the hybrid approach for control allocation, implemented in the FAS, is able to find the exact, feasible solution on the entire attainable command set, optimal in the l_2 sense.

(A3) Approximation of an unattainable command input

In this test case desired virtual control input $\underline{\boldsymbol{\tau}}_d^{HT}$ is chosen to lie outside the attainable command set, i.e. $\underline{\boldsymbol{\tau}}_d^{HT} \in \Phi_v^{HT} \setminus \Phi^{HT}$. As stated in section 5.2.4, the exact, feasible solution of the control allocation problem does not exist in this case. Each method for control allocation leads to an approximate solution. Different approximations (T -, S - and FPI approximation) are shown in Table 6.6. It can be seen that the FPI approximation has the smallest magnitude error and the largest direction error. In contrast, the S -approximation has the largest magnitude error and zero direction error. The choice of approximation depends on the error priority: if directionality is important, the S -approximation is a favourite. Otherwise, if the magnitude error is prioritised, the FPI approximation is the best choice. Figure 6.6 (a) & (c) depict the virtual control inputs $\underline{\boldsymbol{\tau}}_f^{*HT}$ and $\underline{\boldsymbol{\tau}}_s^{*HT}$ (obtained by the T - and S -approximation, respectively) and $\underline{\boldsymbol{\tau}}_f^{*HT}$ (obtained by the fixed-point iterations). Similarly to test case (A2), in the case of FALCON, iterations starting from $\underline{\mathbf{u}}_s^{*HT}$ (red dots in Figure 6.6 (a)) coincide with iterations starting from $\underline{\mathbf{u}}_f^{*HT}$ (black dots) after the second iteration. Figure 6.6 (b) visualises the FPI approximate solution.

The main conclusion of the test case (A3) is that, for the case where the exact, feasible solution does not exist ($\underline{\tau}_d^{HT} \notin \Phi^{HT}$), the hybrid approach provides different approximations (T -, S - and FPI approximation) with different approximation errors.

	FALCON	URIS
Desired virtual control input $\underline{\tau}_d^{HT}$	$[0.90 \ 0.80 \ 0.70]^T$	$[0.70 \ 0.80 \ 0.90]^T$
Pseudoinverse solution \underline{u}^{HT}	$[2.4 \ -0.6 \ 1.0 \ 0.8]^T$	$[1.60 \ -0.20 \ 1.70 \ -0.10]^T$
Feasibility of \underline{u}^{HT}	Unfeasible, since $\underline{u}_1 > 1$	Unfeasible, since $\underline{u}_1 > 1$ & $\underline{u}_3 > 1$
T-approximation \underline{u}_t^{*HT}	$[1.0 \ -0.6 \ 1.0 \ 0.8]^T$	$[1.00 \ -0.20 \ 1.00 \ -0.10]^T$
Obtained virtual control input $\underline{\tau}_t^{*HT}$	$[0.55 \ 0.45 \ 0.35]^T$	$[0.4000 \ 0.4500 \ 0.5750]^T$
Direction error θ_t	4.4565°	3.4453°
Magnitude error $\ \underline{\tau}_d^{HT} - \underline{\tau}_t^{*HT}\ _2$	0.6062	0.5640
S-approximation \underline{u}_s^{*HT}	$[1 \ -0.25 \ 0.4167 \ 0.3333]^T$	$[0.9412 \ -0.1176 \ 1 \ -0.0588]^T$
Obtained virtual control input $\underline{\tau}_s^{*HT}$	$[0.3750 \ 0.3333 \ 0.2917]^T$	$[0.4118 \ 0.4706 \ 0.5294]^T$
Direction error θ_s	0°	0°
Magnitude error $\ \underline{\tau}_d^{HT} - \underline{\tau}_s^{*HT}\ _2$	0.8125	0.5735
FPI solution \underline{u}_f^{*HT}	$[1.0 \ -1.0 \ 1.0 \ 1.0]^T$	$[1 \ -0.0326 \ 1 \ 0.1659]^T$
Obtained virtual control input $\underline{\tau}_f^{*HT}$	$[0.5 \ 0.5 \ 0.5]^T$	$[0.4837 \ 0.5829 \ 0.4667]^T$
Direction error θ_f	5.8275°	8.6994°
Magnitude error $\ \underline{\tau}_d^{HT} - \underline{\tau}_f^{*HT}\ _2$	0.5385	0.5307

Table 6.6 Hybrid approach for unattainable command input.

(A4) Feasible trajectory – the trajectory lies inside the feasible region for pseudoinverse

This test case is intended to demonstrate the ability of the FDAS to solve the control allocation problem for motion in the horizontal plane, where commanded inputs over time create the trajectory $\underline{\mathbf{x}}_d^{HT}(t)$, which completely lies inside the feasible region for the pseudoinverse $\underline{\Phi}_p^{HT}$ (Figure 6.7 (a) – FALCON and Figure 6.8 (a) - URIS). The mission objective was to move the vehicle from the start point A through the pipe, under constraint $\underline{\mathbf{x}}_d^{HT}(t) \subset \underline{\Phi}_p^{HT}$. The trajectory $\underline{\mathbf{x}}_d^{HT}(t)$ was generated by a joystick, using the virtual reality display and dynamic visualisation of the feasible region to accomplish the mission. The pseudoinverse solution of the underlying control allocation problem is feasible $\forall t$, and trajectories $\underline{\mathbf{x}}^{HT}(t)$ and $\underline{\mathbf{x}}_d^{HT}(t)$ coincide³ inside $\underline{\Phi}_p^{HT}$. Time diagrams of components of the vectors $\underline{\mathbf{x}}^{HT}(t)$ and $\underline{\mathbf{x}}_d^{HT}(t)$ are shown in Figure 6.7 (b) (FALCON) and Figure 6.8 (b) (URIS). The plan view of the motion in the $x_E - y_E$ plane is displayed in Figure 6.7 (c) (FALCON) and Figure 6.8 (c) (URIS), together with the Earth-fixed frame $\{E\}$. Figure 6.7 (d) (FALCON) and Figure 6.8 (d) (URIS) display time responses of velocity $v(t)$ and heading $\psi(t)$ ⁴. Snapshots from the virtual reality display, showing the motion of the vehicle in the virtual world, are displayed in Figure 6.7 (e) (FALCON) and Figure 6.8 (e) (URIS). The vehicle begins the motion at the start point A , rotates right,

³ However, if thruster dynamics are not neglected, then trajectories are close to each other, but they don't coincide. Similar conclusion can be drawn in other examples. This topic is discussed in section 6.4.4.

⁴ Heading response is normalised on interval $0^\circ \leq \psi(t) < 360^\circ$, in order to be compatible with the input of the Angular Gauge control "Compass" in Dials & Gauges Blockset. This normalisation introduces jumps in heading responses for cases $\psi(t) < 0^\circ$.

enters the pipe, moves forward, leaves the pipe, decelerates and stops the motion at the end point B .

The main conclusion of this test case is that, if $\underline{\tau}_d^{HT}(t) \subset \Phi_p^{HT}$, then the FDAS ensures that the actual behaviour of the vehicle is the same as the desired behaviour, without undesired effects of thruster velocity saturation.

(A5) Feasible trajectory – the trajectory lies inside the attainable command set

Similar to test case (A4), this example demonstrates the ability of the FDAS to solve the control allocation problem for motion in the horizontal plane, where the trajectory $\underline{\tau}_d^{HT}(t)$ lies inside the attainable command set Φ^{HT} (Figure 6.9 (a) – FALCON and Figure 6.10 (a) - URIS). The mission objective was to move the vehicle from the start point A through the hole in the rock, under constraint $\underline{\tau}_d^{HT}(t) \subset \Phi^{HT}$. As in test case (A4), a joystick was used to generate the trajectory $\underline{\tau}_d^{HT}(t)$. The majority of the trajectory $\underline{\tau}_d^{HT}(t)$ lies inside Φ_p^{HT} , while some parts lie in $\Phi^{HT} \setminus \Phi_p^{HT}$. The hybrid approach for control allocation, implemented in the FDAS, utilises the pseudoinverse method to find the feasible solution for $\underline{\tau}_d^{HT}(t) \in \Phi_p^{HT}$ and the fixed-point iteration method for $\underline{\tau}_d^{HT}(t) \in \Phi^{HT} \setminus \Phi_p^{HT}$. In this way, trajectories $\underline{\tau}^{HT}(t)$ and $\underline{\tau}_d^{HT}(t)$ coincide inside the entire Φ^{HT} . Figure 6.9 (b) (FALCON) and Figure 6.10 (b) (URIS) display the time diagrams of components of vectors $\underline{\tau}^{HT}(t)$ and $\underline{\tau}_d^{HT}(t)$. The plan view of the motion in the $x_E - y_E$ plane is depicted in Figure 6.9 (c) (FALCON) and Figure 6.10 (c) (URIS). Time responses of velocity $v(t)$ and heading $\psi(t)$ are shown in Figure 6.9 (d) (FALCON) and Figure 6.10 (d) (URIS). Snapshots of the journey through the hole in the rock, taken from the virtual reality display, are shown in Figure 6.9 (e) (FALCON) and Figure 6.10 (e)

(URIS). The vehicle begins the motion at the start point A , rotates right, moves forward, rotates left, enters the hole, moves forward, leaves the hole, decelerates and stops the motion at the end point B .

The main conclusion of this test case is similar to the conclusion of test case (A4); that is, if $\underline{\tau}_d^{HT}(t) \subset \Phi^{HT}$, then the FDAS provides that the actual behaviour of the vehicle is the same as desired behaviour, without undesired effects of thruster velocity saturation.

(A6) Partially unfeasible trajectory – the trajectory partially lies outside the attainable command set

This test case is intended to examine the performance of the FDAS for motion in the horizontal plane, where commanded inputs over time create a partially unfeasible trajectory $\underline{\tau}_d^{HT}(t)$, i.e. the trajectory, which partially lies outside the attainable command set Φ^{HT} (Figure 6.11 (a) – FALCON and Figure 6.12 (a) - URIS). The mission objective was to move the vehicle from the start point A through the passage in "Stonehenge", without any constraint on $\underline{\tau}_d^{HT}(t)$. As in the previous cases, the trajectory $\underline{\tau}_d^{HT}(t)$ was generated by a joystick. The hybrid approach for control allocation can approximate unfeasible parts of $\underline{\tau}_d^{HT}(t)$ by three approximation types: T -approximation, S -approximation and FPI approximation. In this test case the FPI approximation was used, such that the FPI approximations lie on the boundary $\partial(\Phi^{HT})$. In this way, trajectories $\underline{\tau}^{HT}(t)$ and $\underline{\tau}_d^{HT}(t)$ coincide inside Φ^{HT} , while thruster velocity saturation implies that they differ outside Φ^{HT} . Time diagrams of components of vectors $\underline{\tau}^{HT}(t)$ and $\underline{\tau}_d^{HT}(t)$ are shown in Figure 6.11 (b) (FALCON) and Figure 6.12 (b) (URIS). The unfeasible parts of $\underline{\tau}_d^{HT}(t)$ are denoted as shadowed regions in these diagrams. They are characterised by disagreement in components of $\underline{\tau}^{HT}(t)$ and $\underline{\tau}_d^{HT}(t)$. The plan view of the motion in the

$x_E - y_E$ plane is displayed in Figure 6.11 (c) (FALCON) and Figure 6.12 (c) (URIS). Figure 6.11 (d) (FALCON) and Figure 6.12 (d) (URIS) display time responses of velocity $v(t)$ and heading $\psi(t)$. Snapshots from the virtual reality display, showing different stages of the motion in the virtual world, are displayed in Figure 6.11 (e) (FALCON) and Figure 6.12 (e) (URIS). The vehicle begins the motion at the start point A , rotates right, moves forward, rotates left, enters the passage in "Stonehenge", moves forward, leaves the passage, decelerates and stops the motion at the end point B . If $\underline{\tau}_d^{HT}(t)$ is feasible, i.e. if $\underline{\tau}_d^{HT}(t) \in \underline{\Phi}^{HT}$, the vector of propulsion forces and moments $\underline{\tau}^{HT}(t)$, exerted by thrusters, is equal to the desired vector $\underline{\tau}_d^{HT}(t)$, generated by the HCU, and the motion of the vehicle is in accordance to command inputs. In situations when $\underline{\tau}_d^{HT}(t)$ becomes unfeasible, vectors $\underline{\tau}^{HT}(t)$ and $\underline{\tau}_d^{HT}(t)$ are not equal and there is a change in the behaviour of the vehicle. When thruster velocity saturation occurs, the vehicle reacts in a different way than in the non-saturation case. An experienced ROV pilot is able to "feel" this change and manually returns the command input $\underline{\tau}_d^{HT}(t)$ back into $\underline{\Phi}^{HT}$. However, the FDAS can automatically provide information about the position of $\underline{\tau}_d^{HT}(t)$ relative to $\underline{\Phi}^{HT}$ in different ways:

- using dynamic visualisation of $\underline{\Phi}^{HT}$, $\underline{\tau}_d^{HT}(t)$ and $\underline{\tau}^{HT}(t)$ in 3D virtual control space,
- using dynamic estimation of components of $\underline{\tau}_d^{HT}(t)$ and $\underline{\tau}^{HT}(t)$,
- using quantitative descriptor (for example, scalar indicator) to indicate the case when $\underline{\tau}_d^{HT}(t)$ lies outside $\underline{\Phi}^{HT}$.

In this way, using information provided by the FDAS, even an inexperienced ROV pilot is able to detect the situation when thruster velocity saturation occurs and to correct the input vector $\underline{\tau}_d^{HT}(t)$ such that it becomes feasible.

The main conclusion of this test case is that an ROV pilot should keep $\underline{\tau}_d^{HT}(t)$ inside $\underline{\Phi}^{HT}$ for all time, because in that case the response of the vehicle to command inputs is exactly the same as expected.

(A7) Difference between affine and bilinear thruster model

The hybrid approach for control allocation, implemented in the FDAS, was derived assuming affine thruster model, shown in Figure 5.3. This test case is intended to compare the performance of the FDAS when two different thruster models are used (affine (3.99) and bilinear (3.98)). Model parameters are given in Appendix D. In the affine thruster model, propeller thrust T and shaft torque Q_e only depend on propeller angular velocity n , that is, the control variable u . In the bilinear thruster model, T and Q_e depend on propeller angular velocity n and ambient water velocity u_a (see (3.81) & (3.82)). Figure 6.13 (a) (FALCON) and Figure 6.14 (a) (URIS) display time diagrams of $\underline{\tau}_d^{HT}(t)$, $\underline{\tau}^{HT}(t)$, $\underline{u}^{HT}(t)$, $\underline{T}_i(t)$ and $v(t)$ for the case when an affine thruster model is used. The same diagrams for bilinear thruster model are displayed in Figure 6.13 (b) (FALCON) and Figure 6.14 (b) (URIS). Vector $\underline{\tau}_d^{HT}(t)$ was created using the "From File" Simulink block. For the affine thruster model the normalised relationship between thrust (force) T_i and control variable u_i is given by

$$\underline{T}_i = \underline{u}_i \quad (6.1)$$

This can be seen from diagrams Figure 6.13 (a2) (FALCON) and Figure 6.14 (a2) (URIS), where $\underline{T}_i(t)$ and $\underline{u}_i(t)$ coincide, $\forall t$, which yields $\underline{\tau}_d^{HT}(t) \equiv \underline{\tau}^{HT}(t)$, as shown in

Figure 6.13 (a1) (FALCON) and Figure 6.14 (a1) (URIS). Equation (6.1) is not valid for bilinear thruster model, which can be seen from Figure 6.13 (b2) (FALCON) and Figure 6.14 (b2) (URIS). The second term $-T_{|n|u_a}|n|u_a$ in (3.98) is responsible for the discrepancy between $\underline{T}_i(t)$ and $\underline{u}_i(t)$. In general, the faster the vehicle, the higher the difference. For this reason, vectors $\underline{\tau}_d^{HT}(t)$ and $\underline{\tau}^{HT}(t)$ for bilinear thruster model are close to each other, but they do not coincide, as shown in Figure 6.13 (b1) (FALCON) and Figure 6.14 (b1) (URIS). The difference between $\underline{\tau}_d^{HT}(t)$ and $\underline{\tau}^{HT}(t)$ for bilinear thruster model produces the loss in velocity of the vehicle, which is visible from the velocity time response. The vehicle moves slower when the bilinear thruster model is used (Figure 6.13 (b3) (FALCON) and Figure 6.14 (b3) (URIS)) than for the affine model (Figure 6.13 (a3) (FALCON) and Figure 6.14 (a3) (URIS)).

The bilinear thruster model is more realistic than the affine model. Although the control allocator, implemented in the FDAS, was derived assuming affine thruster model, this test case demonstrated that the performance of the ROV control system, equipped with the FDAS, is satisfactory for both thruster models.

(A8) Choice of propeller spin direction

This test case is intended to examine the influence of the propeller spin direction on the performance of the FDAS. Two cases with different choice of propeller spin directions (i.e. different spin direction coefficients) were considered. In the first case, symmetrical pairs of horizontal thrusters ($^1HT \& ^3HT$ and $^2HT \& ^4HT$ for FALCON, $^1HT \& ^2HT$ and $^3HT \& ^4HT$ for URIS) had opposite spin direction coefficients ($cw \& ccw$), as indicated in Figure 6.15 (a) (FALCON) and Figure 6.16 (a) (URIS). In the second case, all horizontal thrusters had the same positive propeller spin directions (cw), as indicated in Figure 6.15 (b) (FALCON) and Figure 6.16 (b) (URIS). The mission objective was to move the

vehicle from the start point A through the pipe along the straight line. Snapshots from the virtual reality display, showing different stages of the motion in the virtual world, are shown at the bottom of Figure 6.15 (FALCON) and Figure 6.16 (URIS). The vehicle begins the motion at the start point A , enters the pipe, moves forward, leaves the pipe, decelerates and stops at the end point B . Figure 6.15 (a) (FALCON) and Figure 6.16 (a) (URIS) display time diagrams of $\underline{\tau}_d^{HT}(t)$, $\underline{\tau}^{HT}(t)$, $v(t)$, $\psi(t)$ and the plan view of trajectory $(x_E(t), y_E(t))$ for the first case. Diagrams for the second case are shown in Figure 6.15 (b) (FALCON) and Figure 6.16 (b) (URIS). It can be seen that $\underline{\tau}_d^{HT}(t)$, $\underline{\tau}^{HT}(t)$, $v(t)$ and $\psi(t)$ are identical in both cases. The only difference is noticeable in diagrams showing the plan views of trajectories $(x_E(t), y_E(t))$. In the first case the vehicle moves along the straight line without any drift. In contrast, the vehicle is shifted from the straight line in the second case, such that the drift at the end point B is approximately 28 cm for FALCON and 2 cm for URIS. The reason why the drift is higher for FALCON in the second case is because for straight-line motion all four horizontal thrusters are actuated for FALCON, compared to only two actuated thrusters for URIS. Each thruster produces torque Q_e and the net torque, created as a vector sum of torques of individual thrusters, is higher for FALCON than for URIS in the second case. This net torque is responsible for the difference in trajectories $(x_E(t), y_E(t))$ in both cases, as explained in the following. In the first case, symmetrical pairs of thrusters have opposite spin direction coefficients. This means that, if they rotate with the same velocity, resulting moment from the blade's angular motion will counteract each other and the net angular moment for each pair of symmetrical thrusters in total vector of propulsion forces and moments τ will be zero. Components of the control vector $\underline{u}^{HT}(t)$ for this case are the same as those shown for test case (A7) (affine thruster model) in Figure 6.13 (a2)

(FALCON) and Figure 6.14 (a2) (URIS), since these test cases were simulated using the same simulation settings (see Table 6.4). It can be seen that symmetrical pairs of thrusters have the same velocity during the mission, which yields zero net angular moments from these pairs. That is, the components τ_K , τ_M and τ_N of τ are equal to zero during the mission, which leads to perfect straight-line motion of the vehicle. In contrast, symmetrical pairs of thrusters have the same spin direction coefficients in the second case, which means that resulting torque Q_e from each thruster generates non-zero net angular moment, that is, the components τ_K and τ_M of τ are not equal to zero during the mission, which leads to non-perfect straight-line motion of the vehicle.

The main conclusion of this test case is that the undesired effects of torques Q_e on the straight-line motion of the vehicle can be reduced by careful choice of propeller spin direction: the symmetrical pairs of thrusters should have opposite spin direction coefficients.

(A9) Motion in vertical plane

This test case is intended to examine the performance of the FDAS for motion of FALCON in vertical plane. URIS is not considered in this example, since the modified thruster configuration shown in Figure 3.7 (b) does not allow motion in vertical plane. Figure 6.17 (a) displays time diagrams of $\mathbf{x}_d^{VT}(t) = \mathbf{x}_z(t)$, $v(t)$, $z_E(t)$ and $\psi(t)$. Vector $\mathbf{x}_d^{VT}(t)$ was created using the "From File" Simulink block. Snapshots from the virtual reality display, showing different stages of the motion from different view points, are shown in Figure 6.17. The vehicle begins the motion at the start point A , ascends, decelerates and changes the direction of motion, descends and stops at the end point B . The torque Q_e , exerted by a vertical thruster, produces undesired change of heading, as depicted in time response of $\psi(t)$. The same effect was observed during test trials with

FALCON in the experimental tank at the University of Southampton. Although the HCU sent commands for pure vertical motion, the change of heading was observed during the experiment, and the rate of change of heading was proportional to the vertical velocity of the vehicle: higher the velocity, higher the rate of change. Undesired effect of the torque Q_e can be cancelled by introducing the "Heading-Keeping" controller. The main objective is to achieve $\dot{\psi}(t) = 0$, i.e. $\psi(t) = \text{const}$. The input to the controller is the rate of change of heading $\dot{\psi}(t)$, which is measured by the on-board gyro sensor. The controller generates contra-moment τ'_N (proportional to $\dot{\psi}(t)$) using horizontal thrusters in order to reject the effect of the torque Q_e on heading. In order to prevent collision between the "Heading-Keeping" controller and the desired yaw motion τ_N , generated by the HCU, the controller should be disabled for case $\tau_N \neq 0$ and enabled for case $\tau_N = 0$. In this way, in addition to cancellation of the effect of the torque Q_e , the "Heading-Keeping" controller can be used to stop the rotation of the vehicle immediately after demand τ_N for rotation becomes equal to zero. Namely, in the absence of the controller, the vehicle continues to rotate for short time after $\tau_N = 0$, due to inertia. This "extended" rotational motion can be removed using the "Heading-Keeping" controller, which generates contra-moment when $\tau_N = 0$ to stop the rotation of the vehicle.

The main conclusion of this test case is that the undesired effects of torque Q_e on the heading, observable during the motion of the vehicle in the vertical plane, can be eliminated using the "Heading-Keeping" controller.

(A10) Difference between symmetrical and non-symmetrical T-curve

The hybrid approach for control allocation, implemented in the FDAS, was derived assuming symmetrical T -curve, shown in Figure 5.3, for each thruster. This test case is

intended to compare the performance of the FDAS when two different shapes of T -curves are used (symmetrical (6.2) - (6.3) and non-symmetrical (6.4) - (6.5)).

Symmetrical T -curve:

$$T(u) = Ku \quad (6.2)$$

$$K = T_{n|n|}^+ = T_{n|n|}^- \quad (6.3)$$

Non-symmetrical T -curve:

$$T(u) = \begin{cases} T_{n|n|}^+ u, & u > 0 \\ T_{n|n|}^- u, & u < 0 \end{cases} \quad (6.4)$$

$$\begin{aligned} T_{n|n|}^+ &= (1 + \Delta T_{n|n|})K \\ T_{n|n|}^- &= (1 - \Delta T_{n|n|})K \end{aligned} \quad (6.5)$$

Model parameters are given in Appendix D. Figure 6.18 (a) (FALCON) and Figure 6.19 (a) (URIS) display time diagrams of $\underline{\tau}_d^{HT}(t)$, $\underline{\tau}^{HT}(t)$, $\underline{u}^{HT}(t)$, $\underline{u}^{HT}(t)$, $\underline{T}_i(t)$, $v(t)$ & $\psi(t)$ and plan views of trajectories $(x_E(t), y_E(t))$ for the case when symmetrical T -curves were used. Diagrams for non-symmetrical T -curves are displayed in Figure 6.18 (b) (FALCON) and Figure 6.19 (b) (URIS). Vector $\underline{\tau}_d^{HT}(t)$ was created using the "From File" Simulink block. The vehicle starts the forward motion from the start point A, passes through the pipe (in the case of URIS it comes close to the end of the pipe), decelerates, changes the direction (reverse motion), moves back and finishes the motion at the end point B.

As explained in section 3.7.3, the FDAS transforms the non-symmetrical relationship between T and u into a symmetrical relationship by introducing auxiliary control variable u' (3.89) & (3.90) for each thruster. After the control allocator inside the FDAS finds the solution, the "Correction" block of the FDAS transforms the auxiliary control variable u' for each thruster back into the real control variable u using (3.91) & (3.92),

compensating for the non-symmetrical T -curve and ensuring that the thrust (force) is the same for both variables (see Figure 3.13). For symmetrical T -curves, the auxiliary control variable \underline{u}'_i and the real control variable \underline{u}_i for iHT are equal. For non-symmetrical T -curves, variables \underline{u}'_i and \underline{u}_i are different. The "Correction" block transforms \underline{u}'_i into \underline{u}_i such that individual thruster forces \underline{T}_i are the same for both shapes of T -curves. This leads to $\underline{\tau}^{HT}(t) \equiv \underline{\tau}_d^{HT}(t)$ in both cases, which further leads to identical diagrams of $v(t)$, $\psi(t)$ and $(x_E(t), y_E(t))$.

Although the T -curves for the majority of propellers for underwater vehicles are non-symmetrical, this test case demonstrates that the hybrid approach for control allocation, implemented in the FDAS, is applicable for both shapes of propeller T -curves (symmetrical and non-symmetrical) and obtained performance is acceptable in both cases.

6.3.2 (B) Faulty situations

Test cases in this subsection demonstrate the ability of the FDAS to accommodate different fault types in thrusters. A joystick was used in the FDS to simulate different faulty situations (see Appendix D for more information). Test cases (B1) – (B4) demonstrate the ability of the FDAS to complete the mission in the presence of a single fault in the 2HT . Diagrams for fault-free and faulty cases are shown next to each other, in order to provide easier comparison. The existing control software for FALCON and URIS does not provide a solution for thruster fault accommodation and the only option available is to switch off a faulty thruster. Test cases (B5) and (B6) compare the behaviour of the vehicle in the presence of consecutive faults in thrusters for cases when the FDAS is not active (faulty thruster is switched off) and when the FDAS is active (faulty thruster is accommodated as explained in section 5.5). As stated in test case (A3), the hybrid approach provides different approximations (T -, S - and FPI approximation)

with different approximation errors for cases when the exact, feasible solution does not exists ($\mathbf{r}_d^{HT} \notin \Phi^{HT}$). In test cases (B1) – (B6) the S -approximation is used to approximate unfeasible solutions, since it leads to zero direction error of approximation. This is important, since the priority in faulty situations is to preserve directionality of the command input vector, especially among the main body axes. Other simulation settings are given in Table 6.7. The ROV simulator was used for simulation and results (diagrams and movies) were saved on disk for post-simulation analysis. Command inputs in test cases (B1) – (B6) are generated using pre-defined signals read from a file.

Test case	T-curve	Thruster model	Propeller spin direction	
			FALCON	URIS
(B1) – (B6)	Symmetrical ($T_{n n }^+ = T_{n n }^-$)	Affine (3.99)	${}^1C_{HT} = {}^2C_{HT} = 1$ (CW) ${}^3C_{HT} = {}^4C_{HT} = -1$ (CCW)	${}^1C_{HT} = {}^3C_{HT} = 1$ (CW) ${}^2C_{HT} = {}^4C_{HT} = -1$ (CCW)

Table 6.7 Simulation settings for test cases in group B.

In contrast to the test cases in group A, a new diagram showing time response of criteria is introduced in this subsection. Two criteria are considered:

$$\text{Weighted criterion: } J_{\mathbf{w}_u^{HT}}^{HT} = (\mathbf{u}^{HT})^T \mathbf{W}_u^{HT} \mathbf{u}^{HT} = \|\mathbf{u}^{HT}\|_{\mathbf{w}_u^{HT}}^2 \quad (6.6)$$

and

$$\text{Normal criterion: } J_n^{HT} = (\mathbf{u}^{HT})^T \mathbf{u}^{HT} = \|\mathbf{u}^{HT}\|^2 \quad (6.7)$$

The weighted criterion $J_{\mathbf{w}_u^{HT}}^{HT}$, minimised by the hybrid approach, can be interpreted as a weighted control energy cost function. In contrast, the normal criterion J_n^{HT} represents the actual control energy cost function, that is, a real measure of control effort. In fault-free case $J_{\mathbf{w}_u^{HT}}^{HT} = J_n^{HT}$, whereas in faulty situations, $J_{\mathbf{w}_u^{HT}}^{HT} \geq J_n^{HT}$. Criteria $J_{\mathbf{w}_u^{HT}}^{HT}$ and J_n^{HT} are denoted in diagrams as J_w and J_n , respectively.

(B1) Partial fault - "Jammed propeller"

This test case compares the performance of the FDAS for two cases. In the first case a simulation was performed assuming fault-free states in all thrusters. The second case considered fault-free states in thrusters 1HT , 3HT and 4HT , and a faulty state "Jammed propeller" in 2HT (partial fault, $s_2^{HT} = 0.75$, see Table 5.8). The same command input vector $\underline{\tau}_d^{HT}(t)$, created using the "From File" Simulink block, was used to drive the vehicle in both cases. The mission objective was to move the vehicle from the start point A through the pipe along the straight line. Figure 6.20 (a) (FALCON) and Figure 6.21 (a) (URIS) display time diagrams of $\underline{\tau}_d^{HT}(t)$, $\underline{\tau}^{HT}(t)$, $\underline{u}^{HT}(t)$, $\underline{T}_i(t)$, $J_{\mathbf{w}_n^{HT}}^{HT}(t)$, $J_n^{HT}(t)$, $v(t)$ and $\psi(t)$, the plan view of the trajectory $(x_E(t), y_E(t))$ and partitions $\underline{\Phi}_p^{HT}$ & $\underline{\Phi}^{HT}$ of the virtual control space $\underline{\Phi}_v^{HT}$ for the first case. Diagrams for the second case are shown in Figure 6.20 (b) (FALCON) and Figure 6.21 (b) (URIS). It can be seen that, in the fault-free case, the vehicle performs perfect straight-line motion, vector $\underline{\tau}_d^{HT}(t)$ lies inside $\underline{\Phi}^{HT}$ for all time and $\underline{\tau}_d^{HT}(t) \equiv \underline{\tau}^{HT}(t)$, $J_{\mathbf{w}_n^{HT}}^{HT}(t) \equiv J_n^{HT}(t)$, $\forall t$. In addition, the velocity profile $v(t)$, which follows the profile of $\underline{\tau}_x(t)$, is exactly as desired, since $\underline{\tau}_x(t) = \underline{\tau}_{xd}(t)$. The situation is different for the second case, where the limited usage of 2HT causes shrinking of the $\underline{\Phi}_p^{HT}$ and $\underline{\Phi}^{HT}$ (see section 5.5.2), and the trajectory $\underline{\tau}_d^{HT}(t)$ becomes partially unfeasible, i.e. $\underline{\tau}_d^{HT}(t)$ lies partially outside $\underline{\Phi}^{HT}$. The FDAS uses the S -approximation to approximate unfeasible solutions in these cases, which leads to approximation $\underline{\tau}^{HT}(t) = \underline{\tau}_s^{HT}(t)$ with the same direction as unattainable command input $\underline{\tau}_d^{HT}(t)$, but lower magnitude. The unfeasible part of trajectory $\underline{\tau}_d^{HT}(t)$ is characterised by relationship $\underline{\tau}^{HT}(t) \neq \underline{\tau}_d^{HT}(t)$ and indicated as shadowed regions in time responses shown

in Figure 6.20 (b) (FALCON) and Figure 6.21 (b) (URIS). Pairs of thrusters 1HT & 2HT and 3HT & 4HT are equally actuated for straight-line motion of FALCON ($\underline{u}_1(t) = \underline{u}_2(t)$ and $\underline{u}_3(t) = \underline{u}_4(t)$, respectively). In the case of URIS, $\underline{u}_1(t) = \underline{\tau}_{xd}(t)$, but $\underline{u}_2(t) < \underline{u}_1(t)$ and thrusters 3HT and 4HT are actuated in an asymmetrical sense ($\underline{u}_3(t) = -\underline{u}_4(t)$), such that they produce the moment, which counteracts the moment produced by the difference in actuation of 1HT and 2HT . The velocity profile $v(t)$ follows the profile of $\underline{\tau}_x(t)$, which is now different than $\underline{\tau}_{xd}(t)$, and the vehicle moves slower in shadowed region than in the fault-free case. However, heading $\psi(t)$ is constant in both cases, but trajectories $(x_E(t), y_E(t))$ are slightly different. The first difference stems from the fact that the higher forward velocity in the fault-free case means that the vehicle covers a longer distance than in the faulty situation, i.e. $\overline{AB}_{\text{fault-free}} > \overline{AB}_{\text{"Jammed propeller" in } ^2HT}$. The second difference comes from the shapes of the trajectories $(x_E(t), y_E(t))$. In the first case, it is a perfect straight line, whereas, in the second case, although symmetrical pairs of thrusters have opposite spin direction coefficients, the vehicle is shifted from the straight line, such that the drift at the end point B is approximately 2 cm for FALCON and 0.4 cm for URIS. The drift is caused by unequal actuation of symmetrical thrusters in the second case, which leads to non-zero angular moments from these pairs, resulting in non-perfect straight-line motion of the vehicle (see test case (A8) for more information about propeller spin direction).

This test case demonstrates that, in the presence of a single partial fault in 2HT and its limited usage of 75%, the faulty vehicle is able to perform straight-line motion with satisfactory performance. An unavoidable effect is the drop in forward velocity for cases when the command input vector $\underline{\tau}_d^{HT}(t)$ lies outside modified Φ^{HT} .

(B2) Partial fault - "Heavy jammed propeller"

This test case is similar to case (B1), but in this case a faulty state "Heavy jammed propeller" in 2HT (partial fault, $s_2^{HT} = 0.50$) was considered. Since the usage of 2HT is now limited to 50%, $\underline{\Phi}_p^{HT}$ and $\underline{\Phi}^{HT}$ shrink more than in the previous case and the unfeasible part of trajectory $\underline{\tau}_d^{HT}(t)$ is bigger, as indicated in Figure 6.22 (b) (FALCON) and Figure 6.23 (b) (URIS). In addition, the discrepancy between $\underline{\tau}_d^{HT}(t)$ and $\underline{\tau}^{HT}(t)$, as well as a drop in velocity $v(t)$, is greater than in the previous case. The faulty vehicle covers a shorter distance $(\overline{AB}^{Jammed\ propeller\ in\ {}^2HT} > \overline{AB}^{Heavy\ jammed\ propeller\ in\ {}^2HT})$ and the drift at the end point B is approximately 2.7 cm for FALCON and 0.8 cm for URIS. However, the heading $\psi(t)$ is constant for all time.

This test case shows that, in the presence of a single partial fault in 2HT and its limited usage of 50%, the vehicle is able to perform the straight-line motion with acceptable performance. The price paid is a drop in forward velocity for cases when the command input vector $\underline{\tau}_d^{HT}(t)$ lies outside modified $\underline{\Phi}^{HT}$.

(B3) Partial fault - "Unknown state"

In contrast to test cases (B1) and (B2), a faulty state "Unknown state" in 2HT (partial fault, $s_2^{HT} = 0.25$) was considered in this test case. Since the usage of 2HT is now limited to only 25%, $\underline{\Phi}_p^{HT}$ and $\underline{\Phi}^{HT}$ shrink even more than in previous cases and the unfeasible part of trajectory $\underline{\tau}_d^{HT}(t)$ is even bigger, as indicated in Figure 6.24 (b) (FALCON) and Figure 6.24 (b) (URIS). In addition, disagreement between $\underline{\tau}_d^{HT}(t)$ and $\underline{\tau}^{HT}(t)$, as well as the drop in velocity $v(t)$, is greater than in previous cases. The faulty vehicle covers a shorter distance $(\overline{AB}^{Heavy\ jammed\ propeller\ in\ {}^2HT} > \overline{AB}^{Unknown\ state\ in\ {}^2HT})$ and the drift at the end

point B is approximately 2.6 cm for FALCON and 1.1 cm for URIS. However, the heading $\psi(t)$ is constant for all time, as in previous cases.

This test case shows that, despite the presence of a single partial fault in 2HT and its limited usage of only 25%, the vehicle, equipped with the FDAS, is able to accomplish the straight-line motion with acceptable performance. As in previous cases, the price paid is a drop in velocity for cases when the command input vector $\underline{\tau}_d^{HT}(t)$ lies outside modified $\underline{\Phi}^{HT}$.

(B4) Total fault - "Broken propeller"

A faulty state "Broken propeller" in 2HT (total fault, $s_2^{HT} = 0.00$) was considered in this test case. This is an extreme case, where 2HT is switched off and mission must be accomplished with three remaining horizontal thrusters. Recall from section 5.5.2 that $\underline{\Phi}_p^{HT}$ and $\underline{\Phi}^{HT}$ have the same shape (small parallelepiped inside the virtual control space), as indicated in Figure 6.26 (b) (FALCON) and Figure 6.27 (b) (URIS). The volume of $\underline{\Phi}_p^{HT}$ and $\underline{\Phi}^{HT}$ is smaller than in test cases (B1) – (B3), which yields the widest shadowed region, i.e. the biggest unfeasible part of trajectory $\underline{\tau}_d^{HT}(t)$. In addition, disagreement between $\underline{\tau}_d^{HT}(t)$ and $\underline{\tau}^{HT}(t)$, as well as a drop in velocity $v(t)$, is largest for this test case. The faulty vehicle covers the shortest distance ($\overline{AB}^{\text{"Unknown state" in } {}^2HT} > \overline{AB}^{\text{"Broken propeller" in } {}^2HT}$) and the drift at the end point B is approximately 2.4 cm for FALCON and 1.3 cm for URIS. Nevertheless, again the heading $\psi(t)$ is constant for all time.

This test case reveals an important feature of the FDAS; that is, despite the presence of a total fault in 2HT , which is switched off, the vehicle, equipped with the FDAS, is able to continue the straight-line motion and complete the mission with acceptable performance.

As in previous cases, the unavoidable consequence is drop in velocity for cases when the command input vector $\underline{\mathbf{x}}_d^{HT}(t)$ lies outside modified Φ^{HT} .

(B5) Consecutive faults – passing through the pipe

The performance of two different control architectures are compared in this test case for the straight-line motion of the vehicle in the presence of consecutive faults in the same single thruster. The first architecture, denoted as the "FDAS active", uses the FDAS for the thruster fault accommodation. The second architecture, based on the existing control software for FALCON and URIS and denoted as the "FDAS not active", does not provide method for thruster fault accommodation, and the only available solution in faulty situation is to switch off a faulty thruster. The same command input vector $\underline{\mathbf{x}}_d^{HT}(t)$, created using the "From File" Simulink block, was used to drive the vehicle in both cases. The mission objective was to move the vehicle from the start point A through the pipe along the straight line. Different faulty states in 2HT ("Jammed propeller", "Heavy jammed propeller", "Unknown state" and "Broken propeller") were injected during simulation at time instances $t = 5s, 15s, 25s$ and $35s$, respectively. Figure 6.28 (a) (FALCON) and Figure 6.29 (a) (URIS) display time diagrams of $\underline{\mathbf{x}}_d^{HT}(t)$, $\underline{\mathbf{x}}^{HT}(t)$, $\underline{\mathbf{u}}^{HT}(t)$, $\underline{\mathbf{I}}_i(t)$, $J_{\mathbf{w}_v}^{HT}(t)$, $J_n^{HT}(t)$, $v(t)$ and $\psi(t)$, and the plan view of the trajectory $(x_E(t), y_E(t))$ for the first architecture. It can be seen that, despite the presence of consecutive faults in 2HT , the vehicle keeps a constant heading, with negligible drift from a straight line. The unfeasible parts of trajectory $\underline{\mathbf{x}}_d^{HT}(t)$ are denoted as shadowed regions, in which there is a drop in the forward velocity of the vehicle. The situation is different for the second architecture, as indicated in Figure 6.28 (b) (FALCON) and Figure 6.29 (b) (URIS). In particular, disabling of the faulty thruster, without appropriate reconfiguration, introduces

unbalanced moment components, which cause undesired rotation and the vehicle performs circular motion "passing" through the wall of the pipe. In real applications this unacceptable behaviour could cause damage and even loss of the vehicle. However, the FDAS in the first architecture provides the optimal redistribution of propulsion forces among three remaining thrusters and compensates unbalanced moment components, such that the faulty vehicle performs pure translational, straight-line motion.

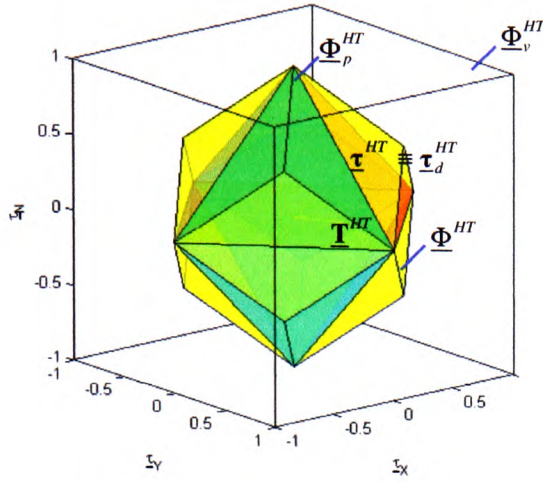
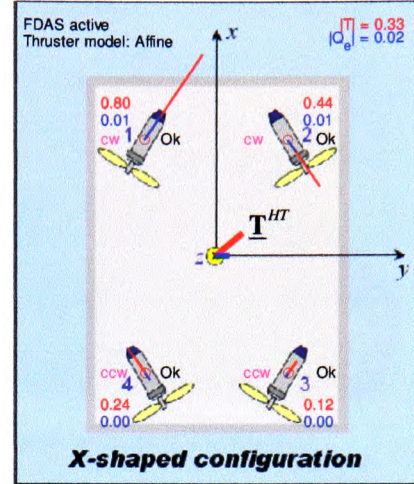
This test case shows that the performance of the control architecture "FDAS active" for the straight-line motion is superior compared to the control architecture "FDAS not active" in the presence of consecutive faults in the same single thruster during the mission.

(B6) Consecutive faults – passing through the hole in the rock

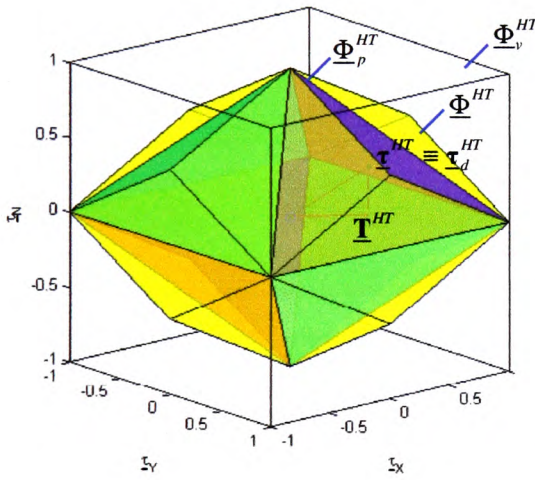
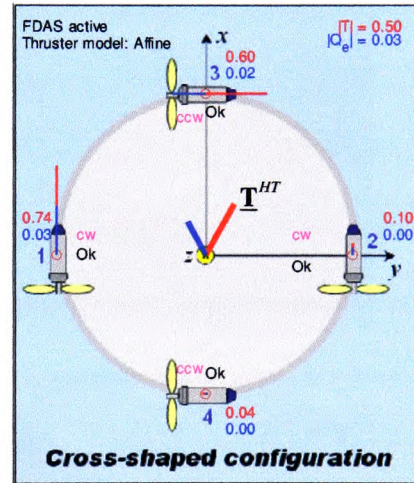
As in the previous case, the performance of two different control architectures "FDAS active" and "FDAS not active" are compared in this test case for the straight-line motion of the vehicle, but this time in the presence of consecutive faults in different thrusters during the mission. The same command input vector $\mathbf{\tau}_d^{HT}(t)$, created using the "From File" Simulink block, was used to drive the vehicle in both cases. The mission objective was to move the vehicle from the start point A through the hole in the rock along the straight line. Different faulty states ("Jammed propeller" in 1HT , "Heavy jammed propeller" in 2HT , "Unknown state" in 3HT and "Broken propeller" in 4HT) were injected during simulation at time instances $t = 5s, 15s, 25s$ and $35s$, respectively. Figure 6.30 (a) (FALCON) and Figure 6.31 (a) (URIS) display time diagrams of $\mathbf{\tau}_d^{HT}(t)$, $\mathbf{\tau}^{HT}(t)$, $\mathbf{u}^{HT}(t)$, $\mathbf{T}_i(t)$, $J_{\mathbf{w}_e}^{HT}(t)$, $J_n^{HT}(t)$, $v(t)$ and $\psi(t)$, and the plan view of the trajectory $(x_E(t), y_E(t))$ for the architecture "FDAS active". It can be seen that, despite the presence of consecutive faults in different thrusters, the vehicle keeps the constant heading, with

negligible drift from a straight line. The unfeasible parts of trajectory $\underline{\tau}_d^{HT}(t)$ are denoted as shadowed regions, in which there is a drop in the forward velocity of the vehicle compared to fault-free case. The situation is very different for the architecture "FDAS not active", as depicted in Figure 6.30 (b) (FALCON) and Figure 6.31 (b) (URIS). Unbalanced moment components, introduced by disabling of faulty thrusters without reconfiguration, produce undesired rotation and the vehicle performs irregular motion "passing" through the wall of the rock. As in the previous case, this behaviour is totally unacceptable and could cause damage of the vehicle in real applications. However, the FDAS compensates for unbalanced moment components in the first architecture in an optimal way, such that the faulty vehicle performs pure translational, straight-line motion. One important distinction between X-shaped and cross-shaped thruster configuration can be observed from diagrams in Figure 6.30 (a) (FALCON) and Figure 6.31 (a) (URIS). For the X-shaped configuration (FALCON) all four horizontal thrusters contribute to the straight-line motion, and a fault in any one of them activates reallocation inside the FDAS and contraction of $\underline{\Phi}^{HT}$ in $\underline{\tau}_x$ direction, leading to the loss of the forward velocity for cases when $\underline{\tau}_{Xd}$ is too high. In contrast, for the cross-shaped configuration (URIS) only thrusters 1HT and 2HT are actuated for the straight-line motion, and faults in thrusters 3HT and 4HT are irrelevant in this case, i.e. the control vector for the straight-line motion is invariant to faults in 3HT and 4HT .

This test case demonstrates that the performance of the control architecture "FDAS active" for the straight-line motion is superior compared to the control architecture "FDAS not active" in the presence of consecutive faults in different horizontal thrusters during the mission.

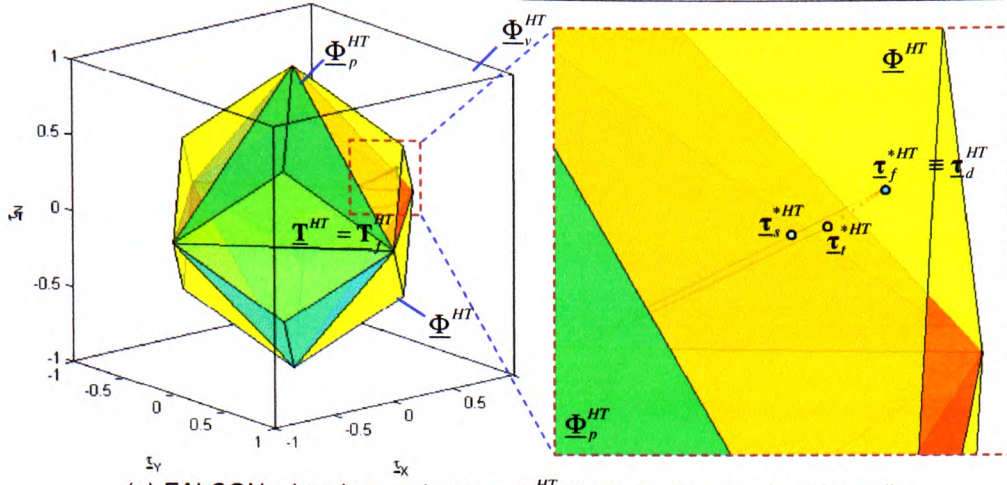
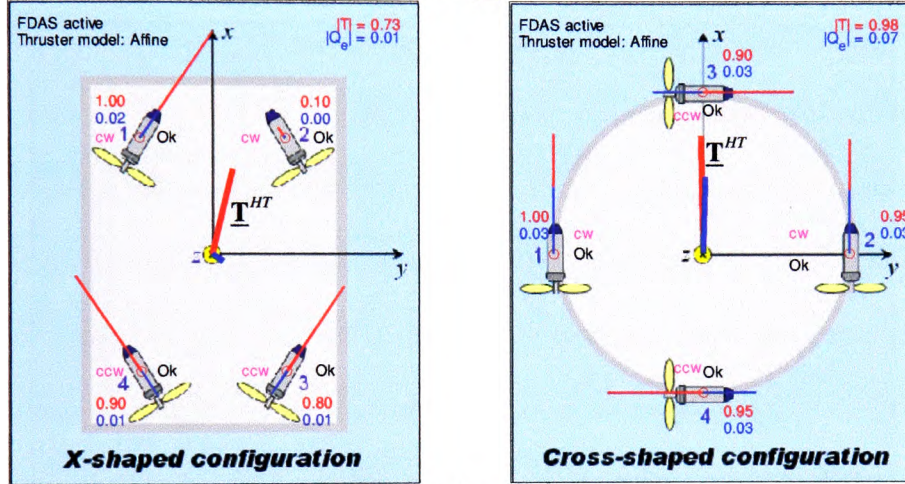
(a) FALCON: virtual control space Φ_v^{HT} .

(b) Force distribution (Horizontal thrusters).

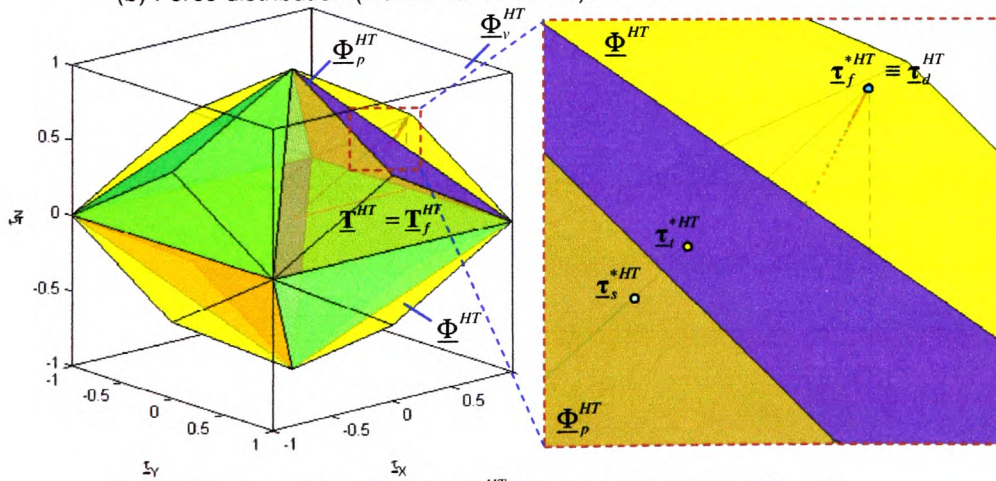
(c) URIS: virtual control space Φ_v^{HT} .

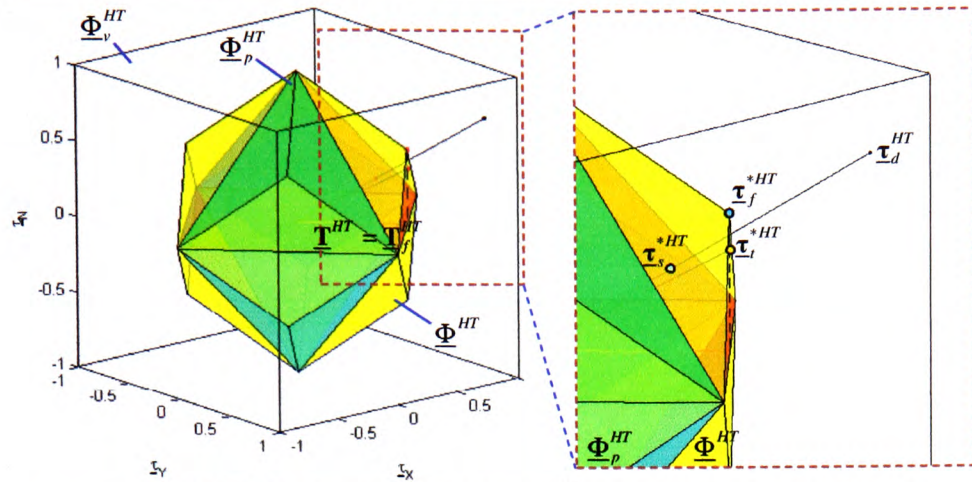
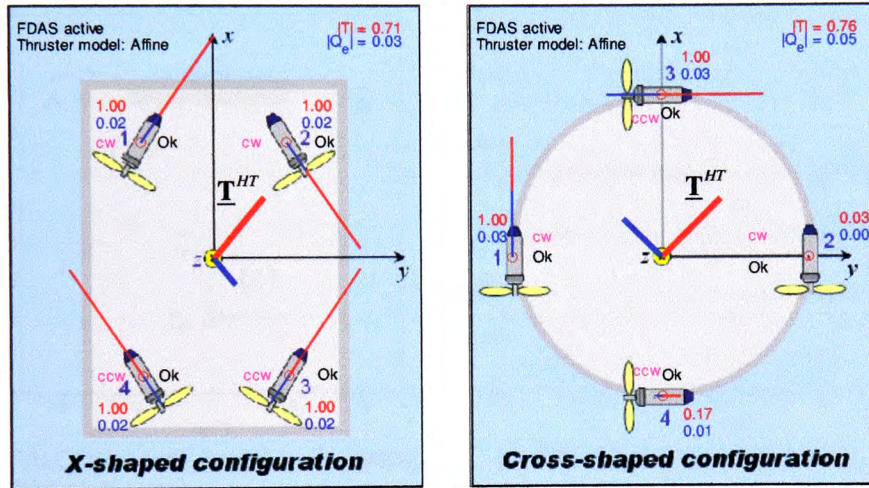
(d) Force distribution (Horizontal thrusters).

Figure 6.4 (A1) Feasible pseudoinverse solution $(\tau_d^{HT} \in \Phi_p^{HT} \Rightarrow \underline{u}^{HT} \in \underline{\Omega}^{HT})$.

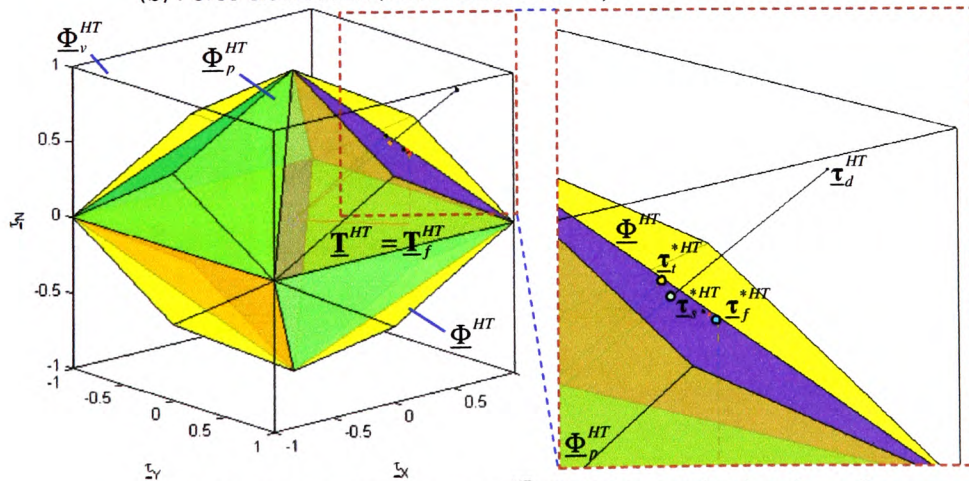
(a) FALCON: virtual control space Φ_v^{HT} (highlighted region is enlarged).

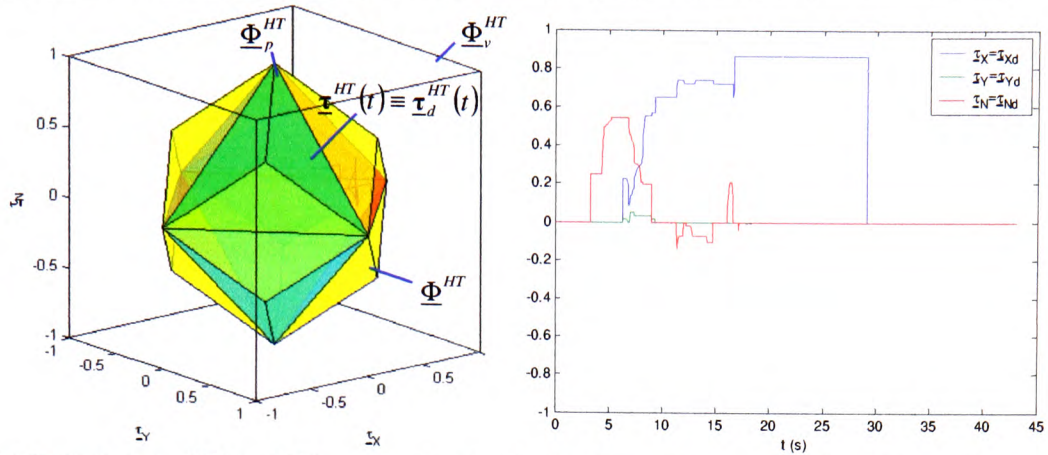
(b) Force distribution (Horizontal thrusters).

(c) URIS: virtual control space Φ_v^{HT} (highlighted region is enlarged).Figure 6.5 (A2) Unfeasible pseudoinverse solution ($\tau_d^{HT} \notin \Phi_p^{HT} \Rightarrow \underline{u}^{HT} \notin \underline{\Omega}^{HT}$).

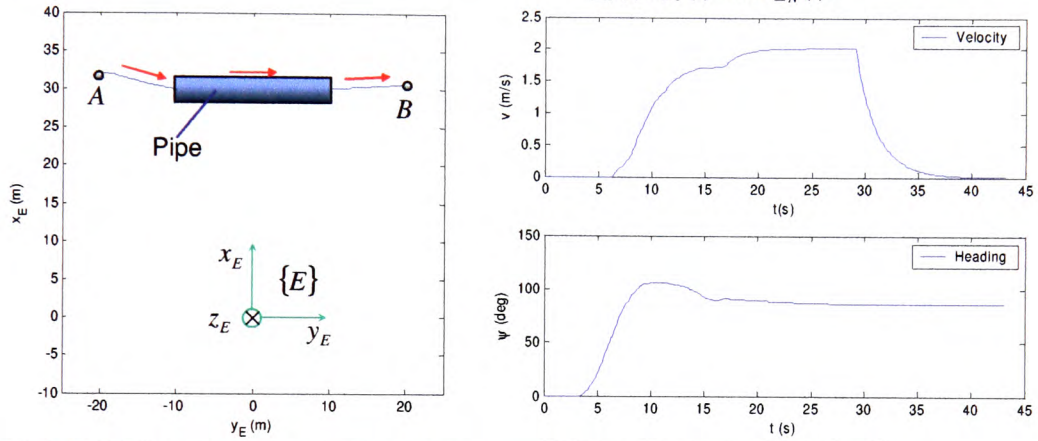
(a) FALCON: virtual control space Φ_v^{HT} (highlighted region is enlarged).

(b) Force distribution (Horizontal thrusters).

(c) URIS: virtual control space Φ_v^{HT} (highlighted region is enlarged).Figure 6.6 (A3) Approximation of an unattainable command input ($\tau_d^{HT} \notin \Phi_v^{HT}$).

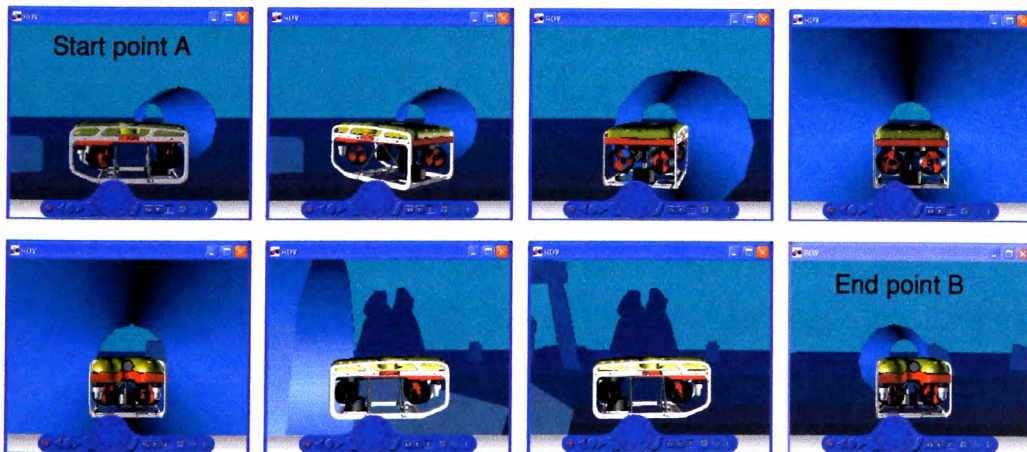


(a) Trajectories $\tau_d^{HT}(t)$ & $\tau^{HT}(t) \in \Phi_p^{HT}$ coincide $\forall t$. (b) Time diagrams $\tau_{xd}(t)$, $\tau_{yd}(t)$, $\tau_{zd}(t)$, $\tau_x(t)$, $\tau_y(t)$ and $\tau_z(t)$.



(c) FALCON moves from A to B, passing through the pipe.

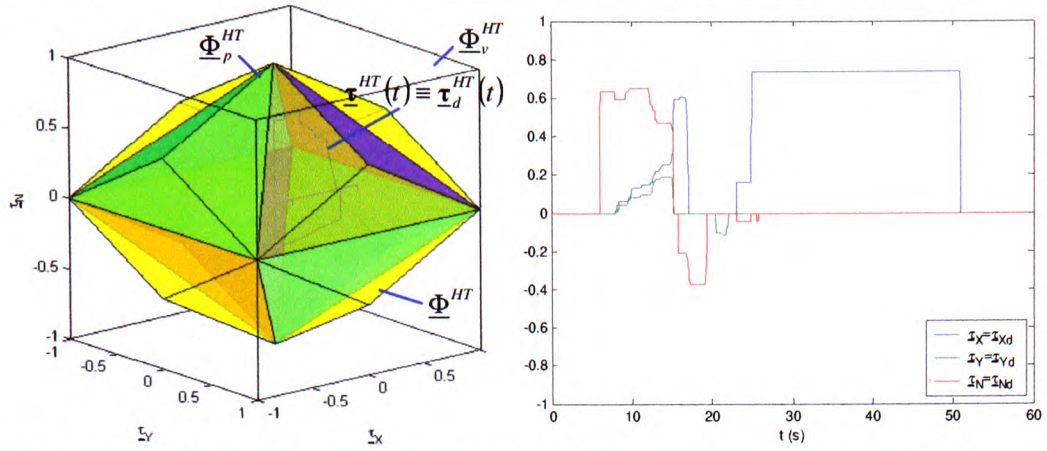
(d) Time diagrams $v(t)$ and $\psi(t)$.



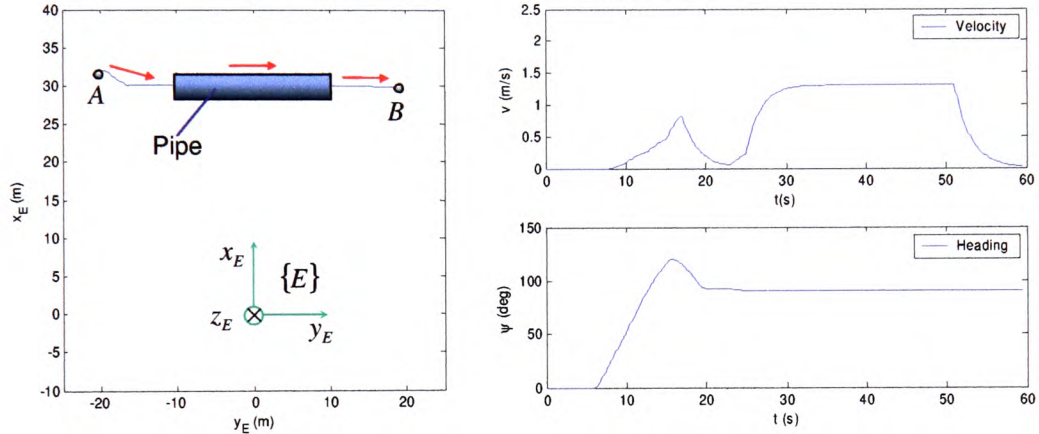
(e) Visualisation of the mission in a virtual world.

Figure 6.7 (A4) Feasible trajectory – FALCON ($\tau_d^{HT}(t) \in \Phi_p^{HT}$).



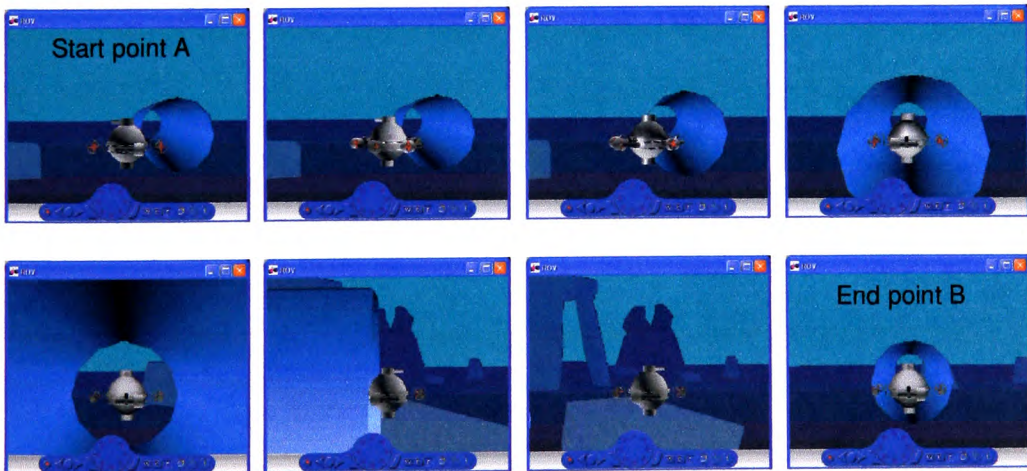


(a) Trajectories $\underline{\tau}_d^{HT}(t)$ & $\underline{\tau}^{HT}(t) \subset \Phi_p^{HT}$ coincide $\forall t$. (b) Time diagrams $\tau_{Xd}(t)$, $\tau_{Yd}(t)$, $\tau_{Nd}(t)$, $\tau_x(t)$, $\tau_y(t)$ and $\tau_N(t)$.



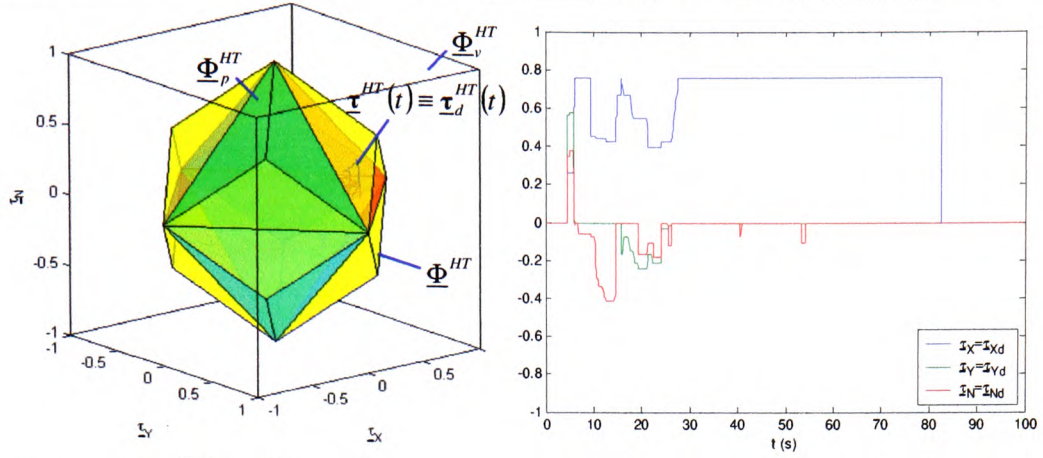
(c) URIS moves from A to B, passing through the pipe.

(d) Time diagrams $v(t)$ and $\psi(t)$.

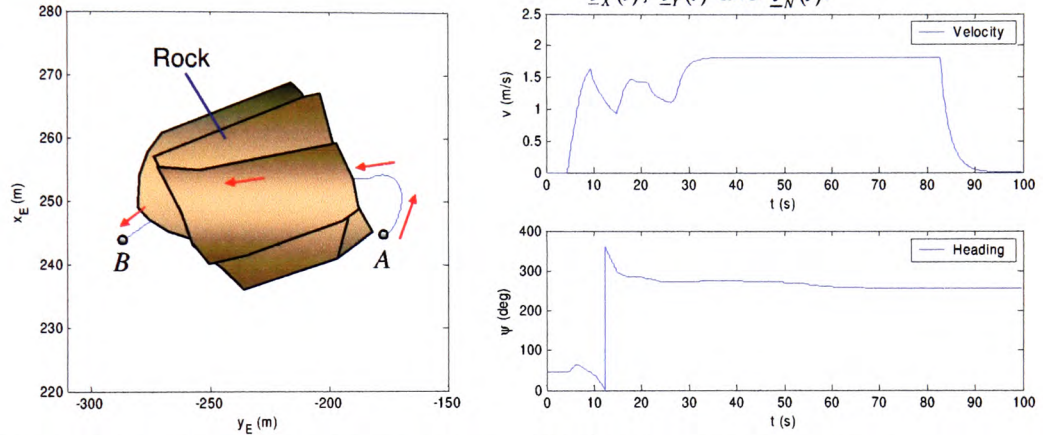


(e) Visualisation of the mission in a virtual world.

Figure 6.8 (A4) Feasible trajectory – URIS $(\underline{\tau}_d^{HT}(t) \subset \Phi_p^{HT})$.

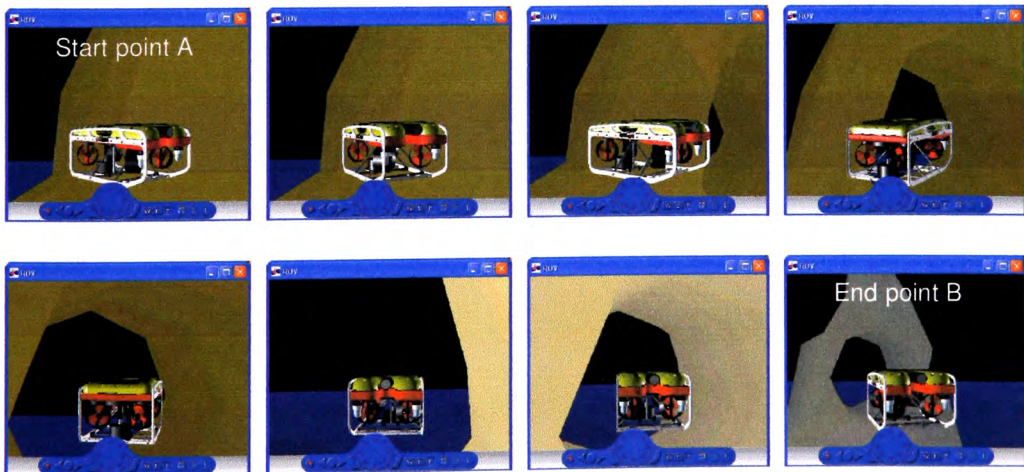


(a) Trajectories $\underline{\tau}_d^{HT}(t) \& \underline{\tau}^{HT}(t) \subset \Phi^{HT}$ coincide $\forall t$. (b) Time diagrams $\underline{x}_d(t)$, $\underline{y}_d(t)$, $\underline{z}_d(t)$, $\underline{x}(t)$, $\underline{y}(t)$ and $\underline{z}(t)$.



(c) FALCON moves from A to B, passing through the hole in the rock.

(d) Time diagrams $v(t)$ and $\psi(t)$.



(e) Visualisation of the mission in a virtual world.

Figure 6.9 (A5) Feasible trajectory – FALCON $(\underline{\tau}_d^{HT}(t) \subset \Phi^{HT})$.

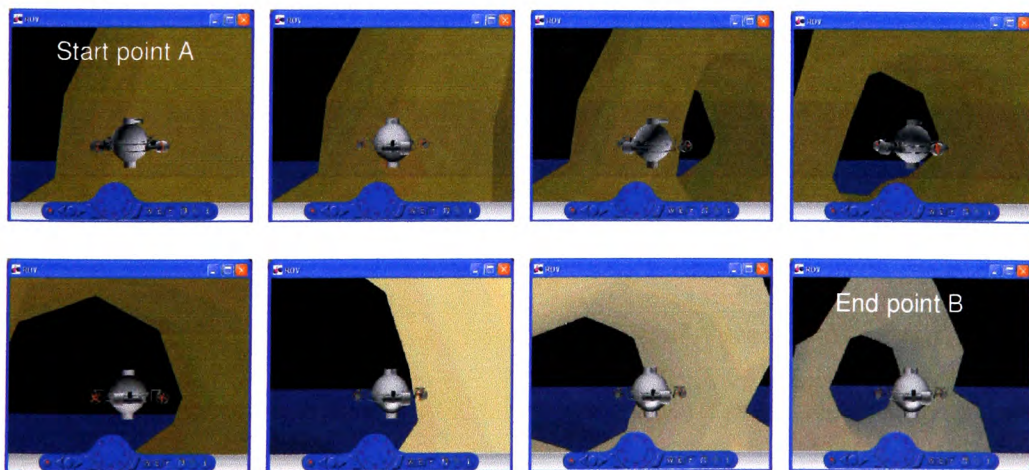
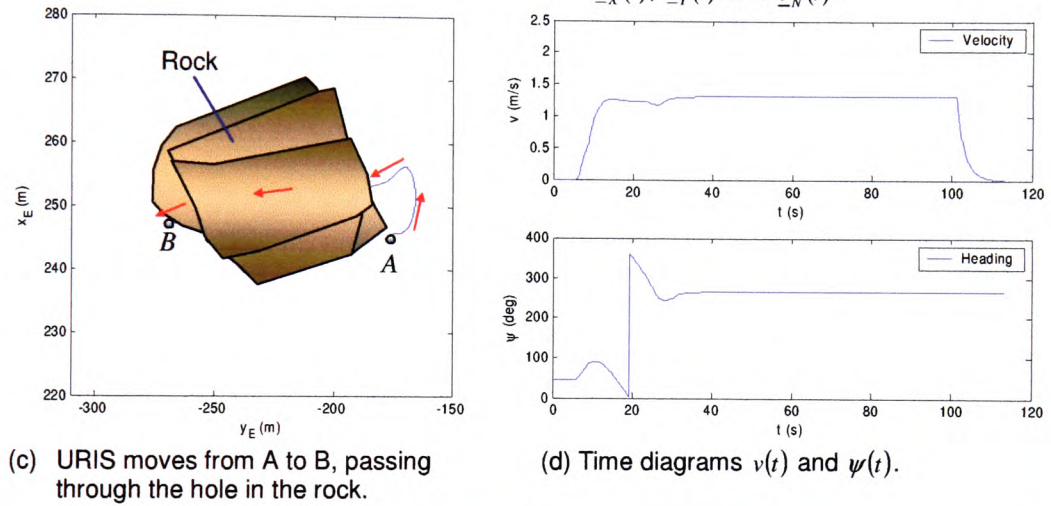
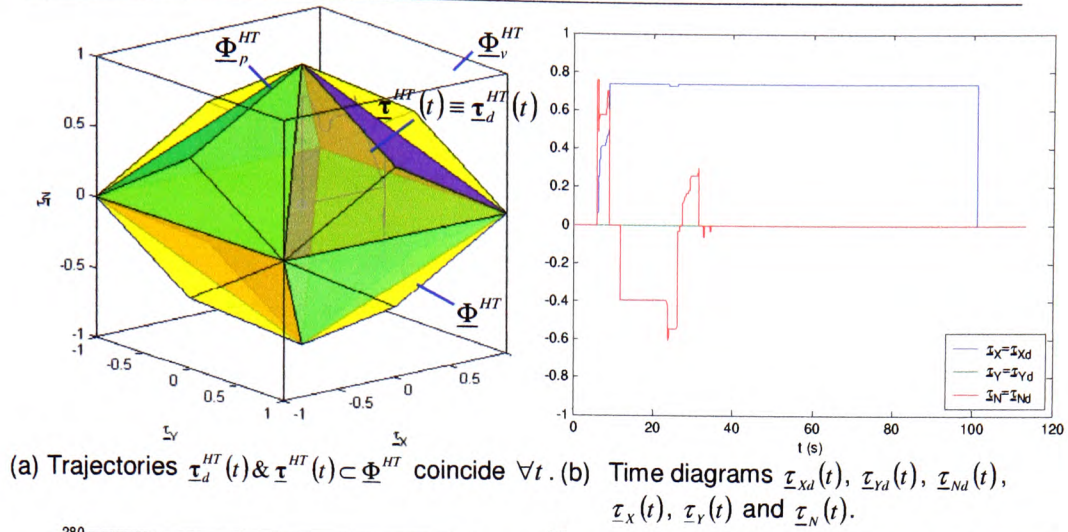
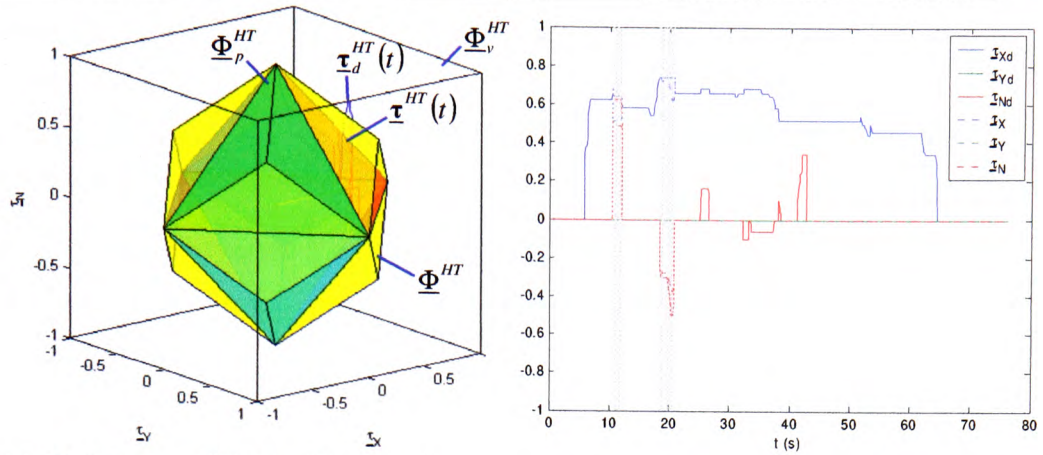
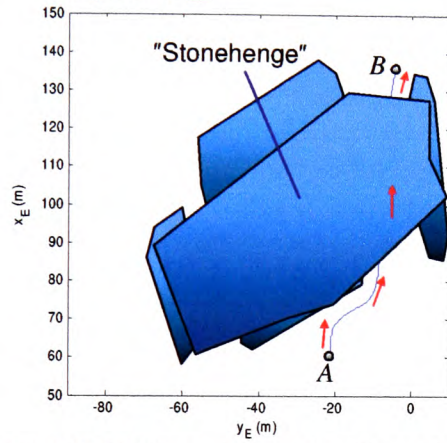


Figure 6.10 (A5) Feasible trajectory – URIS $\left(\underline{\tau}_d^{HT}(t) \in \Phi^{HT}\right)$.

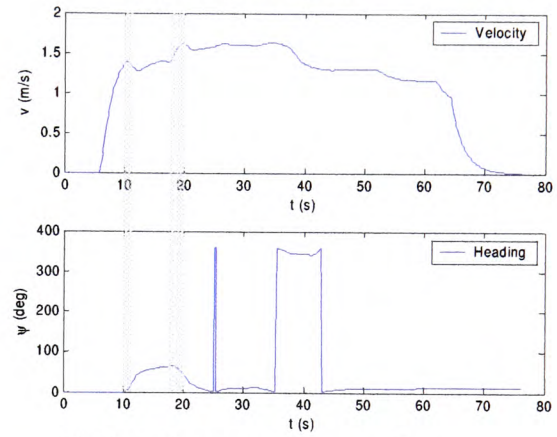


(a) Trajectories $\tau^{HT}(t)$ & $\tau_d^{HT}(t)$ coincide only inside Φ^{HT} .

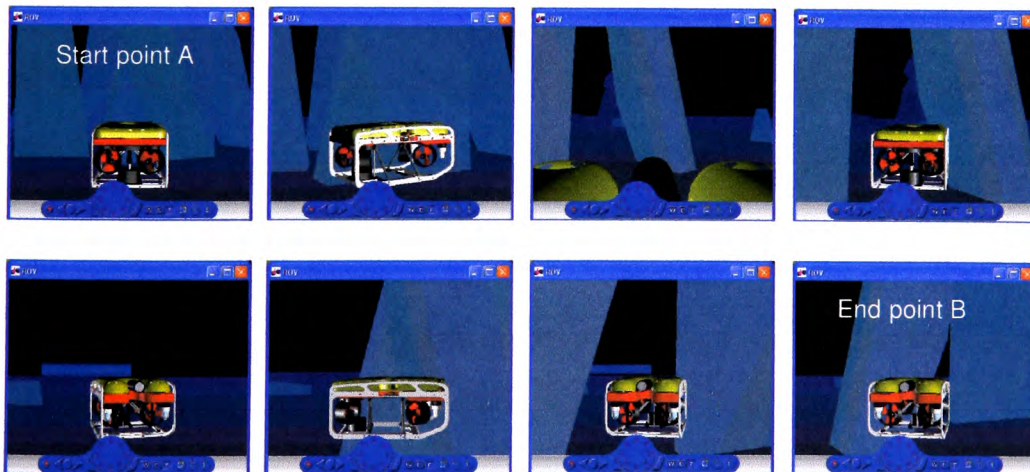
(b) Time diagrams $\tau_{Xd}(t)$, $\tau_{Yd}(t)$, $\tau_{Nd}(t)$, $\tau_X(t)$, $\tau_Y(t)$ and $\tau_N(t)$.



(c) FALCON moves from A to B, passing through the passage in "Stonehenge".



(d) Time diagrams $v(t)$ and $\psi(t)$.



(e) Visualisation of the mission in a virtual world.

Figure 6.11 (A6) Partially unfeasible trajectory – FALCON (parts of $\tau_d^{HT}(t)$ lie outside Φ^{HT}).

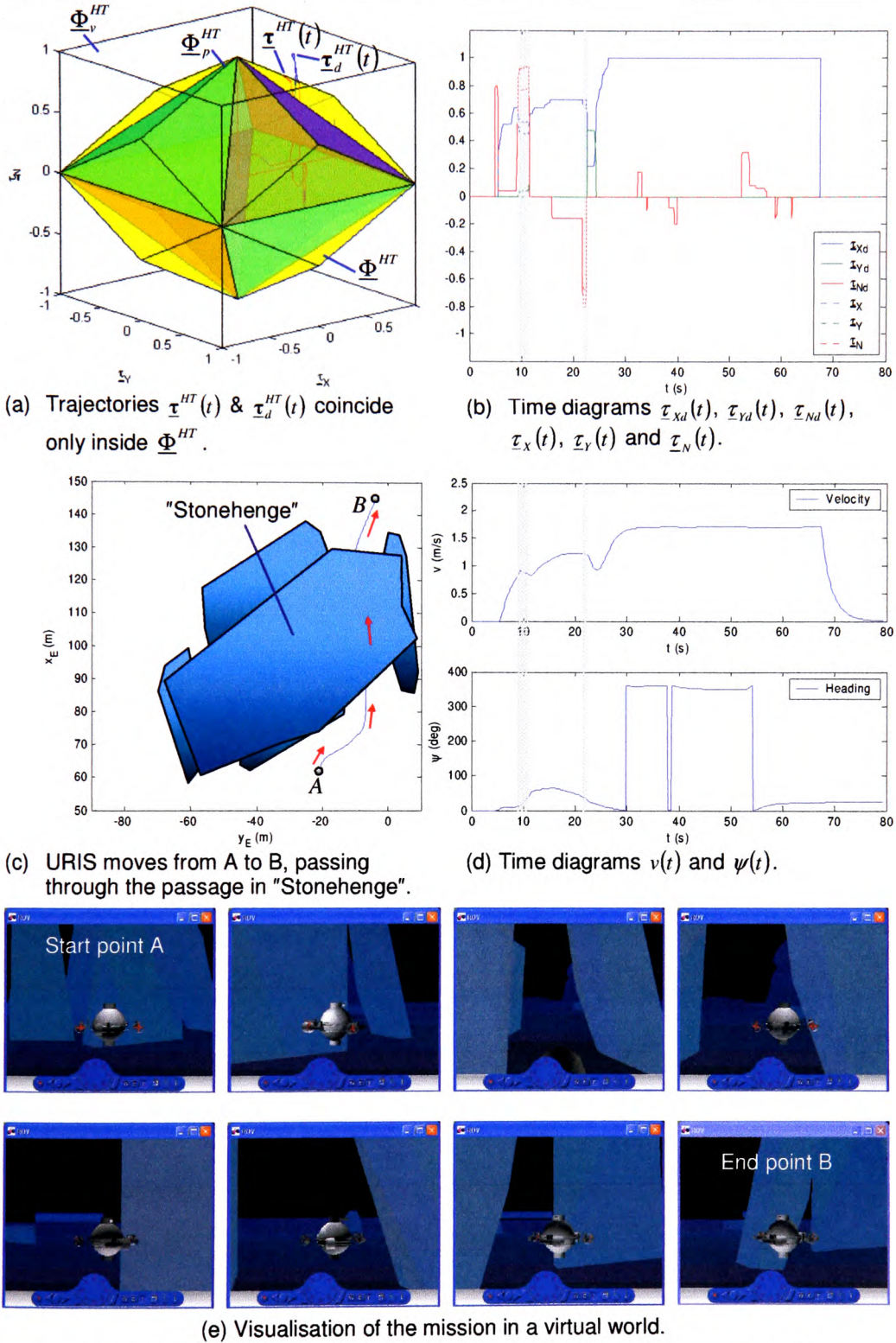
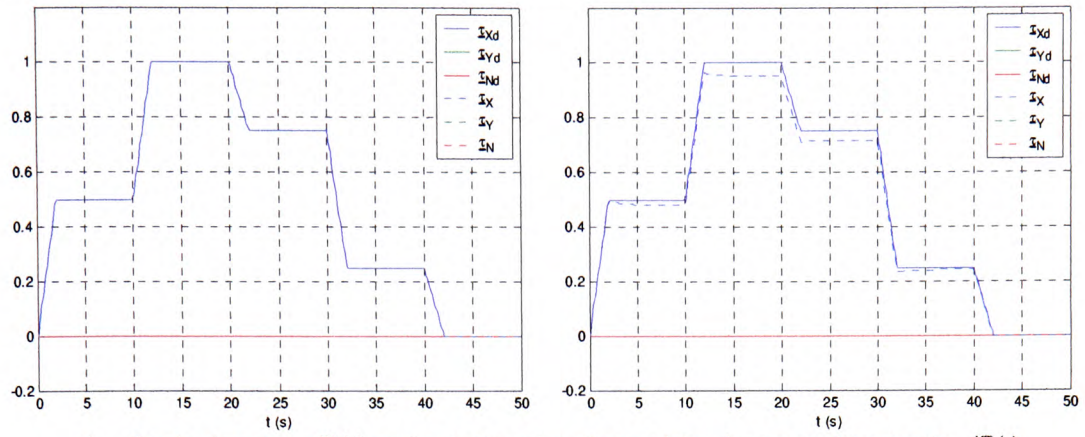
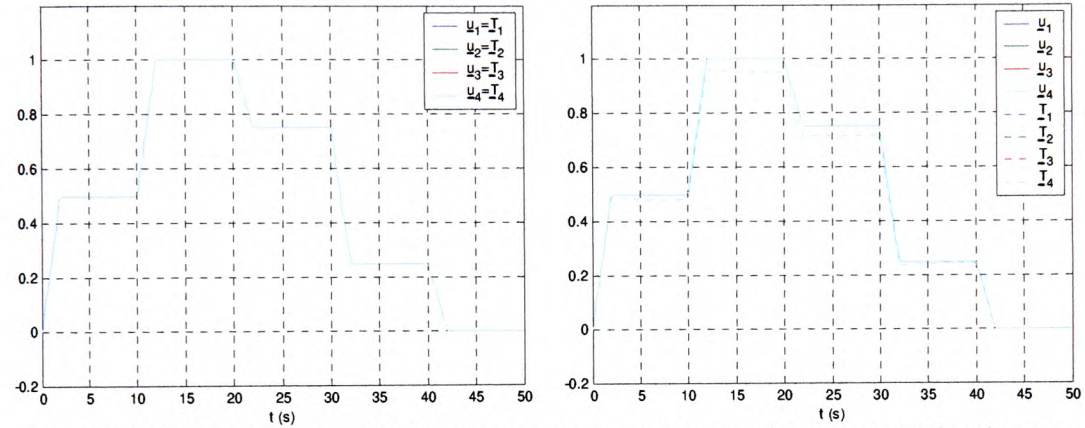


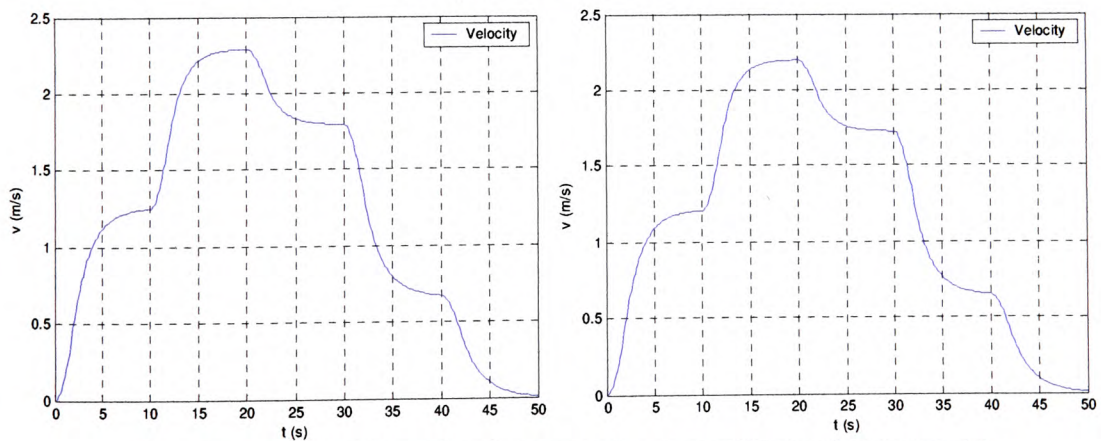
Figure 6.12 (A6) Partially unfeasible trajectory – URIS (parts of $\underline{\tau}_d^{HT}(t)$ lie outside Φ^{HT}).



1. Desired vector $\tau_d^{HT}(t)$ and actual vector of propulsion forces and moments $\tau^{HT}(t)$ coincide for affine thruster model, but they don't coincide for bilinear thruster model.



2. Control variable u_i and thruster force T_i are equal for affine thruster model, but they are not equal for bilinear thruster model.

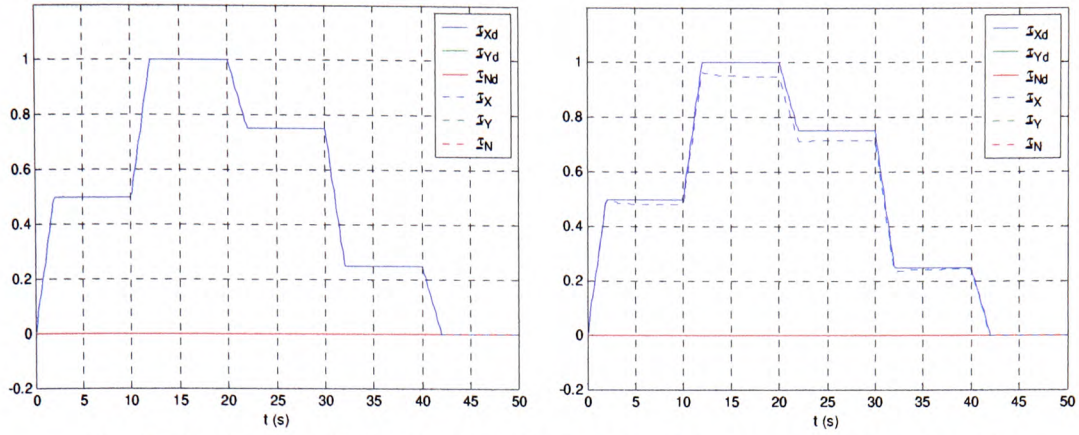


3. The velocity of the vehicle is lower in the case of bilinear thruster model.

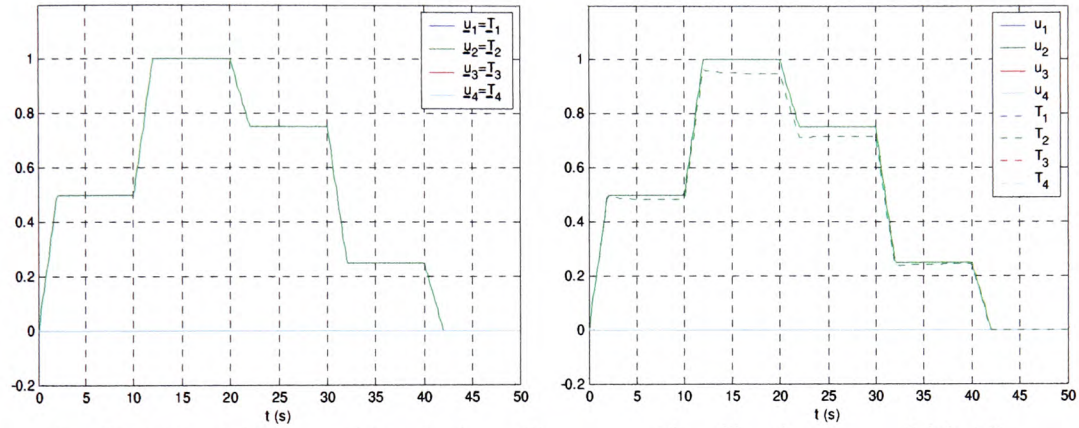
(a) Affine thruster model.

(b) Bilinear thruster model.

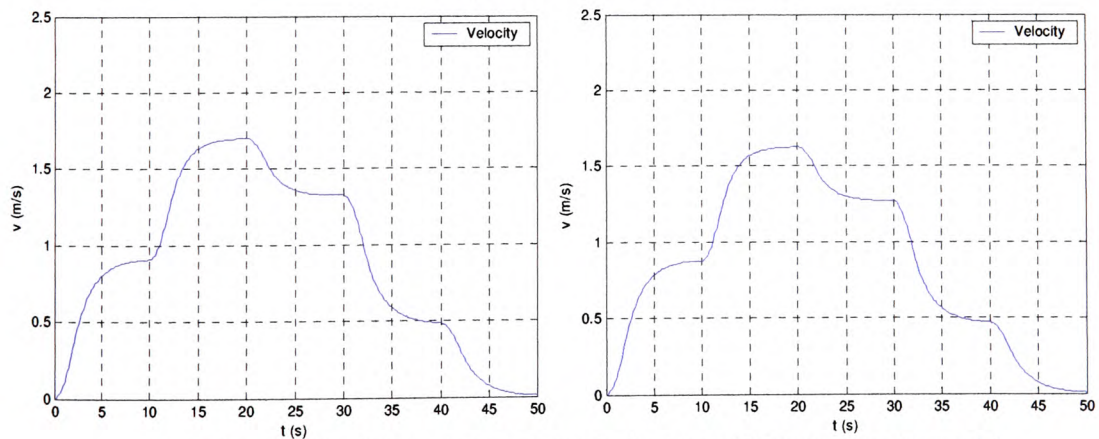
Figure 6.13 (A7) Difference between affine and bilinear thruster model (FALCON).



1. Desired vector $\underline{\tau}_d^{HT}(t)$ and actual vector of propulsion forces and moments $\underline{\tau}^{HT}(t)$ coincide for affine thruster model, but they don't coincide for bilinear thruster model.



2. Control variable \underline{u}_i and thruster force \underline{T}_i are equal for affine thruster model, but they are not equal for bilinear thruster model.

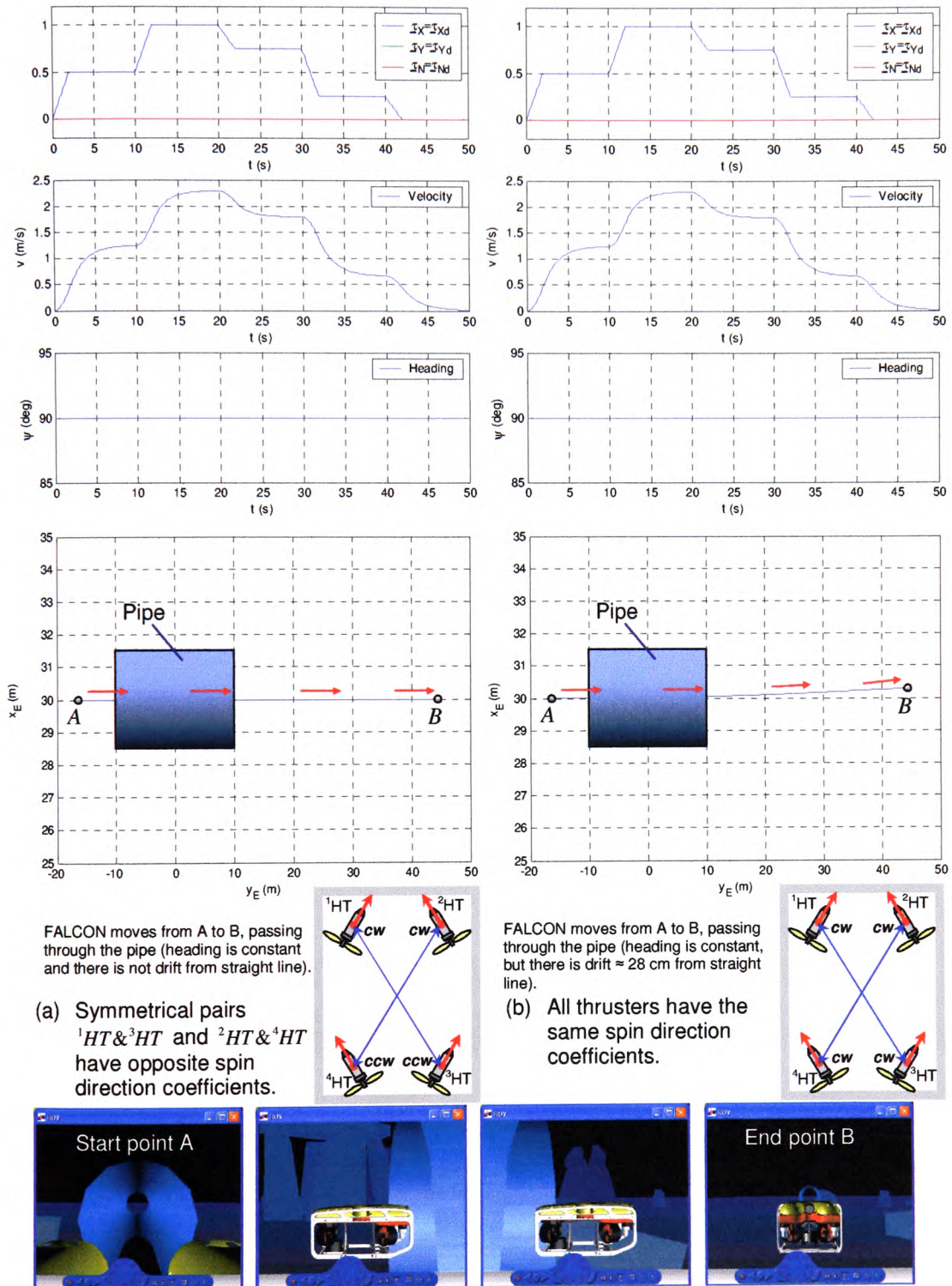


3. The velocity of the vehicle is lower in the case of bilinear thruster model.

(a) Affine thruster model.

(b) Bilinear thruster model.

Figure 6.14 (A7) Difference between affine and bilinear thruster model (URIS).



(c) Visualisation of the mission in a virtual world.

Figure 6.15 (A8) Choice of propeller spin direction (FALCON).

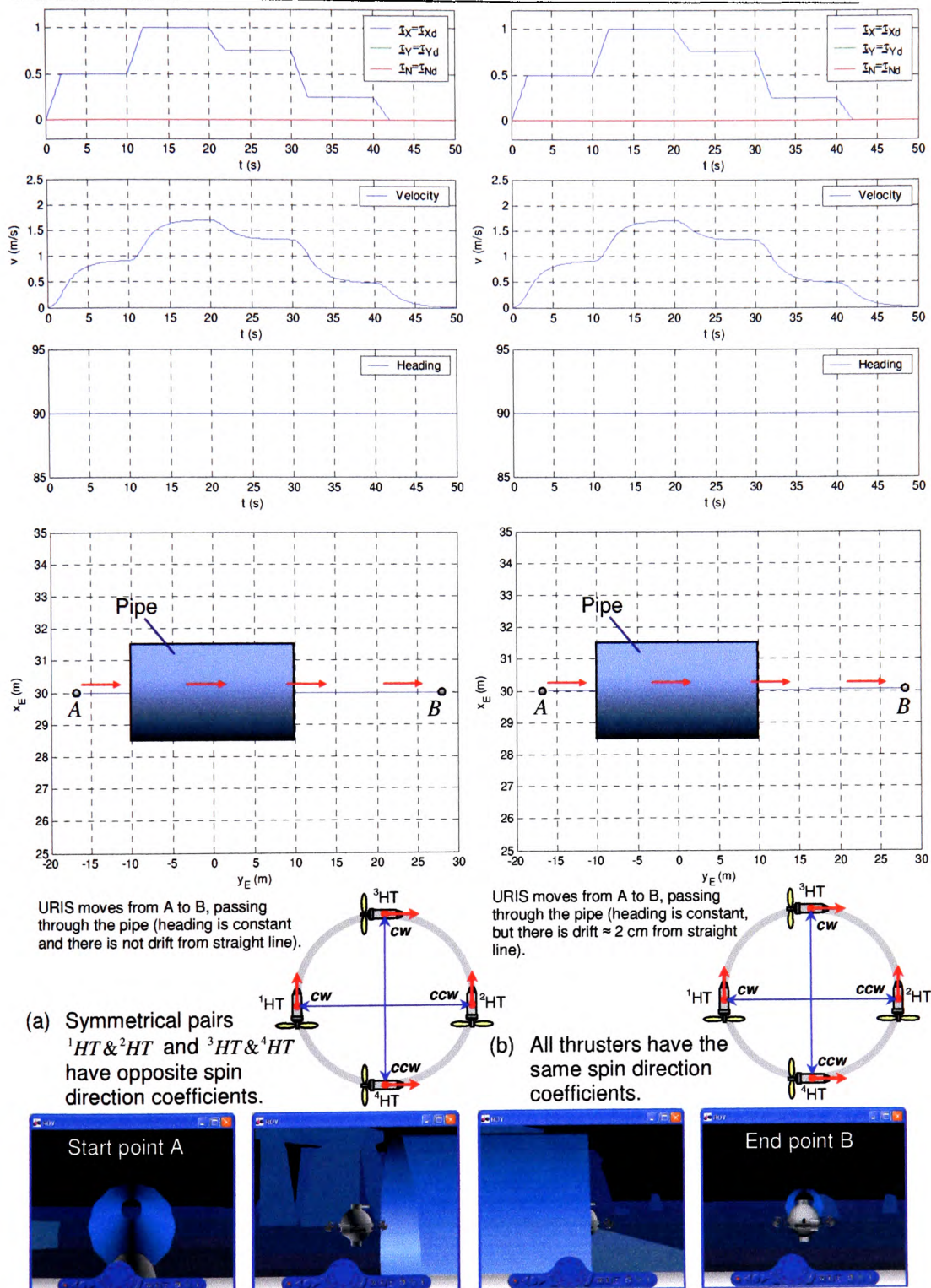
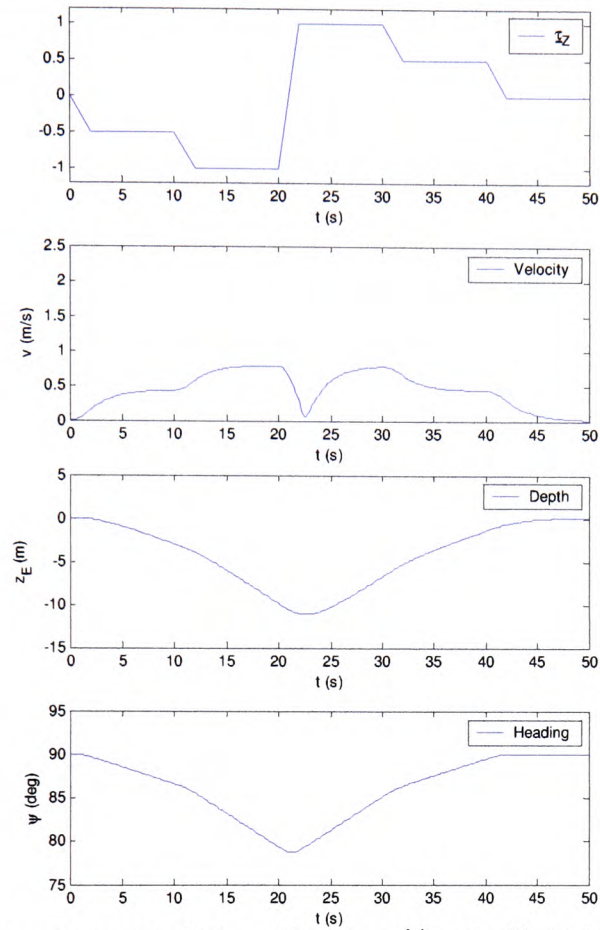
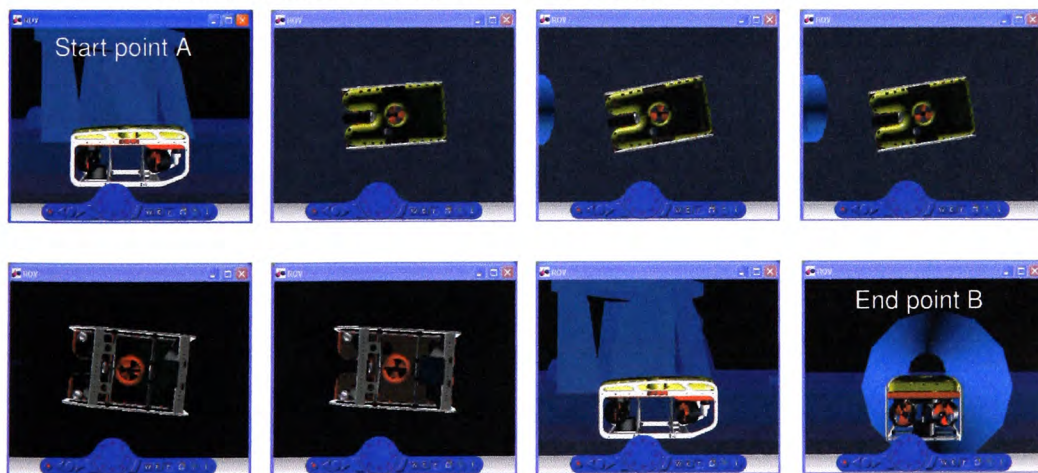


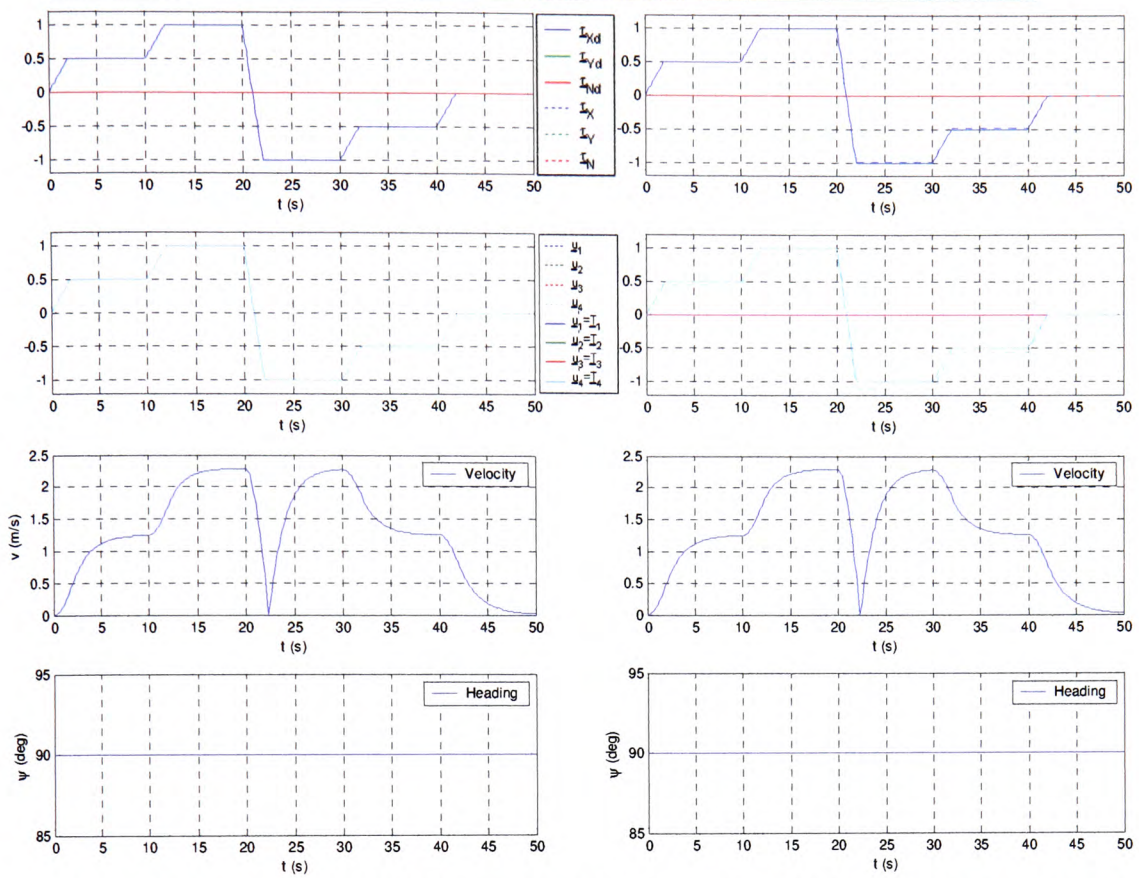
Figure 6.16 (A8) Choice of propeller spin direction (URIS).



(a) Time diagrams: $\underline{x}_d^{VT}(t) = \underline{x}_z(t)$, $v(t)$, $z_E(t)$ and $\psi(t)$. At the beginning, FALCON ascends and then descends. Torque Q_e , exerted by a vertical thruster, introduces undesired change of heading.



(b) Visualisation of the mission in a virtual world.
Figure 6.17 (A9) Motion in vertical plane (FALCON).



"Correction" block in the FDAS (Figure 5.9) transforms the auxiliary control variable \underline{u}_i into the real control variable \underline{u}_i , compensating for non-symmetrical T -curve. As result, $\underline{\tau}^{HT}(t) \equiv \underline{\tau}_d^{HT}(t)$ in both cases.

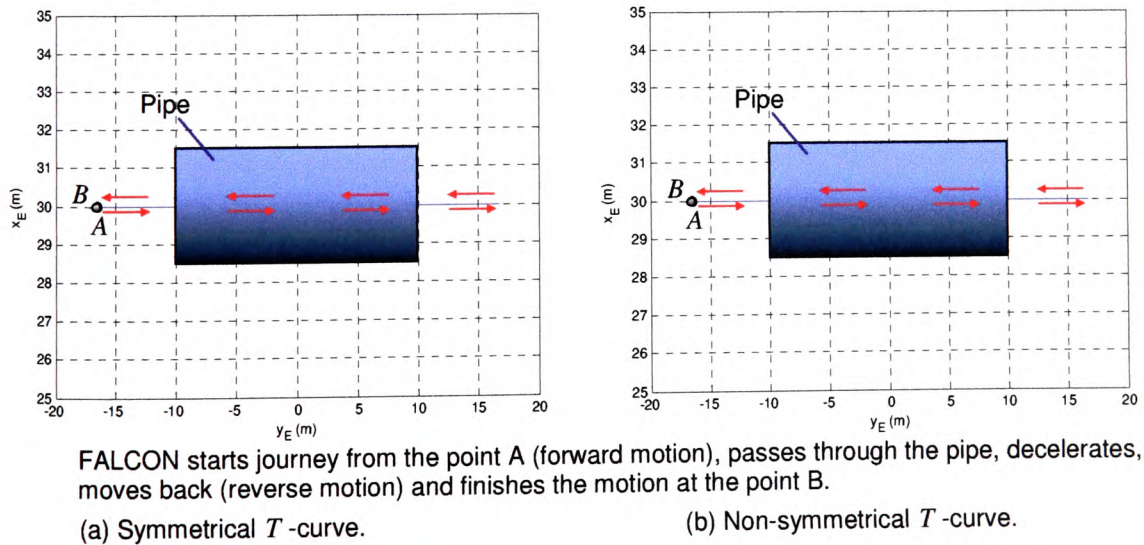
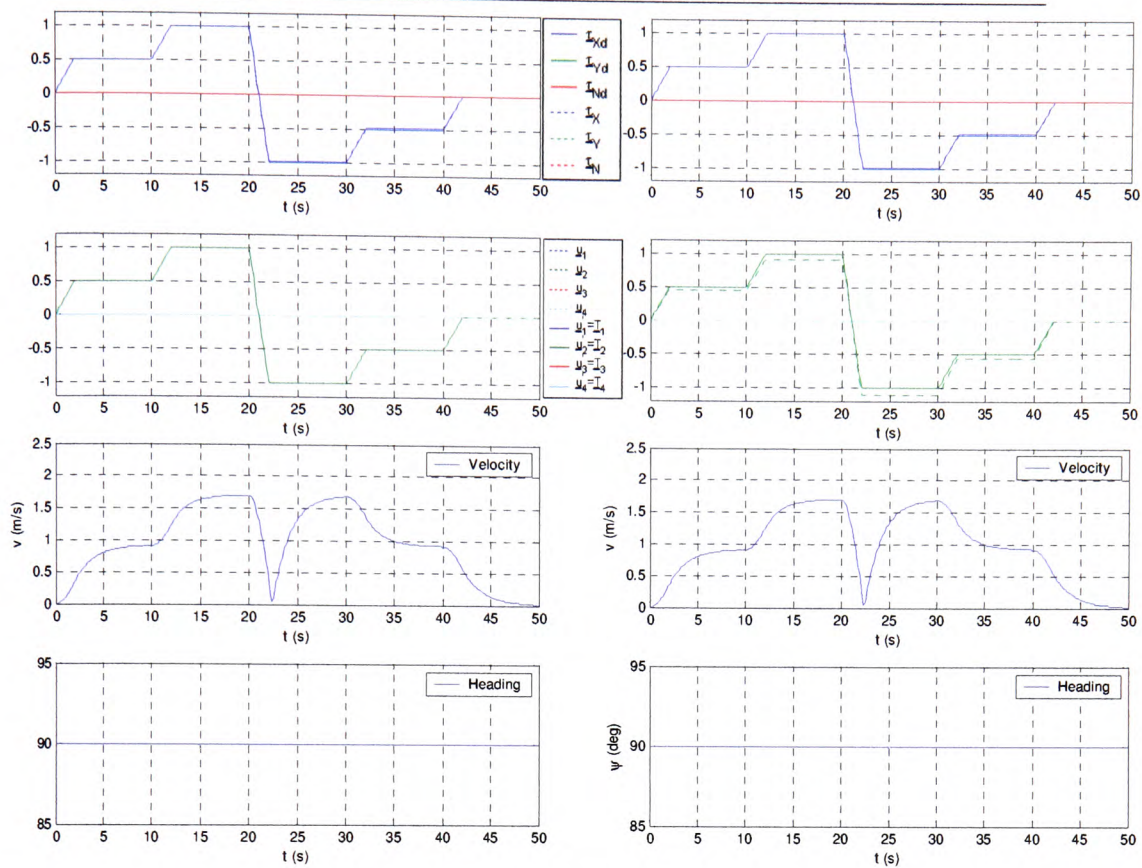
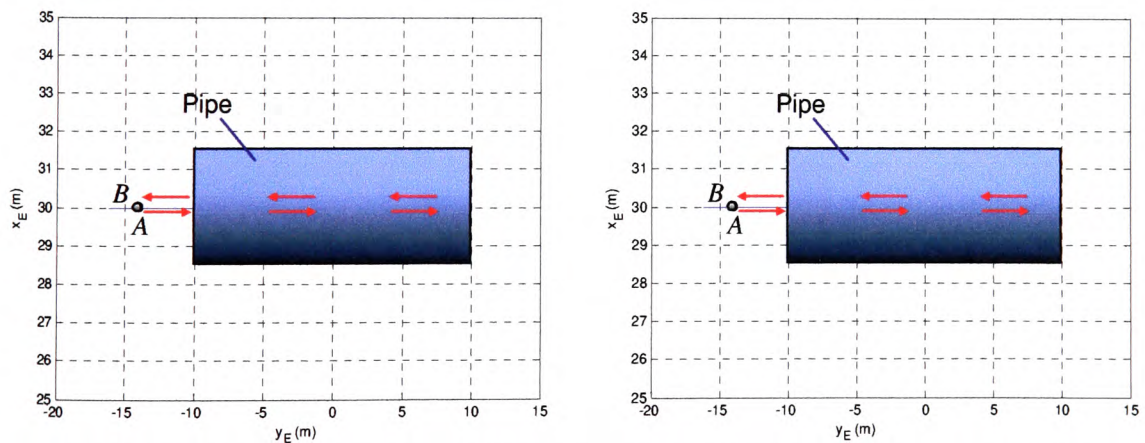


Figure 6.18 (A10) Difference between symmetrical and non-symmetrical T -curve (FALCON).



"Correction" block in the FDAS (Figure 5.9) transforms the auxiliary control variable \underline{u}'_i into the real control variable \underline{u}_i , compensating for non-symmetrical T -curve. As result, $\tau^{HT}(t) \equiv \tau_d^{HT}(t)$ in both cases.



URIS starts journey from the point A (forward motion), passes through the pipe, decelerates, moves back (reverse motion) and finishes the motion at the point B.

(a) Symmetrical T -curve.

(b) Non-symmetrical T -curve.

Figure 6.19 (A10) Difference between symmetrical and non-symmetrical T -curve (URIS).

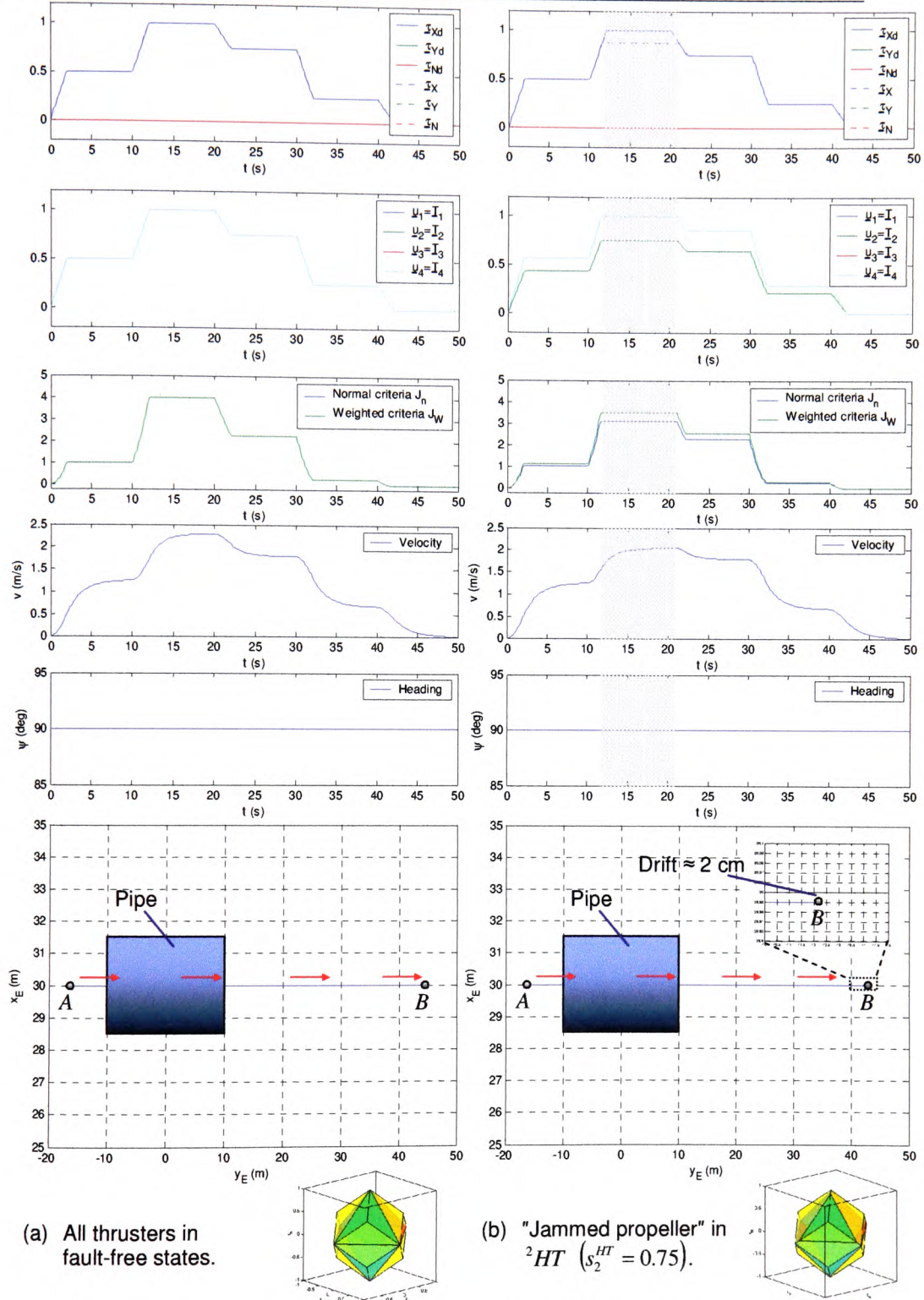


Figure 6.20 (B1) Partial fault - "Jammed propeller" (FALCON).

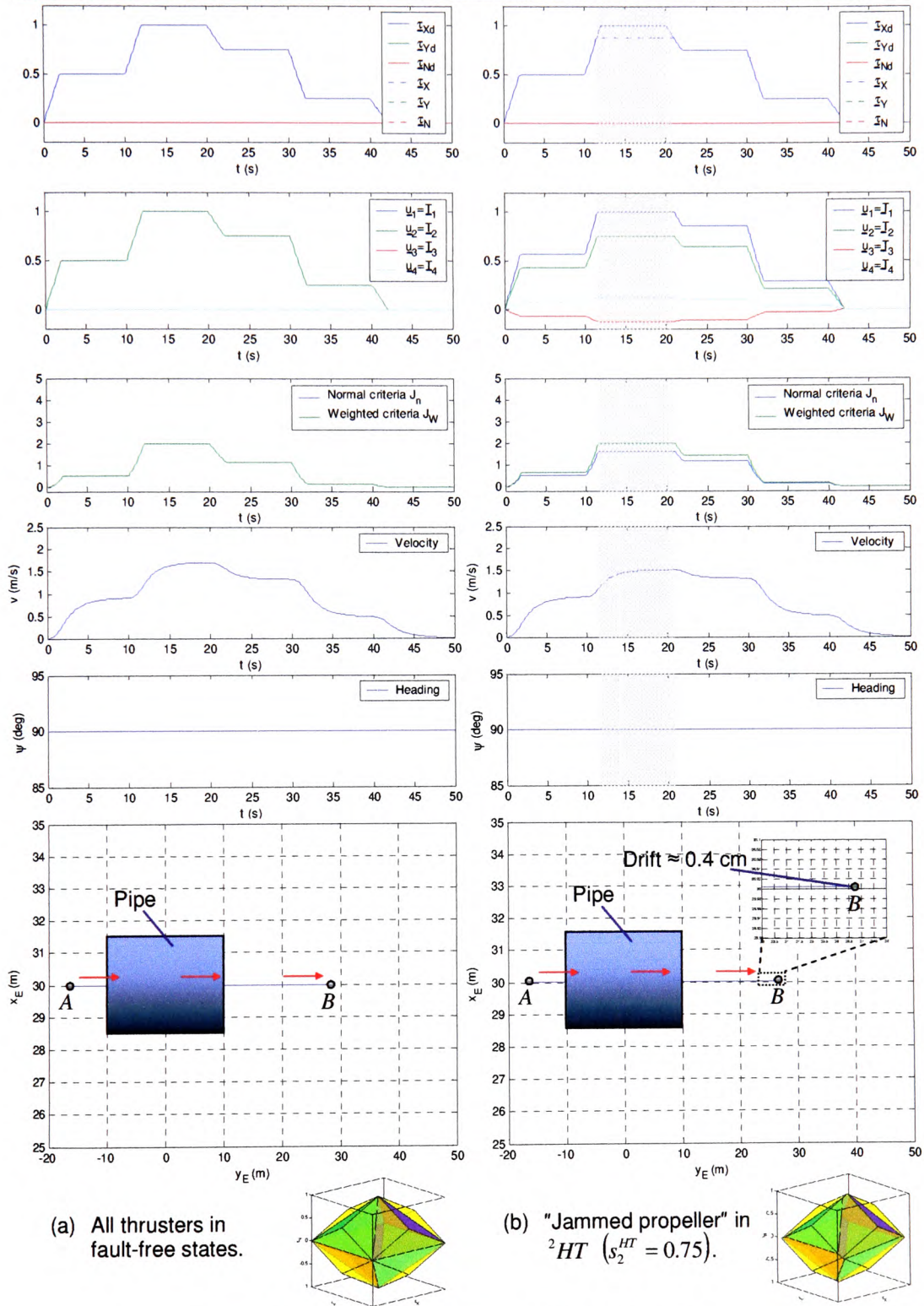


Figure 6.21 (B1) Partial fault - "Jammed propeller" (URIS).

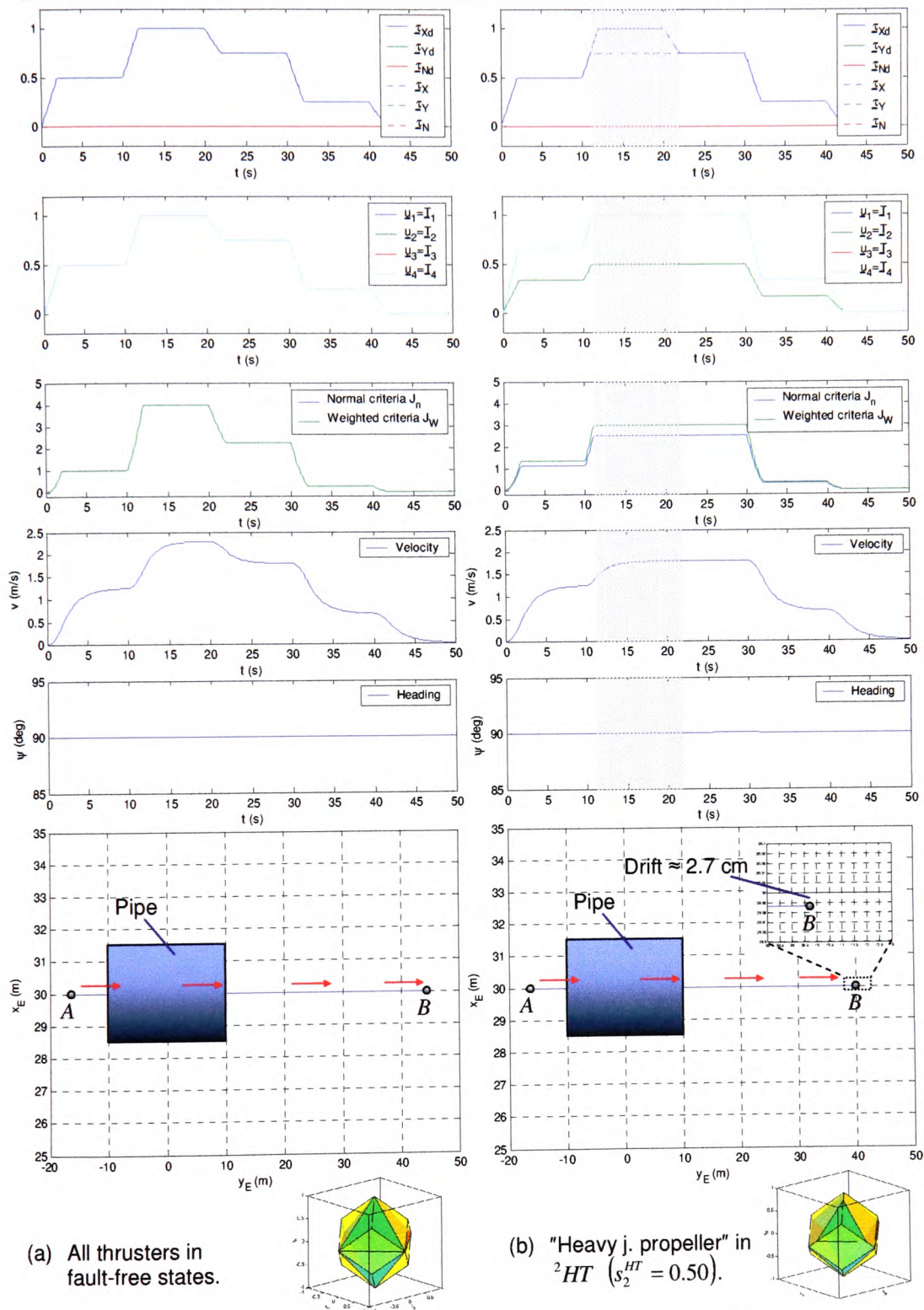


Figure 6.22 (B2) Partial fault - "Heavy jammed propeller" (FALCON).

Chapter 6:

Testing and Evaluation of the FDAS

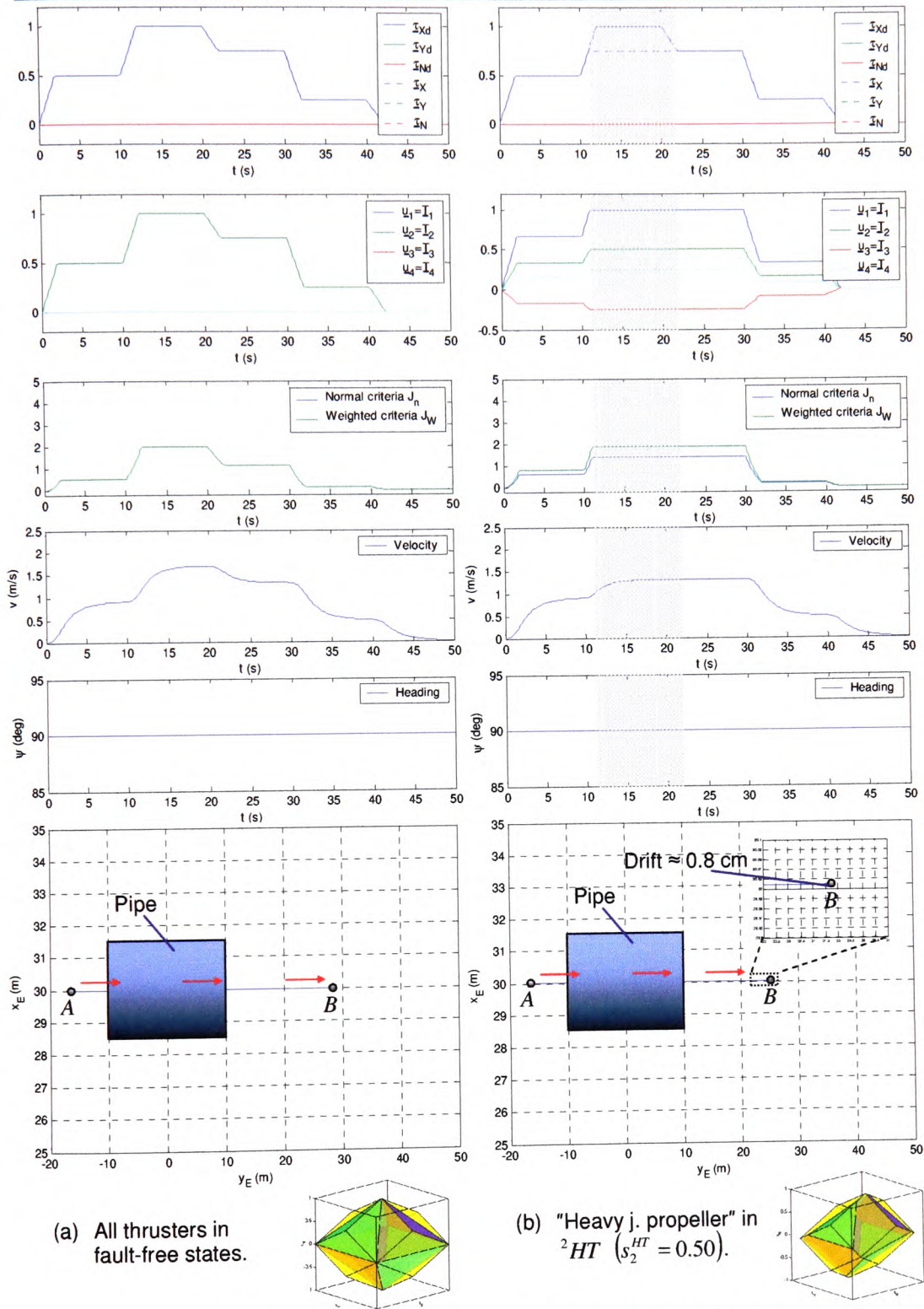


Figure 6.23 (B2) Partial fault - "Heavy jammed propeller" (URIS).

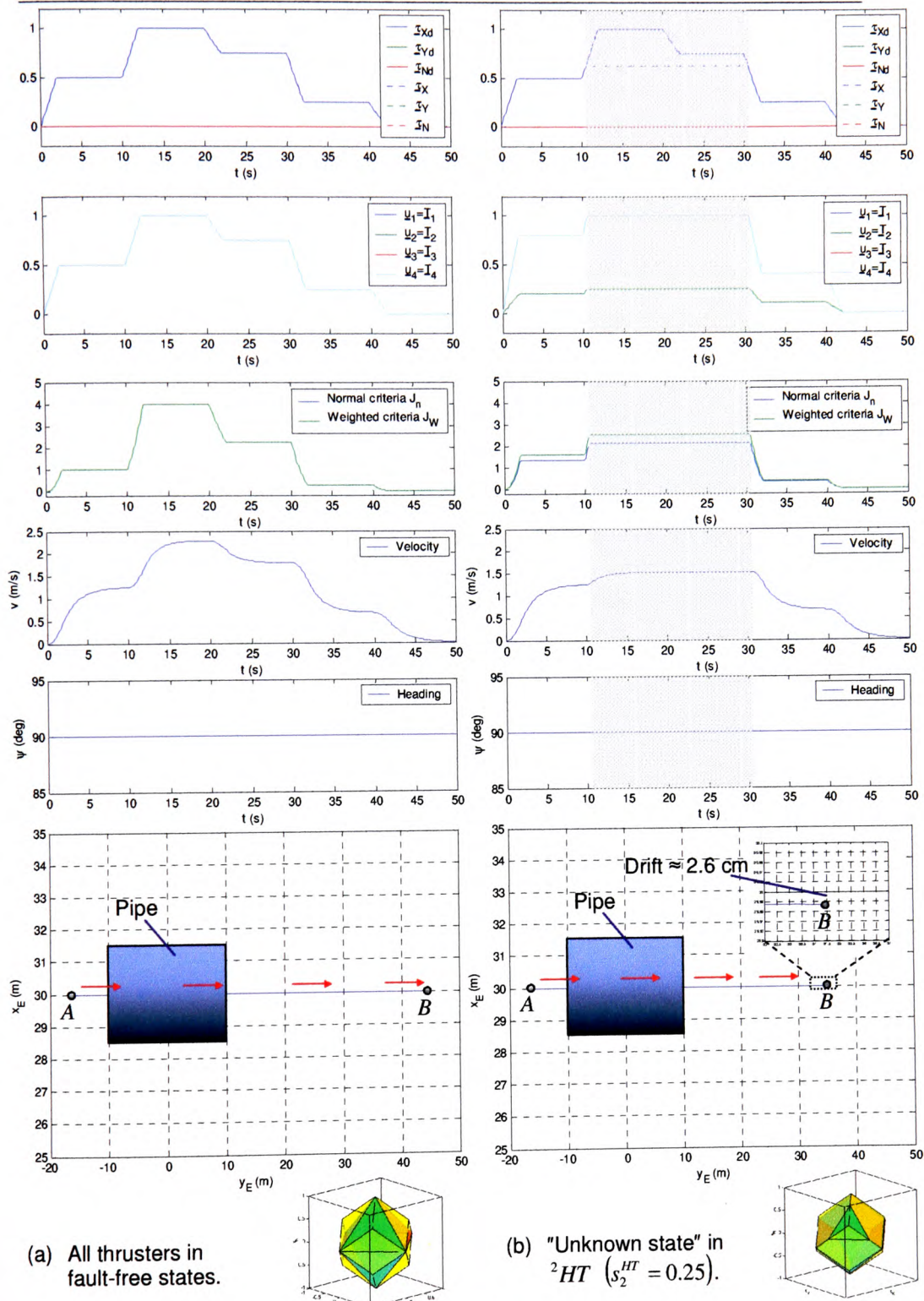


Figure 6.24 (B3) Partial fault - "Unknown state" (FALCON).

Chapter 6:

Testing and Evaluation of the FDAS

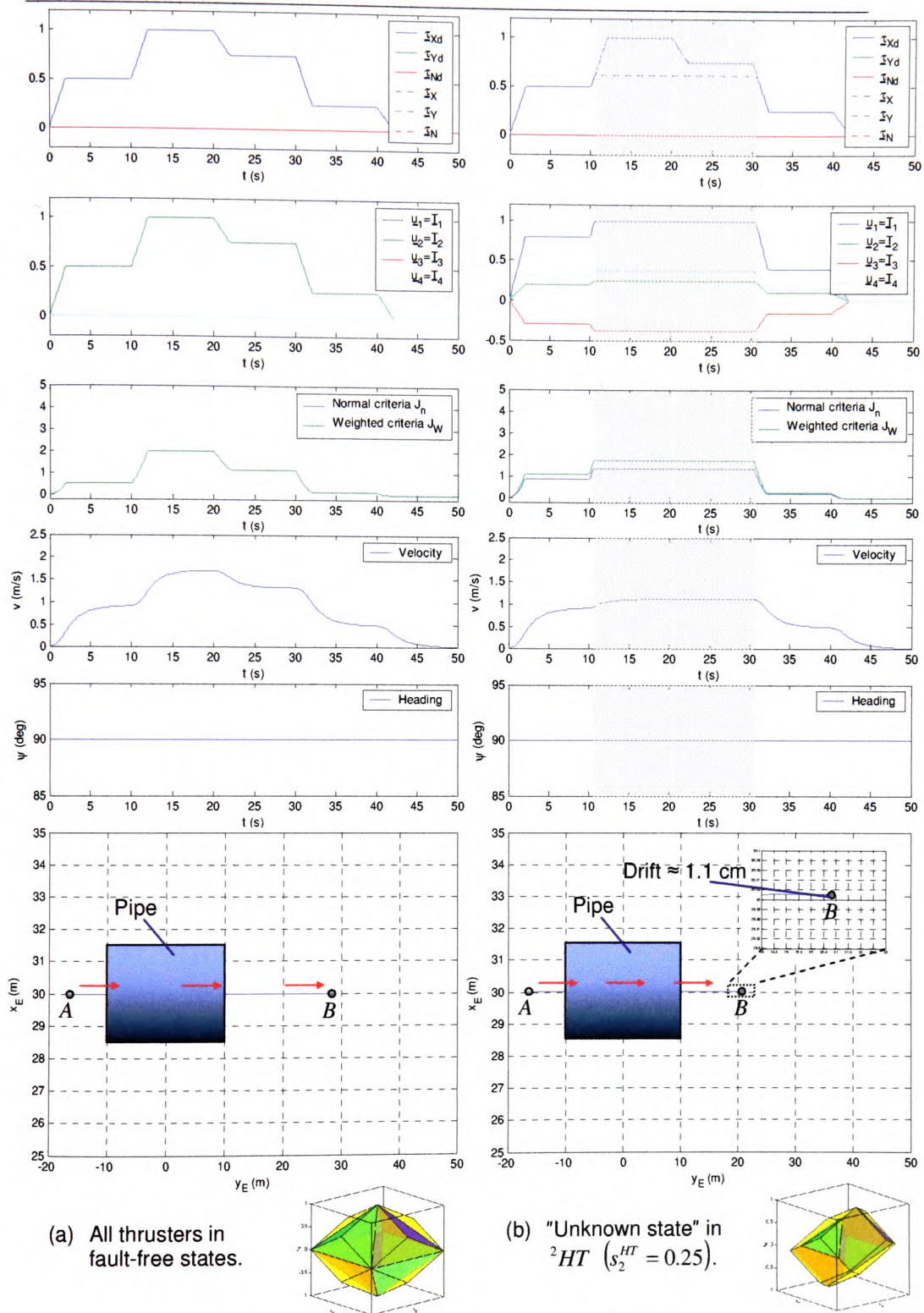


Figure 6.25 (B3) Partial fault - "Unknown state" (URIS).

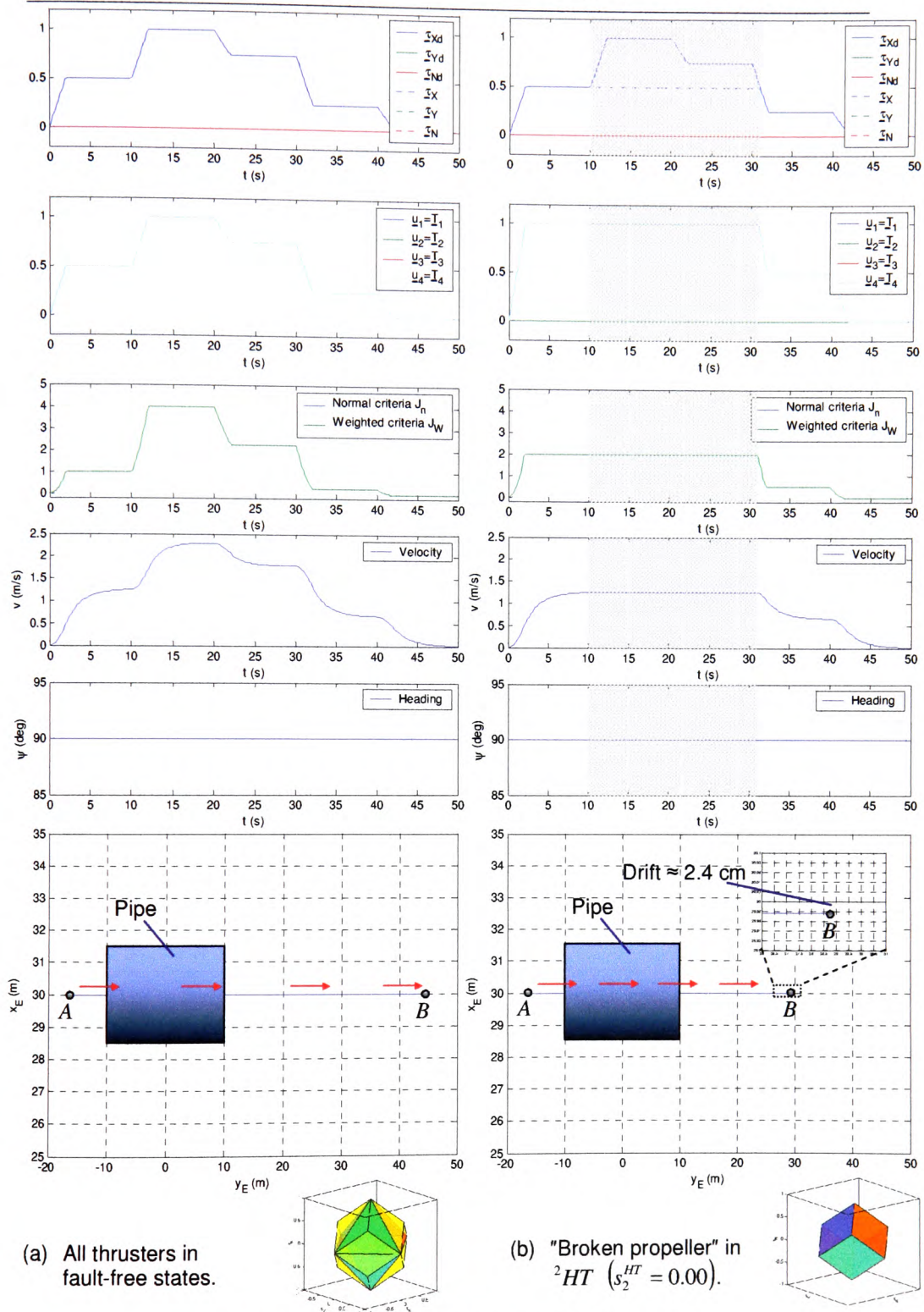


Figure 6.26 (B4) Total fault - "Broken propeller" (FALCON).

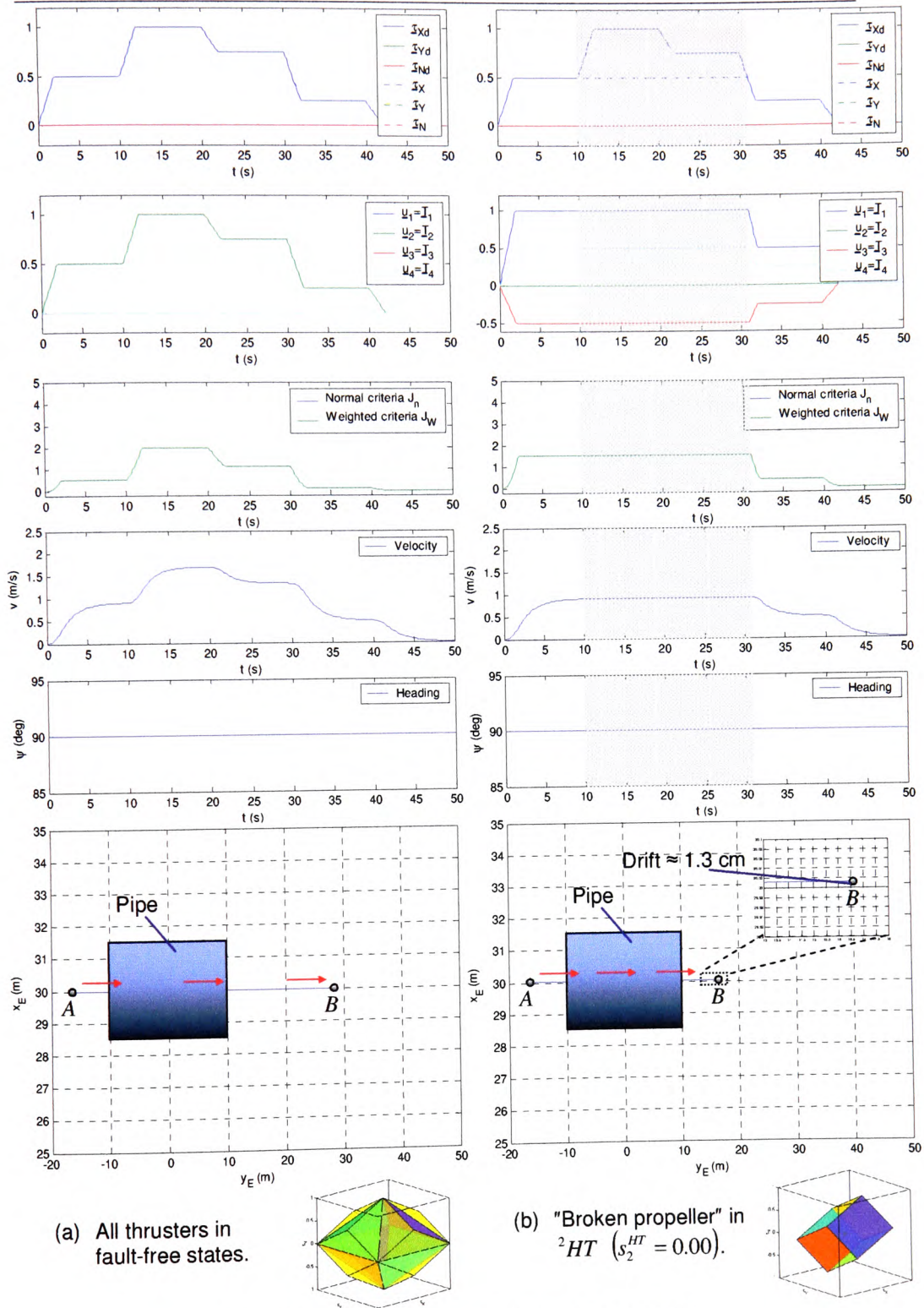
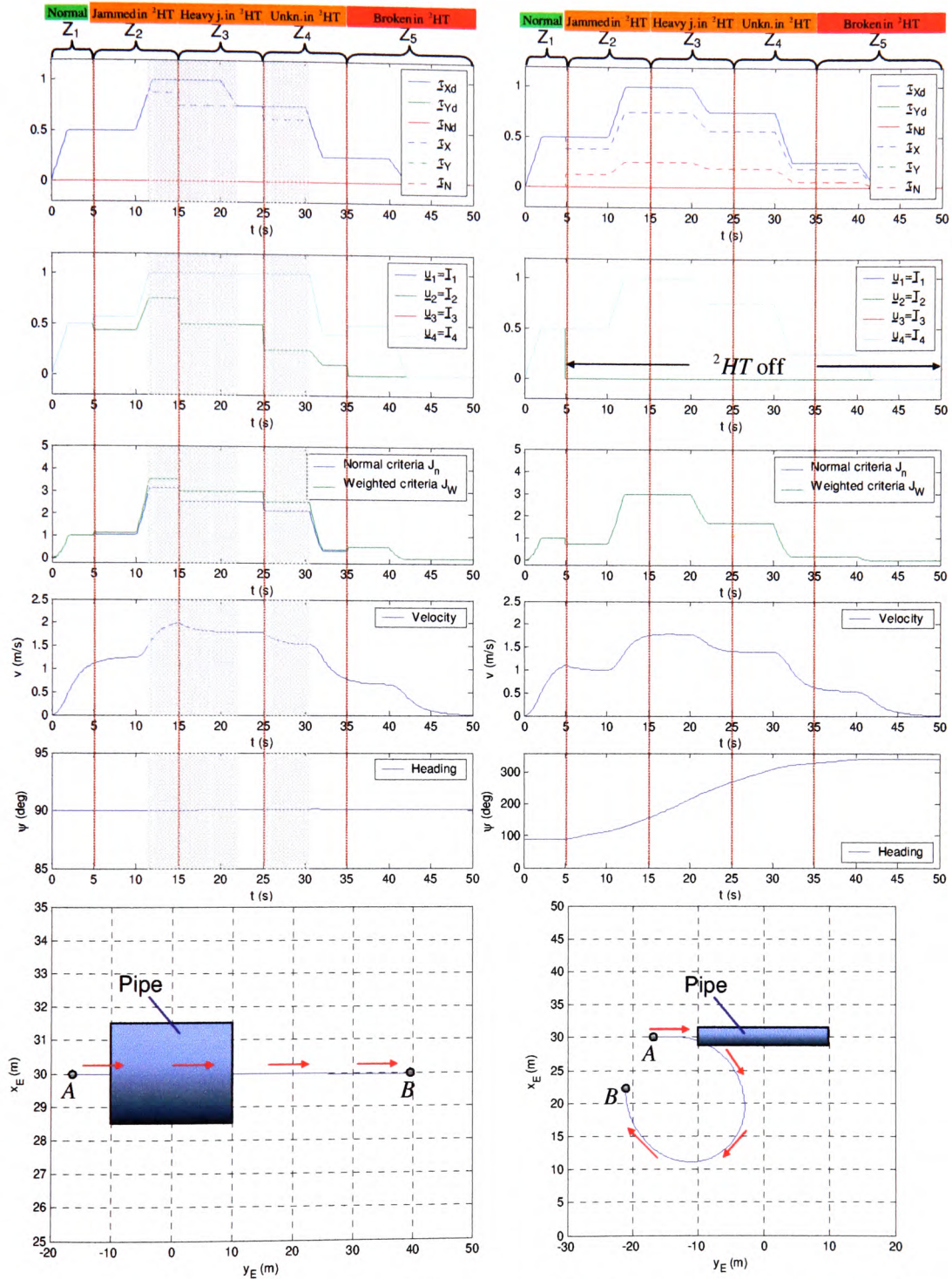
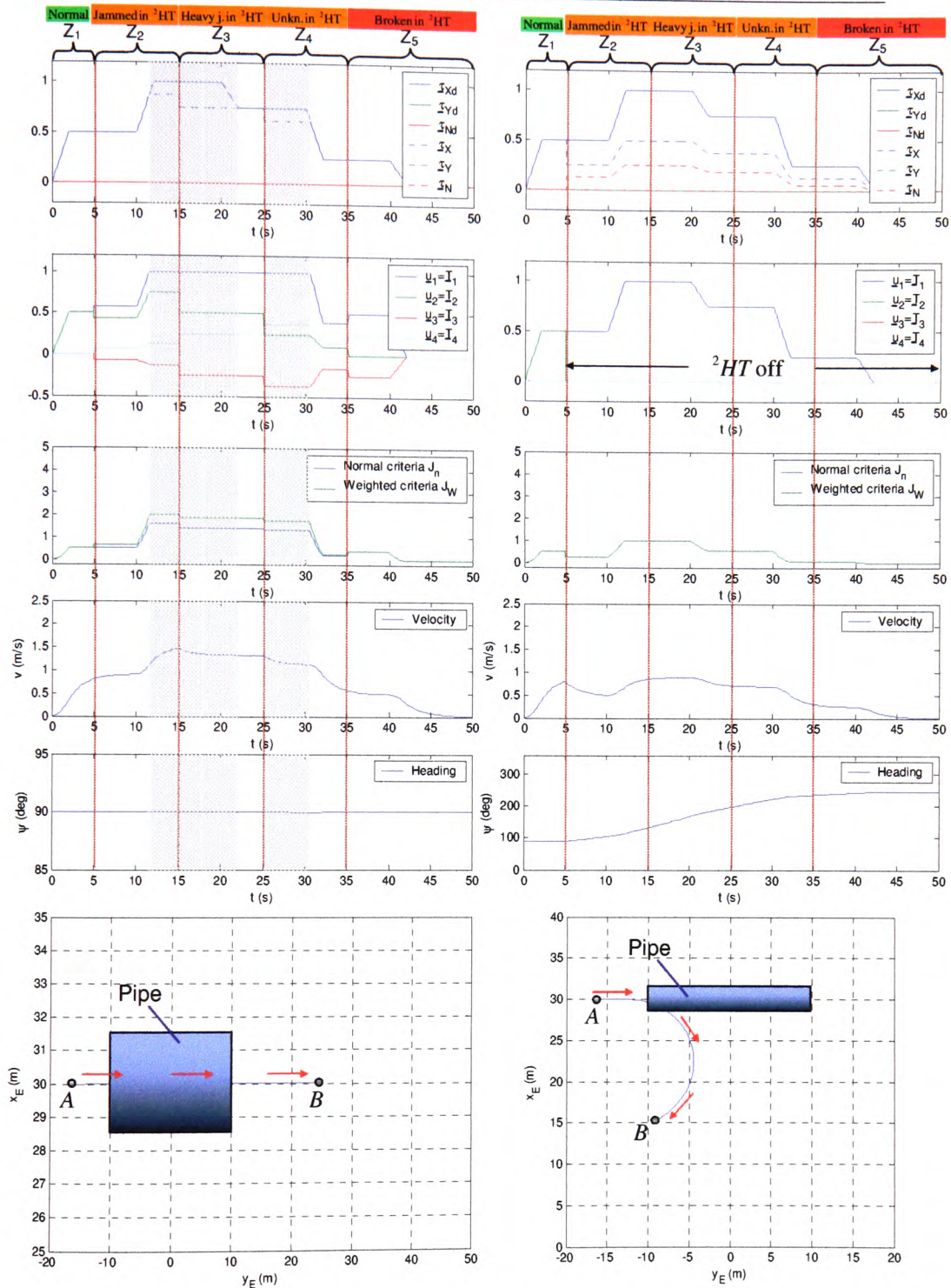


Figure 6.27 (B4) Total fault - "Broken propeller" (URIS).

(a) FDAS active (Different faults in 2HT).(b) FDAS not active (Different faults in 2HT).**Figure 6.28 (B5) Consecutive faults – passing through the pipe (FALCON).**

(a) FDAS active (Different faults in 2HT).(b) FDAS not active (Different faults in 2HT).**Figure 6.29 (B5) Consecutive faults – passing through the pipe (URIS).**

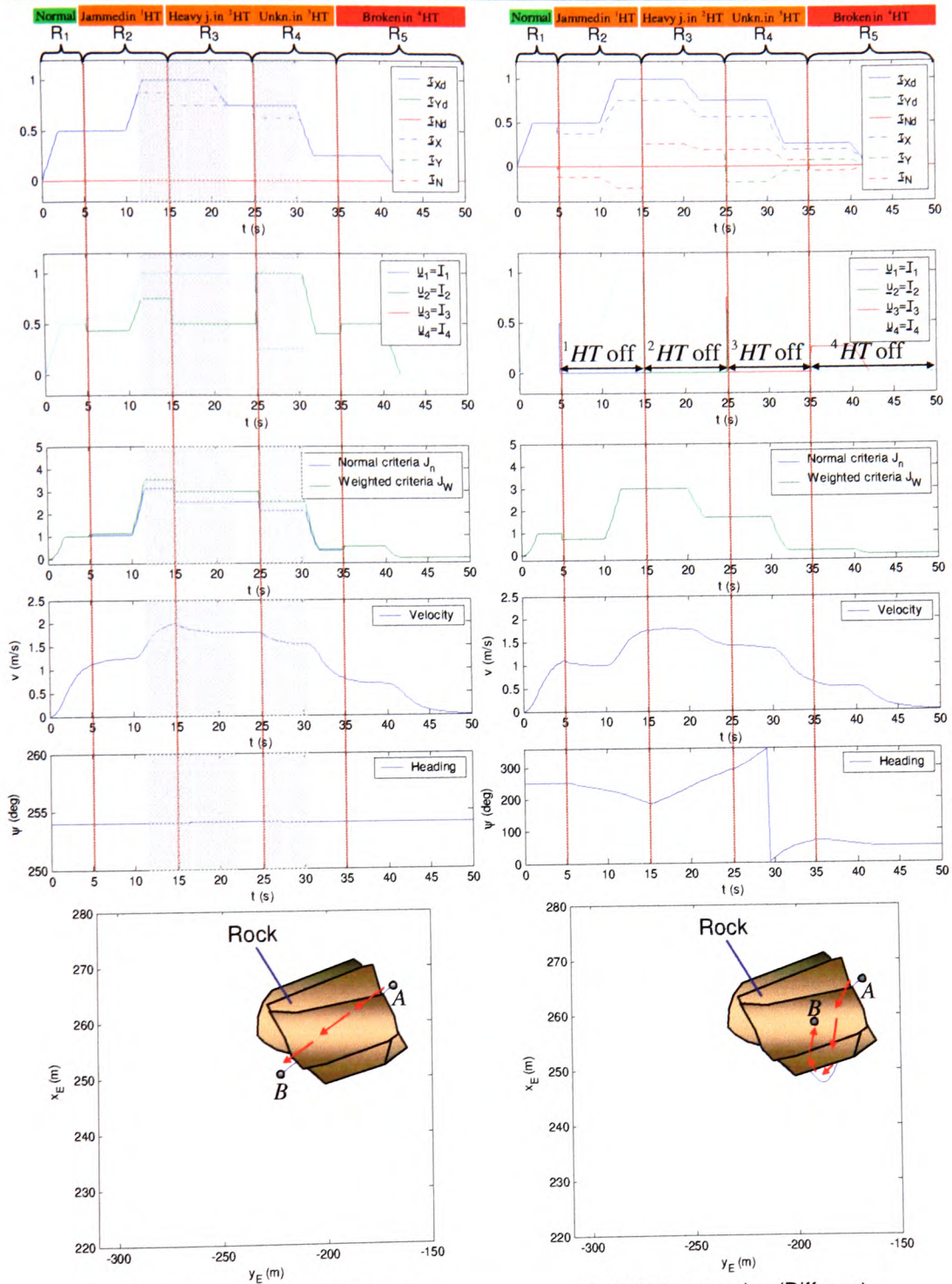
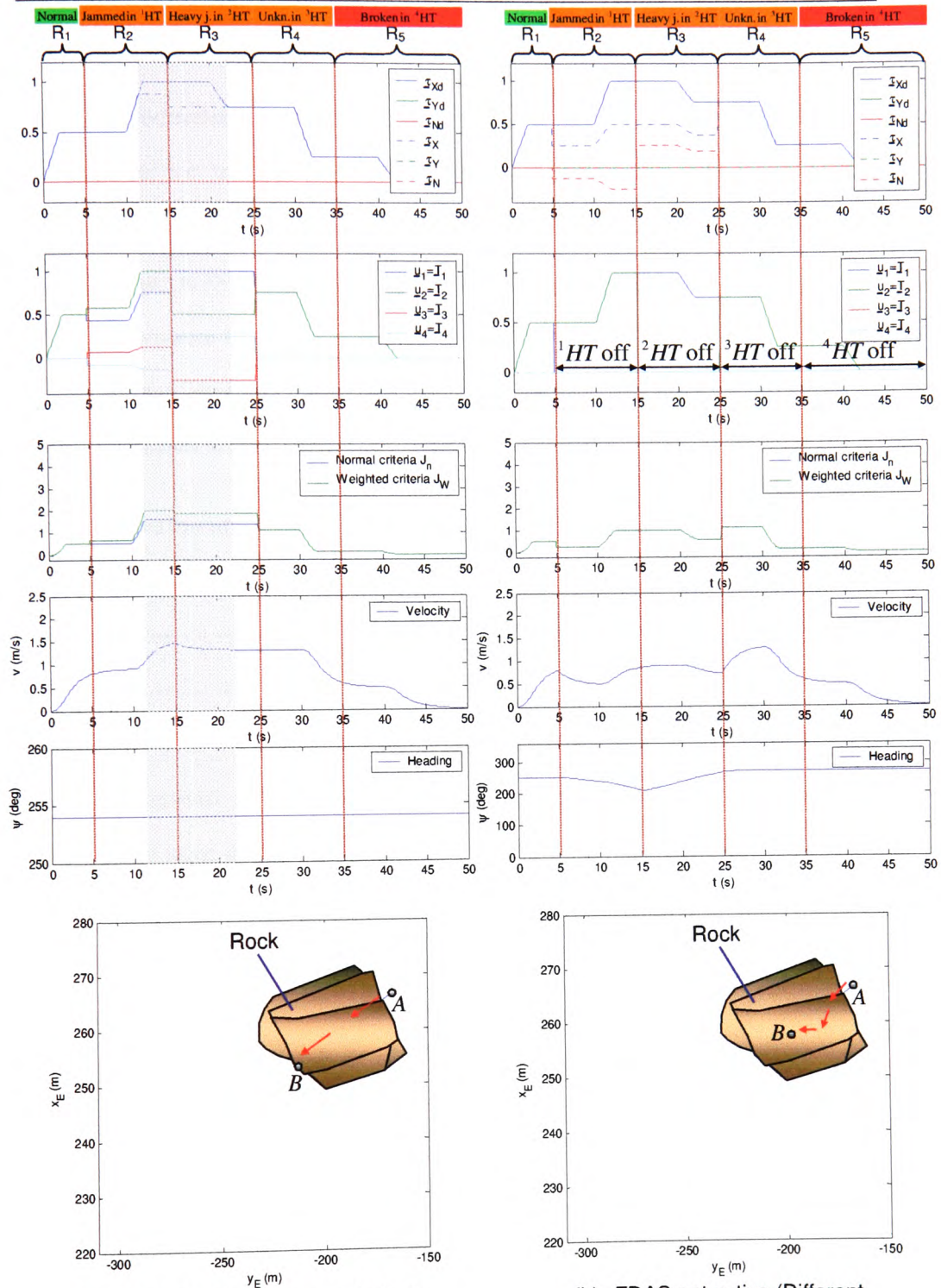


Figure 6.30 (B6) Consecutive faults – passing through the hole in the rock (FALCON).



(a) FDAS active (Different faults in different thrusters).

(b) FDAS not active (Different faults in different thrusters).

Figure 6.31 (B6) Consecutive faults – passing through the hole in the rock (URIS).

6.4 Real-time application

6.4.1 Introduction

A number of test trials with FALCON were scheduled for period December 2003 – January 2004. One part of two-day experiments, performed at the QinetiQ Ocean Basin Tank at Haslar in December, was the evaluation of the proposed FDAS and comparison of the performance between standard control architecture ("FDAS not active", see test case (B5)) and improved control architecture ("FDAS active"). Preliminary results of these experiments are presented in this section.

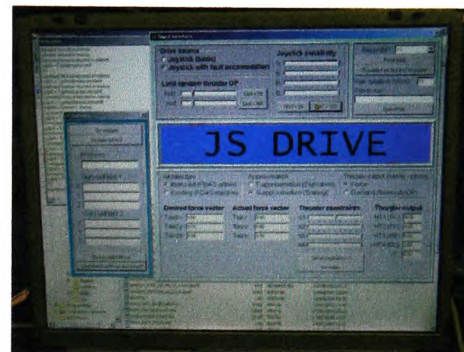
6.4.2 Experiment set-up

In order to test the performance of the FDAS, the standard surface components in the distributed control system of FALCON (HCU and main surface unit, Appendix A, page A-21) were replaced by the joystick and laptop. The laptop was connected to the control network via a RS232-RS485 converter. The joystick was used as the input device to generate command input signals (Figure 6.32 (a)). Both control architectures were implemented using the control application ATC (Advanced Thruster Control) developed in Borland Delphi programming environment (Figure 6.32 (b)), which represents an implementation of the FDAS for real-time control of FALCON. The ATC was used as a master node to control thruster slave nodes. The HCU controls SPEED, LAT SPEED and TURN RATE are implemented in ATC as scaling factors for joystick sensitivity (slider controls) f_x , f_y and f_N , respectively. The hybrid approach for control allocation, implemented in ATC, enables manual selection of saturation bounds s_i^{HT} for each horizontal thruster i^{HT} using slider controls. Different faulty situations can be artificially generated by changing values of these sliders. Slider values $s_i^{HT} = 1$, $0 < s_i^{HT} < 1$ and $s_i^{HT} = 0$ means fault-free state, partial fault and total fault in thruster i^{HT} , respectively.

During the trial, the ATC enables selection of the desired architecture using radio buttons. As stated in test case (B5), control architecture "FDAS not active" (based on the existing control software for FALCON) does not provide for thruster fault accommodation and the only available solution in faulty situation is to switch off a faulty thruster (regardless of fault type). In contrast, the control architecture "FDAS active" uses the FDAS to accommodate thruster faults. Switching between different architectures is instantaneous, enabling direct comparison of their performances during the same test trial. Neutral buoyancy was achieved by addition of weighting cells at the mounting points of the vehicle.



(a) Joystick as input device.



(b) Control application ATC.

Figure 6.32 Experiment set-up.

6.4.3 Day one

Since the Inertial Measurement Unit (IMU)⁵ was not available for the first day of experiments, it was decided to test the control allocation part of the FDAS. In the first experiment, FALCON was placed in a small tank (Figure 6.33). Different test trials were

⁵ Inertial Measurement Unit, developed by Nathan Sowerby (research fellow in IMPROVES project), is a device used to measure linear and angular velocities of the vehicle in body-fixed frame. See section 6.4.4 for more information about IMU.

performed with both architectures (separate and combined motion in four DOF (surge, sway, yaw and heave)). The limited tank size imposed reduction of joystick sensitivity, in order to prevent damage of the vehicle. The first observation was that FALCON has much faster dynamics than the ROV model used in the ROV simulator. This difference in dynamics was predicted in Appendix D (see page D-16). However, the response of the real vehicle on commanded inputs was *qualitatively* the same as the response of the model in ROV simulator, driven by the same command input signals. Different faulty situations were injected on-line, during the test trial. Poor performance was obtained in faulty situations in cases when the FDAS was not active, due to imperfect manual compensation of moment components, induced by disabling of a faulty thruster. In contrast, the control architecture with active FDAS gave superior performance. All three DOF in the horizontal plane were fully controllable. Although the manoeuvring space was limited due to the tank size, the control of FALCON was as easy as in the fault-free case. It is interesting to note that, from the ROV pilot point of view, no change in the response of the vehicle was noticeable in faulty situations, despite the limited usage of the faulty thruster. The reason for invariant behaviour of the vehicle in fault-free and faulty situations is because reduced joystick sensitivity resulted in limitation in the size of the desired control vector, such that it lies inside the attainable command set throughout the experiment, whereby the FDAS is always able to allocate the exact solution of the control allocation problem. In the second experiment, FALCON was driven through the narrow, shallow tunnel, which links two parts of the tank (Figure 6.34). A total fault in 2HT was artificially generated by setting the slider value s_2^{HT} to zero. That is, 2HT was disabled and removed from the control allocation process and the mission had to be performed with only three working horizontal thrusters. This experiment is similar to "passing-through-the-pipe" experiments in the ROV simulator (see test case (B4)). After the

FALCON entered the tunnel, real-time video from on-board camera was used as feedback information to control the motion. The experience gained from simulations resulted in successful completion of the mission. It should be emphasized that undesired drag effect of the umbilical cable was observable during the experiment. This effect was particularly noticeable for low-speed motion of the vehicle.

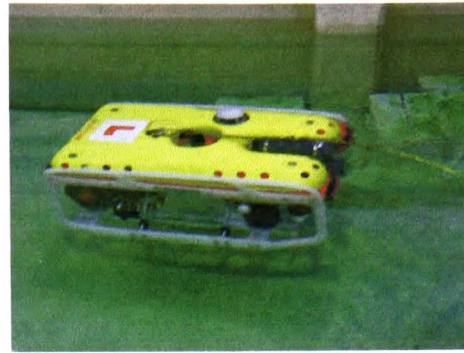
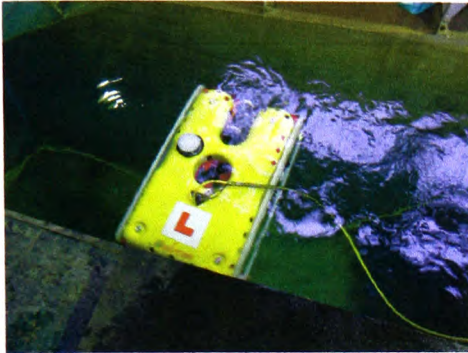


Figure 6.33 Tank trials with FALCON – Day one (the first experiment).



(a) "Faulty" FALCON in small tank.



(b) Narrow tunnel.



(c) Real-time video from on-board camera.



(d) "Faulty" FALCON successfully passed through the tunnel.

Figure 6.34 Tank trials with FALCON – Day one (the second experiment).

6.4.4 Day two

The IMU, shown in Figure 6.35, was available for the second day of experiments. It was fitted on the chassis of the FALCON using waterproof case. From the control point of view, the IMU represents a slave node, supervised by a master node (control application ATC). Experimental data obtained by the IMU (time responses of surge, sway & heave linear accelerations and roll, pitch & yaw rates) were acquired and stored on hard disk for post-experiment analysis and identification of FALCON dynamics. It should be pointed that development of the IMU is an ongoing project and final versions of routines for data processing and interpretation of experimental results are not yet available. This means that the IMU diagrams, presented in this section, should not be considered as final versions and their accuracy will be improved in future work. Test trials were performed in a large tank. The same day SeaEye Marine Ltd. performed test trials with their new vehicle COUGAR. Figure 6.36 displays both vehicles in water, next to each other. Marker lines at the bottom of the tank were used for path following experiments, as explained later in this section. The first experiment of day two was similar to the first experiment of day one. A series of tests was performed, in which FALCON was driven in four DOF, but this time large tank size did not impose any limits on joystick sensitivity, i.e. full range of speed was available. A short description of tests is given in Table 6.8.

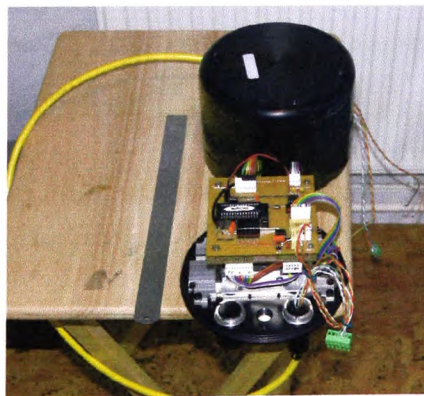


Figure 6.35 Inertial Measurement Unit (IMU).

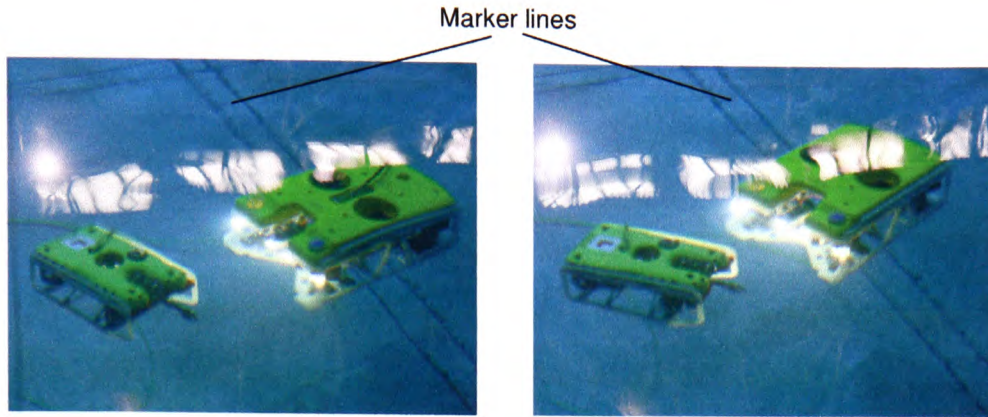


Figure 6.36 FALCON and COUGAR in the tank.

Test	Description	Diagrams
FM1	Free motion in 4 DOF, fault-free state in all thrusters, FDAS active	Figure 6.37 (pg. 6-66)
FM2	Free motion in 4 DOF, total fault in ${}^2\text{HT}$ (disabled), FDAS not active	Figure 6.38 (pg. 6-66)
FM3	Free motion in 4 DOF, total fault in ${}^2\text{HT}$ (disabled), FDAS active	Figure 6.39 (pg. 6-67)
FM4	Free motion in 4 DOF, partial fault in ${}^2\text{HT}$ ($s_2^{\text{HT}} = 0.50$), FDAS active	Figure 6.40 (pg. 6-67)
FM5	Free motion in 4 DOF, partial fault in ${}^2\text{HT}$ ($s_2^{\text{HT}} = 0.20$), FDAS active	Figure 6.41 (pg. 6-67)
FM6	Free motion in 4 DOF, partial fault in ${}^2\text{HT}$ ($s_2^{\text{HT}} = 0.80$), FDAS active	Figure 6.42 (pg. 6-68)

Table 6.8 Description of tests in the first experiment (Day two).

Time diagrams of desired & actual propeller angular velocities and absolute value of motor current for individual thrusters are shown in Figure 6.37 - Figure 6.42. In accordance with FALCON control protocol, all three variables are represented as an integer numbers between -100 and $+100$. Analysing these diagrams, the following features can be observed:

- **Quality of signals.** In contrast to noisy data, obtained in experiments with URIS (see section 5.4.3), adequate signal conditioning and shielding in the case of FALCON resulted in high-quality, low-noise data.

- **Effect of neglected thruster control loop dynamics.** There is a small delay between desired and actual propeller angular velocities, due to dynamics in thruster control loops. This delay is unavoidable, since the thruster control loop is a dynamic system. The cumulative effect of these delays on global control performance is the appearance of delay between the desired and actual behaviour of the vehicle. That is, there is a time delay between commanded inputs and response of the vehicle. The size of this delay is small and acceptable in most ROV applications.
- **Steady-state error.** A steady-state error is noticeable in time response of actual propeller angular velocity. The size of the error is determined by two factors: propeller load and velocity controller settings. The steady-state error is proportional to propeller load: higher the load, higher the error. Since the propeller load depends on the magnitude of a desired propeller angular velocity, similar relationship can be derived between error and velocity: higher the velocity, higher the error. The propeller load is observable from motor current response: higher the load, higher the current. The velocity controller is developed by Seaeye Marine Ltd. as a digital PID controller, and adjustable parameters are proportional, integral and derivative gains. Actual parameter settings are obtained using heuristic methods. Although the current performance of the velocity controller is satisfactory for typical ROV applications, the size of steady-state error can be reduced using advanced controller designs.
- **Nonzero response of actual propeller velocity on zero demand.** Figure 6.38 and Figure 6.39 display time diagrams of desired and actual propeller angular velocities for the case of total error in 2HT . Although 2HT was disabled ($n_d^{HT_2} = 0$) all the time during the experiments FM2 and FM3, inconsistent water

flow around propellers caused its rotation, resulting in nonzero actual propeller velocity ($n^{HT_2} \neq 0$) in some segments, as indicated in Figure 6.38 and Figure 6.39. This undesired propeller rotation introduces uncertainty inside critical zone, making successful fault detection and isolation difficult to achieve. Similar behaviour for zero-velocity case was experienced during test trials with URIS. As stated in section 5.4.3, the solution for this problem is the exclusion of zero-velocity segments from the training and on-line fault detection phase.

- **Limitation of desired propeller angular velocity in faulty situations.** In the case of partial fault in 2HT , the desired angular velocity is bounded between limits determined by $|n_d^{HT_2}|_{\max} = \text{round}\left(100\sqrt{|s_2^{HT}|}\right)$.

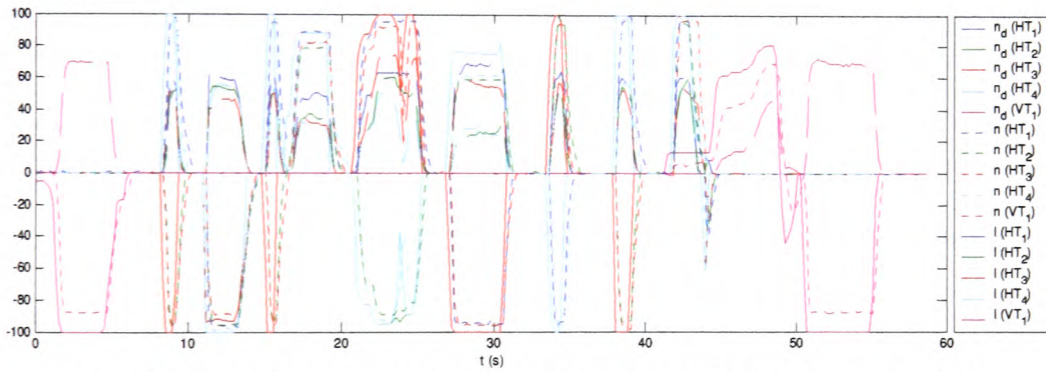


Figure 6.37 Test FM1 (free motion in 4 DOF, fault-free state in all thrusters, FDAS active).

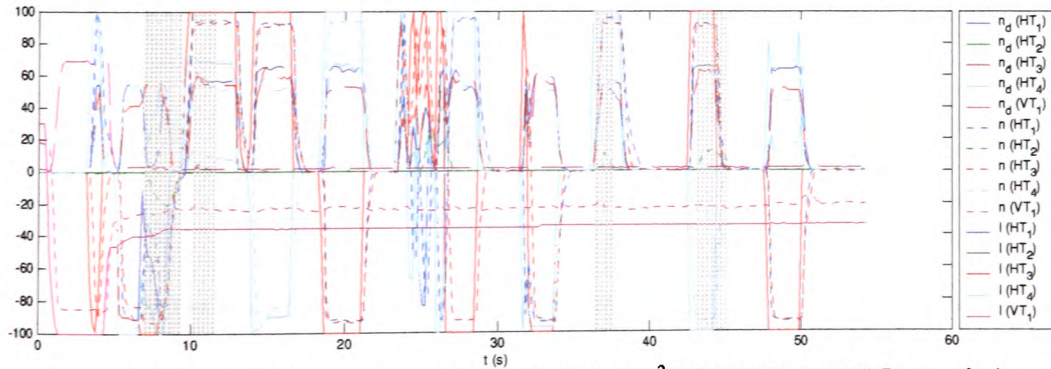


Figure 6.38 Test FM2 (free motion in 4 DOF, total fault in 2HT (disabled), FDAS not active).

Segments with $n_d^{HT_2} = 0$ and $n^{HT_2} \neq 0$ are highlighted.

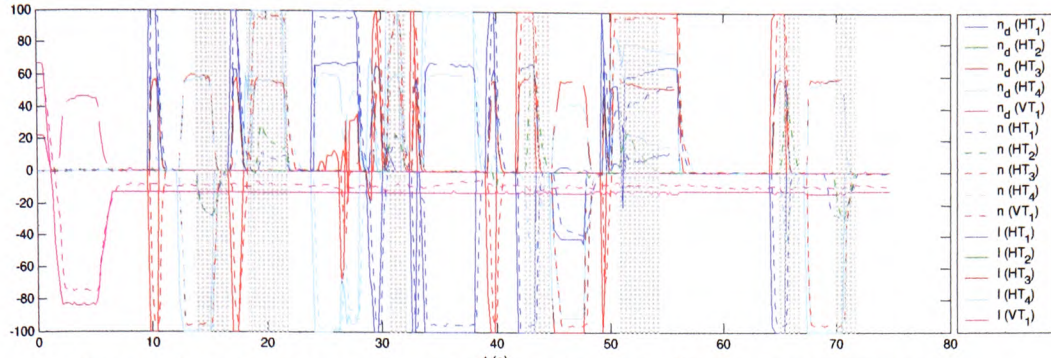


Figure 6.39 Test FM3 (free motion in 4 DOF, total fault in 2HT (disabled), FDAS active).

Segments with $n_d^{HT_2} = 0$ and $n^{HT_2} \neq 0$ are highlighted.

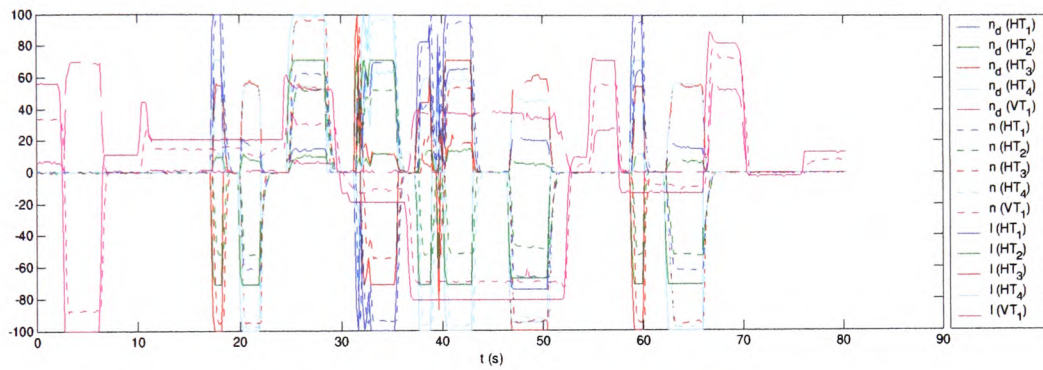


Figure 6.40 Test FM4 (free motion in 4 DOF, partial fault in 2HT ($s_2^{HT} = 0.50$), FDAS active).

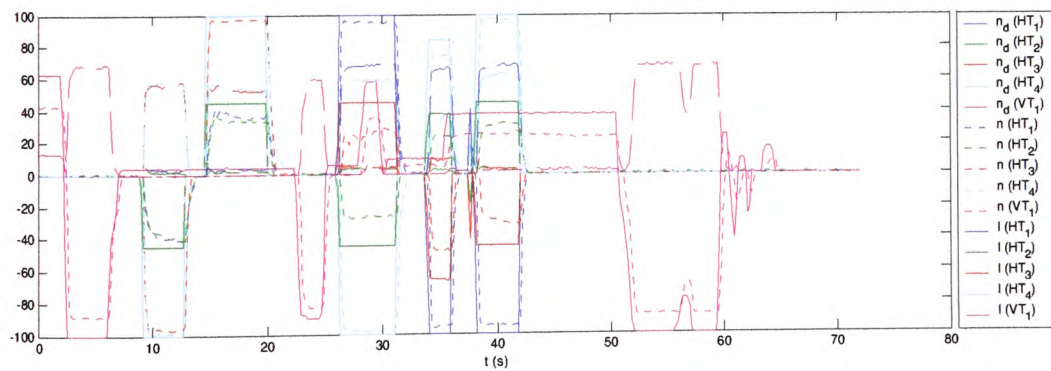


Figure 6.41 Test FM5 (free motion in 4 DOF, partial fault in 2HT ($s_2^{HT} = 0.20$), FDAS active).

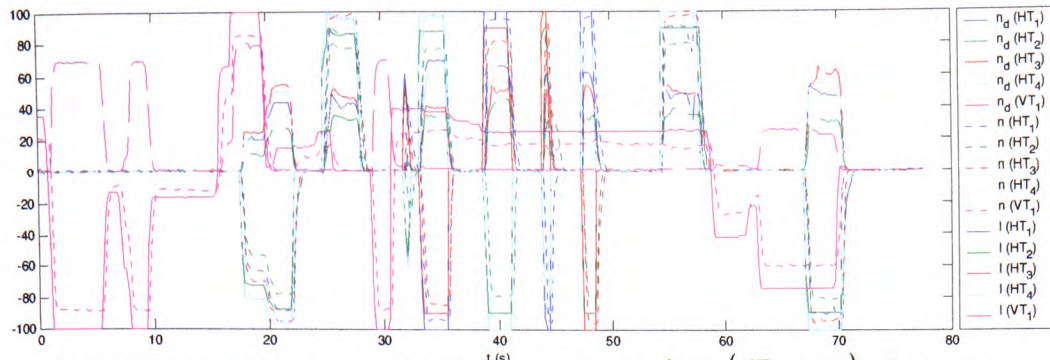


Figure 6.42 Test FM6 (free motion in 4 DOF, partial fault in 2HT ($s_2^{HT} = 0.80$), FDAS active).

In the second experiment of day two, the path following task⁶ was performed in different faulty conditions. The aim was to evaluate and compare the performance of the standard and improved control architectures performing path following tasks in different faulty situations. FALCON had to follow the virtual rectangular path, created by marker lines at the bottom of the tank (see Figure 6.36). The path following task, performed in the clockwise direction, started and ended at the same point (lower left corner of the rectangle). A series of tests was performed, each time with different fault state in thruster 2HT . A short description of these tests is given in Table 6.9.

Test	Description	Diagrams
PF1	Path following, fault-free state in all thrusters, FDAS active	Figure 6.43 (pg. 6-70)
PF2	Path following, total fault in 2HT (disabled), FDAS not active	Figure 6.44 (pg. 6-71)
PF3	Path following, total fault in 2HT (disabled), FDAS active	Figure 6.45 (pg. 6-72)
PF4	Path following, partial fault in 2HT ($s_2^{HT} = 0.50$), FDAS active	Figure 6.46 (pg. 6-73)
PF5	Path following, partial fault in 2HT ($s_2^{HT} = 0.20$), FDAS active	Figure 6.47 (pg. 6-74)

Table 6.9 Description of tests in the second experiment (Day two).

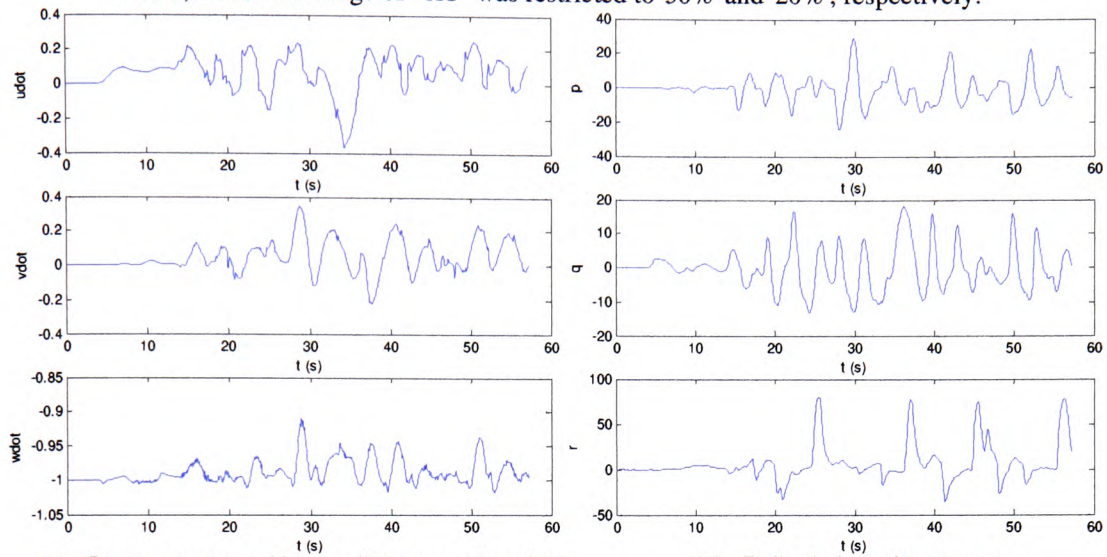
⁶ Path following tests with FALCON were performed in open-loop mode, i.e. command signals were generated exclusively by the ROV pilot using joystick, without any controller-in-the-loop. The ROV pilot put great effort to perform path following task neutrally, without favouring any control architecture.

Time diagrams of the IMU measurements (surge, sway & heave linear accelerations and roll, pitch & yaw rates), desired & actual propeller angular velocities and heading are shown in Figure 6.43 - Figure 6.47. Heading responses are obtained by integration of yaw rates. In order to improve readability, responses of motor currents are omitted in these diagrams. Analysing these diagrams, the following observations are noticeable:

- **Quality of signals.** The level of the noise in the IMU measurements is low, due to good signal conditioning and shielding inside the IMU.
- **Improvements obtained by using the FDAS.** Although the reference path was the same for each test PF1 – PF5, certain differences are noticeable in the obtained time responses. In particular, those obtained in test PF2 (Figure 6.44) are more disturbed (i.e. they have richer frequency contents) than in the other tests. The reason for this dissimilarity stems from the fact that the test PF2 was performed with disabled thruster 2HT and the FDAS was not active, so that the ROV pilot had to try to maintain the heading by applying manual compensation for unbalanced moment components, induced by switching off thruster 2HT . However, this is a very difficult task and non perfect compensation led to poor tracking performance and an oscillatory character of the yaw response. In contrast, test PF3 was performed in a similar way, i.e. with disabled 2HT , but this time the FDAS was active. The ability of the FDAS to accommodate total fault in 2HT by performing automatic reallocation among remaining operable thrusters resulted in a much smoother yaw response, as indicated in Figure 6.45. This demonstrates the efficiency of the FDAS and highlights its main advantage compared to the existing control architecture: in the case of a fault in a single thruster, the ROV pilot will control the vehicle in the same way as in the fault-free case, without need to perform any compensation. However, the attainable command set shrinks

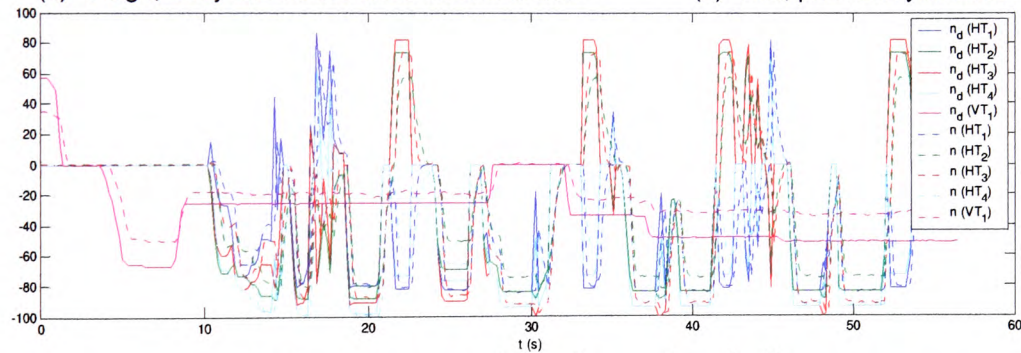
for the case of a fault and velocity saturation bounds are reduced, resulting in a reduction in ROV velocity. Smooth yaw responses are also obtained in tests PF4

& PF5, where the usage of 2HT was restricted to 50% and 20%, respectively.

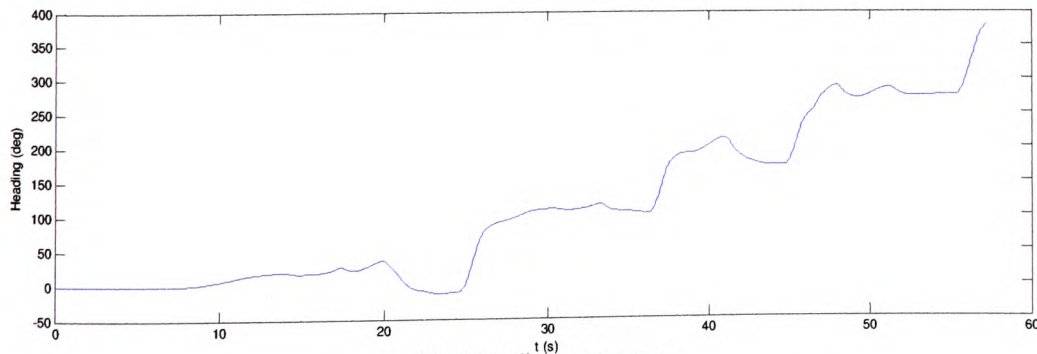


(a) Surge, sway and heave linear accelerations.

(b) Roll, pitch and yaw rates.

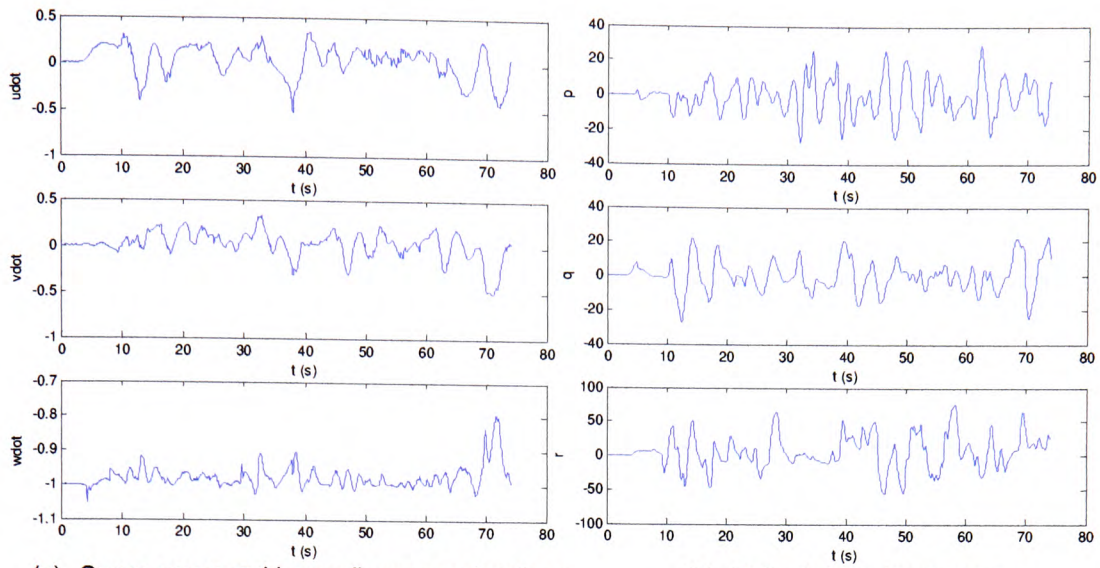


(c) Desired and actual propeller angular velocities.



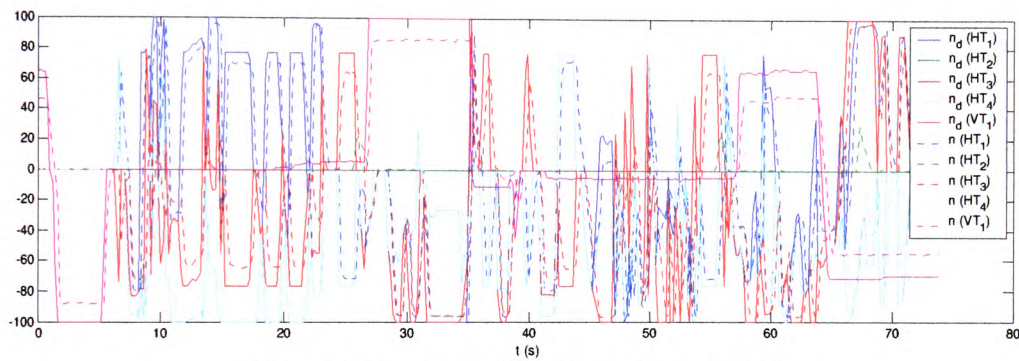
(d) Heading response.

Figure 6.43 Test PF1 (path following, fault-free state in all thrusters, FDAS active).

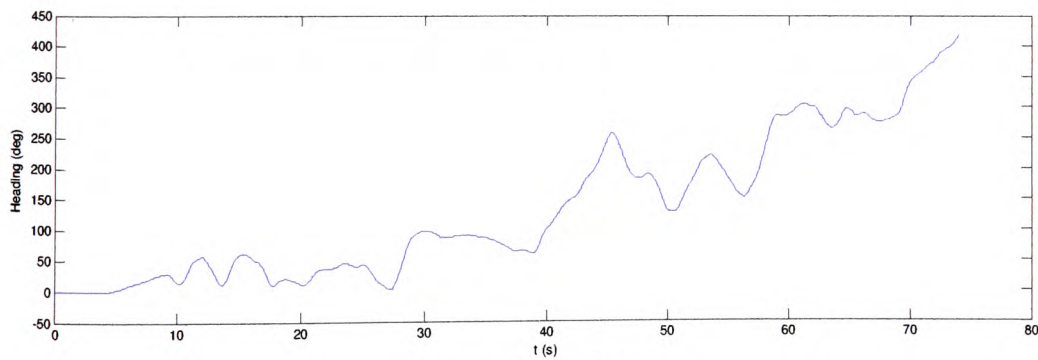


(a) Surge, sway and heave linear accelerations.

(b) Roll, pitch and yaw rates.

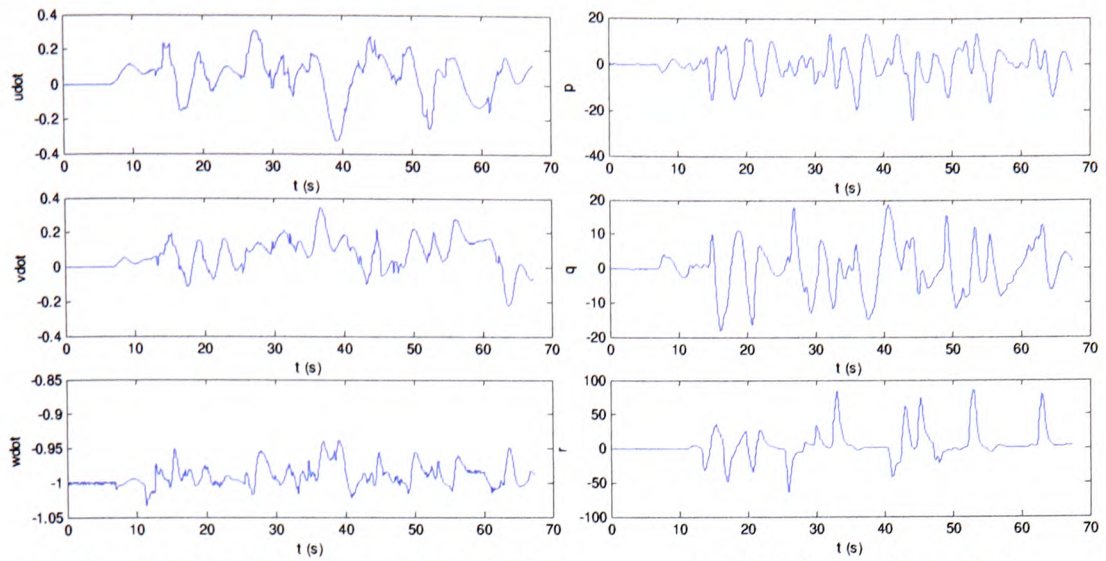


(c) Desired and actual propeller angular velocities.



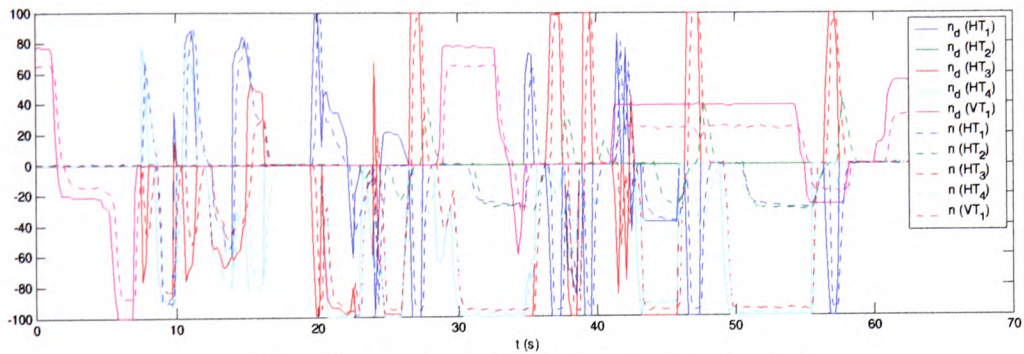
(d) Heading response.

Figure 6.44 Test PF2 (path following, total fault in 2HT (disabled), FDAS not active).

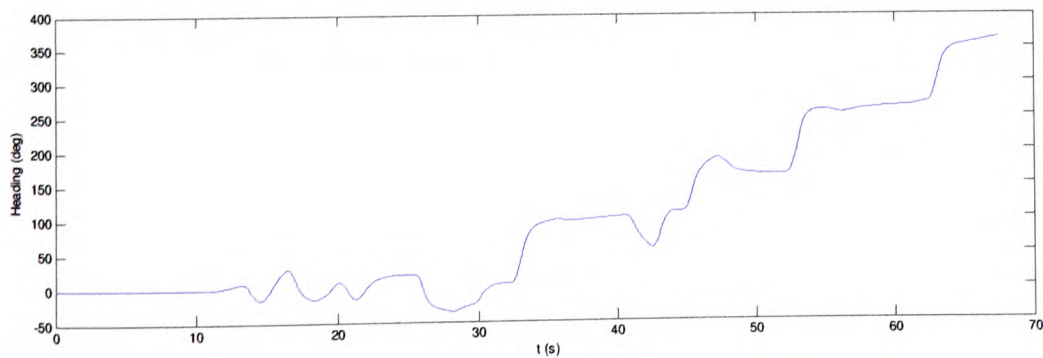


(a) Surge, sway and heave linear accelerations.

(b) Roll, pitch and yaw rates.

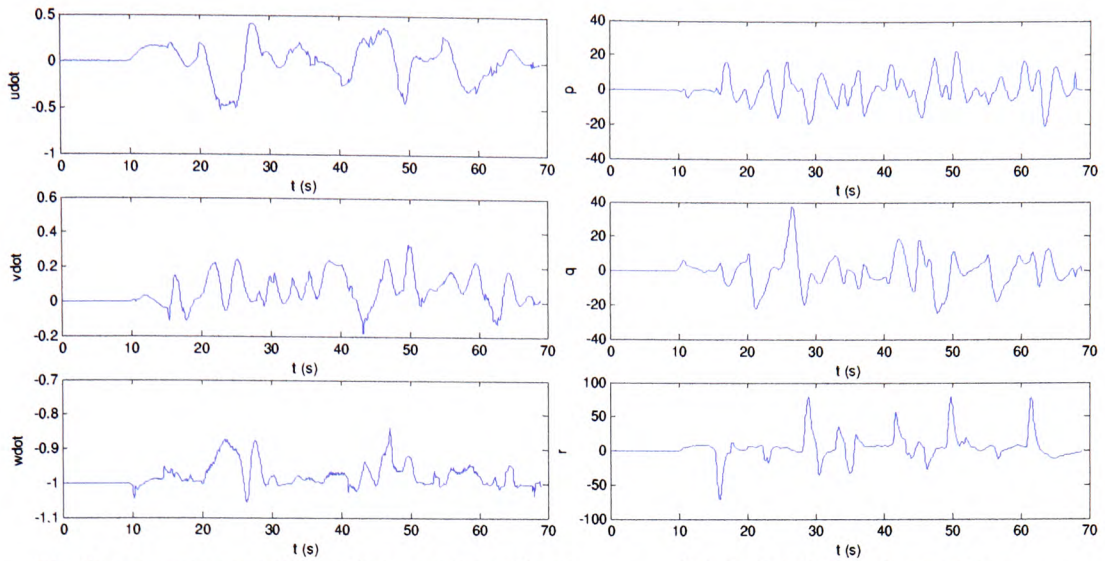


(c) Desired and actual propeller angular velocities.



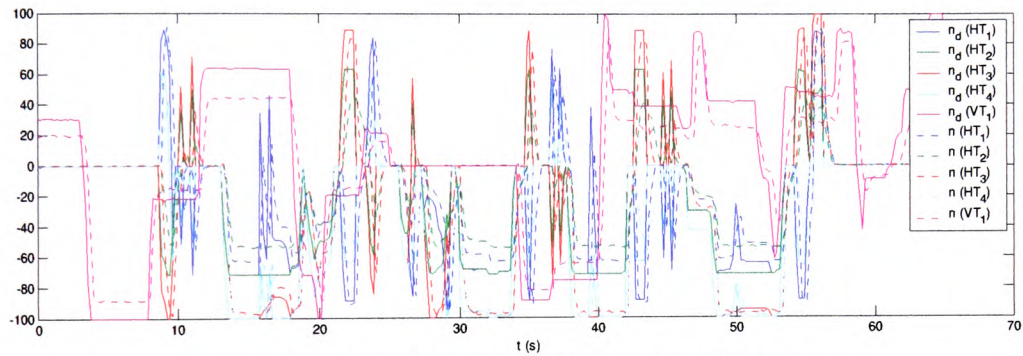
(d) Heading response.

Figure 6.45 Test PF3 (path following, total fault in 2HT (disabled), FDAS active).

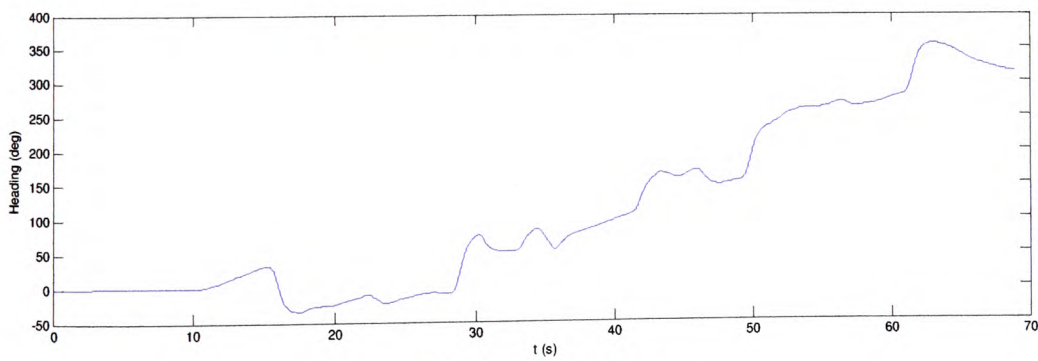


(a) Surge, sway and heave linear accelerations.

(b) Roll, pitch and yaw rates.

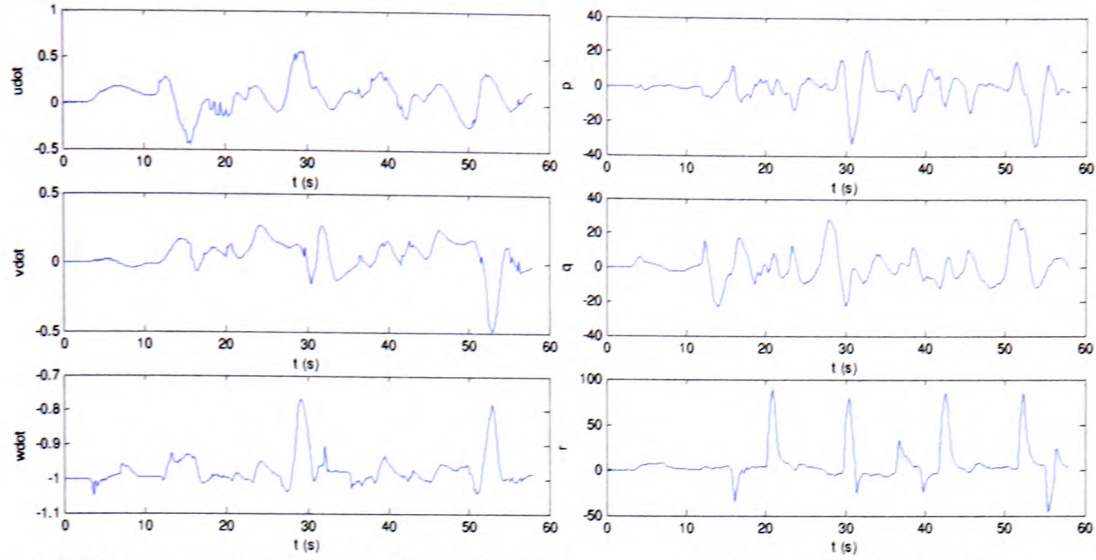


(c) Desired and actual propeller angular velocities.



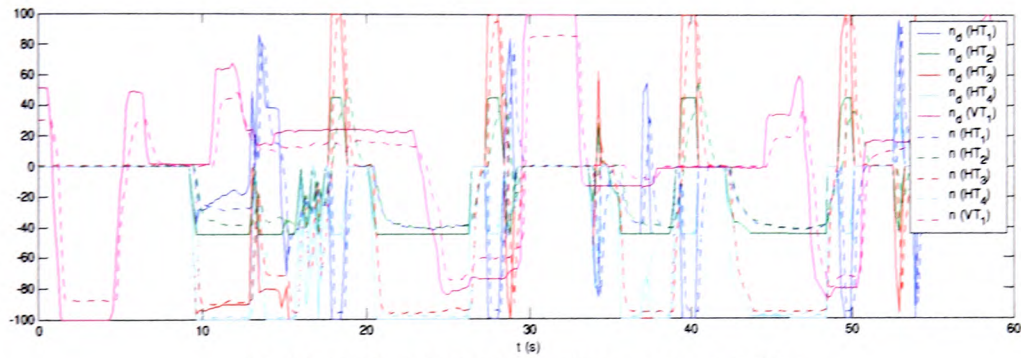
(d) Heading response.

Figure 6.46 Test PF4 (path following, partial fault in 2HT ($s_2^{HT} = 0.50$), FDAS active).

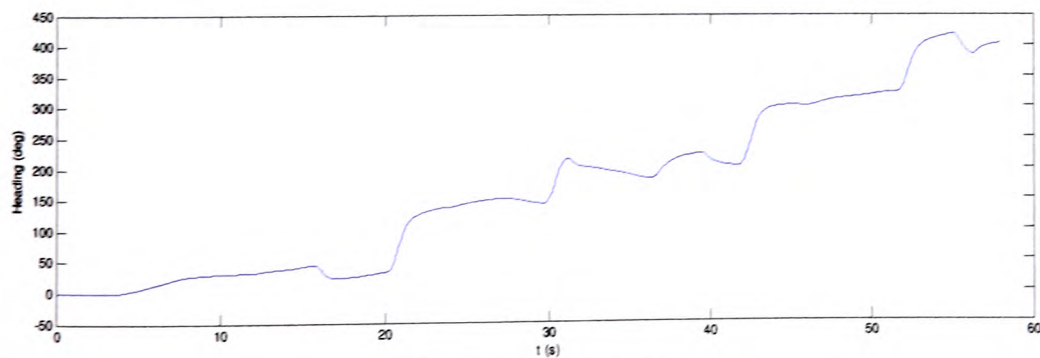


(a) Surge, sway and heave linear accelerations.

(b) Roll, pitch and yaw rates.



(c) Desired and actual propeller angular velocities.



(d) Heading response.

Figure 6.47 Test PF5 (path following, partial fault in 2HT ($s_2^{HT} = 0.20$), FDAS active).

6.5 Concluding remarks

This chapter has evaluated the performance of the FDAS and highlighted its key features. Using both simulations and real-world tests, the performance was evaluated through a series of representative test cases, in order to examine the behaviour of the FDAS in fault-free and faulty conditions.

The evaluation of the FDU was conducted using the full set of data acquired during experiments with URIS. Despite the poor quality of training data, results show that the FDU identifies the new thruster state correctly, in a short time after the change in state, without any wrong detection and false alarms. It was not possible to perform a full set of tests with faulty thrusters to train the FDU for FALCON. These tests will be performed in follow-on work and results will be published at a later date.

Analysing the simulation results and real-world tests, presented in this chapter, the following conclusions can be drawn:

- The hybrid approach for control allocation, implemented in the FDAS, allocates the exact, feasible solution of the control allocation problem on the entire attainable command set. This solution minimises the control energy cost function, the most suitable optimisation criteria for underwater applications. In these cases the actual behaviour of the vehicle is the same as the desired behaviour, without undesirable effects of thruster velocity saturation.
- In cases when the command inputs lies outside the attainable command set, the exact, feasible solution does not exists and the hybrid approach provides different approximations with different approximation errors. The FPI approximation has the smallest magnitude error and the largest direction error. In contrast, the S -approximation has the largest magnitude error and zero direction error. Situations when the command inputs extend over the boundaries of the attainable command

set can be signalised by the FDAS in different ways (graphically or using scalar indicators). These situations should be avoided by the ROV pilot, since they lead to the uncontrollable state of the vehicle.

- The performance of the ROV control system, equipped with the FDAS, is satisfactory for both thruster models (affine and bilinear), although the control allocation problem was formulated assuming the affine model. Reduction of the thrust, obtained in the case of bilinear thruster model, is acceptable in typical underwater applications.
- Undesired effects of propeller torque on motion in the horizontal plane can be reduced by careful choice of propeller spin direction: the symmetrical pairs of thrusters should have opposite spin direction coefficients.
- Undesired change of heading, caused by propeller torque exerted by a vertical thruster, can be eliminated using the "Heading-Keeping" controller.
- The FDAS automatically compensates for non-symmetrical shape of propeller T - curves in an optimal way.
- In the presence of a single partial or total fault in a horizontal thruster, the FDAS-controlled faulty vehicle is still fully controllable in all three DOF in the horizontal plane with very good performance. An unavoidable effect is shrinking of the attainable command set and the drop in velocity for cases when the command input vector is not attainable.
- The performance of the improved control architecture ("FDAS active") is superior compared to the control architecture "FDAS not active" in the presence of partial or total fault in a single horizontal thruster or consecutive faults in different horizontal thrusters.



- The effect of neglected thruster control loop dynamics is the appearance of a small delay between desired and actual behaviour of the vehicle, which is acceptable in most ROV applications.

Chapter 7: Conclusions and Further Work

7.1 *Introduction*

This chapter reviews and summarises the work described in the thesis, lists and discusses the main contributions and proposes suggestions for further work.

7.2 *Review of the thesis*

Chapter 1 is an introductory chapter, which provides background and motivation for work described in the thesis, identifies aims and objectives, lists the main contributions and gives an overview of chapter contents.

A literature survey is undertaken in Chapter 2 in order to explore the existing methods for fault diagnosis and accommodation of dynamic systems. Previous work on fault diagnosis and accommodation for underwater vehicles is discussed in more detail. Assumption of zero output of faulty sensor and absence of total solution for partial thruster faults are identified as weak points of the existing approaches. Due to similarity between the control allocation problem formulation for underwater vehicle and aircraft, recent advances in the control allocation techniques for aircraft are described in more detail. These techniques were used in Chapter 5 as a foundation to build a novel thruster fault diagnosis and accommodation system for overactuated, open-frame underwater vehicles.

Experimenting with real underwater vehicles is time consuming and expensive. When designing advanced control systems, it is desirable to develop a dynamic model of the vehicle, which is useful for simulation purposes and investigation of different control algorithms. Chapter 3 provides background into modelling and simulation of ROVs. The full, non-linear ROV dynamic model in 6 DOF is described in a compact, vectorial form. Three different attitude representations (Euler angle attitude representation, Euler parameters attitude representation and Modified Rodrigues parameters attitude representation) are discussed. Simulation diagrams for all three attitude representations

are developed, enabling easy simulation of ROV dynamics and kinematics. Two mostly used thruster models (affine and bilinear) are described and presented in a vectorial form, suitable for simulation purposes. A novel approach was proposed, whereby the affine thruster model is transformed into a symmetrical form, enabling easier normalisation and visualisation of the control allocation problem. A full thruster model is described, including dynamics of the thruster control loop. This model was used as a basis to build a realistic model of the propulsion system, which is implemented as a part of the ROV simulator.

The geometry of the general control allocation problem is fully exploited in Chapter 4. The general formulation of the control allocation problem was used to establish the criteria for separation of system control architecture into two independent tasks (control law and control allocation), thereby allowing the control allocation to be considered separately from the control law. Treating control allocation independently of the control law is convenient because actuator constraints can be taken into account and reconfiguration can be performed, without having to redesign the control law, providing easy adaptation to a particular application. Due to separation principle, the control allocation algorithm is the same for ROVs and AUVs. The task of the control allocator in both cases is to determine appropriate control settings for individual actuators, which produce the desired set of forces and moments. Actuator dynamics and non-monotonic nonlinearities are identified as the two most important difficulties for using control allocation in real applications. A number of methods for the solution of the general control allocation problem were presented, including the pseudoinverse method, fixed-point iteration method, direct control allocation method and daisy chain control allocation method. The simple control allocation problem, with the two-dimensional virtual control space and the three-dimensional true control space, is used as a common example to

demonstrate main features of each method. Key features and limitations of each method are given with clear geometric interpretation. The hybrid approach for control allocation is gradually introduced. The pseudoinverse method is a member of a family of generalised inverses and is the one that yields minimum control energy. The main disadvantage of the pseudoinverse method is its inability to find the exact solution of the control allocation problem on the entire attainable command set, i.e. the feasible region for pseudoinverse is a subset of attainable command set. In contrast, the fixed-point iteration method can find the exact solution on the entire attainable command set. The price paid is the necessity to perform iterations at each program step. The number of iterations depends on design parameters and choice of initial iteration. The hybrid approach for control allocation originates from the integration of positive features of the pseudoinverse and fixed-point iteration method: the pseudoinverse method is used for cases when control inputs lie inside the feasible region for pseudoinverse, and the fixed-point iterations method is used otherwise.

The control allocation problem for overactuated, open-frame underwater vehicles was defined in Chapter 5, using concepts and principles developed in Chapter 4. The problem was defined in normalised form, in order to make it more understandable and easier to visualize and solve. The hybrid approach for control allocation, introduced in Chapter 4, is extended for the case of the three-dimensional virtual control space and the four-dimensional true control space and used as a foundation to build an enhanced control allocator, with fault detection and accommodation capabilities. The hybrid approach for control allocation is implemented as a two-step process. The pseudoinverse solution is found in the first step. Then the feasibility of the solution is examined analysing its individual components. If violation of actuator constraint(s) is detected, the fixed-point iteration method is activated in the second step. In this way, the hybrid approach is able to

allocate the exact solution, optimal in the l_2 sense, inside the entire attainable command set. This solution minimises a control energy cost function, the most suitable criteria for underwater applications. A novel thruster fault detection and accommodation system for overactuated open-frame underwater vehicles was presented, which includes two subsystems: FAS and FDS. The FAS performs a hybrid approach for control allocation. This primary task of control allocation is enhanced with the FDS, able to monitor the state of the thrusters and inform the FAS about any malfunctions using the total fault indicator vector, carrying the codes of faulty states for each thruster. The FDS is a hybrid, on-line, model-free approach, based on the integration of SOM and fuzzy C -means clustering methods. In the training phase the FDS uses data obtained during test trial to find SOM prototypes for each fault type. In the detection phase the FDS categorises the fault type by comparing the position of feature vector relative to these maps. The FAS uses information provided by the FDS to accommodate faults by performing an appropriate reconfiguration, i.e. to reallocate control energy among operable thrusters. Using the fault code table, the FAS penalises the faulty thruster by increasing the corresponding weight in the weighting matrix, updating the criteria and restricting the saturation bounds. Despite the fact that in some cases it is necessary to perform iterations, the overall fault diagnosis and accommodation process is very fast, due to the computational efficiency of the FDAS algorithm, where the heaviest numerical calculations are performed off-line. This aspect of computational efficiency, combined with the adoption of a matrix formulation of the control allocation problem, means that the addition of the FDAS can be accomplished without the need to extend the cycle time.

Chapter 6 evaluated the performance of the FDAS and highlighted its key features. Using simulations and real-world tests, the performance was evaluated through a series of representative test cases, in order to examine the behaviour of the FDAS in fault-free and

faulty conditions. Evaluation of the FDU is conducted using the full set of data acquired during experiments with URIS. Despite the poor quality of training data, results show that the FDU identifies the new thruster state correctly in a short time after the change in state, without any wrong detection and false alarms. Simulation results show that the FDAS allocates the exact, feasible solution of the control allocation problem on the entire attainable command set. Using different indicators and visualisation tools, the FDAS informs the ROV pilot about position of actual command inputs relative to attainable command set. Using this information, even an inexperienced ROV pilot is able to detect the situation when thruster velocity saturation occurs and to correct the command inputs such that it becomes attainable. Results also show that the performance of the ROV control system, equipped with the FDAS, is satisfactory for both thruster models (affine and bilinear), although the control allocation problem was formulated assuming the affine model. Correct choice of propeller spin direction, such that the symmetrical pairs of thrusters should have opposite spin direction coefficients, reduces undesired effects of propeller torque on motion in the horizontal plane. Undesired change of heading, caused by propeller torque exerted by a vertical thruster, can be eliminated using the "Heading-Keeping" controller. The FDAS automatically compensates for non-symmetrical shape of propeller T -curves in an optimal way, making the command input space to appear symmetrical from the ROV pilot point of view. Preliminary results of real-world tests show that disabling the faulty thruster without appropriate reallocation leads to poor tracking performance and oscillatory character of yaw response. However, the FDAS provides automatic reallocation in faulty situations, keeping all three DOF in the horizontal plane fully controllable and making it easier to control the motion of the faulty vehicle and continue the mission. The control allocation problem was formulated and solved neglecting dynamics in thruster control loop. Results from real-world application

show that the effect of neglected thruster control loop dynamics is the appearance of a small delay between desired and actual behaviour of the vehicle, which is acceptable in most ROV applications.

7.3 Realisation of aims and objectives

This section discusses accomplishments in realisation of the aims and objectives, given in section 1.3.

7.3.1 Aims

The thruster fault diagnosis and accommodation system, proposed in Chapter 5 and implemented in the ROV simulator (Appendix D), fulfil the following requirements:

- In fault-free case, optimal control allocation is guaranteed for all possible command inputs, since the hybrid approach for control allocation finds the exact solution on the entire attainable command set. This solution is optimal in the l_2 sense, i.e. it minimises a control energy cost function.
- In faulty situations, the fault diagnosis part of the system immediately detects and isolates any fault in a thruster using fault detection units and delivers knowledge about faults in the form of a fault indicator vector. The fault accommodation part of the system uses this vector to accommodate fault and eventually switch off the faulty thruster. At the same time, control reallocation is performed by redistribution of control energy among remaining operable thruster, such that the mission can be continued with a minimal loss of control performance.

7.3.2 Objectives

- **Explore the existing methods for fault diagnosis and accommodation in dynamic systems.** An extensive literature survey is undertaken in Chapter 2. This

chapter provides an overview of traditional and modern approaches to fault diagnosis and accommodation in dynamic systems.

- **Identify which of these methods are applicable to meet requirements and specific implementation issues, defined in the IMPROVES project.** Requirements (section 5.1.2) and implementation issues (section 5.1.3) imposed certain limits on the choice of available approaches and restricted the designer freedom. A hybrid, on-line, model-free approach, based on the integration of SOM and fuzzy *C*-means clustering methods, was chosen to build the fault diagnosis part of the system. The fault accommodation part uses the hybrid approach for control allocation, which utilizes the best features of the pseudoinverse method and fixed-point iterations method.
- **Develop module for detection of thruster faults.** The fault diagnosis subsystem uses fault detection units (section 5.4.3) to monitor the state of thrusters. These units are software modules, able to detect and isolate external and internal thruster faults.
- **Formulate and solve the control allocation problem for fault-free case.** The control allocation problem for overactuated, open-frame underwater vehicles is formulated in section 5.2. The solution of the problem is presented in section 5.5.2.
- **Extend the algorithm to cover faulty situations.** Introducing weights in problem formulations (section 5.2.5) provides a framework to accommodate thruster faults. The faulty thruster is penalised by increasing its weight and restricting the constraint bounds.
- **Develop the simulation model to test the algorithm.** The ROV simulator (Appendix D) is developed as Simulink model in the MATLAB environment in

order to test the FAS algorithm. The model includes the non-linear model of an ROV with 6 DOF, propulsion system, hand control unit (HCU) and thruster fault detection and accommodation system (FDAS), proposed in Chapter 5. Simulation results are presented in Chapter 6.

- **Verify the performance of the algorithm with real-world applications.** Preliminary results from the real-world application, presented in Chapter 6, confirm that the extension of the existing control system with the FAS algorithm leads to significant improvements in the global control performance of the vehicle in faulty situations.

7.4 *Main contributions*

This section discusses the main contributions of the work presented in the thesis.

- **Development of the on-line fault detector units, able to detect external and internal thruster faults.** The fault detection units are developed using a new, hybrid, on-line, model-free approach, based on the integration of SOM and fuzzy C-means clustering methods. In the training phase fault detection units use data obtained during test trial to find SOM prototypes for each fault type. In the detection phase the FDU algorithm is employed, which categorises the fault types by comparing the position of feature vector relative to these maps. Both external and internal thruster faults can be detected by FDU. The outputs of the FDUs are integrated inside the FDS into the total fault indicator vector, carrying the codes of faulty states for each thruster.
- **Development of the hybrid approach for control allocation, able to allocate optimal solutions of the control allocation problem in fault-free and faulty situations.** The hybrid approach for control allocation, based on the integration of the pseudoinverse and the fixed-point method, is implemented as a two-step

process in the FAS algorithm. The weighted pseudoinverse solution is found in the first step. Then the feasibility of the solution is examined analysing individual components of the solution. If violation of actuator constraint(s) is detected, the fixed-point iteration method is activated in the second step, which results in guaranteed feasible solution. In this way the hybrid approach is able to allocate the exact solution, optimal in the l_2 sense, inside the entire attainable command set. This solution minimises the control energy cost function, which is the most suitable criteria for underwater applications. Formulation of the control allocation problem using weighted norms provides an easy way to accommodate thruster faults, which is performed by increasing the weight of a faulty thruster and restricting the corresponding constraint bounds. Real-world application confirmed significant improvements in the manoeuvring capabilities of the faulty vehicle, obtained by augmentation of the existing control system with the FAS algorithm.

- **Visualisation of thruster velocity saturation bounds using the feasible region concept.** Visualisation of thruster velocity saturation bounds, i.e. attainable command set, implemented as part of the FDAS, provides insight into constraints imposed by thruster configuration. During missions, the ROV pilot generates command inputs, which stretch out over the command space in different directions. Thruster velocity saturation occurs in some directions easier than in other. The position of saturation bounds inside the command space is determined by thruster configuration. In order to avoid thruster velocity saturation, command inputs should not stretch outside the saturation bounds. Any fault in a thruster causes a change in the shape of the attainable command set. The FDAS provides a dynamic update of saturation bounds such that the ROV pilot is informed with the effects of thruster fault accommodation, incorporated in the new shape of the

attainable command set. In this way the ROV pilot can easily adapt to newly created changes and continue the mission.

- **Formulation of the control problem in normalised form.** Normalisation of the control allocation problem for underwater vehicles makes it more understandable and easier to visualize and solve. During normalisation, all physical parameters are removed from the control effectiveness matrix and included in limit constraints, which are used to scale individual components of control vectors. The normalisation process is not restricted to underwater applications and can be applied to simplify the control allocation in other fields, such as aerospace applications. The normalisation also allows the FDAS algorithm to be easily adapted to vehicles with different thruster configurations, i.e. vehicles with different number of thrusters with varying orientations and positions.
- **Development of a method to compensate for non-symmetrical propeller T -curve.** Normalisation of the control allocation problem can not be performed for the case of a non-symmetrical propeller T -curves. In order to compensate for non-symmetry, a new method has been developed, which introduces an auxiliary control variable, making the propeller T -curve temporarily symmetrical. In this way, the control allocation problem becomes linear and easy to solve using existing techniques.
- **Design of a simulation model with virtual reality display.** The ROV simulator can be used as a foundation to build advanced control systems and to compare different control strategies in future development of AUVs. With some improvements of the user interface, it may also be used for training purpose. A virtual reality display raises the level of graphical presentation into the new

dimension. Real-world situations can be easily simulated, without the need to perform time-consuming and expensive trials.

7.5 *Further work*

The proposed FDAS is a part of the low-level control layer and its modular design enables easy integration into existing control laws. Further work will involve the following activities:

- A full set of tests with faulty thrusters, similar to those described in section 5.4.3, will be repeated with FALCON, and the obtained data will be used to train the FDS part of the FDAS.
- Integration of the FDAS into existing control architectures

URIS: The FDS will be built, using the FDU developed in section 5.4.3. The final version of the FDAS will be integrated with the existing control system of URIS.

FALCON: The control application ATC will be updated with the full implementation of the FDAS. Then the FDAS will be incorporated into Seaeye Marine Ltd. distributed control system and tested in a real working environment.

- The global control performance of FALCON will be improved by redesigning the velocity controller inside thruster control units using advanced control techniques. The main objective will be reduction of a steady-state error, currently present in time responses of actual propeller angular velocity.
- Design of a universal controller for an AUV, robust to a partial/total fault in a single thruster, will be a real challenge. The main idea is to use information about effects of a fault in a thruster (new shape of the attainable command set) to alleviate criteria and set points in higher control layers. In this way, AUV will be

able automatically to accommodate fault, adapt to change in shape of the attainable command set and continue the mission in presence of thruster fault.

- An important part of further work will be the integration of feasible region with real-time video presented to the ROV pilot from the on-board camera.

Appendices

Appendix A: ROV models – Technical Details

Introduction

This appendix contains technical details and specifications for two ROVs that are used in the ROV simulator to evaluate the performance of the FDAS. These pages are taken from:

- http://eia.udg.es/~pere/uris/uris_auv.htm (URIS – general description)
- <http://www.maxonmotor.com> (URIS – thruster control unit)
- <http://www.seaeye.com/falcon.html> (FALCON)

All information is valid as of 13th October 2003. The material presented herein is not modified by the author.

Underwater Robotic Intelligent System (URIS)

Description

The URIS (Underwater Robotic Intelligent System) vehicle is still in the development process. It was designed with the aim of developing a small, light weight, low cost AUV¹ to be used as a research testbed in a water tank testing facility. With URIS we expect that a single person will be able to run an experiment with the vehicle. Moreover, the vehicle is small enough to transport in a conventional car. This vehicle has been conceived as an AUV, hence it carries its own source of power, which gives about an hour of autonomy. The vehicle can also be powered by an external source using an umbilical cable. This option facilitates running long-term experiments. The hull has been designed as a sphere. As a result, it offers equal hydrodynamic coefficients in any direction. There have been

¹ Throughout the thesis the URIS is considered as an ROV and test trials in July 2002 were performed using a joystick as a command input device. However, the research group at University of Girona developed control architecture such that the URIS can be treated as an AUV.

Appendix A: ROV models – Technical Details

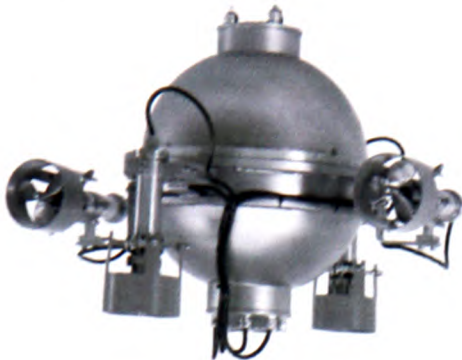
precedents with this shape, such as the ODIN AUV from the University of Hawaii-USA or the ROBIN robot from CNRIAN-Italy, as well as the HYBALL from the Heriot-Watt University-UK. The hull consists of two stainless steel hemispheres joined with wing nuts and bolts. The vehicle's mass has been distributed in a way so that the centre of mass is below the buoyancy centre (as in the GARBI vehicle) making the vehicle passively stable in *roll* and *pitch*. Propulsion is achieved with 4 thrusters placed equidistant on the vehicle's exterior. Due to the stability in *pitch* and *roll*, there are four degrees of freedom; X, Y, Z and Yaw. The vehicle incorporates a magnetic compass, a pressure sensor, water speed sensors, DGPS, water leakage sensors, computer vision and a laser-based 3D computer vision system (still in development). An on board Pentium PC 104 is in charge of the control of the vehicle's sub-systems. During the development, the optional umbilical cable is used to connect the on board computer to the surface computer where the development environment (Tornado/VxWorks) resides. The sphere radius is about 35 cm and the weight is approximately 35 kg.

Features

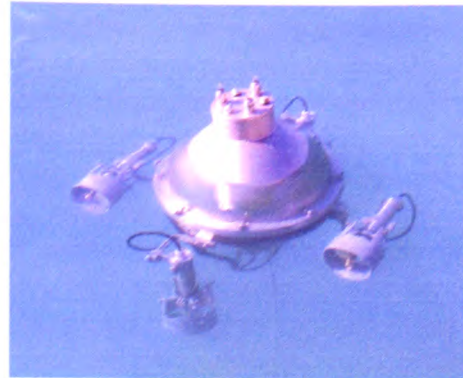
Type	Autonomous Underwater Vehicle (AUV)
D.O.F.	4 (x,y,z,Yaw)
Stability	Passively stable in Roll and Pitch
Propulsion	4 thrusters (20W x 15V DC motor + dynamo)
Energy	4 packages of NiCd batteries (50 W x 12V)
Max. depth	30 meters
Sensors	<ul style="list-style-type: none">• Magnetic compass (Yaw)• Pressure sensor (z)• Vision system (RGB+laser)• Down-looking camera• Speed sensor• DGPS• Water and battery charge detection

Appendix A: ROV models – Technical Details

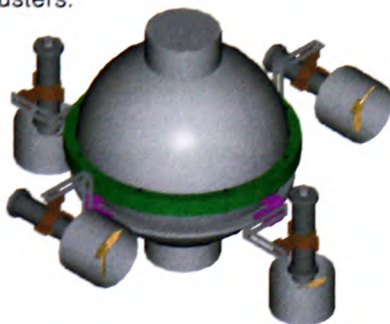
Pictures



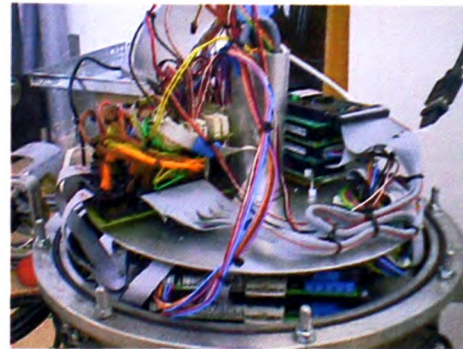
Original thruster configuration of URIS, with two horizontal and two vertical thrusters.



URIS in the water.



Model of URIS in CAD software.



Internal connections.

Components of thruster control loop

Thruster control loop for URIS consists of the following components:

- DC motor Maxon RE 25, Ø25 mm, Graphite Brushes, 20 Watt (Order Number: 118750)
- DC-Tacho Ø22 mm, 0.52 V (Order Number: 118909)
- Planetary Gearhead GP 32A, Ø32 mm, 0.75 – 4.5 Nm (Order Number: 114467)
- Thruster control unit 4-Q-DC Servoamplifier ADS 50/5 (Order number: 145391)

Technical specifications of the thruster control unit are given in the following.

The ADS 50/5 is a powerful servoamplifier for driving permanent magnet DC motors up to 250 Watts. Four modes can be selected by DIP switches on the board:

- Speed control using tacho signals
- Speed control using encoder signals
- IxR compensated speed control
- Torque or current control

The ADS 50/5 is protected against excess current, excess temperature and short circuit on the motor winding. With the FET power transistors incorporated in the servoamplifier, an efficiency of up to 95 % is achieved. A built in motor choke combined with the high PWM frequency of 50 kHz allows the connection of motors with a very low inductivity. In most cases an external choke can be omitted.

Thanks to the wide input power supply range of 12 - 50 VDC, the ADS 50/5 is very versatile and can be used with various power supplies. The aluminium housing makes installation simple, with terminal markings for easy connection.

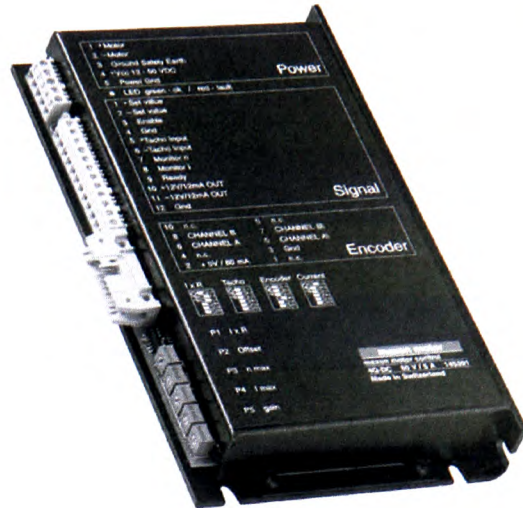


Table of Contents

1 Safety Instructions	2
2 Performance Data	3
3 Minimum External Wiring for Different Modes of Operation	4
4 Operating Instructions	5
5 Functions	7
6 Additional Possible Adjustments	10
7 Operating Status Display	12
8 Error Handling	13
9 EMC-compliant Installation	13
10 Block Diagram	14
11 Dimension Drawing	14

The latest edition of this operating instructions may be downloaded from the internet as a PDF-file under <http://www.maxonmotor.com>, category «Service», subdirectory «Downloads», Order number 145391.

1 Safety Instructions

**Skilled Personnel**

Installation and starting of the equipment shall only be performed by experienced, skilled personnel.

**Statutory Regulations**

The user must ensure that the servoamplifier and the components belonging to it are assembled and connected according to local statutory regulations.

**Load Disconnected**

For primary operation the motor should be free running, i.e. with the load disconnected.

**Additional Safety Equipment**

An electronic apparatus is not fail-safe in principle. Machines and apparatus must therefore be fitted with independent monitoring and safety equipment. If the equipment breaks down, if it is operated incorrectly, if the control unit breaks down or if the cables break, etc., it must be ensured that the drive or the complete apparatus is kept in a safe operating mode.

**Repairs**

Repairs may be made by authorised personnel only or by the manufacturer. It is dangerous for the user to open the unit or make repairs to it.

**Danger**

Do ensure that during the installation of the ADS 50/5 no apparatus is connected to the electrical supply. After switching on, do not touch any live parts.

**Max. Supply Voltage**

Make sure that the supply voltage is between 12 and 50 VDC. Voltages higher than 53 VDC or of wrong polarity will destroy the unit.

**Short circuit and earth fault**

The ADS 50/5 amplifier is not protected against winding short circuits against ground safety earth or Gnd!

**Motor choke**

The built in motor choke allows operation with almost all maxon DC motors with an output power higher than 10 Watt. For a few exceptions (A-max Ø26, 11 W as well as RE Ø25 and RE Ø26 with terminal inductance lower than 350 µH) an extra inductance (choke) of at least 200 µH is necessary.

**Electrostatic Sensitive Device (ESD)**

2 Performance Data

2.1 Electrical data

Supply voltage V_{CC} (Ripple < 5%)	12 - 50 VDC
Max. output voltage	$0.9 \cdot V_{CC}$
Max. output current I_{max}	10 A
Continuous output current I_{cont}	5 A
Switching frequency	50 kHz
Efficiency	95 %
Band width current controller	2.5 kHz
Built-in motor choke	160 μ H / 5 A

2.2 Inputs

Set value	-10 ... +10 V ($R_i = 20\text{ k}\Omega$)
Enable	+4 ... +50 VDC ($R_i = 15\text{ k}\Omega$)
Input voltage DC tachometer „Tacho Input“	min. 2 VDC, max. 50 VDC ($R_i = 14\text{ k}\Omega$)
Encoder signals „Channel A, A \bar , B, B \bar “	max. 100 kHz, TTL level

2.3 Outputs

Current monitor „Monitor I“ , short-circuit protected	-10 ...+10 VDC ($R_o = 10\text{ k}\Omega$)
Speed monitor „Monitor n“ , short-circuit protected	-10 ...+10 VDC ($R_o = 10\text{ k}\Omega$)
Status reading „READY“	
Open collector	max. 30 VDC ($I_L \leq 20\text{ mA}$)

2.4 Voltage outputs

Aux. voltage, short-circuit protected	+12 VDC, -12 VDC, max. 12 mA
Encoder supply voltage	+5 VDC, max. 80 mA

2.5 Trim potentiometers

IxR compensation	
Offset	
n_{max}	
I_{max}	
gain	

2.6 LED indicator

2 coloured LED	READY / ERROR
green = ok, red = error	

2.7 Ambient temperature- / Humidity range

Operating	-10 ... +45°C
Storage	-40 ... +85°C
Non condensating	20 ... 80 %

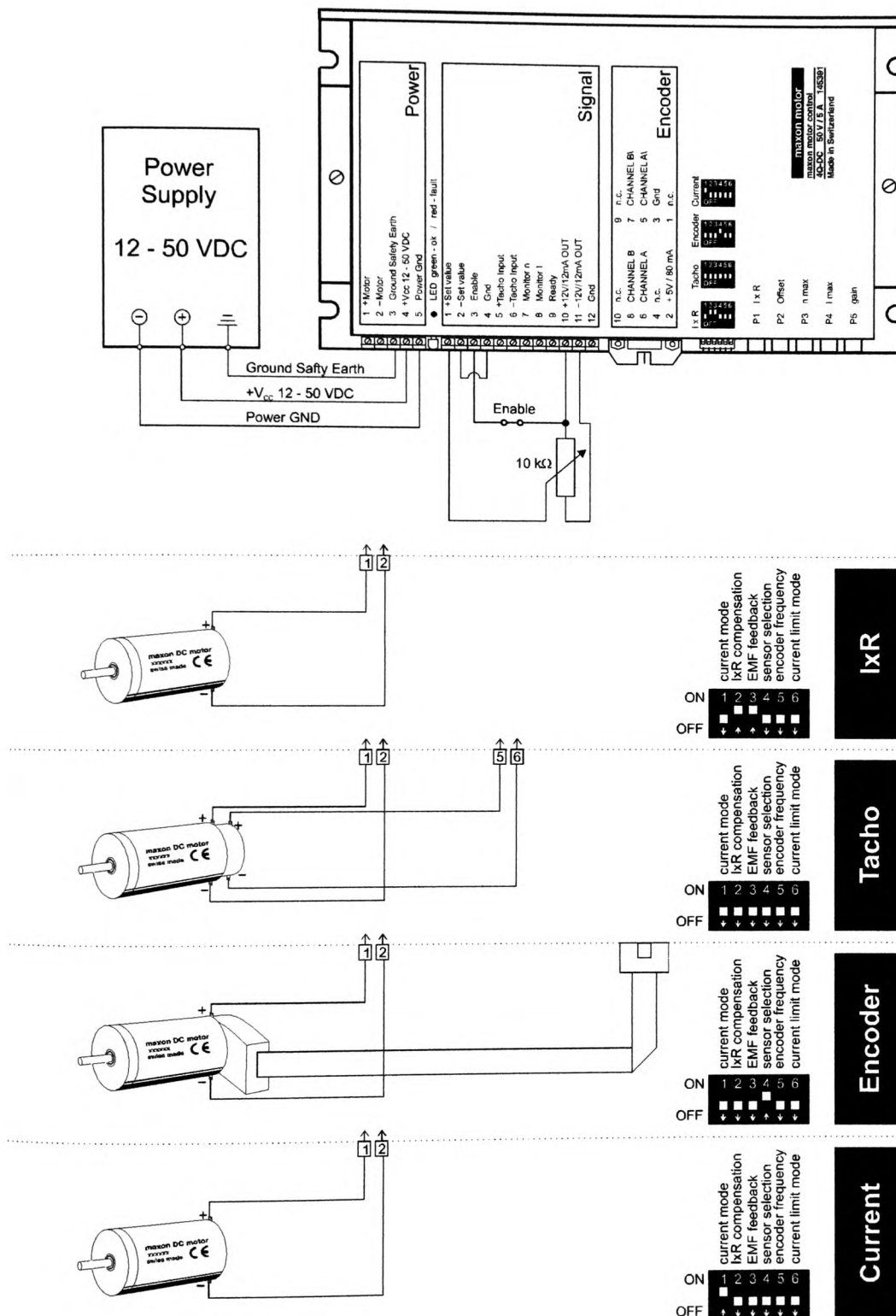
2.8 Mechanical data

Weight	ca. 360 g
Dimensions	see dimension drawing
Mounting plate	for M4 screws

2.9 Terminal

PCB-clamps	Power (5 poles), Signal (12 poles)
Pitch	3.81 mm
suitable for wire cross section	0.14 - 1 mm ² multiple-stranded wire or
	0.14 - 1.5 mm ² single wire
Encoder	Plug DIN41651
	for flat cable, pitch 1.27 mm, AWG 28

3 Minimum External Wiring for Different Modes of Operation



4 Operating Instructions

4.1 Determine power supply requirements

You may make use of any available power supply, as long as it meets the minimal requirements spelled out below.

During set up and adjustment phases, we recommend separating the motor mechanically from the machine to prevent damage due to uncontrolled motion.

Power supply requirements

Output voltage	V_{CC} min. 12 VDC; max. 50 VDC
Ripple	< 5 %
Output current	depending on load, continuous 5 A (short-time 10 A)

The required voltage can be calculated as follows:

Known values:

- Operating torque M_B [mNm]
- Operating speed n_B [rpm]
- Nominal motor voltage U_N [Volt]
- Motor no-load speed at U_N , n_0 [rpm]
- Speed/torque gradient of the motor $\Delta n / \Delta M$ [rpm/mNm⁻¹]

Sought values:

- Supply voltage V_{CC} [Volt]

Solution:

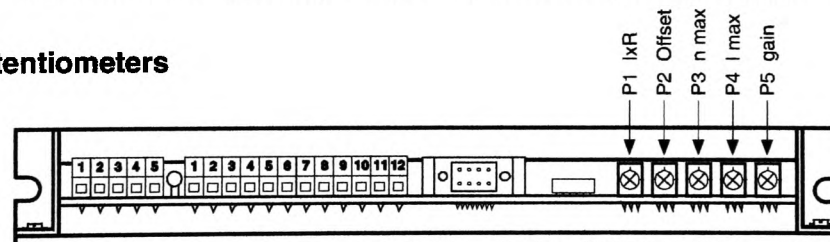
$$V_{CC} = \frac{U_N}{n_0} \cdot \left(n_B + \frac{\Delta n}{\Delta M} \cdot M_B \right) \cdot \frac{1}{0.9} + 2 [V]$$



Choose a power supply capable of supplying this calculated voltage under load. The formula takes into account a max. PWM cycle of 90 % and a 2 volt max. voltage drop.

Consider:

The power supply must be able to buffer the back-fed energy from brake operation e.g. in a condenser. With electronically stabilized power supply units it is to ensure, that the overcurrent protection responds in no operating condition.

4.2 Function of the potentiometers

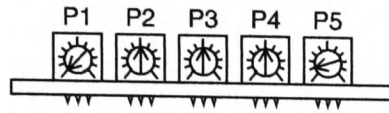


Potentiometer		Function	Turn to the	
			left 	right 
P1	I _{xR}	I _{xR} compensation	weak compensation	strong compensation
P2	Offset	Adjustment $n = 0$ at set value 0 V	motor turns CCW	motor turns CW
P3	n_{max}	max. speed at 10 V set value	speed slower	speed faster
P4	I_{max}	current limit	lower min. 0.5 A	higher max. 10 A
P5	gain	amplification	lower	higher

4.3 Adjustment of the Potentiometers

4.3.1 Pre-adjustment

With the pre-adjustment, the potentiometers are set in a preferred position.
ADS units in original packing are already pre-adjusted.



Pre-adjustment of potentiometers		
P1	I _{xR}	0 %
P2	Offset	50 %
P3	n _{max}	50 %
P4	I _{max}	50 %
P5	gain	10 %

4.3.2 Adjustment

Encoder mode
DC-Tacho mode
I_{xR} compensation

- Adjust set value to maximum (e.g. 10 V) and turn potentiometer **P3** n_{max} so far that the required speed is achieved.
- Set potentiometer **P4** I_{max} at the limiting value desired. Maximum current in the 0 ... 10 A range can be adjusted in linear fashion with potentiometer **P4**.
Important: The limiting value I_{max} should be below the max. continuous current as shown on the motor data sheet and may not exceed 5 A continuously.
- Increase potentiometer **P5** gain slowly until the amplification is set large enough.
Caution: If the motor vibrates or becomes loud, the amplification is adjusted too high.
- Adjust set value to 0 V, e.g. by short circuiting the set value. Then set the motor speed to 0 rpm with the potentiometer **P2** Offset.

In addition, only in the case of I_{xR} compensation:

- Slowly increase potentiometer **P1** I_{xR} until the compensation is set large enough so that in the case of high motor load the motor speed remains the same or decreases only slightly.
Caution: If the motor vibrates or becomes loud, the amplification is adjusted too high.

Current controller mode

- Set potentiometer **P4** I_{max} at the limiting value desired. Maximum current in the 0 ... 10 A range can be adjusted in linear fashion with potentiometer **P4**.
Important: The limiting value I_{max} should be below the max. continuous current as shown in the motor data sheet and may not exceed 5 A continuously.

Note 1

A set value in the -10 ... +10 V range is equal to a current range of approx. +10 ... -10 A

Note 2

Configured as a current controller, P1, P2, P3 and P5 are not activated.

5 Functions

5.1 Inputs

5.1.1 Set value

The set value input is wired as a differential amplifier.

Input voltage range	-10 ... +10 V
Input circuit	differential
Input resistance	20 k Ω (differential)
Positive set value	(+ Set Value) > (- Set Value) negative motor voltage or current motor shaft turns CCW
Negative set value	(+ Set Value) < (- Set Value) positive motor voltage or current motor shaft turns CW

5.1.2 Enable

If a voltage is given at "Enable", the servoamplifier switches the motor voltage to the winding connections. If the "Enable" input is not switched on or is connected to the Gnd, the power stage will be highly resistant and will be disabled. The "Enable" input is short-circuit protected.

Enable	Minimum input voltage	+ 4.0 VDC
	Maximum input voltage	+ 50 VDC
	Input resistance	15 k Ω
	Switching time	typ 500 μ s (by 5 V)
Disable	Minimum input voltage	0 VDC
	Maximum input voltage	+ 2.5 VDC
	Input resistance	15 k Ω
	Switching time	typ 100 μ s (by 0 V)

5.1.3 DC Tacho

Minimum input voltage	2.0 V
Maximum input voltage	50 V
Input resistance	14 k Ω

Speed control range:

The speed range is set using Potentiometer **P3** n_{max} (max. speed at maximum set value).

For full speed control with ± 10 V, the tacho input voltage range must be at least ± 2 V.

Example for DC-Tacho with 0.52 V / 1000 rpm:

2.0 V tacho voltage is equivalent to a speed of approx. 3850 rpm. If the full set value range has been used, the lowest adjustable speed with the n_{max} potentiometer is 3850 rpm.

Lower speed ranges can be reached through a reduced set value range or by using a DC tacho with a higher output voltage, such as 5 V / 1000 rpm.

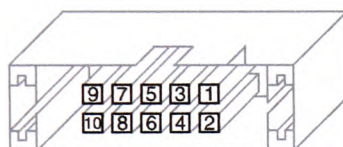
5.1.4 Encoder

Encoder supply voltage	+ 5 VDC max. 80 mA	
Maximum encoder frequency	DIP - Switch 5 ON: 10 kHz	
	DIP - Switch 5 OFF: 100 kHz	
Voltage value	TTL	
	low	max. 0.8 V
	high	min. 2.0 V

It is strongly recommended that the encoder be used with a built-in line driver. If the encoder is used **without** a line driver (without ChA\ and ChB\), speed breakdowns and max. speed limits must be expected because of the slower switching slope.

The servoamplifier does not need any home impulse I and I\.

Male header (front view)



Pin configuration at "Encoder" input:

1	n.c.	Not connected
2	+5 V	+ 5 VDC max. 80 mA
3	Gnd	Ground
4	n.c.	Not connected
5	A\	Inverted Channel A
6	A	Channel A
7	B\	Inverted Channel B
8	B	Channel B
9	n.c.	Not connected
10	n.c.	Not connected

This pin configuration is compatible with the flat cable plugs in Encoder HEDL 55xx (with Linedriver) and the MR encoders with line driver, type ML and L.

5.2 Outputs

5.2.1 Current monitor „Monitor I“

The servoamplifier makes a current actual value available for monitoring purposes. The signal is proportional to the motor current.

The „Monitor I“ output is short-circuit protected.

Output voltage range	-10 ... +10 VDC
Output resistance	10 k Ω
Gradient	approx. 0.8 V/A
positive voltage on current monitor output	corresponds to a negative motor current
negative voltage on current monitor output	corresponds to a positive motor current

5.2.2 Speed monitor „Monitor n“

The speed monitor is primarily intended for the qualitative estimation of the dynamics. The absolute speed is determined by the properties of the speed sensors and by the setting of the n_{\max} potentiometer. The output voltage of the speed monitor is proportional to the number of revolutions. The output voltage of the speed monitor is 10 V when the maximum number of revolutions set by the n_{\max} potentiometer has been reached.

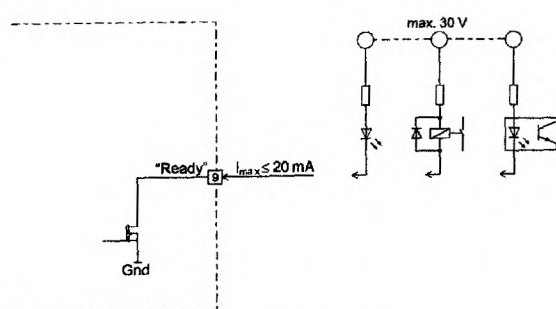
The „Monitor n“ output is short-circuit protected.

Output voltage range	-10 ... +10 VDC
Output resistance	10 k Ω

Example:	-10 V	corresponding speed	$-n_{\max}$	(CCW)
	0 V	corresponding speed	0 rpm	
	+10 V	corresponding speed	$+n_{\max}$	(CW)

5.2.3 Status reading „Ready“

The „Ready“ signal can be used to report the state of operational readiness or a fault condition on a master control unit. The „Open Collector“ output is, in normal cases, i.e., no faults, switched to Gnd. In the case of a fault with excess temperature or excess current, the output transistor is not conducting (high resistance).








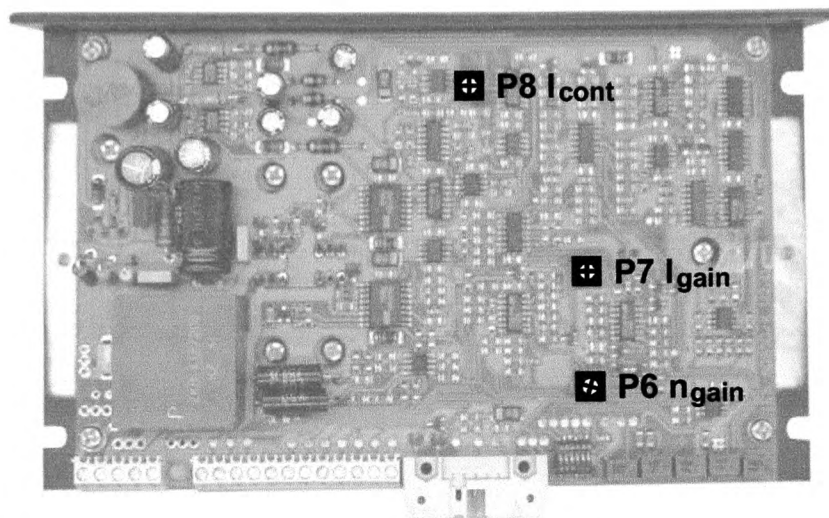
An external additional voltage is required:

Input voltage range	max. 30 VDC
Load current	≤ 20 mA

The fault condition is stored. In order to reset the fault condition, the servoamplifier must be re-released (Enable). If the cause of the fault situation cannot be removed, the output transistor will immediately change to the not conducting state again.

6 Additional Possible Adjustments

	Potentiometer		Function	Position	
				left 	right 
	P6	n_{gain}	speed gain	low	high
	P7	I_{gain}	current gain	low	high
	P8	I_{cont}	continuous current limit	lower	higher



6.1 Adjustments potentiometer P6 n_{gain} and potentiometer P7 I_{gain}

In most applications, regulation setting is completely satisfactory using potentiometers P1 to P5. In special cases the transient response can be optimized by setting the P6 "speed regulation gain" potentiometer. The P7 "current regulator gain" potentiometer can, in addition, be adapted to the dynamics of the current regulator.

It is recommended that the success of changes to the settings of P6 n_{gain} and P7 I_{gain} be checked by measuring the transient response with an oscilloscope at the "Monitor n" and "Monitor I" outputs.

Pre-adjustment P6 $n_{\text{gain}} = 25 \%$ and P7 $I_{\text{gain}} = 50 \%$.

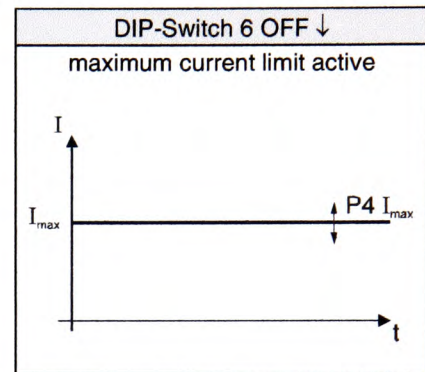
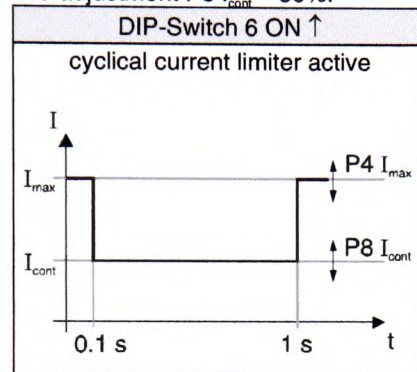
6.2 Adjustments potentiometer P8 I_{cont} and current limit mode DIP-Switch 6

It is standard that a maximum current limiter is activated (DIP switch 6 OFF). In this way the motor current is limited to the value set on potentiometer P4 I_{max} (0.5 ... 10 A).

If DIP switch 6 is turned to ON, a cyclical current limiter is also activated. This current limiter method makes a certain level of motor protection against thermal overload possible.

For 0.1 seconds the motor current is limited to the value set on potentiometer P4 I_{max} (0.5 ... 10 A) and then for 0.9 seconds current is limited to the value set on potentiometer P8 I_{cont} (0.5 ... 10 A). After one second the cycle will repeat itself.

Pre-adjustment P8 I_{cont} = 50%.



6.3 Maximal encoder frequency DIP-Switch 5

DIP switch 5 permits selection of the maximum encoder input frequency. A max. encoder frequency of 100 kHz is standard.

DIP-Switch 5 ON ↑	
Max. Input frequency is 10 kHz	
Encoder pulse per turn	maximum motor speed
16	37 500 rpm
32	18 750 rpm
64	9 375 rpm

DIP-Switch 5 OFF ↓	
Max. Input frequency is 100 kHz	
Encoder pulse per turn	maximum motor speed
100	60 000 rpm
500	12 000 rpm
1000	6 000 rpm

Note:

To achieve good control characteristics, encoders with low impulse counts per turn should be run with the DIP switch 5 ON ↑.

7 Operating Status Display

A two coloured red/green LED shows the operating mode.

7.1 No LED

Reason:

- No supply voltage
- Fuse fault
- Wrong polarity of supply voltage

7.2 LED shines green

- Supply voltage applied
- No error status (overheating or overcurrent)

7.3 LED shines red

- If the power stage temperature exceeds a limit of approx. 75°C, the power stage is switches off (Disable-status). The LED shows red.
- If a motor current of more than approx. +/- 12.5 A is detected at the current actual value, the power stage will be switched off (disable status). The LED shows red.

The fault condition is stored. In order to reset the fault condition, the servoamplifier must be re-released (Enable). If the cause of the fault condition cannot be eliminated, the error output will be disabled again immediately.

Reason:

- High ambient temperature
- max. continuous current > 5 A
- bad convection
- Short circuit on the motor winding

8 Error Handling

Defect	Possible source of defect	Measures
Shaft does not rotate	Supply voltage <12 VDC	check power plug pin 4
	Enable not activated	check signal plug pin 3
	Set value is 0 V	check signal plug pin 1 and pin 2
	Current limit too low	check adjustment pot. P4 I _{max}
	Wrong operational mode	check DIP switch settings
	Bad contacts	check wiring
	Wrong wiring	check wiring
Speed is not controlled	Encoder mode: encoder signals	check plug encoder
	DC-Tacho mode: tacho signals	check plug signal pin 5 and 6 (polarity)
	IxR mode: compensation wrong	check adjustment pot. P1

9 EMC-compliant Installation

HF blocking

HF current blocking generally improves resistance to interference compared to external interference couplings by means of a ferrite toroidal core in a line (power or signal line).

Shield earthing

The earth impedance must have the lowest possible resistance.

Connecting cable

Power and signal lines must generally be installed as screened lines on a low-coupling basis and without looping.

"Power" terminal clamps

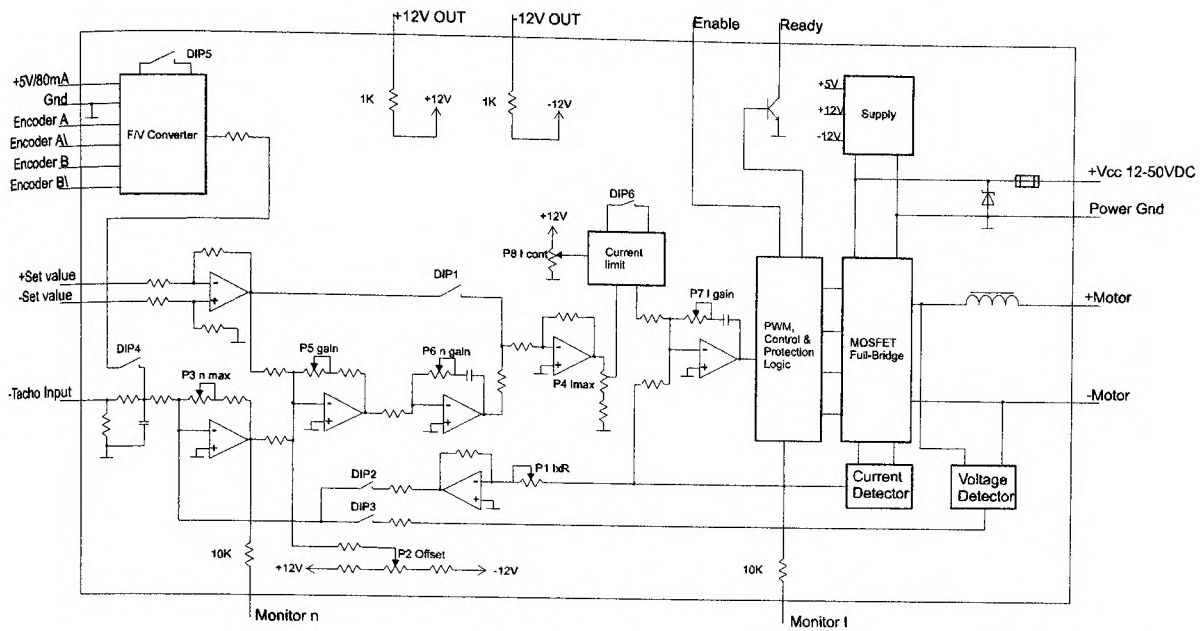
A screened cable should be used to minimise noise emissions. The screen must be connected to the amplifier side and also coupled with the motor on the motor side, via the plug casing wherever possible.

"Signal" terminal clamps

Signal lines for sensitive analogue signals must also be screened. The signal lines' screen should be earthed on one side, the amplifier's side.

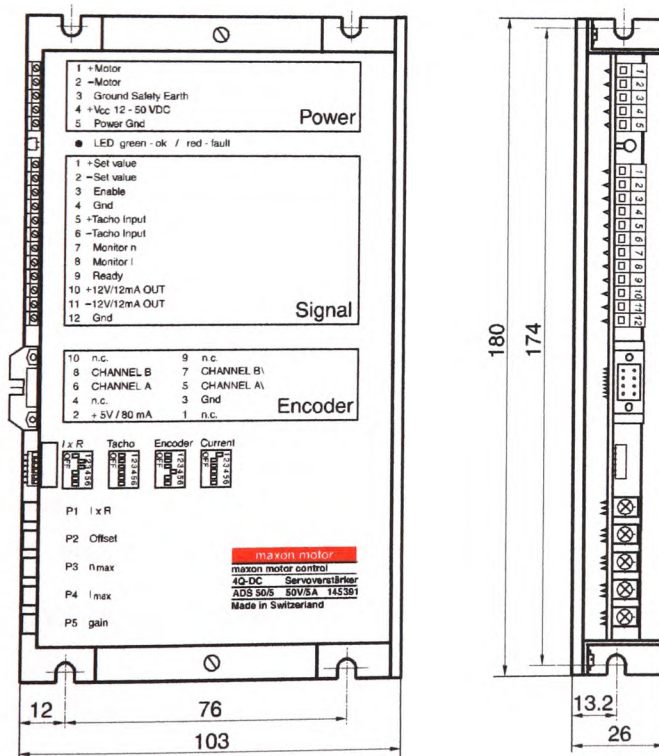
In practical terms, only the complete unit, comprising all individual components (motor, amplifier, power supply unit, EMC filter, cabling etc) can be subjected to an EMC test in order to ensure noise-free and CE-approved operation.

10 Block Diagram

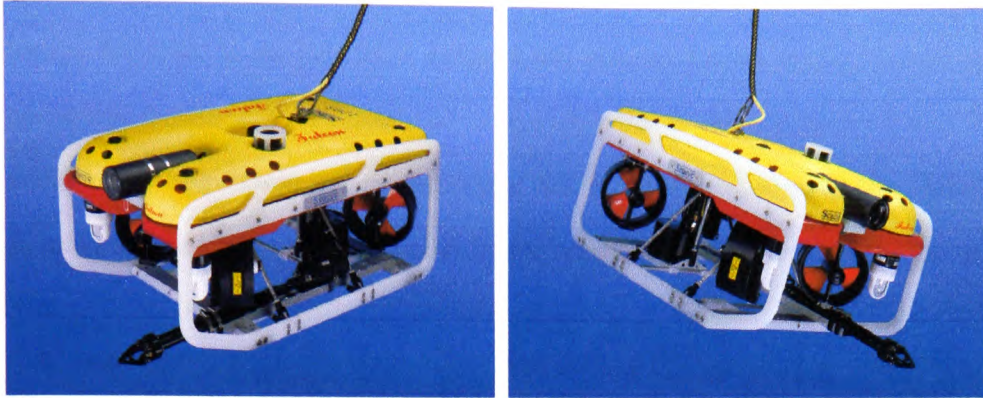


11 Dimension Drawing

Dimensions in [mm]



FALCON



Seaeye FALCON Features:

- 300 metre depth rating, 16 kilo payload,
- Magnetically coupled brushless DC thrusters with velocity feedback loop,
- 4 Vectored and 1 vertical thruster,
- 50 kgf thrust with 1:1 power to weight ratio,
- Distributed intelligence control system,
- Integral system diagnostics,
- High resolution colour camera on 180° Tilt Platform,
- Variable intensity 150 watts of lighting,
- Auto heading, depth, compass and rate gyro,
- Portable surface control system with video overlay and daylight readable display,
- Low drag umbilical,
- Single phase A/C power input - universal 100-270 VAC at 2.5 kw.

Introduction

Since 1986, Seaeye Marine's focus has been to supply the most powerful and reliable electric ROVs for operations in the challenging environment of the international offshore

Appendix A: ROV models – Technical Details

oil and gas industry. Today, Seaeye has the widest range of electric powered systems operating worldwide and the largest share of this demanding market.

Seaeye FALCON is the first ROV designed and built by Seaeye Marine to meet the operational requirements of coastal and inshore operators. In developing this system the same criteria of performance, reliability and ease of operation were set, but with the added challenge of creating a system that was to be portable and affordable.

Seaeye FALCON & its Surface Units

The Seaeye FALCON incorporates many of the features that have proved so successful in other Seaeye ROVs but with a number of technical innovations never before seen in a lower cost ROV. The Seaeye FALCON sets a new standard for inshore and coastal operations to 300 metres depth.

The vehicle

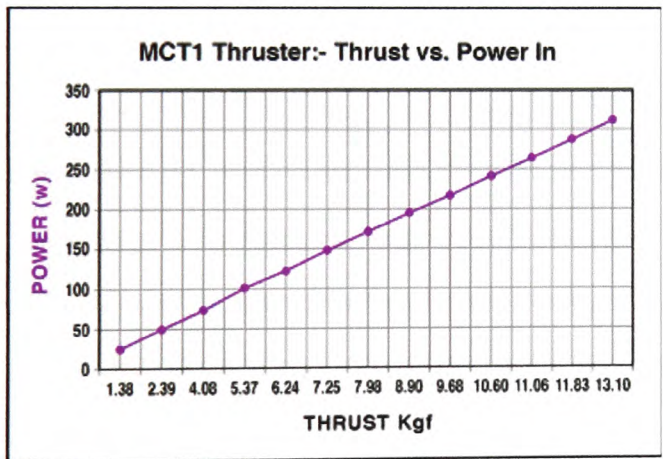
Vehicle Specifications	
Maximum Working Depth:	300 msw
Length:	1000 mm
Height:	500 mm
Width:	600 mm
Launch Weight:	50kg or 62kg with additional buoyancy module fitted
Thrust Forward:	50 kgf
Thrust Lateral:	28 kgf
Thrust Vertical:	13 kgf
Payload:	4kg or 16kg with additional buoyancy module fitted

Propulsion

All Seaeye ROVs feature brushless DC thrusters, which, apart from having the greatest power density, have drive electronics with velocity feedback for precise and rapid thrust control. These thrusters are interfaced to a fast PID control system along with a solid-state rate gyro for enhanced azimuth stability, a feature that automatically prevents overshoot

Appendix A: ROV models – Technical Details

on a change of heading. These essential building blocks enable Seaeye Marine to provide superior control and response from their powerful ROVs, setting them apart from the competition. The innovation for Seaeye FALCON was the development of a Magnetically Coupled brushless DC Thruster unit (MCT1) capable of resisting higher torque loads than competing units. Seaeye FALCON is powered by 5 Seaeye Magnetically Coupled Thruster units (MCT1) each capable of achieving 13 kgf thrust at 320W or a combined forward thrust (bollard pull) of 50 kgf. For an ROV weighing only 50 kilos this represents an impressive 1:1 power to weight ratio. These thrusters run cool without being oil filled and having no moving shaft seals, are low maintenance, extremely reliable and ideal for use in sensitive areas such as fisheries and on reefs. Thruster vector angles can be changed by the operator to reconfigure the vehicle to best suit the particular mission requirements.



MCT1 Thruster Performance

Thruster Configuration
4 Vectored Horizontal Thrusters
1 Vertical Thruster

Appendix A: ROV models – Technical Details

Control System

The FALCON is the first Seaeye ROV to enter production with a distributed intelligence control system. This is a multi-drop network which allows up to 128 devices to be connected together on a single RS485 serial network. Each device connected to the network, be it a thruster, light, compass or a future option, contains a microprocessor and interface electronics and is called a 'node'. These 'nodes' are controlled by a master processor in the Surface Unit and are fully isolated to maximise system reliability. Each node is connected to the Network's Star Point at the ROV junction box. The Star Point is a printed circuit board that provides each node with its own fused power supply and telemetry. This modular approach eliminates the need for a complex central electronics pod and significantly reduces the number of subsea connectors used. The result is a very flexible modern and simplified system architecture designed to improve reliability and ease of maintenance.

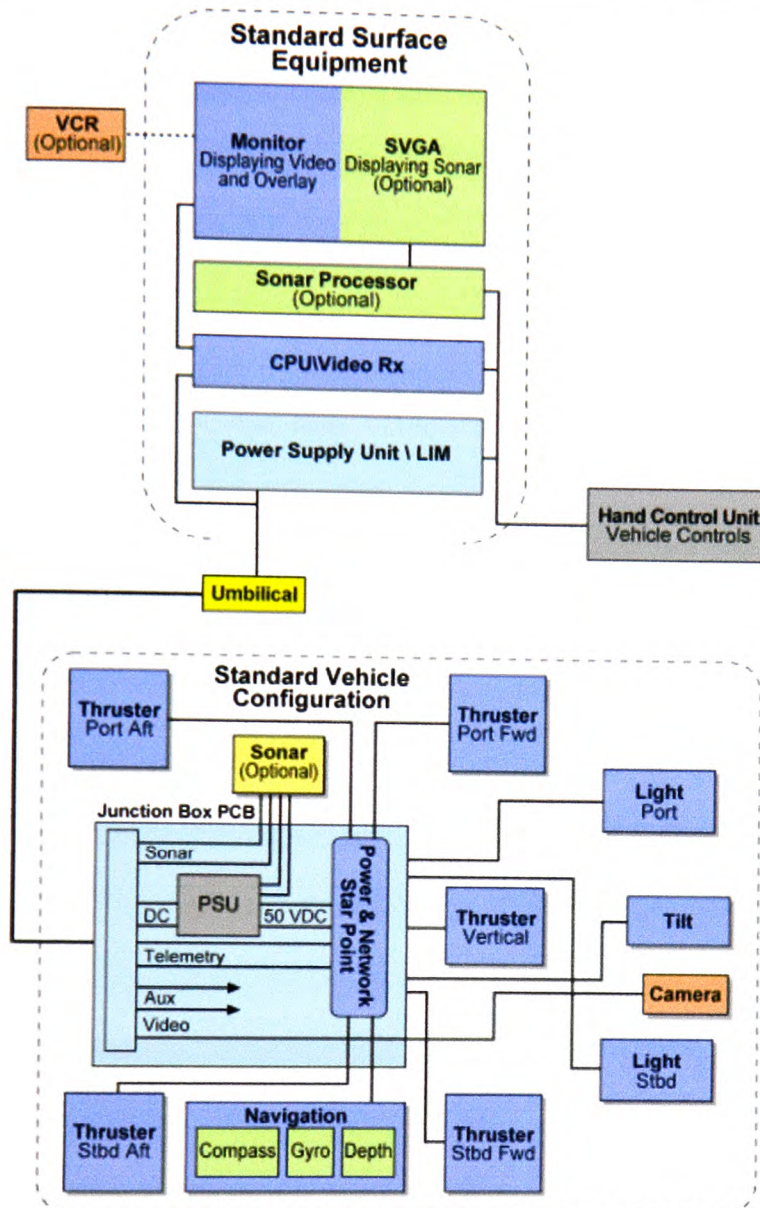
Control System Diagnostics

A software routine automatically checks each node when the system is powered up with alarms provided on the video overlay to alert the operator. Local diagnostics are also provided for each node in the Junction Box. A test button is provided with colour coded LEDs that confirm green if the fuse is intact or red if it is blown on a particular node. A further LED indicates the presence of telemetry to the node.

The ROV Junction Box

The one atmosphere junction box contains the Network Star Point and the video line-driver. The line driver allows quick and easy 'gain' adjustments to be made if different umbilical lengths are to be used. A standard bulkhead connector is used to connect each node to the Network Star Point at the junction box.

Appendix A: ROV models – Technical Details



FALCON control architecture

Navigation System & Auto Functions

Seaeye FALCON's navigation sensors and aids are housed in a single hardened aluminium pod. A compass is provided for heading information and a solid state rate gyro for auto heading control. A depth sensor provides depth information in feet or metres on the video overlay as well as control of auto depth.

Appendix A: ROV models – Technical Details

Specification	
Compass Accuracy:	$\pm 2^\circ$
Depth Sensor Accuracy:	$\pm 0.5\%$ of FSD
Gyro:	0.1 %/s
Surface Update Rate:	<40mS

Surface equipment

Power Requirements

Single phase "Universal Input" of between 100VAC - 270VAC at 2.5kw (compatible with generators fitted with auto voltage regulation).

Standard Surface Unit

The FALCON's power supply unit, processors and video systems are rack mountable and supplied in portable enclosures. The power supply unit provides a galvanically isolated 500VDC output protected by a L.I.M. All equipment is fitted into a transit case complete with handles and heavy splash-proof covers.



Appendix A: ROV models – Technical Details

Standard Hand Control Unit

FALCON's Standard Hand Control Unit (HCU) is formed within a rugged case and contains all the ergonomically designed vehicle controls.



Seaeye FALCON hand controller & 5 metre lead

Hand Control Unit specifications	
Height:	180mm
Width:	280mm
Depth:	120mm
Weight:	0.5kg

Vehicle Controls

The following Seaeye FALCON controls are provided:

- Single 3 axis joystick for horizontal vehicle control,
- Rotary control for vertical thruster power UP or DOWN,
- Rotary control for lights intensity,
- Auto pilot functions for both heading & depth,
- Vehicle power switches,
- Auxiliary vehicle controls (including manipulator).

Appendix B: Some Results from Optimal Control Theory

Some results from optimal control theory are presented in this appendix. The material presented herein is mostly used in Chapter 4 and Chapter 5.

Definition B.1 (l_p norm)

The l_p norm of a vector $\mathbf{u} \in \mathfrak{R}^m$ is defined as

$$\|\mathbf{u}\|_p = \left(\sum_{i=1}^m |u_i|^p \right)^{\frac{1}{p}}, \quad 1 \leq p \leq \infty \quad (\text{B.1})$$

For $p = 1, 2, \infty$ the following norms are obtained:

$$l_1 : \quad \|\mathbf{u}\|_1 = \sum_{i=1}^m |u_i| \quad (\text{B.2})$$

$$l_2 : \quad \|\mathbf{u}\|_2 = \sqrt{\sum_{i=1}^m u_i^2} \quad (\text{B.3})$$

$$l_\infty : \quad \|\mathbf{u}\|_\infty = \max_{1 \leq i \leq m} |u_i| \quad (\text{B.4})$$

Definition B.2 (Weighted sphere)

The weighted sphere $S_{\mathbf{W}}(\mathbf{u}_0, r)_p$ with centre \mathbf{u}_0 and radius r is set of vectors $\mathbf{u} \in \mathfrak{R}^m$ for which $\|\mathbf{W}(\mathbf{u} - \mathbf{u}_0)\|_p \leq r$, where \mathbf{W} is a positive definite weighting matrix. If $\mathbf{W} = \mathbf{I}_m$, the sphere $S_1(\mathbf{u}_0, r)_p = S(\mathbf{u}_0, r)_p$ is called a standard sphere.

Definition B.3 (Range and Nullspace)

Let $L: X \rightarrow Y$ be a linear operator. The range space $\mathcal{R}(L)$ is defined as the set of values in Y that are reached from X by application of L . That is, $\mathcal{R}(L) = \{\mathbf{y} = L\mathbf{x} : \mathbf{x} \in X\}$.

The nullspace $\mathcal{N}(L)$ is defined as the set of values in X that are transformed to $\mathbf{0}$ in Y by L . That is, $\mathcal{N}(L) = \{\mathbf{x} \in X : L\mathbf{x} = \mathbf{0}\}$.

Appendix B: Some Results from Optimal Control Theory

Lema B.1 (Pseudoinverse)

The problem

$$\min_{\mathbf{u}} \|\mathbf{u}\|_2$$

subject to $\mathbf{B}\mathbf{u} = \mathbf{v}$

where $\mathbf{v}(t) \in \mathcal{R}^k$, $\mathbf{u}(t) \in \mathcal{R}^m$, $m > k$ and $\text{rank}(\mathbf{B}) = k$, has the unique solution

$$\mathbf{u} = \mathbf{B}^+ \mathbf{v} \quad (\text{B.5})$$

where

$$\mathbf{B}^+ = \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} \quad (\text{B.6})$$

is the pseudoinverse of \mathbf{B} . The operator $^+$ is called the pseudoinverse operator.

Proof: The Lagrangian of this optimisation problem is defined as

$$L(\mathbf{u}, \boldsymbol{\lambda}) = \frac{1}{2} \mathbf{u}^T \mathbf{u} + \boldsymbol{\lambda}^T (\mathbf{v} - \mathbf{B}\mathbf{u}) \quad (\text{B.7})$$

where $\boldsymbol{\lambda}$ denotes the Lagrange multipliers. Differentiating the Lagrangian L with respect to \mathbf{u} yields

$$\frac{\partial L}{\partial \mathbf{u}} = \mathbf{u} - \mathbf{B}^T \boldsymbol{\lambda} \quad (\text{B.8})$$

$$\frac{\partial L}{\partial \mathbf{u}} = \mathbf{0} \Rightarrow \mathbf{u} = \mathbf{B}^T \boldsymbol{\lambda} \quad (\text{B.9})$$

$$\mathbf{v} = \mathbf{B}\mathbf{u} = \mathbf{B}\mathbf{B}^T \boldsymbol{\lambda} \quad (\text{B.10})$$

Since $\text{rank}(\mathbf{B}) = k$, matrix $\mathbf{B}\mathbf{B}^T$ is non-singular and the optimal solution for the Lagrange multipliers is

$$\boldsymbol{\lambda} = (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{v} \quad (\text{B.11})$$

Finally, the optimal solution of the given optimisation problem is

$$\mathbf{u} = \mathbf{B}^T \boldsymbol{\lambda} = \underbrace{\mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1}}_{\mathbf{B}^+} \mathbf{v} = \mathbf{B}^+ \mathbf{v} \quad (\text{B.12})$$

Appendix B: Some Results from Optimal Control Theory

Lema B.2 (Weighted pseudoinverse)

The problem

$$\min_{\mathbf{u}} \|\mathbf{W}\mathbf{u}\|_2$$

subject to $\mathbf{B}\mathbf{u} = \mathbf{v}$

where \mathbf{W} is a positive definite matrix, has the unique solution

$$\mathbf{u} = \mathbf{B}_w^+ \mathbf{v} \quad (\text{B.13})$$

where the weighted pseudoinverse \mathbf{B}_w^+ is defined by

$$\mathbf{B}_w^+ = \mathbf{W}^{-1}(\mathbf{B}\mathbf{W}^{-1})^+ = \mathbf{W}^{-1}\mathbf{B}^T(\mathbf{B}\mathbf{W}^{-1}\mathbf{B}^T)^{-1} \quad (\text{B.14})$$

Proof: Substitution $\mathbf{e} = \mathbf{W}\mathbf{u}$ leads to $\mathbf{u} = \mathbf{W}^{-1}\mathbf{e}$ and the equivalent problem can be defined as

$$\min_{\mathbf{u}} \|\mathbf{e}\|_2$$

subject to $\mathbf{B}(\mathbf{W}^{-1}\mathbf{e}) = \mathbf{v} \Leftrightarrow (\mathbf{B}\mathbf{W}^{-1})\mathbf{e} = \mathbf{v}$

Now, the solution can be found using Lema B.1:

$$\mathbf{e} = (\mathbf{B}\mathbf{W}^{-1})^+ \mathbf{v} \Rightarrow \mathbf{u} = \mathbf{W}^{-1}(\mathbf{B}\mathbf{W}^{-1})^+ \mathbf{v}$$

Lema B.3 (Pseudoinverse – General case)

The problem

$$\min_{\mathbf{u}} \|\mathbf{W}(\mathbf{u} - \mathbf{u}_p)\|_2$$

subject to $\mathbf{B}\mathbf{u} = \mathbf{v}$

where \mathbf{W} is a positive definite matrix, has a solution (Härkegård, 2003)

$$\mathbf{u} = \mathbf{F}\mathbf{u}_p + \mathbf{G}\mathbf{v} \quad (\text{B.15})$$

where

$$\mathbf{F} = \mathbf{I} - \mathbf{G}\mathbf{B} \quad (\text{B.16})$$

$$\mathbf{G} = \mathbf{W}^{-1}(\mathbf{B}\mathbf{W}^{-1})^+ \quad (\text{B.17})$$

Proof: Substituting $\mathbf{e} = \mathbf{W}(\mathbf{u} - \mathbf{u}_p)$ yields $\mathbf{u} = \mathbf{u}_p + \mathbf{W}^{-1}\mathbf{e}$ and the equivalent problem can be defined as

$$\begin{aligned} & \min_{\mathbf{u}} \|\mathbf{e}\|_2 \\ & \text{subject to } \mathbf{B}(\mathbf{u}_p + \mathbf{W}^{-1}\mathbf{e}) = \mathbf{v} \Leftrightarrow (\mathbf{B}\mathbf{W}^{-1})\mathbf{e} = \mathbf{v} - \mathbf{B}\mathbf{u}_p \end{aligned}$$

Now, the solution can be found using Lema B.1:

$$\begin{aligned} \mathbf{e} &= (\mathbf{B}\mathbf{W}^{-1})^+(\mathbf{v} - \mathbf{B}\mathbf{u}_p) \Rightarrow \\ \mathbf{u} &= \mathbf{u}_p + \mathbf{W}^{-1} \underbrace{((\mathbf{B}\mathbf{W}^{-1})^+(\mathbf{v} - \mathbf{B}\mathbf{u}_p))}_{\mathbf{e}} = \\ &= \underbrace{(\mathbf{I} - \mathbf{W}^{-1}(\mathbf{B}\mathbf{W}^{-1})^+\mathbf{B})}_{\mathbf{F}} \mathbf{u}_p + \underbrace{\mathbf{W}^{-1}(\mathbf{B}\mathbf{W}^{-1})^+}_{\mathbf{G}} \mathbf{v} = \mathbf{F}\mathbf{u}_p + \mathbf{G}\mathbf{v} \end{aligned}$$

References

HÄRKEGÅRD, O. (2003). *Backstepping and Control Allocation with Application to Flight Control*. PhD thesis, Department of Electrical Engineering, Linköping University, Sweden.

Appendix C: Some Results from 3D Geometry

Introduction

In this appendix, some results from the 3D geometry are presented. The material, taken from the Internet web site (<http://astronomy.swin.edu.au/~pbourke/geometry>), is used in Chapter 4 (for geometric interpretation of the control allocation problem) and Chapter 5 (for visualisation of the feasible region).

The intersection of two planes

Given the two planes π_1 and π_2

$$\pi_1 : \quad \mathbf{N}_1^T \cdot \mathbf{p} = d_1 \quad (\text{C.1})$$

$$\pi_2 : \quad \mathbf{N}_2^T \cdot \mathbf{p} = d_2 \quad (\text{C.2})$$

where \mathbf{N}_1 and \mathbf{N}_2 are normal vectors on the planes and $\mathbf{p} = [x \ y \ z]^T$ is a point that belongs to both planes. To find the intersection of the planes π_1 and π_2 , three cases are possible:

1. The intersection is a line, if the planes are not parallel or coincident,
2. The intersection does not exist, if the planes are parallel and not coincident,
3. If the planes are coincident, the intersection is the same plane.

An easy way to verify that two planes are not parallel is to check that the cross product of their normals is not a zero vector, i.e.

$$\mathbf{N}_1 \times \mathbf{N}_2 \neq \mathbf{0} \quad (\text{C.3})$$

In the following, it is assumed that planes are not parallel, i.e. that condition (C.3) is satisfied and that the intersection of the planes π_1 and π_2 is a line l (Figure C.1). The equation of the line l can be written as

Appendix C: Some Results from 3D Geometry

$$\mathbf{p} = c_1 \mathbf{N}_1 + c_2 \mathbf{N}_2 + t \mathbf{N}_1 \times \mathbf{N}_2 \quad (\text{C.4})$$

where t is the parameter of the line. Taking the dot product of both side of (C.4) with \mathbf{N}_1^T yields

$$\mathbf{N}_1^T \cdot \mathbf{p} = d_1 = \mathbf{N}_1^T \cdot (c_1 \mathbf{N}_1 + c_2 \mathbf{N}_2 + u \mathbf{N}_1 \times \mathbf{N}_2) = c_1 \mathbf{N}_1^T \cdot \mathbf{N}_1 + c_2 \mathbf{N}_1^T \cdot \mathbf{N}_2 \quad (\text{C.5})$$

since $\mathbf{N}_1^T \cdot (\mathbf{N}_1 \times \mathbf{N}_2) = 0$. In the same way it is possible to obtain

$$\mathbf{N}_2^T \cdot \mathbf{p} = d_2 = \mathbf{N}_2^T \cdot (c_1 \mathbf{N}_1 + c_2 \mathbf{N}_2 + u \mathbf{N}_1 \times \mathbf{N}_2) = c_1 \mathbf{N}_1^T \cdot \mathbf{N}_2 + c_2 \mathbf{N}_2^T \cdot \mathbf{N}_2 \quad (\text{C.6})$$

Solving (C.5) and (C.6) for c_1 and c_2 yields

$$c_1 = \frac{d_1 \mathbf{N}_2^T \cdot \mathbf{N}_2 - d_2 \mathbf{N}_1^T \cdot \mathbf{N}_2}{\Delta} \quad (\text{C.7})$$

$$c_2 = \frac{d_2 \mathbf{N}_1^T \cdot \mathbf{N}_1 - d_1 \mathbf{N}_1^T \cdot \mathbf{N}_2}{\Delta} \quad (\text{C.8})$$

where

$$\Delta = (\mathbf{N}_1^T \cdot \mathbf{N}_1)(\mathbf{N}_2^T \cdot \mathbf{N}_2) - (\mathbf{N}_1^T \cdot \mathbf{N}_2)^2 \quad (\text{C.9})$$

Note that $\Delta \neq 0$, since it is assumed that condition (C.3) holds.

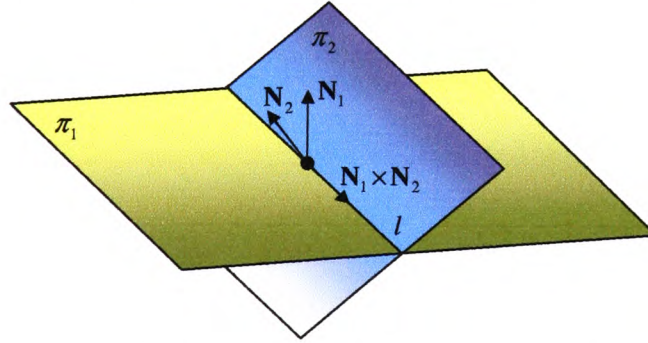


Figure C.1 The intersection of two non-parallel planes.

The intersection of three planes

The intersection of three planes, if no two of them are parallel, is a point. Given the three planes π_1 , π_2 and π_3

Appendix C: Some Results from 3D Geometry

$$\pi_1: \quad \mathbf{N}_1^T \cdot \mathbf{p} = d_1 \quad (\text{C.10})$$

$$\pi_2: \quad \mathbf{N}_2^T \cdot \mathbf{p} = d_2 \quad (\text{C.11})$$

$$\pi_3: \quad \mathbf{N}_3^T \cdot \mathbf{p} = d_3 \quad (\text{C.12})$$

the intersection point \mathbf{P} (see Figure C.2) is given by

$$\mathbf{P} = \frac{d_1(\mathbf{N}_2 \times \mathbf{N}_3) + d_2(\mathbf{N}_3 \times \mathbf{N}_1) + d_3(\mathbf{N}_1 \times \mathbf{N}_2)}{\mathbf{N}_1^T \cdot (\mathbf{N}_2 \times \mathbf{N}_3)} \quad (\text{E.13})$$

Note that the denominator in (C.13) is equal to zero if any two of the planes are parallel.

The assumption that no two of planes are parallel ensures that denominator is different than zero.

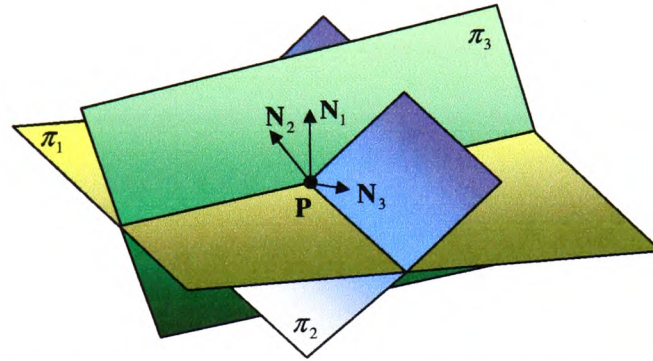


Figure C.2 The intersection of three planes, where no two of them are parallel.

Appendix D: ROV simulator

Introduction

A general ROV simulator, developed as Simulink model in MATLAB environment, is described in this appendix. This simulator was used to derive simulation results, presented in Chapter 6. The model includes the non-linear model of an ROV with 6 DOF, propulsion system, hand control unit (HCU) and thruster fault detection and accommodation system (FDAS), described in Chapter 5. In order to enhance user interface and to improve understanding of the underlying hybrid approach for control allocation, a virtual underwater world has been developed with two ROV models (FALCON and URIS) in a realistic underwater environment. A joystick is used as the input device to generate command input signals. Different fault conditions can be injected into the simulation using joystick buttons.

Requirements

Software requirements

- **MATLAB** (version 6.5 or higher),
- **Simulink** (version 5.0 or higher),
- **Dials & Gauges Blockset** (version 1.1.2 or higher),
- **Virtual Reality Toolbox** (version 3.0 or higher)
- **Windows 98, 2000 or XP**

Hardware requirements

Virtual Reality Display active	Virtual Reality Display not active
<ul style="list-style-type: none">• Pentium 4 or Athlon based PC	<ul style="list-style-type: none">• Pentium 4 or Athlon based PC
<ul style="list-style-type: none">• 256 Mb RAM	<ul style="list-style-type: none">• 128 Mb RAM
<ul style="list-style-type: none">• Powerful graphic card (min 64 Mb memory & advanced Open GL)	<ul style="list-style-type: none">• Standard graphic card

Appendix D: ROV simulator

Quick start

1. Set current directory: C:\WINDOWS\Desktop\PhD\Matlab\ROV.
2. Open model rov.mdl.
3. Double click on the block "Underwater World" to open the block parameters dialog box.
4. Click on the View button to open Virtual Reality Display.
5. Click on the Ok button to close the block parameters dialog box.
6. Arrange windows and displays as you like (see Figure D.1 for one possibility).
7. Start simulation by choosing Simulation\Start from menu. After simulation is started, new displays (*Feasible Region* and *Horizontal Thrusters*) will appear on the screen.

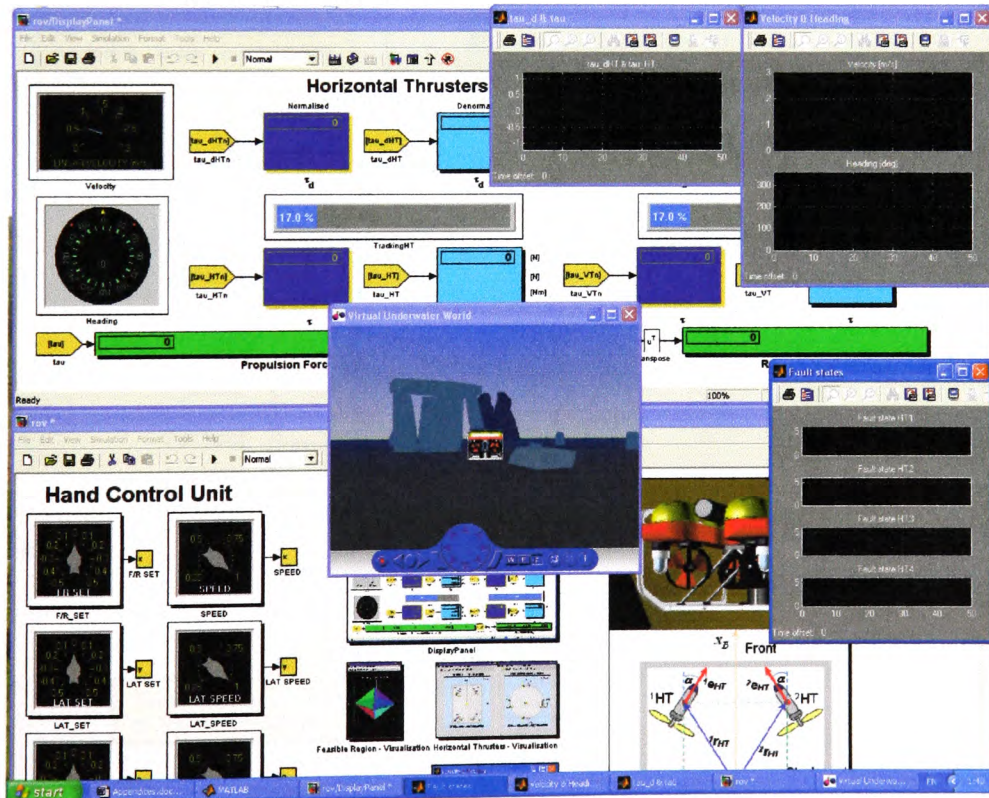


Figure D.1 Screenshot showing a possible arrangement of windows and displays.

ROV simulator history

June – September 2001

Lauvdal and Fossen (1994) developed a MATLAB simulation program for marine and flight vehicles. The complete MATLAB-simulator included the main simulation program (sim.m) and 16 different vehicle models, originally written as MATLAB script files. One of these vehicles was a ROV, weighted $m = 185\text{ kg}$ and described as a non-linear model with 6 DOF. In order to use the powerful features of Simulink, the original script file was transformed into a S-function and associated with a Simulink custom block. This block and other auxiliary blocks were wrapped in the Subsystem block "ROV model". All model parameters were adjustable through the associated block parameters dialog box. The first version of the ROV simulator is shown in Figure D.2.

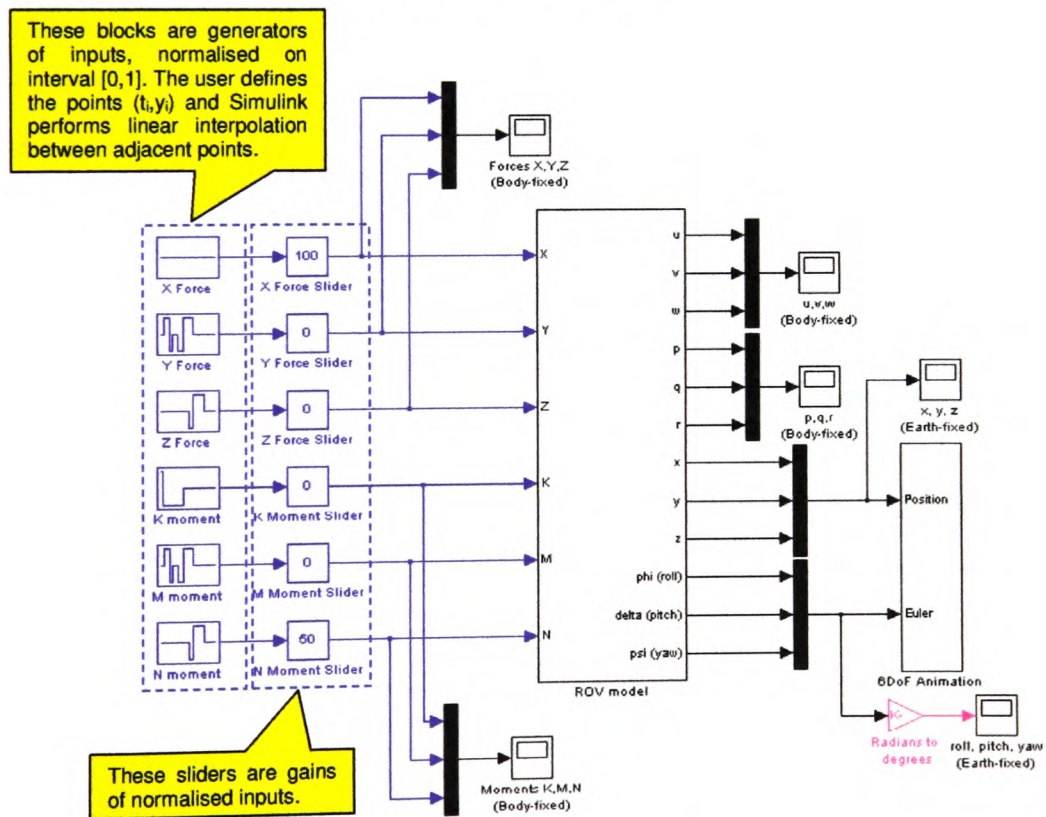


Figure D.2 The first version of the ROV simulator.

October 2001 – April 2002

The next stage was the design of the propulsion system and the HCU. The block diagram showing the second version of the ROV simulator is displayed in Figure D.3. Since the only available actuators for FALCON and URIS are thrusters, the propulsion system was built using the full thruster model (shown in Figure 3.16 & Figure 3.17), assuming the vector form of the affine thruster model (3.99). Both thruster configurations were implemented. The HCU was developed combining a joystick (as input device) and joystick & knob controls from the Dials & Gauges Blockset. The signal conditioning block performed shaping and denormalisation of the HCU output. The first version of the control allocator, obtained by logic reasoning, performed simple, sub-optimal transformation from desired control vector to actuator settings. Heading and velocity responses were visualised using gauge controls from the Dials & Gauges Blockset. The position of the ROV in space was visualised by a single point (representing the position of CG in $\{E\}$). The orientation of the vehicle was visualised using the linear velocity vector (originated in CG) and its projections to co-ordinate planes of $\{E\}$.

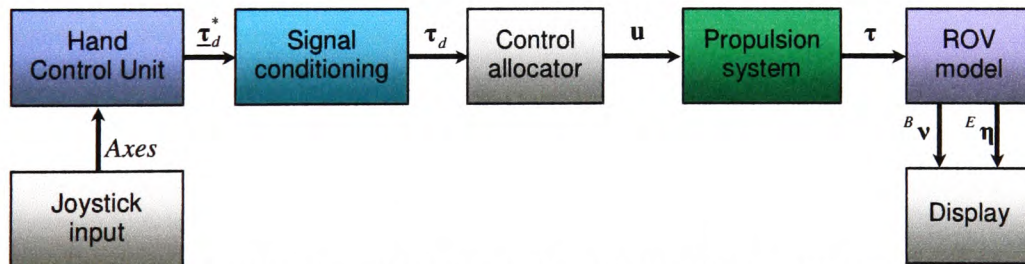


Figure D.3 Block diagram showing the second version of the ROV simulator.

May - October 2002

The block diagram of the third version is displayed in Figure D.4. This time the first version of the FDAS was included into the ROV simulator. The FDS was implemented as the state machine, realised as MATLAB function, with joystick buttons signal as input. The aim was to use the joystick to generate the fault indicator vector. The FAS,

implemented as a series of MATLAB functions, used pseudoinverse to find the solution of the control allocation problem. Unfeasible solutions were approximated by T - or S - approximation. The parameters of the fault code table were adjustable through the block parameters dialog box. The user interface was enhanced with two new displays. The first display performed dynamic visualisation of the feasible region inside the virtual control space. The second display visualised the distribution of propulsion forces and moments among thrusters.

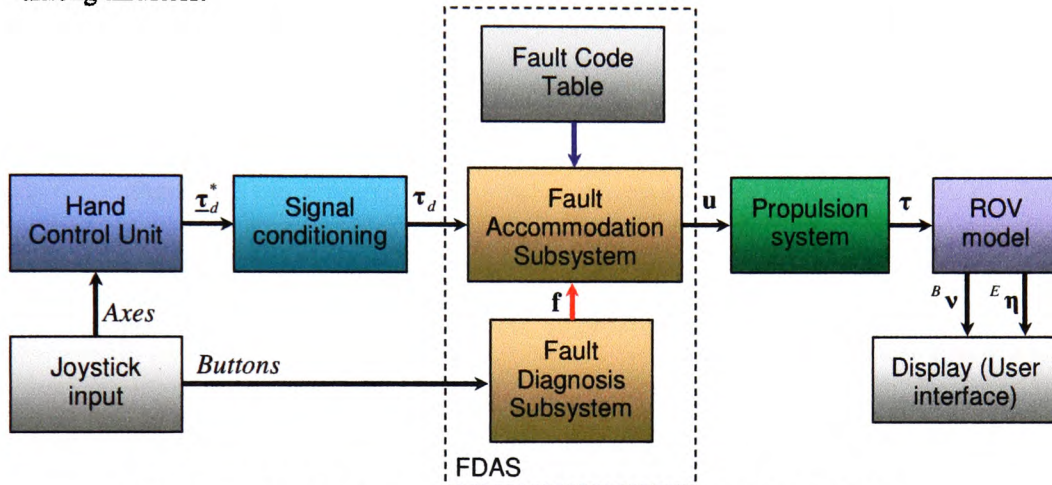


Figure D.4 Block diagram showing the third version of the ROV simulator.

November 2002 – September 2003

The graphic presentation of simulation results was significantly improved in the fourth version of the ROV simulator by introducing the Virtual Reality Display, as indicated in Figure D.5. A virtual underwater world has been developed with models of FALCON and URIS in a realistic underwater environment. Different testing objects were created, in order to evaluate manoeuvring capabilities and performance of the vehicle. The vehicle can be viewed from different angles, using the joystick hat switch (point of view) control. Since the simulation with active Virtual Reality Display and thruster control loop dynamics was slow and uncomfortable for work, the later was removed from the model (the effects of neglected thruster control loop dynamics are discussed in section 6.4.4).

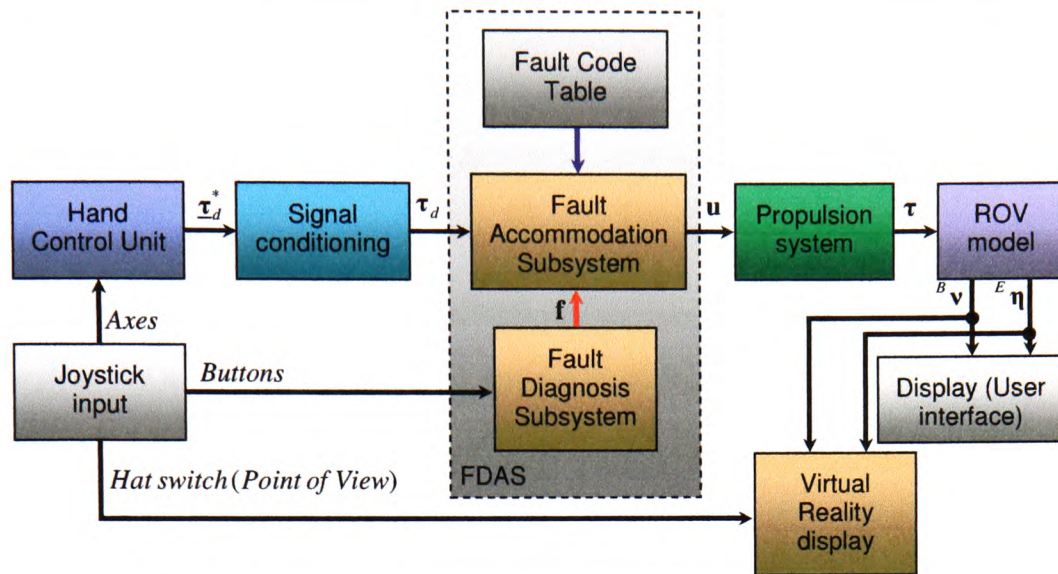


Figure D.5 Block diagram showing the fourth version of the ROV simulator.

October – December 2003

In this period overall performance of the ROV simulator was improved. Internal reorganisation of the code was performed, in order to accelerate execution, improve efficiency and compatibility with the nomenclature used in Chapter 5. The new version of the FDAS, using the hybrid approach for control allocation², has been developed. The display showing the feasible region was enhanced adding the visualisation of the attainable command set. A set of new displays was developed, including time responses of desired and actual propulsion forces, velocity and heading. In addition, a set of "To Workspace" sink blocks was added, in order to save representative variables for post-simulation analysis.

² During the conference MCMC 2003 (Girona, Spain), the author discussed the control allocation problem with Professor Tor A. Johansen, Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim. Weaknesses of the pseudoinverse solution were identified and the initial idea about improvements of the solution was developed, which resulted in a novel, hybrid approach for control allocation, described in section 5.5.2.

Description

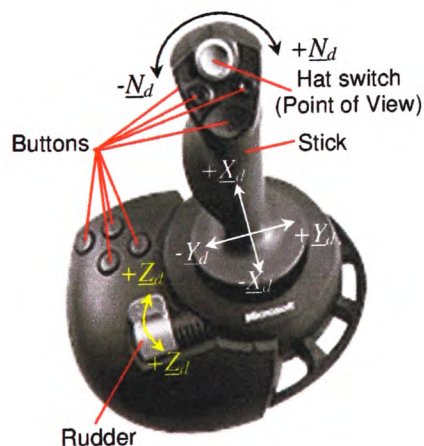
Block diagram of the ROV simulator (last version) is identical to those shown in Figure D.5. A description of each component is given in the following.

Hand Control Unit & Joystick

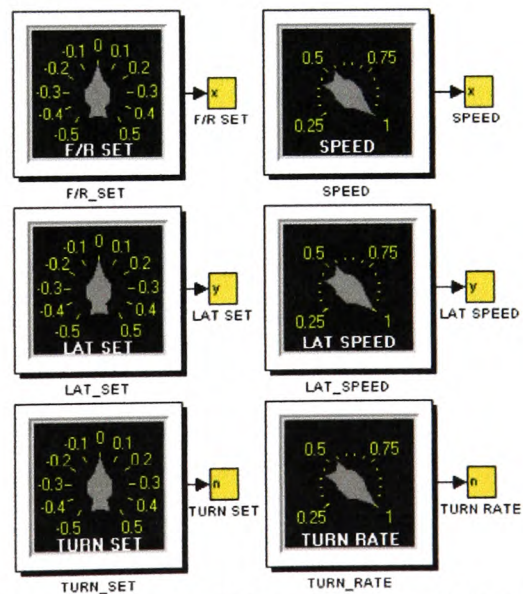
The HCU (Figure D.6 (a)), used by the ROV pilot to control motion of the vehicle, is implemented with functionally equivalent combination of Joystick (Figure D.6 (b)) and a group of knob controls from the Dials & Gauges Blockset (Figure D.6 (c)). HCU controls and their functions are described in Table D.1.



(a) Real HCU.



(b) Joystick.



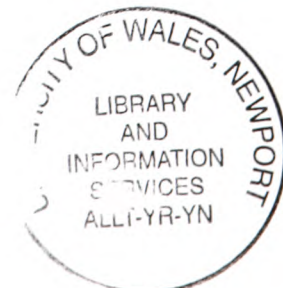
(c) Knob controls ("trimmers") from Dials & Gauges Blockset.

Figure D.6 Implementation of the HCU.

Appendix D: ROV simulator

Control	Function	
Joystick	Controls movement of the vehicle as follows:	
	Stick forward: $\underline{X}_d^* \in [0,1]$	ROV motion: <i>forward</i>
	Stick reverse: $\underline{X}_d^* \in [-1,0]$	ROV motion: <i>reverse</i>
	Stick right: $\underline{Y}_d^* \in [0,1]$	ROV motion: <i>starboard</i>
	Stick left: $\underline{Y}_d^* \in [-1,0]$	ROV motion: <i>port</i>
	Stick rotation right: $\underline{N}_d^* \in [0,1]$	ROV motion: <i>turn to starboard</i>
	Stick rotation left: $\underline{N}_d^* \in [-1,0]$	ROV motion: <i>turn to port</i>
	Rudder reverse: $\underline{Z}_d^* \in [0,1]$	ROV motion: <i>descend</i>
	Rudder forward: $\underline{Z}_d^* \in [-1,0]$	ROV motion: <i>ascend</i>
	Hat switch: used by the Virtual Reality Display to change Camera Side View	
	Buttons: used by the FDS to inject different thruster faults	
F/R SET	Rotating the control clockwise or anticlockwise introduce the proportional amount of the force in forward or reverse direction, respectively.	Total desired surge force: $\underline{\tau}_{Xd}^* = sat_{[-1,1]}[(\underline{X}_d^* + \text{F/R SET}) \cdot \text{SPEED}]$
SPEED	Scaling factor of the force in forward/reverse direction.	
LAT SET	Rotating the control clockwise or anticlockwise introduce the proportional amount of the force in right or left direction, respectively.	Total desired sway force: $\underline{\tau}_{Yd}^* = sat_{[-1,1]}[(\underline{Y}_d^* + \text{LATSET}) \cdot \text{LAT SPEED}]$
LAT SPEED	Scaling factor of the force in right/left direction.	
TURN SET	Rotating the control clockwise or anticlockwise introduce the proportional amount of the moment for rotation in right or left direction, respectively.	Total desired yaw moment: $\underline{\tau}_{Nd}^* = sat_{[-1,1]}[(\underline{N}_d^* + \text{TURNSET}) \cdot \text{TURN RATE}]$
TURN RATE	Scaling factor of the moment for rotation in right/left direction.	

Table D.1 HCU controls.



Appendix D: ROV simulator

Signal conditioning

This block performs two functions. The first function is conditioning of raw signals (\underline{X}_d^* , \underline{Y}_d^* , \underline{N}_d^* and \underline{Z}_d^*) obtained from joystick and their integration with knob controls ("trimmers") in accordance to Table D.1. The parameters are adjustable through the block parameters dialog box "Signal Conditioning" (Figure D.7), which include *Dead Zone* **1** (region of zero output around stick centre position), *Gain* **2** (scaling factor $g > 1$ to ensure that the entire range $[-1,1]$ is covered) and *Number of discretisation levels (HT)* **3** and (VT) **4**. Discretisation is performed to compensate imprecise joystick output and to emulate the signals from real HCU, which are discretised. Parameters **3** and **4** denote the number of discretisation levels on interval $[0,1]$. After conditioning and integration, signals are saturated to interval $[-1,1]$ and two vectors are obtained:

$$\underline{\tau}_d^{HT} = [\underline{X}_d \quad \underline{Y}_d \quad \underline{N}_d]^T \text{ and } \underline{\tau}_d^{VT} = [\underline{Z}_d].$$

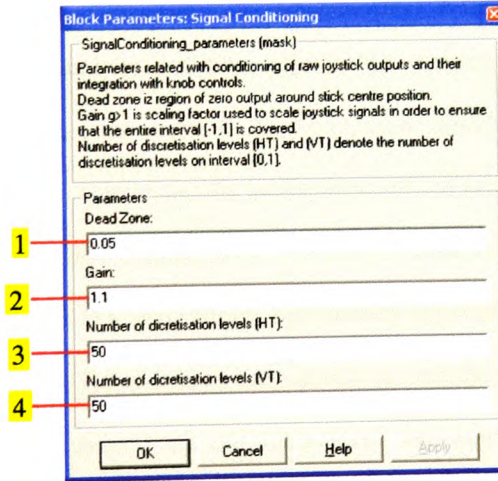


Figure D.7 Block parameters dialog box "Signal Conditioning".

The second function is selection of the command input source: *Joystick* or *Constant command input*. The block representation in Simulink model is shown in Figure D.8 (a), and associated block parameters dialog box is shown in Figure D.8 (b).

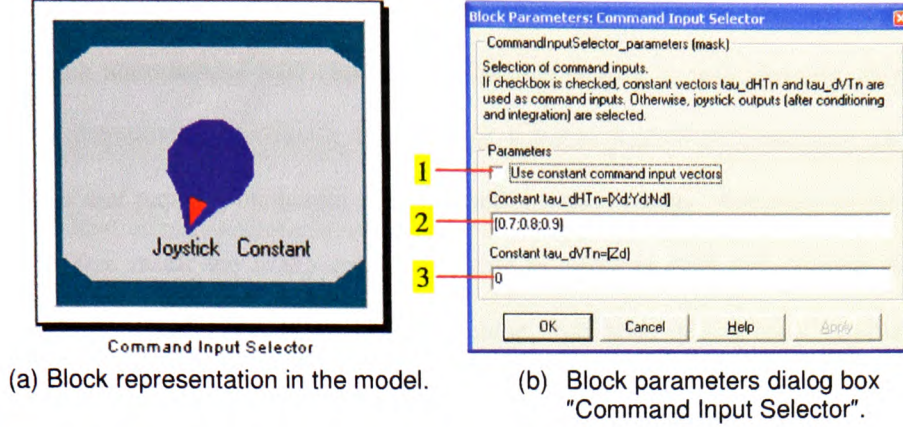


Figure D.8 Selection of the command input source.

If checkbox 1 is checked, constant command input vectors τ_d^{HT} (2) and τ_d^{VT} (3) are selected as source signals. Otherwise, joystick outputs (after conditioning and integration) are selected.

Fault diagnosis subsystem

The FDS, described in section 5.4, is able to detect faults in a single thruster in real-time conditions. The FDS monitors real-time signals for external and internal faults and generates a total fault indicator vector \mathbf{f} . However, these signals are not available during simulation. In order to be able to test the performance of the proposed FDAS, the FDS in the ROV simulator is implemented as a MATLAB function wrapped in a Simulink subsystem block (Figure D.9 (a)), whose functionally equivalent Stateflow machine is shown in Figure D.9 (c). The aim is to use joystick to generate the fault indicator vector \mathbf{f} . Faults in vertical thruster(s) are not considered. The machine has two superstates (*FDAS active* and *FDAS not active*) and two outputs (fault indicator vector \mathbf{f} and FDAS status signal s). The *FDAS active* superstate ($s=1$) is the parent of the five substates (*Fault-free*, *Fault in HT1*, *Fault in HT2*, *Fault in HT3* and *Fault in HT4*). The *FDAS not active* superstate ($s=0$) is the parent of two substates (*Fault-free* and *Selected thruster*

Appendix D: ROV simulator

uncontrollable). Transitions between states are triggered by pressing any of the active joystick buttons AB1-AB7. Inside each substate there is a mark showing which active joystick button will activate it. For example, substate *Fault in HT1* has mark AB1, which means that pressing the button AB1 will activate this substate. Activation of the substate *Fault-free* inside the *FDAS active* superstate by pressing AB5 will generate the vector $\mathbf{f} = [1 \ 1 \ 1 \ 1]$ as the output of the machine. Activation of any other substate, whose parent is the *FDAS active* superstate, will modify the output vector as explained in the following. Substates *Fault in HT1*, *Fault in HT2*, *Fault in HT3* and *Fault in HT4* have associated faulty codes (2 - "Jammed", 3 - "Heavy jammed", 4 - "Broken", 5 - "Unknown"). The block parameters dialog box, shown in Figure D.9 (b), establishes the link between substates and faulty codes using combo boxes 1, 2, 3 and 4. After activation any of these substates, associated faulty code replaces the corresponding element in vector \mathbf{f} . For example, pressing the AB2 causes activation of the *Fault in HT2* substate with associated faulty code 4 ("Broken") and new output vector of the machine is $\mathbf{f} = [1 \ 4 \ 1 \ 1]$. The *FDAS not active* superstate is included in the machine in order to demonstrate difficulties related with the control of the vehicle's motion in the presence of a failure in a single horizontal thruster for the case when the FDAS is not active (see examples (B5) & (B6) in Chapter 6 for more information). Selection of the faulty (uncontrollable) thruster is performed using combo box 5 in Figure D.9 (b). In particular, pressing AB6 will switch off the selected uncontrollable thruster HT2, but fault accommodation will not be performed. Control of the vehicle's motion is very hard in this case because disabling one of the thrusters without reconfiguration introduces unbalanced moment components, which cause undesired rotation of the vehicle. By pressing AB2 the same case (broken propeller in HT2) can be simulated, but this time with FDAS turned

Appendix D: ROV simulator

on. The FDAS provides optimal redistribution of propulsion forces among three remaining thrusters in order to compensate unbalanced components.

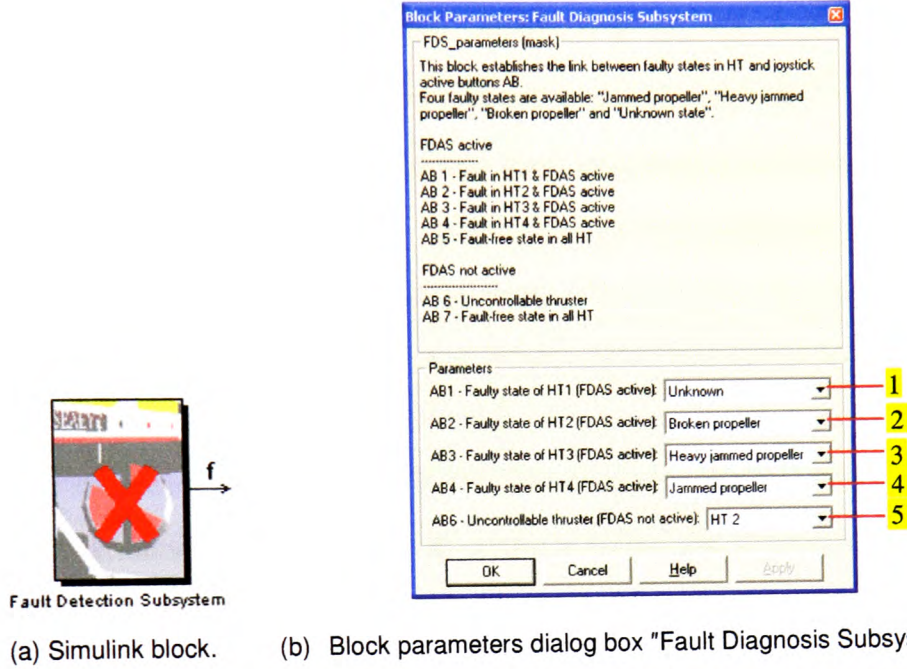


Figure D.9 Implementation of the FDS.

Fault accommodation subsystem

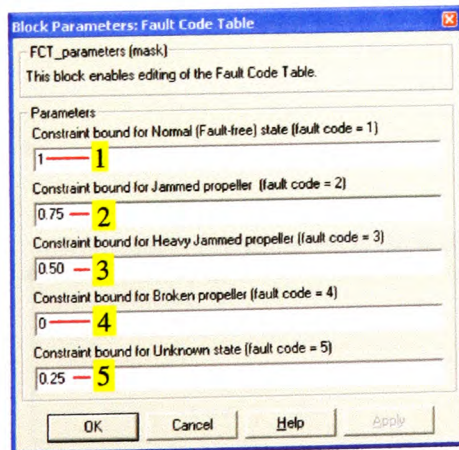
The inputs to the FAS block (Figure D.10 (a)) are the fault indicator vector \mathbf{f} (the output of the FDS) and vectors $\underline{\mathbf{t}}_d^{HT}$ and $\underline{\mathbf{t}}_d^{VT}$, in accordance to selected command input source. The FAS performs the hybrid approach for control allocation using the FAS algorithm (Algorithm 5.2, page 5-65). The fault code table can be edited using the block parameters

Appendix D: ROV simulator

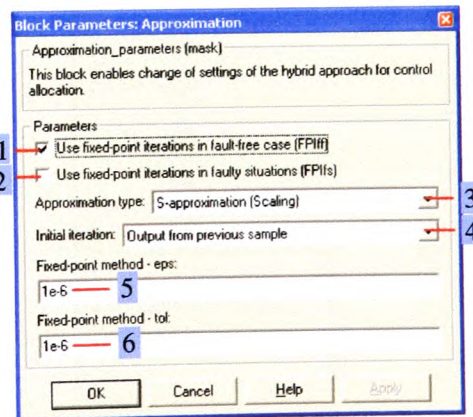
dialog box shown in Figure D.10 (b). In accordance to Table 5.8, the parameters 1, 2, 3, 4 and 5 denote constraint bounds s_i^{HT} for thruster states "Fault-free", "Jammed propeller", "Heavy jammed propeller", "Broken propeller" and "Unknown state", respectively. The output of the FAS is a composite control vector $\underline{u}^* = \begin{bmatrix} \underline{u}^{*HT} & \underline{u}^{*VT} \end{bmatrix}$ (see page 5-33). The inversion and approximation stages are realised as MATLAB functions. The hybrid approach settings can be changed using the block parameters dialog box, associated with the approximation stage and shown in Figure D.10 (c). In particular, if the check box 1 (2) is checked, the fixed-point iteration method is enabled in fault-free case (faulty situations). The type of approximation (T -approximation or S -approximation) can be selected using the combo box 3. The initial iteration for the fixed-point iterations can be chosen using the combo box 4. Available choices are: T -approximation, S -approximation or Output from previous sample. Design parameters eps and tol of the fixed-point iterations are adjustable using the fields 5 and 6.



(a) Simulink block.



(b) Block parameters dialog box "Fault Code Table".



(c) Block parameter dialog box "Approximation".

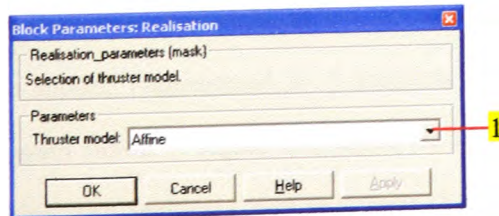
Figure D.10 Implementation of the FAS.

Appendix D: ROV simulator

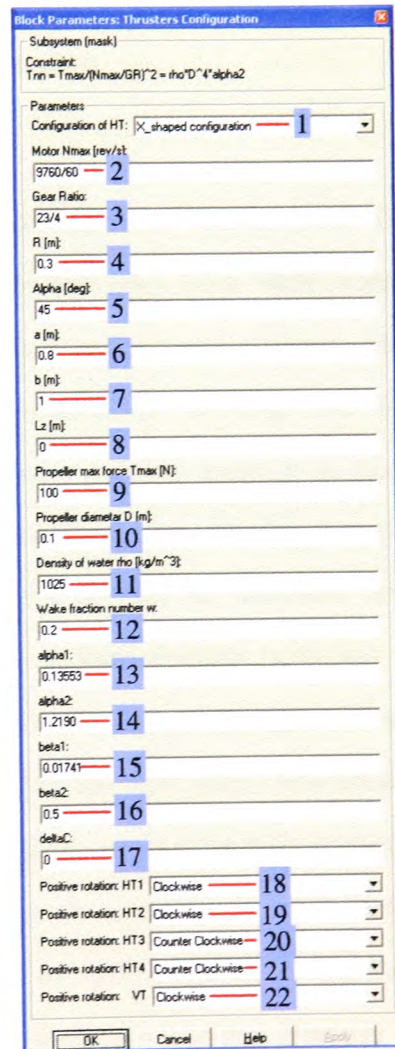
Propulsion system

Components "Correction", "Transformation", "Adaptation" and "Thrusters" of the overall FDAS architecture (Figure 5.9) are implemented as one, composite MATLAB function, wrapped in a Simulink subsystem block called Realisation. The input to the block is control vector \underline{u}^* , the output of the FAS block. The output is the vector of propulsion forces τ . The block parameters dialog boxes, shown in Figure D.11 (a) & (b), enables the selection of affine or bilinear thruster model (1) and customizing the thruster configuration parameters (1 - 22).

Label	Reference
1, 4, 5, 6, 7, 8	Figure 3.7 (pg. 3-26) Table 3.5, Table 3.6
2, 3, 9	Technical specifications
10, 11	Eq. (3.83) (pg. 3-33)
12	Eq. (3.72) (pg. 3-30)
13, 14, 15, 16	Eq. (3.79) (pg. 3-32) Eq. (3.80) (pg. 3-33)
16	Eq. (6.5) (pg. 6-21)
18, 19, 20, 21, 22	Example 3.1 (pg. 3-27)



(a) Block parameters dialog box "Realisation".



(b) Block parameters dialog box "Thruster configuration".

Figure D.11 Implementation of the propulsion system.

Parameters T_{\max} , N_{\max} , GR (Gear Ratio), ρ and α_2 are related by constraint

$$\frac{T_{\max}}{\left(\frac{N_{\max}}{GR}\right)^2} = \rho D^4 \alpha_2 = T_{\eta|\eta|} \quad (D.1)$$

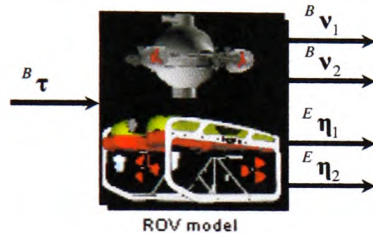
As stated in section 3.7.4, modelling of FALCON's propulsion system is an ongoing research project and, unfortunately, the thruster model is not yet available. In order to perform simulations and testing of the FDAS, some parameters of the model had to be assumed. However, obtained simulation results are expected and acceptable, which confirm that assumed parameters are close to real parameters.

ROV model

The ROV dynamics and kinematics model is implemented as a *S*-function block. This block and other auxiliary blocks are wrapped in the Subsystem block "ROV model", shown in Figure D.12 (a). In the original MATLAB script (Lauvdal and Fossen, 1994) model-dependent parameters were included (hard-coded) into the script file. This approach was not practical from the user point of view, since any change in the model required editing of the script. In order to make the model more user-friendly, all model-dependent parameters are transferred from the script file into the associated block parameters dialog box, which is easy accessible from the ROV simulator. In this way, all adjustable parameters of the ROV model can be easily changed through the block parameters dialog box, depicted in Figure D.12 (b). This flexibility enables easy addition and simulation of other ROVs, not included in the original version of the ROV simulator. The model, implemented in *S*-function, is functionally equivalent to the simulation diagram shown in Figure 3.4, where the attitude of the ROV is represented with Euler angles. The additional damping terms (lift & drag forces and rotational damping) are included in the *S*-function (see (Fossen, 2002) for more information).

Appendix D: ROV simulator

Label	Reference
1, 2, 3, 4	Section 3.3 (pg. 3-3)
5	Eq. (3.38) (pg. 3-13)
6	Eq. (3.57) (pg. 3-19)
7	Eq. (3.53) (pg. 3-18)
8, 9	(Fossen, 2002)



(a) Simulink block.

(b) Block parameters dialog box "ROV model".

Figure D.12 Implementation of the ROV model.

Development of dynamic models of FALCON and URIS are ongoing projects, expected to be completed in spring 2004. In order to test the performance of the FDAS, some ROV model had to be included in the ROV simulator. Since the original script (Lauvdal and Fossen, 1994) had already incorporated a ROV model in 6 DOF, it was decided to keep this model until the work on modelling and identification of FALCON and URIS is completed and corresponding dynamic models become available. Hence, parameters shown in Figure D.12 (b) describe dynamics of the ROV model from the original script file. The ROV, weighted 185 kg , is neutrally buoyant ($W = B$) and its dynamics are much slower than the real dynamics of FALCON and URIS. This should be taken into account during interpretation of simulation results and responses in the Virtual Reality

Display. When dynamic models of FALCON and URIS become available, it is expected to obtain faster response of the vehicle and better agreement between virtual reality behaviour and real-time behaviour. All simulations in Chapter 6 were performed assuming the ROV model from the original MATLAB script. Because of this reason, simulation results should be interpreted from the qualitative perspective.

Display (User interface)

The standard user interface includes the following displays:

Fault states. Time diagrams of fault states of each HT are shown in this display (Figure D.13 (a)).

τ_d & τ . This display shows time diagrams of components of vectors $\tau_d^{HT} = [\tau_{xd} \ \tau_{yd} \ \tau_{Nd}]^T$ and $\tau^{HT} = [\tau_x \ \tau_y \ \tau_N]^T$ (Figure D.13 (b)).

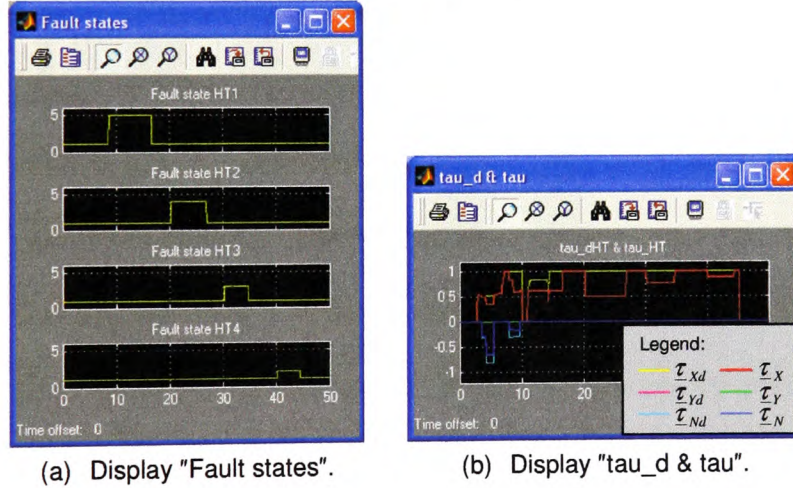


Figure D.13 Displays "Fault states" and "tau_d & tau".

Display Panel. Various information is displayed in this panel (Figure D.14). Scaling factor f^{HT} for S -approximation indicates the position of τ_d^{HT} relative to Φ_p^{HT} : if $f^{HT} = 100\%$, then $\tau_d^{HT} \in \Phi_p^{HT}$. Otherwise, $f^{HT} < 100\% \Rightarrow \tau_d^{HT} \notin \Phi_p^{HT}$.

Velocity & Heading. This display shows time diagrams of linear velocity $v(t) = \|\mathbf{v}_1\|_2$ and heading $\psi(t) = \eta_2(3)$ (Figure D.15). In order to be compatible with the input of the Angular Gauge control "Compass" in Dials & Gauges Blockset (Figure D.14), the heading response is normalised on interval $0^\circ \leq \psi(t) < 360^\circ$. This normalisation introduces jumps in heading responses for cases $\psi(t) < 0^\circ$.

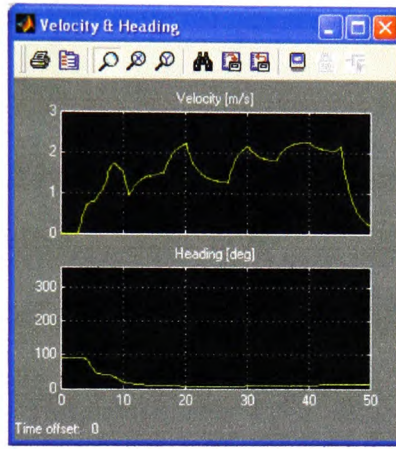
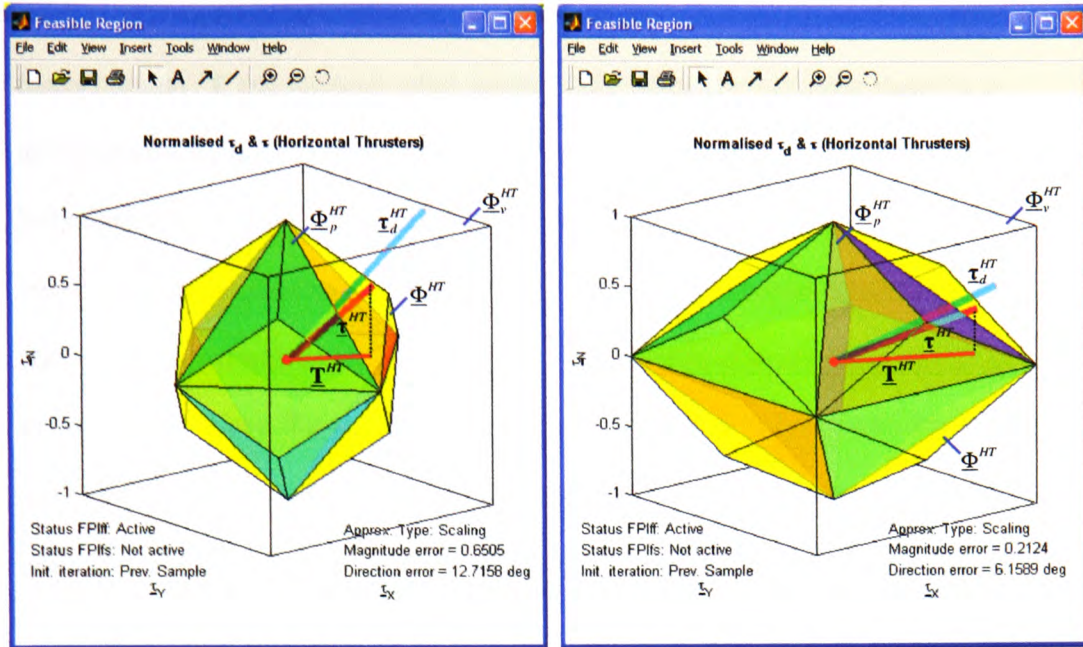


Figure D.15 Display "Velocity & Heading".

Feasible Region. Vectors \mathbf{u}_d^{HT} and \mathbf{u}^{HT} are displayed in Figure D.16 relative to the feasible region for pseudoinverse Φ_p^{HT} and attainable command set Φ^{HT} (see section 5.5.2 for more information). This display is dynamically updated during simulation to reflect any change in thruster's states and command inputs. In addition, values of parameters 1, 2, 3 and 4 from the block parameters dialog box "Approximation" (Figure D.10 (c)) and magnitude & approximation errors are indicated at the bottom of the display, in order to inform the user about the current settings of the hybrid approach for control allocation. Magnitude and approximation errors are defined as $\|\mathbf{u}_d^{HT} - \mathbf{u}^{HT}\|_2$

and $\theta = \arccos \frac{(\mathbf{u}_d^{HT})^T \cdot \mathbf{u}^{HT}}{\|\mathbf{u}_d^{HT}\|_2 \|\mathbf{u}^{HT}\|_2}$, respectively (see Definition 4.1 on page 4-38).

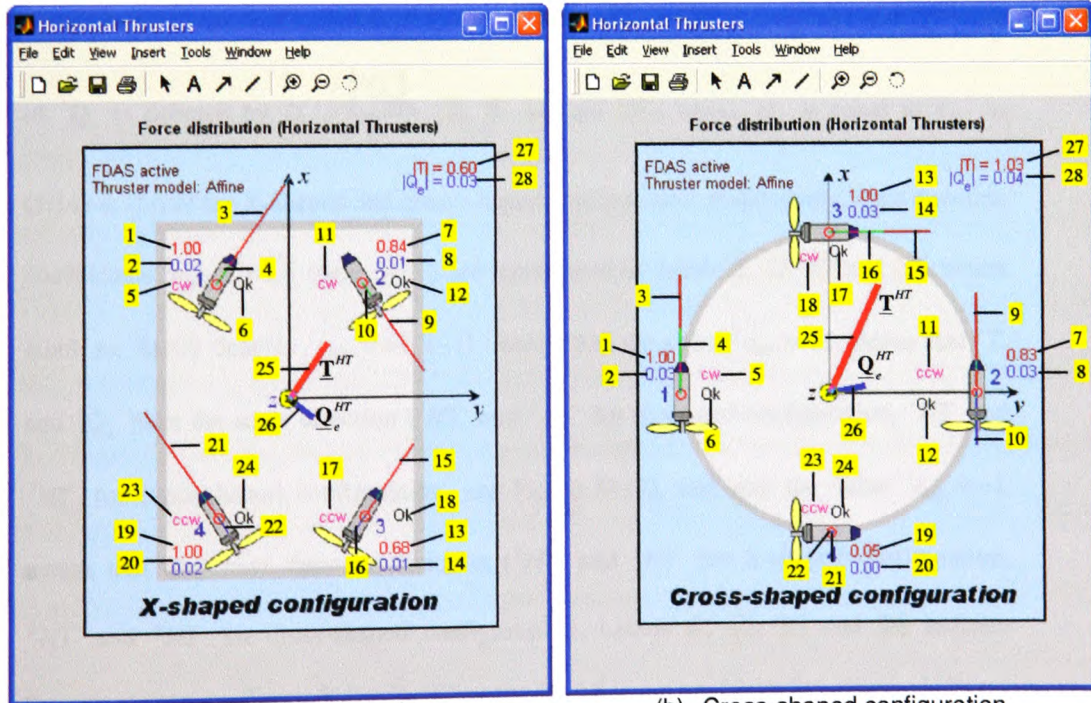
Appendix D: ROV simulator



(a) X-shaped configuration.

(b) Cross-shaped configuration

Figure D.16 Display "Feasible Region".



(a) X-shaped configuration.

(b) Cross-shaped configuration

Figure D.17 Display "Horizontal Thrusters".

Appendix D: ROV simulator

Horizontal thrusters. This display (Figure D.17) visualises the distribution of propulsion forces (thrusts) ${}^i\mathbf{T}$ and moments (shaft torques) ${}^i\mathbf{Q}_e$, exerted by horizontal thrusters, in the horizontal plane. The display shows the plan (top) view of the vehicle. Axes of the body-fixed frame $\{B\}$ are denoted as follows: longitudinal axes x (directed to front side), transversal axes y (directed to starboard) and normal axes z (directed to bottom). Each iHT is represented with a symbolic picture, showing its physical position and orientation in the horizontal plane. The length of all vectors is scaled to fit the display and to improve visibility. The force vector ${}^i\mathbf{T}$, $i = \overline{1,4}$, is represented by a red line, with origin at the centre point of iHT (3, 9, 15 and 21). The normalised module of ${}^i\mathbf{T}$ is represented by $\underline{T}_i = \frac{\|{}^i\mathbf{T}\|_2}{T_m}$ (1, 7, 13 and 19). The moment vector ${}^i\mathbf{Q}_e$, $i = \overline{1,4}$, is represented by a blue line, with origin at the centre point of iHT (4, 10, 16 and 22). The normalised module of ${}^i\mathbf{Q}_e$ is denoted by $\underline{Q}_{ei} = \frac{\|{}^i\mathbf{Q}_e\|_2}{Q_m}$ (2, 8, 14 and 20), where Q_m is equal to τ_{Nm} in (5.14) & (5.19) for X-shaped and cross-shaped configuration, respectively. Spin direction coefficients ${}^ic_{HT}$, $i = \overline{1,4}$ (page 3-27), are represented by labels 5, 11, 17 and 23, where label *cw* (*ccw*) denotes ${}^ic_{HT} = +1$ (-1). Recall that the value ${}^ic_{HT} = +1$ means that ${}^i\mathbf{T}$ and ${}^i\mathbf{Q}_e$ have the same direction (1HT and 2HT for X-shaped configuration, 1HT and 3HT for cross-shaped configuration, see Figure D.17), and that the value ${}^ic_{HT} = -1$ means that they have opposite direction (3HT and 4HT for X-shaped configuration, 2HT and 4HT for cross-shaped configuration). Labels 6, 12, 18 and 24 indicate thruster states (see Figure D.9 (c)). When the FDAS is active, labels *Ok*, *Jammed*, *Heavy j.*, *Broken* and *Unknown* are used to represent the states "Fault-free", "Jammed propeller", "Heavy jammed propeller", "Broken propeller" and "Unknown state", respectively. When

Appendix D: ROV simulator

the FDAS is not active, label *Ok* denotes "Fault-free" state, and the label *Off* denotes uncontrollable thruster, selected using combo box 5 in the block parameters dialog box, shown in Figure D.9 (b). The vector of total forces $\underline{\mathbf{T}}^{HT}$, exerted by horizontal thrusters, is represented by a thick red line 25, with origin at *CG*. Similarly, the vector of total shaft torques $\underline{\mathbf{Q}}_e^{HT}$ is represented by a thick blue line 26, with origin at *CG*. Modules of $\underline{\mathbf{T}}^{HT}$ and $\underline{\mathbf{Q}}_e^{HT}$ are represented by 27 and 28, respectively. Vector $\underline{\mathbf{T}}^{HT}$ is normalised such that it represents the orthogonal projection of vector $\underline{\mathbf{t}}^{HT}$ in $\underline{\mathbf{z}}_x - \underline{\mathbf{z}}_y$ plane, as indicated in Figure D.16.

Virtual Reality Display

User interface is further enhanced with a virtual underwater world using the Virtual Reality Toolbox³. Simulating the ROV simulator generates signal data for ROV dynamics and kinematics (vectors ${}^B\mathbf{v}(t)$ and ${}^E\mathbf{\eta}(t)$). By connecting the Simulink model to a virtual world, this data can be used to control and animate the virtual underwater world. The connection between the Simulink model and virtual underwater world is shown in Figure D.18. A brief description of individual components is given in the following.

Underwater world. The block "Underwater world" (block 1 in Figure D.18) provides the GUI interface to output signals from Simulink to the virtual underwater world. Double click on this block opens the block parameters dialog box, shown in Figure D.19. Description of individual parameters is given in the following.

³ Virtual Reality Toolbox is a solution for viewing and interacting with dynamic systems in a three-dimensional virtual reality environment. It extends the capabilities of MATLAB and Simulink into the world of virtual reality graphics. The Virtual Reality Modelling Language (VRML) is used to define a virtual world that is displayed with a VRML viewer and connected to a Simulink model with VR sink block. See Virtual Reality Toolbox User's Guide (2002) for more information.

Appendix D: ROV simulator

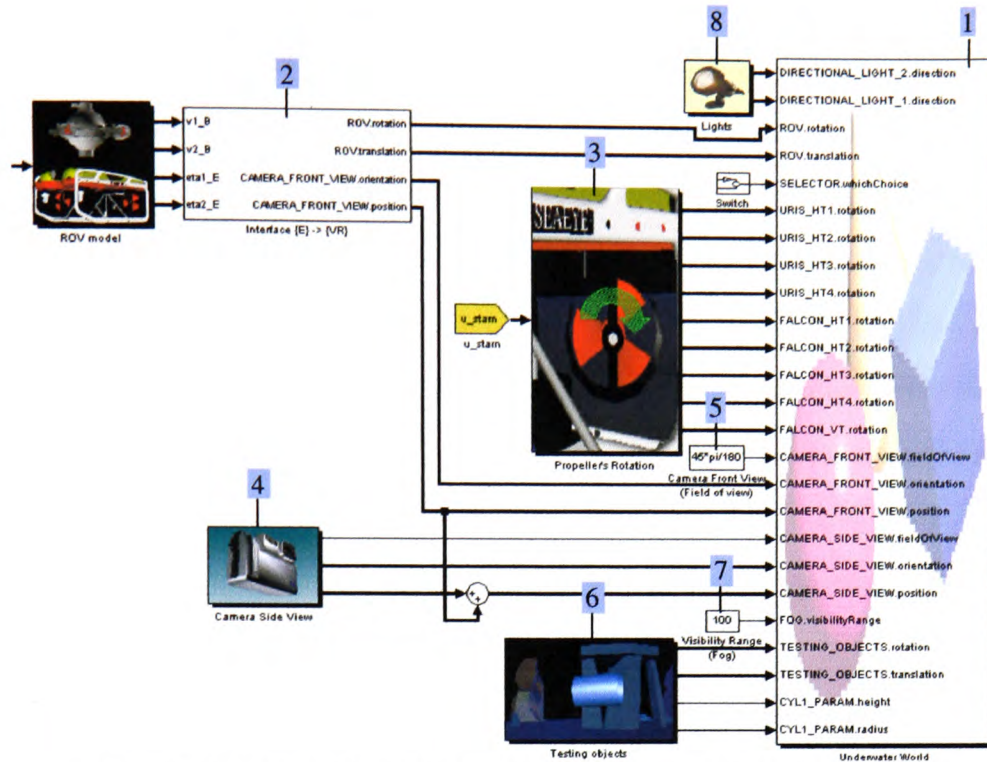


Figure D.18 Connection between Simulink model and virtual underwater world.

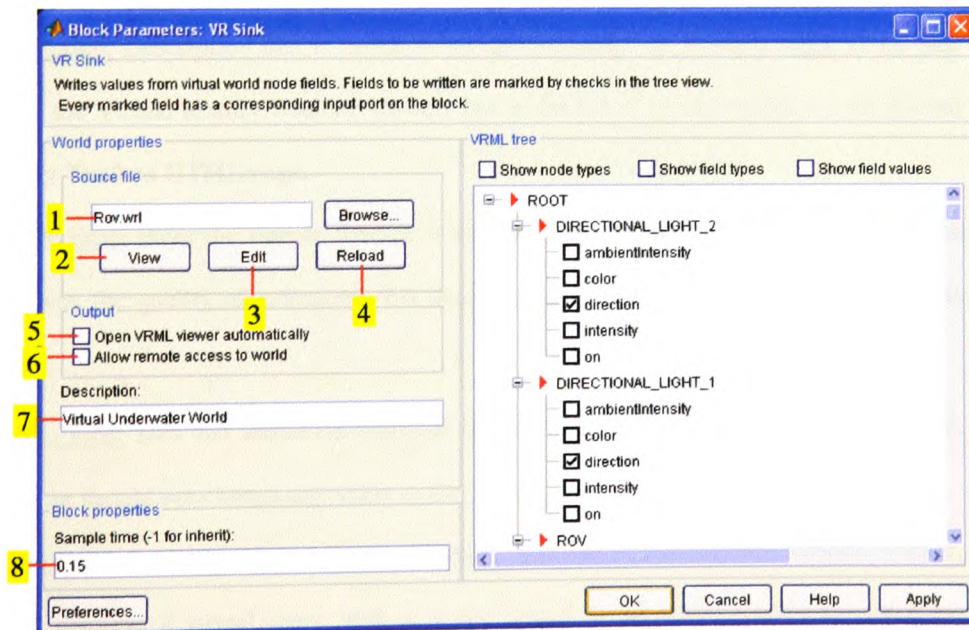


Figure D.19 Block parameters dialog box "Underwater World".

Appendix D: ROV simulator

Source file: VRML file name **1** specifying the virtual world which this block is connected to. The View button **2** allows viewing the world in the Virtual Reality Toolbox viewer or a Web browser. The Edit button **3** launches an external VRML editor, and the Reload button **4** reloads the world after the changes are made. By default, the full path to the associated `.wrl` file appears in this text box. If only the filename is entered in this box, the Virtual Reality Toolbox assumes that the `.wrl` file resides in the same directory as the model file. The default filename of the virtual underwater world is `rov.wrl`, and this file resides in the same directory `C:\WINDOWS\Desktop\PhD\Matlab\ROV` as the main file `rov.mdl`.

Open VRML viewer automatically: If this check box is checked, the default VRML viewer displays the virtual world after loading the Simulink model.

Allow remote access to world: If this check box is checked, the virtual world is accessible for viewing on a client computer. If it is not selected, the world is visible only on the host computer (see Virtual Reality Toolbox User's Guide (2002) for more information).

Description: Description that is displayed in all virtual reality object listings, in the title bar of the Virtual Reality Toolbox viewer, and in the list of virtual worlds on the Virtual Reality Toolbox HTML page.

Sample time: Enter the sample time or -1 for inherited sample time. Lower sample time improves the quality of animation, but decrease simulation speed. The default value is `0.15 s`.

VRML Tree: This box shows the structure of the VRML file and the virtual world itself. The user should not change any value in the VRML Tree.

Figure D.20 shows the Virtual Reality Display. It is the Virtual Reality Toolbox Viewer, which displays a virtual scene with a control panel at the bottom. Only two of these controls (Viewpoint control and Headlight toggle) are used in the ROV simulator. These

Appendix D: ROV simulator

controls will be described in the following. More information about the other controls can be found in Virtual Reality Toolbox User's Guide (2002).

Viewpoint Control: There are three buttons on the control panel that affect the viewpoint of the scene. The centre circular button resets the camera to the current viewpoint. Two different viewpoints are defined in the virtual underwater world: *Camera Front View* and *Camera Side View*. The right and left arrows associated with Viewpoint Control can be used to browse through these predefined viewpoints. It is also possible to use the Page Up and Page Down keys to navigate through these viewpoints.

Headlight toggle: This control is used to turn the camera headlight and the lighting of the scene on or off. When Headlight is off, the camera does not emit light. It is recommended to turn the camera headlight off during simulation, since two sources of directional lights are already available to light the scene.

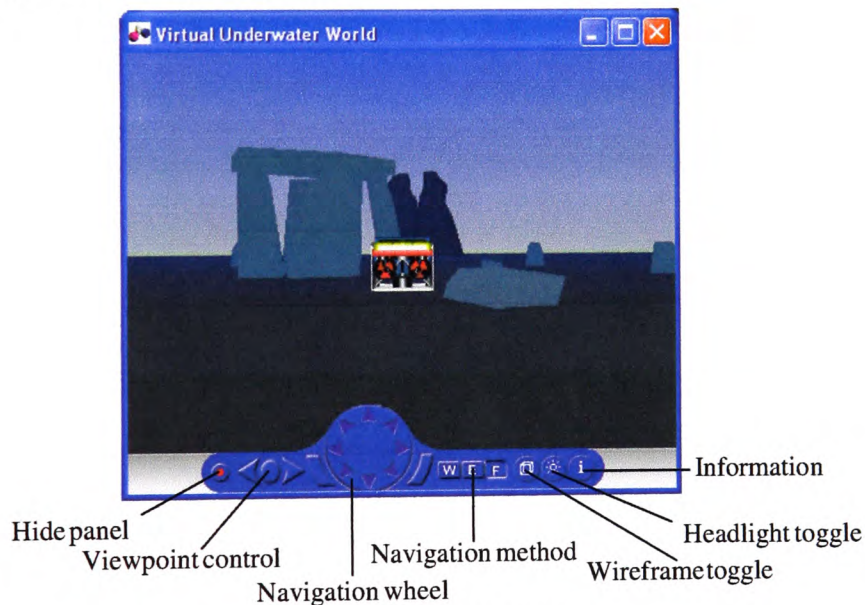


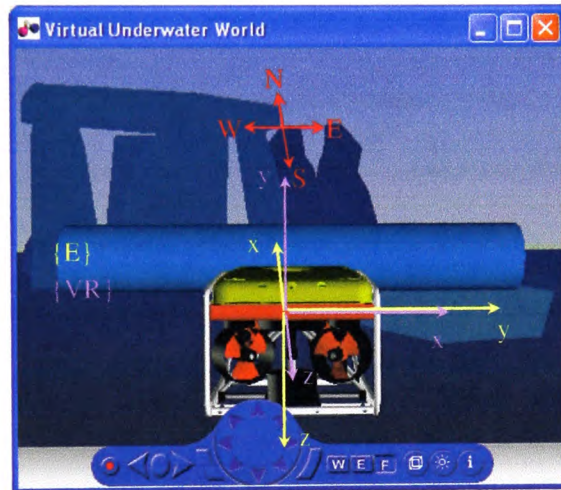
Figure D.20 Virtual Reality Display.

Interface $\{E\} \rightarrow \{VR\}$. VRML uses the right-handed Cartesian coordinate system $\{VR\}$, which is different from the MATLAB graphics coordinate system. VRML uses the world coordinate system in which the x -axis points to the right, y -axis points upward and the

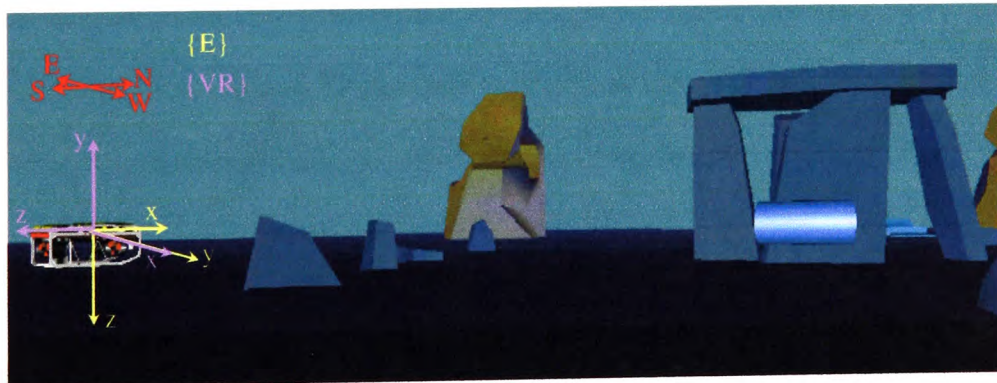
Appendix D: ROV simulator

z -axis places objects nearer or farther from the front of the screen. Each rotation in VRML requires four parameters (three coordinates of principal axes unit vector and principal angle). Because of this, quaternions are a natural way for attitude representation of objects in a virtual world.

Position and orientation of $\{VR\}$ relative to $\{E\}$ is indicated in Figure D.21, which displays FALCON in its initial position ${}^E\eta_1(0) = [0 \ 0 \ 0]^T$ and ${}^E\eta_2(0) = [0 \ 0 \ 0]^T$ in the virtual underwater world. The scene is displayed from two different perspectives (rear view, Figure D.21 (a), and side view, Figure D.21 (b)).



(a) Rear view.



(b) Side view.

Figure D.21 Coordinate frames $\{E\}$ and $\{VR\}$ in virtual underwater world.

Appendix D: ROV simulator

Origins of $\{E\}$ and $\{VR\}$ was chosen to coincide. Position vectors ${}^E\mathbf{\eta}_1 = [x_E \ y_E \ z_E]^T$ and ${}^{VR}\mathbf{\eta}_1 = [x_{VR} \ y_{VR} \ z_{VR}]^T$ are related by

$$\begin{aligned} -x_E &\rightarrow +z_{VR} \\ +y_E &\rightarrow +x_{VR} \\ -z_E &\rightarrow +y_{VR} \end{aligned} \quad (D.2)$$

Orientation vectors ${}^E\mathbf{\eta}_2 = [\phi_E \ \theta_E \ \psi_E]^T$ and ${}^{VR}\mathbf{\eta}_2 = [\phi_{VR} \ \theta_{VR} \ \psi_{VR}]^T$ are related by

$$\begin{aligned} -\phi_E &\rightarrow +\psi_{VR} \\ +\theta_E &\rightarrow +\phi_{VR} \\ -\psi_E &\rightarrow +\theta_{VR} \end{aligned} \quad (D.3)$$

The block "Interface $\{E\} \rightarrow \{VR\}$ " (block 2 in Figure D.18) performs transformation of ${}^E\mathbf{\eta}_1 \rightarrow {}^{VR}\mathbf{\eta}_1$ and ${}^E\mathbf{\eta}_2 \rightarrow {}^{VR}\mathbf{\eta}_2$ using (D.2) and (D.3). In addition, ${}^{VR}\mathbf{\eta}_2$ is transformed into quaternion form using the "Euler Angles to Quaternions" block from Aerospace Blockset.

Propeller's rotation. The block "Propeller's rotation" (block 3 in Figure D.18) generates signals to animate the rotation of propeller blades. Double click on this block opens the block parameters dialog box, shown in Figure D.22. The angular velocity of propeller blades in animation depends on the performance of the host computer: faster the computer, faster the animation. Parameter AS (I) can be used for fine tuning of animation speed, in order to obtain natural rotational motion of propeller blades.

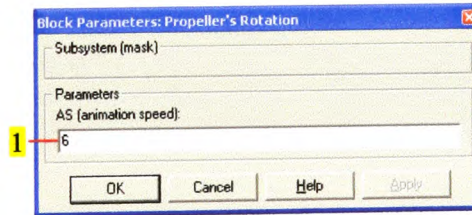


Figure D.22 Block parameters dialog box "Propeller's rotation".

Camera Side View. This viewpoint enables viewing of the scene from the eight fixed points V_1, V_2, \dots, V_8 , as indicated in Figure D.23. These points lie on the sphere \mathcal{S} . The origin of \mathcal{S} coincides with the origin of $\{B\}$. The orientation of \mathcal{S} is fixed in $\{E\}$. Hence,

sphere S moves together with the vehicle, such that points V_i , $i = \overline{1,8}$, have constant distance r from the origin, constant elevation φ from the horizontal plane and constant orientation in $\{E\}$. Only one of these viewpoints can be active at any time. The hat switch (Point of View control of joystick) is used to select/change active point, as shown in Figure D.24.

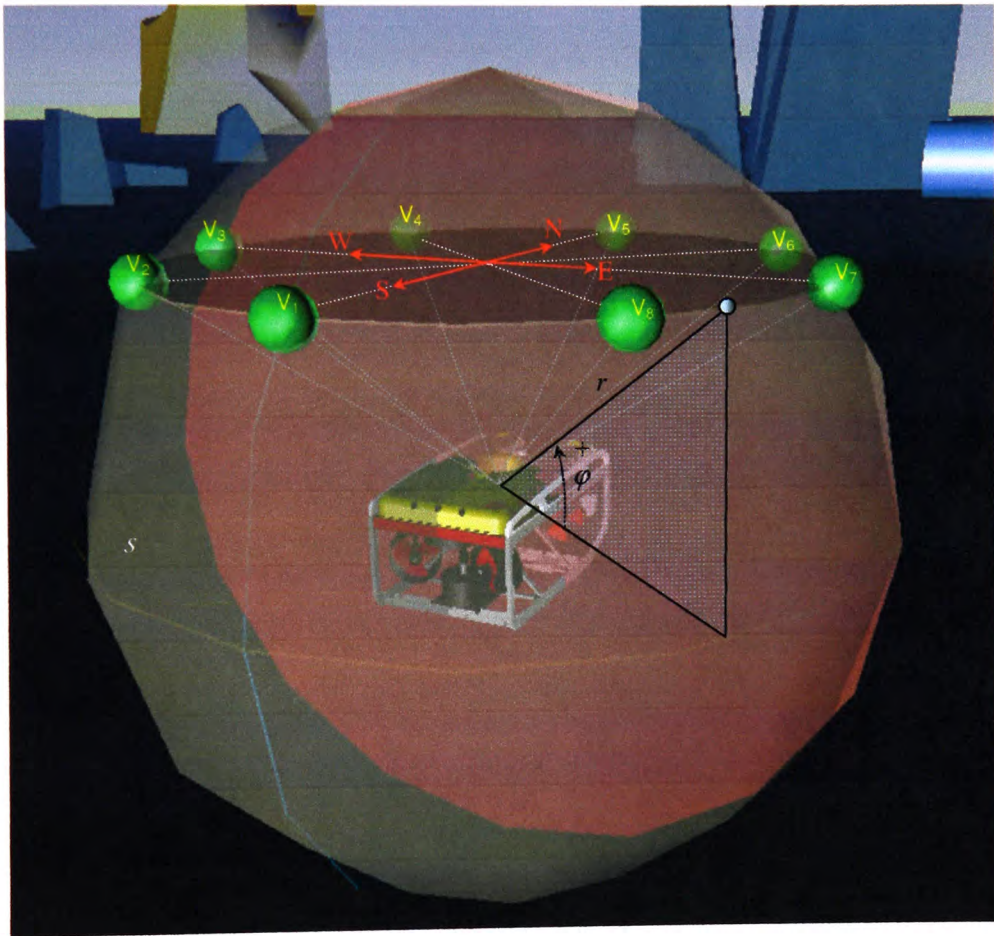


Figure D.23 Viewpoint *Camera Side View*.

Double click on the block "Camera Side View" (block 4 in Figure D.18) opens the block parameters dialog box, shown in Figure D.25. The adjustable parameters are: distance r (1), elevation φ (2) and *Field of View* (3) (*FoV*, the parameter which defines the scene

Appendix D: ROV simulator

perspective). The FoV parameter can be used to obtain telephoto effect (*compressed perspective*) and wide-angle lens effect (*exaggerated perspective*).

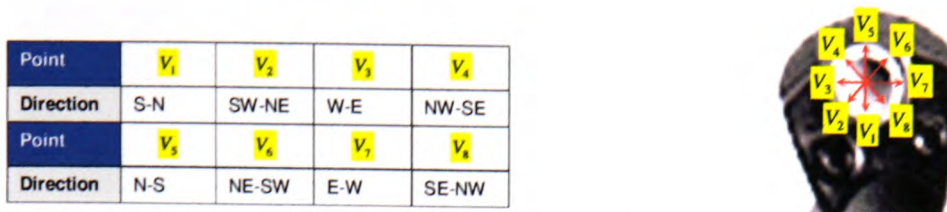


Figure D.24 Hat switch is used to choose which point V_i is active.

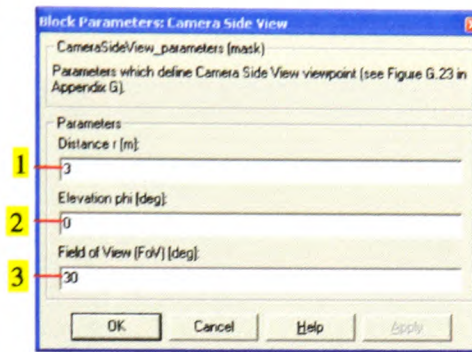


Figure D.25 Block parameters dialog box "Camera Side View".

The concept of *Camera Side View* viewpoint is illustrated by two examples. In the first example, Figure D.26 displays the scene seen from the viewpoint V_1 . The parameter $\varphi = 0^\circ$ is fixed, and parameters r and FoV are variable. Figure D.26 (b) displays the initial view ($r = 3m, FoV = 30^\circ$). Figure D.26 (a) displays the same scene for $FoV = 50^\circ$. Similarly, Figure D.26 (c) shows the scene for $r = 5m$. It can be seen that increasing FoV produces wide-angle lens effect (exaggerated perspective, where a view appears much wider than observer's eyes would normally see it).

In the second example, the same scene is seen from different viewpoints $V_i, i = \overline{1,8}$ (Figure D.27). All parameters have fixed values ($r = 3m, \varphi = 10^\circ, FoV = 45^\circ$). Heading of the FALCON is ${}^E\psi = 137^\circ$.

Appendix D: ROV simulator



(a) $r = 3m, \varphi = 0^\circ, FoV = 50^\circ$. (b) $r = 3m, \varphi = 0^\circ, FoV = 30^\circ$. (c) $r = 5m, \varphi = 0^\circ, FoV = 30^\circ$.

Figure D.26 Different perspectives obtained from the viewpoint V_1 by varying r and FoV .

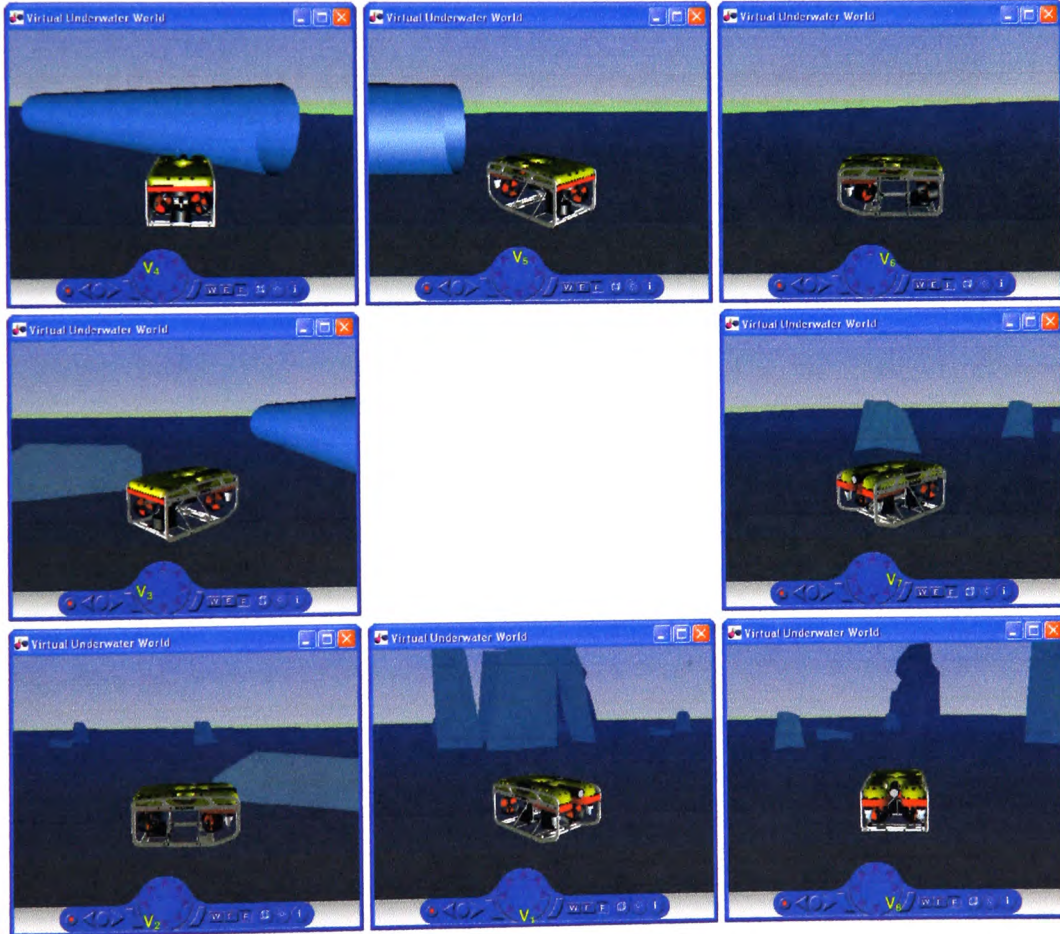


Figure D.27 Scene seen from different viewpoints V_i , $i = \overline{1,8}$ with fixed parameters $r = 3m$, $\varphi = 10^\circ$ and $FoV = 45^\circ$.

Appendix D: ROV simulator

Camera Front View. This is a body-fixed dynamic viewpoint, bounded to the vehicle, which gives the picture similar to one from the on-board camera. The position of this viewpoint was chosen such that the small part of the front side of the vehicle is visible (see Figure D.28). The *FoV* parameter, which defines perspective, can be adjusted using the block parameter dialog box, associated with the block 5 in Figure D.18.

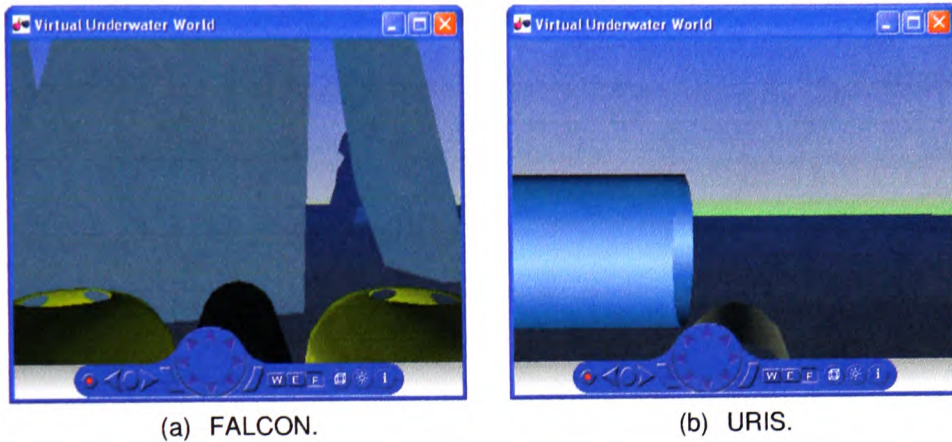


Figure D.28 Viewpoint *Camera Front View*.

Testing objects. Relative position between objects in the virtual underwater world⁴ is shown in Figure D.29. The ROV (FALCON) is shown in the initial position (origin of the Earth-fixed frame). Three particular objects (the rock with a hole in the middle, the long pipe (cylinder) and "Stonehenge"-like group of rocks) are used in examples in section 6.3 to evaluate the performance of the FDAS. Before the simulation is started, the initial positions P_1 and P_2 can be used to place the vehicle close to the pipe or the rock with the hole, respectively, as indicated in Figure D.30. Double click on the block "Testing objects" (block 6 in Figure D.18) opens the block parameters dialog box, shown in Figure D.31. Parameters in this dialog box define the geometry of the pipe. By default,

⁴ Virtual underwater world in ROV simulator was inspired by wonderful underwater world "Ocean Walk", designed by Ryoichiro Debachi (<http://www.atom.co.jp/vrml2/ocean>)

Appendix D: ROV simulator

the pipe is located at the origin of $\{VR\}$ and oriented vertically. Parameter 1 defines translation of the pipe relative to origin. Parameter 2 determines rotation of the pipe relative to default vertical orientation. Figure D.32 displays orientation of the pipe for the case $Rotation = [0 \ 0 \ 1 \ -80^\circ]$. Parameters 3 and 4 define the height (length) and radius of the pipe.

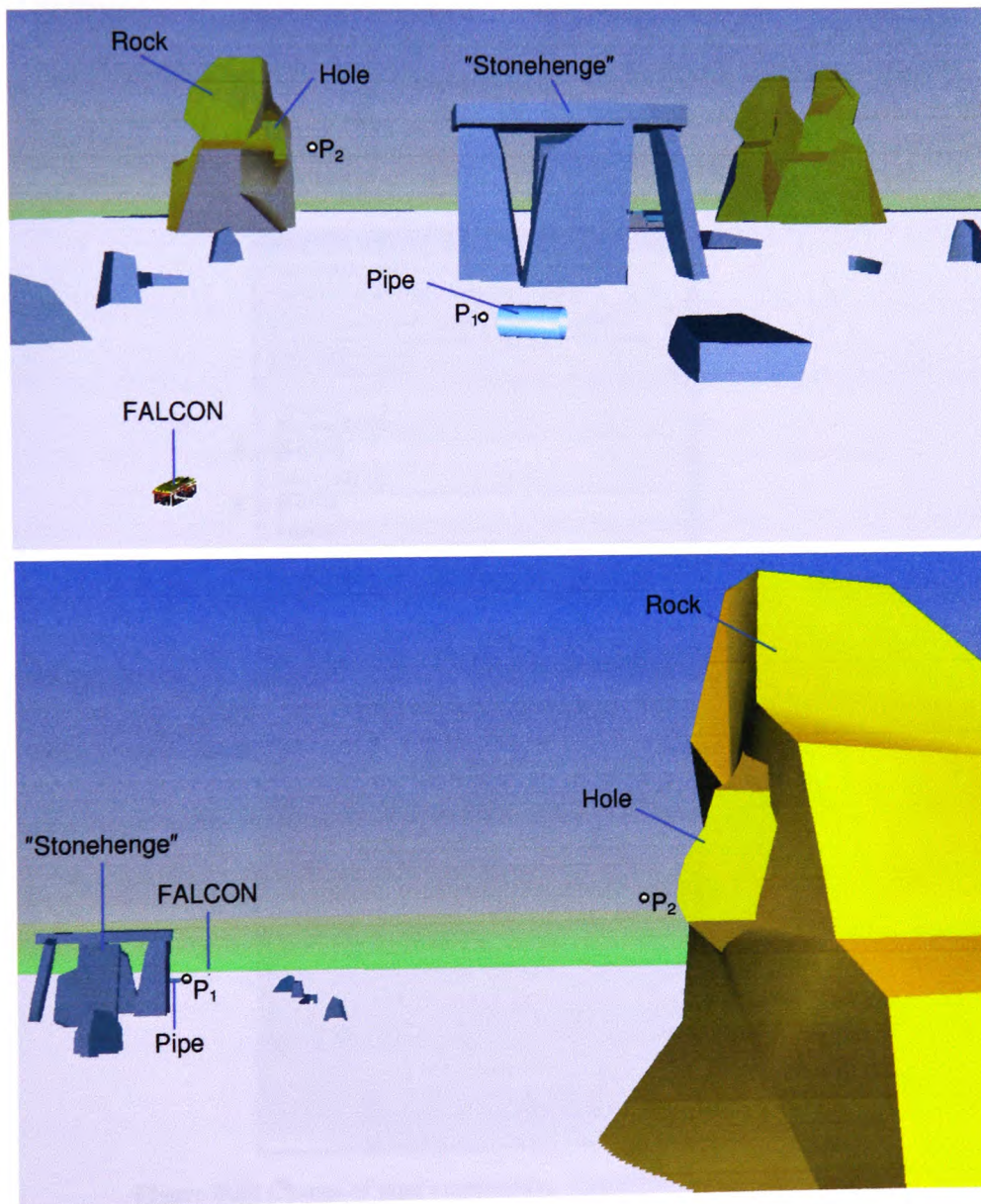
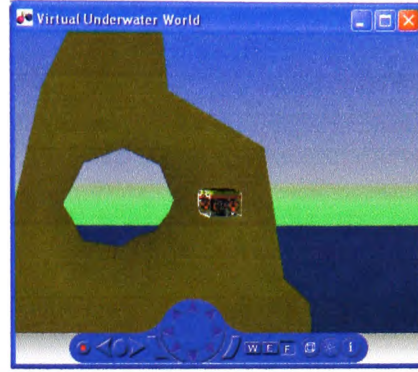


Figure D.29 Relative position between objects in the virtual underwater world.

Appendix D: ROV simulator



(a) P_1 (${}^E\eta_{10}=[30 \ -16 \ 0]$, ${}^E\eta_{20}=[0 \ 0 \ 90^\circ]$) (b) P_2 (${}^E\eta_{10}=[270 \ -155 \ 0-35]$, ${}^E\eta_{20}=[0 \ 0 \ 254^\circ]$)

Figure D.30 Initial positions of the vehicle, close to the pipe and the rock with the hole.

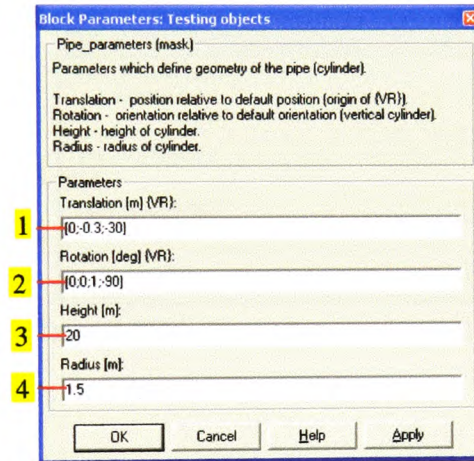


Figure D.31 Block parameters dialog box "Testing objects".



Figure D.32 Change of pipe's orientation: $Rotation=[0 \ 0 \ 1 \ -80^\circ]$.

Appendix D: ROV simulator

Visibility range. In real operating conditions, the visibility range varies from a few meters to hundreds of meters, depending on the cleanliness of the water. Different visibility ranges can be simulated using the fog effect. The fog colour (dark blue) and fog type (exponential) are fixed parameters and cannot be changed from the ROV simulator. The visibility range of the fog can be changed using the block 7 in Figure D.18.

Lights. The light in the virtual underwater world is provided from the two light sources (directional lights *Light 1* and *Light 2*). Double click on the block "Lights" (block 8 in Figure D.18) opens the block parameters dialog box, shown in Figure D.33. Parameters 1 and 2 define direction of the light sources *Light 1* and *Light 2*, respectively.

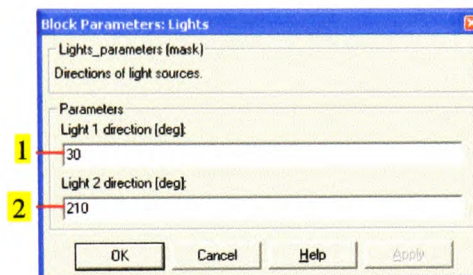


Figure D.33 Block parameters dialog box "Lights".

Run-time behaviour

One of the features of the ROV simulator is dynamic updating. This means that, while a simulation is running, any change of parameters can be performed without need to stop or pause the simulation. For example, during simulation the thruster configuration can be changed from the X-shaped to cross-shaped; propeller spin directions can be changed; thruster model can be changed from affine to bilinear; different faults can be injected using joystick buttons; different viewpoints of the scene can be selected, etc. All displays and windows are automatically updated to accommodate these changes, and their effects can be immediately observed and analysed.



References

- FOSSSEN, T.I. (2002). *Marine Control Systems*. Marine Cybernetics AS, Trondheim, Norway.
- LAUVDAL, T. and FOSSSEN, T. I. (1994). *MATLAB simulation program for marine and flight vehicles*. NTH report nr. 94-43-W. University of Trondheim, Norway.
- The MathWorks Inc. (2002). *Virtual Reality Toolbox User's Guide*. Documentation for Virtual Reality Toolbox 3.0.

Appendix E: Published Papers & Awards

List of published and submitted papers

1. OMERDIC, E. and ROBERTS, G.N. (2003). Thruster Fault Accommodation for Underwater Vehicles. *1st IFAC Workshop on Guidance and Control of Underwater Vehicles GCUV'03*. 9th –11th April 2003, Newport, South Wales, UK.
2. OMERDIC, E., ROBERTS, G.N. and RIDAO, P. (2003). Fault Detection and Accommodation for ROVs. *6th IFAC Conference on Manoeuvring and Control of Marine Craft MCMC'2003*. 17-19 September 2003, Girona, Spain.
3. OMERDIC, E. and ROBERTS, G.N. (2004). Thruster fault diagnosis and accommodation for open-frame underwater vehicles. Special issue GCUV 2003, *Control Engineering Practice*, to appear.
4. SOWERBY, N.W.A., PEREZ, T., OMERDIC, E. and ROBERTS, G.N. (2004). System identification and fault accommodation for thruster propelled UUVs. *Advances in technology for underwater vehicles (ATUV)*. Two day international conference. London, UK.
5. OMERDIC, E. and ROBERTS, G.N. (2004). Extension of feasible region of control allocation for open-frame underwater vehicles. *IFAC Conference on Control Applications in Marine Systems (CAMS 2004)*, Ancona, Italy.

List of awards

1. SUT prize for best multimedia presentation (GCUV 2003, Paper 1.)
2. IFAC prize for best on-line demonstration (MCMC 2003, Paper 2.)

THRUSTER FAULT ACCOMMODATION FOR UNDERWATER VEHICLES

E. Omerdic and G.N. Roberts

University of Wales College, Newport
Mechatronics Research Centre, NP20 5XR, UK
email: {edin.omerdic, geoff.roberts}@newport.ac.uk

Abstract: This paper introduces a new approach associated with fault detection and accommodation for remotely operated underwater vehicles. The proposed fault detection and accommodation system (FDAS) consists of two subsystems: fault detection subsystem (FDS) and fault accommodation subsystem (FAS). The FDS uses fault detector units (FDU), associated with each thruster, to monitor their healthy state. Robust and reliable fault detection units are based on integration of self-organising maps and fuzzy logic clustering methods. The FAS uses information provided by the FDS to accommodate faults and perform an appropriate reconfiguration. A control energy cost function is used as the optimisation criteria. In fault-free and faulty cases the FAS finds the optimal solution, which minimise this criteria. The feasible region concept, related with the problem of thruster velocity saturation, is described. *Copyright © 2003 IFAC*

Keywords: Fault detection, Fault-tolerant systems, Feasible region, Optimal design, Simulators.

1. INTRODUCTION

Underwater vehicles are liable to faults or failures during underwater missions. Thrusters are the most common source of faults. In all but the most trivial cases the existence of a fault may lead to cancelling the mission. Implication of small faults could be very expensive and time consuming. Although good design practice tries to minimize the occurrence of faults and failures, there is a certain probability that faults will occur. Recognition that such events do occur enables system designers to develop strategies by which the effect they exert is minimised. In the case of a partial or total fault in a horizontal thruster it is possible to reconfigure the control system in order to keep the high level of the control performance and complete a mission. This paper introduces new approach associated with fault detection and accommodation for remotely operated underwater vehicles to cope with this problem.

Rauch (1994) proposes three approaches for control reconfiguration of the underwater vehicles:

- to have specific procedures to accommodate modelled (anticipated) potential faults,

- to represent unmodelled (unanticipated) faults as unknown forces and moments,
- to use a high level system which replans the mission to mitigate the effects of the fault.

In the first approach the procedure monitors system response to look for specific failure modes. When a failure mode is recognized, the procedure implements the appropriate control change. Control reconfiguration can be based on stored control laws tailored to each anticipated fault condition. The proposed fault detection and accommodation system belongs to this group. The second approach represents the unmodelled (unanticipative) faults as unknown forces or moments acting on the vehicle (Sullivan, *et al.*, 1992). Forces and moments are estimated on-line first. Then, the adaptive control is adjusted to compensate for these disturbances. A rapid estimate recognises the existence of a fault, but a slower, more deliberate estimate determines the correlation of the fault with a force or moment. The third approach treats high-level fault accommodation (Farrel, *et al.*, 1993). The implementation extends the range of mission responses by approximating decision-making of an experienced crew. The two facets of the response are reactive capability (which

directly accommodates to the recognised faults) and assessment capability (which determines how to modify the mission to accommodate faults).

A fault-tolerant system for use in an experimental AUV was outlined in (Yang, *et al.*, 1998; Yang, *et al.*, 1999). The system was subdivided into individual fault-tolerant subsystems for dealing with thrusters and sensor failures separately. The thruster subsystem consisted of a rule base for detection and isolation purposes, and an algorithm for reconfiguring the thrusters control matrix by eliminating the corresponding column to accommodate the failure. Only a total fault (failure) of the thruster was considered. The authors used a constraint-based method instead of the pseudo-inverse method to find inverse of the thruster configuration matrix. Experimental investigation was conducted on a 6 DoF AUV, ODIN at the University of Hawaii to evaluate the performance of the proposed approaches and experimental results showed that the overall system was capable of performing effectively.

If the number of control inputs is equal to or more than the number of controllable DoF, it is possible to find an “optimal” distribution of the control energy, which minimises the quadratic energy cost function i.e. a measure of the control effort (Fossen, 1995). This approach uses the generalised inverse matrix to find the optimal control vector. The problem of thruster velocity saturation was not considered.

Ideas from (Rauch, 1994; Yang, *et al.*, 1998; Yang, *et al.*, 1999; Fossen, 1995) were used as the initial basis for the design of the novel FDAS for a ROV. A general thruster model is described in the second section. The problem of the optimal distribution of propulsion and control forces, exerted by thrusters, is described in the third section. An original solution of the problem is proposed, for two common configurations of horizontal thrusters. The concept of a feasible region, described in the fourth section, has been developed in order to cope with the problem of thruster velocity saturation. This problem is neglected by many authors, who found “optimal” control laws, without taking care of saturation. The significance of this problem is discussed. The fifth section describes architecture of the proposed FDAS. Finally, conclusions are given in the sixth section.

The proposed fault detection and accommodation system has been implemented as a Matlab Simulink model and includes a non-linear model of an ROV with 6 DoF, thruster dynamics and a hand control unit. Different fault conditions can be simulated in order to investigate the performance of the proposed FDAS.

2. THRUSTER MODEL

Many modern underwater vehicles have four horizontal thrusters that control three degrees of freedom (surge, sway and yaw). Hence, the control system for motion in the horizontal plane is over actuated and there is redundancy, which provides a space to perform reconfiguration in the case of a fault in a single horizontal thruster. Two underwater vehicles with different thruster configurations are used to demonstrate the performance of the proposed FDAS (FALCON, SeaEye Marine Ltd. and URIS, University of Girona, see Fig. 1.). The thruster configuration of URIS has two horizontal and two vertical thrusters. Due to its flexibility, vertical thrusters were reoriented to the horizontal in order to examine the performance of the proposed FDAS.

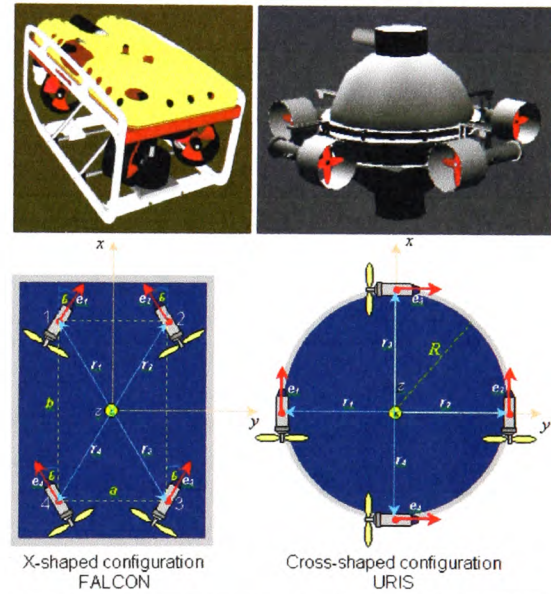


Fig. 1. Two common configurations of the horizontal thrusters: FALCON (left) and URIS (right).

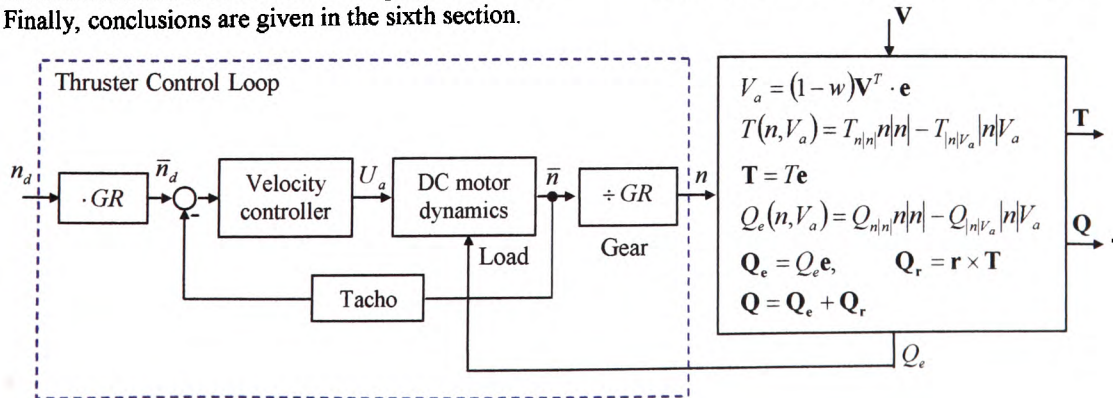


Fig. 2. Simplified diagram showing full thruster model, including thruster control loop dynamics.

A full thruster model, including dynamics of the thruster control loop, is shown in Fig. 2. A DC motor, designed for underwater operating conditions, drives a thruster in the ROV. Input to the thruster control loop is a control variable n_d (desired angular velocity). The motor is equipped with a tachometer, which measures actual angular velocity. A gearbox is used to reduce the output speed and to enhance the output torque. Orientation and position of the thruster in the body-fixed frame are defined with vectors \mathbf{e} and \mathbf{r} , respectively. Each thruster exerts thrust \mathbf{T} and torque \mathbf{Q}_e . The thrust \mathbf{T} also generates moment $\mathbf{Q}_r = \mathbf{r} \times \mathbf{T}$, so the total moment vector exerted by a thruster is given by $\mathbf{Q} = \mathbf{Q}_e + \mathbf{Q}_r$. Contributions of each thruster are summed together to form force vector $\boldsymbol{\tau}$. The undesired effects of torques \mathbf{Q}_e on the control performance can be reduced by carefully choice of propeller blade's spin direction: for the couple of symmetrical thrusters blades should have opposite spin direction i.e. one should rotate clockwise and other anti clock-wise. In such a way, resulting moment from the blade's angular motion will counteract each other and the net angular moment for each pair of symmetrical thrusters in total force vector $\boldsymbol{\tau}$ will be zero, if they rotate with the same velocity. The full thruster model includes motor dynamics, tacho and velocity controller inside the thruster control loop. However, the dynamics of the thruster control loop are much faster than the dynamics of the vehicle and can be initially neglected.

In the general case, the thrust T and torque Q , exerted by a thruster, are complicated functions depending on the vehicle's velocity vector \mathbf{V} and control variable n . The bilinear thruster model, shown in Fig. 2. and described in (Fossen, 1995; Fossen and Blanke, 2000), can be used to approximate these functions. However, in practical applications, the bilinear thruster model can be approximated by an affine model, where ROV velocity \mathbf{V} is equal to zero. Hence, optimal distribution of propulsion and control forces will be initially found with the assumption that the relationship between propeller thrust/torque and control variable is given by the affine thruster model. The dynamics of the thruster control loop will be initially neglected. The influence of these factors on the performance of the ROV control system can be later investigated in the ROV simulator using the bilinear thruster model with the dynamics of the thruster control loop included.

3. OPTIMAL DISTRIBUTION OF PROPULSION FORCES

The input command vector $\boldsymbol{\tau}_d = [X_d, Y_d, N_d]^T$ for the motion in the horizontal plane, generated by the hand control unit, belongs to the input space \mathfrak{I}

$$\mathfrak{I} = \{(X_d, Y_d, N_d) : |X_d| \leq X_{\max}, |Y_d| \leq Y_{\max}, |N_d| \leq N_{\max}\} \quad (1)$$

and has three components (desired forces in x and y directions and desired moment about z axis in the body-fixed frame). The problem is to find the optimal control vector $\mathbf{u} = [u_1, u_2, u_3, u_4]^T, u_i = n_i |n_i|$, which belongs to the control space \mathfrak{N}

$$\mathfrak{N} = \{(u_1, u_2, u_3, u_4)\} \quad (2)$$

that should be applied to drive each horizontal thruster such that the total force vector $\boldsymbol{\tau} = [X, Y, N]^T$ exerted by thrusters, which belongs to the output space \mathfrak{R}

$$\mathfrak{R} = \{(X, Y, N) : |X| \leq X_{\max}, |Y| \leq Y_{\max}, |N| \leq N_{\max}\} \quad (3)$$

is as close as possible to the input vector $\boldsymbol{\tau}_d$. For the affine thruster model the relation between $\boldsymbol{\tau}$ and \mathbf{u} is described by

$$\boldsymbol{\tau} = \mathbf{B}\mathbf{u} \quad (4)$$

where \mathbf{B} is a thruster configuration matrix. General expressions for the matrix \mathbf{B} are

URIS (cross-shaped configuration)

$$\mathbf{B} = \begin{bmatrix} C & C & 0 & 0 \\ 0 & 0 & C & C \\ RC & -RC & RC & -RC \end{bmatrix} \quad (5)$$

FALCON (X-shaped configuration)

$$\mathbf{B} = \begin{bmatrix} C \cos \theta & C \cos \theta & C \cos \theta & C \cos \theta \\ C \sin \theta & -C \sin \theta & C \sin \theta & -C \sin \theta \\ CA & -CA & -CA & CA \end{bmatrix} \quad (6)$$

where $C = T_{|n|n}$ and $A = 0.5(b \sin \theta + a \cos \theta)$. A very important issue, which shouldn't be neglected, is thruster velocity saturation i.e. the thruster cannot rotate faster than its maximum speed. This leads to a mapping from the original control space \mathfrak{N} to the constrained control space \mathfrak{N}^*

$$\mathfrak{N}^* = \{(u_1, u_2, u_3, u_4) : |u_i| \leq u_{\max}\} \quad (7)$$

The system of equations (4) has many solutions, because there are 3 equations (3 DoF) and 4 unknowns (4 horizontal thrusters), but only one (optimal) solution will minimise the criteria J (weighted norm of the control vector i.e. control energy cost function), defined by

$$J = \frac{1}{2} \mathbf{u}^T \mathbf{W} \mathbf{u} \quad (8)$$

The problem can be formulated as follows: find $\mathbf{u}^* \in \mathfrak{N}^*$, which minimise criteria (8) subject to constraint (4).

The weighting matrix \mathbf{W} is a positive definite, diagonal matrix, weighting the control energy¹:

$$\mathbf{W} = \begin{bmatrix} w_1 & 0 & 0 & 0 \\ 0 & w_2 & 0 & 0 \\ 0 & 0 & w_3 & 0 \\ 0 & 0 & 0 & w_4 \end{bmatrix} \quad (9)$$

The optimal control vector $\mathbf{u} \in \mathbb{N}$ is obtained by using the general inverse (Fossen, 1995):

$$\mathbf{u} = \mathbf{B}_w^+ \boldsymbol{\tau}_d = (\mathbf{W}^{-1} \mathbf{B}^T (\mathbf{B} \mathbf{W}^{-1} \mathbf{B}^T)^{-1}) \boldsymbol{\tau}_d \quad (10)$$

Using normalised variables $\underline{\mathbf{u}}, \underline{\boldsymbol{\tau}}$ and $\underline{\boldsymbol{\tau}}_d$ simplifies the calculations and improves the readability

$$\underline{\mathbf{u}} = [\underline{u}_1, \underline{u}_2, \underline{u}_3, \underline{u}_4]^T = \left[\frac{u_1}{u_{\max}}, \frac{u_2}{u_{\max}}, \frac{u_3}{u_{\max}}, \frac{u_4}{u_{\max}} \right]^T \quad (11)$$

$$\underline{\boldsymbol{\tau}} = [\underline{X}, \underline{Y}, \underline{N}]^T = \left[\frac{X}{X_{\max}}, \frac{Y}{Y_{\max}}, \frac{N}{N_{\max}} \right]^T \quad (12)$$

$$\underline{\boldsymbol{\tau}}_d = [\underline{X}_d, \underline{Y}_d, \underline{N}_d]^T = \left[\frac{X_d}{X_{\max}}, \frac{Y_d}{Y_{\max}}, \frac{N_d}{N_{\max}} \right]^T \quad (13)$$

Normalised spaces are defined as

$$\underline{\mathbb{Z}} = \{(\underline{X}_d, \underline{Y}_d, \underline{N}_d) : |\underline{X}_d| \leq 1, |\underline{Y}_d| \leq 1, |\underline{N}_d| \leq 1\} \quad (14)$$

$$\mathbb{N}^* = \{(\underline{u}_1, \underline{u}_2, \underline{u}_3, \underline{u}_4) : |\underline{u}_i| \leq 1\} \quad (15)$$

$$\mathbb{R} = \{(\underline{X}, \underline{Y}, \underline{N}) : |\underline{X}| \leq 1, |\underline{Y}| \leq 1, |\underline{N}| \leq 1\} \quad (16)$$

In normalised form the relationship between $\underline{\mathbf{u}}$ and $\underline{\boldsymbol{\tau}}_d$ is given by the simple matrix $\underline{\mathbf{B}}_w^+$:

$$\underline{\mathbf{u}} = \underline{\mathbf{B}}_w^+ \underline{\boldsymbol{\tau}}_d \quad (17)$$

URIS (cross-shaped configuration)

$$\underline{\mathbf{B}}_w^+ = \frac{1}{\sum_{i=1}^4 w_i} \begin{bmatrix} 2(w_2 + w_3 + w_4) & (w_3 - w_4) & 2(w_3 + w_4) \\ 2(w_1 + w_3 + w_4) & (w_4 - w_3) & -2(w_3 + w_4) \\ (w_1 - w_2) & (w_1 + w_2 + 2w_4) & 2(w_1 + w_2) \\ (w_2 - w_1) & (w_1 + w_2 + 2w_3) & -2(w_1 + w_2) \end{bmatrix} \quad (18)$$

FALCON (X-shaped configuration)

$$\underline{\mathbf{B}}_w^+ = \frac{1}{\sum_{i=1}^4 w_i} \begin{bmatrix} 2(w_3 + w_4) & 2(w_2 + w_3) & 2(w_2 + w_4) \\ 2(w_3 + w_4) & -2(w_1 + w_4) & -2(w_1 + w_3) \\ 2(w_1 + w_2) & 2(w_1 + w_4) & -2(w_2 + w_4) \\ 2(w_1 + w_2) & -2(w_2 + w_3) & 2(w_1 + w_3) \end{bmatrix} \quad (19)$$

The optimal solution consists of three transformations: *inversion*, *scaling* and *realisation* (Fig. 3). *Inversion* uses the relation (17) to transform the input vector $\underline{\boldsymbol{\tau}}_d$ from the 3D input space (unit cube $\underline{\mathbb{Z}}$) to the temporary control vector $\underline{\mathbf{u}}$ in the 4D

control space \mathbb{N} . *Scaling* maps the vector $\underline{\mathbf{u}} \in \mathbb{N}$ to the constrained vector $\underline{\mathbf{u}}^* \in \mathbb{N}^*$ using the formula

$$\underline{\mathbf{u}}^* = k \underline{\mathbf{u}} \quad (20)$$

$$k = \min \left(\frac{u_{\max}}{\max_i |u_i|}, 1 \right) \quad (21)$$

Depending on the position of the input vector $\underline{\boldsymbol{\tau}}_d$ inside the input space, it is possible that one or more components of the control vector $\underline{\mathbf{u}}$ have an absolute value greater than unity, i.e. it is possible that the vector $\underline{\mathbf{u}}$ lies outside the unit cube. In this case, $k < 1$ and scaling multiplies each component of $\underline{\mathbf{u}}$ by k . The resulting vector $\underline{\mathbf{u}}^* \in \mathbb{N}^*$ has reduced size and remains on the boundary of the unit cube, but keeps the same direction as the original vector $\underline{\mathbf{u}}$. This is very important, because in this case vector $\underline{\boldsymbol{\tau}}$ will have the same direction as $\underline{\boldsymbol{\tau}}_d$, but reduced size. It would be wrong to saturate just individual components of $\underline{\mathbf{u}}$ which are greater than one and to leave other components unchangeable, because in this case vectors $\underline{\boldsymbol{\tau}}$ and $\underline{\boldsymbol{\tau}}_d$ would differ in the direction and size. *Realisation* has a different meaning, depending on the method used (simulation or real-time application). For simulation purposes, realisation means actuation of the thrusters with the control vector $\underline{\mathbf{u}}^*$ and generation of the force vector exerted by thrusters $\underline{\boldsymbol{\tau}} = \underline{\mathbf{B}} \underline{\mathbf{u}}^*$. In real-time applications, realisation is performed by denormalisation $\underline{\mathbf{u}}^* \rightarrow \mathbf{u}^*$ and transformation of the control vector \mathbf{u}^* to the control vector \mathbf{n}_d , which is applied as set values for the thrusters' speed.

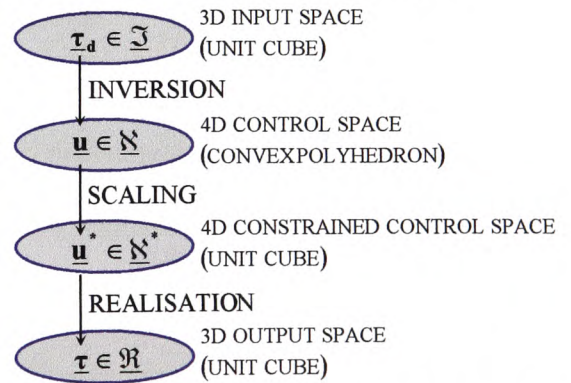


Fig. 3. Optimal solution for the distribution of the propulsion forces.

4. FEASIBLE REGION

The value of the scaling factor k depends on the position of the input vector $\underline{\boldsymbol{\tau}}_d$ in the input space $\underline{\mathbb{Z}}$. If the input vector $\underline{\boldsymbol{\tau}}_d$ belongs to the specific region

¹ $w_i = 1, i = \overline{1,4}$ in a fault-free case (see section 5).

$\Phi \subset \mathfrak{I}$, then $k=1$. Otherwise, $\tau_d \in \mathfrak{I} \setminus \Phi \Rightarrow k < 1$. This leads to the following definition:

Definition 1: Feasible region Φ is defined as

$$\Phi = \{\tau_d \in \mathfrak{I} : k = 1\} \quad (22)$$

In other words, the feasible region is defined as a set of all input vectors from the input space for which the corresponding control vector defined by (17) belongs to a 4D unit cube. Vectors from the input space, which lie outside the feasible region, are mapped by (17) to the control vectors that lie outside the 4D unit cube and must be scaled to fit this cube, because of the constraint (thruster velocity saturation). The shape of the feasible region can be found by solving a set of equations (saturation bounds)

$$|u_i| = 1, \quad i = 1, 2, 3, 4 \quad (23)$$

The feasible region is a convex polyhedron inside the input space determined by intersection of the four pairs of parallel planes defined by (23). Feasible region is shown in Fig. 4.

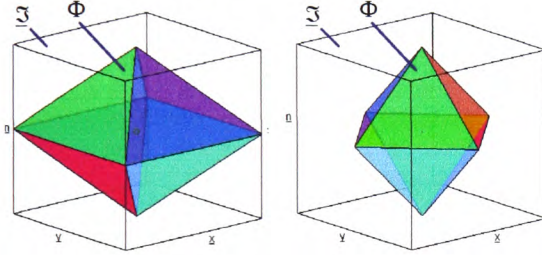


Fig. 4. Input space and feasible region for URIS (left) and FALCON (right).

The following expressions summarise the main properties of the feasible region concept:

$$\tau_d \in \Phi \Rightarrow k = 1 \Rightarrow \underline{u}^* = \underline{u} \Rightarrow \underline{\tau} = \tau_d \quad (24)$$

$$\tau_d \in \mathfrak{I} \setminus \Phi \Rightarrow k < 1 \Rightarrow \underline{u}^* = k\underline{u} \Rightarrow \underline{\tau} = k\tau_d \quad (25)$$

Hence, input vectors that belong to the interior and boundary of the feasible region in the input space are mapped to the same vectors in the output space (one-to-one mapping). Vectors from the exterior of the feasible region in the input space are scaled and mapped onto the boundary of the feasible region in the output space. If thruster velocity saturation is neglected, then scaling is not needed and solution (10) guarantees that each point in the output space can be reached i.e. it performs a one-to-one mapping between full input and output spaces. However, in real applications saturation always exist and the effective part of the output space that can be reached has the shape of the feasible region. The main conclusion of this important issue is that desired input vector should always be located inside or on the boundary of the feasible region for the particular thruster configuration, regardless of how it is generated (either by joystick or control algorithm).

Example 1: This example demonstrates the features of the proposed solution and clarifies the feasible region concept.

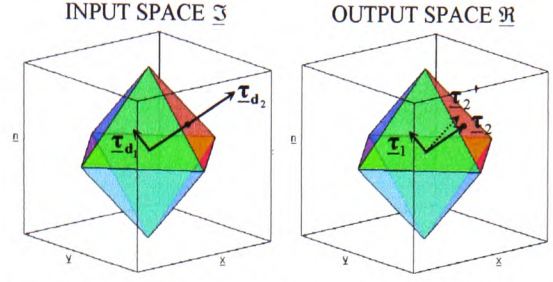


Fig. 5. Input space, output space and feasible region for FALCON, with two characteristic cases.

Fig. 5. displays two characteristic cases described by (24) and (25). In the first case, input vector τ_{d1} belong to the feasible region and resulting output vector τ_1 is the same as input vector (26). In the second case, input vector τ_{d2} lies outside the feasible region and output vector τ_2 (collinear with τ_{d2}) is the best approximation of τ_{d2} , which lies on the boundary of the feasible region in the output space (27). Analysing the temporary control vector \underline{u}_2 it can be seen that $\underline{u}_2(1) = 2.1 > 1$. If only this component is saturated to unity (28), instead of scaling all components, than the resulting output vector τ_2' is not collinear with τ_{d2} .

$$\tau_{d1} = \begin{bmatrix} -0.06 \\ -0.22 \\ +0.24 \end{bmatrix} \Rightarrow \underline{u}_1 = \begin{bmatrix} -0.04 \\ -0.08 \\ -0.52 \\ +0.40 \end{bmatrix} \xRightarrow{(17)} \Rightarrow k_1 = 1 \Rightarrow \begin{cases} \underline{u}_1^* = \underline{u}_1 \\ \tau_1 = \tau_{d1} \end{cases} \quad (26)$$

$$\tau_{d2} = \begin{bmatrix} 0.8 \\ 0.6 \\ 0.7 \end{bmatrix} \Rightarrow \underline{u}_2 = \begin{bmatrix} +2.1 \\ -0.5 \\ +0.7 \\ +0.9 \end{bmatrix} \xRightarrow{(17)} \Rightarrow k_2 = 0.48 \Rightarrow \begin{cases} \underline{u}_2^* = 0.48\underline{u}_2 \\ \tau_2 = 0.48\tau_{d2} \end{cases} \quad (27)$$

$$\tau_{d2} = \begin{bmatrix} 0.8 \\ 0.6 \\ 0.7 \end{bmatrix} \Rightarrow \underline{u}_2 = \begin{bmatrix} +2.1 \\ -0.5 \\ +0.7 \\ +0.9 \end{bmatrix} \Rightarrow \underline{u}_2' = \begin{bmatrix} +1.0 \\ -0.5 \\ +0.7 \\ +0.9 \end{bmatrix} \Rightarrow \tau_2' = \begin{bmatrix} 0.525 \\ 0.325 \\ 0.425 \end{bmatrix} \quad (28)$$

5. THRUSTER FAULT DETECTION AND ACCOMMODATION SYSTEM

The proposed FDAS, shown in Fig. 6., consists of two subsystems: a fault detection subsystem (FDS) and a fault accommodation subsystem (FAS). The FDS uses fault detector units (FDU), associated with each thruster, to monitor their healthy state. The FAS uses information provided by FDS to accommodate faults and perform an appropriate reconfiguration. The FDU provide reliable and fast information about faults in thrusters. It is trained to recognize external faults (such as a jammed or broken propeller) and internal faults (such as low bus voltage, lost communication with control centre etc.).

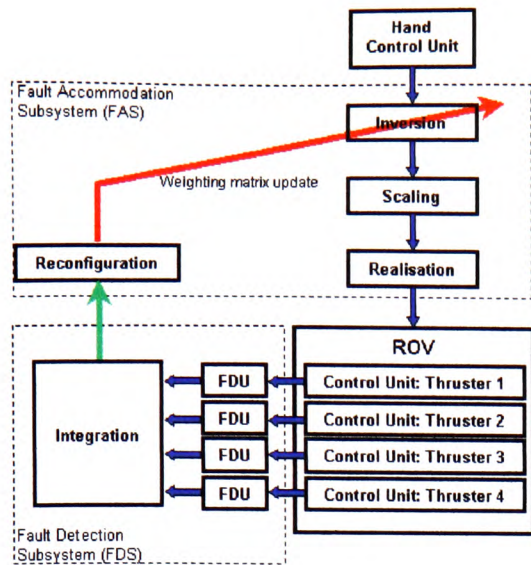


Fig. 6. The proposed FDAS for ROV.

The model-free approach for detection of the external faults integrates self-organising map (SOM) and neuro-fuzzy clustering techniques. More information about FDS can be found in (Omerdic, *et al.*, 2003). The output of each FDU is mapped to the weights of the weighting matrix \mathbf{W} (9) used in the criteria (8). In a fault-free case all weights are equal to one. A faulty thruster is penalised such that its weight is increased according to predefined rules. Hence, a fault in a single thruster is immediately detected and appropriate reconfiguration is performed by updating the weights in the inversion transformation. In the case of a fault in a single thruster, the feasible region shrinks, as shown in Fig. 7. and 8. In particular, Fig. 7. displays the feasible region for the case of a partial fault in Thruster 2 and its usage of 50%. In this case Thruster 2 is penalised by increasing its weight ($w_2 = 3$) and its corresponding saturation bound in (23) is changed to $|u_2| = 0.5$. This guarantees equality between input and output vectors inside the feasible region and, at the same time, the value of $|u_2|$ will never be greater than 0.5. Similarly, Fig. 8. shows the feasible region for the case of a total breakdown in Thruster 2. In this case Thruster 2 is switched off and $w_2 \rightarrow \infty$. Only three remaining thrusters are capable to track the input vector without any error inside the feasible region, because of the inherent redundancy in the thruster configuration.

6. CONCLUSION

A novel thruster fault detection and accommodation system for ROVs has been presented in the paper. In the case of a partial or total fault in a thruster it is possible to reconfigure the control system in order to keep the high level of the control performance and complete a mission. Knowledge about position of the input vector inside the feasible region and its distance from the saturation bounds can be used to improve existing control laws and to avoid

discrepancy between predicted and real behaviour of the vehicle caused by thruster velocity saturation.

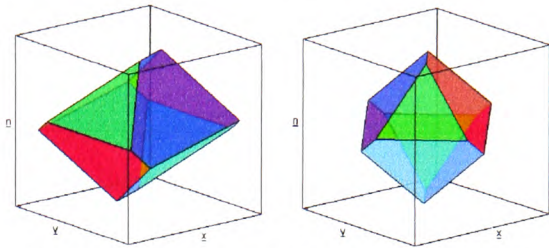


Fig. 7. Feasible region for URIS (left) and FALCON (right) in the case of a partial fault in Thruster 2 and its usage of 50%.

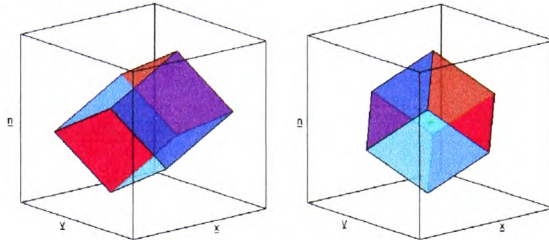


Fig. 8. Feasible region for URIS (left) and FALCON (right) in the case of a total breakdown (failure) in Thruster 2 and its usage of 0%.

REFERENCES

- Farrel, J., T. Berger and B. Appleby. (1993). Using Learning Techniques to Accommodate Unanticipated Faults. *IEEE Control Syst. Mag.*
- Fossen, T.I. 1995. *Guidance and control of ocean vehicles*. John Wiley & Sons, Chichester.
- Fossen, T.I. and M. Blanke (2000). Nonlinear Output Feedback Control of Underwater Vehicle Propellers Using Feedback Form Estimated Axial Flow Velocity. *IEEE Journal of Oceanic Engineering*, 25(2). pp. 241-255.
- Omerdic, E., G.N. Roberts and P. Ridao (2003). Fault detection and accommodation for ROVs. Submitted to 6th IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC 2003), Girona, Spain.
- Rauch, H.E. (1994). Intelligent Fault Diagnosis and Control Reconfiguration. *Plenary Speech from 1994 IEEE International Symposium on Intelligent Control*.
- Sullivan, J.W., P. McCarty, G. Yoshimoto, R. Gargan and R. Pelavin (1992). Intelligent System for Autonomous UUV Monitoring, Diagnosis and Control. *Proceedings 1992 AIAA Guidance, Navigation and Control Conf.*
- Yang, K.C., J. Yuh and S.K. Choi (1998). Experimental Study of Fault-Tolerant System Design for Underwater Robots. *Proceeding of the 1998 IEEE International Conference on Robotics & Automation*, Leuven, Belgium.
- Yang, K.C.H., J. Yuh and S.K. Choi (1999). Fault-tolerant system design of an autonomous underwater vehicle – ODIN: an experimental study. *International Journal of Systems Science*, 30 (9). pp. 1011-1019.

FAULT DETECTION AND ACCOMMODATION FOR ROVS

E. Omerdic¹, G.N. Roberts¹ and P. Ridao²

¹*University of Wales College, Newport
Mechatronics Research Centre, NP20 5XR, UK
email: {edin.omerdic, geoff.roberts}@newport.ac.uk*

²*University of Girona, Spain
Institute of Informatics and Applications, Campus Montilivi, E-17071 Girona
email: pere@eia.udg.es*

Abstract: A new approach for fault detection and accommodation for remotely operated underwater vehicles is proposed in this paper. The proposed Fault Detection and Accommodation System (FDAS) consists of two subsystems: Fault Detection Subsystem (FDS) and Fault Accommodation Subsystem (FAS). The FDS uses fault detector units (FDU), associated with each thruster, to monitor their healthy state. Robust and reliable fault detection units are based on integration of self-organising maps and fuzzy logic clustering methods. These units are able to detect internal and external faulty states of a thruster. The FAS uses information provided by the FDS to accommodate faults and perform an appropriate reconfiguration. A control energy cost function is used as the optimisation criteria. In fault-free and faulty cases the FAS finds the optimal solution, which minimise this criteria. The proposed FDS is evaluated with data obtained during test trials. *Copyright © 2003 IFAC*

Keywords: Fault detection, Fault-tolerant systems, Optimal design, Simulators.

1. INTRODUCTION

This paper introduces new approach associated with fault detection and accommodation for remotely operated underwater vehicles (ROVs). Underwater vehicles are liable to faults or failures during underwater missions. Thrusters are one of the most common and most important sources of faults. In all but the most trivial cases the existence of a fault may lead to cancelling the mission. The implication of small faults could be very expensive and time consuming. Although good design practice tries to minimize the occurrence of faults and failures, there is a certain probability that faults will occur. Recognition that such events do occur enables system designers to develop strategies by which the effect they exert is minimised. A large number of underwater vehicles have four horizontal thrusters for the motion in the horizontal plane in three degrees of freedom (surge, sway and yaw). This paper, together with (Omerdic and Roberts, 2003), demonstrates that, for this class of vehicles, in the case of a partial or total fault in a horizontal thruster, it is possible to reconfigure the control system in an

optimal manner, in order to maintain a high level of manoeuvrability of the faulty vehicle and complete the mission.

In the literature, numerous applications of fault diagnosis are reported for aeronautical and aerospace systems, automotive and traffic systems, chemical processes, electrical and electronic systems, nuclear plants, power systems and transportations systems (Isermann and Balle, 1997). Only recently attention has been devoted to fault diagnosis for ROVs. The integration of fault-tolerant capabilities within the frameworks of the various control architectures for unmanned underwater vehicles is still an open problem. A comprehensive discussion of this topic for general fault-tolerant systems is reported in (Blanke, *et al.*, 2001; Blanke, 2001).

Takai and Ura (1999) proposed a model-based approach for self-diagnosis of an autonomous underwater vehicle (AUV). A key element of the self-diagnosis scheme was a recurrent neural network representation of the dynamics of the AUV. The scheme was implemented on a test bed AUV, and

results showed its ability to cope with sensor and actuator failures. A fault-diagnostic system for unmanned underwater vehicles was designed and tested in real operating conditions by (Alessandri, *et al.*, 1999). They considered total and partial actuator faults. An approximate model of the vehicle was used. Fault detection and diagnosis was accomplished by evaluating any change in the normal behaviour of the system by comparing the state, the parameters and other related quantities of the observed process with those of the normal and faulty processes. On the basis of the healthy and fault models, a bank of estimators was used for the nominal plant, the left and the right actuator fault. Extended Kalman filters were implemented in the process of residual generation for each actuator fault type, including the no-fault case. This scheme showed effective isolation, at the cost of greater computational efforts. Experimental results proved the effectiveness of the proposed approach. A fault-tolerant system for use in an experimental AUV was outlined in (Yang, *et al.*, 1998; Yang, *et al.*, 1999). The system was subdivided into individual fault-tolerant subsystems for dealing with thruster and sensor failures separately. The thruster subsystem consisted of a rule base for detection and isolation purposes, and an algorithm for reconfiguring the thruster control matrix by eliminating the corresponding column to accommodate the failure. Only a total fault (failure) of the thruster was considered. The authors used a constraint-based method instead of the pseudo-inverse method to compute the inverse of the thruster configuration matrix. An experimental investigation was conducted on a 6 DoF AUV, ODIN at the University of Hawaii to evaluate the performance of the proposed approaches and experimental results showed that the overall system was capable of performing effectively. A fault detection, isolation and accommodation system, based on operationally experienced faults in ROV actuators, is proposed in (Bono, *et al.*, 1999). The authors designed a fault management system for underwater vehicles, able to satisfy the basic requirement of handling experienced faults (e.g. flooded thruster) and conventional zero output failures treated in the literature. In addition, the fault management system had to be easily integrated in the hierarchical control architectures. The authors published experience from the sea trials, when the water penetrated inside the thruster and modified the internal electrical connections in such a way that the actual angular speed was higher than the desired one, and current consumption was higher than normal. Fault detection was performed by monitoring the servo-amplifiers residuals, while fault isolation required the vehicle to execute steady-state manoeuvres. Actuator fault accommodation was performed by inhibiting the faulty thruster and by reconfiguring the distribution of the control actions cancelling the corresponding column in the Thruster Control Matrix (TCM). The problem of optimal distribution of propulsion forces for over actuated underwater vehicles is addressed in (Podder, *et al.*, 2000). The authors investigate how to exploit the

excess number of thrusters to accommodate thruster faults. First, a redundancy resolution scheme is presented, which takes into account the presence of excess number of thrusters along with any thruster faults and determines the reference thruster forces to produce the desired motion. In the next step, these reference thruster forces are utilized in the thruster controller to generate the required motion. This approach resolves the thruster redundancy in the Cartesian space and allows the AUV to track the task-space trajectories with asymptotic reduction of the task-space errors. Results from both computer simulations and experiments were provided to demonstrate the viability of the proposed scheme. The paper is a development of the preliminary concept proposed in (Podder and Sarkar, 1999). If the number of control inputs is equal to or more than the number of controllable DoF, it is possible to find an "optimal" distribution of the control energy, which minimises the quadratic energy cost function i.e. a measure of the control effort (Fossen, 1995). This approach uses the generalised inverse matrix to find the optimal control vector. However, the problem of thruster velocity saturation was not considered. The method for process condition monitoring, based on integration of fuzzy inference system and Self-Organising Map (SOM), is proposed in (Cuadrado, *et al.*, 2001). The method identifies regions in the SOM visualisation space, corresponding to different conditions of a monitored process, by means of a fuzzy rule system, which incorporate expert knowledge about process in region identification procedure.

Ideas from (Podder and Sarkar, 1999; Podder, *et al.*, 2000; Yang, *et al.*, 1998; Yang, *et al.*, 1999; Fossen, 1995) were used as the initial basis for the design of the novel FDAS for a ROV, presented herein. A general thruster model is described in the second section. The third section describes the architecture of the proposed FDAS. Details about FDS are given in the fourth section. The performance of the proposed FDS is evaluated with data obtained during test trials and results are given in the fifth section. Finally, the sixth section summarizes the concluding remarks.

2. THRUSTER MODEL

Many modern underwater vehicles have four horizontal thrusters that control three degrees of freedom (surge, sway and yaw). Hence, the control system for motion in the horizontal plane is over actuated and there is redundancy, which provides a space to perform reconfiguration in the case of a fault in a single horizontal thruster. Two underwater vehicles with different thruster configurations are used to demonstrate the performance of the proposed FDAS (FALCON, SeaEye Marine Ltd. and URIS, University of Girona. See Fig. 1.). In order to examine the performance of the proposed FDAS, the original thruster configuration of URIS, with two horizontal and two vertical thrusters, is transformed into a configuration with four horizontal thrusters.

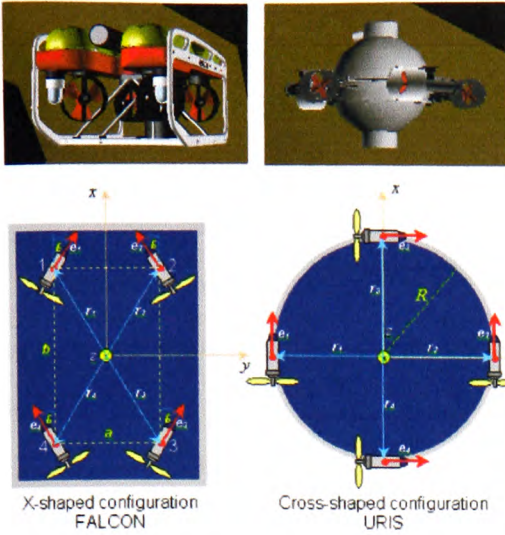


Fig. 1. Two common configurations of the horizontal thrusters: FALCON (left) and URIS (right).

A full thruster model, including dynamics of the thruster control loop, is shown in Fig. 2. A DC motor, designed for underwater operating conditions, drives a thruster. The input to the thruster control loop is a control variable n_d (desired angular velocity). The motor is equipped with a tachometer, which measures actual angular velocity. A gearbox is used to reduce the output speed and to enhance the output torque. Orientation and position of the thruster in the body-fixed frame are defined by vectors \mathbf{e} and \mathbf{r} , respectively. Each thruster exerts thrust \mathbf{T} and torque \mathbf{Q}_e (Fig. 3). Depending on propeller spin direction, vectors \mathbf{T} and \mathbf{Q}_e have the same direction (for clockwise rotation looking from the back) or opposite direction (for counter clockwise rotation). The thrust \mathbf{T} also generates moment $\mathbf{Q}_r = \mathbf{r} \times \mathbf{T}$, so the total moment vector exerted by a thruster is given by $\mathbf{Q} = \mathbf{Q}_e + \mathbf{Q}_r$. Contributions of each thruster are summed together to form force vector $\boldsymbol{\tau}$. In the general case, the thrust T and torque Q , exerted by a thruster, are complicated functions depending on the vehicle's velocity vector \mathbf{V} and control variable n . The bilinear thruster model, shown in Fig. 2. and described in (Fossen, 1995; Fossen and Blanke, 2000), can be used to approximate these functions. However, in practical applications, the bilinear thruster model can be approximated by an affine model, where ROV velocity \mathbf{V} is equal to zero. Hence, optimal distribution of propulsion and control forces will be initially found with the assumption that the relationship between propeller thrust/torque and control variable is given by the affine thruster model. The dynamics of the thruster control loop were initially neglected. The influence of these factors on the performance of the ROV control system will be investigated later in the ROV simulator using the bilinear thruster model with the dynamics of the thruster control loop included.

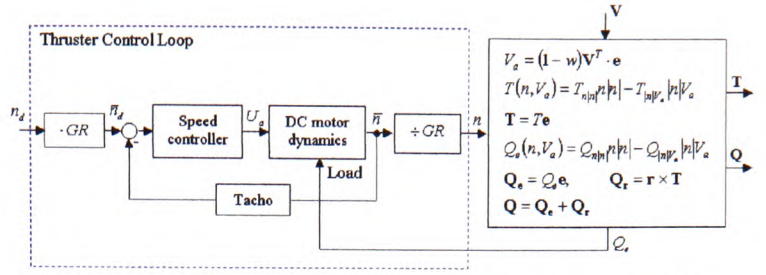


Fig. 2. Simplified diagram showing full thruster model, including thruster control loop dynamics.

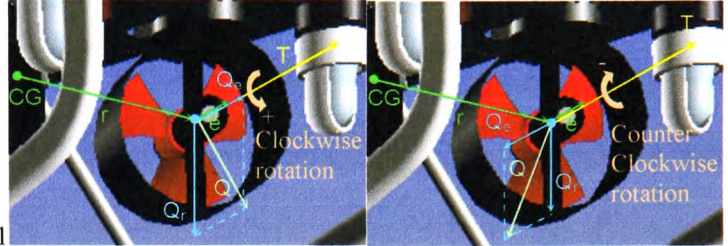


Fig. 3. Thrust and torque, exerted by a thruster, for two possible propeller spin direction: clockwise (left) and counter clockwise (right).

3. THRUSTER FAULT DETECTION AND ACCOMMODATION SYSTEM

The proposed FDAS, shown in Fig. 4., consists of two subsystems: a Fault Detection Subsystem (FDS) and a Fault Accommodation Subsystem (FAS). The FDS uses fault detector units (FDU), associated with each thruster, to monitor their state. The FAS uses information provided by FDS to accommodate faults and perform an appropriate reconfiguration. Full description of the FAS can be found in (Omerdic and Roberts, 2003). In the same paper the concept of a feasible region has been developed in order to cope with the important problem of thruster velocity saturation. The FDUs, which provide reliable and fast information about faults in thrusters, are described in more detail in the next section.

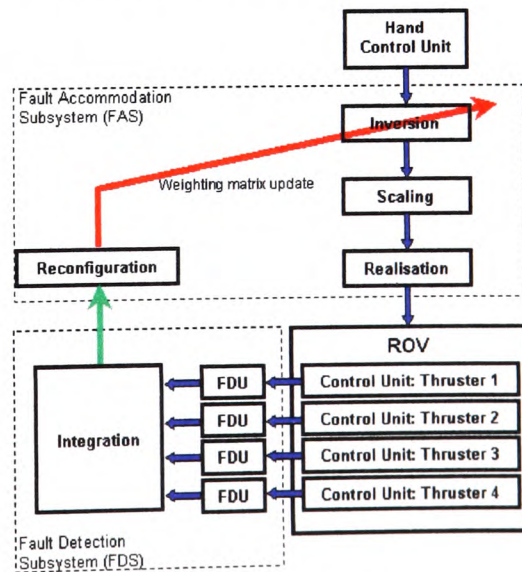


Fig. 4. The proposed FDAS for ROV.

The fault detection and accommodation process involves these steps:

1. FDU detects fault (type and degree of damage in thruster),
2. The corresponding weight is increased i.e. weighting matrix is updated and criteria (control energy cost function) is changed,
3. Inversion and scaling find the new control vector which minimise new criteria,
4. New control vector is denormalised and used to actuate the thrusters.

Hence, a partial or total fault in a single thruster is immediately detected and appropriate reconfiguration is performed by updating the weights in the inversion transformation.

4. FAULT DETECTION SUBSYSTEM

Thrusters are liable to different fault types during the underwater mission e.g. propellers can be jammed or broken, water can penetrate inside the Thruster Control Unit (TCU), communication between TCU and main control unit (MCU) can be lost etc. Some of these faults (partial faults) are not critical and the thruster is able to continue operation in the presence of a fault with reduced maximum velocity (% of usage $0 < p < 100$). In other cases (total faults – failures) thruster must be switched off ($p = 0$) and mission has to be continued with remaining thrusters. Thruster faults are classified into two main classes (Table 1): *internal faults* (e.g. temperature of the windings is out of range, lost communication between TCU and MCU, water penetration inside TCU, drop in bus voltage etc.) and *external faults* (e.g. jammed or broken propeller). Indicator f , the output of the FDU, is the code for the fault. The last column in the table represents desired percentage of usage p of the thruster. Signals for detection of internal faults are already available in existing TCU for both vehicles. In particular, TCU for URIS, based on Maxon Servoamplifier ADS 50/5, has status-reading signal “Ready”, which can be used to report internal fault (excess temperature or excess current). Similarly, communication protocol for FALCON provides temperature of the windings and bus voltage of each thruster. In order to build a universal FDU, capable to detect both internal and external faults, it is necessary to augment existing internal protection with a software module for fast and reliable detection of external faults.

Table 1 Classification of thruster faults

Thruster state	Class	Type	Indicator f	% of usage p
Normal	-	-	1	100
Jammed propeller	Ext.	Partial	2	75
Heavy jammed pr.	Ext.	Partial	3	50
Broken pr.	Ext.	Total	4	0
Unknown fault	Ext.	Partial	5	25
Internal fault	Int.	Total	6	0

The problem of thruster fault detection for underwater vehicle has special features, due to environmental conditions in which the vehicle operates. The most important requirements that the FDU should fulfil are:

- Reliable and fast fault detection, without false alarms,
- Easy integration with the existing control system,
- On-line learning and adaptation to the new types of faults,
- Cost efficient i.e. the FDU should use resources already available, without introducing new hardware,
- Easy transfer to and implementation in other vehicles.

By carefully examination of the available resources in the existing TCUs, the model-free approach, based on integration of Self-organising Map (SOM) and fuzzy clustering techniques, is chosen as the best candidate for FDU to fulfil all these requirements. It is possible to monitor actual speed of the motor shaft n and current consumption I of the thruster for both vehicles (for URIS, these signals are called ‘Monitor n ’ and ‘Monitor I ’, respectively; for FALCON, communication protocol enables to read output speed and winding current). By monitoring n and I , together with desired speed n_d , the FDU should be able to detect external thruster fault. Finally, the universal FDU integrates both parts (internal and external) into one unit, which is able to detect internal and external faults (Fig. 5). Integration is performed using priority scheme, where total faults have higher priority than partial faults.

Implementation of the FDU involves two phases: off-line training and on-line fault detection. First step for the training phase is acquisition of training data. Test trials were performed with URIS at University of Girona in July 2002 (Fig. 6.), and training data were saved in files. Normal state and three different fault cases were considered (jammed, heavy jammed and broken propeller). Jammed propeller was simulated such that an object was attached to the propeller. When the thruster is actuated, propeller and the object rotate together, representing additional load for the motor. Heavy jammed propeller was simulated with two objects attached. In order to simulate broken propeller, blades were removed from the shaft. Each record in file consists of acquired data from the TCU (n_d , n and I) and associated fault code f . Sampling time was 0.1 s, long enough to ensure that all transient responses decay.

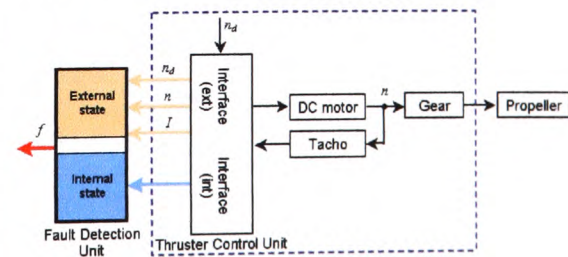


Fig. 5. Connection between FDU and TCU.

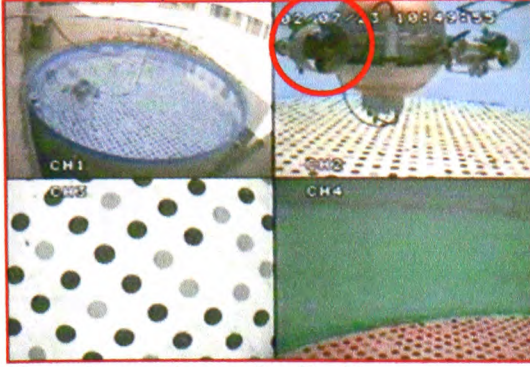
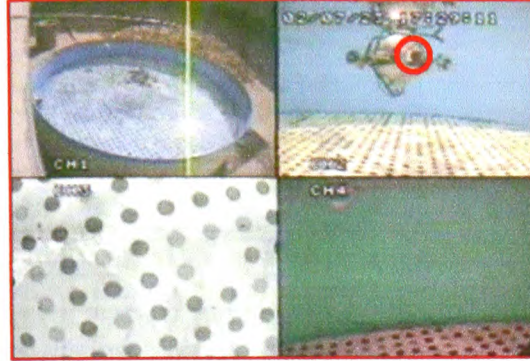
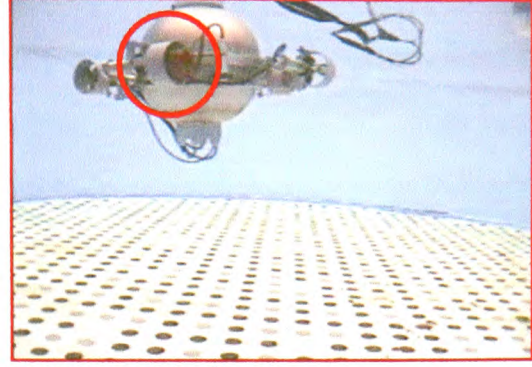
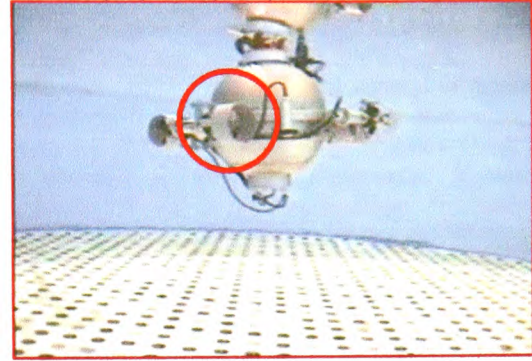
Normal ($f = 1$)Heavy jammed propeller ($f = 3$)Jammed propeller ($f = 2$)Broken propeller ($f = 4$)

Fig. 6. Test trials for acquisition of training data.

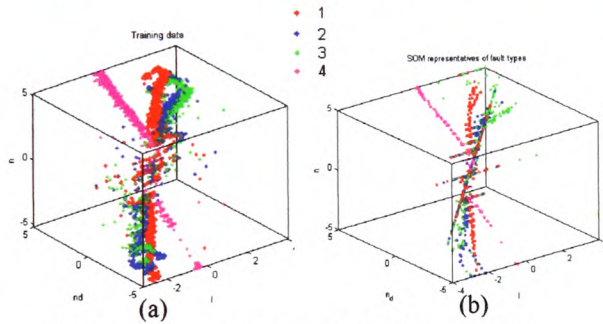


Fig. 7. (a) Training data; (b) SOM representatives.

URIS motion was controlled by joystick, such that all range of possible thruster speeds was covered with enough data points. Fig. 7a. displays the training data in the 3D input space, where each fault type is coded with colour. Because the real-time experiments were undertaken during URIS development stage, poor signal conditioning, wiring and shielding generated the noisy data. Nevertheless, design of robust FDU, able to cope with poor data quality is a real challenge. Preprocessing (filtering and elimination of outliers) reduces presence of the noise. The main idea is to replace each fault type (Fig. 7.a) with SOM (Fig. 7.b), which serves as a representative of the particular fault type. A fault with code $f = i$ is replaced with SOM i . Each SOM i is one-dimensional array of 100 neurons. Each of these neurons has associated codebook (prototype) vector with three coordinates. The distribution of codebook vectors in the input space is found using fuzzy c -means clustering. Each codebook is cluster centre and representative of all data from its cluster. Fig. 7.b shows distribution of codebook vectors.

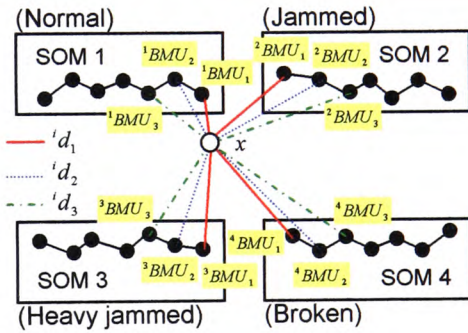


Fig. 8. On-line detection: position of feature vector is determined relative to SOM representatives.

During the on-line fault detection phase, three closest codebook vectors (Best Matching Units – BMUs) from each map to feature vector x (which consists of preprocessed, actual measurements of n_d , n and l) are computed, together with corresponding distances (Fig. 8.), where iBMU_j means j^{th} BMU in SOM i , while id_j means Euclidian distance between x and iBMU_j , $i = \overline{1,4}$, $j = \overline{1,3}$. In the next step matrix $M = [{}^id_j]_{4 \times 3}$ is created. Minimum values of each column of M are found and the indices of the minimum values are stored in row vector b . For example, $b = [1 \ 3 \ 2]$ means that the closest first BMU is in SOM 1, second – in SOM 3 and third – in SOM 2. In order to avoid false detection, final decision about faults is done using present and past vectors b , which are stored in the buffer with size $s \times 3$. If all buffer elements have the same value, then the fault indicator f , output of the FDU, is set to this value.

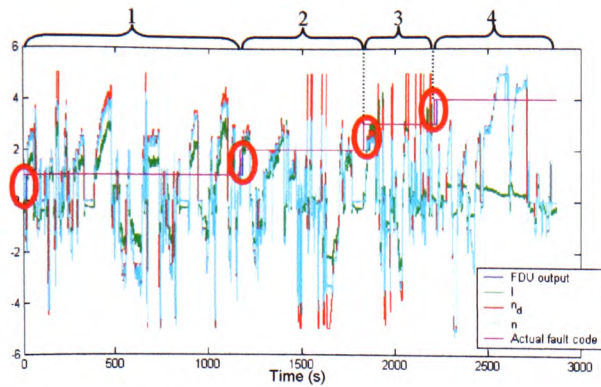


Fig. 9. Evaluation of the proposed FDS.

5. FDS EVALUATION

A large data set was acquired during test trials and only a part of this data was used for training. Evaluation of the proposed FDS is performed using entire data set. Only signals n_d , n and I are presented as inputs to the FDS, which must estimate the faulty state of the thruster using only these inputs. Fig. 9. displays actual fault code and FDU output, together with actual data. It can be seen that the FDU, identifies the new faulty state correctly in a short time after the change in faulty state. This delay is unavoidable, because the thruster must spend some time in faulty state before it can be identified. Delay is proportional to the buffer size s . A conservative value $s = 25$ was used in Fig. 9., in order to prevent a wrong detection. It is expected that the buffer size and delay will be reduced in the case of FALCON, due to advanced signal conditioning and better quality of measured signals.

5. CONCLUDING REMARKS

This paper is focused on FDS, integral part of a novel FDAS for underwater vehicles. The proposed FDS is hybrid, on-line, model-free approach, based on integration of SOM and fuzzy clustering methods. In training phase FDS uses data obtained during test trial to find SOM representatives for each fault type. In detection phase FDS makes decision by comparing position of feature vector relative to these maps. Evaluation results demonstrate efficiency and robustness of the FDS. Future work will include implementation of the proposed approach and its integration into existing control architecture for both vehicles. Special attention will be devoted to the design of universal controller, robust to a partial/total fault in a single thruster. An important part of the future work is the integration of feasible region with real-time picture presented to the ROV pilot from the on-board camera.

REFERENCES

- Alessandri, A., M. Caccia and G. Veruggio (1999). Fault detection of actuator faults in unmanned underwater vehicles. *Control Engineering Practice*, 7, pp. 357-368.
- Blanke, M. (2001). Enhanced maritime safety through diagnosis and fault tolerant control. *IFAC Conference on Control Applications and Marine Systems, CAMS 2001*, Glasgow. (Invited plenary)
- Blanke, M., M. Staroswiecki and N. Eva Wu (2001). Concepts and methods in fault-tolerant control. *Proc. American Control Conference*, Washington. (Invited tutorial)
- Bono, R., Ga. Bruzzone, Gi. Bruzzone and M. Caccia (1999). ROV actuator fault diagnosis through servo-amplifiers' monitoring: an operational experience. *MTS/IEEE Oceans'99*, vol. 3, pp. 1318-1324, Seattle, USA.
- Cuadrado, A.A., I. Díaz, A.B. Díez, F. Obeso and J.A. González (2001). Fuzzy Inference Maps for Condition Monitoring with Self-Organising Maps. *International Conference in Fuzzy Logic and Technology*, pp. 55-58, Leicester, UK.
- Fossen, T.I. 1995. *Guidance and control of ocean vehicles*. John Wiley & Sons, Chichester.
- Fossen, T.I. and M. Blanke (2000). Nonlinear Output Feedback Control of Underwater Vehicle Propellers Using Feedback Form Estimated Axial Flow Velocity. *IEEE Journal of Oceanic Engineering*, 25(2), pp. 241-255.
- Isermann, R. and P. Ballé (1997). Trends in the application of model-based fault detection and diagnosis of technical process. *Control Engineering Practice*, 5 (5), pp. 709-719.
- Omerdic, E. and G.N. Roberts (2003). Thruster fault accommodation for underwater vehicles. *1st IFAC Workshop on Guidance and Control of Underwater Vehicles GCUV'03*. pp. 221-226. 9th -11th April 2003, Newport, South Wales, UK.
- Podder, T.K. and N. Sarkar (1999). Fault Tolerant Decomposition of Thruster Forces of an Autonomous Underwater Vehicle. *Proceeding of the 1999 IEEE International Conference on Robotics & Automation*, Detroit, USA.
- Podder, T.K., G. Antonelli and N. Sarkar (2000). Fault Tolerant Control of an Autonomous Underwater Vehicle Under Thruster Redundancy: Simulations and Experiments. *Proceeding of the 2000 IEEE International Conference on Robotics & Automation*, San Francisco, USA.
- Takai, M. and T. Ura (1999). Development of a system to diagnose autonomous underwater vehicle. *International Journal of Systems Science*, 30 (9), pp. 981-988.
- Yang, K.C., J. Yuh and S.K. Choi (1998). Experimental Study of Fault-Tolerant System Design for Underwater Robots. *Proceeding of the 1998 IEEE International Conference on Robotics & Automation*, Leuven, Belgium.
- Yang, K.C.H., J. Yuh and S.K. Choi (1999). Fault-tolerant system design of an autonomous underwater vehicle - ODIN: an experimental study. *International Journal of Systems Science*, 30 (9), pp. 1011-1019.



Thruster fault diagnosis and accommodation for open-frame underwater vehicles

Edin Omerdic*, Geoff Roberts

Mechatronics Research Centre, University of Wales College, Newport Allt-yr-yn Campus, P.O. Box 180, Newport NP20 5XR, UK

Received 1 December 2003; accepted 20 December 2003

Abstract

This paper introduces a novel thruster fault diagnosis and accommodation system (FDAS) for open-frame underwater vehicles. Basically, the FDAS is a control allocator, but this primary function is enhanced with the ability of automatic thruster fault detection and accommodation. The proposed FDAS consists of two subsystems: a fault diagnosis subsystem (FDS) and a fault accommodation subsystem (FAS). The FDS uses fault detector units (FDUs), associated with each thruster, to monitor their state. Robust and reliable FDUs are based on integration of self-organising maps and fuzzy logic clustering methods. These units are able to detect internal and external faulty states of thrusters. The FAS uses information provided by the FDS to accommodate faults and perform an appropriate control reallocation. A control energy cost function is used as the optimisation criteria. The FAS uses weighted pseudo-inverse to find the solution of the control allocation problem, which minimise this criteria. Two approximations (truncation or scaling) can be used to ensure feasibility of the solution. The proposed FDS is evaluated with data obtained during test trials. The feasible region concept, related with the problem of thruster velocity saturation, is developed in order to provide geometrical interpretation of the control allocation problem. The proposed FDAS is implemented as a Simulink model (ROV simulator), in order to evaluate its performance in different faulty situations.

© 2003 Published by Elsevier Ltd.

Keywords: Fault detection; Fault-tolerant systems; Feasible region; Optimal design; Simulators; Underwater vehicles

1. Introduction

A large number of open-frame underwater vehicles have no other actuators except thrusters. This paper introduces new approach associated with thruster fault diagnosis and accommodation for this class of underwater vehicles. The work presented herein is applicable to wide class of open-frame underwater vehicles. However, the application is focused on two remotely operated vehicles (ROVs) with different thruster configuration. The paper expands upon the work previously reported by the authors (Omerdic & Roberts 2003; Omerdic, Roberts, & Ridao, 2003).

Underwater vehicles are liable to faults or failures during underwater missions. Thrusters are one of the most common and most important sources of faults. In

all but the most trivial cases the existence of a fault may lead to cancelling the mission. The implication of small faults could be very expensive and time consuming. Although good design practice tries to minimize the occurrence of faults and failures, there is a certain probability that faults will occur. Recognition that such events do occur enables system designers to develop strategies by which the effect they exert is minimised. A large number of underwater vehicles have more horizontal thrusters for the motion in the horizontal plane than controllable DOF. This paper demonstrates that, for this class of vehicle, in the case of a partial or total fault in a horizontal thruster, it is possible to reconfigure the control system in an optimal manner, in order to maintain a high level of manoeuvrability of the faulty vehicle and complete the mission.

In the literature numerous applications of fault diagnosis are reported for aeronautical and aerospace systems, automotive and traffic systems, chemical processes, electrical and electronic systems, nuclear plants, power systems and transportation systems (for

*Corresponding author. Tel.: +44-78-132-40-419; fax: +44-1633-432-442.

E-mail addresses: edin.omerdic@newport.ac.uk (E. Omerdic), geoff.roberts@newport.ac.uk (G. Roberts).

example, see Isermann & Ballé, 1997). Only recently attention has been devoted to fault diagnosis for unmanned underwater vehicles. The integration of fault-tolerant capabilities within the frameworks of the various control architectures for unmanned underwater vehicles is still an open problem. A comprehensive discussion of this topic for general fault-tolerant systems is reported in Blanke, Staroswiecki, and Eva Wu (2001), Blanke (2001).

Takai and Ura (1999) proposed a model-based approach for self-diagnosis of an autonomous underwater vehicle (AUV). A key element of the self-diagnosis scheme was a recurrent neural network representation of the dynamics of the AUV. The scheme was implemented on a test bed AUV, and results showed its ability to cope with sensor and actuator failures.

A fault-diagnostic system for unmanned underwater vehicles was designed and tested in real operating conditions by Alessandri, Caccia, and Veruggio (1999). They considered total and partial actuator faults. An approximate model of the vehicle was used. Fault detection and diagnosis was accomplished by evaluating any change in the normal behaviour of the system by comparing the state, the parameters and other related quantities of the observed process with those of the normal and faulty processes. On the basis of the healthy and faulty models, a bank of estimators was used for the nominal model, the left and right actuator faults. Extended Kalman filters were implemented in the process of residual generation for each actuator fault type, including the no-fault case. This scheme showed effective isolation, at the cost of greater computational efforts.

A fault-tolerant system for use in an experimental AUV was outlined in Yang, Yuh, and Choi (1999, 1998). The system was subdivided into individual fault-tolerant subsystems for dealing with thruster and sensor failures separately. The thruster subsystem consisted of a rule base for detection and isolation purposes, and an algorithm for reconfiguring the thruster control matrix by eliminating the corresponding column to accommodate the failure. Only a total fault (failure) of the thruster was considered. The authors used a constraint-based method instead of the pseudo-inverse method to compute the inverse of the thruster configuration matrix. An experimental investigation was conducted on a 6 DOF AUV, ODIN at the University of Hawaii to evaluate the performance of the proposed approaches and experimental results showed that the overall system was capable of performing effectively.

A fault detection, isolation and accommodation system, based on operationally experienced faults in ROV actuators, is proposed in Bono, Bruzzone, Bruzzone, and Caccia (1999). The authors designed a fault management system for underwater vehicles, able to satisfy the basic requirement of handling experienced

faults (e.g. flooded thruster) and conventional zero output failures treated in the literature. In addition, the fault management system had to be easily integrated in the hierarchical control architectures. The authors published experience from the sea trials, when the water penetrated the thruster and modified the internal electrical connections in such a way that the actual angular speed was higher than the desired one, and current consumption was higher than normal. Fault detection was performed by monitoring the servo-amplifiers residuals, while fault isolation required the vehicle to execute steady-state manoeuvres. Actuator fault accommodation was performed by inhibiting the faulty thruster and by reconfiguring the distribution of the control actions cancelling the corresponding column in the thruster control matrix.

The problem of optimal distribution of propulsion forces for over actuated underwater vehicles is addressed in Podder, Antonelli, and Sarkar (2000). The authors investigate how to exploit the excess number of thrusters to accommodate thruster faults. First, a redundancy resolution scheme is presented, which takes into account the presence of excess number of thrusters along with any thruster faults and determines the reference thruster forces to produce the desired motion. In the next step, these reference thruster forces are utilized in the thruster controller to generate the required motion. This approach resolves the thruster redundancy in the Cartesian space and allows the AUV to track the task-space trajectories with asymptotic reduction of the task-space errors. The results from both computer simulations and experiments were provided to demonstrate the viability of the proposed scheme. The paper is a development of the preliminary concept proposed in Podder and Sarkar (1999).

If the number of control inputs is equal to or more than the number of controllable DOF, it is possible to find an "optimal" distribution of the control energy, which minimises the quadratic energy cost function i.e. a measure of the control effort (Fossen, 1995). This approach uses the generalised inverse matrix to find the optimal control vector. However, the problem of thruster velocity saturation was not considered.

The method for process condition monitoring, based on the integration of a fuzzy inference system and a self-organising map (SOM), is proposed in Cuadrado, Díaz, Díez, Obeso, and González (2001). The method identifies regions in the SOM visualisation space, corresponding to different conditions of a monitored process, by means of a fuzzy rule system, which incorporate expert knowledge about process in the region identification procedure.

Significant efforts have been undertaken in research community over last two decades to solve the control allocation problem for modern aircraft. Different methods were proposed such as direct control allocation

(Durham, 1994, 1993), optimisation-based methods using l_2 norm (Enns, 1998; Virnig & Bodden, 1994; Snell, Enns, & Garrard, 1992) and l_1 norm (Ikeda & Hood, 2000; Enns, 1998; Lindfors, 1993), fixed-point method (Burken, Lu, & Wu, 1999; Burken, Lu, Wu, & Bahm, 2001) and daisy chain control allocation (Bordignon, 1996). However, the problem of fault accommodation for underwater vehicles is closely related with the control allocation problem for aircrafts. In both cases, the control allocation problem can be defined as the determination of the actuator control values that generate a given set of desired or commanded forces and moments.

For the unconstrained control allocation problem with a control energy cost function used as optimisation criteria the optimal solution is weighted pseudo-inverse (Fossen, 1995). Pseudo-inverse is a special case of general inverse (GI). GI solutions have the advantage of being relatively simple to compute and allowing some control in distribution of control energy among available actuators. However, in real applications actuator constraints must be taken into account, which leads to a constrained control allocation problem. Handling of constrained controls is the most difficult problem for GI approach. In some cases, the solution obtained by the generalised inverse approach is not feasible, i.e. it violates actuator constraints. Durham (1993) demonstrated that, except in certain degenerate cases, a general inverse cannot allocate controls inside a constrained control subset Ω that will map to all attainable command set Φ , i.e. only a subset of Φ can be covered. Two methods are suggested to handle cases where attainable control inputs cannot be allocated. The first approach (truncation) calculates a GI solution and truncates any controls (components of control vector) which exceed their limits. The second approach (scaling) maintains the direction of the desired control input command by scaling unfeasible pseudo-inverse solution to the boundary of Ω (Bordignon, 1996). Even if the controls do not saturate, care must be taken in choosing the GI. When weighted pseudo-inverse solutions are used for problems where the actuator settings are measured in different physical dimensions, the elements of the weighting matrix must be chosen carefully if the resulting solution is desired to be invariant to changes in units and coordinate systems (Doty, Melchiorri, & Bonivento, 1993). The FDAS overcomes this problem by performing normalisation, such that all physical parameters are removed from the thruster control matrix and included in limit constraints, which are used during normalisation process to scale individual components of vectors on standard interval $[-1, 1]$.

Recently, some authors reformulated the constrained control allocation problem as a quadratic programming (QP) problem. QP generally refers to the numerical solution of the optimisation problems with an l_2 norm.

An explicit solution approach is developed by Tøndel, Johansen, and Bemporad (2001). An on-line algorithm is presented in Tøndel, Johansen, and Bemporad (2003), while the application to marine vessels is given in Johansen, Fossen, and Tøndel (2002). An alternative to the explicit solution is to use an iterative solution to solve the QP problem. The drawback with the iterative solution is that several iterations may have to be performed at each sample in order to find the optimal solution. An advantage of the iterative approach is that there is more flexibility for on-line reconfiguration. Computational complexity is also greatly reduced by a “warm start”, i.e. the numerical solver is initialised with the solution of the optimisation problem from the previous sample (Fossen, 2002).

Ideas from Podder et al. (2000), Yang et al. (1999), Yang et al. (1998), Bordignon (1996), Fossen (1995) were used as the initial basis for the design of the FDAS, presented herein. Two common thruster configurations are described in the second section. The third section describes a standard control architecture for ROVs (without the FDAS) and an improved architecture (with the FDAS). A general thruster model is described in the fourth section. The control allocation problem is analysed in the fifth section, which begins with the general problem formulation, followed by the decomposition of motion. A normalisation procedure is then described, followed by the problem formulation for horizontal thrusters and an analysis of the choice of weighting matrix. Finally, the section ends with the remarks about a vertical thruster. The sixth section describes the architecture of the proposed FDAS. This section begins with the short description of the overall architecture, followed by a detail description of the FDS and the FAS structure. The potential application of the FDAS is addressed in the seventh section. Finally, the eighth section summarizes the concluding remarks and contributions of the paper.

2. Thruster configuration

For underwater vehicles the most common actuators are (Fossen, 2002):

- *Azimuth thrusters*: Thruster units that can be rotated an angle α about the z-axis during the mission and produce two force components (F_x, F_y) in the horizontal plane. They are attractive in dynamic positioning systems, since they can produce forces in different directions, leading to an overactuated control problem that can be optimised with respect to power and possible faulty situations.
- *Fixed direction (non-rotatable) thrusters*: In contrast to azimuth thrusters, where an angle α can vary with time, fixed direction thrusters are characterised with a

fixed angle $\alpha = \alpha_0$, i.e. orientation of these thrusters is fixed in advance and cannot be changed during the mission.

- **Control surfaces:** Control surfaces can be mounted at different locations to produce lift and drag forces, like fins for diving, rolling and pitching, rudders for steering, etc.

Two underwater vehicles (FALCON, SeaEye Marine Ltd. and URIS, University of Girona, see Fig. 1) with different thruster configurations are used to demonstrate the performance of the FDAS. These ROVs have no other actuators except fixed direction thrusters, and the following discussion will be concentrated on this type of actuators, while more information about other types can be found in Fossen (2002).

The main assumption for successful control allocation in the case of a fault in a single thruster is that the control system for motion in the horizontal plane is overactuated. From this reason the original thruster configuration of URIS, with two horizontal and two vertical thrusters, is transformed into a configuration with four horizontal thrusters, without any vertical thruster, as shown in Fig. 1b. This modification was possible, because the tank for test trials and acquiring of training data at University of Girona was very shallow and there was no space and need for motion in vertical plane. Three-dimensional view of the FALCON and URIS are shown, respectively, in Fig. 1a and b. Plan views of the vehicles, with corresponding configuration

of the horizontal thrusters, are shown in bottom part of Fig. 1. The origin of the body-fixed reference frame $\{B\}$ is chosen to coincide with the centre of gravity. The axes are chosen to coincide with the principal axes of inertia and they are defined as:

- x_B —longitudinal axes (directed to front side),
- y_B —transversal axes (directed to starboard),
- z_B —normal axes (directed from top to bottom).

The FALCON has four horizontal thrusters, denoted as iHT , $i = 1, 4$ and one vertical 1VT (not shown in Fig. 1a—bottom). The URIS has only four horizontal thrusters, denoted in the same way. Thruster configuration of the FALCON enables direct control of 4 DOF: surge, sway, yaw and heave, as indicated in Fig. 2a. In a similar way, modified thruster configuration of the URIS allows direct control of only 3 DOF: surge, sway and yaw (Fig. 2b).

Vector τ_d (Fig. 3) can be decomposed into two parts as $\tau_d = [\tau_d^{HT} \ \tau_d^{VT}]^T$, where τ_d^{HT} represents desired surge force τ_x , sway force τ_y and yaw moment τ_N for motion in the horizontal plane, and τ_d^{VT} is equal to heave force τ_z for motion in the vertical plane. The control allocation problem for motion of the FALCON in the vertical plane is straightforward, since the vector τ_d^{VT} is a scalar, i.e. there is one-to-one correspondence between the controllable DOF (heave) and the vertical thruster. However, in the general case the vector τ_d^{VT} can have three components (heave force τ_z , roll moment τ_K and

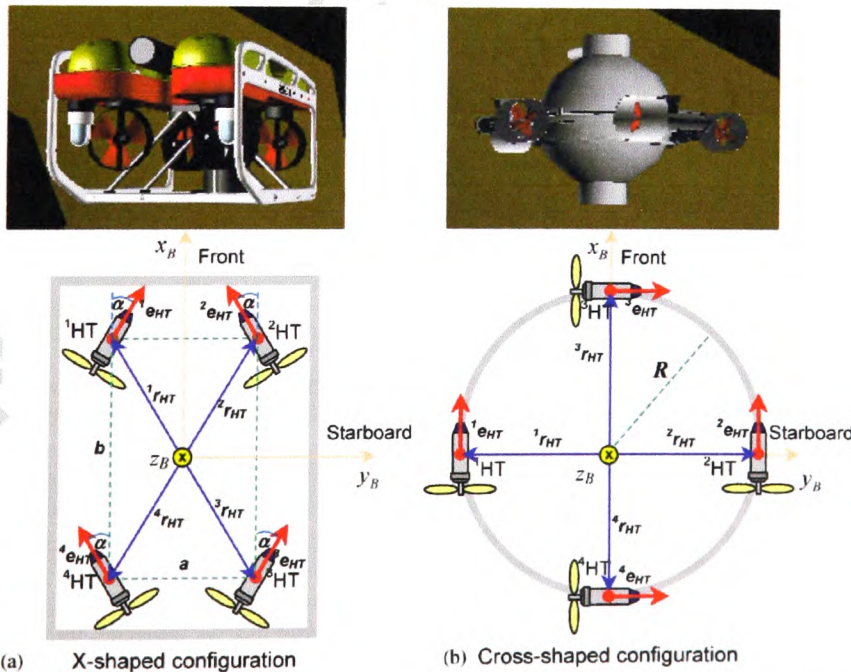


Fig. 1. Two common configurations of the horizontal thrusters: (a) FALCON. (b) URIS.

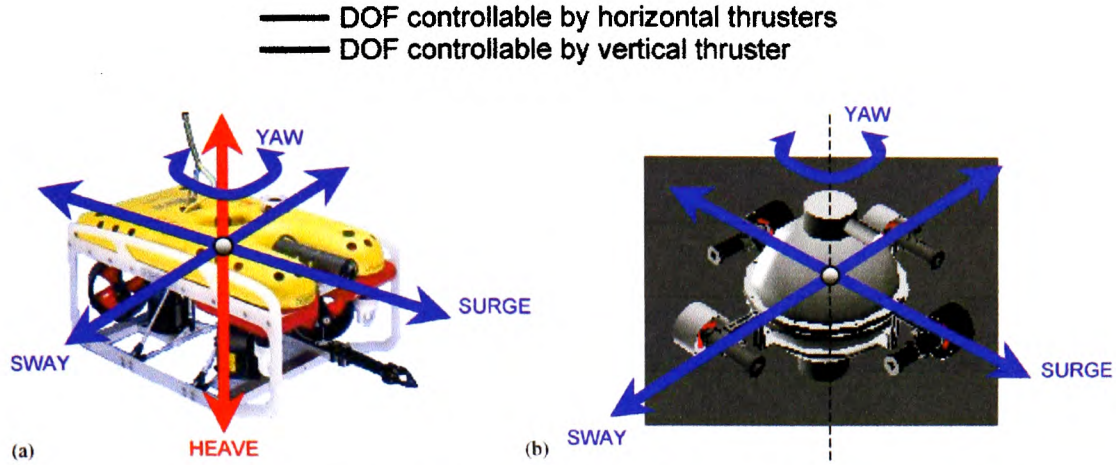


Fig. 2. Relationship between thruster configuration and controllable DOF: (a) FALCON. (b) URIS.

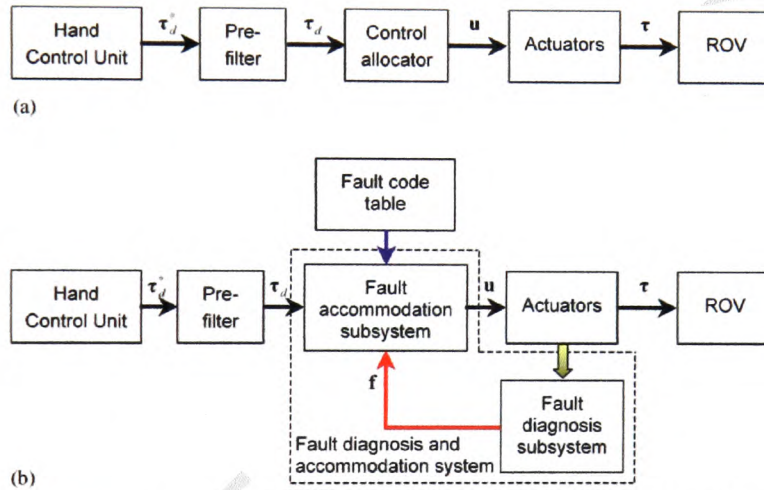


Fig. 3. Open-loop ROV control structure: (a) Standard architecture, without FDAS. (b) Improved architecture, with FDAS.

pitch moment τ_M). Typical example is ODIN (Yang et al., 1999, 1998), with four horizontal and four vertical thrusters, where each of vectors τ_d^{HT} and τ_d^{VT} have three components.

3. Control architecture

A standard open-loop ROV control structure is shown in Fig. 3a. The ROV pilot uses the Hand Control Unit (HCU) to generate vector τ_d^* , which can be interpreted as a desired vector of propulsion forces and moments among axes in the body-fixed frame. Raw signals from the HCU, packed in vector τ_d^* , pass through the low-pass pre-filter to smooth out the commanded input and to protect the actuators from damage caused by abrupt changes of set points. The output of the pre-filter is the desired vector of propulsion forces and moments (virtual control input)

τ_d . The control allocator maps the vector τ_d into the vector (true control input) u , representing control settings for individual actuators. After actuation with u , the actuators generate a vector of propulsion forces and moments (total control effect) τ , which is applied as the input to the ROV dynamics block, and determine the behaviour of the vehicle. The main objective of the control allocation is to ensure that the condition $\tau = \tau_d$ is satisfied for all attainable τ_d .

The control allocator in the standard structure shown in Fig. 3a is replaced by the fault accommodation subsystem (FAS) in the improved control structure, shown in Fig. 3b. The FAS performs weighted pseudo-inverse method for control allocation. The primary task of control allocation is enhanced with the fault diagnosis subsystem (FDS), able to monitor state of the thrusters and inform the FAS about any malfunctions using the total fault indicator vector f , carrying the codes of faulty states for each thruster. The FAS uses information

Table 1
Position vectors for different thruster configurations

	Horizontal thrusters				Vertical thruster
	${}^1r_{HT}$	${}^2r_{HT}$	${}^3r_{HT}$	${}^4r_{HT}$	${}^1r_{VT}$
X-shaped configuration (FALCON)	$\begin{bmatrix} b/2 \\ -a/2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} b/2 \\ a/2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -b/2 \\ a/2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -b/2 \\ -a/2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ l_z \end{bmatrix}$
Cross-shaped configuration (URIS)	$\begin{bmatrix} 0 \\ -R \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ R \\ 0 \end{bmatrix}$	$\begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -R \\ 0 \\ 0 \end{bmatrix}$	—

Table 2
Orientation vectors for different thruster configurations

	Horizontal thrusters				Vertical thruster
	${}^1e_{HT}$	${}^2e_{HT}$	${}^3e_{HT}$	${}^4e_{HT}$	${}^1e_{VT}$
X-shaped configuration (FALCON)	$\begin{bmatrix} \cos \alpha \\ \sin \alpha \\ 0 \end{bmatrix}$	$\begin{bmatrix} \cos \alpha \\ -\sin \alpha \\ 0 \end{bmatrix}$	$\begin{bmatrix} \cos \alpha \\ \sin \alpha \\ 0 \end{bmatrix}$	$\begin{bmatrix} \cos \alpha \\ -\sin \alpha \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
Cross-shaped configuration (URIS)	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$	—

provided by the FDS to accommodate faults by performing an appropriate reconfiguration, i.e. to reallocate control energy among operable thrusters. The overall fault diagnosis and accommodation process is very fast, despite the fact that in some cases it is necessary to perform iterations, due to the computational efficiency of the FDAS algorithm, where the heaviest numerical calculations are performed off-line, in advance.

4. Thruster model

In the general case an ROV has p thrusters ${}^1Th, {}^2Th, \dots, {}^pTh$. Each thruster iTh , $i = \overline{1, p}$ exerts thrust (force) iT and torque (moment) iQ_e (Fig. 4).¹ The position vector ${}^ir = [{}^ir_x \ {}^ir_y \ {}^ir_z]^T$ determines the position of the point of attack of the force iT , relative to the $\{B\}$. The force iT also generates the moment ${}^iQ_r = {}^ir \times {}^iT$, so that the total moment vector exerted by the thruster is given by ${}^iQ = {}^iQ_e + {}^iQ_r$. The orientation of the thruster iTh relative to the $\{B\}$ is defined by the unit vector ${}^ie = [{}^ie_x \ {}^ie_y \ {}^ie_z]^T$. The vector ie shows the positive direction of the force iT . This means that, if the propeller angular velocity is positive, it will exert the force iT in the direction of ie . Otherwise, the force iT is opposite to ie . Depending on propellers spin direction, vectors iT and iQ_e have the same direction (for clockwise rotation looking from the back of the propellers) or the opposite direction (for counterclock-

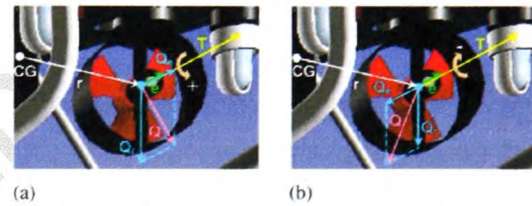


Fig. 4. Thrust and torque, exerted by a thruster, for two possible propellers spin directions: (a) Clockwise. (b) Counter clockwise.

wise rotation). An elegant way to describe this relationship is to introduce a *spin direction coefficient* s : the value $s = +1(-1)$ means that the force vector iT and the torque vector iQ_e have the same (opposite) direction. Assume that propeller angular velocity is positive and the propellers spin direction is clockwise (looking from the back of the propellers, see Fig. 4a). The direction of the torque vector iQ_e is determined using the right-hand rule. In this case, the force vector iT and the torque vector iQ_e have the same direction as the vector ie and $s = +1$. If the propeller spin direction is counter clockwise (Fig. 4b), then vectors iT and ie have the same direction, while vectors iQ_e and ie have the opposite direction and $s = -1$.

Position vectors for different configurations are given in Table 1, while Table 2 shows orientation vectors. Parameters a , b , α and R can be obtained from the technical specifications of the vehicles.

A simplified block diagram of full thruster model, including the dynamics of the thruster control loop, is shown in Fig. 5. A DC motor drives a thruster. The input to the thruster control loop is a control variable n_d

¹ Fig. 4 displays front side of propellers.

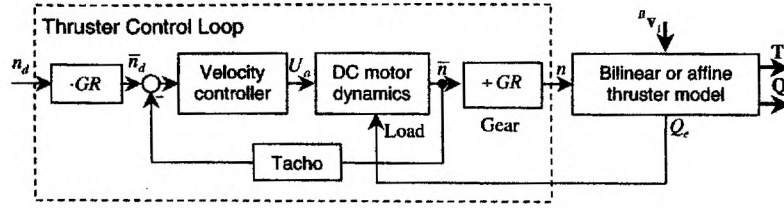


Fig. 5. Simplified diagram showing full thruster model, including thruster control loop dynamics.

(desired angular velocity). The motor is equipped with a tachometer, which measures actual angular velocity \bar{n} . A gearbox with the gear ratio $GR > 1$ is used to reduce the output angular velocity \bar{n} ($n = (1/GR)\bar{n}$) and to enhance the output torque. This means that the input n_d must be multiplied by GR ($\bar{n}_d = GRn_d$). A typical thruster control loop is implemented as an independent device called *thruster control unit* (TCU) with integrated power amplifiers and controlled by a microcontroller. The velocity controller is usually implemented as a digital PID controller, although the other designs are possible.

In the general case, the thrust T and torque Q_e exerted by a thruster, are complicated functions depending on the vehicle's velocity vector $\mathbf{v}_1 = [u \ v \ w]^T$ and control variable n . The bilinear thruster model, described in (Fossen, 1995; Fossen & Blanke, 2000), can be used to approximate these functions. In the general case, the thrust T and torque Q_e can be calculated from

$$\begin{aligned} T(n, V_a) &= \rho D^4 K_T(J_0)n|n|, & \mathbf{T} &= T\mathbf{e}, \\ Q_e(n, V_a) &= \rho D^5 K_Q(J_0)n|n|, & \mathbf{Q}_e &= sQ_e\mathbf{e}, \end{aligned} \quad (1)$$

where ρ is density of water, D is propeller diameter, K_T and K_Q are non-dimensional thrust and torque coefficients, n is propeller shaft speed, s is spin direction coefficient and J_0 is advance ratio, defined as

$$J_0 = \frac{u_a}{nD}, \quad (2)$$

where u_a is ambient water velocity. Relationships between coefficients K_T , K_Q and J_0 are given by

$$\begin{aligned} K_T(J_0) &= \alpha_2 - \alpha_1 J_0, \\ K_Q(J_0) &= \beta_2 - \beta_1 J_0, \end{aligned} \quad (3)$$

where α_i and β_i are four positive non-dimensional constants. Eqs. (3) are obtained by linear approximation of the experimental curves $K_T(J_0)$ and $K_Q(J_0)$, obtained in open water test in a cavitation tunnel or a towing tank (Fossen & Blanke, 2000). Eqs. (1) and (3) imply that the mathematical expressions for the thrust T and torque Q_e can be written as the bilinear thruster model:

$$\begin{aligned} T(n, u_a) &= T_{n|n}|n| - T_{n|u_a}|n|u_a, & \mathbf{T} &= T\mathbf{e}, \\ Q_e(n, u_a) &= Q_{n|n}|n| - Q_{n|u_a}|n|u_a, & \mathbf{Q}_e &= sQ_e\mathbf{e}, \end{aligned} \quad (4)$$

where

$$\begin{aligned} T_{n|n} &= \rho D^4 \alpha_2, & T_{n|V_a} &= \rho D^3 \alpha_1, \\ Q_{n|n} &= \rho D^5 \beta_2, & Q_{n|V_a} &= \rho D^4 \beta_1. \end{aligned} \quad (5)$$

The relationship between ambient water velocity u_a around the thruster with the orientation defined with the vector \mathbf{e} and the vehicle's linear velocity \mathbf{v}_1 is given by

$$u_a = (1 - \omega)\mathbf{v}_1^T \cdot \mathbf{e}, \quad (6)$$

where $\omega > 0$ (typically 0.1–0.4) is the wake fraction number. It can be seen from (6) that u_a can be found as projection of the vector \mathbf{v}_1 on the vector \mathbf{e} , scaled by factor $(1 - \omega)$.

However, in practical applications, the bilinear thruster model (4) can be approximated by an *affine model* (7), where it is assumed that $u_a = 0$:

$$\begin{aligned} T(n, V_a) &= T_{n|n}|n|, & \mathbf{T} &= T\mathbf{e}, \\ Q_e(n, V_a) &= Q_{n|n}|n|, & \mathbf{Q}_e &= sQ_e\mathbf{e}. \end{aligned} \quad (7)$$

Initially, the control allocation problem will be formulated and solved under the following assumptions:

1. the dynamics of thruster control loop is neglected² (Fig. 5),
2. the relationship between propeller thrust/torque and the control variable is given by modified version of affine thruster model (7), described below.

The first assumption is realistic, since the time constants of DC motors used to drive thrusters of the FALCON and the URIS are very small and sample time, dictated by control software, is about 40 ms, long enough to ensure that all transient responses between two samples in thruster control loop disappear.

The second assumption means that:

- shaft torque Q_e is neglected, since it is small compared to Q_r , $\mathbf{Q}_r = \mathbf{r} \times \mathbf{T}$,
- the effect of the ambient water velocity u_a on propeller thrust T is neglected,

²One of the most important difficulties for successful control allocation is actuator dynamics. Fortunately, in many cases actuator dynamics is much faster than the dynamics of the other parts of the system. In these cases, the most common solution is that actuator dynamics are simply neglected. This will work as long as the closed-loop system is designed to be substantially slower than the actuator servo systems.

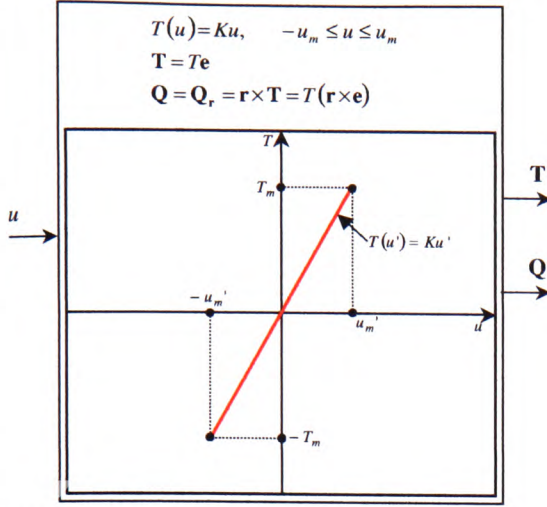


Fig. 6. Thruster model used in control allocation for underwater vehicles.

- the symmetrical relationship between thrust T and the control variable $u = n|n|$ is used (see Fig. 6).

Under these assumptions, the control allocation problem for open-frame underwater vehicles (in particular, FALCON and URIS) is linear and can be solved using techniques described in the following. The influence of the neglected factors on the performance of the ROV control system can be investigated using an ROV simulator incorporating the bilinear thruster model (4) and the dynamics of the thruster control loop.

5. Control allocation

5.1. General problem formulation

In the following it is assumed that the fixed direction (non-rotable) thrusters are only available actuators for control allocation. Under the assumptions mentioned above, a general thruster iTh , $i = \overline{1, p}$ is modelled by a modified affine model shown in Fig. 6. Vector of forces and moments, exerted by thruster iTh , can be written as

$${}^i\tau = \begin{bmatrix} {}^iT \\ {}^iQ \end{bmatrix} = \begin{bmatrix} {}^iT^i e \\ {}^iT({}^i r \times {}^i e) \end{bmatrix} = \begin{bmatrix} {}^i e_x \\ {}^i e_y \\ {}^i e_z \\ ({}^i r \times {}^i e)_x \\ ({}^i r \times {}^i e)_y \\ ({}^i r \times {}^i e)_z \end{bmatrix} {}^iT. \quad (8)$$

Superposition of the individual contributions ${}^i\tau$, $i = \overline{1, p}$ leads to total vector of propulsion forces and moments τ :

$$\tau = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ \tau_K \\ \tau_M \\ \tau_N \end{bmatrix} = \sum_{i=1}^p {}^i\tau = \sum_{i=1}^p \begin{bmatrix} {}^i e \\ ({}^i r \times {}^i e) \end{bmatrix} {}^iT$$

$$= \underbrace{\begin{bmatrix} {}^1 e_x & \dots & {}^i e_x & \dots & {}^p e_x \\ {}^1 e_y & \dots & {}^i e_y & \dots & {}^p e_y \\ {}^1 e_z & \dots & {}^i e_z & \dots & {}^p e_z \\ ({}^1 r \times {}^1 e)_x & \dots & ({}^i r \times {}^i e)_x & \dots & ({}^p r \times {}^p e)_x \\ ({}^1 r \times {}^1 e)_y & \dots & ({}^i r \times {}^i e)_y & \dots & ({}^p r \times {}^p e)_y \\ ({}^1 r \times {}^1 e)_z & \dots & ({}^i r \times {}^i e)_z & \dots & ({}^p r \times {}^p e)_z \end{bmatrix}}_T \underbrace{\begin{bmatrix} {}^1 T \\ \vdots \\ {}^i T \\ \vdots \\ {}^p T \end{bmatrix}}_T = T\mathbf{f}, \quad (9)$$

where $T \in \mathbb{R}^{6 \times p}$ is the *thruster configuration matrix* and $\mathbf{f} \in \mathbb{R}^p$ is vector of *control forces*. For azimuth thrusters ${}^i e = {}^i e(\alpha)$ and ${}^i r = {}^i r(\alpha)$, which means that ${}^i\tau = {}^i\tau(\alpha)$ and $T = T(\alpha)$. However, for fixed direction thrusters $\alpha = \alpha_0 = \text{const.}$ and $T = T(\alpha_0) = \text{const.}$

Substituting ${}^iT = {}^i K u$ in (9) yields

$$\tau = \begin{bmatrix} {}^1 e_x & \dots & {}^i e_x & \dots & {}^p e_x \\ {}^1 e_y & \dots & {}^i e_y & \dots & {}^p e_y \\ {}^1 e_z & \dots & {}^i e_z & \dots & {}^p e_z \\ ({}^1 r \times {}^1 e)_x & \dots & ({}^i r \times {}^i e)_x & \dots & ({}^p r \times {}^p e)_x \\ ({}^1 r \times {}^1 e)_y & \dots & ({}^i r \times {}^i e)_y & \dots & ({}^p r \times {}^p e)_y \\ ({}^1 r \times {}^1 e)_z & \dots & ({}^i r \times {}^i e)_z & \dots & ({}^p r \times {}^p e)_z \end{bmatrix}$$

$$\times \underbrace{\begin{bmatrix} {}^1 K & 0 & 0 \\ \vdots & \ddots & \vdots \\ 0 & {}^i K & 0 \\ \vdots & \vdots & \ddots \\ 0 & 0 & {}^p K \end{bmatrix}}_K \underbrace{\begin{bmatrix} u \\ \vdots \\ u \\ \vdots \\ u \end{bmatrix}}_u = T\mathbf{K}\mathbf{u}, \quad (10)$$

where $K \in \mathbb{R}^{6 \times p}$ is the *force coefficient matrix* and $\mathbf{u} \in \mathbb{R}^p$ is the *control vector*. Introducing substitution

$$\mathbf{B} = T\mathbf{K}, \quad (11)$$

where $\mathbf{B} \in \mathbb{R}^{6 \times p}$ is the *thruster control matrix*, (10) can be rewritten as

$$\tau = \mathbf{B}\mathbf{u}. \quad (12)$$

Zero-row in \mathbf{B} means that the corresponding DOF is not directly controllable with the particular thruster configuration.

Assuming that thrusters are identical, the coefficients ${}^i K$ are the same for all thrusters,

$${}^1 K = \dots = {}^p K = K \quad (13)$$

Eq. (11) can be simplified as

$$\mathbf{B} = T\mathbf{K} = T(K\mathbf{I}_p) = K(T\mathbf{I}_p) = K\mathbf{T}. \quad (14)$$

Each component ${}^i u$ of the control vector \mathbf{u} is limited by

constraint

$$-u_m \leq u \leq u_m, \quad i = \overline{1, p}. \quad (15)$$

Constraint (15) represents thruster velocity saturation, i.e. the physical construction of the thruster iTh imposes velocity limitations because the thruster cannot rotate faster than its maximum velocity. For the control vector \mathbf{u} set of constraints (15) can be written in compact vector form as

$$-\mathbf{u}_m \leq \mathbf{u} \leq \mathbf{u}_m, \quad (16)$$

where

$$\mathbf{u}_m = [u_m \dots u_m \dots u_m]^T. \quad (17)$$

For identical thrusters, $u_m = \dots = u_m = u_m$ and

$$\mathbf{u}_m = u_m \begin{bmatrix} 1 & \dots & 1 & \dots & 1 \end{bmatrix}^T.$$

The constrained control subset Ω is defined as a set of all control vectors \mathbf{u} which satisfy (16).

Finally, the general constrained control allocation problem for open-frame underwater vehicles can be formulated as

For given $\boldsymbol{\tau}$, find $\mathbf{u} \in \Omega$ such that $\mathbf{B}\mathbf{u} = \boldsymbol{\tau}$.

If condition $\mathbf{u} \in \Omega$ is removed, the problem becomes unconstrained.

Thruster configuration matrices for the FALCON and the URIS, shown in Table 3, are obtained from (14), assuming that all thrusters are identical. Parameter A is given as $A = (b/2)\sin \alpha + (a/2)\cos \alpha$.

It can be seen that the uncontrollable DOF for the FALCON are *roll* and *pitch*, since the fourth and the fifth row of \mathbf{B} are zero-rows. In a similar way, uncontrollable DOF for the URIS are *heave*, *roll* and *pitch*.

5.2. Decomposition of motion

In the following, the general control allocation problem will be separated into two subproblems and each will be treated individually. The first subproblem is related with the motion in the horizontal plane, and the second with the motion in the vertical plane. Decomposition of the motion is given in Table 4 for the FALCON and in Table 5 for the URIS.

Table 3

Thruster control matrix for different thruster configurations

FALCON	URIS
$\mathbf{B} = \begin{bmatrix} \cos \alpha & \cos \alpha & \cos \alpha & \cos \alpha & 0 \\ \sin \alpha & -\sin \alpha & \sin \alpha & -\sin \alpha & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ A & -A & -A & A & 0 \end{bmatrix}$	$\mathbf{B} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ R & -R & R & -R \end{bmatrix}$

Table 4

Decomposition of the FALCON motion

FALCON
<p>Horizontal thrusters (motion in horizontal plane)</p> $\boldsymbol{\tau}^{HT} = \begin{bmatrix} \tau_X \\ \tau_Y \\ \tau_N \end{bmatrix} = \mathbf{K} \underbrace{\begin{bmatrix} \cos \alpha & \cos \alpha & \cos \alpha & \cos \alpha \\ \sin \alpha & -\sin \alpha & \sin \alpha & -\sin \alpha \\ A & -A & -A & A \end{bmatrix}}_{\mathbf{B}^{HT}} \underbrace{\begin{bmatrix} u_1^{HT} \\ u_2^{HT} \\ u_3^{HT} \\ u_4^{HT} \end{bmatrix}}_{\mathbf{u}^{HT}}$
<p>Vertical thruster (motion in vertical plane)</p> $\boldsymbol{\tau}^{VT} = [\tau_Z] = \underbrace{\mathbf{K}}_{\mathbf{B}^{VT}} \underbrace{u_5^{VT}}_{\mathbf{u}^{VT}}$

Table 5

Decomposition of the URIS motion

URIS
<p>Horizontal thrusters (motion in horizontal plane)</p> $\boldsymbol{\tau}^{HT} = \begin{bmatrix} \tau_X \\ \tau_Y \\ \tau_N \end{bmatrix} = \mathbf{K} \underbrace{\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ R & -R & R & -R \end{bmatrix}}_{\mathbf{B}^{HT}} \underbrace{\begin{bmatrix} u_1^{HT} \\ u_2^{HT} \\ u_3^{HT} \\ u_4^{HT} \end{bmatrix}}_{\mathbf{u}^{HT}}$
<p>Vertical thruster (motion in vertical plane)</p> <p>Controllable DOF: no motion in the vertical plane</p>

It can be seen that the motion of the FALCON in the vertical plane is determined by vertical thruster and heave force is directly proportional to the value of control signal. In the case of a partial fault in a vertical thruster, the only available solution is to limit angular velocity of the thruster. In the case of total fault (failure), the thruster must be switched off and the vehicle must be recovered for repair.

The situation is different for motion in the horizontal plane, where the number of horizontal thrusters is four and the number of controllable DOF is three. In this case inherent redundancy in thruster configuration enables successful control allocation in the case of partial or even total fault in a horizontal thruster.

The control allocation of horizontal thrusters is the topic of the discussion in the following sections. Before the full problem is formulated, relevant vectors and matrices will be normalised, in order to make problem more understandable and easier to visualise and solve.

5.3. Normalisation

Normalisation means that vector components are divided by their maximum values, such that each

component is dimensionless number that lies between -1 and $+1$. Normalised vectors and matrices are underlined, in order to distinguish them from the standard nomenclature. Normalisation procedure will be explained on the example of the X-shaped thruster configuration (the FALCON). Recall from Fig. 6 that maximum thruster force is given by

$$T_m = Ku_m. \quad (18)$$

The first step is to find maximum values (modules) of the surge and sway forces and the yaw moment. Three characteristic cases are indicated in Fig. 7. It can be seen that

$$\tau_{Xm} = 4T_m \cos \alpha = 4Ku_m \cos \alpha \Rightarrow K \cos \alpha = \frac{\tau_{Xm}}{4u_m}, \quad (19)$$

$$\tau_{Ym} = 4T_m \sin \alpha = 4Ku_m \sin \alpha \Rightarrow K \sin \alpha = \frac{\tau_{Ym}}{4u_m}, \quad (20)$$

$$\tau_{Nm} = 4T_m A = 4Ku_m A \Rightarrow KA = \frac{\tau_{Nm}}{4u_m}. \quad (21)$$

The second step is to substitute expressions (19)–(21) in the standard relation $\tau^{HT} = \mathbf{B}^{HT} \mathbf{u}^{HT}$ as follows:

$$\tau^{HT} = \begin{bmatrix} \tau_X \\ \tau_Y \\ \tau_N \end{bmatrix} = \underbrace{\begin{bmatrix} K \cos \alpha & K \cos \alpha & K \cos \alpha & K \cos \alpha \\ K \sin \alpha & -K \sin \alpha & K \sin \alpha & -K \sin \alpha \\ KA & -KA & -KA & KA \end{bmatrix}}_{\mathbf{B}^{HT}} \underbrace{\begin{bmatrix} u_1^{HT} \\ u_2^{HT} \\ u_3^{HT} \\ u_4^{HT} \end{bmatrix}}_{\mathbf{u}^{HT}} = \begin{bmatrix} \frac{\tau_{Xm}}{4u_m} & \frac{\tau_{Xm}}{4u_m} & \frac{\tau_{Xm}}{4u_m} & \frac{\tau_{Xm}}{4u_m} \\ \frac{\tau_{Ym}}{4u_m} & -\frac{\tau_{Ym}}{4u_m} & \frac{\tau_{Ym}}{4u_m} & -\frac{\tau_{Ym}}{4u_m} \\ \frac{\tau_{Nm}}{4u_m} & -\frac{\tau_{Nm}}{4u_m} & -\frac{\tau_{Nm}}{4u_m} & \frac{\tau_{Nm}}{4u_m} \end{bmatrix} \begin{bmatrix} u_1^{HT} \\ u_2^{HT} \\ u_3^{HT} \\ u_4^{HT} \end{bmatrix}. \quad (22)$$

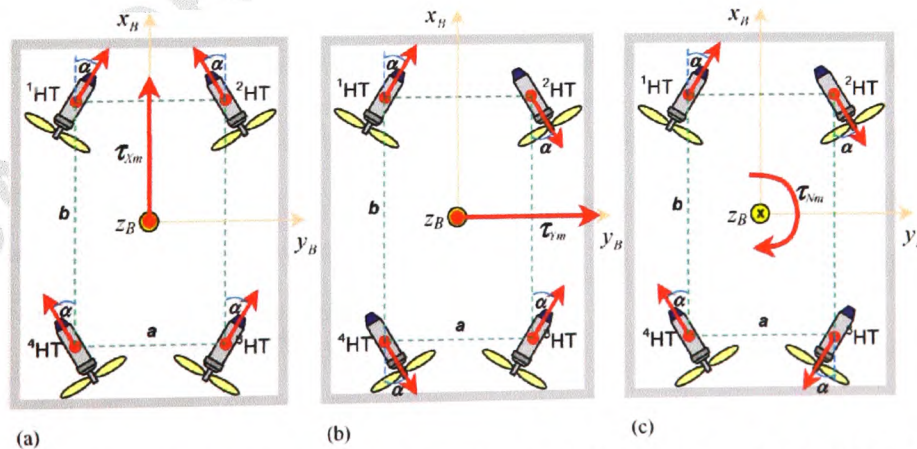


Fig. 7. Three cases for finding the maximum modules of force and moment vectors (X-shaped thruster configuration): (a) Max. surge force τ_{Xm} . (b) Max. sway force τ_{Ym} . (c) Max. yaw moment τ_{Nm} .

Finally, the last step is to rewrite (22) in the normalised form as follows:

$$\begin{bmatrix} \tau_X \\ \tau_{Xm} \\ \tau_Y \\ \tau_{Ym} \\ \tau_N \\ \tau_{Nm} \end{bmatrix}_{\tau^{HT}} = \underbrace{\begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \end{bmatrix}}_{\mathbf{B}^{HT}} \begin{bmatrix} u_m \\ u_m \\ u_m \\ u_m \\ u_m \\ u_m \end{bmatrix}_{\mathbf{u}^{HT}} \Leftrightarrow \tau^{HT} = \mathbf{B}^{HT} \mathbf{u}^{HT}. \quad (23)$$

The normalised form (23) has a number of advantages compared to the standard form $\tau^{HT} = \mathbf{B}^{HT} \mathbf{u}^{HT}$ as follows:

1. Components of the vectors τ^{HT} and \mathbf{u}^{HT} are dimensionless number, restricted to the standard interval $[-1, +1]$. This enables better understanding and easier visualisation of the problem.
2. All physical parameters are removed from the matrix \mathbf{B}^{HT} during the normalisation process. The compact form of \mathbf{B}^{HT} simplifies calculations and, as will be shown later, leads to the very simple representation of the weighted pseudo-inverse solution and a clear geometric interpretation of the control allocation problem.

Normalisation for the cross-shaped thruster configuration (URIS) can be performed in a similar way. The final expression is given by

$$\begin{bmatrix} \tau_X \\ \tau_{Xm} \\ \tau_Y \\ \tau_{Ym} \\ \tau_N \\ \tau_{Nm} \end{bmatrix}_{\tau^{HT}} = \underbrace{\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \end{bmatrix}}_{\mathbf{B}^{HT}} \begin{bmatrix} u_m \\ u_m \\ u_m \\ u_m \\ u_m \\ u_m \end{bmatrix}_{\mathbf{u}^{HT}} \Leftrightarrow \tau^{HT} = \mathbf{B}^{HT} \mathbf{u}^{HT}. \quad (24)$$

5.4. Problem formulation for horizontal thrusters

The control allocation problem for the motion in the horizontal plane can be formulated using normalised variables as follows:

For given τ^{HT} , find $\mathbf{u}^{HT} \in \mathcal{Q}^{HT}$ such that $\mathbf{B}^{HT} \mathbf{u}^{HT} = \tau^{HT}$.

In the following, the problem is analysed in more details from the general control allocation perspective.

- The true control input is

$$\mathbf{u}^{HT} = \begin{bmatrix} u_1^{HT} \\ u_2^{HT} \\ u_3^{HT} \\ u_4^{HT} \end{bmatrix} = \begin{bmatrix} u_m \\ u_m \\ u_m \\ u_m \end{bmatrix} \in \mathbb{R}^4 \quad (m = 4).$$

- The virtual control input is

$$\tau^{HT} = \begin{bmatrix} \tau_X \\ \tau_{Xm} \\ \tau_Y \\ \tau_{Ym} \\ \tau_N \\ \tau_{Nm} \end{bmatrix} \in \mathbb{R}^3 \quad (k = 3).$$

- Control effectiveness matrix \mathbf{B}^{HT} is given by

$$\text{FALCON: } \mathbf{B}^{HT} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \end{bmatrix}. \quad (25)$$

$$\text{URIS: } \mathbf{B}^{HT} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \end{bmatrix}. \quad (26)$$

- Actuator position constraints are

$$\begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \leq \begin{bmatrix} u_1^{HT} \\ u_2^{HT} \\ u_3^{HT} \\ u_4^{HT} \end{bmatrix} \leq \begin{bmatrix} +1 \\ +1 \\ +1 \\ +1 \end{bmatrix}.$$

Equation $\mathbf{B}^{HT} \mathbf{u}^{HT} = \tau^{HT}$ represents the system of equations

$$\frac{1}{4} u_1^{HT} + \frac{1}{4} u_2^{HT} + \frac{1}{4} u_3^{HT} + \frac{1}{4} u_4^{HT} = \tau_X, \quad (27)$$

$$\text{FALCON: } \frac{1}{4} u_1^{HT} - \frac{1}{4} u_2^{HT} + \frac{1}{4} u_3^{HT} - \frac{1}{4} u_4^{HT} = \tau_Y,$$

$$\frac{1}{4} u_1^{HT} - \frac{1}{4} u_2^{HT} - \frac{1}{4} u_3^{HT} + \frac{1}{4} u_4^{HT} = \tau_N,$$

$$\frac{1}{2} u_1^{HT} + \frac{1}{2} u_2^{HT} = \tau_X.$$

$$\text{URIS: } \frac{1}{2} u_3^{HT} + \frac{1}{2} u_4^{HT} = \tau_Y, \quad (28)$$

$$\frac{1}{4} u_1^{HT} - \frac{1}{4} u_2^{HT} + \frac{1}{4} u_3^{HT} - \frac{1}{4} u_4^{HT} = \tau_N.$$

Each equation in (27) and (28) represents a hyperplane in \mathcal{R}^4 . Consequently, (27) and (28) can be rewritten as

$$\begin{aligned} \pi_X: \quad \mathbf{N}_X^T \cdot \mathbf{u}^{HT} &= \tau_X, \\ \pi_Y: \quad \mathbf{N}_Y^T \cdot \mathbf{u}^{HT} &= \tau_Y, \\ \pi_N: \quad \mathbf{N}_N^T \cdot \mathbf{u}^{HT} &= \tau_N, \end{aligned} \quad (29)$$

where normal vectors \mathbf{N}_X , \mathbf{N}_Y and \mathbf{N}_N , defined as

$$\begin{aligned} \mathbf{N}_X^T &= \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T, \\ \text{FALCON: } \mathbf{N}_Y^T &= \begin{bmatrix} 1 & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \end{bmatrix}^T, \\ \mathbf{N}_N^T &= \begin{bmatrix} 1 & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \end{bmatrix}^T. \end{aligned} \quad (30)$$

$$\begin{aligned} \mathbf{N}_X^T &= \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}^T, \\ \text{URIS: } \mathbf{N}_Y^T &= \begin{bmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}^T, \\ \mathbf{N}_N^T &= \begin{bmatrix} 1 & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \end{bmatrix}^T \end{aligned} \quad (31)$$

are orthogonal on the hyperplanes π_X , π_Y and π_N , respectively. The intersection of these hyperplanes is a convex set, denoted by \mathcal{K}^{HT} , which represent the set of all points \mathbf{u}^{HT} that satisfy $\mathbf{B}^{HT} \mathbf{u}^{HT} = \boldsymbol{\tau}^{HT}$.

The thruster velocity saturation constraints determine the constrained control subset \mathcal{Q}^{HT} , that is, the unit four-dimensional hypercube in \mathcal{R}^4 :

$$\mathcal{Q}^{HT} = \{\mathbf{u}^{HT} \in \mathcal{R}^4 \mid \|\mathbf{u}^{HT}\|_\infty \leq 1\} \subset \mathcal{R}^4. \quad (32)$$

Intersection of \mathcal{K}^{HT} and \mathcal{Q}^{HT} is a solution set, denoted by \mathcal{Z}^{HT} . Geometric interpretation of the control allocation problem for the motion in the horizontal plane using normalised variables is given by

For a given $\boldsymbol{\tau}^{HT}$, find intersection $\mathcal{Z}^{HT} = \mathcal{K}^{HT} \cap \mathcal{Q}^{HT}$.

The control effectiveness matrix \mathbf{B}^{HT} performs a linear transformation from the true control space \mathcal{R}^3 to the virtual control space \mathcal{R}^4 . The image of $\mathcal{Q}^{HT} \subset \mathcal{R}^4$ is called the attainable command set and denoted by Φ^{HT} . The boundary of Φ^{HT} is denoted by $\partial(\Phi^{HT})$. The character of the solution is closely related with the position of the vector $\boldsymbol{\tau}^{HT}$ relative to Φ^{HT} . Three cases are possible:

- If $\boldsymbol{\tau}^{HT} \notin \Phi^{HT}$, then \mathcal{Z}^{HT} is empty (i.e. no exact solution exists),
- If $\boldsymbol{\tau}^{HT} \in \partial(\Phi^{HT})$, then \mathcal{Z}^{HT} has exactly one element (i.e. there is one, unique solution³),

³The matrix \mathbf{B}^{HT} satisfies the linear independency condition (non-complanar controls), since every 3×3 partition of \mathbf{B}^{HT} is non-singular. This leads to a unique solution of the control allocation problem for the case when virtual control input lies on the boundary $\partial(\Phi^{HT})$.

- \mathcal{Z}^{HT} has more than one element (i.e. there are many solutions).

In order to extract a unique, “best” solution from a solution set, it is necessary to introduce criteria, which is minimised by the chosen solution. The most suitable criteria for underwater applications is a control energy cost function.

The optimal control input \mathbf{u}^{HT} is given as a solution to a two-step optimisation problem:

$$\mathbf{u}^{HT} = \arg \min_{\mathbf{u}^{HT} \in \mathcal{Z}^{HT}} \|\mathbf{W}_u^{HT} \mathbf{u}^{HT}\|_2, \quad (33)$$

$$\boldsymbol{\Psi}^{HT} = \arg \min_{\mathbf{u}^{HT} \in \mathcal{Q}^{HT}} \|\mathbf{B}^{HT} \mathbf{u}^{HT} - \boldsymbol{\tau}^{HT}\|_2. \quad (34)$$

Problem (33)–(34) can be interpreted as follows:

Given $\boldsymbol{\Psi}^{HT}$, the set of feasible control inputs that minimise $\|\mathbf{B}^{HT} \mathbf{u}^{HT} - \boldsymbol{\tau}^{HT}\|_2$, find the control input \mathbf{u}^{HT} that minimises $\|\mathbf{W}_u^{HT} \mathbf{u}^{HT}\|_2$.

The design parameter \mathbf{W}_u^{HT} is a positive definite weighting matrix, weighting the control energy, and can be used for thruster prioritisation, i.e. to decide which thruster should be used primarily. The weighting matrix \mathbf{W}_u^{HT} is usually chosen to be a diagonal matrix

$$\mathbf{W}_u^{HT} = \begin{bmatrix} w_1^{HT} & 0 & 0 & 0 \\ 0 & w_2^{HT} & 0 & 0 \\ 0 & 0 & w_3^{HT} & 0 \\ 0 & 0 & 0 & w_4^{HT} \end{bmatrix}, \quad (35)$$

where $w_i^{HT} > 0$ is the weight associated with the thruster i^{HT} , $i = 1, 4$. Using \mathbf{W}_u^{HT} , a faulty thruster is penalised by increasing its weight, as explained in the following.

5.5. Weighting matrix \mathbf{W}_u^{HT} for fault-free case

In the fault-free case, all horizontal thrusters have the same priority and \mathbf{W}_u^{HT} is chosen to be equal to identity matrix

$$\mathbf{W}_u^{HT} = \mathbf{I}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (36)$$

5.6. Weighting matrix \mathbf{W}_u^{HT} for faulty situations

Two faulty situations are possible: a *partial* fault and a *total* fault (failure).

In the case of a partial fault in i^{HT} , the thruster is typically allowed to continue operation with the restricted usage, i.e. the new constraint (saturation) bounds are

$$-s_i^{HT} \leq u_i^{HT} \leq s_i^{HT}, \quad (37)$$

where $0 < s_i^{HT} < 1$. The numerical value of s_i^{HT} depends on the type of the fault and is selected in advance for each particular fault type. For example, restricted constraint bound $s_i^{HT} = 0.75$ can be selected for a faulty state "Jammed propeller" of i^{HT} , which means that the thruster's operating range is restricted to 75% of its nominal range. In addition to the change of the constraint bounds, the weight w_i^{HT} of the faulty thruster is increased using

$$w_i^{HT} = 1 + \Delta w_i^{HT}, \quad (38)$$

where

$$\Delta w_i^{HT} = 2 \left(\frac{1}{s_i^{HT}} - 1 \right). \quad (39)$$

The weight update (39) is introduced to penalise the faulty thruster, prioritise healthy thrusters and to compensate restricted usage of the faulty thruster in an optimal way.

In the case of a total fault in i^{HT} , the thruster is switched off and removed from the allocation process. The same effect can be achieved using formulation (33) by allowing $w_i^{HT} \rightarrow \infty$. In this way, the redundancy is eliminated from the system of equation $\mathbf{B}^{HT} \mathbf{u}^{HT} = \mathbf{r}^{HT}$, which can now be solved in a standard way.

5.7. Remarks on vertical thrusters

For the motion of the FALCON in the vertical plane normalisation yields

$$\begin{aligned} \mathbf{r}^{VT} = [\tau_Z] = \underbrace{\mathbf{K}}_{\mathbf{B}^{VT}} \underbrace{\mathbf{u}^{VT}}_{\mathbf{u}^{VT}} \Bigg\} &\Rightarrow \underbrace{\begin{bmatrix} \tau_Z \\ \tau_{Zm} \end{bmatrix}}_{\mathbf{r}^{VT}} = \underbrace{[1]}_{\mathbf{B}^{VT}} \underbrace{\begin{bmatrix} 1 \mathbf{u}^{VT} \\ \mathbf{u}_m \end{bmatrix}}_{\mathbf{u}^{VT}} \\ \mathbf{K} = \frac{\tau_{Zm}}{\mathbf{u}_m} & \Leftrightarrow \mathbf{r}^{VT} = \mathbf{B}^{VT} \mathbf{u}^{VT}. \end{aligned} \quad (40)$$

The weighting matrix for vertical thruster \mathbf{W}_u^{VT} becomes a scalar, defined as

$$\mathbf{W}_u^{VT} = w_i^{VT}, \quad (41)$$

where $w_i^{VT} > 0$ is the weight associated with the thruster i^{VT} .

In the fault-free case, $w_i^{VT} = 1$.

In the case of a partial fault, the new restricted constraint (saturation) bounds are

$$-s_i^{VT} \leq u_i^{VT} \leq s_i^{VT}, \quad (42)$$

where $0 < s_i^{VT} < 1$. Since $\mathbf{B}^{VT} = 1$, weighted pseudo-inverse solution \mathbf{B}_w^{+VT} does not depend on w_i^{VT} . Hence, restriction of constraint bounds (42) is the only action that is undertaken in the case of a partial fault in the vertical thruster.

In the case of a total fault, the thruster i^{VT} must be switched off and removed from the allocation process. In this case, heave becomes uncontrollable DOF.

6. Fault diagnosis and accommodation system

6.1. Architecture

The overall functional architecture of the proposed FDAS, shown in Fig. 8, represents the expanded version of the improved architecture shown in Fig. 2b. The description of the architecture will be given in an hierarchical way, such that the general description and the main idea of the method are presented in this section, while more details about individual components can be found in the following subsections.

The input to the FDAS is the vector \mathbf{r}_d , obtained by filtering (smoothing) the desired vector of propulsion forces and moments \mathbf{r}_d^* , generated by the HCU. The output of the FDAS is the vector of desired thruster velocities \mathbf{n} , transformed into the form that is compatible (acceptable) by the TCUs. A short description of the individual components is given in the following.

6.1.1. FDS

FDU: The FDS uses FDUs to monitor the state of the thrusters. The FDU is a software module associated with the thruster, able to detect internal faults (for example, temperature of the windings exceeds limits) and external faults (for example, jammed propeller). The output of the FDU is a fault indicator f_i , the code of the fault.

Integration: The fault indicators f_i are integrated into the total fault indicator vector \mathbf{f} inside this block. The vector \mathbf{f} is a carrier of thrusters' states.

6.1.2. FAS

Demux: In accordance with the decomposition of the motion, shown in Tables 4 and 5, this block symbolically indicates separation of the vector \mathbf{r}_d into two parts: \mathbf{r}_d^{HT} , representing the DOF (surge, sway and yaw) controllable by horizontal thrusters, and \mathbf{r}_d^{VT} , representing the DOF (heave) controllable by vertical thruster.

Pseudo-inverse: This block finds weighted pseudo-inverse solution of the control allocation problem, separately for horizontal and vertical thrusters. For horizontal thrusters, the solution \mathbf{u}^{HT} is given by (49). For vertical thruster, the solution \mathbf{u}^{VT} is given by (51).

Approximation: The weighted pseudo-inverse solution \mathbf{u}^{HT} can be feasible (satisfies all constraints) or unfeasible (violates some constraint(s)). The output \mathbf{u}^{*HT} of this block must be feasible solution all the time. Hence, if \mathbf{u}^{HT} is feasible, then $\mathbf{u}^{*HT} = \mathbf{u}^{HT}$. Otherwise, T -approximation (truncation) or S -approximation (scaling) is performed to find feasible approximation \mathbf{u}^{*HT} .

Co-ordinator: The role of this block is to undertake remedial actions in accordance to the context of the total fault indicator vector \mathbf{f} and the instructions, stored in the fault code table. For each possible fault type the

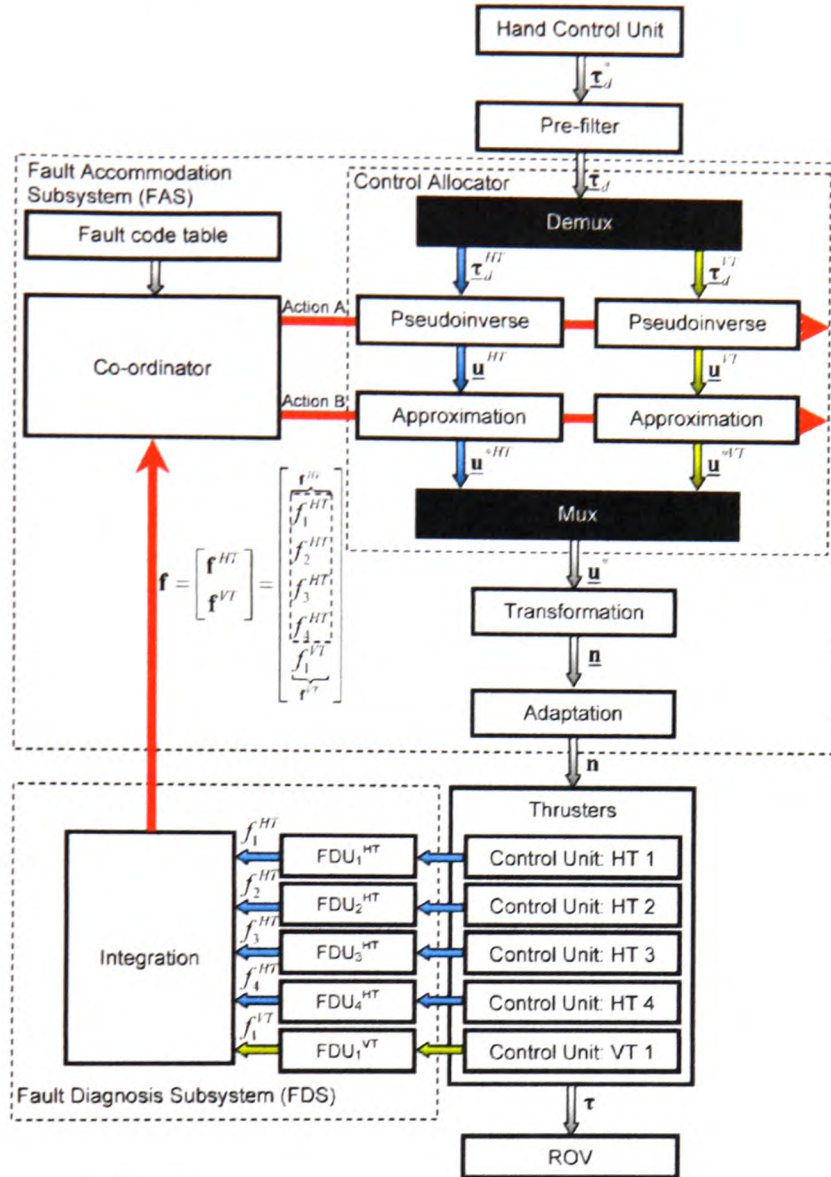


Fig. 8. Overall functional architecture of the proposed FDAS.

fault code table has stored corresponding actions A and B. The action A is related with the weight updates of weighting matrices, used to find the weighted pseudo-inverse solution. The action B is related with change of constraint bounds, in accordance to fault type.

Mux: This block performs opposite role of Demux block, i.e. it merges feasible solutions \underline{u}^{*HT} and \underline{u}^{*VT} into composite solution vector \underline{u}^* .

Transformation: The vector \underline{u}^* cannot be directly applied to drive thrusters. It must be transformed into the vector of desired thruster velocities \underline{n} . This block performs this function, using transformation $n_i = \text{sgn } u_i \sqrt{|u_i|}$ for each component.

Adaptation: Different TCUs accept data in different format. For example, the desired angular velocity for the TCU of the FALCON must be represented as an integer number between -100 and $+100$. In contrast, the same variable must be converted into the voltage in order to be applied to drive the thruster of the URIS. This block transforms the vector \underline{n} into the vector $\underline{\tau}$, which has the form adapted for the particular TCU.

Vector \underline{n} is used to drive the thrusters, which generate a vector of propulsion forces and moments $\underline{\tau}$. The proposed FDAS guarantees that the condition $\underline{\tau} = \underline{\tau}_d$ is satisfied for all $\underline{\tau}_d^{HT}$ that lies inside the convex polyhedron Φ_p^{HT} (feasible region for pseudo-inverse),

which is a subset of the attainable command set Φ^{HT} . That is, if $\tau_d^{HT} \in \Phi_p^{HT}$, the FDAS will find the exact solution of the control allocation problem, optimal in l_2 sense. Otherwise, the solution obtained by the FDAS is a very good approximation that lies on the boundary $\partial(\Omega^{HT})$, which depends on design parameters (weighting matrices) and type of approximation.

6.2. Fault diagnosis subsystem

6.2.1. Fault classification

Thrusters are liable to different fault types during the underwater mission. Some of these faults (partial faults) are not critical and the thruster is able to continue operation in the presence of a fault with the restricted usage, i.e. reduced maximum velocity. In other cases (total faults—failures) the thruster must be switched off and the mission has to be continued with the remaining operable thrusters. Thruster faults are classified into two main classes (Omerdic et al., 2003):

- **Internal faults** (e.g. temperature of the windings is out of range, lost communication between the TCU and main processor, drop in bus voltage, etc.)
- **External faults** (e.g. jammed or broken propeller).

Table 6
The fault code table

Thruster state	Class	Type	Indicator f_i	Constraint bound s_i
Normal (Fault-free)	—	—	1	1.00
Jammed propeller	Ext.	Partial	2	0.75
Heavy jammed propeller	Ext.	Partial	3	0.50
Broken propeller	Ext.	Total	4	0.00
Unknown fault	Ext.	Partial	5	0.25
Internal fault	Int.	Total	6	0.00

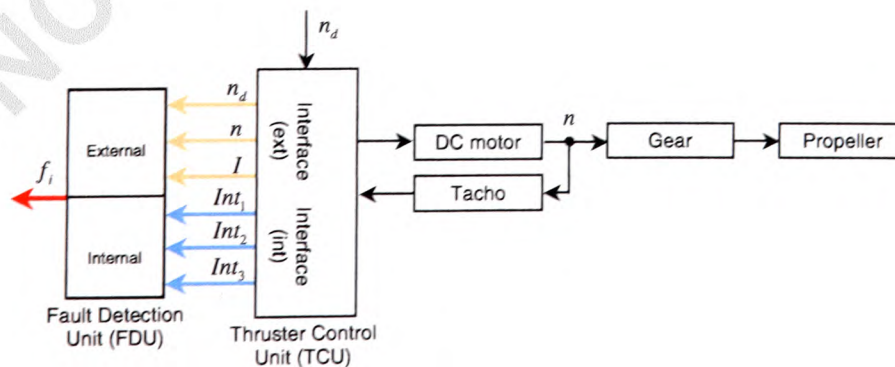


Fig. 9. Block diagram showing connections between the FDU and the TCU for thruster 'Th'.

6.2.2. Fault code table

Relationships between thruster states, fault types and remedial actions are stored in the fault code table (Table 6). It must be emphasized, at this point, that this fault code table is just a suggestion, intended to reveal the main ideas of the proposed FDAS. New states (rows) can be added, and the existing relationships can be changed, in order to cover intermediate cases and meet specific requirements.

6.2.3. Fault detection unit

The FDU is used for monitoring the thruster states and reporting any faulty situation. The FDU is a software module, able to detect internal and external faults. Connections between the FDU and the TCU for arbitrary thruster 'Th' are indicated in Fig. 9.

Signals Int_1, Int_2, \dots for detection of internal faults are already available in existing TCUs for both vehicles. In particular, the TCU for the URIS, based on Maxon Servoamplifier ADS 50/5, has status-reading signal $Int_1 = \text{"Ready"}$, which can be used to report internal faults (excess temperature or excess current). Similarly, communication protocol for FALCON provides monitoring of the winding temperature (Int_1) and bus voltage (Int_2) of each thruster. In order to build a universal FDU, capable of detecting both internal and external faults, it is necessary to augment the existing

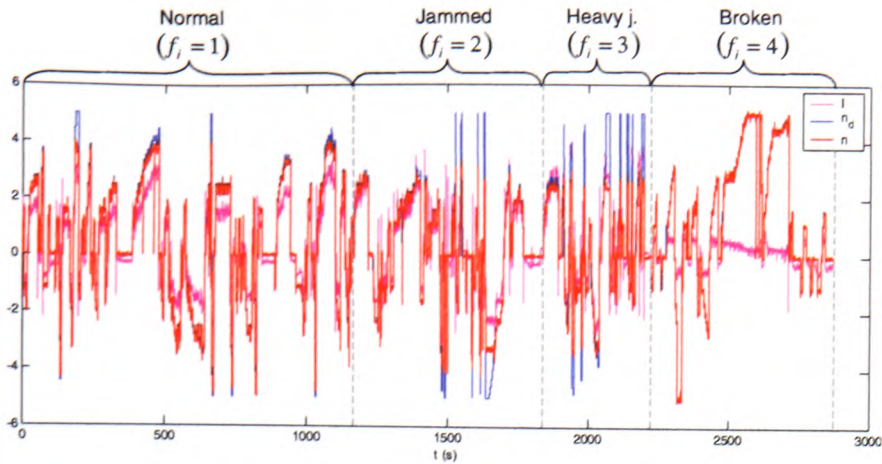


Fig. 10. Time diagrams of raw training data.

internal protection with a software module for fast and reliable detection of external faults.

For detection of external faults available signals are actual velocity of the motor shaft n and current consumption I of the thruster. For the URIS, these signals are called “Monitor n ” and “Monitor I ”, respectively; for the FALCON, the communication protocol enables output speed and winding current to be read. By monitoring n and I , together with desired speed n_d obtained as output of the FDAS, the FDU must be able to detect external thruster fault.

Finally, the universal FDU integrates both parts (internal and external) into one unit, which is able to detect internal and external faults (Fig. 9). Integration is performed using a priority scheme, where total faults have higher priority than partial faults. Indicator f_i , the output of the FDU, is the code of the fault.

Implementation of the FDU involves two phases: *off-line training* and *on-line fault detection*.

Off-line training phase: The first stage in the training phase is acquisition of training data. Test trials were performed with the URIS at University of Girona in July 2002, and training data were saved in files. Normal state and three different fault cases were considered (jammed, heavy jammed and broken propeller).⁴ A jammed propeller was simulated by attaching an object to it. When the thruster is actuated, the propeller and the object rotate together, representing additional load for the motor. Heavy-jammed propeller was simulated with two objects attached. In order to simulate broken propeller, blades were removed from the shaft. Each record in file consists of acquired data from the TCU (n_d , n and I) and associated fault code f_i . Sampling time was 0.1 s, long enough to ensure that all transient

⁴However, in real applications the number of faulty cases can be higher. In addition, partially broken or damaged propeller blades can be used to cover intermediate cases.

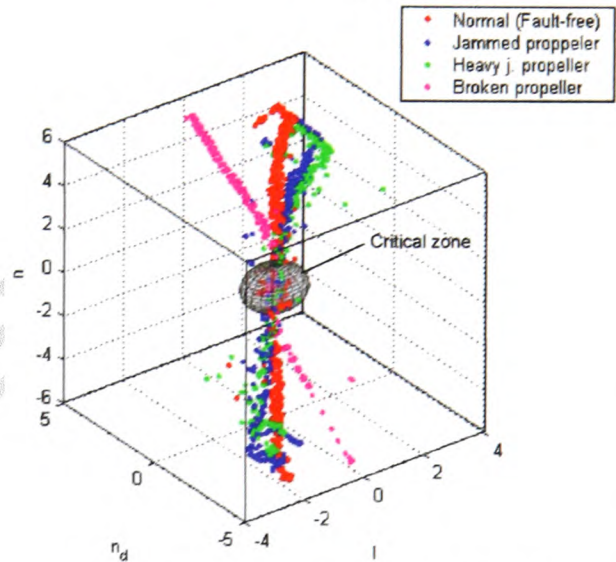


Fig. 11. Pre-processed training data in the 3D space.

responses in thruster control loop disappear. The motion of the URIS was controlled by a joystick, such that all range of possible thruster velocities was covered with enough data points. The real-time experiments were undertaken during the development stage of the URIS, what meant that the inadequate signal conditioning, wiring and shielding resulted in noisy data.

The time diagrams of the raw training data are shown in Fig. 10. Signals from different fault types are connected next to each other in order to make easier their comparison. Noise and outliers (data items that lie very far from the main body of the data) are particularly noticeable in current response.

Data pre-processing filters the raw training data in order to remove outliers and reject noise. Pre-processed

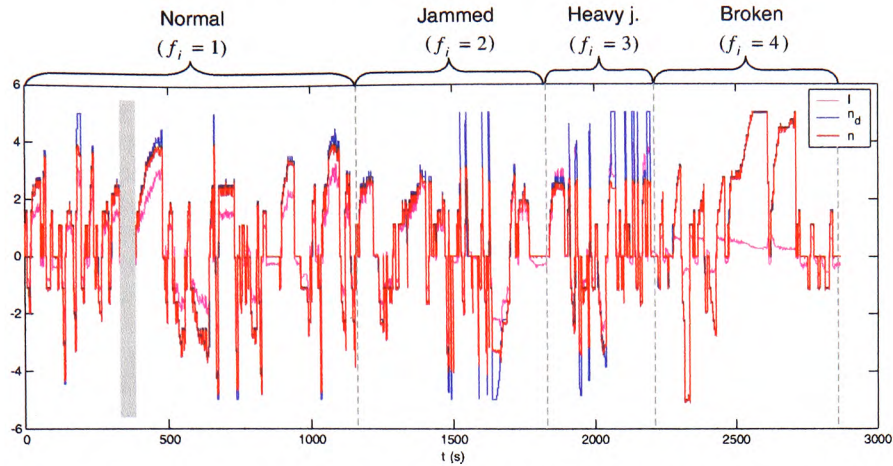


Fig. 12. Time diagrams of pre-processed training data. One of the zero-velocity segments is highlighted. These segments are excluded from the training process.

training data in 3D space are shown in Fig. 11.⁵ Fig. 12 displays time diagrams of pre-processed data. Normalisation step in pre-processing is optional, since all variables have the same range and no one is dominant.

Analysing the distribution of the training data in Fig. 11, the *first* feature that can be noticed is that each fault-type creates certain pattern. In the ideal case, these patterns should be well defined curves. However, the presence of the noise and outliers in training data results that patterns shown in Fig. 11 are “cloudy”. The *second* feature is that the zone around $n_d \approx 0$ (called the *critical zone*) is filled with data from different fault types in such a way that it is very hard to distinguish individual fault types. Geometrically, the critical zone represents intersection of different fault-type patterns. This makes successful fault detection in the critical zone difficult to achieve. In particular, for zero-velocity case $n_d = 0$ thruster does not rotate. Successful and reliable fault detection in this case is impossible, since external faults cannot be detected without shaft rotation. The solution for this problem is an exclusion of the zero-velocity segments from the training process, i.e. the training is performed considering data records with $n_d \neq 0$, as shown in Fig. 12. However, the same exclusion is performed during the on-line fault detection phase.

The problem of thruster fault detection for underwater vehicles has special features, due to environmental conditions in which the vehicle operates. The most important requirements that the FDU should fulfil are:

- reliable and fast fault detection, without false alarms,
- easy integration with the existing control system,
- on-line learning and adaptation to the new types of faults,

⁵Matlab function `medfilt1` was used to filter data and remove outliers at the same time.

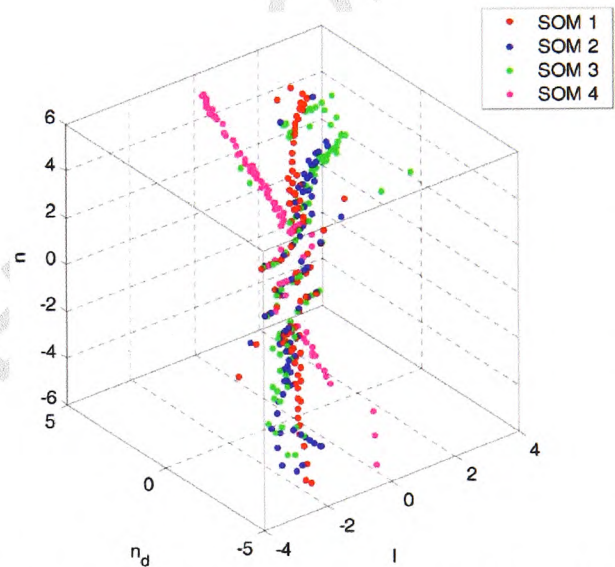


Fig. 13. The second stage in off-line training phase: different fault types (patterns) are replaced by SOM representatives.

- cost efficient i.e. the FDU should use resources already available, without introducing new hardware,
- easy transfer to and implementation in other vehicles.

By carefully examination of the available resources in the existing TCUs, the model-free approach, based on integration of an SOM and fuzzy clustering techniques, is chosen as the best candidate for FDU to fulfil all these requirements (Omerdic et al., 2003). The main idea of the second stage in the training phase is to replace each fault type (pattern) in Fig. 11 with an SOM as shown in Fig. 13, which serves as a representative of the particular fault type. A fault type with code $f_i = k$ is replaced with

SOM k . Each SOM k is an one-dimensional array of 100 neurons. Each of these neurons has an associated prototype vector with three coordinates. The distribution of prototype vectors (Fig. 13) in the input space was found using fuzzy c -means clustering and approximately 80% data from each fault type. Each prototype vector is cluster centre and representative of all data from its cluster.

Finally, in the third and the last stage in the training phase the structure of the SOM representatives is saved on hard disk for future use. In this way, heavy demanding calculations are performed off-line, during the training phase, which enables fast and efficient fault detection during the on-line phase.

On-line fault detection phase: During the initialisation stage of the on-line fault detection phase, the main processor reads the structure of the SOM representatives, saved on hard disk during the training phase, and stores it in the working memory for fast access.

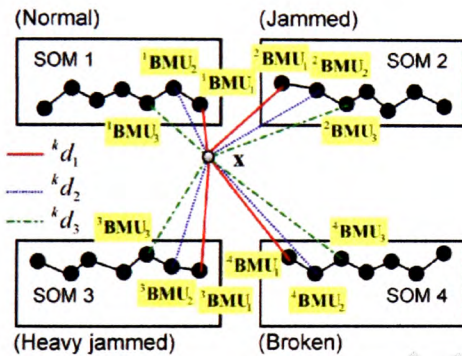


Fig. 14. On-line fault detection phase: position of the feature vector is determined relative to SOM representatives by finding three closest BMUs in each SOM.

After the initialisation is finished, the fault detection is performed by repeating the following steps at each program cycle:

Three closest codebook vectors (Best Matching Units—BMUs) from each map to feature vector x (which consists of pre-processed, actual measurements of n_d , n and I) are computed, together with corresponding distances (Fig. 14), where ${}^k\text{BMU}_j$ means j th BMU in SOM k , while ${}^k d_j$ means Euclidian distance between x and ${}^k\text{BMU}_j$, $k = \overline{1,4}$, $j = \overline{1,3}$. In the next step matrix $\mathbf{M} = [{}^k d_j]_{4 \times 3}$ is created. Minimum values of each column of \mathbf{M} are found and the indices of the minimum values are stored in row vector \mathbf{b} . For example, $\mathbf{b} = [1 \ 3 \ 2]$ means that the closest first BMU is in SOM 1, second in SOM 3 and third in SOM 2. In order to avoid false detection, the final decision about faulty state is accomplished using present and past vectors \mathbf{b} , which are stored in the buffer with size $s \times 3$. If all buffer elements have the same value, then the fault indicator f_i , output of the FDU, is set to this value. Finally, outputs of each FDU are integrated to form the total fault indicator vector \mathbf{f} :

$$\mathbf{f} = [f_1 \ f_2 \ f_3 \ f_4]^T \quad (43)$$

In a fault-free case all four indicators f_i are equal to 1, in accordance to the fault code table. It is assumed that at any time instance at most one horizontal thruster can be faulty i.e. at most one of the indicators f_i can be different than 1. Simultaneous faults in multiple thrusters are very rare in practical operations and in these cases unavoidable loss in controllability of some DOF will occur. Only faults in a single thruster are considered in this paper.

6.2.4. Evaluation

As stated before, a large data set was acquired during test trials and only a part of this data was used for

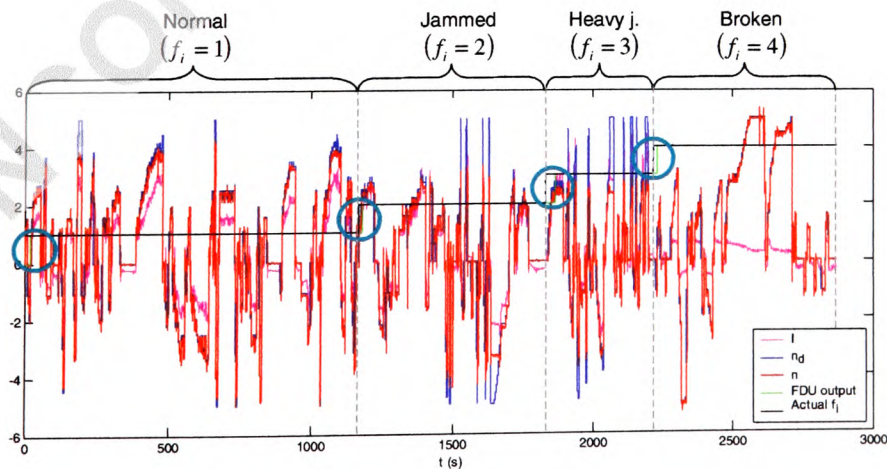


Fig. 15. Evaluation of the FDU.

training. The capability of the proposed FDU to detect external faults is evaluated using the entire data set. Only signals n_d , n and I are presented as inputs to the FDU, which must estimate the state of the thruster using only these inputs. Fig. 15 displays actual fault code and FDU output, together with training data. It can be seen that the FDU identifies the new thruster state correctly in a short time after the change in state, as highlighted in Fig. 15. This delay is unavoidable, because the thruster must spend some time in a faulty state before the fault can be identified. Delay is proportional to the buffer size s . A conservative value $s = 25$ was used in Fig. 15, in order to prevent a wrong detection. It is expected that the buffer size and delay will be reduced in future similar experiments with the FALCON, due to advanced signal conditioning and better quality of measured signals.

6.3. Fault accommodation subsystem

6.3.1. Introduction

An ROV pilot uses the HCU (Fig. 8) to generate the input command vector τ_d . The FDS finds the total fault indicator vector \mathbf{f} with information about healthy state of each thruster. The FAS uses these two vectors and relationships in the fault code table to solve the control allocation problem separately for motion in horizontal and vertical plane.

The solution method adopted in the FAS relies on the fact that explicit solution to the unconstrained control allocation problem:

$$\min_{\mathbf{u}} \|\mathbf{W}\mathbf{u}\|_2 \quad (44)$$

subject to

$$\mathbf{B}\mathbf{u} = \tau_d \quad (45)$$

is given by (Fossen, 1995)

$$\mathbf{u} = \mathbf{B}_w^+ \tau_d = (\mathbf{W}^{-1} \mathbf{B}^T (\mathbf{B} \mathbf{W}^{-1} \mathbf{B}^T)^{-1}) \tau_d, \quad (46)$$

where \mathbf{B}_w^+ is weighted pseudo-inverse of \mathbf{B} . Solution (46) is obtained using Lagrange multipliers.

Horizontal plane: For motion in the horizontal plane the weighted pseudo-inverse matrix $\mathbf{B}_{w_u}^{+HT}$ is given by

$$\text{FALCON: } \mathbf{B}_{w_u}^{+HT} = \frac{1}{\sum_{i=1}^4 w_i^{HT}} \times \begin{bmatrix} 2(w_3^{HT} + w_4^{HT}) & 2(w_2^{HT} + w_4^{HT}) & 2(w_2^{HT} + w_4^{HT}) \\ 2(w_3^{HT} + w_4^{HT}) & -2(w_1^{HT} + w_4^{HT}) & -2(w_1^{HT} + w_3^{HT}) \\ 2(w_1^{HT} + w_2^{HT}) & 2(w_1^{HT} + w_4^{HT}) & -2(w_2^{HT} + w_4^{HT}) \\ 2(w_1^{HT} + w_2^{HT}) & -2(w_2^{HT} + w_3^{HT}) & 2(w_1^{HT} + w_3^{HT}) \end{bmatrix}, \quad (47)$$

$$\text{URIS: } \mathbf{B}_{w_u}^{+HT} = \frac{1}{\sum_{i=1}^4 w_i^{HT}} \times \begin{bmatrix} (2w_2^{HT} + w_3^{HT} + w_4^{HT}) & (w_3^{HT} - w_4^{HT}) & 2(w_3^{HT} + w_4^{HT}) \\ (2w_1^{HT} + w_3^{HT} + w_4^{HT}) & (w_4^{HT} - w_3^{HT}) & -2(w_3^{HT} + w_4^{HT}) \\ (w_1^{HT} - w_2^{HT}) & (w_1^{HT} + w_2^{HT} + 2w_4^{HT}) & 2(w_1^{HT} + w_2^{HT}) \\ (w_2^{HT} - w_1^{HT}) & (w_1^{HT} + w_2^{HT} + 2w_3^{HT}) & -2(w_1^{HT} + w_2^{HT}) \end{bmatrix}. \quad (48)$$

These expressions are obtained by combining (25), (26) and (46).

The weighted pseudo-inverse solution of the unconstrained control problem for motion in the horizontal plane is given by

$$\mathbf{u}^{HT} = \mathbf{B}_{w_u}^{+HT} \tau_d^{HT}. \quad (49)$$

Vertical plane: Combining (40), (41) and (46), the weighted pseudo-inverse matrix $\mathbf{B}_{w_v}^{+VT}$ for motion in the vertical plane becomes scalar

$$\text{FALCON: } \mathbf{B}_{w_v}^{+VT} = 1. \quad (50)$$

The weighted pseudo-inverse solution of the unconstrained control problem for motion in the vertical plane is given by

$$\mathbf{u}^{VT} = \mathbf{B}_{w_v}^{+VT} \tau_d^{VT} = \tau_d^{VT}. \quad (51)$$

6.3.2. Feasibility of the weighted pseudo-inverse solution

The input command vector $\tau_d^{HT} = [\tau_X \ \tau_Y \ \tau_N]^T$ for the motion in the horizontal plane, generated by the HCU, belongs to the virtual control space Φ_v^{HT} , the unit cube in \mathcal{R}^3 :

$$\Phi_v^{HT} = \{\tau^{HT} \in \mathcal{R}^3 \mid \|\tau^{HT}\|_\infty \leq 1\} \subset \mathcal{R}^3. \quad (52)$$

For the constrained control allocation problem, where the constraint $\mathbf{u}^{HT} \in Q^{HT}$ is required to be satisfied, solution (49) may become unfeasible, depending on the position of τ_d^{HT} inside Φ_v^{HT} . The virtual control space Φ_v^{HT} can be partitioned into characteristic regions, as indicated in Fig. 16. The two characteristic regions inside Φ_v^{HT} are Φ_p^{HT} (feasible region for pseudo-inverse) and $\Phi^{HT} \supset \Phi_p^{HT}$ (attainable command set). The shape of Φ^{HT} is found from condition $\mathbf{u}^{HT} = \mathbf{B}_{w_u}^{+HT} \tau_d^{HT} \in Q^{HT}$. It should be emphasised that, for the general constrained control allocation problem, there is an infinite number of exact solutions for $\tau_d^{HT} \in \Phi_p^{HT}$, while no exact solution exists for $\tau_d^{HT} \in \Phi_v^{HT} \setminus \Phi_p^{HT}$. The weighted pseudo-inverse is able to find the exact feasible solution of the control allocation problem, optimal in the l_2 sense, only if $\tau_d^{HT} \in \Phi_p^{HT}$. Otherwise, for $\tau_d^{HT} \in \Phi_v^{HT} \setminus \Phi_p^{HT}$ the solution obtained by pseudo-inverse is unfeasible and cannot be directly applied to the thrusters. An unfeasible solution means that some components (controls) of the control vector \mathbf{u} violate constraints. In this case the unfeasible pseudo-inverse solution is approximated in order to get

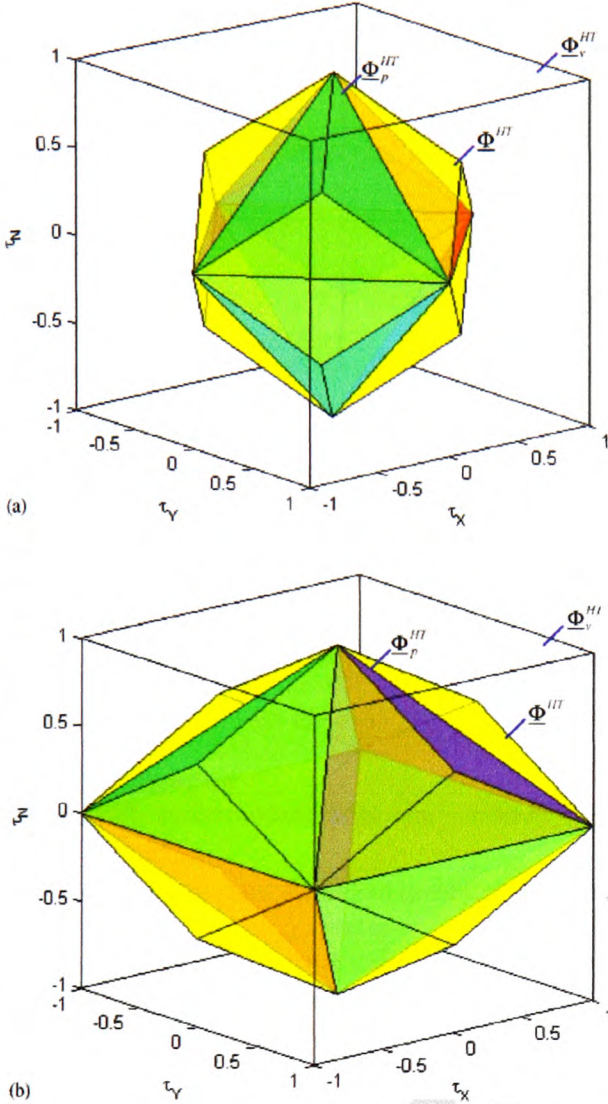


Fig. 16. Partitions of the virtual control space for motion in the horizontal plane: (a) FALCON. (b) URS.

a feasible approximation. However, some other methods, such as direct control allocation or fixed-point method, are able to find the exact, unique, feasible solution for all attainable $\tau_d^{HT} \in \Phi^{HT}$.

6.3.3. Approximation of unfeasible solution

In the literature, two common approximations are used to the approximate unfeasible pseudo-inverse solution \underline{u}^{HT} : T -approximation (truncation) and S -approximation (scaling). In case of T -approximation, the approximation $\underline{u}_t^{*HT} \in \partial(Q^{HT})$ is obtained from \underline{u}^{HT} by truncating (clipping) all controls which exceed their control constraints. In contrast, the S -approximation \underline{u}_s^{*HT} is obtained by scaling unfeasible solution \underline{u}^{HT} to the boundary $\underline{u}_s^{*HT} \in \partial(Q^{HT})$ by factor f :

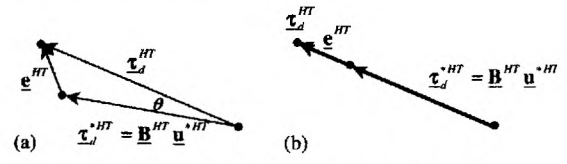


Fig. 17. Approximation error.

$$\underline{u}_s^{*HT} = f \underline{u}^{HT}, \quad (53)$$

where

$$f = \min \left(\frac{1}{\max_{i \in \{1,2,3,4\}} \left| \frac{\tau_{di}^{HT}}{\tau_{di}^{*HT}} \right|}, 1 \right). \quad (54)$$

The approximation error is defined as $\underline{e}^{HT} = \tau_d^{HT} - \tau_{dt}^{*HT}$, where $\tau_{dt}^{*HT} = \underline{B}^{HT} \underline{u}_t^{*HT}$ (see Fig. 17). In order to be able to compare different approximations, two scalar errors are introduced: *direction error* $\theta = \arccos \frac{\tau_d^{HT} \cdot \tau_{dt}^{*HT}}{\|\tau_d^{HT}\|_2 \|\tau_{dt}^{*HT}\|_2}$ and *magnitude error* $\|\underline{e}^{HT}\|_2 = \|\tau_d^{HT} - \tau_{dt}^{*HT}\|_2$. The direction error represents the angle between τ_d^{HT} and τ_{dt}^{*HT} , while the magnitude error represents the module of the approximation error vector \underline{e}^{HT} (Fig. 17). In the case when $\theta = 0$, the approximation τ_{dt}^{*HT} preserves the direction of the original vector τ_d^{HT} .

6.3.4. Example (Approximation of unfeasible pseudo-inverse solution)

Let $\tau_d^{HT} = [0.70 \ 0.20 \ 0.25]^T$ for X-shaped thruster configuration in fault-free case (Fig. 18). It can be seen that $\tau_d^{HT} \in \Phi^{HT} \setminus \Phi_p^{HT}$. The weighted pseudo-inverse solution $\underline{u}^{HT} = [1.15 \ 0.25 \ 0.65 \ 0.75]^T$ is obtained combining (36) and (49). This solution is unfeasible, since $u_1^{HT} > 1$.

The T -approximation is given by $\underline{u}_t^{*HT} = [1.00 \ 0.25 \ 0.65 \ 0.75]$ and this is feasible solution that lies on the boundary $\underline{u}_t^{*HT} \in \partial(Q^{HT})$. This solution leads to approximate $\tau_{dt}^{*HT} = \underline{B}^{HT} \underline{u}_t^{*HT} = [0.6625 \ 0.1625 \ 0.2125]^T$ that lies outside Φ_p^{HT} (see Fig. 18). The magnitude error is $\|\underline{e}_t\|_2 = 0.0650$ and direction error is $\theta = 2.6362^\circ$.

The first step for the S -approximation is to use (54) to find the scaling factor $f = 0.8696$. Then the approximation $\underline{u}_s^{*HT} = [1.0000 \ 0.2174 \ 0.5652 \ 0.6522] \in \partial(Q^{HT})$ is found using (53). This solution leads to an approximation $\tau_{ds}^{*HT} = \underline{B}^{HT} \underline{u}_s^{*HT} = [0.6087 \ 0.1739 \ 0.2174]^T$ that lies on boundary $\partial(\Phi_p^{HT})$ and represents intersection of τ_d^{HT} and $\partial(\Phi_p^{HT})$ (see Fig. 18). The magnitude error is $\|\underline{e}_s\|_2 = 0.1004$ and direction error is $\theta = 0$, i.e. the S -approximation τ_{ds}^{*HT} preserves the direction of the original vector τ_d^{HT} . Comparing errors it can be seen that T -approximation leads to feasible approximation

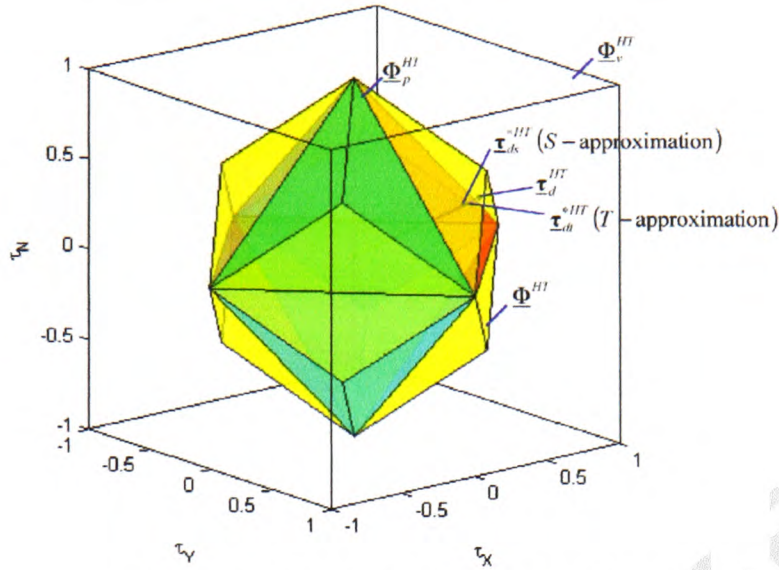


Fig. 18. Approximation of unfeasible pseudo-inverse solution.

with lower magnitude error, while S -approximation finds a feasible approximation that has the same direction as τ_d^{HT} , i.e. with direction error equal to zero.

6.3.5. FAS algorithm

The fault accommodation process involves these steps:

1. FDS detects thrusters' states and generates total fault indicator vector $\mathbf{f} =$

$$[\underbrace{f_1^{HT} f_2^{HT} f_3^{HT} f_4^{HT}}_{\mathbf{f}^{HT}} \underbrace{f_1^{VT}}_{\mathbf{f}^{VT}}]^T.$$

2. Using Table 6, "Co-ordinator" (Fig. 8) transforms vector \mathbf{f} into the association vector $\mathbf{s} =$

$$[\underbrace{s_1^{HT} s_2^{HT} s_3^{HT} s_4^{HT}}_{\mathbf{s}^{HT}} \underbrace{s_1^{VT}}_{\mathbf{s}^{VT}}]^T, \text{ which relates thruster states and saturation bounds of each thruster.}$$

This transformation is called action B. For example, $\mathbf{f} = [2 \ 1 \ 1 \ 1 \ 1]^T$ means "Jammed propeller" state in 1HT and fault-free states in others. This vector is mapped to the association vector $\mathbf{s} = [0.75 \ 1 \ 1 \ 1 \ 1]^T$ i.e. 1HT should operate at 75%, while the others should work at full power (see Table 6).

3. If a faulty thruster is horizontal, then "Co-ordinator" updates its weight in the weighting matrix \mathbf{W}_u^{HT} using (38) and (39), and set the other weights to unity. If a faulty thruster is vertical, then there is no need for weight update. The procedure of weighting update is called action A. New weights for the given example are $w_1^{HT} = 5/3$, $w_2^{HT} = w_3^{HT} = w_4^{HT} = w_1^{VT} = 1$.

4. "Pseudo-inverse" and "Approximation" find the new, feasible control vector for the given input and actual state of thrusters. "Pseudo-inverse" uses (47)–(49) to find the pseudo-inverse solution for motion in

the horizontal plane, and (51)—for vertical plane. If the pseudo-inverse solution is unfeasible, "Approximation" block makes it feasible utilising T - or S -approximation.

5. Feasible control vectors for motion in the horizontal and vertical plane are merged, transformed into the form determined by the TCUs and used to actuate thrusters.

In the case of a fault in a single thruster, the feasible region for pseudo-inverse Φ_p^{HT} and attainable command set Φ_v^{HT} shrink, as shown in Fig. 19. In particular, Fig. 19a and c display the partitions of the virtual control space Φ_v^{HT} (regions Φ_p^{HT} and Φ_v^{HT}) for the case of a partial fault ("Heavy-jammed propeller") in 2HT . In this case, 2HT is penalised by increasing its weight ($w_2^{HT} = 3$, Eqs. (38) and (39)) and corresponding saturation bound s_2^{HT} is changed to $s_2^{HT} = 0.5$. This guarantees equality between desired and actual vectors (τ_d^{HT} and τ^{HT}) inside Φ_p^{HT} shown in Fig. 19a and c and, at the same time, the value of $|u_2^{HT}|$ will never be greater than 0.5. Similarly, Fig. 19b and d show the feasible region for the case of a total breakdown in 2HT . In this case, 2HT is switched off and $w_2^{HT} \rightarrow \infty$. In this way, the redundancy is eliminated from the system of equation $\mathbf{B}^{HT} \mathbf{u}^{HT} = \tau^{HT}$, which can now be solved in a standard way. From this reason Φ_p^{HT} and Φ_v^{HT} coincide in Fig. 19b and d. Only three remaining thrusters are capable to track the desired vector τ_d^{HT} without any error inside Φ_p^{HT} shown in Fig. 19b and d. This means that, in the case of a total failure in a single thruster, mission can be continued and the control allocation will be successful if the desired vector τ_d^{HT} stays inside Φ_p^{HT} for particular fault case.

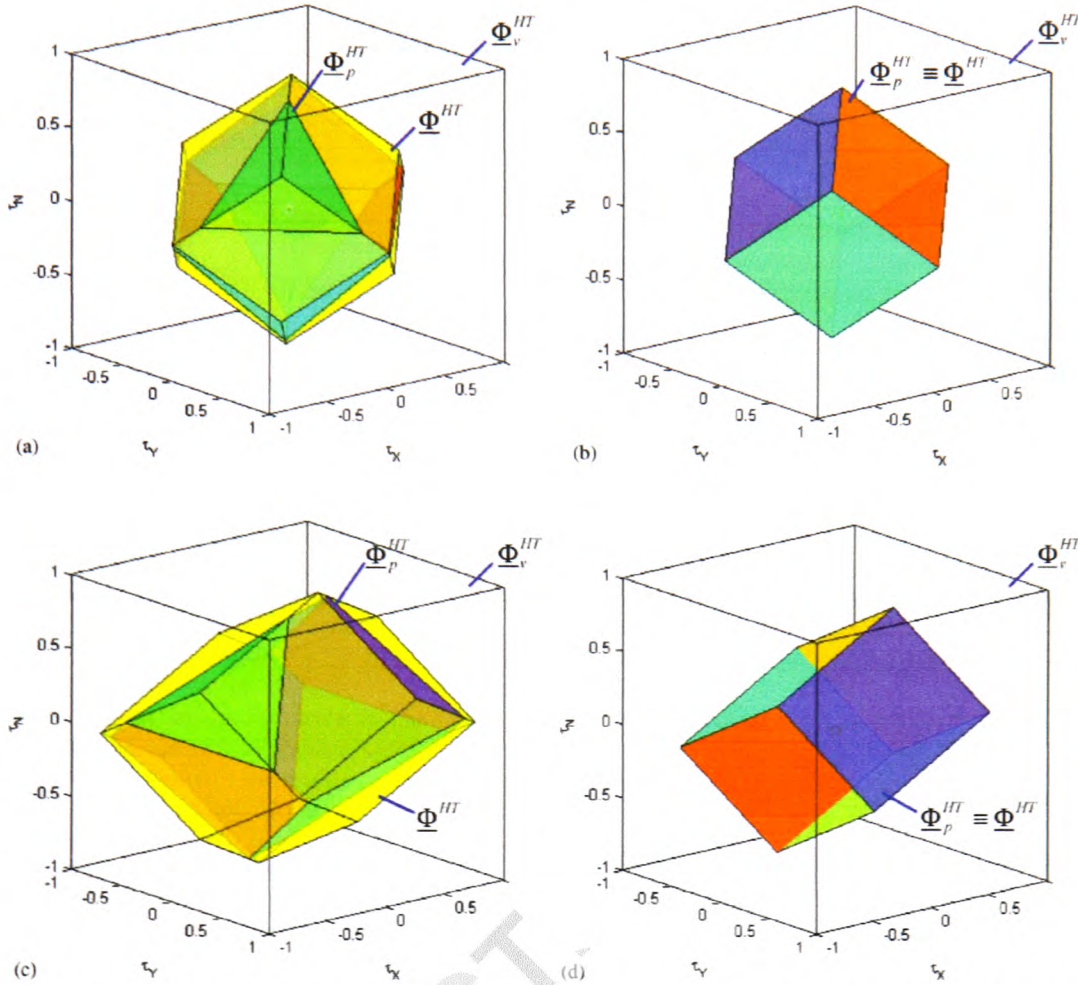


Fig. 19. Partitions of the virtual control space for different faulty situations: (a) FALCON: "Heavy j. propeller" ($s_2^{HT} = 0.5$). (b) FALCON: "Broken propeller" ($s_2^{HT} = 0$). (c) URIS: "Heavy j. propeller" ($s_2^{HT} = 0.5$). (d) URIS: "Broken propeller" ($s_2^{HT} = 0$).

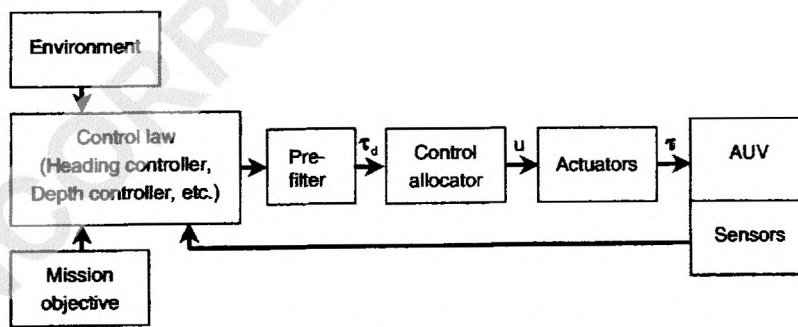


Fig. 20. Typical closed-loop AUV control structure.

7. Potential applications

In standard open-loop ROV control structure (Fig. 3a), vector τ_d is generated by an ROV pilot using HCU. In contrast, Fig. 20 displays a typical, closed-loop AUV control structure. In order to achieve mission objectives, the control law uses actual knowledge about the

environment and sensor measurements to find a reference inputs for a set of controllers (heading controller, depth controller, etc.). The outputs of the controllers are integrated into a vector that is similar to the output of the HCU in Fig. 3a. The control allocator performs in exactly the same way as in the previous case. Hence, from the control allocator point of view, it does

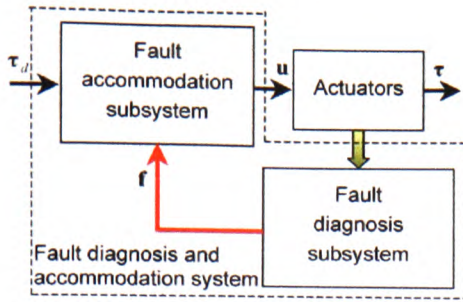


Fig. 21. Relationship between the FDAS and a typical control structure for open-frame underwater vehicles (ROVs or AUVs).

not matter how the virtual control input τ_d is generated (by the ROV pilot or the control law). The control allocation algorithm is the same for both structures. The task of the control allocator in both cases is to determine appropriate control settings for individual actuators, which produce the desired set of forces and moments.

It is useful at this point to consider the role (potential application) of the FDAS in the control structures shown in Figs. 3a and 20. As indicated in Fig. 21, the FDAS performs the control allocation task, but this primary task is enhanced with the ability to monitor the state of the thrusters and, if necessary, perform automatic reconfiguration, i.e. redistribution of propulsion forces among the operable thrusters, as explained in previous sections. In this way, using control allocation the actuator selection task is separated from the regulation task in the control design. That is, the control law, which maps the desired response to a set of commands (objectives), is not dependent on the design of the control allocation system, which relates these commands with settings and positions of individual actuators.

Treating control allocation independently of the control law is convenient because of the following:

- *Actuator constraints can be taken into account:* In real applications actuator saturation always exists. If one actuator saturates, some of methods for control allocation are able to redistribute control energy among other available actuators to compensate for the inability of a saturated thruster to produce its nominal control effect. In this way, available control resources are fully exploited before the closed-loop performance is degraded.
- *Reconfiguration can be performed:* If the effectiveness of the actuators change over time, or in the case of an actuator total or partial fault, reconfiguration i.e. redistribution of control energy among a set of available actuators can be performed, without having to redesign the control law.

8. Concluding remarks

A novel thruster fault detection and accommodation system for overactuated open-frame underwater vehicles has been presented. The FDAS includes two subsystems: FDS and FAS. The FDS is a hybrid, on-line, model-free approach, based on integration of SOM and fuzzy clustering methods. In the training phase, the FDS uses data obtained during test trial to find SOM representatives for each fault type. In the detection phase, the FDS makes decision about fault type by comparing the position of feature vector relative to these maps. The results demonstrate efficiency and robustness of the FDS. The FAS uses the output of the FDS to accommodate faults and perform reconfiguration by updating weights used in the optimisation criteria and thruster velocity saturation bounds. This paper demonstrates that, for open-frame underwater vehicles with four horizontal thrusters, in the case of a partial or total fault in a horizontal thruster, it is possible to reconfigure the control system in an optimal manner, in order to maintain a high level of manoeuvrability and mission completion.

Important contributions of the paper are:

- normalisation of the control allocation problem, which leads to easier understanding and visualisation of the problem,
- design of enhanced control allocator, able to reallocate control energy among operable thrusters in an optimal way and continue the mission in the presence of thruster velocity saturation and a fault in a single thruster.
- visualisation of the feasible region for pseudo-inverse, which provides a framework to visualise thruster velocity saturation bounds and to incorporate knowledge about saturation margins into control law. This is a very important enhancement, which improves existing controllers and provides better performance of the control system in real-world applications.

The most important issue is that every control law should generate input vectors that lie inside the feasible region for pseudo-inverse. Knowledge about position of the input vector inside the feasible region and its distance from the saturation bounds can be used to improve existing control laws and to avoid discrepancy between predicted and real behaviour of the vehicle caused by thruster velocity saturation. Currently, most ROVs' energy is provided, via the umbilical cable, from power supplies located on the mother ship. AUVs and some ROVs utilise batteries for the energy supply and excessive power consumption adversely affects available time on mission. The proposed FDAS provides an optimal solution for the distribution of propulsion

forces for fault-free case and fault in a single thruster, that minimises a control energy cost function. The importance of an optimal solution can be summarised as minimum control energy means maximum operational time from battery, and minimum usage of thruster means maximisation of thruster life.

However, the FDAS is able to find the exact solution of the control allocation problem only on a subset of attainable command set, i.e. in small zones around the feasible region, where the pseudo-inverse solution is unfeasible, still it is possible to find the exact solution by applying other techniques, such as direct control allocation or fixed-point methods. An extension of the FDAS to cover these zones is described in (Omerdic, Roberts, & Toal, 2004).

Underwater vehicles, which use other types of actuators (like control surfaces) beside thrusters, are not covered in this paper, but the same fault diagnosis and accommodation concept can be extended to cover this class of underwater vehicles by including new actuators into control architecture and reformulating the control allocation problem.

The proposed FDAS is a part of the low-level control layer and its modular design enables easy integration into existing control laws. Future work will include implementation of the proposed approach and its integration into existing control architecture for both vehicles. Special attention will be devoted to the design of a universal controller, robust to a partial/total fault in a single thruster. An important part of this work will be the integration of feasible region with real-time video presented to the ROV pilot from the on-board camera.

References

- Alessandri, A., Caccia, M., & Veruggio, G. (1999). Fault detection of actuator faults in unmanned underwater vehicles. *Control Engineering Practice*, 7, 357–368.
- Blanke, M. (2001). Enhanced maritime safety through diagnosis and fault tolerant control. *IFAC conference on control applications and marine systems, CAMS 2001*, Glasgow (Invited plenary).
- Blanke, M., Staroswiecki, M., & Eva Wu, N. (2001). Concepts and methods in fault-tolerant control. *Proceedings of the American control conference*, Washington (Invited tutorial).
- Bono, R., Bruzzone, G., Bruzzone, G., & Caccia, M. (1999). ROV actuator fault diagnosis through servo-amplifiers' monitoring: An operational experience. *MTS/IEEE Oceans' 99*, Vol. 3 (pp. 1318–1324), Seattle, USA.
- Bordignon, K. A. (1996). *Constrained control allocation for systems with redundant control effectors*. Ph.D. thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.
- Burken, J., Lu, P., & Wu, Z. (1999). *Reconfigurable flight control designs with applications to the X-33 vehicle*. NASA/TM-1999-206582.
- Burken, J., Lu, P., Wu, Z., & Bahm, C. (2001). Two reconfigurable flight-control design methods: Robust servomechanism and control allocation. *Journal of Guidance, Control and Dynamics*, 24(3), 482–493.
- Cuadrado, A. A., Díaz, I., Díez, A. B., Obeso, F., & González, J. A. (2001). Fuzzy inference maps for condition monitoring with self-organising maps. *International conference in fuzzy logic and technology* (pp. 55–58), Leicester, UK.
- Doty, K. L., Melchiorri, C., & Bonivento, C. (1993). A theory of generalised inverses applied to robotics. *The International Journal of Robotics Research*, 12(1), 1–19.
- Durham, W. C. (1993). Constrained control allocation. *Journal of Guidance, Control, and Dynamics*, 16(4), 717–725.
- Durham, W. C. (1994). Constrained control allocation: Three moment problem. *Journal of Guidance, Control, and Dynamics*, 17(2), 330–336.
- Enns, D. (1998). Control allocation approaches. In *AIAA guidance, navigation, and control conference and exhibit* (pp. 98–108), Boston.
- Fossen, T. I. (1995). *Guidance and control of ocean vehicles*. Chichester: Wiley.
- Fossen, T. I. (2002). *Marine control systems*. Trondheim, Norway: Marine Cybernetics AS.
- Fossen, T. I., & Blanke, M. (2000). Nonlinear output feedback control of underwater vehicle propellers using feedback from estimated axial flow velocity. *IEEE Journal of Oceanic Engineering*, 25(2), 241–255.
- Ikeda, Y., & Hood, M. (2000). An application of l_1 optimisation to control allocation. In *AIAA guidance, navigation and control conference and exhibit*, Denver.
- Isermann, R., & Ballé, P. (1997). Trends in the application of model-based fault detection and diagnosis of technical process. *Control Engineering Practice*, 5(5), 709–719.
- Johansen, T. A., Fossen, T. I., & Tøndel, P. (2002). Efficient optimal constrained control allocation via multi-parametric programming. *AIAA Journal of Guidance, Control and Dynamics*, submitted for publication.
- Lindfors, I. (1993). Thrust allocation method for the dynamic positioning system. In *Proceedings of the 10th international ship control systems symposium (SCSS' 93)* (pp. 3.93–3.106), Ottawa.
- Omerdic, E., & Roberts, G. N. (2003). Thruster fault accommodation for underwater vehicles. *First IFAC workshop on guidance and control of underwater vehicles GCUV' 03* (pp. 221–226), 9–11 April 2003, Newport, South Wales, UK.
- Omerdic, E., Roberts, G. N., & Ridao, P. (2003). Fault detection and accommodation for ROVs. *Sixth IFAC conference on manoeuvring and control of marine craft (MCMC 2003)*, Girona, Spain.
- Omerdic, E., Roberts, G. N., & Toal, D. (2004). Extension of feasible region of control allocation for open-frame underwater vehicles. *IFAC conference on control applications in marine systems (CAMS 2004)*, Ancona, Italy, submitted for publication.
- Podder, T. K., Antonelli, G., & Sarkar, N. (2000). Fault tolerant control of an autonomous underwater vehicle under thruster redundancy: Simulations and experiments. *Proceeding of the 2000 IEEE international conference on robotics & automation*, San Francisco, USA.
- Podder, T. K., & Sarkar, N. (1999). Fault tolerant decomposition of thruster forces of an autonomous underwater vehicle. *Proceeding of the 1999 IEEE international conference on robotics & automation*, Detroit, USA.
- Snell, S. A., Enns, D. F., & Garrard, W. L. (1992). Nonlinear inversion flight control for a supermanoeuvrable aircraft. *Journal of Guidance, Control and Dynamics*, 15(4), 976–984.
- Takai, M., & Ura, T. (1999). Development of a system to diagnose autonomous underwater vehicle. *International Journal of Systems Science*, 30(9), 981–988.
- Tøndel, P., Johansen, T. A., & Bemporad, A. (2001). An algorithm for multi-parameter quadratic programming and explicit MPC solutions. In *Proceedings of the IEEE conference on decision and control (CDC' 01)* (pp. TuP11-4).

- 1 Tøndel, P., Johansen, T. A., & Bemporad, A. (2003). Evaluation of
piecewise affine control via binary search tree. *Automatika*, 39,
743–749. 9
- 3 Virnig, J. C., & Bodden, D. S. (1994). Multivariable control allocation
and control law conditioning when control effectors limit. In *AIAA*
5 *guidance, navigation and control conference and exhibit*, Scottsdale.
- 7 Yang, K. C., Yuh, J., & Choi, S. K. (1998). Experimental study of
fault-tolerant system design for underwater robots. *Proceeding of*
the 1998 IEEE international conference on robotics & automation,
Leuven, Belgium. 11
- Yang, K. C. H., Yuh, J., & Choi, S. K. (1999). Fault-tolerant system
design of an autonomous underwater vehicle—ODIN: An experi-
mental study. *International Journal of Systems Science*, 30(9), 1011–
1019.

UNCORRECTED PROOF