Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3523

Analysis and Implementation of Control Strategies for Multibody Robotic Systems

Eugen Nosov

Course of Study: Informatik

Examiner: Prof. Dr. rer. nat. habil. Paul Levi

Supervisor: Dr.-Ing. Eugen Meister

Commenced: Juli 11, 2013

Completed: Januar 10, 2014

CR-Classification: I.2.0, I.2.8, D.1.5, D.2.5

Abstract

Multibody robotic organisms are reconfigurable dynamic systems. Thus the shape and the structure of a modular robot is not constant. Depending on the changing environment or on the current task of the robot its structure can be rearranged accordingly. Hence, that would also change the dynamic model of the robotic organism. Thus the kinematics and dynamics of the current modular structure of the robot, and with it also an appropriate control strategy, can not be calculated in advance.

Contents

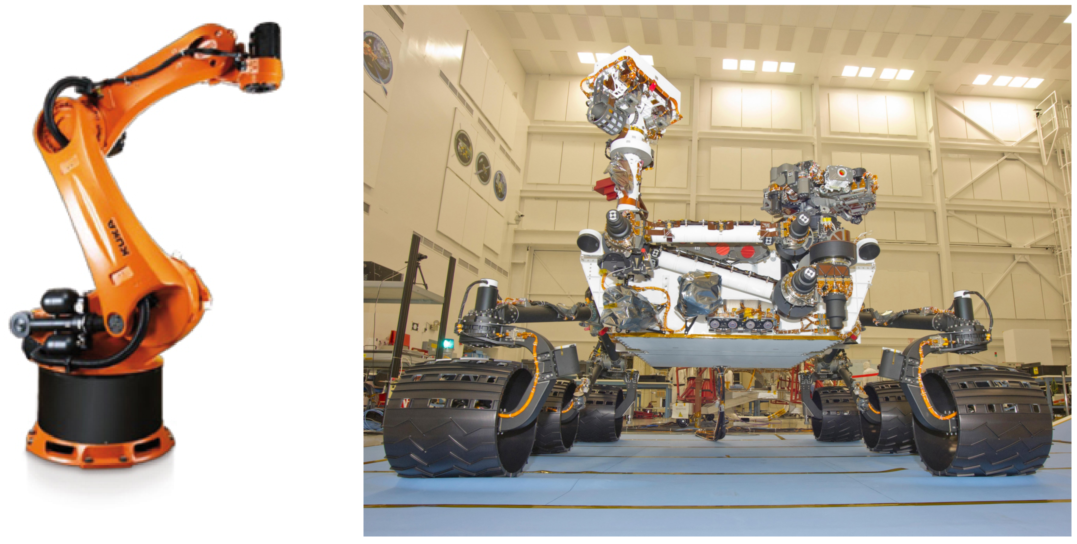
List of Figures

List of Tables

List of Listings

1 Introduction

Powerful microelectronic components, smaller drive mechanisms and high-performance storage batteries allows the building of small and simple but also robust and efficient robotic systems. In the last decades the modularity in the robotics has increased in importance. Due to the possibility of changing the own structure modular systems provide a high degree of flexibility and opens new fields of application in the robotics. For the most robots which have a constant shape the dynamic models of their systems are obtained manually in advance. Knowledge of the appropriate dynamic model allows the derivation of a control method that match the desired behavior to the highest degree. Such robots are developed for a certain purposes only. Figure 1.1 shows some examples of such robotic systems.



(a) Industrial robot [KUK12] **(b)** Mars rover Curiosity [NAS12]

Figure 1.1: Robotic modules developed in the projects SYMBRION and REPLICATOR.

However the modular self-reconfigurable robotic systems may change their shape and structure accordingly to their tasks or the environment properties. An example for modular self-reconfigurable robotic systems can be seen in the Figure 1.2. Hence, the

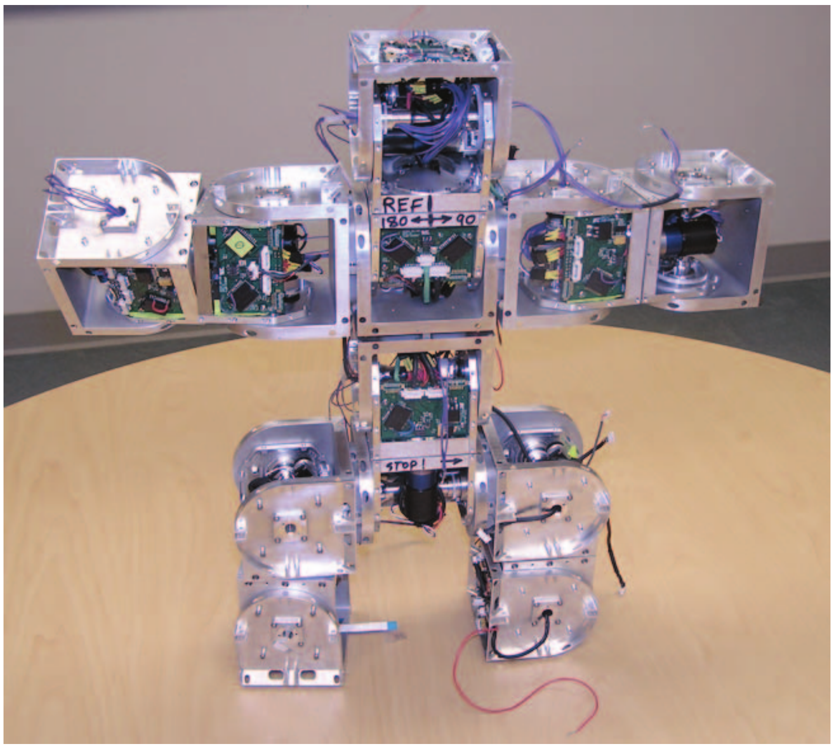


Figure 1.2: SuperBot [YSS+07].

dynamic model of the modular robot can not be considered as constant. Therewith the kinematics and dynamics of the multibody system must be computed either for each possible configuration in advance or each time the structure is changed. Due to numerous possibilities in the combination of the particular modules it is rarely realizable to prepare a dynamic model for each configuration. Calculating the models each time by hand makes the robot highly dependent on the supervision. A reasonable approach would be to let the robot acquire its own dynamic model and therewith also the appropriate control mechanism autonomously. A method for automatic model generation of the modular self-reconfigurable robotic system have been already introduced by Chen and Yang [CY98].

This thesis describes several mechanisms of the classical control theory and control strategies which can be implemented for use by the robots itself.

1.1 Goals

The goals for this thesis are to analyze the control strategies for modular self-reconfigurable robots and furthermore to implement the chosen approaches on real robots. In order to reach this goal a software framework have to be developed, which on the one hand is capable to deal with modeling techniques based on geometrical approach, and on the other hand it should involve control design implementation methods.

1.2 Previous Work

The implementation of the control mechanisms which are described in this thesis extends the functionality of the existing C++ software framework that was proposed in the previous work [Nos13]. The control implementation uses the functionality of the existing framework for generation of the dynamic model of the multibody system. The model generation in their case is based on the approach proposed by Chen and Yang. The derivation of the appropriate control strategies is partly based on the work of Meister and Gutenkunst [MGL13]. They propose the implementation of several control strategies in a software framework based on MATLAB simulations.

1.3 Organization of this Thesis

This thesis is organized as follows:

Chapter 1 makes a brief introduction in short insight in current module-based robotic systems.

Chapter 2 provides a theoretical background needed for derivation of closed-form motion equations.

Chapter 3 describes several control strategies and gives appropriate examples.

Chapter 4 explains the implementation of the proposed control strategies.

Chapter 5 demonstrates several practical results that have been performed on the real modular robotic system.

Chapter 6 finalizes the thesis and gives a short insight in future works.

2 Theoretical Background

This chapter gives a brief introduction in theoretical basics needed for derivation of the closed-form motion equations of dynamic model. Each motion of a solid state body can be divided in two components: translational part and rotational part. This is the principle that was used by Sir Robert S. Ball for developing the *screw theory*. The theory of screws is a powerful concept for description of kinematics and dynamics of multi-body-systems.

2.1 Rigid Body Motion

The basis for dynamics and kinematics in the robotics is built with the concepts of rigid body motion (Definition 2.1.1). The rigid body motion is the central component of the screw theory.

Definition 2.1.1 (Rigid Motion)
*A **rigid motion** of an object is a continuous movement of the particles in the object such that the distance between any two particles remains fixed at all times [MSZ94].*

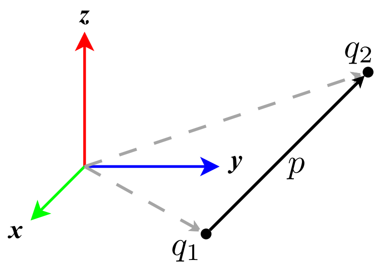
The definition of the rigid body motion includes two major characteristics:

1. The distance between two arbitrary points of the transformed rigid body is preserved.
2. The cross product of two arbitrary vectors constructed by the points of rigid body is preserved.

This properties mean that each rigid body motion transforms an orthonormal coordinate system again to an orthonormal coordinate system.

2.2 Lie Groups

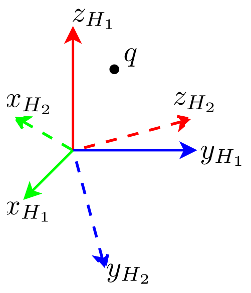
As mentioned above each rigid body motion consists of translational and rotational part. The translation part can be described as a movement along the vector connecting the start and the end positions of the motion:



$$q_2 = q_1 + p. \tag{2.1}$$

With $q_1 \in \mathbb{R}^3$ and $q_2 \in \mathbb{R}^3$ as start and end positions respectively and $p \in \mathbb{R}^3$ as translation vector.

For rotation it is important to distinguish between the global coordinate frame and the coordinate frame belonging to rotated body



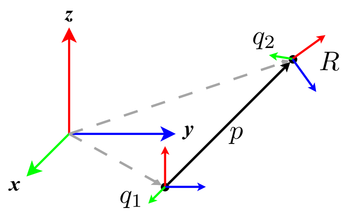
$$q_{H_2} = Rq_{H_1}. \tag{2.2}$$

The basis vectors of the rotated frame H_2 described in the global coordinates H_1, build the Rotation matrix $R = [x_{H_2}, y_{H_2}, z_{H_2}]$. In this case q_{H_2} are the coordinates of the point q described in the body coordinate frame H_2 and q_{H_1} denotes the same point but relative to the global coordinate frame H_1. Thus the rotation matrix $R \in \mathbb{R}^{3\times3}$ represents the function that rotates a body relative to the global coordinate system. Since rotation matrices build an orthonormal basis and their column vectors are ordered in right hand manner they belongs to the *special orthogonal group* $SO(3)$:

$$SO(3) = \{R \in \mathbb{R}^{3\times3} : RR^T = I,\ \det R = +1\}$$

Because of non-singularity rotation matrices build also a Lie group. More detailed information about the rotation matrices and their properties can be found in [MSZ94].

Now the combination of the both motion parts produce a mapping that describes a whole rigid body motion



$$q_2 = Rq_1 + p \tag{2.3}$$

This mapping can be represented as a 4×4 matrix that includes both motion parts, translation and rotation. Such form is also called *homogeneous representation* of rigid body motion

$$g = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}. \tag{2.4}$$

Homogeneous matrices belong to *special Euclidean group* $SE(3)$. $SE(3)$ is a further important Lie group.

$$SE(3) = \{(p, R) : p \in \mathbb{R}^3,\ R \in SO(3)\}$$

Both Lie groups are essential for kinematic and dynamic calculation in robotics. [Sel05, p. 4ff] provides further explanation of Lie groups used in robotics.

2.3 Lie Algebras

A further important concept in calculation of kinematic and dynamics are Lie algebras. Lie algebra is defined together with the bilinear map called Lie bracket, which satisfy following conditions:

- Skew-symmetry: $[a, b] = -[b, a]$,
- Jacobi identity: $[a, [b, c]] + [c, [a, b]] + [b, [c, a]] = 0$.

If elements are square matrices, the Lie bracket is a matrix commutator $[A, B] = AB - BA$. [MG12] Lie algebra of $SO(3)$, denoted as $so(3)$, is the space of the 3×3 skew-symmetric matrices:

$$so(3) = \{\widehat{\omega} \in \mathbb{R}^{3\times 3} :\ \widehat{\omega}^T = -\widehat{\omega}\}, \tag{2.5}$$

$$\widehat{\omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \tag{2.6}$$

with the Lie bracket

$$[\widehat{\omega}_1, \widehat{\omega}_2] = \widehat{\omega}_1\widehat{\omega}_2 - \widehat{\omega}_2\widehat{\omega}_1, \quad \widehat{\omega}_1, \widehat{\omega}_2 \in so(3),$$

where $\omega = [\omega_x, \omega_y, \omega_z]^T$ is the angular velocity vector of the rigid body. As can be seen on the structure of the elements of $so(3)$, the product of $\hat{a} \in so(3)$ and an arbitrary vector $b \in \mathbb{R}^3$ is equal to the cross product $a \times b$ (Equation 2.7)

$$\widehat{a}b = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{bmatrix} = a \times b. \tag{2.7}$$

Similar to $so(3)$, $se(3)$ is the Lie algebra of $SE(3)$. It consists of 4×4 matrices of the form

$$\widehat{s} = \begin{bmatrix} \widehat{\omega} & v \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times4}, \tag{2.8}$$

with $\widehat{\omega} \in so(3)$ and translational velocity of the body $v \in \mathbb{R}^3$. The derivations of Lie algebras can be found in [Sel05].

2.4 Matrix Exponential

The relations between Lie groups and Lie algebras are provided by exponential mappings. In [MSZ94], Murray describes their derivation using exponential coordinates. For $SO(3)$ and $so(3)$ it can be shown, that a rotation matrix can be calculated using the direction ω and the absolute value q of the corresponding angular velocity

$$R(\omega, q) = e^{\widehat{\omega}q}. \tag{2.9}$$

The exponential mapping $e^{\widehat{\omega}q} : so(3) \to SO(3)$ is also called matrix exponent because of the matrix form of $\widehat{\omega} \in \mathbb{R}^{3\times 3}$. The matrix exponent has a following form

$$e^{\widehat{\omega}q} = I + \frac{\widehat{\omega}}{\|\omega\|}\sin(\|\omega\|q) + \frac{\widehat{\omega}^2}{\|\omega\|^2}(1-\cos(\|\omega\|q)). \tag{2.10}$$

Since the absolute value of the angular velocity is provided in separate quantity q it is convenient to use a unit vector ω with $\|\omega\| = 1$ to describe the axis of the rotational motion

$$e^{\widehat{\omega}q} = I + \widehat{\omega}\sin q + \widehat{\omega}^2(1-\cos q). \tag{2.11}$$

The above formulation, also called *Rodrigues' formula*, allows an efficient computation of matrix exponents.

The relation between $SE(3)$ and $se(3)$ is based on the matrix exponent $e^{\widehat{\omega}} : se(3) \to SE(3)$. An element $\widehat{s} \in se(3)$ can be mapped in $SE(3)$ with

$$e^{\widehat{s}q} = \begin{bmatrix} e^{\widehat{\omega}}q & (I - e^{\widehat{\omega}})(\omega \times v) + \omega\omega^T vq \\ 0 & 1 \end{bmatrix} \in SE(3). \tag{2.12}$$

In case of pure translational motion of rigid body, where $\omega = 0$, the formulation is simplified to

$$e^{\widehat{\omega}}q = \begin{bmatrix} I & vq \\ 0 & 1 \end{bmatrix}, \quad \omega = 0 \tag{2.13}$$

and for rotations only to

$$e^{\widehat{\omega}}q = \begin{bmatrix} e^{\widehat{\omega}}q & 0 \\ 0 & 1 \end{bmatrix}, \quad v = 0. \tag{2.14}$$

2.5 Twist

The elements of the $se(3)$ also referred to as *Twists* play a central role in the formulation of the kinematics in robotics.A twist represents the infinitesimal version of a screw motion. Since a twist contains information about translation and rotation it describes a instantaneous motion of a solid state body completely. The matrix representation of a twist can be transformed to a 6-dimensional vector which parametrizes the twist:

$$\widehat{s}^{\vee} = \begin{bmatrix} \widehat{\omega} & v \\ 0 & 0 \end{bmatrix}^{\vee} = \begin{bmatrix} v \\ \omega \end{bmatrix} = s. \tag{2.15}$$

The $\vee$ (*vee*) operator provides the transformation of a twist from $se(3)$ to $\mathbb{R}^6$. The $\wedge$ (*wedge*) operator do the inverse transformation:

$$\begin{bmatrix} v \\ \omega \end{bmatrix}^{\wedge} = \begin{bmatrix} \widehat{\omega} & v \\ 0 & 0 \end{bmatrix}. \tag{2.16}$$

In this thesis the term twist will denote the 6-dimensional representation $s \in \mathbb{R}^6$. On positions where the other representation is meant it will be stated explicitly.

Representing particular motion parts, twists would look as follows

$$s = \begin{bmatrix} 0 \\ \omega \end{bmatrix} \tag{2.17}$$

for pure rotation and

$$s = \begin{bmatrix} v \\ 0 \end{bmatrix} \tag{2.18}$$

for pure translation.

2.6 Wrench

Similar to twists the term *wrench* describes the forces and torques both internal and external. A wrench is also a 6-dimensional vector

$$F = \begin{bmatrix} f \\ \tau \end{bmatrix} \in \mathbb{R}^6, \tag{2.19}$$

where $f, \tau \in \mathbb{R}^3$ are a linear force and a rotational torque. While twists describe the kinematics of rigid body motion, wrenches play the central role in description of the dynamics.

2.7 Adjoint Mapping

Since the twists of the rigid body motion are often described in body coordinates it is necessary to transform a twist from body coordinates to spatial coordinates. This transformation is provided by *adjoint* mapping $Ad_H : se(3) \to se(3)$:

$$Ad_H = \begin{bmatrix} R & \widehat{p}R \\ 0 & R \end{bmatrix} \in \mathbb{R}^{6\times 6}, \tag{2.20}$$

which is associated with

$$H = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \in SE(3). \tag{2.21}$$

The adjoint transformation is invertible

$$Ad_H^{-1} = \begin{bmatrix} R^T & -\widehat{(R^T p)}R^T \\ 0 & R^T \end{bmatrix} = \begin{bmatrix} R^T & -R^T\widehat{p} \\ 0 & R^T \end{bmatrix} = Ad_{H^{-1}}. \tag{2.22}$$

It is also often necessary to transform a twist from body coordinates of a rigid body to twist coordinates of an other body. Such adjoint mapping is denoted as $ad_s : se(3) \rightarrow se(3)$

$$ad_s = \begin{bmatrix} \widehat{\omega} & \widehat{v} \\ 0 & \widehat{\omega} \end{bmatrix} \in \mathbb{R}^{6\times 6} \tag{2.23}$$

and is associated with a twist

$$s = \begin{bmatrix} v \\ \omega \end{bmatrix} \in se(3). \tag{2.24}$$

The inverse of ad_s is given by

$$ad_s^{-1} = \begin{bmatrix} -\widehat{\omega} & -\widehat{v} \\ 0 & -\widehat{\omega} \end{bmatrix} = -ad_s = ad_{-s}. \tag{2.25}$$

2.8 Equations of Motion

Using the previuos theoretical background the motion equation that describes the current dynamic model of the modular robotic system can be derived as shown in the work of Chen and Yang [CY98]:

$$\begin{aligned} \mathbf{V} &= \mathbf{G}\mathbf{S}\dot{\mathbf{q}}, && (2.26)\\ \dot{\mathbf{V}} &= \mathbf{T_{H_0}}\dot{V_0} + \mathbf{T}\mathbf{S}\ddot{\mathbf{q}} + \mathbf{T}\mathbf{A_1}\mathbf{V}, && (2.27)\\ \mathbf{F} &= \mathbf{T^T}\mathbf{F^e} + \mathbf{T^T}\mathbf{M}\dot{\mathbf{V}} + \mathbf{T^T}\mathbf{A_2}\mathbf{M}\mathbf{V}, && (2.28)\\ \boldsymbol{\tau} &= \mathbf{S^T}\mathbf{F}, && (2.29) \end{aligned}$$

where

$$
\begin{aligned}
\dot{\mathbf{q}} &= column[\dot{q}_1, \dot{q}_2, \dots, \dot{q}_n] \in R^{n\times 1} \text{ is the joint velocity vector;}\\
\ddot{\mathbf{q}} &= column[\ddot{q}_1, \ddot{q}_2, \dots, \ddot{q}_n] \in R^{n\times 1} \text{ is the joint acceleration vector;}\\
\mathbf{V} &= column[V_1, V_2, \dots, V_n] \in R^{6n\times 1} \text{ is the generalized body velocity vector;}\\
\dot{\mathbf{V}} &= column[\dot{V}_1, \dot{V}_2, \dots, \dot{V}_n] \in R^{6n\times 1} \text{ is the generalized body acceleration vector;}\\
\mathbf{F} &= column[F_1, F_2, \dots, F_n] \in R^{6n\times 1} \text{ is the body wrench vector;}\\
\mathbf{F^e} &= column[F_1^e, F_2^e, \dots, F_n^e] \in R^{6n\times 1} \text{ is the external wrench vector;}\\
\boldsymbol{\tau} &= column[\tau_1, \tau_2, \dots, \tau_n] \in R^{n\times 1} \text{ is the applied torque/forces vector;}\\
\mathbf{S} &= diag[s_1, s_2, \dots, s_n] \in R^{6n\times n} \text{ is the joint twist matrix;}\\
\mathbf{M} &= diag[M_1, M_2, \dots, M_n] \in R^{6n\times 6n} \text{ is the combined generalized mass matrix;}\\
\mathbf{A_1} &= diag[-ad_{s_1\dot{q}_1}, -ad_{s_2\dot{q}_2}, \dots, -ad_{s_n\dot{q}_n}] \in R^{6n\times 6n};\\
\mathbf{A_2} &= diag[-ad^T_{V_1}, -ad^T_{V_2}, \dots, -ad^T_{V_n}] \in R^{6n\times 6n};
\end{aligned}
$$

$$
\mathbf{T_{H_0}} = \begin{bmatrix} Ad_{H_{01}^{-1}} \\ Ad_{H_{02}^{-1}} \\ \vdots \\ Ad_{H_{0n}^{-1}} \end{bmatrix} \in \mathbb{R}^{6n\times 6},
$$

$$
\mathbf{T} = \begin{bmatrix} I_{6\times6} & 0_{6\times6} & 0_{6\times6} & \cdots & 0_{6\times6} \\ Ad_{H_{12}^{-1}} & I_{6\times6} & 0_{6\times6} & \cdots & 0_{6\times6} \\ Ad_{H_{13}^{-1}} & Ad_{H_{23}^{-1}} & I_{6\times6} & \cdots & 0_{6\times6} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ Ad_{H_{1n}^{-1}} & Ad_{H_{2n}^{-1}} & Ad_{H_{3n}^{-1}} & \cdots & I_{6\times6} \end{bmatrix} \in \mathbb{R}^{6n\times 6n}.
$$

The system model is described then by the following equation

$$
\mathbf{M(q)\ddot{q}} + \mathbf{C(q,\dot{q})\dot{q}} + \mathbf{N(q)} = \boldsymbol{\tau}, \tag{2.30}
$$

where

$$
\begin{aligned}
\mathbf{M(q)} &= \mathbf{S^T T^T M T S}, && (2.31)\\
\mathbf{C(q,\dot{q})} &= \mathbf{S^T T^T (M T A_1 + A_2 M) T S}, && (2.32)\\
\mathbf{N(q)} &= \mathbf{S^T T^T M T_{H_0}} \dot{V}_0 + \mathbf{S^T T^T F^e}. && (2.33)
\end{aligned}
$$

M(q) is the mass matrix, $\mathbf{C(q,\dot{q})}$ describes the Coriolis and centrifugal accelerations and **N(q)** represents gravitational and external forces.

3 Applied Control Strategies

The equations for kinematics and dynamics of a given multibody robotic organism are automatically generated by the C++ framework that was described previously [Nos13]. Thus the geometrical model for a given robot construction can be established. With the knowledge of the system, which is represented by the geometrical model, a control strategy need to be considered to achieve an optimal behavior of motions of the robot. The object is that the motions of each joint module and the robotic organism as a whole are as much as possible consistent with the desired trajectory. There are many approved control design strategies that can be used for linear and nonlinear systems. Optimal tracking problem or Internal Model Control (IMC) for example are intended for control of linear systems. Nonlinear examples of control strategies are exact linearization method, asymptotic set-point following control or sliding mode control. The geometrical model generated by the framework represents a nonlinear system. Therefore a linearization approach is necessary for reducing the complexity of controller design. The most common linearization methods are based on the Jacobian linearization at a certain point of operation. Another approved technique to control nonlinear system is the use of observers. This method is particularly advantageous if not system states or system outputs can be measured.

As described in the previous chapter the dynamics of the robotic organism is represented by the following equation of motion:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q) = \tau. \tag{3.1}$$

For motion control the equation is transformed by solving it for $\ddot{q}$, so that the system to be controlled is described by

$$\ddot{q} = M(q)^{-1}(\tau - C(q,\dot{q})\dot{q} - N(q)). \tag{3.2}$$

The second-order differential equation can be transferred into a vector representation of two first-order differential equations

$$\dot{\begin{bmatrix} q \\ \dot{q} \end{bmatrix}} = \begin{bmatrix} \dot{q} \\ -M(q)^{-1}(C(q,\dot{q})\dot{q} + N(q)) \end{bmatrix} + \begin{bmatrix} 0 \\ M(q)^{-1} \end{bmatrix} \tau. \tag{3.3}$$

This version fits the form of representation that is commonly used in the classical control theory for description of nonlinear systems:

$$\dot{x} = f(x) + g(x)\tau, \tag{3.4}$$

$$y = h(x), \tag{3.5}$$

where

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}, \dot{x} = \dot{\begin{bmatrix} q \\ \dot{q} \end{bmatrix}}, \tag{3.6}$$

$$f(x) = \begin{bmatrix} x_2 \\ -M(x_1)^{-1}(C(x_1, x_2)x_2 + N(x_1)) \end{bmatrix}, \tag{3.7}$$

$$g(x) = \begin{bmatrix} 0 \\ M(x_1)^{-1} \end{bmatrix}, \tag{3.8}$$

$$h(x) = x_1. \tag{3.9}$$

x is the new system state that combines angles and angular velocities of the joints. The output of the system is denoted by y.

This chapter describes several techniques of the control theory that can be applied for control of the geometrically generated model:

- Open-loop control.
- Pure PID control.
- Local stabilization.
- Computed-torque control.
- State reconstruction using the extended Kalman filter.

3.1 Structural Properties

To be able to design a proper control mechanism, one needs a deeper understanding of the system to be controlled. The Equation 3.2 describes a nonlinear, multivariable, time invariant system. Its dimension equals the number of the modules which it consists of. In the transformed representation in the Equation 3.3 the system dimension is doubled. Although the system has a multivariable nature, in many cases it can be considered as a singlevariable one using q and $\dot{q}$ as scalar values. If the assumption makes a significant difference, then the multivariable nature of the system is emphasized explicitly. In the following, some significant structural properties of the nonlinear system are specified.

3.1.1 Relative Degree

The term *relative degree* comes from the control theory of linear systems. It is defined there as follows:

Definition 3.1.1 (Relative Degree (linear systems))
The difference between the numbers of pole and zeros of a linear system is called ***relative degree****.*

Poles and zeros are determined using *transfer function* that is applicable only for linear systems. For nonlinear dynamic systems *relative degree* is defined more generally so that the concept of poles and zeros forms a special case of it. The definition is based on *Lie derivatives* which are determined as follows:

$$L_f^i h(x) = \frac{\partial L_f^{i-1} h(x)}{\partial x} f(x), \quad L_f^0 h(x) = h(x), \tag{3.10}$$

$$L_g L_f h(x) = \frac{\partial L_f h(x)}{\partial x} g(x). \tag{3.11}$$

Definition 3.1.2 (Relative Degree (nonlinear systems))
The ***relative degree*** *of a system at the operating point $x_s = 0$ is the natural number r for which the following holds:*

1. $L_g L_f^i h(x) = 0$ *for* $i = 0, 1, \dots, r-2$,
2. $L_g L_f^{r-1} h(x) \neq 0$

in the neighborhood of x_s.

Thereby relative degree r is the number of the first time derivative of the controlled variable y that depends directly on the input signal τ. Using the upper definitions the relative degree of the generated dynamic system (Equations 3.4 and 3.5) can be determined. The appropriate Lie derivatives are calculated as follows:

$$L_g L_f^0 h(x) = L_g h(x) = \frac{\partial h(x)}{\partial x} g(x) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ M(x_1)^{-1} \end{bmatrix} = 0,$$
$$L_g L_f^1 h(x) = \frac{\partial L_f h(x)}{\partial x} g(x) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ M(x_1)^{-1} \end{bmatrix} = M(x_1)^{-1} \neq 0.$$

As a result $r = 2$. The mass matrix $M(q) \neq 0$ at $q = 0$ as well in the neighborhood of $q = 0$, thus the Lie derivative $L_g L_f h(x) = M(x_1)^{-1} \neq 0$ at x_s and in its neighborhood. For this reason the relative degree is also called *well-defined*. In addition $r > 0$ means that the controlled variable y does not depend explicitly on the input signal τ. According to that, the dynamic system has no *feed-through*, such systems also called *strictly proper*. In fact, the generated dynamic system is a multivariable one with x and $\dot{x} \in \mathbb{R}^n$, where n is the number of the joint modules that compose the multi-body robotic organism. Thus the relative degree is actually a vector of dimension n whose entries are relative degrees of each particular differential equation of the n-dimensional multivariable system represented by the Equation 3.2. Due to Equations 3.7 and 3.8 it is obvious that all the entries of r have the same value:

$$r = \begin{bmatrix} 2 \\ 2 \\ \vdots \\ 2 \end{bmatrix} \in \mathbb{R}^n \tag{3.12}$$

3.1.2 Zero Dynamics

A further basic structural property of a nonlinear dynamic system is the *zero dynamics*.

Definition 3.1.3 (Zero Dynamics)
*The **zero dynamics** describes the internal dynamics of a system in the case that the controlled variable $y(t)$ input signals $u(t)$ and initial conditions $x(0)$ for all the times $t \geq 0$ is null.*

It is a crucial task in control design to determine the zero dynamics of a system because it cannot be changed via state or output feedback and is invariant according the coordinate transformation. The zero dynamics describes the internal dynamics of a system when the output is equal null. After the Definition 3.1.3 the zero dynamics is determined via setting the system output to zero:

$$y \overset{!}{\equiv} 0 \Rightarrow x_i \overset{!}{\equiv} 0 \quad i = 1, 2, \dots, r,$$

which includes that the first r time derivatives $x_1, x_2, \dots, x_r$ are also zero and particularly:

$$\dot{x_r} = -M(x_1)^{-1}(C(x_1, x_2)x_2 + N(x_1)) + M(x_1)^{-1}\tau_z \overset{!}{\equiv} 0, \tag{3.13}$$

where τ_z denotes the input signal that determines the zero dynamics of the system at the operating point $x_s = 0$. τ_z is calculated as follows:

$$\tau_z = -\frac{L_f^r h(x)}{L_g L_f^{r-1} h(x)} = \frac{M(x_1)^{-1}(C(x_1, x_2)x_2 + N(x_1))}{M(x_1)^{-1}} \overset{x_1 \equiv 0, x_2 \equiv 0}{=} N(0) \tag{3.14}$$

The matrix $N(0)$ contains the gravitational and external forces in an equilibrium state, where there is no angular deflection in each joint module.

As shown above, in the case the relative degree is equal the system dimension $r = n$ the considered system has no zero dynamics and therefore is called *minimum-phase* system. This property is essential for the desired control quality. There are several control design strategies that require a controlled system to be minimum-phase for successful exertion. For example the approach of exact input/output linearization can only be applied for minimum-phase systems. With a deeper knowledge of the dynamic system more adequate control strategies can be considered to achieves a desired motion behavior of the robotic organism.

3.2 Open-Loop Control

If a desired trajectory $q_d(t)$ is known and its first and second derivatives $\dot{q}_d(t)$ and $\ddot{q}_d(t)$ exist as well then the necessary torques are calculated using the Equation 3.1:

$$\tau = M(q_d)\ddot{q}_d + C(q_d, \dot{q}_d)\dot{q}_d + N(q_d). \tag{3.15}$$

The received torque vector τ serves as input for the generated dynamic system (Equation 3.2). This method requires that the initial states of the system and desired trajectory are exactly identical: $q(0) = q_d(0)$, $\dot{q}(0) = \dot{q}_d(0)$. Additionally the generated model (Equation 3.1) is assumed to perfectly describe the kinematics and dynamics of the robot. Due to the uniqueness of the solutions of differential equations, it follows that $q(t) = q_d(t)$ for all $t \geq 0$. This approach is called *open-loop* control because the current state or output are not used for composition of the system input [MSZ94]. The appropriate block diagram is depicted in the Figure 3.1.

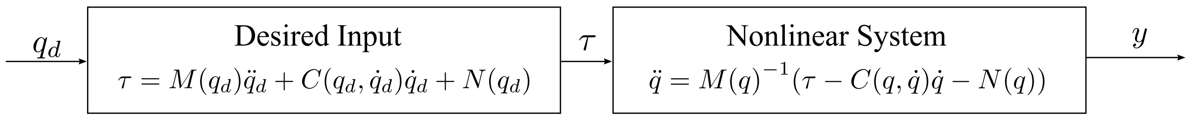

q_d
Desired Input
$\tau = M(q_d)\ddot{q}_d + C(q_d, \dot{q}_d)\dot{q}_d + N(q_d)$
τ
Nonlinear System
$\ddot{q} = M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - N(q))$
y

Figure 3.1: Open-Loop Control

Since the controller does not use the current system state there is no need of the appropriate sensors. But the lack of the system feedback results the small robustness of the open-loop control strategy. There is no guarantee that the initial state of the system $q(0)$, $\dot{q}(0)$ exactly fits the initial state of the trajectory $q_d(0)$, $\dot{q}_d(0)$. Each further possible deviation $q_d - q$ also remains uncorrected.

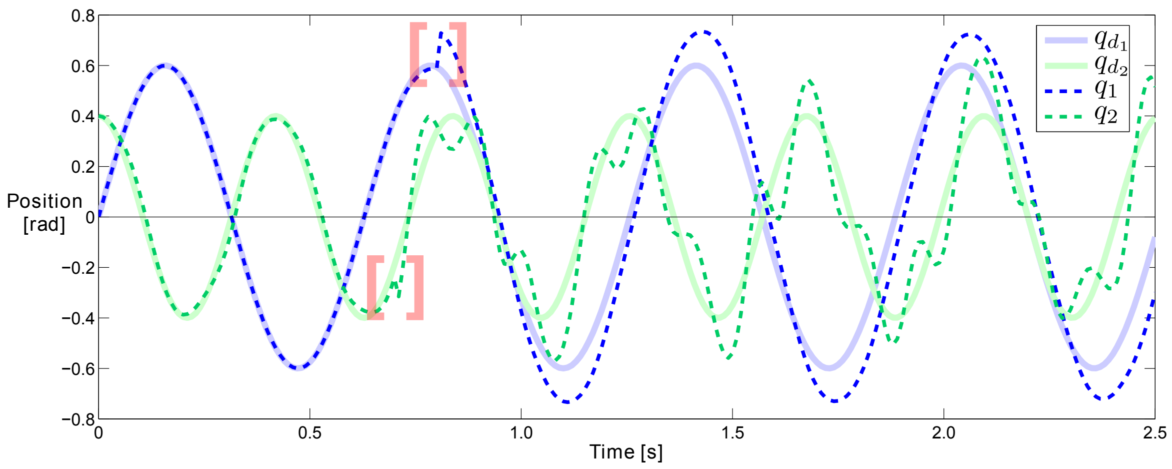


Figure 3.2: Open-loop control of double pendulum, disturbance behavior.

Figure 3.2 shows angular position trajectories of joint modules of a double pendulum. The desired sinusoidal trajectory and the output signal of the system are simulated by MATLAB®. The two marked places denote the step-shaped disturbance of the system state that may be caused by the robot environment. The error affects the system output but it can not be corrected to the desired behavior due to the lack of the system feedback. The accumulated difference $q_d - q$ would lead to an unpredictable and chaotic behavior. For this reason, the input signal must be composed including the feedback information of the current system state. Further sections describe feedback based control strategies.

3.3 PID Control

A system feedback provides information about the current state of the dynamic system. Thus the control task consist in measuring of the current system state or controlled variable, comparing it with the desired value and changing the control signal if an error occurs. The input value must be changed so that the error is minimal and the desired and controlled values remain consistent. The PID control provides the error compensation in a simple manner. The advantages of a PID controller are its simplicity and clear physical meanings [Sic08]. The physical meanings of the PID control are described as follows:

P : The proportional part lets the current deviation affect the next state directly.

I : The previous behavior affects the system through the integral component.

D : The differential component evaluates the relative change of the control error so that the controller can respond to preceding behavior of the system.

Because of its simplicity and solid control characteristics, the PID control and its variations are commonly used control methods. As shown in Figure 3.3, the PID controller provides the calculation of the control signal τ depending on the deviation $e = x_d - x = [e_q,\ e_{\dot{q}}]^T$, where $x_d = [q_d,\ \dot{q}_d]^T$ is the reference signal and $x = [q,\ \dot{q}]^T$ denotes the current state vector.

3.3.1 PD Control

Although the system, which is described in the Equation 3.2 is nonlinear, under certain conditions a linear approach can be used for control design. A simple variation of PID control design is the PD control. PD controller is a linear controller, that is based on the linearization of the system in the neighborhood of the operating point. It is

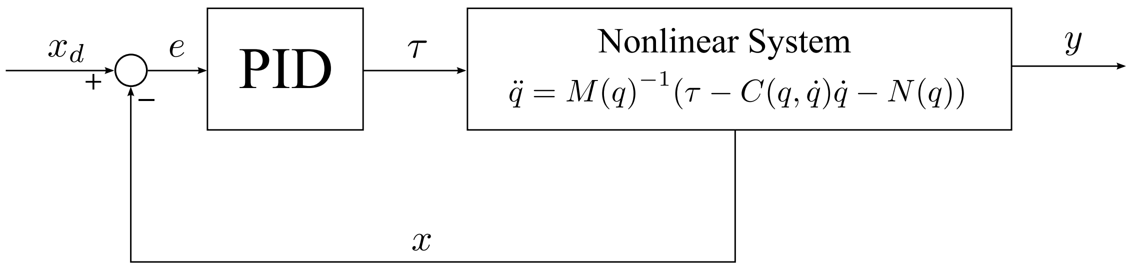


Figure 3.3: Block diagram of the PID control.

assumed that the stability of the system linearization locally determines the stability of the system itself. In its simplest form, a PD control law is described as follows:

$$\tau = K_p e_q + K_d e_{\dot{q}} = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}), \tag{3.16}$$

where K_p and $K_d \in \mathbb{R}^{n\times n}$ are positive definite gain matrices. Combining the equation terms 3.1 and 3.16 the closed-loop dynamic equation becomes

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q) - K_d(\dot{q}_d - \dot{q}) - K_p(q_d - q) = 0. \tag{3.17}$$

Siciliano [Sic08] has shown that the PD controlled dynamic system is stable in the equilibrium state x_s in the sense of Lyapunov. However it can not conclude that the regulation error converges to zero by Lasalle's theorem. It means that the PD controlled system is not able to track the desired trajectory exactly. Thus there is always a steady-state error in the PD motion control. The step response signal of the PD controlled double pendulum demonstrates the steady-state error in the Figures 3.4 and 3.5.

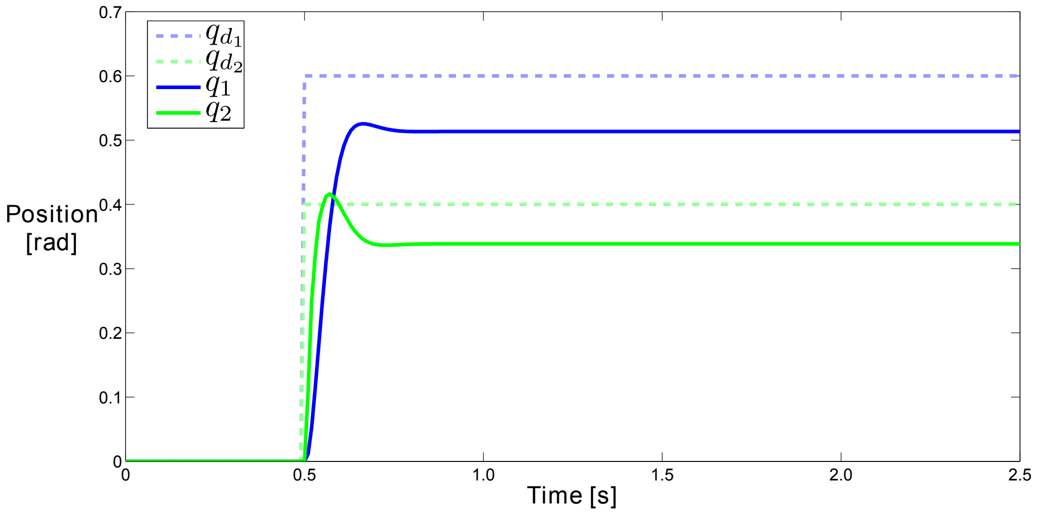


Figure 3.4: PD control: Step response of double pendulum.

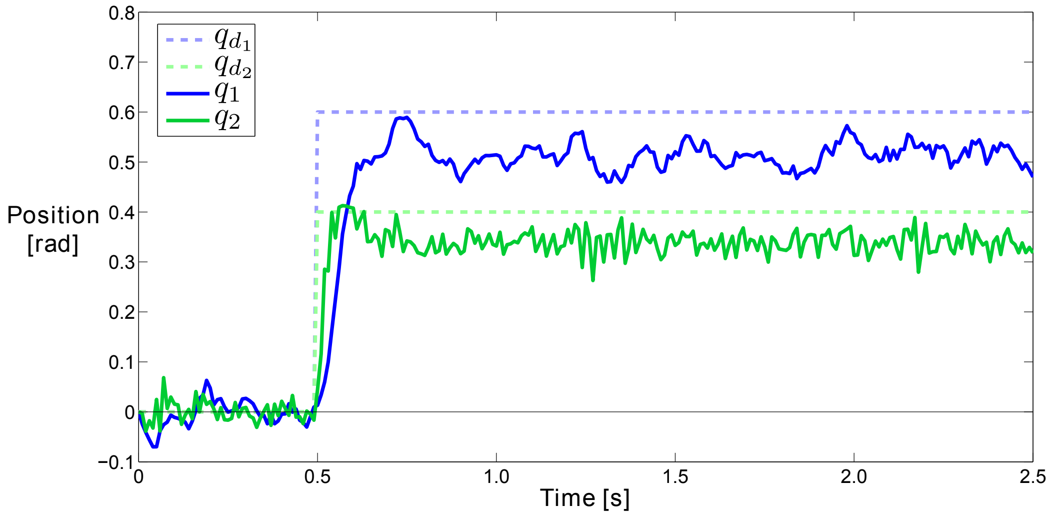


Figure 3.5: PD control: Step response of double pendulum, measurement noise.

The system response of the sinusoidal reference signal is shown in the Figure 3.6. At the times $t_1 = 0.55\ s$ and $t_2 = 1.0\ s$ the state disturbances $d_1 = -0.15\ rad$ and $d_2 = 0.15\ rad$ are simulated. As can be seen, in contrast to the open-loop approach, the state deviations are corrected by the PD controller to match the reference trajectory.

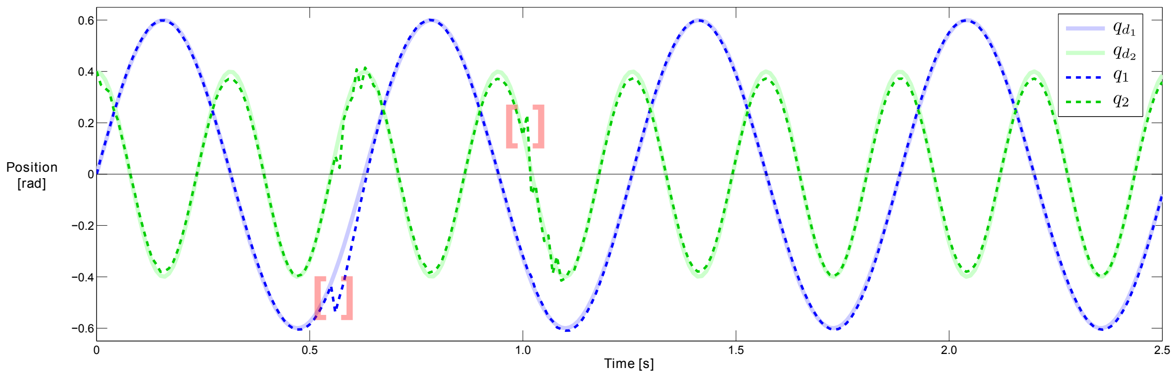


Figure 3.6: PD control of double pendulum, disturbance behavior.

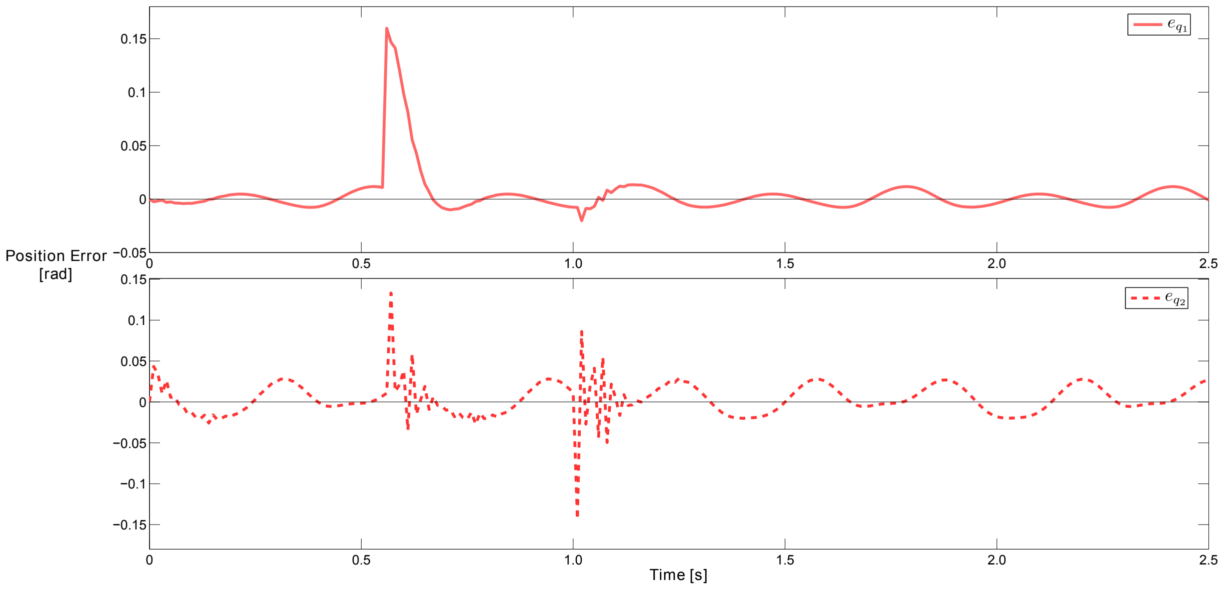


Figure 3.7: Position error of the PD controlled double pendulum.

It is also noticed that the disturbance that is acting on a joint module, is propagated to the neighbor modules in a reduced extent. The Figure 3.7 shows the graph of deviations between the reference signals and the calculated trajectory of the joint modules of the double pendulum. The mutual action of the dynamics of the particular modules becomes even more clear here.

3.3.2 PD Control with Gravity Compensation

As can be seen the PD control provides a suitable method for motion control of the multi-body robotic organisms. But as shown in the Figures 3.4 and 3.5 its set-point regulation is not optimal because of the steady-state error that PD controller is not able to to correct. Kelly [Kel97] proposes the inclusion of the gravitational component for the calculation of the control signal:

$$\tau = K_p e_q + K_d e_{\dot{q}} + N(q). \tag{3.18}$$

In this case the closed-loop dynamic equation has the following form:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} - K_d(\dot{q}_d - \dot{q}) - K_p(q_d - q) = 0. \tag{3.19}$$

Gravity compensation acts as a bias correction, compensating only for the amount of forces that create overshooting and an asymmetric transient behavior [Sic08]. Siciliano has also shown, that this form of the controlled system is asymptotically stable. Thus the regulation error converges asymptotically to zero. A step response of the double pendulum which is controlled by the PD control with gravity compensation, is depicted in the Figure 3.8. As can be seen the steady-state error is neutralized in comparison to the pure PD control method (Figure 3.4).

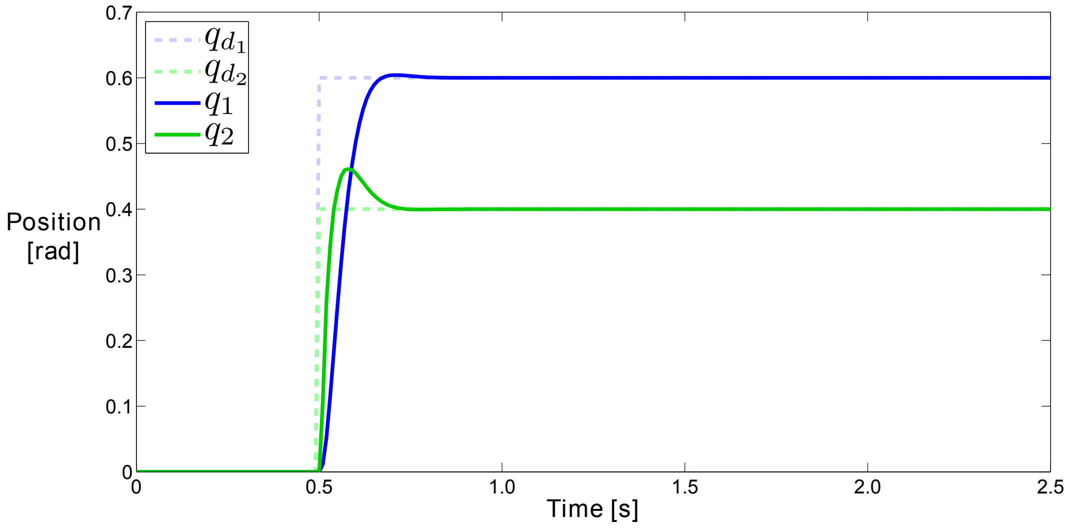


Figure 3.8: Step responese of the double pendulum with PD control and gravity compensation.

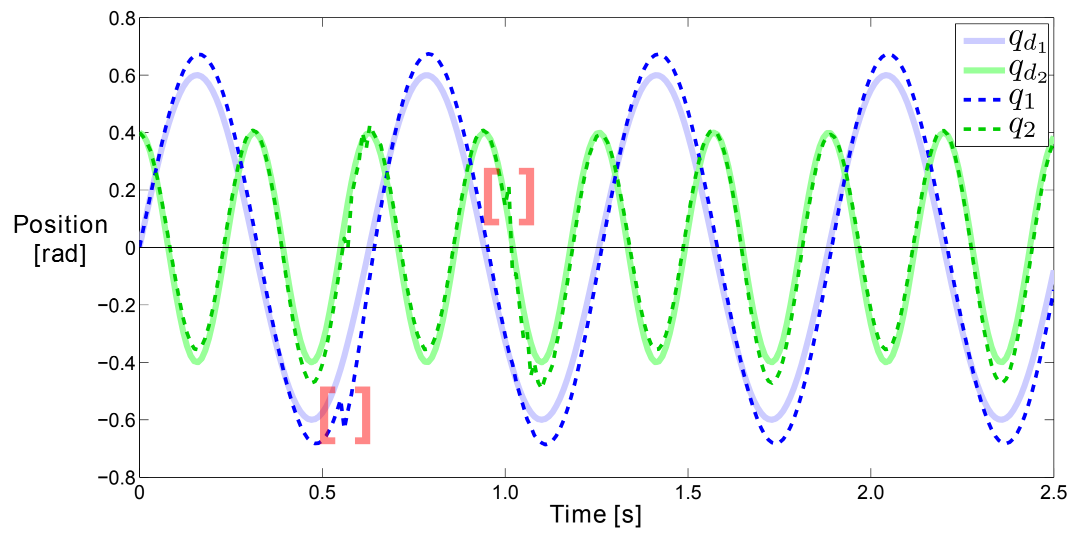


Figure 3.9: PD control of double pendulum with gravity compensation, disturbance behavior.

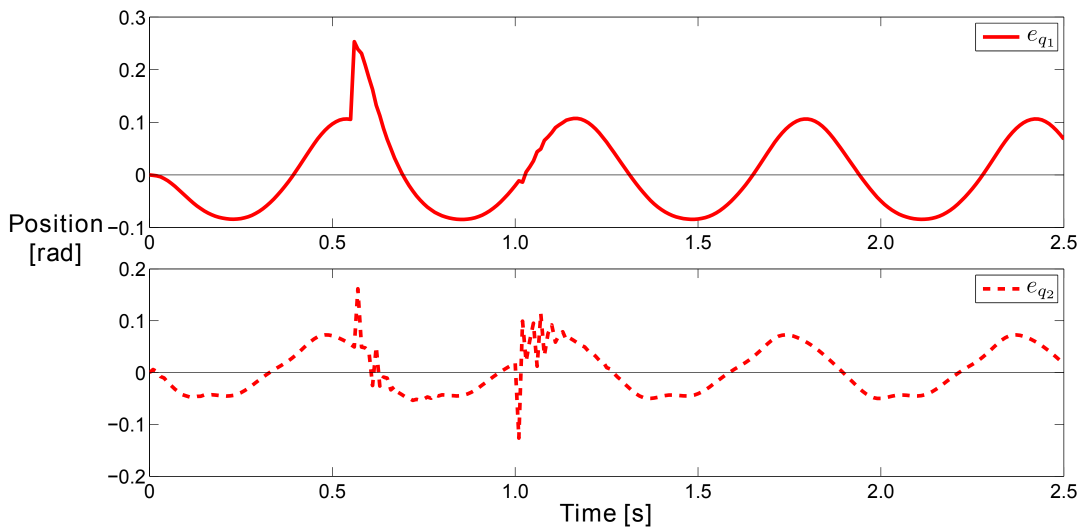


Figure 3.10: Trajectory tracking of the double pendulum with PD control and gravity compensation.

Due to the solid behavior against the steady-state errors PD control with the gravity compensation is suitable for set-point motion regulation. Figure 3.9 demonstrate the control approach for tracking of the desired trajectory. Although the disturbances

$d_1(t_1 = 0.55) = -0.15$ rad and $d_2(t_2 = 1.0) = 0.15$ rad are quickly corrected and they do not cause a steady-state error, the entire tracking properties are suboptimal in comparison with the simple PD control method. The tracking error of the gravity compensated PD controller is shown in the Figure 3.10. The error amplitude is distinctly greater compared to the one of the pure PD control (Figure 3.7).

3.3.3 Augmented PD Control

Since it is advantageous to include the gravity vector $N(q)$ for calculation of the control signal τ, it is obvious that also the mass matrix $M(q)$ and the Coriolis matrix $C(q, \dot{q})$ can be used for the same purpose. Murray et al. [MSZ94] propose the so called *augmented PD control law*:

$$\tau = M(q)\ddot{q}_d + C(q, \dot{q})\dot{q}_d + N(q) + K_d e_{\dot{q}} + K_p e_q. \tag{3.20}$$

Therewith the closed-loop dynamic system is described as follows:

$$M(q)e_{\ddot{q}} + C(q, \dot{q})e_{\dot{q}} + K_d e_{\dot{q}} + K_p e_q = 0. \tag{3.21}$$

With $e_q = q_d - q$, $e_{\dot{q}} = \dot{q}_d - \dot{q} = \dot{e}_q$ and $e_{\ddot{q}} = \ddot{q}_d - \ddot{q} = \ddot{e}_q$ the Equation 3.21 describes the error dynamics of the controlled system.

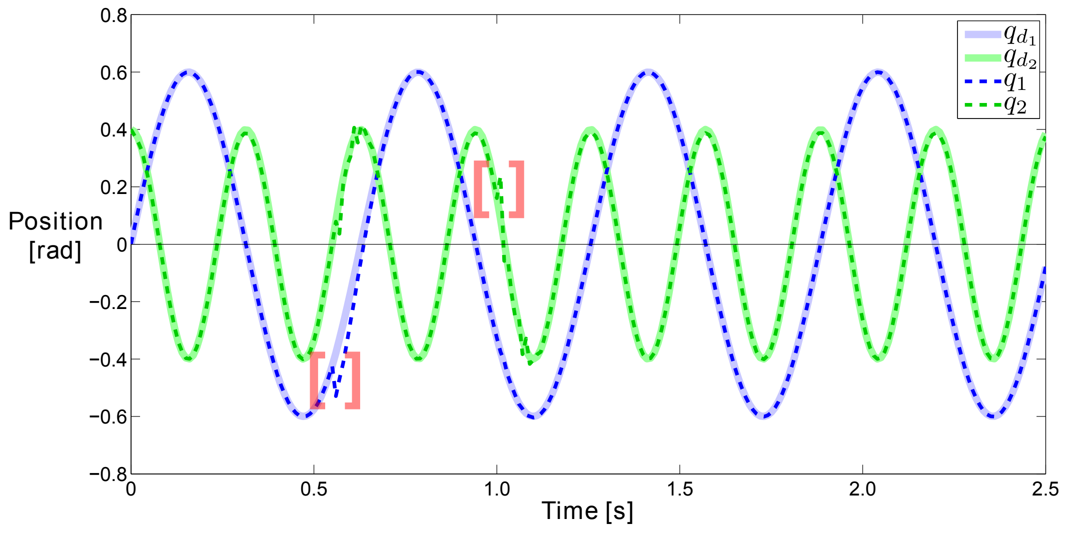


Figure 3.11: Trajectory tracking of the augmented PD control.

Murray et al. [MSZ94] have also shown that augmented PD control law results in exponential trajectory tracking if K_v and K_p are positive definite. The trajectory tracking of double pendulum with the augmented PD control is shown in the Figure 3.11. The associated error tracking is depicted in Figure 3.12. Due to the high robustness the augmented PD controller is proper for tracking regulation.

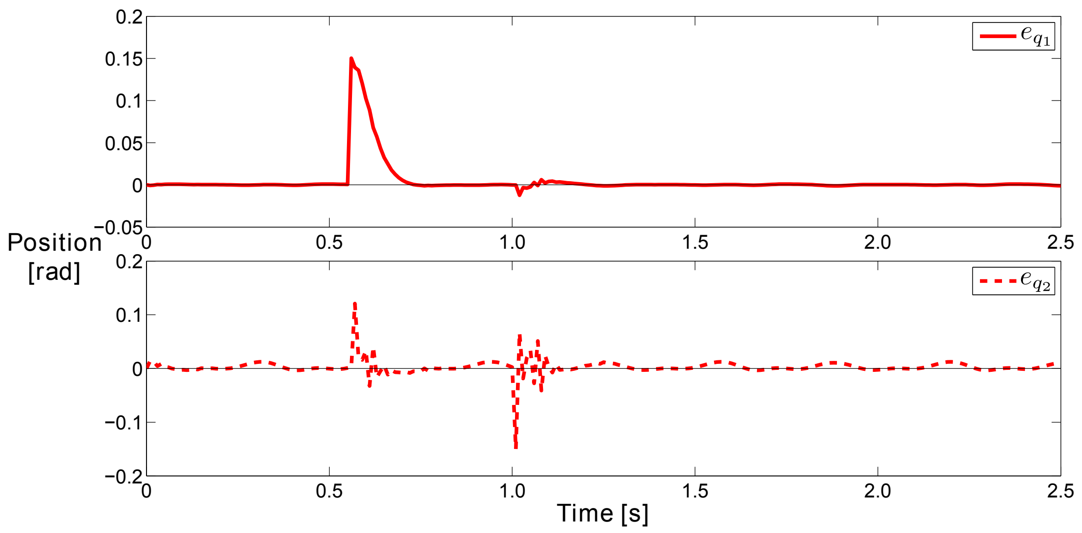


Figure 3.12: Tracking error of the augmented PD control.

For set-point regulation, where $\ddot{q}_d \equiv \dot{q}_d \equiv 0$, the augmented PD control low (Equation 3.20) simplifies to the PD control law with the gravity compensation that is described in the section 3.3.2. Thus the augmented PD control strategy is a solid approach for the set-point regulation as well as for tracking of the desired trajectory.

3.3.4 PID Control

An other approach for avoiding the steady-state errors is to upgrade the PD control law with an integral component. The integral component provides the information about the whole deviation that was made from the start of the control $t = 0$ until the current moment. Including the integral component the control signal is evaluated as follows:

$$\tau = K_p e_q + K_i \int e_q \, \mathrm{d}t + K_d e_{\dot{q}}, \tag{3.22}$$

where K_i, just like K_p and K_d, is a positive definite gain matrix. The global asymptotic stability of the PID controlled robotic motion systems was proven by Arimoto [Ari84].

Step responses of the PID controlled double pendulum which are depicted in figures 3.13 and 3.14, demonstrate the robustness against the steady-state errors.

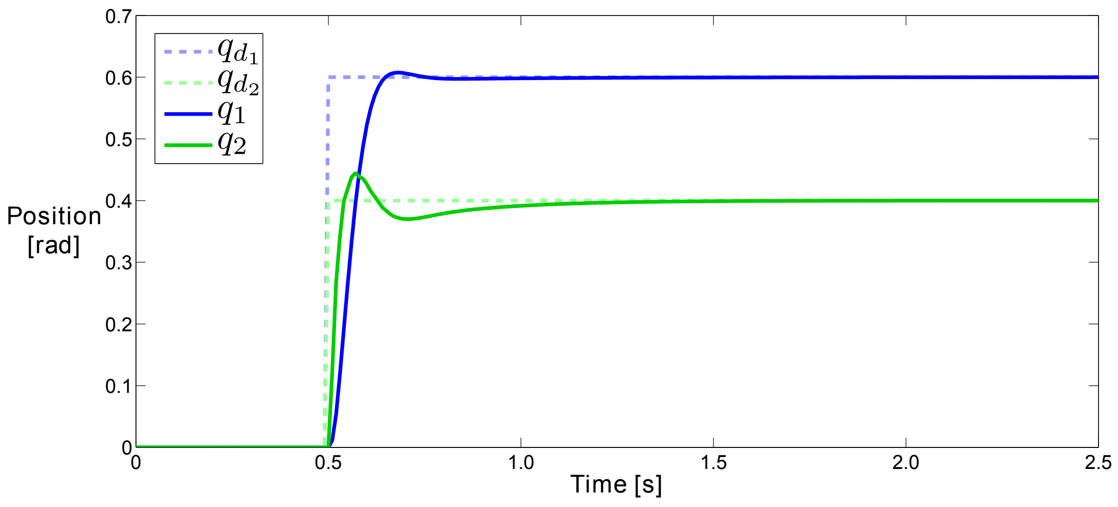


Figure 3.13: PID step response of double pendulum

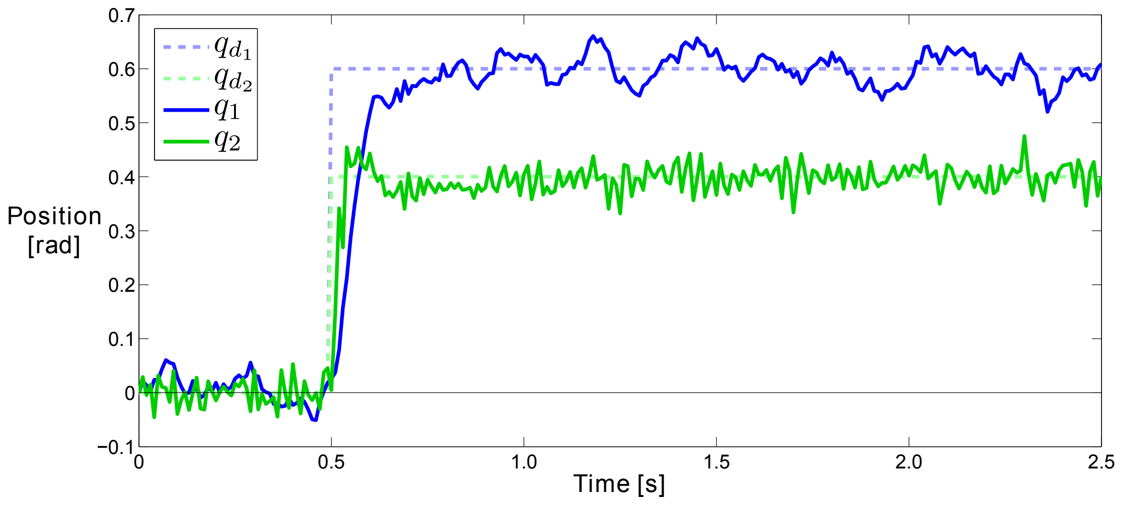


Figure 3.14: PID step response of double pendulum, measurement noise.

Additionally the PID control shows a robust behavior against disturbances and measurement noise as presented in Figures 3.15 and 3.16. Due to solid control properties the PID control strategy is suitable for both set-point regulation and trajectory tracking. Analogically to the PD control variations the PID control design also can be upgraded by additional components, for example gravity compensation or friction compensation.

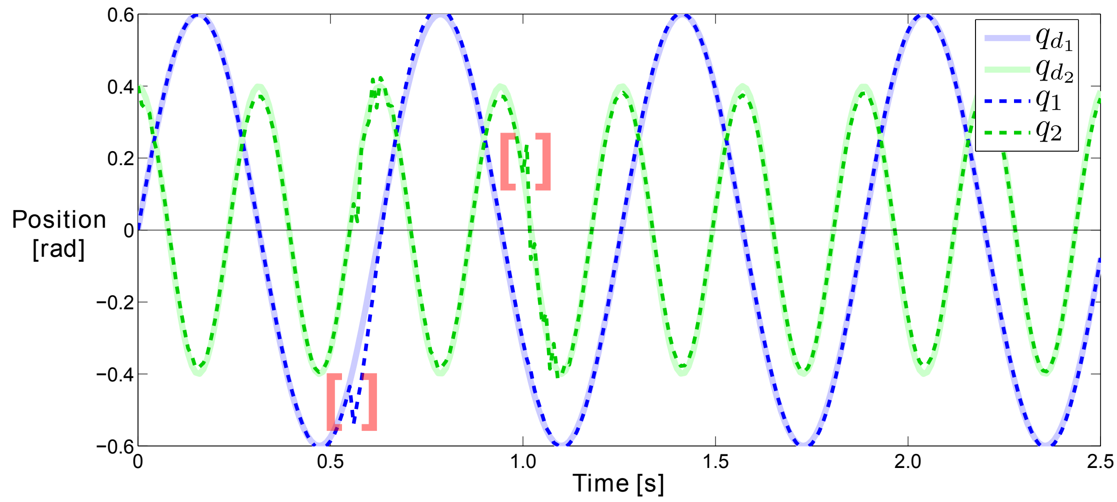


Figure 3.15: PID control of double pendulum, disturbance behavior.

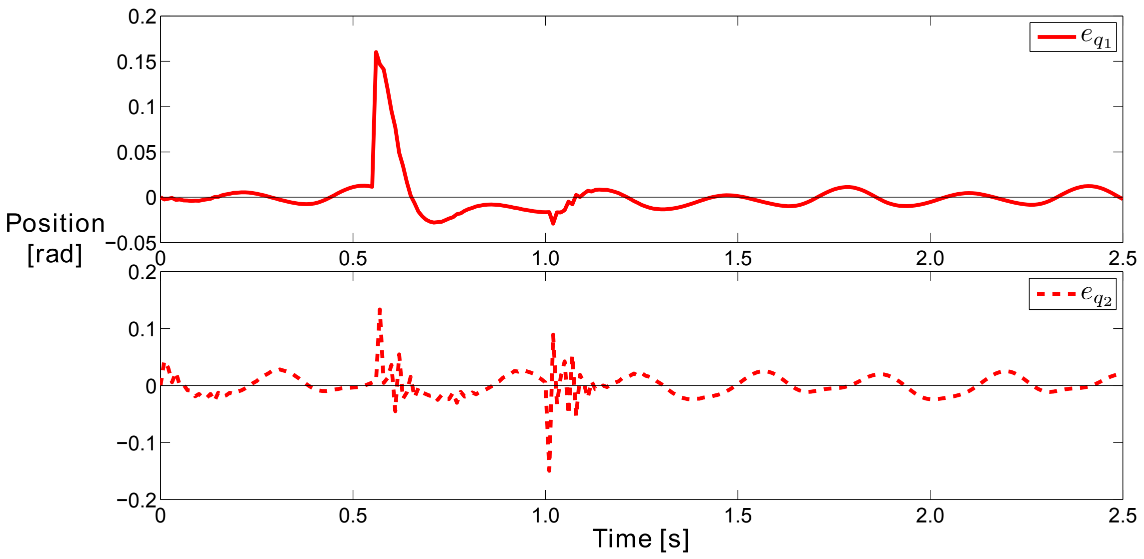


Figure 3.16: Position error of the PID controller

3.3.5 Challenges of PID Control

In many cases the control strategies that are based on the PID control design are considered to have solid control properties while still having relatively simple structure. But this simplicity holds also several problems in use of the PID controls. Often the difficulties are caused by the system itself which is to be controlled. This section gives a small overview of some such problems and describes possible solutions.

PID Gain Tuning

Stability analysis requires that the matrices K_p, K_d and K_i are positive definite. But the identification of the practical values of the matrices is a non-trivial task. Often the finding of the values is a simple trial and error method. There are several tools that can be used for identification of the gain matrices. For example step response, Bode plot or Nyquist plot are common additives for obtaining the values. Ziegler and Nichols [ZN42] as well as Chien et al. [CHR52] have introduced several rules which are commonly used for tuning the gain parameters. The listed methods, however, are suitable for specified systems whose structure is already known. The autonomous multi-body robotic systems can change their structure depending on the number and composition of the particular modules. Thus the gain parameters also should be estimated autonomously. Åström and Hägglund [ÅH84] as well as Tor Steinar Schei [Sch92] have proposed methods for automatic tuning of the PID controllers which are based on phase and amplitude margins of the system to be controlled.

Integrator Windup

Applying the integral component in the PID control law also may cause problems that can result in the instability of the controller. Most real systems are bounded in their movement and so the most robotic systems have only a limited operating range. Each Scout or Backbone module has a revolute joint that can operate only in the range $[-90°, 90°]$ in its surface of revolution relatively to its null position. Actually, the maximal angles are limited to $\pm 45°$ by software to avoid possible damages. Using the integral component for control of a bounded system may results in growing of the summarized error while the measured values do not change due to the operating limits. This can be avoided by limiting the integral component so that the summarized error remains unchanged if the measured values is stopped by its limits.

Computing Time

The software implementation of a common PID control is very simple and are not dependent on high computational capabilities. Some variations of the PID control, however, may include expressions which can require more time for calculation. For example gravity compensated or augmented PD controls need the matrices $M(q)$, $C(q, \dot{q})$ and $N(q)$ to be calculated in each loop run. This would take approximately the same time as for the system evaluation, thus the calculation time of the each loop run wold be doubled at least. Considering that the calculation time grows exponentially with the number of modules included in the robotic organism, the computing time of the control law evaluation may be critical. This can be avoided by using the once calculated matrices in both expressions: control law (Equation 3.20) and system evaluation (Equation 3.2). The more detailed implementation description is given in the Chapter 4.

3.4 Local Stabilization

The idea consists in Jacobian linearization of the system in a certain point of operation to obtain a linear calculation rule for the system input. The assumption is that the linearized input signal can sufficiently fit the control of the nonlinear system for small state variations about the operating point. Equilibrium states are mostly used as operating points. Linearization of the system and input dynamics in the operating point are evaluated as follows:

$$A = \frac{\partial f}{\partial x}|_{x=x_s}, \qquad b = g(x_s) \overset{!}{\neq} 0, \tag{3.23}$$

where A is the Jacobian linearization of the system function $f(x)$. Both values A and b are calculated at the point of operation x_s. The condition $b \neq 0$ ascertains that the system remains controllable in the rest position as well. Using the pair $\{A, b\}$ the nonlinear system in the Equation 3.4 is approximately described in the neighborhood of the operating point x_s by the linear equivalent:

$$\dot{x} = Ax + b\tau. \tag{3.24}$$

It is helpful for further analyze to assume that the operation point x_s lies in zero. In the case that the real equilibrium state differs from zero point, the system can be transferred into one with $x_s = 0$ using the following state transformation:

$$x^* = x - x_s, \tag{3.25}$$

where x^* is the state of the new system which operating point lies in zero.

There are several conditions that a nonlinear system must satisfy, so that a linear control is possible. The formulation of those conditions requires some definitions to be made before.

Definition 3.4.1 (Controllability)
*A linear time-invariant system is called **controllable** if a control function $u(t)$ exists that transfers the system from any initial state into any final state in any time.*

The controllability of a system that is represented by the Equation 3.24, can be determined by the Hautus-Test as described in the following theorem:

Theorem 3.4.1 (Hautus-Test for Controllability)
*The pair $\{A, b\}$ is called **controllable** if and only if the following holds*

$$rank(\lambda I - A|b) = n \quad \forall \lambda \in \mathbb{C},$$

where n is the system dimension and λ denotes the eigenvalues of A.

Definition 3.4.2 (Stabilizability)
*A system is called **stabilizable** if a control function $u(t)$ exists that transfers the system in the zero state (not necessarily in finite time).*

Using the upper Definitions the following prerequisite for local stabilization can be formulated:

Theorem 3.4.2 (Local Stabilization)
If the pair $\{A, b\}$ is controllable then the linear controller that is based on $\{A, b\}$ is stabilizable.

In the next steps the dynamic 3.2 system is tested for controllability. First the Jacobian matrix A of the function $f(x)$ (Equation 3.7) and the input matrix b at the operating point $x_s = 0$ are evaluated as follows:

$$\begin{aligned} A &= \frac{\partial f}{\partial x}|_{x=0} \qquad (3.26) \\ &= \begin{bmatrix} 0 & I \\ \frac{\partial[-M(x_1)^{-1}(C(x_1,x_2)x_2+N(x_1))]}{\partial x_1}|_{x=0} & \frac{\partial[-M(x_1)^{-1}(C(x_1,x_2)x_2+N(x_1))]}{\partial x_2}|_{x=0} \end{bmatrix} \\ &= \begin{bmatrix} 0 & I \\ -\frac{\partial}{\partial x_1}[M(x_1)^{-1}N(x_1)]|_{x=0} & -M(0)^{-1}C(0,0) \end{bmatrix} \in \mathbb{R}^{2n\times 2n}, \end{aligned}$$

$$\begin{aligned} b &= g(0) \\ &= \begin{bmatrix} 0 \\ M(0)^{-1} \end{bmatrix} \in \mathbb{R}^{2n \times n}. \end{aligned} \tag{3.27}$$

Since the original dynamic system (Equation 3.2) is multivariable, the dimension of the square Jacobian matrix is equal the doubled number the joint modules. Now the Hautus-Test (Theorem 3.4.1) can be applied for the pair $\{A, b\}$:

$$\begin{aligned} &rank[\lambda I - A \mid b] \\ &= \left[\begin{array}{cc|c} \lambda I & -I & 0 \\ \frac{\partial}{\partial x_1}[M(x_1)^{-1}N(x_1)]|_{x=0} & \lambda I + M(0)^{-1}C(0,0) & M(0)^{-1} \end{array}\right] \end{aligned} \tag{3.28}$$

The matrices $-I$ and $M(0)^{-1}$ have both the rank n independently of λ or each other. Thus all row vectors of $[\lambda I - A \mid b]$ are linear independent so that the matrix has the full rank:

$$rank[\lambda I - A \mid b] = 2n \ \ \forall \lambda \in \mathbb{C}. \tag{3.29}$$

Hence, there is a stable linear control design based on the values of A and b. The block diagram of the system with a linear controller is depicted in the Figure 3.17, where

$$A' = \left[-\frac{\partial}{\partial q}[M(q)^{-1}N(q)]|_{q=0} \quad -M(0)^{-1}C(0,0)\right]. \tag{3.30}$$

$$b' = M(0), \tag{3.31}$$

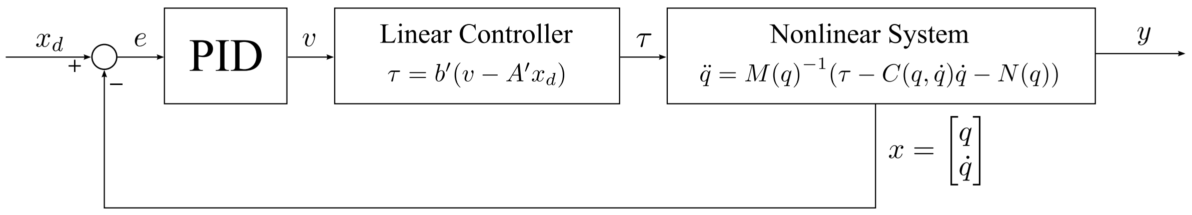

x_d
+
−
e
PID
v
Linear Controller
$\tau = b'(v - A'x_d)$
τ
Nonlinear System
$\ddot{q} = M(q)^{-1}(\tau - C(q,\dot{q})\dot{q} - N(q))$
y
$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$

Figure 3.17: Local Stabilization

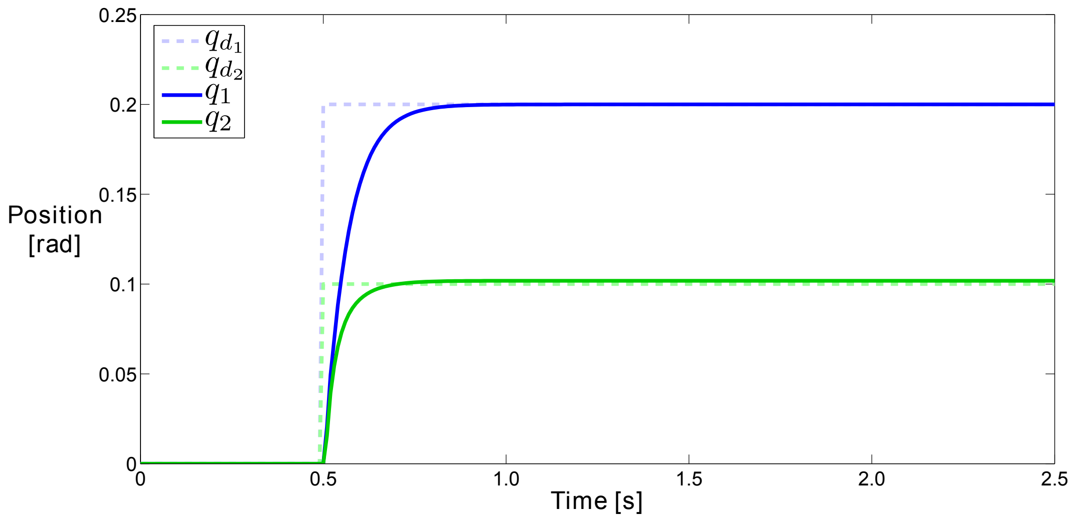


Figure 3.18: Step response of locally stabilized double pendulum, small amplitude.

Often linear controls which are based on the Jacobian linearization run well enough in the neighborhood of an operating point. But there is no guarantee for the global stability. The Figures 3.18 and 3.19 demonstrate the step responses of double pendulum with local stabilization. As expected, the controlled system have optimal stability properties in the close neighborhood of the equilibrium state $x_s = 0$ (Figure 3.18). But with growing amplitudes the steady-state error grows significantly (Figure 3.19).

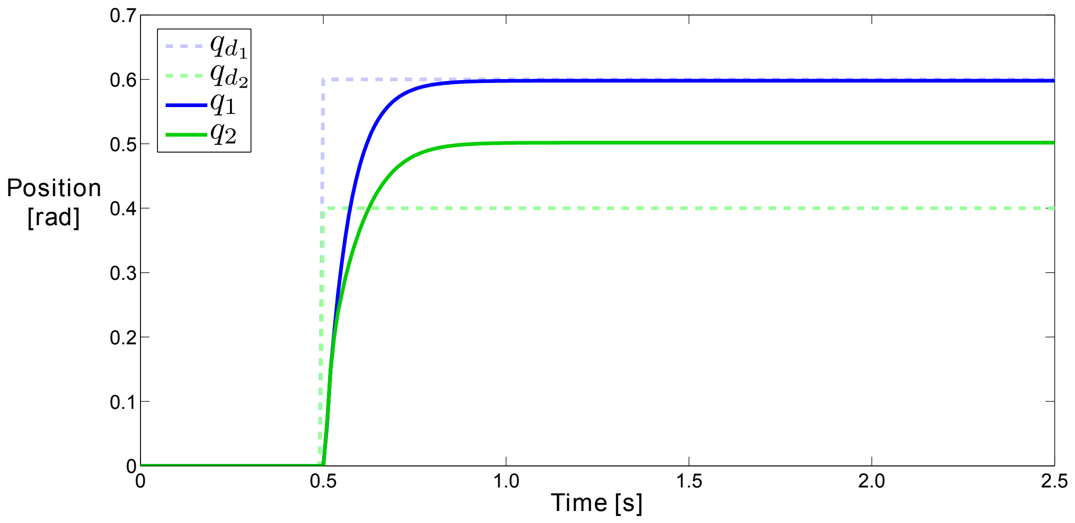


Figure 3.19: Step response of locally stabilized double pendulum, large amplitude.

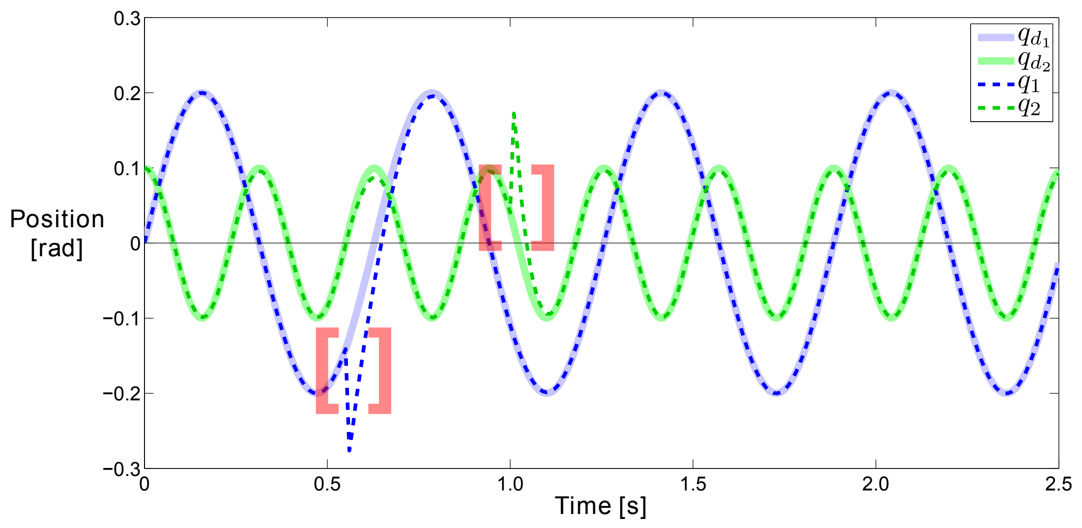


Figure 3.20: Trajectory of locally stabilized double pendulum, small amplitude.

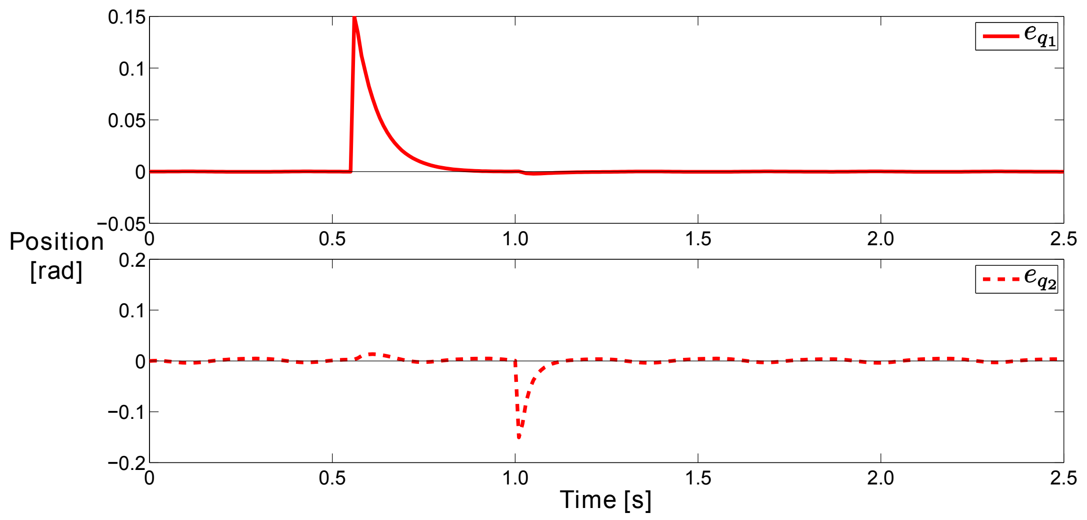


Figure 3.21: Tracking error of locally stabilized double pendulum, small amplitude.

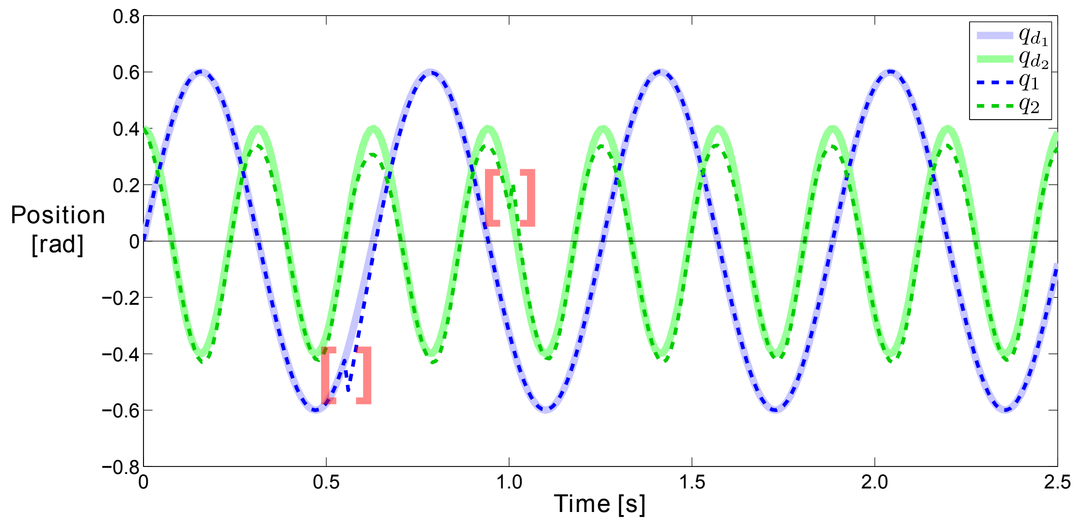


Figure 3.22: Trajectory of locally stabilized double pendulum, large amplitude.

A similar behavior can be observed during the tracking of a desired trajectory. The trajectories with different amplitudes are depicted in the Figures 3.20 and 3.22 and the correspondent trajectory errors in the Figures 3.21 and 3.23. As can be seen, in the close neighborhood of the equilibrium state the accurateness of the tracking of the desired trajectory is very high. On the other side the error that occurs during the larger deflections potentially may not be neglected. However, the local stabilized system shows solid robustness against the outside influence. The disturbances at times $t_1 = 0.55$ s and $t_2 = 1.0$ s are being corrected in the exponential manner.

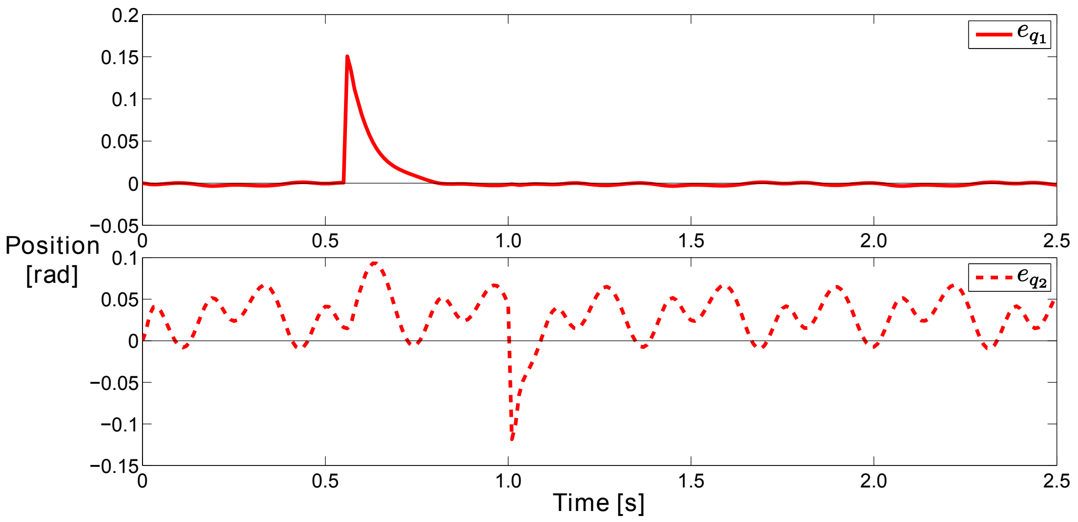


Figure 3.23: Tracking error of locally stabilized double pendulum, large amplitude.

The method of local stabilization can be considered for set-point regulations as well as for trajectory tracking on condition that the system is moved in the close neighborhood of the equilibrium sate. In that case the controlled system shows relatively solid robustness and precision. For larger amplitudes the linearization does not match the real system what leads to an inaccuracy in the regulation. Depending on the system requirements the resulting error may not be neglected.

3.5 Computed Torque

In the section 3.2 an open-loop control method is introduced. The main disadvantage of the approach is the lack of any system feedback which results in the fact that each motion error remains uncorrected. This section describes a control design that use a similar control law but includes a system feedback for calculation of the control signal. Consider the following control law with the system state feedback

$$\tau = M(q)\ddot{q}_d + C(q,\dot{q})\dot{q} + N(q). \tag{3.32}$$

With this control input the dynamic system (Equation 3.1) becomes

$$M(q)\ddot{q} = M(q)\ddot{q}_d. \tag{3.33}$$

Since the mass matrix is always positive definite, the equation simplifies to

$$\ddot{q} = \ddot{q}_d. \tag{3.34}$$

This means that if the initial positions and velocities of the particular modules matches the desired positions and velocities, the joint modules will follow the desired trajectory which is given by $\ddot{q}_d$. In the form given by the Equation 3.32 the control law is still vulnerable to the initial conditions errors and outside disturbances. The control law can be made more robust by adding the system state feedback to the calculation of the desired acceleration $\ddot{q}_d$:

$$v = \ddot{q}_d + K_d(\dot{q}_d - \dot{q}) + K_p(q_d - q) = \ddot{q}_d + K_d\dot{e} + K_p e, \tag{3.35}$$

where $e = q_d - q$ and K_d and K_p are positive definite gain matrices, analogously to the PD control. v describes the new value that is used for calculation of the control input instead of $\ddot{q}_d$. Therewith the control law becomes

$$\begin{aligned}\tau &= M(q)v + C(q,\dot{q})\dot{q} + N(q) \\ &= M(q)(\ddot{q}_d + K_d\dot{e} + K_p e) + C(q,\dot{q})\dot{q} + N(q).\end{aligned} \tag{3.36}$$

The Equation 3.36 describes the so called *computed torque* control law. Thus the error dynamics of the closed-loop dynamic system is described as follows

$$M(q)(\ddot{e} + K_d\dot{e} + K_p e) = 0. \tag{3.37}$$

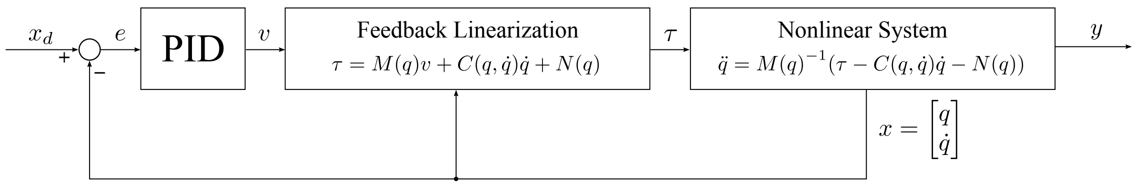
Since $M(q)$ is positive definite in q, the equation simplifies to

$$\ddot{e} + K_d\dot{e} + K_p e = 0. \tag{3.38}$$

Murray et al. [MSZ94] have shown that the computed torque control law (Equation 3.36) applied to the dynamic system (Equation 3.1) results in exponential trajectory tracking if the matrices K_d and K_p are positive definite The equation of the computed torque control law can be written as follows

$$\tau = \underbrace{M(q)\ddot{q}_d + C(q,\dot{q})\dot{q} + N(q)}_{\tau_{\mathrm{ff}}} + \underbrace{M(q)(K_d\dot{e} + K_p e)}_{\tau_{\mathrm{fb}}}. \tag{3.39}$$

Therewith the control law of computed torque method can be considered as a composition of two components. The term τ_{ff} is the *feedforward* component. It provides the amount of torque necessary to drive the system along the desired trajectory. The term τ_{fb} specifies the *feedback* component. It provides the correction torques for reducing any error in the trajectory of modules [MSZ94]. The block diagram of the computed torque approach is shown in the Figure 3.24.



x_d
$+$
$-$
e
PID
v
Feedback Linearization
$\tau = M(q)v + C(q,\dot{q})\dot{q} + N(q)$
τ
Nonlinear System
$\ddot{q} = M(q)^{-1}(\tau - C(q,\dot{q})\dot{q} - N(q))$
y
$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$

Figure 3.24: Computed Torque Method

The set-point and tracking properties of the computed torque control can be seen in the Figures 3.25, 3.26 and 3.27. Double pendulum controlled by computed torque strategy is resistant to steady-state errors and shows a solid robustness against the outside influences and errors in initial conditions. The outside disturbances simulated on the double pendulum are marked in the Figure 3.26. The trajectory error (Figure 3.27) which is caused by those disturbances, is corrected to the desired value exponentially.

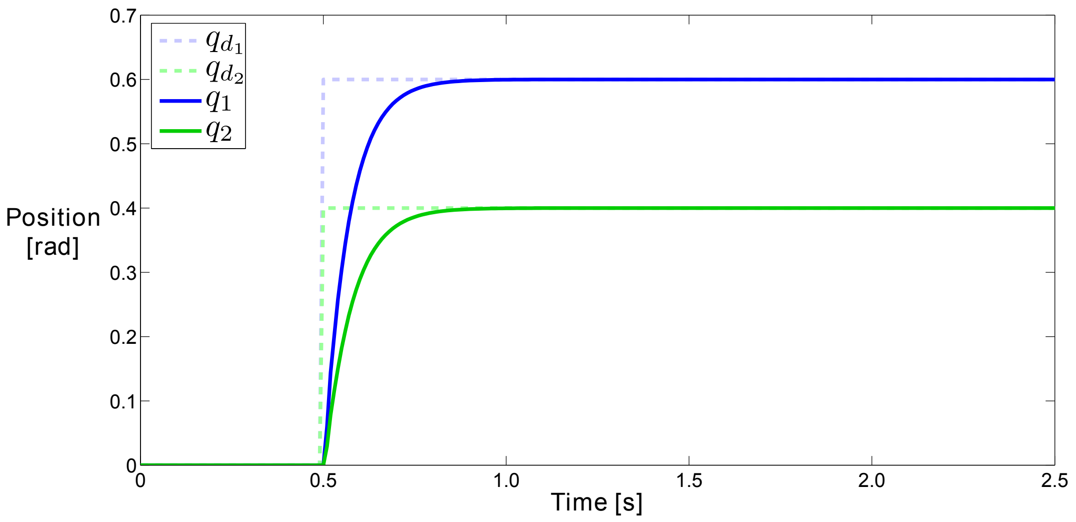


Figure 3.25: Step response of double pendulum with computed torque control.

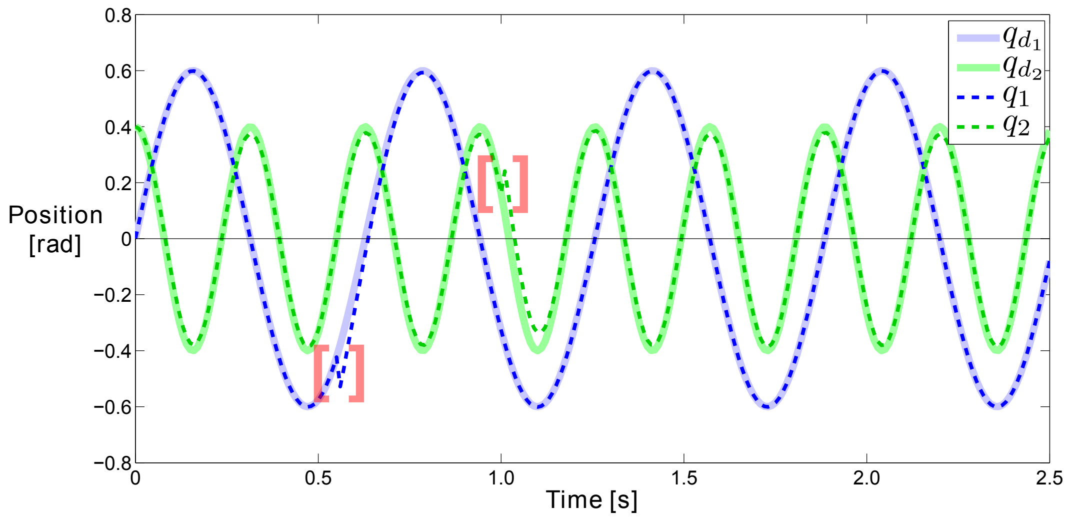


Figure 3.26: Trajectory of double pendulum with computed torque control, disturbance behavior.

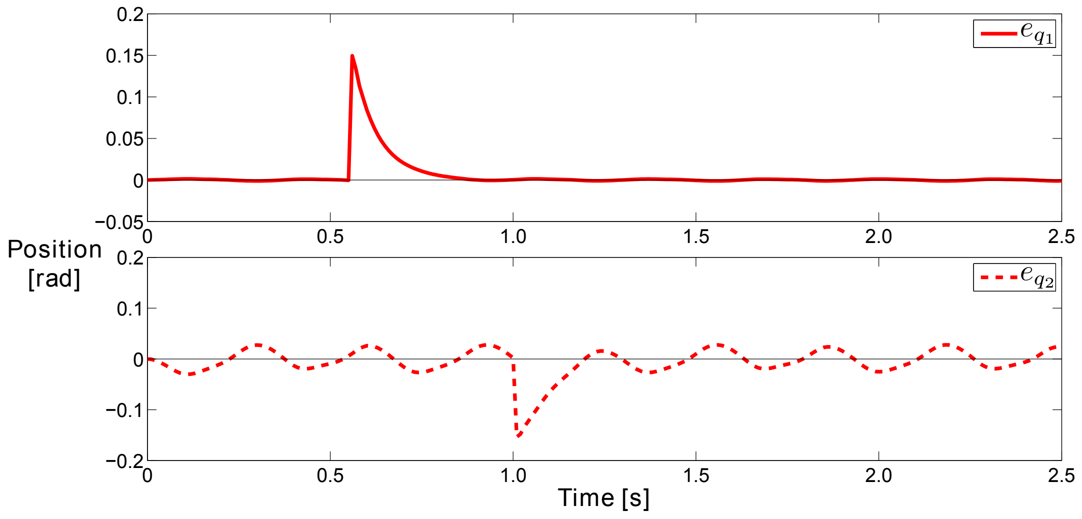


Figure 3.27: Trajectory error of double pendulum with computed torque control, disturbance behavior.

The computed torque strategy is an example of a more general control technique, the so called *feedback linearization*, where a nonlinear system is rendered linear via full-state nonlinear feedback. A large advantage of this technique is that it converts a nonlinear dynamic system into a linear one and thus allows the use of any synthesis tool known from the linear control theory [MSZ94]. An important note about the computed torque control is that it requires an accurate knowledge about the parameters of the dynamical model. In addition, the control input is computed in real time which can be problematic for large multi-body robotic organisms.

3.6 Extended Kalman Filter

The control strategies described in the previous sections are based on the full-state feedback of the dynamic system. This requires that all state variable must be measurable anytime. But often some state values can not be measured according to missing sensors or other reasons. In this case the missing state variables must be estimated via so called *observer*. Based on the knowledge of the measurable state variables and the system input and output an observer calculates the missing state values which are used later in the control computation.

Each robot module, Backbone or Scout, has a revolute joint with one degree of freedom. The absolute hinge angles of the modules can be measured using the hall sensors. But there are no sensors for measuring of the angular velocities of the revolute joints. Thus an observer is necessary to obtain the full set of states of the nonlinear system model. One of the promising approaches to reconstruct the missing or noisy states of the dynamic system is the *Kalman filter*. The Kalman filter estimates the system dynamics using a form of feedback control. The operation cycle of the Kalman filter can be devided in two steps. In the first step the filter estimates the system state. The estimation quality is then improved in the second step using the measurements of the feedback. The corrected state estimate is used then in the first step of the next loop for preceding state estimation. The first step of the Kalman filter is called *time update* or *prediction step*, the second step is the *measurement update* or *correction step*. Since the classical Kalman filter is proposed to deal with linear systems, Welch and Bishop [WB95] have introduced the *extended Kalman filter* (EKF) for use with the nonlinear dynamic systems. The EKF use the Jacobian linearization of the dynamic model of the system and the one of the system feedback. This section describes the functionality of the EKF.

Assuming that the nonlinear discrete-time dynamic model is given in the state space representation:

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}) \in \mathbb{R}^n, \tag{3.40}$$

$$y_k = h(x_k, v_k) \in \mathbb{R}^m, \tag{3.41}$$

where x is the state vector of the system. Te time step is given by k. Thus the nonlinear function f relates the state x at time step k to the state at time step $k+1$. Further parameters of the function f are the control input u and the zero-mean process noise w. The nonlinear function h provides the measurement vector y according to the current state and the measurement noise v. The zero-mean noises w and v are considered as independent of each other, white, and with normal probability distributions:

$$p(w) \sim N(0, Q), \tag{3.42}$$
$$p(v) \sim N(0, R), \tag{3.43}$$

where $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ denote the covariance matrices for measurement error and system noise respectively. Since the particular values of w and v are not known, the state and measurement vectors can be approximated by neglecting w and v:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0), \tag{3.44}$$
$$\hat{y}_k = h(\hat{x}_k^-, 0), \tag{3.45}$$

where $\hat{x}_k^-$ is defined as *a priori* state estimate at time step k and $\hat{x}_k$ as *a posteriori* state estimate from the previous time step. The linearization of the dynamic system (3.40, 3.41) about the estimates $\hat{x}_{k+1}^-$ and $\hat{y}_k$ is formulated as follows

$$x_k \approx \hat{x}_k^- + A(x_{k-1} - \hat{x}_{k-1}) + W w_{k-1}, \tag{3.46}$$
$$y_k \approx \hat{y}_k + H(x_k - \hat{x}_k^-) + V v_k, \tag{3.47}$$

where

- x_k and y_k are the current state and measurement vectors,
- $\hat{x}_k^-$ and $\hat{y}_k$ are the approximate state and measurement vectors,
- $\hat{x}_k$ is an *a posteriori* estimate of the state at step k,
- w_k and v_k are zero-mean process and measurement noises as described in the Equations 3.42 and 3.43,
- A is the Jacobian matrix of partial derivatives of f relative to x:

  $$A_{ij} = \frac{\partial f_i}{\partial x_j}(\hat{x}_{k-1}, u_{k-1}, 0) \in \mathbb{R}^{n \times n}$$

- W is the Jacobian matrix of partial derivatives of f relative to w:

  $$W_{ij} = \frac{\partial f_i}{\partial w_j}(\hat{x}_{k-1}, u_{k-1}, 0) \in \mathbb{R}^{n \times n}$$

- H is the Jacobian matrix of partial derivatives of h relative to x:

$$H_{ij} = \frac{\partial h_i}{\partial x_j}(\hat{x}_k^-, 0) \in \mathbb{R}^{m \times n}$$

- V is the Jacobian matrix of partial derivatives of h relative to v:

$$V_{ij} = \frac{\partial h_i}{\partial v_j}(\hat{x}_k^-, 0) \in \mathbb{R}^{m \times m}$$

Since the linearizations about $\hat{x}_k^-$ and $\hat{y}_k$ are performed in each time step, the matrices A_k, W_k, H_k and V_k are in general not constant and must be calculated in each time step anew.

Further the a priori state error and the measurement residual are defined as follows

$$\hat{e}_{x_k}^- \equiv x_k - \hat{x}_k^-, \tag{3.48}$$
$$\hat{e}_{y_k} \equiv y_k - \hat{y}_k. \tag{3.49}$$

Since the actual state vector x_k is the value to be estimated by the EKF, there is no possibility to use it directly as defined in the Equation 3.48. As already mentioned above, the system output or rather its measured value can be used for estimation of x_k. Substituting the linearized state and measurement vectors (Equations 3.46 and 3.47) in the Equations 3.48 and 3.49 the error dynamics is described as follows

$$\hat{e}_{x_k}^- \approx A(x_{k-1} - \hat{x}_{k-1}) + \epsilon_{k-1}, \tag{3.50}$$
$$\hat{e}_{y_k} \approx H\hat{e}_{x_k}^- + \eta_k, \tag{3.51}$$

where ϵ_{k-1} and η_k are independent, zero-mean, random variables with the covariance matrices WQW^T and VRV^T. The measurement residual $\hat{e}_{y_k}$ can be used in a separate process for estimation of the prediction state error $\hat{e}_{x_k}^-$. Using the Equation 3.48 the *a posteriori* state estimate can be obtained in the original nonlinear process as follows

$$\hat{x}_k = \hat{x}_k^- + \hat{e}_{x_k}, \tag{3.52}$$

where $\hat{e}_{x_k}$ is an estimation of the actual prediction error $\hat{e}_{x_k}^-$. The estimation of $\hat{e}_{x_k}$ is performed by the following Kalman filter equation

$$\hat{e}_{x_k} = K_k \hat{e}_{y_k}, \tag{3.53}$$

where the matrix $K_k \in \mathbb{R}^{n \times m}$ represent the factor that is intended to minimize the *a posteriori* error covariance $P_k = E[\hat{e}_{x_k}, \hat{e}_{x_k}^T] \in \mathbb{R}^{n \times n}$. Substituting the Equation 3.53 back into Equation 3.52 and using the Equation 3.49 results in the following Equation for *a posteriori* state estimate

$$\begin{aligned} \hat{x}_k &= \hat{x}_k^- + K_k \hat{e}_{y_k} \\ &= \hat{x}_k^- + K_k(y_k - \hat{y}_k) \end{aligned} \tag{3.54}$$

The calculation of the matrix K that was introduced by Welch and Bishop has the following form

$$K_k = \frac{P_k^- H_k^T}{H_k P_k^- H_k^T + V_k R_k V_k^T}, \tag{3.55}$$

where $P_k^- = E[\hat{e}_{x_k}^-, \hat{e}_{x_k}^{-T}]$ denotes the *a priori* estimate error covariance. It is obvious that in case of small measurement error covariance R_k the gain K weights the residual more heavily and especially,

$$\lim_{R_k \to 0} K_k = H_k^{-1}.$$

In case the *a priori* estimate error P_k^- approaches zero, the residual is weighted less heavily:

$$\lim_{P_k^- \to 0} K_k = 0.$$

The detailed derivations of K can be found in the works of Maybeck et al. [MS80], Brown and Hwang [BH92] or Jacobs [Jac93].

Now the particular equations of the extended Kalman filter can be divided in two groups as described in the beginning of the section. The time update equations provide the *a priori* estimates $\hat{x}_k^-$ and P_k^-:

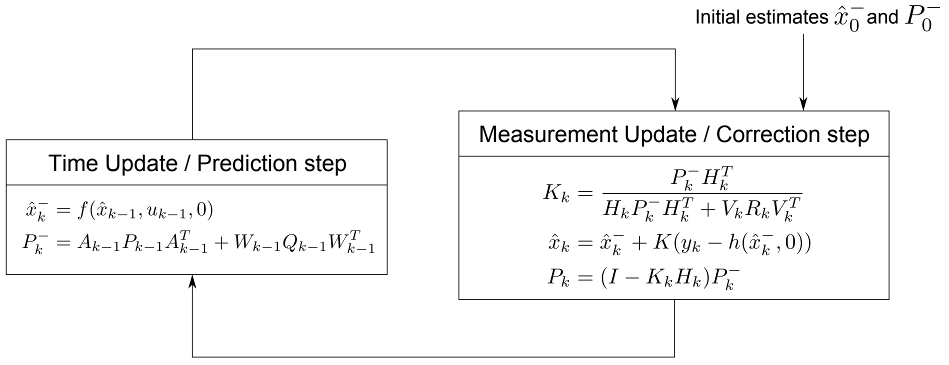
$$\begin{aligned} \hat{x}_k^- &= f(\hat{x}_{k-1}, u_{k-1}, 0), \\ P_k^- &= A_{k-1} P_{k-1} A_{k-1}^T + W_{k-1} Q_{k-1} W_{k-1}^T \end{aligned} \tag{3.56}$$

Afterward the values $\hat{x}_k^-$ and P_k^- are used in the measurement update equations to obtain the improved estimates $\hat{x}_k$ and P_k:

$$K_k = \frac{P_k^- H_k^T}{H_k P_k^- H_k^T + V_k R_k V_k^T} \tag{3.57}$$

$$\hat{x}_k = \hat{x}_k^- + K(y_k - h(\hat{x}_k^-, 0)) \tag{3.58}$$

$$P_k = (I - K_k H_k) P_k^- \tag{3.59}$$

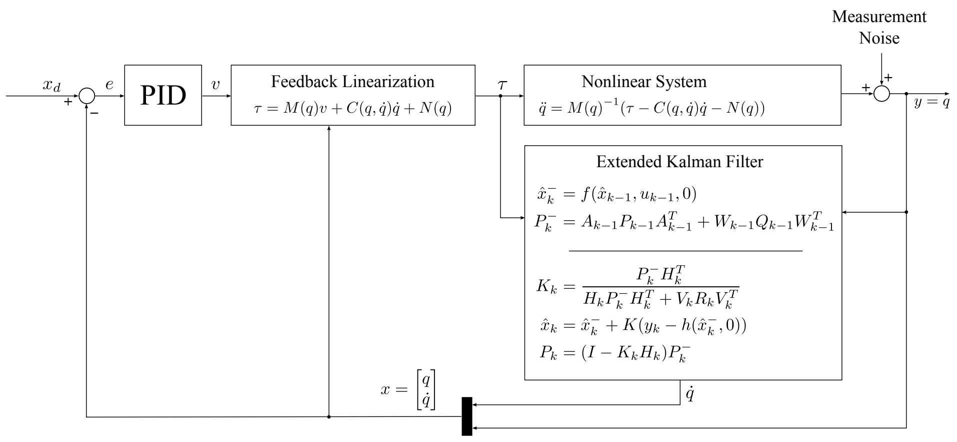


Initial estimates $\hat{x}_0^-$ and P_0^-
Time Update / Prediction step
$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0)$
$P_k^- = A_{k-1}P_{k-1}A_{k-1}^T + W_{k-1}Q_{k-1}W_{k-1}^T$
Measurement Update / Correction step
$K_k = \frac{P_k^- H_k^T}{H_k P_k^- H_k^T + V_k R_k V_k^T}$
$\hat{x}_k = \hat{x}_k^- + K(y_k - h(\hat{x}_k^-, 0))$
$P_k = (I - K_k H_k)P_k^-$

Figure 3.28: Functionality schema of the extended Kalman filter.

Figure 3.28 demonstrate the described steps of the operation loop of the EKF with the appropriate equations.

Because of the nonlinearity of the observed system, the matrices A_k, H_k, W_k and V_k in general are different in each calculation run of the EKF. Therefore they must be calculated at each time step k anew. In many cases the measurement error and system noise covariance matrices, R_k and Q_k respectively, are constant and can be obtained in advance using the knowledge about the system and sensor properties. But often R_k and Q_k depends on the system dynamics and must be recomputed at each time step like other parameters.



Measurement Noise
x_d
e
PID
v
Feedback Linearization
$\tau = M(q)v + C(q,\dot{q})\dot{q} + N(q)$
τ
Nonlinear System
$\ddot{q} = M(q)^{-1}(\tau - C(q,\dot{q})\dot{q} - N(q))$
$y = q$
Extended Kalman Filter
$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0)$
$P_k^- = A_{k-1}P_{k-1}A_{k-1}^T + W_{k-1}Q_{k-1}W_{k-1}^T$
$K_k = \frac{P_k^- H_k^T}{H_k P_k^- H_k^T + V_k R_k V_k^T}$
$\hat{x}_k = \hat{x}_k^- + K(y_k - h(\hat{x}_k^-, 0))$
$P_k = (I - K_k H_k)P_k^-$
$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$
$\dot{q}$

Figure 3.29: Extended Kalman Filter

Since the EKF serves only for reconstruction of the missing states, it can not replace a control mechanism. For example, the EKF can be used with a computed torque method to control a dynamic system. A combination of the dynamic model, EKF, PID and computed torque control is represented in the Figure 3.29. A step response of a simulated double pendulum with the computed torque control is shown in the Figure 3.30. The measurement of the system output is affected by a zero-mean white noise with normal probability distribution $\mathcal{N}(0, 0.0003)$, where the variance $\sigma^2 = 0.0003$ is set for the standard deviation to be mostly in the in the interval of $\pm 3°$. The measurement noise graph is depicted in the Figure 3.31.

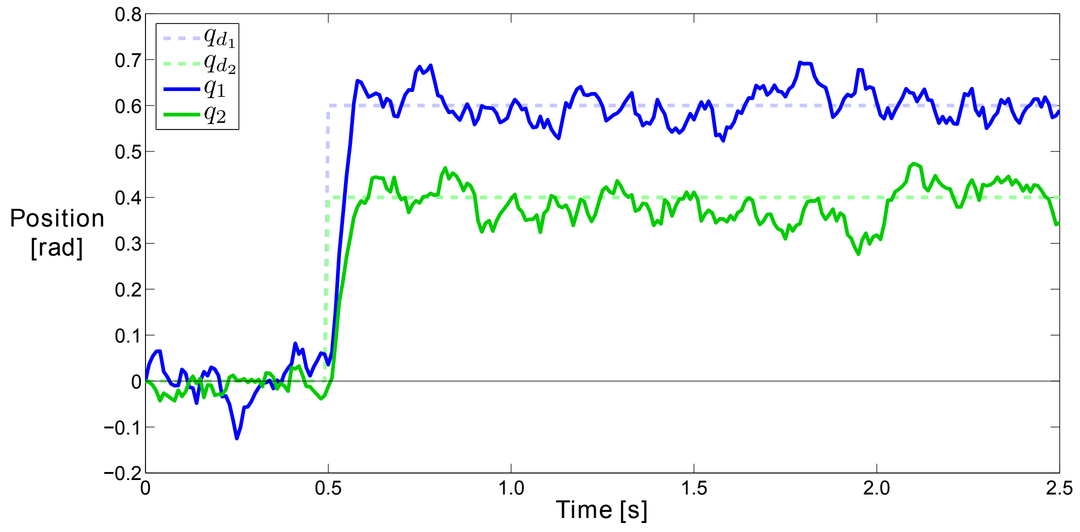


Figure 3.30: Step response of double pendulum with EKF and computed torque.

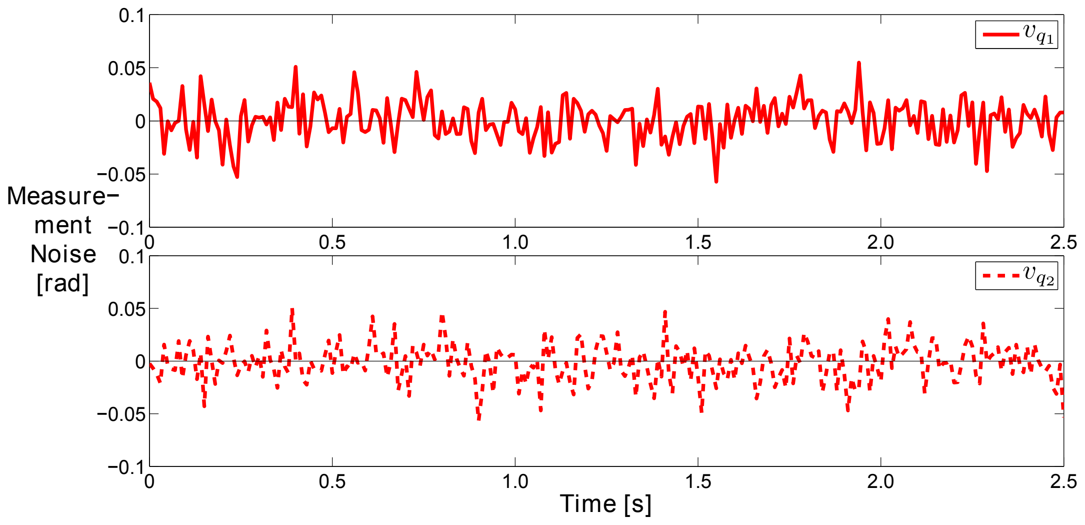


Figure 3.31: Simulated noise obtained during the measurement of the system output.

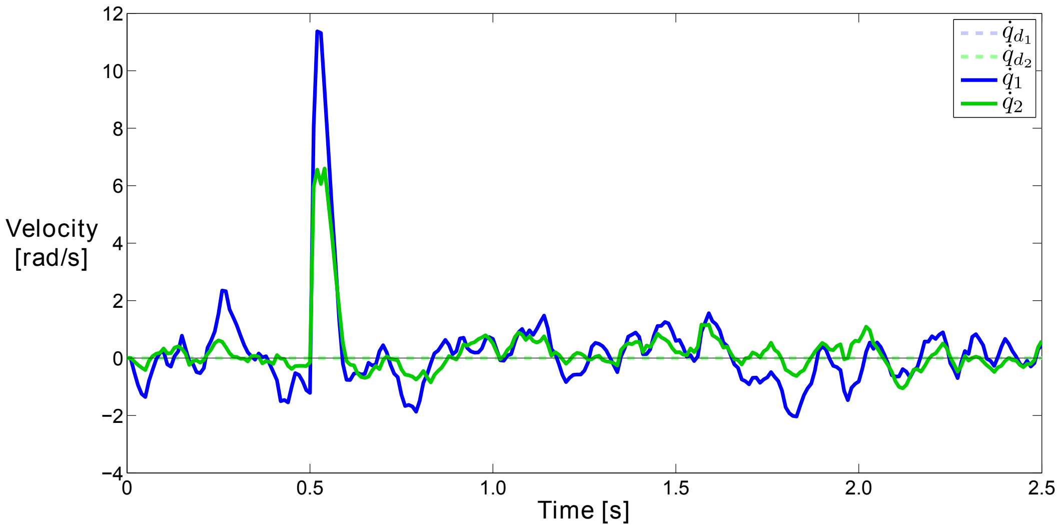


Figure 3.32: Module velocities of double pendulum calculated by EKF.

The velocities of the modules of double pendulum are are represented in the Figure 3.32 and the difference between the velocities which are calculated by the system model and velocities provided by EKF, is shown in the Figure 3.33.

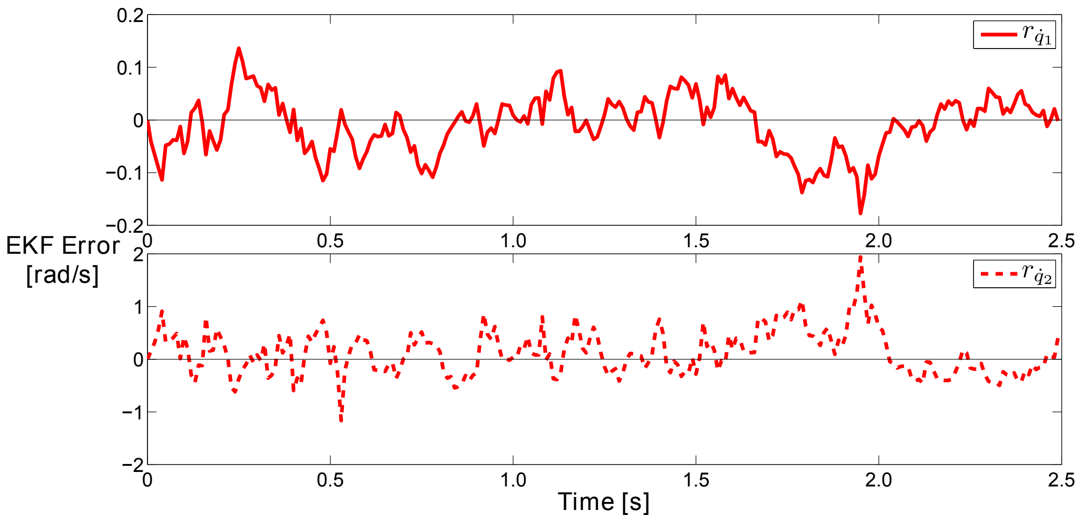


Figure 3.33: The difference between the modeled velocities and the velocities calculated by EKF: $r = \dot{q} - \dot{q}_{EKF}$.

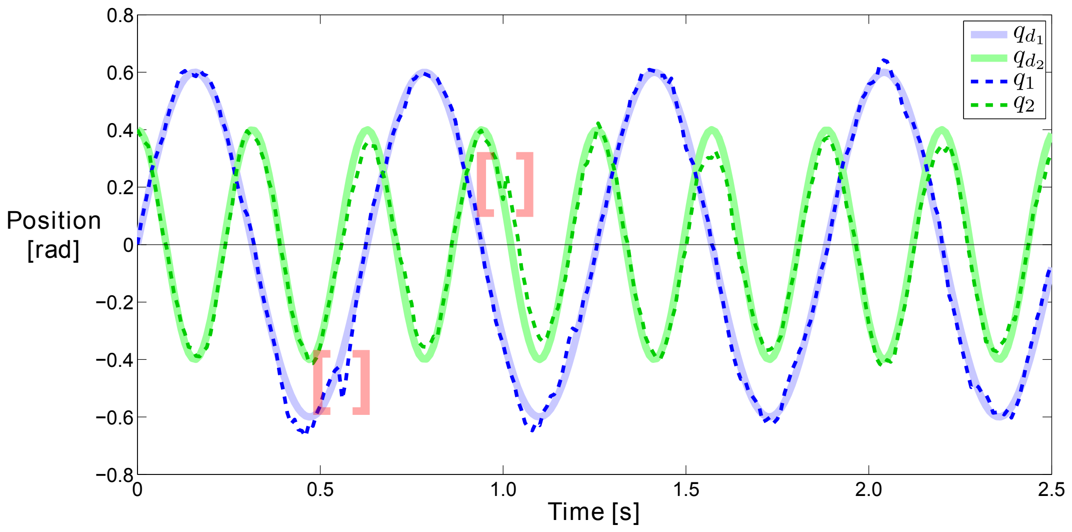


Figure 3.34: Trajectory tracking of double pendulum with EKF and computed torque control.

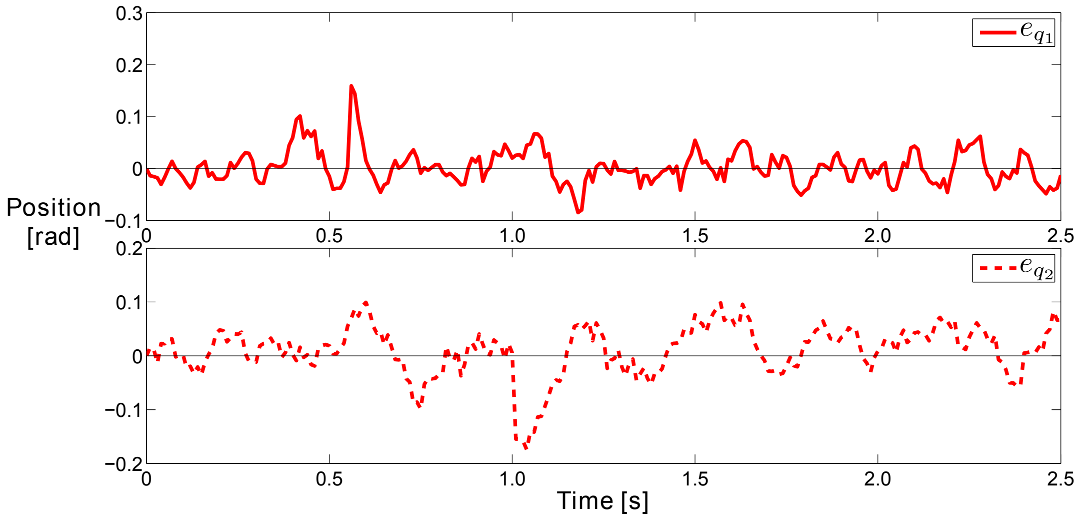


Figure 3.35: Trajectory error of double pendulum with EKF and computed torque control.

The tracking of the desired trajectory of the double pendulum is shown in the Figure 3.34. The marked places denote again the simulated disturbances of 0.15 and -0.15 radian. The trajectory error of double pendulum with the EKF is depicted in the Figure 3.35. The parameters Q, W, R and V are chosen as constant in these examples with the following values

$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0001 \end{bmatrix}, \tag{3.60}$$

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.61}$$

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.62}$$

$$V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.63}$$

The initial estimate $\hat{x}_0^-$ is set to the value of the initial state of the system model in each example. The initial value P_0^- is set in all examples as

$$P = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0001 \end{bmatrix}. \tag{3.64}$$

Apart from the impreciseness caused by the measurement noise the extended Kalman filter provides suitable properties needed for reconstruction of missing or noised states. The accuracy of the EKF can be improved by a higher tuning of the parameters Q, W, R and V adapting them in each time step instead of using constant values.

4 Implementation

This section describes the translation of the control strategies which are described in the previous section, into a software implementation. The idea is that the implemented control designs are accessible for a modular robotic organism. The robotic system should be able to autonomously establish a control strategy depending on its current dynamic model. Therewith the control step can be added to the procedure for the generation of the dynamic model that was introduced by Chen and Yang [CY98]. The updated diagram of the procedure is shown in the Figure 4.1.

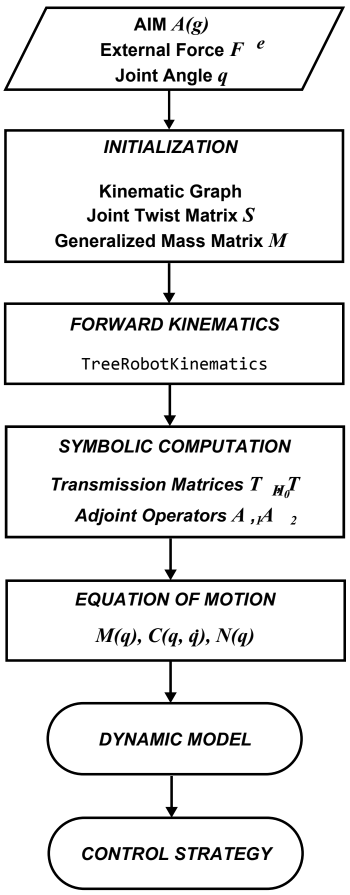


Figure 4.1: Dynamic model generation with control.

A C++ software framework for generation of the dynamic model of a self-reconfigurable robotic system was already implemented [Nos13]. At this point the existing framework should be extended with components which are needed for control realization. Before the implementation of the control mechanisms is described, several general changes in the current version of the framework are proposed. The previous version of the framework was based on the computer algebra system *SymbolicC++* [Sym10]. Computer algebra systems allows the execution of symbolic calculations which, in contrast to the numeric ones, are performed on variables that may not have a certain value at the time of the calculation. On the one hand, symbolic calculations can possibly simplify computational process before the real values are used. In the procedure of the dynamic model generation the equation of motion (Equatinon 2.30) must be derived once only. On the other hand, the time for the symbolic derivation of the motion equations grows exponentially with dimension of the system. Thus time that is required for large robotic structures can amount several hours and more. Using numerical calculations The equation of motion must be derived each time when the system is evaluated. This also requires more time for large topologies of robotic structures. But the calculation time can be reduced by use of the Blackfin digital signal process (DSP) library [Bla13]. The DSP library provides functions for basic vector and matrix operations which are optimized for use with the Blackfin microprocessors. The robot modules Scout and Backbone which are used in this project, are equipped with a Blackfin CPU, thus it make sense to use the DSP library for vector and matrix calculations of the framework.

Due to the changeover to the DSP library the basic classes `Vector` and `Matrix` and its basic operations are redefined in the new version of the framework. The Listings 4.1 and 4.2 illustrate the internal structure of the classes `Vector` and `Matrix`.

```
class Vector {
public:
 // constructors
 Vector(void);
 explicit Vector(const unsigned int n);
 Vector(const unsigned int n, const double data[]);
 Vector(const unsigned int n, const double d);
 Vector(const Vector & v);
 // destructor
 ~Vector();

 void swap(Vector & v) throw();

 // operators
 Vector & operator = (const Vector & v);
 Vector operator + (const Vector & v) const;
```

```
  Vector & operator += (const Vector & v);
  Vector operator - (const Vector & v) const;
  Vector operator * (const double d) const;
  Vector operator / (const double d) const;
  double operator | (const Vector & v) const; // dot product
  Vector operator % (const Vector & v) const; // cross product
  double operator [] (const unsigned int i) const;
  double & operator [] (const unsigned int i);

  unsigned int size(void) const;

  std::ostream & output(std::ostream & os) const;

  friend class Matrix;

protected:
  unsigned int dim;
  double * data;
};
```

Listing 4.1: Class Vector

```
class Matrix {
public:
  // constructors
  Matrix(void);
  Matrix(const unsigned int r, const unsigned int c);
  Matrix(const unsigned int r, const unsigned int c, const double data[]);
  Matrix(const unsigned int r, const unsigned int c, const double d);
  Matrix(const Matrix & m);
  // destructor
  ~Matrix();

  void swap(Matrix & m) throw();

  // operators
  Matrix & operator = (const Matrix & m);
  Matrix operator + (const Matrix & m) const;
  Matrix & operator += (const Matrix & m);
  Matrix operator - (const Matrix & m) const;
  Matrix operator * (const double d) const;
  Matrix operator * (const Matrix & m) const;
  Vector operator * (const Vector & v) const;
  double operator () (const unsigned int r, const unsigned int c) const;
  double & operator () (const unsigned int r, const unsigned int c);

  static Matrix identity(const unsigned int dim);

  Matrix transpose(void) const;
  Matrix inverse(void) const;

  unsigned int rows(void) const;
  unsigned int cols(void) const;

  std::ostream & output(std::ostream & os) const;

  friend class Vector;

protected:
  unsigned int rows_;
  unsigned int cols_;
  double * data;
};
```

Listing 4.2: Class Matrix

The examples shown in the Listing 4.3 demonstrate the use of the DSP library. The functions `vecvadd`, `vecdot` and `matsmlt` perform the operations for vector addition, dot product and matrix-scalar multiplication respectively. The integration of the DSP library proceeds very easily because the library functions and the classes `Vector` and `Matrix` operate on the same primitive data objects.

```
// vector-vector addition
Vector Vector::operator +(const Vector & v) const {
  assert(this->dim == v.dim);
  Vector result(this->dim);
  vecvadd(this->data, v.data, result.data, this->dim);
  return result;
}

// dot product
double Vector::operator |(const Vector & v) const {
  assert(this->dim == v.dim);
  return vecdot(this->data, v.data, this->dim);
}

// matrix-scalar multiplication
Matrix Matrix::operator *(const double d) const {
  Matrix result(this->rows_, this->cols_);
  matsmlt(this->data, d, this->rows_, this->cols_, result.data);
  return result;
}
```

Listing 4.3: Example use of the DSP library

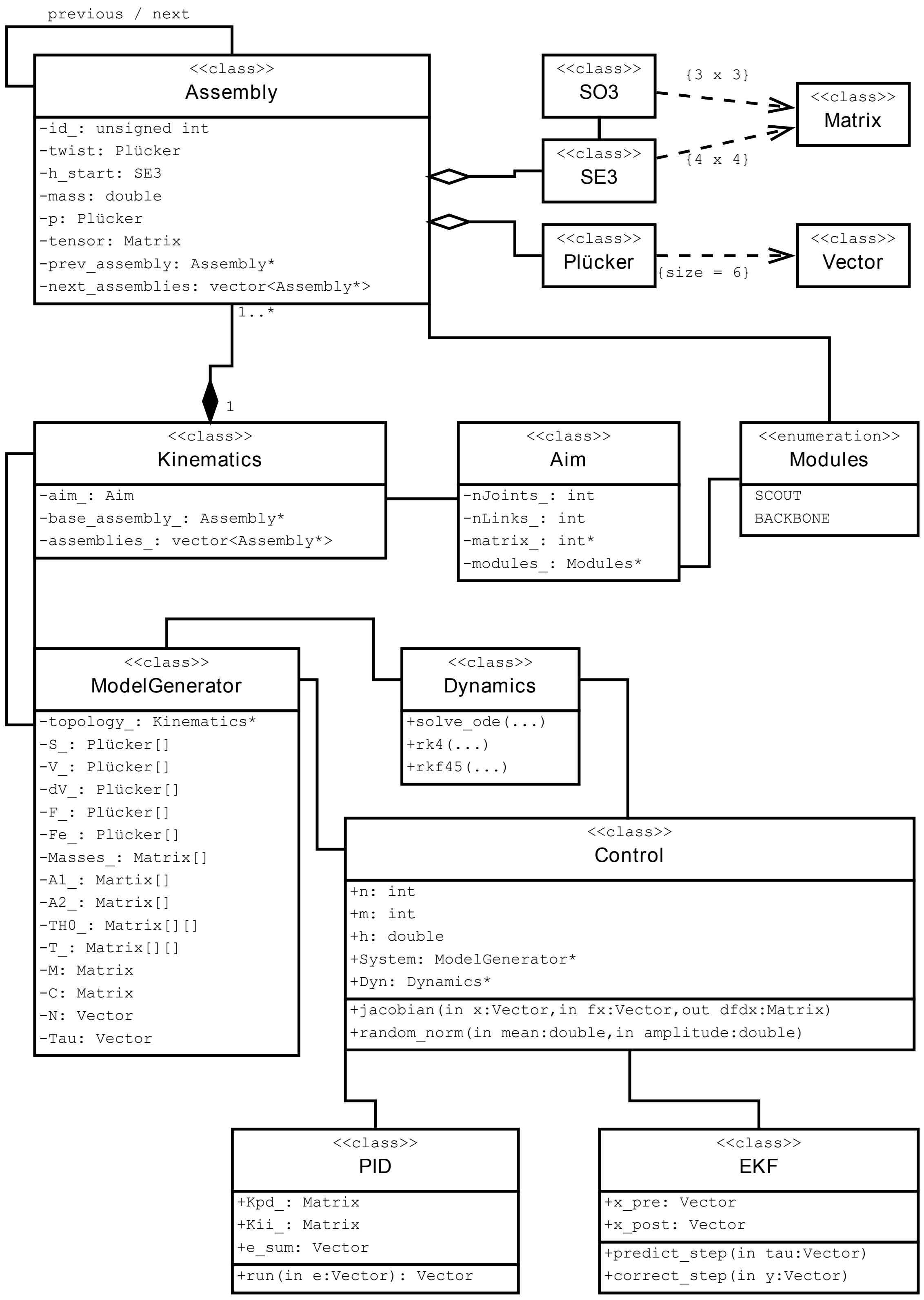


Figure 4.2: The class diagram of the C++ framework

Since the classes `Vector` and `Matrix` provide the similar functionality as the previous symbolic implementations, the other classes of the framework did not have to be greatly modified. The extension of the existing software framework that provides the control realization are the classes `PID`, `EKF` and `Control`. These and the other classes of the framework with their associations are illustrated in the class diagram shown in the Figure 4.2.

As the names suggest the classes `PID` and `EKF` represent the particular control tools. The PID controller is implemented by the class `PID`. The class saves the gain matrices K_p, K_d and K_i and use them for signal processing. The Listing 4.4 shows the structure of the `PID` class.

```
class PID {
public:
  PID(const Matrix & Kp, const Matrix & Kd, const Matrix & Ki);
  ~PID();

  Vector run(const Vector & e, const Vector & de);

private:
  Vector e_sum;
  Matrix Kp_;
  Matrix Kd_;
  Matrix Ki_;
};
```

Listing 4.4: PID controller

Depending on the values of the particular gain matrices the class implements different controller types. Table 4.1 lists the possible combination.

Controller type	K_p	K_d	K_i
P	> 0	$= 0$	$= 0$
I	$= 0$	$= 0$	> 0
PI	> 0	$= 0$	> 0
PD	> 0	> 0	$= 0$
PID	> 0	> 0	> 0

Table 4.1: Type of the PID controller depending on the values of the gain matrices.

Other combinations are possible but not reasonable, since there are no appropriate control realizations.

The numerous parameter which are used for state estimation in the extended Kalman filter are stored in the class `EKF`. The two steps of the estimation process of the EKF are implemented in the functions `predict_step` and `correct_step` which are represented in the Listings 4.5 and 4.6 respectively.

```
void EKF::predict_step(const Vector & u)
{
 Vector dx(x_pre.size());

 System->f(x_post, u, dx);
 Dynamics->rk4(x_post, dx, u, h, x_pre);

 Control->jacobian(x_post, u, dx, A);
 A = A * h;
 P = A * P * A.transpose() + W * Q * W.transpose();
}
```

Listing 4.5: Prediction Step of EKF

```
void EKF::correct_step(const Vector & z)
{
 K = P * H.transpose() * (H * P * H.transpose() + V * R * V.transpose()).inverse();
 x_post = x_pre + K * (z - H * x_pre);
 P = (I - K * H) * P;
}
```

Listing 4.6: Correction Step of EKF

Since the particular control tools are available for use, a mechanism is necessary which would allow an ordered combination of them to obtain a desired closed-loop control design. The class `Control is intended` for such task. The control strategies which are described in this thesis, can be generalized by the diagram depicted in the following Figure.

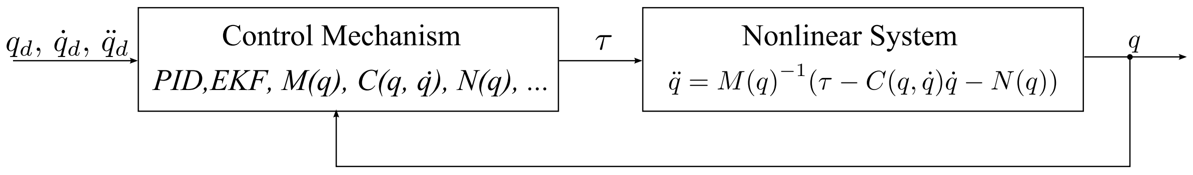
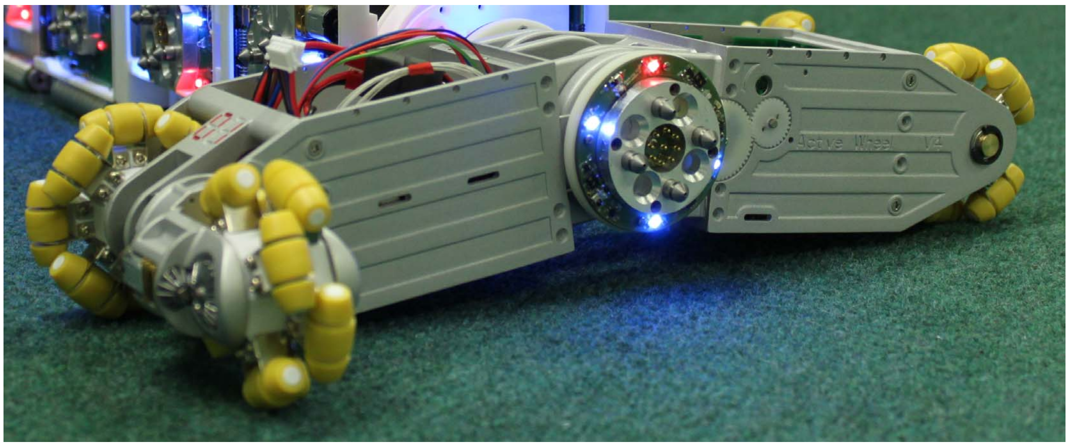

$q_d,\ \dot{q}_d,\ \ddot{q}_d$
Control Mechanism
PID,EKF, M(q), C(q, q̇), N(q), ...
τ
Nonlinear System
$\ddot{q} = M(q)^{-1}(\tau - C(q,\dot{q})\dot{q} - N(q))$
q

Figure 4.3: Schematic diagram of the control mechanism.

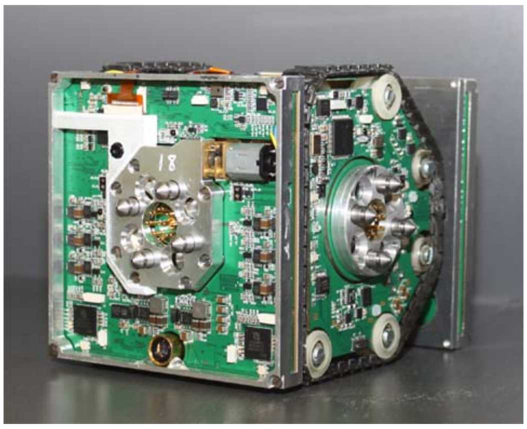
The class `Control` provides the relevant functionality for realization of the certain control mechanisms. It allow the use of the several control tools like PID controller or extended Kalman filter. Additionally `Control` is associated with the classes `ModelGenerator` and `Dynamics` which allow the calculation of the desired control signal τ, provided that the state $[q,\ \dot{q}]^T$ is measured respectively estimated. The class is provided with the currnet values of the desired motion trajectories q_d, $\dot{q}_d$ and $\ddot{q}_d$. The desired trajectory is then compared with the current measurements/estimates q and $\dot{q}$. Afterward the resulting trajectory error is propagated for the calculation of the desired control signal τ. This step may contain the computations of several temporary values like the signal v in the computed torque method. At least the the control signal τ is sent to the controlled system. Hence, the class `Control` allows the building of a control mechanism that is represented as a dynamic system with the inputs q_d, $\dot{q}_d$, $\ddot{q}_d$, q and $\dot{q}$ and the output τ.

5 Practical Results

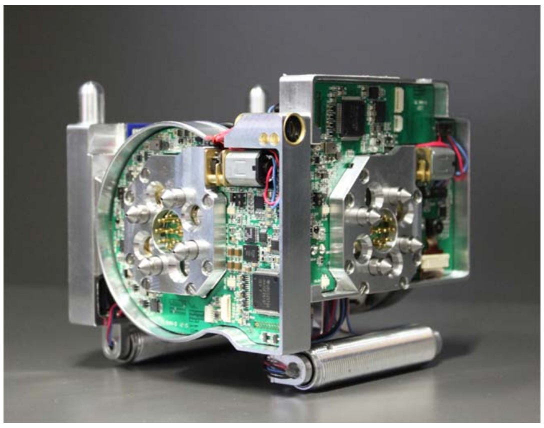
This chapter summarizes the experience which was gained during the tests with the real robots. The tests have been performed on the robotic modules developed in the European research projects SYMBRION [SYM12] and REPLICATOR [REP12]. As results of the both projects three types of the robotic modules have been created: *Active Wheel* 5.1a, *Scout* 5.1b and *Backbone* 5.1c.



(a) Active Wheel [KSH+11].



(b) Scout [MGL13]



(c) Backbone [MGL13].

Figure 5.1: Robotic modules developed in the projects SYMBRION and REPLICATOR.

Since the Active Wheels in contrast to Scout and Backbone are developed to transport the other modules and not for building the multibody organisms, they are not relevant to the purposes described in this thesis. Both cube-shaped modules, Scout and Backbone, have a revolute joint with one degree of freedom. They have four uniform docking elements on the each side in the horizontal plane: front, rear, left and right. The top and bottom sides are not equipped with docking elements because this would prohibit the rotational motion. The docking elements allow an Ethernet connection between the assembled modules thus the robots can communicate with each other. The communication between the modules in the tests have been provided by the IPC protocol that is implemented within the *irobot* middleware. Irobot provides the basic functionality for hardware regulation like operating the actuators, obtaining sensor data or Ethernet communication.

5.1 Preparations

The same test software have been running on each module of the example topologies. Before the execution can be started the base module must be determined. In the real environments it would be performed by the modular robot autonomously but in the tests the base have been selected by hand. In order for the proper communication each module have obtained its own IP address.

For the tests a two-dimensional snake-like topology and a H-shaped four-legged organism have been chosen. All the topologies have been composed of the Backbone modules. In both tests the robotic system have been controlled using the computed torque method. The system output have been estimated by the extended Kalman filter.

5.2 2-D Snake-Like Robot

The schematic composition of the snake-like example robot is represented in the Figure 5.2. The modules of the snake have been ordered so that all revolute joints have operated in the same X-Y-plane. Each module have been provided with a sinus signal with a constant difference in the phase as the desired motion trajectory.

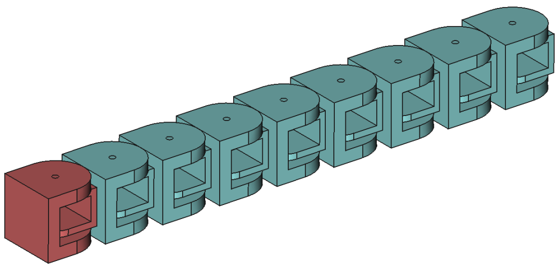


Figure 5.2: Schematic of the two-dimensional snake-like modular organism. The base module is denoted red.

The desired trajectory have been calculated by the base module. Te particular signals then have been provided to the each slave module (including base itself) using the IPC protocol and the Ethernet connection. After the estimation of the current angles and applying the control functions each module send the computed control signal to the irobot middleware for execution.

Each joint module and the multibody robotic organism as a whole have shown solid tracking results. Since the operating plane of the modules have lied orthogonally to the gravitational forces, the gravity have should not be greatly compensated. On the other hand the friction between the modules and the surface have could not be neglected due to the great mass of the whole organism. The friction disturbance have could be reduced by equipping several modules with small wheels.

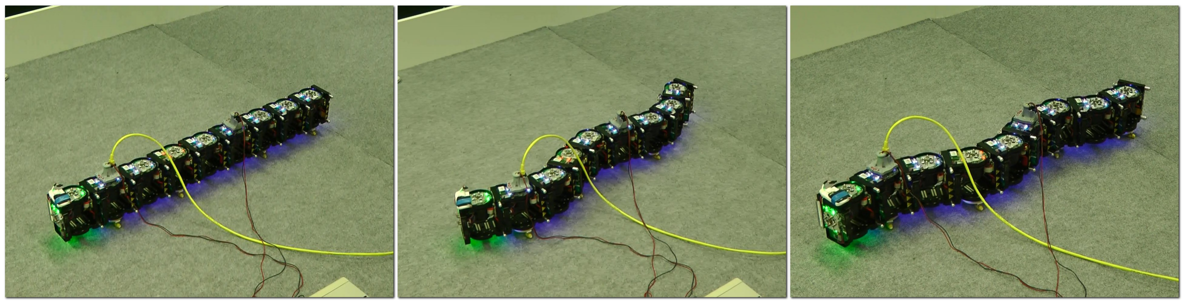


Figure 5.3: 2-D snake-like robotic organism.

The Figure 5.3 demonstrate the movement of the snake-like robotic organism at several times. The sinus-like movement of the organism is clearly recognizable.

5.3 Four-Legged Robot

In this example an other movement type of the modular robotic organism have been tested. To be movable a four-legged (Figure 5.4) robot has to have at least two orthogonally placed joints in each leg. Additionally the organism requires a synchronization mechanism to move the legs in the appropriate order. At this point the base have calculated the different signals for vertical and horizontal movements of the legs. For simplicity the diagonally opposite legs have been moved simultaneously.

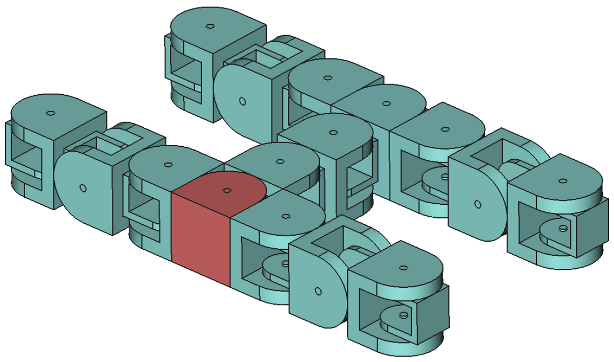


Figure 5.4: Schematic of the four-legged modular organism. The base module is denoted red.

The movement of the four-legged modular robot is shown in the Figure 5.5. In contrast to the previous example the gravitational forces have affected the motion of the robot to a great extent, since the four vertical joints in the legs have had to bear the weight of the complete organism. Apart from that the organism has executed the desired movements only with a small inaccuracy.

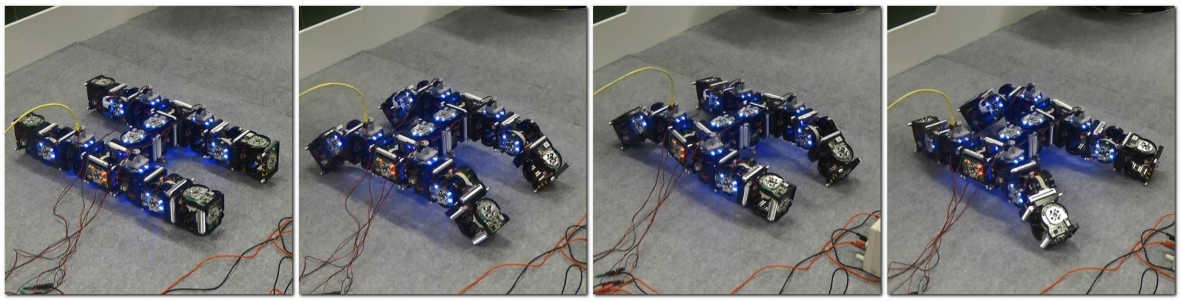


Figure 5.5: Four-legged robot.

6 Conclusion

This thesis has given an overview about the use of the method of the classical control theory for dynamic systems which are characterized by the modularity of their structures. The presented control methods have been added to the C++ software framework that have been developed in the previous work [Nos13]. The evaluation of the particular control strategies has been performed by simulation in MATLAB and by the execution of the test with the real modular robotic organisms. Therefore the robot have been used which had been developed in the projects SYMBRION and REPLICATOR. The results have shown the solid control properties of the applied strategies. Although many operations like parameter tuning have must be done by hand now, the future multibody robotic organisms will be able to derive a suitable control strategy for a given configuration completely autonomous.

Bibliography

[ÅH84] K. Åström, T. Hägglund. Automatic tuning of simple regulators with specifications on phase and amplitude margins. *Automatica*, 20(5):645 – 651, 1984. doi:http://dx.doi.org/10.1016/0005-1098(84)90014-1. URL http://www.sciencedirect.com/science/article/pii/0005109884900141. (Cited on page 39)

[Ari84] S. Arimoto. Stability and robustness of PID feedback control for robot manipulators of sensory capability. *Robotics Research,1st Int. Symp.*, pp. 783–799, 1984. URL http://ci.nii.ac.jp/naid/80014492902/en/. (Cited on page 36)

[BH92] R. G. Brown, P. Y. Hwang. *Introduction to random signals and applied Kalman filtering*, volume 1. John Wiley & Sons New York, 1992. (Cited on page 53)

[Bla13] Blackfin docs. Blackfin DSP library, 2013. URL http://blackfin.uclinux.org/doku.php?id=toolchain:libbfdsp. (Cited on page 62)

[CHR52] K. Chien, J. Hrones, J. Reswick. On the Automatic Control of Generalized Passive Systems. *Transactions of the ASME*, 74:175–185, 1952. (Cited on page 39)

[CY98] I.-M. Chen, G. Yang. Automatic Model Generation for Modular Reconfigurable Robot Dynamics. *Journal of Dynamic Systems, Measurement, and Control*, 120, 1998. (Cited on pages 10, 20 and 61)

[Jac93] O. L. R. Jacobs. *Introduction to control theory*, volume 2. Oxford University Press Oxford, 1993. (Cited on page 53)

[Kel97] R. Kelly. PD Control with Desired Gravity Compensation of Robotic Manipulators A Review. *The International Journal of Robotics Research*, 16(5):660–672, 1997. (Cited on page 33)

[KSH$^+$11] S. Kernbach, F. Schlachter, R. Humza, J. Liedke, S. Popesku, S. Russo, T. Ranzani, L. Manfredi, C. Stefanini, R. Matthias, C. Schwarzer, B. Girault, P. Alschbach, E. Meister, O. Scholz. Heterogeneity for Increasing Performance and Reliability of Self-Reconfigurable Multi-Robot Organisms. *IROS11, workshop on "Reconfigurable Modular Robotics", San Francisco, 2011*, 2011. (Cited on page 69)

[KUK12] KUKA Roboter GmbH. KR 300-2 PA. http://www.kuka-robotics.com/NR/rdonlyres/6DA192D5-C840-4957-ADC2-4060E2F8F47F/0/PR_KR300_2_PA_01.jpg, 2012. *(Last visited on November 14th 2012)*. (Cited on page 9)

[MG12] E. Meister, A. Gutenkunst. Self-Adaptive Framework for Modular and Self-Reconfigurable Robotic Systems. In *ADAPTIVE 2012, The Fourth International Conference on Adaptive and Self-Adaptive Systems and Applications*, pp. 30 – 37. 2012. (Cited on page 15)

[MGL13] E. Meister, A. Gutenkunst, P. Levi. Dynamics and Control of Modular and Self-Reconfigurable Robotic Systems. *International Journal on Advances in Intelligent Systems*, 6(1&2):66–78, 2013. (Cited on pages 11 and 69)

[MS80] P. S. Maybeck, G. M. Siouris. Stochastic Models, Estimation, and Control, Volume I. *Systems, Man and Cybernetics, IEEE Transactions on*, 10(5):282–282, 1980. doi:10.1109/TSMC.1980.4308494. (Cited on page 53)

[MSZ94] R. M. Murray, S. S. Sastry, L. Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1994. (Cited on pages 13, 14, 16, 28, 35, 47 and 50)

[NAS12] NASA. Curiosity. http://www.nasa.gov/images/content/657465main_pia15791-43_800-600.jpg, 2012. *(Last visited on November 14th 2012)*. (Cited on page 9)

[Nos13] E. Nosov. *Analysis and Implementation of Geometrical Formulation of Dynamic Equations on Embedded Devices*. Diploma thesis, University of Stuttgart, 2013. (Cited on pages 11, 23, 62 and 73)

[REP12] REPLICATOR. *REPLICATOR: Robotic Evolutionary Self-Programming and Self-Assembling Organisms, 7th Framework Programme Project No FP7-ICT-2007.2.1*. European Communities, 2008-2012. (Cited on page 69)

[Sch92] T. S. Schei. A method for closed loop automatic tuning of {PID} controllers. *Automatica*, 28(3):587 – 591, 1992. doi:http://dx.doi.org/10.1016/0005-1098(92)90182-F. URL http://www.sciencedirect.com/science/article/pii/000510989290182F. (Cited on page 39)

[Sel05] J. Selig. Lie Groups and Lie Algebras in Robotics. In J. Byrnes, editor, *Computational Noncommutative Algebra and Applications*, volume 136 of *NATO Science Series II: Mathematics, Physics and Chemistry*, pp. 101–125. Springer Netherlands, 2005. doi:10.1007/1-4020-2307-3_5. URL http://dx.doi.org/10.1007/1-4020-2307-3_5. (Cited on pages 15 and 16)

[Sic08] B. Siciliano, editor. *Springer handbook of robotics: with 84 tables*. Springer, Berlin, 2008. URL http://swbplus.bsz-bw.de/bsz263176339kap.htm. (Cited on pages 29, 30 and 33)

[Sym10] http://issc.uj.ac.za/symbolic/symbolic.html, 2010. *(Last visited on November 29th 2012)*. (Cited on page 62)

[SYM12] SYMBRION. *SYMBRION: Symbiotic Evolutionary Robot Organisms, 7th Framework Programme Project No FP7-ICT-2007.8.2*. European Communities, 2008-2012. (Cited on page 69)

[WB95] G. Welch, G. Bishop. An Introduction to the Kalman Filter, 1995. (Cited on page 50)

[YSS$^+$07] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, G. S. Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *Robotics & Automation Magazine, IEEE*, 14(1):43–52, 2007. (Cited on pages 5 and 10)

[ZN42] J. Ziegler, N. Nichols. Optimum settings for automatic controllers. *Transactions of the ASME*, 64(11), 1942. (Cited on page 39)

All links were last followed on January 9, 2014.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature