# Control Primitives for Fast Helicopter Maneuvers

by

Barış Eren PERK

Submitted to the Department of Mechanical Engineering Department in partial fulfillment of the requirements for the degrees of

Master of Science in Mechanical Engineering

and

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

[September 2006]

August 2006

Author ........[illegible]........................................
Department of Mechanical Engineering Department
August 11, 2006

Certified by ...........................................................
J.J.E. Slotine
Professor of Mechanical Engineering and Information Sciences
Professor of Brain and Cognitive Sciences
Thesis Supervisor

Accepted by ...........................................................
Lallit Anand
Chairman, Department Committee on Graduate Students

# Control Primitives for Fast Helicopter Maneuvers

by

Barış Eren PERK

Submitted to the Department of Mechanical Engineering Department
on August 11, 2006, in partial fulfillment of the
requirements for the degrees of
Master of Science in Mechanical Engineering
and
Master of Science in Electrical Engineering and Computer Science

## Abstract

In this paper, we introduce a framework for learning aggressive maneuvers using dynamic movement primitives (DMP) for helicopters. Our ultimate goal is to combine these DMPs to generate new primitives and demonstrate them on a 3-DOF (3 Degrees of Freedom) helicopter. An observed movement is approximated and regenerated using DMP methods. After learning the movement primitives, the partial contraction theory is used to combine them. We imitate the aggressive maneuvers that are performed by a human and use these primitives to achieve new maneuvers that can fly over an obstacle. Experiments on the Quanser 3-DOF Helicopter demonstrate the effectiveness of our proposed method. In addition, we linearly combine DMPs and propose a new type of DMP. We also analyze Matsuoka's oscillator and Hopf oscillator using contraction theory.

Thesis Supervisor: J.J.E. Slotine
Title: Professor of Mechanical Engineering and Information Sciences
Professor of Brain and Cognitive Sciences

# Acknowledgements

First and foremost, I would like to express my gratitude to my advisor Prof. J.J.E. Slotine for his patience, encouragement and advising that improved me and my research.

I would like to thank my great friend Selcuk Bayraktar who supported me to apply my work and provided me the helicopter.

I would like to thank Dave Roberts for his support to me for using the Aero Lab.

I would like thank my labmates for their friendships.

Finally, I would like to thank my family. They were always supportive and trustful to me.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The role of UAVs (Unmanned Aerial Vehicles) have gained significant importance in the last decades. They have many advantages (agility, low surface area, ability to work in constrained or dangerous places) compared to their conventional precedents. However, a key question still remains: Are we really utilizing their full capacity? In this thesis, we will use a novel approach that combines different fields to give a qualitative answer to this question.

## 1.1 Thesis outline

We start with showing the background of our approach in Chapter 2.

Chapter 3 contains the analysis of the DMP algorithm using contraction theory. The learning aspect of the algorithm is also studied in this chapter

Chapter 4 describes the method used in coupling the primitives.

Chapter 5 starts with describing the experimental setup and the controller that are used on our application. Next, we describe the Simulations and the Experiments step by step, starting from trajectory learning to the final tests.

Chapter 6 contains the analyzes of Matsuoka's and Hopf oscillator.

Chapter 7 contains the extensions of the DMP algorithm.

This thesis concludes with Chapter 8.

# Chapter 2

# Research Approach

## 2.1 Background

We can classify the background into three fields that each has specific importance.

### 2.1.1 Biological Motivation

The animal nervous system is an expert in controlling the body to accomplish certain movements despite environmental challenges. For example, a squirrel can climb a tree very quickly despite the many dynamic constraints. We are still far away from explaining how the nervous system controls motion, but there has been a number of studies which lead us closed to the answer.

In the experiments [23], monkeys were used to carry out a movement of a forearm to reach a visual target (See Fig. 2-1) and reward was given if they hold at this position for one second. During this process, their forearm was fastened by an apparatus that lets them extend their forearm in the horizontal plane without the sight of their arm. Tests were performed before and after the medical surgery that deprives the monkeys of a sensory input from their forearm. In both intact and the deafferented monkeys, initial position of the arm changed before the beginning of the movement (150-200 ms). However, in both cases, the forearm moved to the target precisely. This experiment suggests that **the movement can be achieved regardless of**

**the initial conditions** and the control variable is the equilibrium point achieved by agonist and antagonist muscles (See Fig. 2-2).

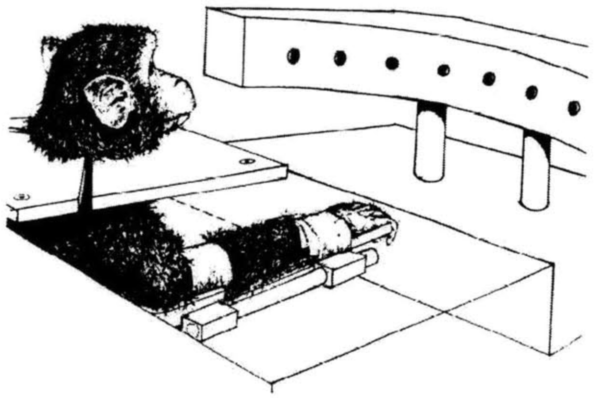


Figure 2-1: Monkey is seated on a chair and its arm is fastened to the splint. Targets are located at 5 degrees intervals. In experiments, the monkey was not allowed to see its arm. The room was darkened. [23]

It is also found that the success of the deafferented monkey is closely related to the positions between the body and the arm apparatus. When they changed the posture of the monkey, it was seen that deafferented monkey's approach to the target was inaccurate, although the intact monkey achieved the same movement. This experiment underscores the fact that **performance is closely connected with feedback control.**

The same experimental setup was used again over monkeys to determine the characteristic of the trajectory [24]. In one experiment, when the intact monkey started to approach the target, a brief torque was applied to drive the elbow quickly to the desired position. However, it was noted that the elbow returned to an intermediate position between starting and the target point before turning back to the target position (See Fig. 2-3). We can see that while returning back to intermediate location, flexor muscles showed EMG (electromyographic) activity.

In another experiment which was performed in deafferented monkeys, the arm was located at the target position and stabilized there using torque motor. The monkey didn't expect a reward, because the target was not illuminated. In addition, the

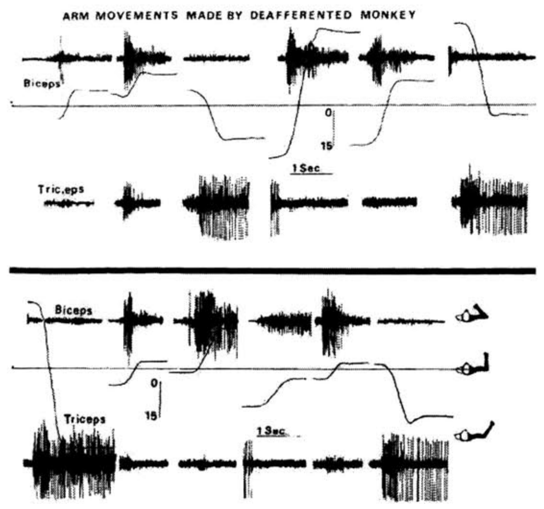


Figure 2-2: Unloaded arm movements of the monkey . In every trial, the initial position was different and monkey managed to reach the target position. [23]

monkey was insensible to the displacement because of the lack of sensory inputs. In its new position, the target was illuminated. However, instead of staying in the current position, the trained monkey moved its arm first to the location where its arm was originally located and moved it back to the target location (See Fig. 2-4). **The last two experiments suggest that the movement called "virtual trajectory" is composed of more than one equilibrium point and CNS (central nervous system) uses this stability of lower level of the motor system to simplify the generation of movement primitives [24] .**

Experiments on frogs [30, 25] show us another clue to understand the movement primitives. They microstimulated spinal cord and measured the forces at the ankle. This process is repeated with ankle replaced at nine to 16 locations and it is observed that collection of measured forces is always convergent to a single equilibrium point (See Fig. 2-5).

In microstimulation of the spinal cord, the forces acting on the ankle is not stationary. It changes with time. It was also observed that snapshot of these force fields converge into an equilibrium point (See Fig. 2-6) . **These findings also suggest that inverse kinematics plays a crucial role in determining the endpoint**

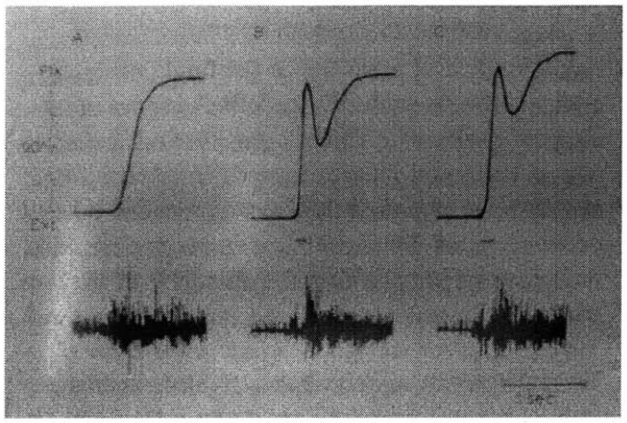


Figure 2-3: Trajectories of the forearm and EMG data of the biceps observed from intact monkeys. The bar below the trajectories indicates the duration of the disturbance. A shows the trajectory without disturbances. In B and C, the arm was driven to the target early using external torque. The forearm moved to the position between original and target locations. [24]

**trajectory [31].**

### 2.1.2 Imitation Learning

Confucius once said: "By three methods we may learn wisdom: first, by reflection, which is noblest; second, by imitation, which is easiest; and third, by experience, which is the most bitter. Imitation takes place when an agent learns a behavior by observing the execution of that behavior by a teacher (Bakker-Kuniyoshi [10]). Imitation is not inherent to humans and is also observed in animals. For example, experiments show that kittens exposed to adult cats manipulate levers to retrieve food much faster than the control group (Galef [27]).

There has been a number of applications of imitation learning in the field of robotics. Studies on locomotion [4, 5, 33], humanoid robots [6, 7],[28], [26], and human-robot interactions [32, 19] have used either imitation learning or movement primitives. The emphasis on these studies is on primitive derivation and movement classification [29]; combinations of the primitives [20, 15, 21, 22, 43, 41] and primitive models [16, 18, 45, 41] in order to extract behaviors.

In this work, we use a novel approach that segments a DMP [17] into two primitives and combine them consecutively in different initial conditions to generate aggressive maneuvers. Our approach is designed in view of the experiments on frogs and monkeys which suggest that we are faced with an inverse-kinematics algorithm that adapts to the environment and changes in a sequence of equilibrium points irrespective of the initial conditions. The equilibrium points in our algorithm have not necessarily time bounds and we take the original maneuvers from human-piloted flight data.

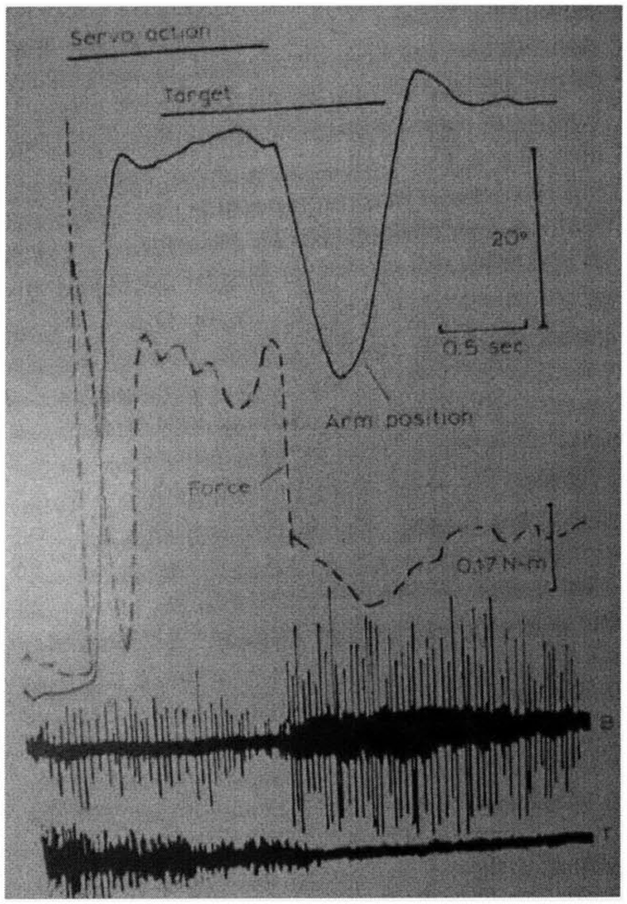


Figure 2-4: Trajectories and the EMG data of the forearm movements of the deafferented monkey. Before the target illustration, torque is applied to drive the arm to the target location. When the target is activated, the arm returned back to a position between original and target position. The upper horizontal bar and the lower horizontal bars indicate duration of servo action and the target light respectively. The continuous lines show arm position and the dashed lines demonstrates torque. B=biceps, T=triceps. [24]

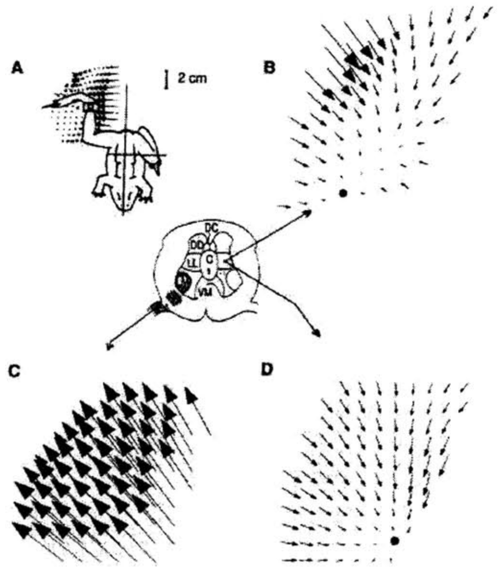


Figure 2-5: Force fields created by microstimulation . A= Convergence force fields(CFF) around the frog's leg, B= CFF taken from deafferented frog, C= Force fields extracted the region of motoneurons, D= CFF taken by microstimulation of a spinal cord at 40 Hz for 300 ms, Center= Transverse section of the spinal cord. [30]

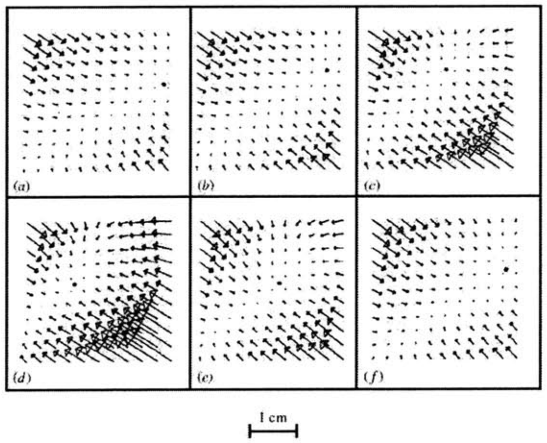


Figure 2-6: Snapshots of the equilibrium points following the microstimulation. [25]

# Chapter 3

# Analysis of DMP

This section outlines the analysis of the DMP algorithm [17] which we later combine using contraction theory.

## 3.1 DMP Algorithm

DMP is a trajectory generation algorithm which interpolates between the start and end points of a path based on learning. We use the preferred path of the human operator as the learning criterion. The system can be represented by

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) \tag{3.1}$$

$$\tau \dot{y} = z + f, \tag{3.2}$$

where $y$, $\dot{y}$ and $\ddot{y}$ characterize the desired trajectory. Hence, the canonical system is given by

$$\tau \dot{v} = \alpha(\beta_z(g - x) - v) \tag{3.3}$$

$$\tau \dot{x} = v, \tag{3.4}$$

where $g$ is the desired end point. Assuming that the $f$-function is zero, system will converge to $g$ exponentially. The goal of the DMP algorithm is to modify this exponential path. Therefore, the $f$-function makes the system non-linear and allows us to generate different trajectories between the origin and the $g$ point.

The $f$-function is a normalized mixture of Gaussians functions which approximates the final trajectory, i.e. it has the general form

$$f(x,v,g) = \frac{\sum_{i=1}^{N} \Psi_i w_i v}{\sum_{i=1}^{N} \Psi_i}, \tag{3.5}$$

where

$$\Psi_i = exp\{-h_i(x/g - c_i)^2\}. \tag{3.6}$$

## 3.2 Rhythmic DMPs

The DMP algorithm can also be extended to the rhythmic movements [47] by changing the canonical system with the following:

$$\begin{aligned} \tau\dot{\phi} &= 1 && (3.7) \\ \tau\dot{r} &= -\mu(r - r_0) && (3.8) \end{aligned}$$

where $\phi$ corresponds to $x$ in Eq. 3.3 as a temporal variable. Similar to the discrete system, control policy:

$$\begin{aligned} \tau\dot{z} &= \alpha_z(\beta_z(y_m - y) - z) && (3.9) \\ \tau\dot{y} &= z + f && (3.10) \\ f(x,v,g) &= \frac{\sum_{i=1}^{N} \Psi_i w_i^T \tilde{v}}{\sum_{i=1}^{N} \Psi_i} && (3.11) \\ \psi &= exp\{h_i(\cos(\phi - c_i) - 1)\} && (3.12) \end{aligned}$$

where $y_m$ is a basis point for learning and $\tilde{v} = [r\cos(\phi), r\sin(\phi)]^T$.

## 3.3 Learning of primitives using DMPs

Learning aspect of the algorithm comes into play with the computation of the weights ($w_i$) of the Gaussians. Weights are derived from Eq.3.1 and Eq.3.2 using the training trajectory $y_{demo}$ and $\dot{y}_{demo}$ as variables . Once the parameters of the $f$-function are learned, then DMP can simply be used to generate the original trajectory. In DMP, spatial and temporal shifts are achieved by adjusting the $g$ and $\tau$ respectively and DMP functions as shown below.

- Spatial adjustments: Recall that a function ($f$) is said to be linear if it has the property:

$$f(\alpha x + \beta x) = \alpha f(x) + \beta f(y) \quad f : \Re^m \rightarrow \Re^n \tag{3.13}$$

  where $x, y \in \Re^m \quad \alpha, \beta \in \Re$. In spatial adjustments, the first system [Eq.(3.1), Eq.(3.2)] can be seen as a linear system. It is due to the fact that variable $v$ in $f$-function is only multiplied by time-varying constant. From linearity, output ($y$) is a scaled version of the input ($g$).

- Temporal adjustments: The second system [(Eq.(3.3) Eq.(3.4)] is linear and responses against the adjustments of $\tau$ proportionally . Hence, the multiplier of variable $v$ in $f$-function is still a time-varying constant which is temporally scaled by $\tau$. From superposition, we can say that temporal adjustment of the whole system can be achieved by changing the variable $\tau$.

These arguments can also be extended to the rhythmic DMP for modulations.

## 3.4 Analysis of DMP Using Contraction Theory

Consider a system

$$\frac{d}{dt}\begin{bmatrix} \delta z_1 \\ \delta z_2 \end{bmatrix} = \begin{bmatrix} F_{11} & 0 \\ F_{21} & F_{22} \end{bmatrix}\begin{bmatrix} \delta z_1 \\ \delta z_2 \end{bmatrix} \tag{3.14}$$

where $z_1$ and $z_2$ represent the first and the second system of DMP respectively. It can be seen that the second system is contracting without any coupling term and bounded $F_{21}\delta z_1$ represents exponentially decaying disturbance. Similar to $F_{11}$, $F_{22}$ is contracting in the second equation. Hence, there is a hierarchy among the contracting systems and the whole system globally exponentially converges to a single trajectory. Hierarchical contraction can also be seen in rhythmic DMPs, since the derivatives of $x = r\cos(\phi)$ and $y = r\sin(\phi)$ are contracting.

Although the system will eventually contract to the $g$ point, there will be a time delay due to the hierarchy between second and the first system. We can decrease this delay by increasing the number of weights in our equation.

Stability of the DMPs can easily be analyzed. Once the original trajectory is mapped into the DMP, the system behaves linearly for a given input as shown before. Moreover, contraction property guarantees the convergence to a single trajectory. In this perspective, it is easy to show that learning the trajectories is not constrained by the stationary goal points that do not have a velocity compound.

# Chapter 4

# Coupling of DMPs Using Contraction Theory

In this section, we use partial contraction theory [14] to couple DMPs.

## 4.1 One-way Coupling

One-way coupling configuration of contraction theory allows a system to converge to its coupled pair smoothly. Theory for the one way coupling states the following:

$$\dot{x}_1 = f(x_1, t) \tag{4.1}$$

$$\dot{x}_2 = f(x_2, t) + u(x_1) - u(x_2) \tag{4.2}$$

In a given system, if $f - u$ is contracting, then $x_2 \rightarrow x1$ from any initial condition.

A typical example for one way coupling is an observer design. While the first system represents the real plant and the second system represents the mathematical model of the first system. The states of the second system will converge to the states of the first system and result in the robust estimation of the real system states. However, for our experiments, we interpret contraction as to imitate the transition between two states. It will be shown in chapter V how the end of one trajectory becomes the initial condition of the second trajectory and contraction accomplishes

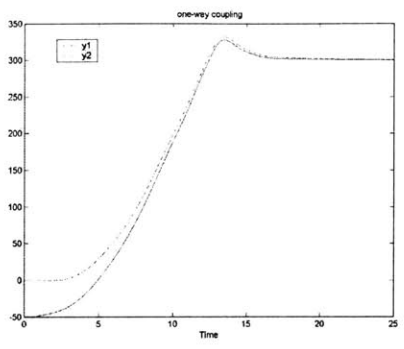


Figure 4-1: One-way coupling for a regular DMP

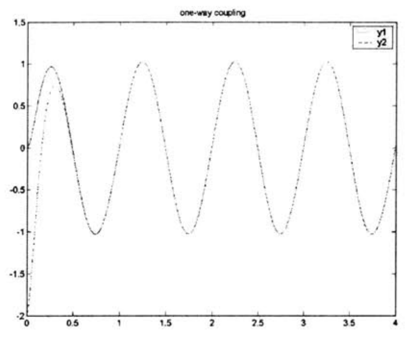


Figure 4-2: One-way coupling for a rhythmic DMP

the smooth transition.

In DMPs, we couple the two systems using the following equations:

$$\ddot{y_1} = g_1 - y_1 - \dot{y_1} + f(y_1) \tag{4.3}$$

$$\ddot{y_2} = g_2 - y_2 - \dot{y_2} + u(y_1) - u(y_2) \tag{4.4}$$

$$u(x_i) = g_i + f(x_i) \tag{4.5}$$

$$f(x, v, g) = \frac{\sum_{i=1}^{N} \Psi_i w_i v}{\sum_{i=1}^{N} \Psi_i} \tag{4.6}$$

$$\Psi_i = exp\{-h_i(x/g - c_i)^2\} \tag{4.7}$$

A toy example of the equations listed above can be seen in Fig. 4-1 and Fig. 4-2. In this setting, $y_2$ is the first trajectory primitive, which contracts to $y_1$ – the second trajectory primitive.

One-way coupling has many advantages over its precedents. In [41], trajectories are achieved by stretching temporal and spatial components of the system and there is a direct relation between initial and end points. Moreover, there are discontinuities in terms of derivatives of the trajectory at the transition regions between primitives. Giese [42] solves the problem of discontinuities by first taking the derivatives of the original trajectories, then combining the derivatives, and finally integrating them again using initial conditions. However, this method adversely affects the accuracy of the trajectories. Hence, our method improves on [41] and [42] by generating more accurate trajectories independent of initial points. We summarize the advantages for using dynamical systems as control policies as follows:

- It is easy to incorporate perturbations to dynamical systems.
- It is easy to represent the primitives.
- Convergence to to the goal position is guaranteed due to the attractor dynamics.
- It is easy to modify for different tasks.
- At the transition regions, discontinuities are avoided using initial conditions for the derivatives.
- Partial contraction theory forces the coupling from any initial condition.

Also in [17], the system is driven between stationary points. However, biological experiments suggest that we are faced with a "virtual trajectory" composed of equilibrium points that has velocity compounds. For this reason, we showed that we can achieve this property by combining nonconstant points.

# Chapter 5

# Experiments on Helicopter

In this section we apply the motion primitives on the helicopter.

## 5.1 Experimental Setup and Controller

### 5.1.1 3 DOF Helicopter

We used Quanser Helicopter (see Figure 5-1) in our experiments. The helicopter is an under-actuated and minimum-phase system having two propellers at the end of the arm. Two DC motor are mounted below the propellers to create the forces which drive propellers. The motors' axes are parallel and their thrust is vertical to the propellers. We have three degrees of freedom (DOF): pitch (vertical movement of the propellers), roll (circular movement around the axis of the propellers) and travel (movement around the vertical base) in contrast with conventional helicopters with six degrees of freedom.

### 5.1.2 System Dynamics

In system model[8], origin of our coordinate system is at the bearing and slip-ring assembly. The combinations of our actuators form the collective ($T_{col} = T_L + T_R$) and cyclic ($T_{cyc} = T_L - T_R$) forces and they are used as inputs in our system. The pitch and roll motions are controlled by collective and cyclic thrust respectively. Motion in

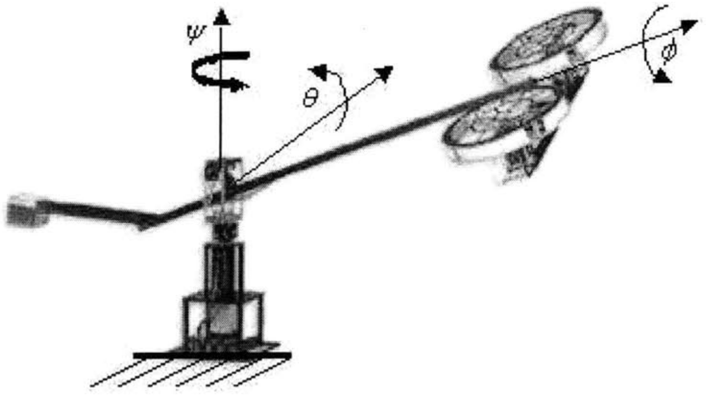


Figure 5-1: Transverse momentum distributions.[8]

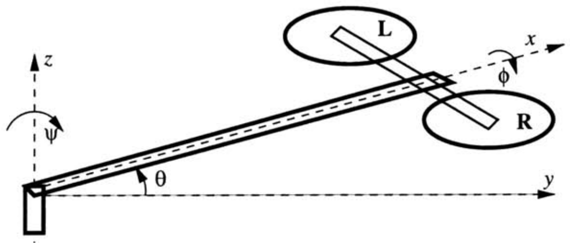


Figure 5-2: Schematic of the 3DOF helicopter.[8]

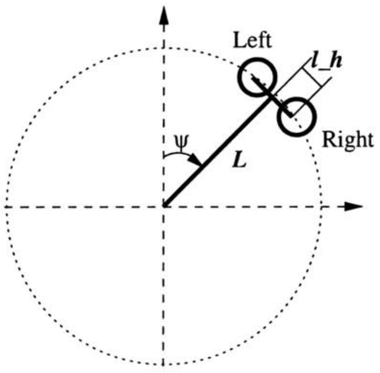


Figure 5-3: Top view. [8]

travel angle is controlled by the components of thrust. Positive roll results in positive change of angle. The schematics of helicopter are shown in Figures 5-2 and 5-3

Let $J_{xx}$, $J_{yy}$, and $J_{zz}$ denote the moment of inertia of our system dynamics. For simplicity, we ignore the products of inertia terms. Then, the equations of motion are as follows (cf. Ishutkina [8]):

$$\begin{aligned} J_{zz}\ddot{\psi} &= (T_L + T_R)L\cos(\theta)\sin(\phi) - (T_L - T_R)l_h\sin(\theta)\sin(\phi) - \text{Drag} \\ J_{yy}\ddot{\theta} &= -Mgl_\theta\sin(\theta + \theta_0) + (T_L + T_R)L\cos(\phi) \\ J_{xx}\ddot{\phi} &= -mgl_\phi\sin(\phi) + (T_L - T_R)l_h \end{aligned} \tag{5.1}$$

- $M$ is the total mass of the helicopter assembly
- $m$ is the mass of the rotor assembly
- $L$ is the length of the main beam from the slip-ring pivot to the rotor assembly
- $l_h$ is the distance from the rotor pivot to each of the propellers
- Drag $= \frac{1}{2}\rho(\dot{\psi}L)^2(S_0 + S_0' \sin(\phi))L$
- $S_0$ and $S_0'$ are the effective drag coefficients times the reference area and $\rho$ is the density of air

Since the power to the actuators are given by the voltages, a thrust to voltage relationship has been empirically determined and implemented using a lookup table. Figure5-4[9]

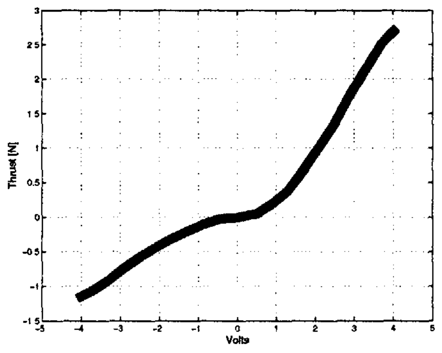


Figure 5-4: Motor Volt to Thrust Relationship.

### 5.1.3 Feedback Linearization

As shown in the above section the system is described by equations which are nonlinear in the states, but nonlinear in terms of control inputs. In practice, we want to design a tracking controller [11] for a 3DOF helicopter which can track the trajectories in elevation and travel. To use the techniques of input-output linearization [12] and design a tracking controller while avoiding singularities, we reduce the system model to dominant terms:

Table 5.1: Parameter Values

| Parameter | Value | Unit | Description |
|---|---|---|---|
| $m$ | 1.15 | $kg$ | mass of the rotor assembly |
| $M$ | 3.57 | $kg$ | mass of the whole setup |
| $m_0$ | 0.031 | $kg$ | effective mass at hover |
| $L$ | 0.66 | $m$ | length from pivot point to the heli body |
| $l_h$ | 0.177 | $m$ | length from pivot point to the rotor |
| $J_{xx}$ | 0.0344 | $kgm^2$ | moment of inertia about $x$-axis |
| $J_{yy}$ | 1.0197 | $kgm^2$ | moment of inertia about $y$-axis |
| $J_{zz}$ | 1.0349 | $kgm^2$ | moment of inertia about $z$-axis |
| $J_{yz}$ | $-0.0018$ | $kgm^2$ | product of inertia |
| $J_{zy}$ | $-0.0018$ | $kgm^2$ | product of inertia |
| $R$ | 0.1 | $m$ | radius of the rotor |
| $g$ | 9.8 | $m/s^2$ | gravitational constant |
| $l_\phi$ | 0.004 | $m$ | length of pendulum for roll axis |
| $l_\theta$ | 0.014 | $m$ | length of pendulum for pitch axis |

$$\ddot{\theta} = \frac{-Mgl_\theta}{J_{yy}} sin(\theta + \theta_0) + \frac{L}{J_{yy}} cos(\phi) Tcol \tag{5.2}$$

$$\ddot{\phi} = \frac{l_h}{J_{xx}} T_{cyc} - \frac{mgl_\phi}{J_{xx}} sin(\phi) \tag{5.3}$$

$$\ddot{\psi} = \frac{L}{J_{zz}} cos(\theta) sin(\phi) T_{col} \tag{5.4}$$

To achieve tracking tasks we derive the equivalent input:

$$v = \ddot{x}_d - 2\lambda\dot{\tilde{x}} - \lambda^2\tilde{x}$$

with $x = [\theta\phi\psi]$ and $\tilde{x} = x - x_d$ being the tracking error. Since, Eq.5.3 is decoupled from the other two equations and one can use dynamic inversion to write:

$$T_{col} = \frac{V_\theta - c_0 sin(\theta + \theta_0)}{c_1 cos(\theta)}$$

$$T_{col} = \frac{V_\psi}{c_4 cos(\theta) sin(\phi)}$$

we can solve for appropriate $\phi_d$ through

$$tan(\phi_d) = \frac{c_1 V_\psi}{(V_\theta - c_0 sin(\theta + \theta_0))c_4 cos(\theta)}$$

where c's substitute for the appropriate constants from the dynamic equations. Once the corresponding $\phi_d$ has been found, $\dot{\phi}_d$ and $\ddot{\phi}_d$ are set to zero and $V_\theta$ is calculated:

$$V_\phi = \ddot{\phi}_d - 2\lambda_\phi(\dot{\phi} - \dot{\phi}_d) - \lambda_\phi^2(\phi - \phi_d)$$

Finally, we find

$$\begin{aligned} T_{cyc} &= \frac{V_\phi - c_3 sin(\phi)}{c_2} \\ T_{col} &= \frac{V_\psi - c_5 T_{cyc} sin(\theta) sin(\phi)}{c_4 cos(\theta) sin(\phi)} \end{aligned}$$

To use feedback linearization control law, it is necessary to measure all the states of the system. Since only the angles of 3DOF helicopter are measured, we have applied a derivative and a low pass filter to collect the derivatives of the states. Once implemented in Simulink via S-functions, the controller described above is robust and has excellent tracking characteristics.

## 5.2 Trajectory Generation

In our experimental setup, we used an operator with a joystick to create aggressive trajectories to pass an obstacle. However, generating aggressive trajectories with the joystick is a difficult task even for the operator. Therefore, we designed an augmented control for the joystick to enhance the performance of the helicopter. In particular, we used "up" and "down" movements of the joystick to increase or decrease the $T_{col}$ that is applied to the actuators. For the "right" and "left" movements of the joystick, we preferred to control the roll angle using PD control.

In the original maneuver, the obstacle's distance and the highest point are in the coordinates where $\psi$ and $\theta$ angles are 220 and 60 respectively and the helicopter starts with coordinates where $\psi = 28$, $\theta = 317$ and $\phi = -17$ (see Figure 5-5).

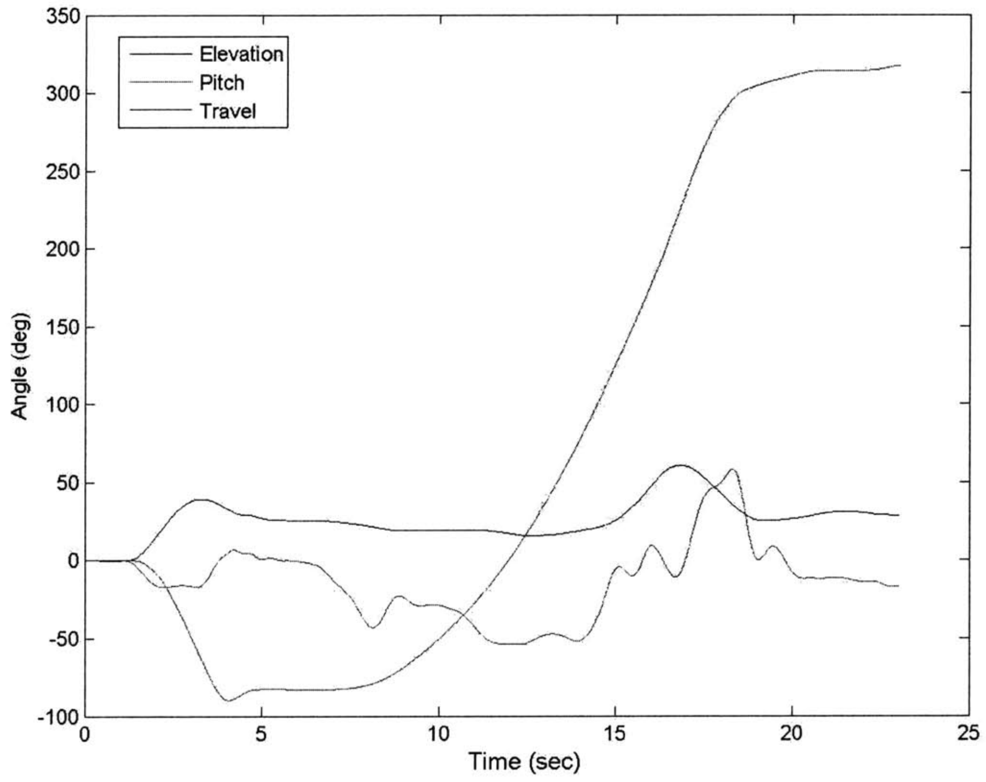


Figure 5-5: Original maneuver achieved by an operator

## 5.3 Trajectory Learning

To fly over different obstacles, the acquired primitive is segmented into two primitives at the highest pitch angle. Figures 5-6 and Fig. 5-7 show the results of DMP algorithm for the pitch angle. Since the Quanser Helicopter has 3-DOF, for any given trajectory, we use DMP to approximate paths on roll and travel components as well. In Fig.

5-6, the top left graph is a result for the pitch angle, where green line represents the operator input for the trajectory and the blue line represents the fitting that the DMP computes. Other figures show the time evolution of the DMP parameters. It can also be seen that the generated trajectories capture the operator's driving characteristics.

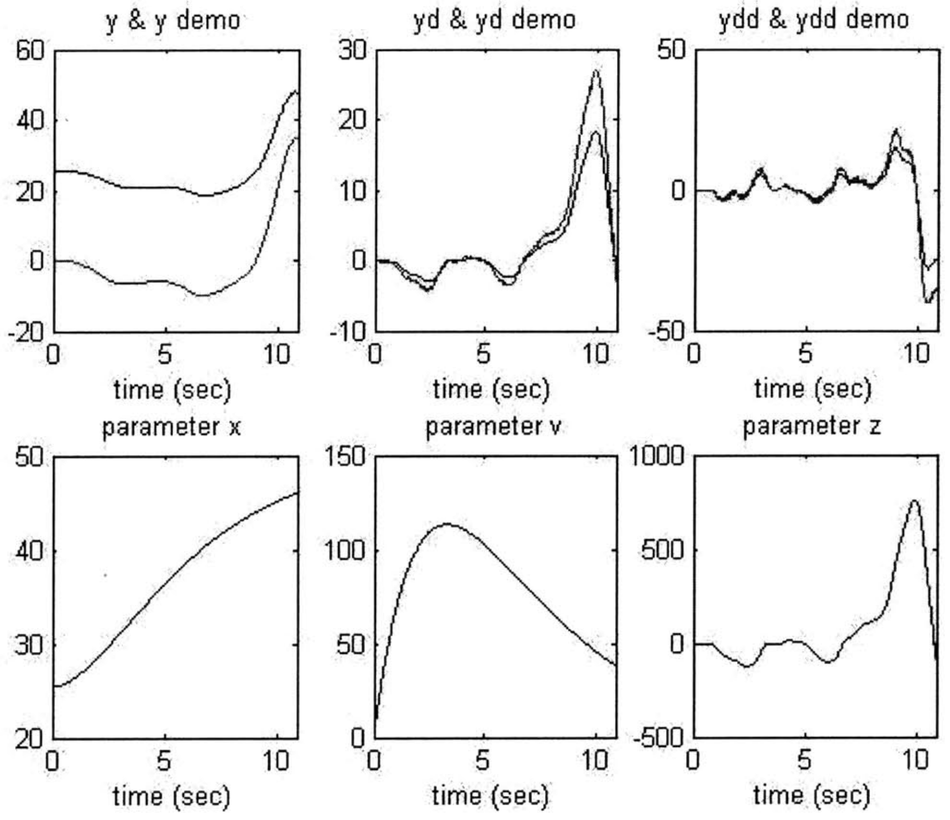


Figure 5-6: Trajectory generation for the first primitive - pitch

## 5.4 Synchronization of primitives

The two primitives created in the previous sections are defined as trajectories between certain start and end points. However, the end point of the first trajectory does not necessarily matches with the starting point of the second trajectory. We use the contraction theory to force the first trajectory to converge to the second trajectory.

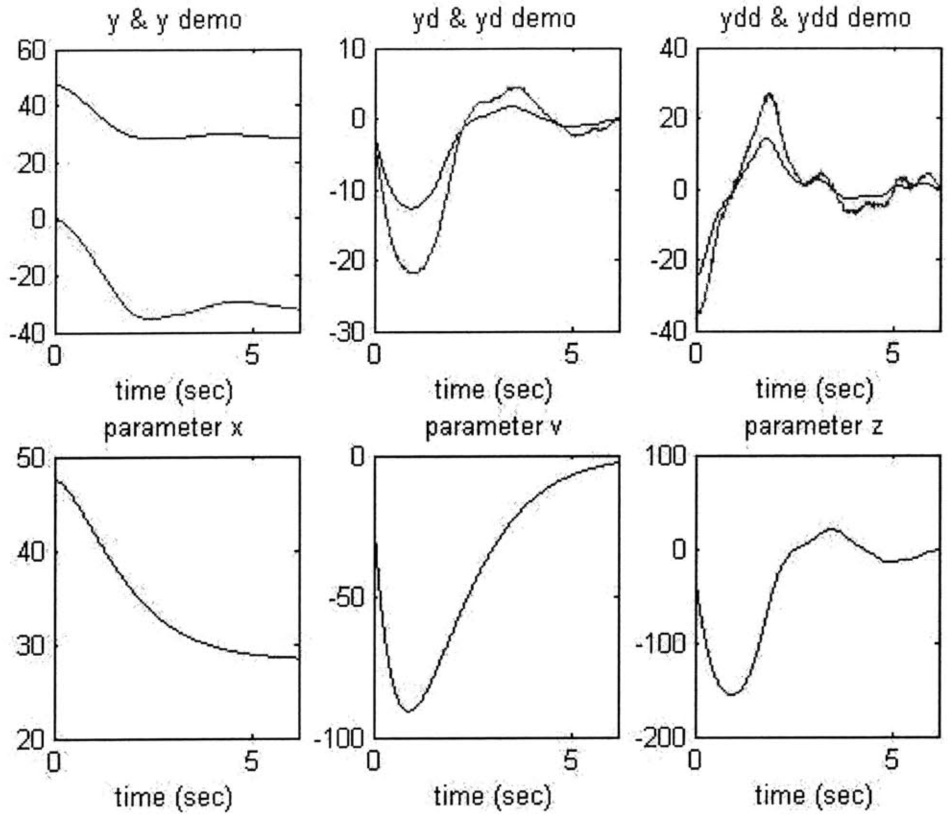


Figure 5-7: Trajectory generation for the second primitive - pitch

However, since we want to use the contraction as a transition between two trajectories, coupling is enabled towards the end of first primitive. Figure 5-8 shows how the two trajectories evolve in time. In the first primitive, the goal positions of $\psi$ and $\theta$ angles are changed to 150° and 50° respectively, where original angles are $\psi = 220°$ and $\theta = 60°$. In the second primitive, the goal position of the $\psi$ angle is changed from 317° to 300°. Solid line shows the travel, dashed line shows roll, and dash-dot line shows the pitch angles while the vertical dotted line indicates the switching time from the first primitive to the second one. It can be seen that the trajectories fit each other smoothly at the transition points.

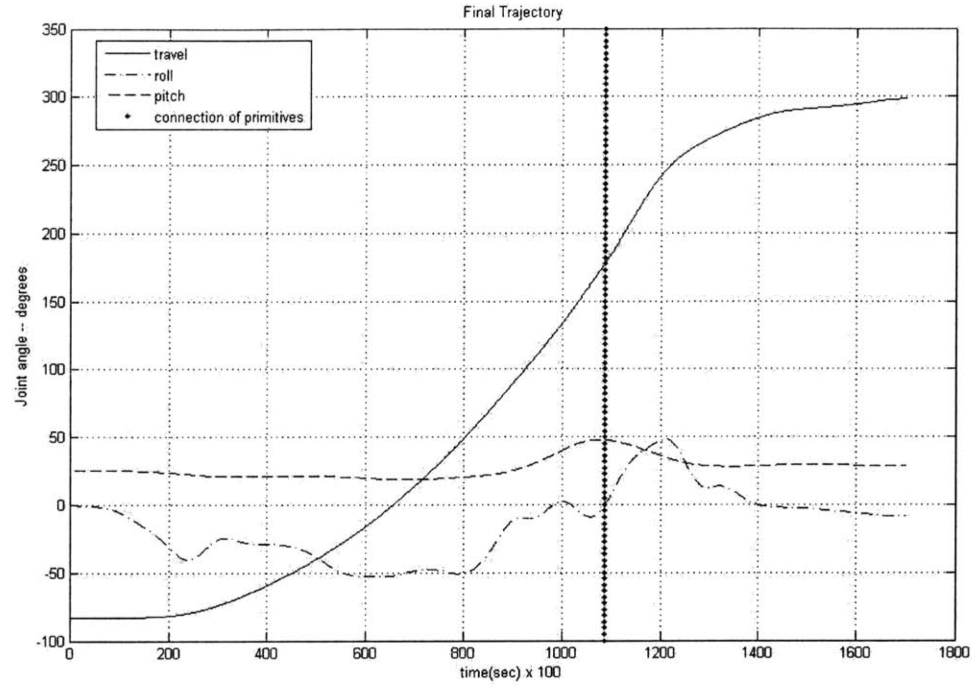


Figure 5-8: Time evolution of primitive-1 and primitive-2 merged.

## 5.5 Experiments on the Helicopter

Tracking performance of our helicopter with respect to the desired trajectory is shown in Figure 5-9. It can be seen that the helicopter followed the desired ($\psi$ and $\theta$) angles almost perfectly. The trajectory of the roll angle is different than the desired since we control two parameters ($\psi$ and $\theta$) and the goal positions of the DMPs are different. However, it can be seen that they follow the same pattern. The last part of the roll trajectory manifests an oscillation which can be prevented by roll control, since the other parameters are almost constant. The tracking performance can further be improved by applying discrete nonlinear observers to get better velocity and acceleration values. Figure 5-10 shows snapshots of the maneuver.

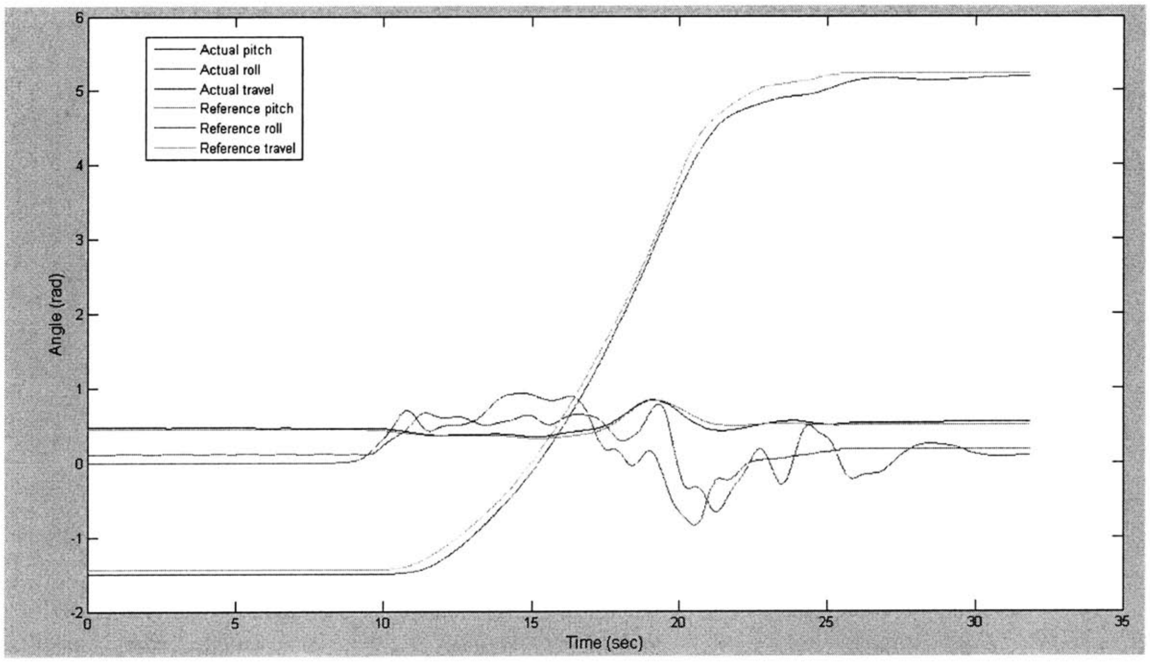
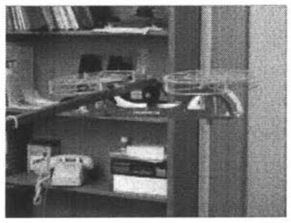
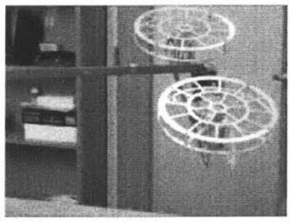
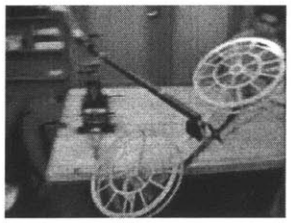


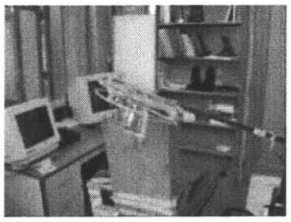
Figure 5-9: Tracking performance of the helicopter

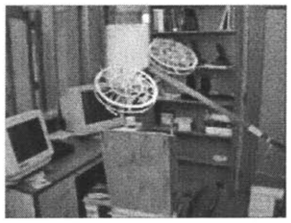

(a) 1

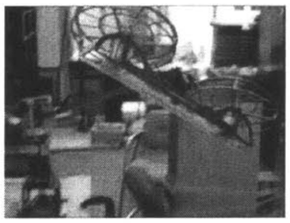

(b) 2

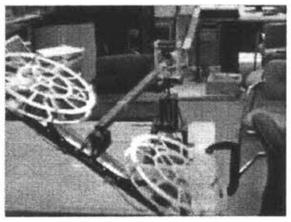

(c) 3

(d) 4


(e) 5


(f) 6


(g) 7


(h) 8

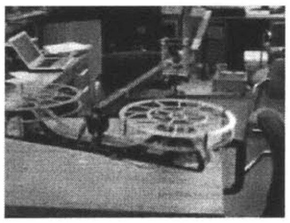

(i) 9

Figure 5-10: Snapshots of the obstacle avoidance maneuver.

# Chapter 6

# Analysis of Oscillators Using Contraction Theory

In biological systems, various types of oscillations and rhythmic movements can be observed. Studies on these behaviors suggest that spinal cord possesses central pattern generators (CPGs) to have such a property [49]. In this perspective, there are several types of CPG models in the literature. The purpose of this chapter is to analyze two of them (the Hopf oscillator and Matsuoka's oscillator) using contraction theory.

## 6.1 Hopf Oscillator

In [46], Ijspeert design a generic central pattern generator (CPG) that can adapt to given periodic signals. In the model, he represents the periodic function as a combination of Hopf oscillators like fourier series and each oscillator's frequency and amplitude are learned using Hebbian learning (See Fig.6-1). Here, we will analyze the Hopf oscillators after learning. The system can be represented by

$$\dot{x} = \gamma(\mu - r^2)x - wy \tag{6.1}$$

$$\dot{y} = \gamma(\mu - r^2)y + wx \tag{6.2}$$

where $r = \sqrt{x^2 + y^2}$ and $w$ is the coupling term which is defined through learning.

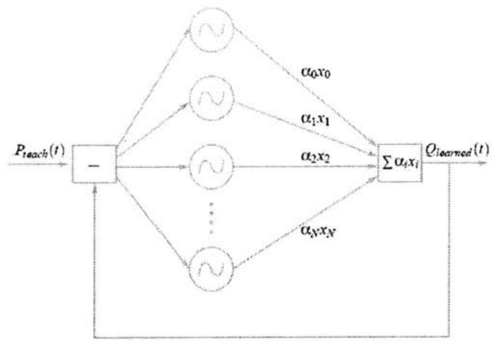


Figure 6-1: General structure of the adaptive central pattern generators.[1]

First, we assume that we have a system without coupling:

$$\dot{x} = \gamma(\mu - r^2)x \tag{6.3}$$

$$\dot{y} = \gamma(\mu - r^2)y \tag{6.4}$$

In polar coordinates, the system becomes

$$\dot{r} = \gamma(\mu - r^2)r \tag{6.5}$$

From Lyapunov, it is easy to show that $r$ converges to $\pm\sqrt{\mu}$ and the jacobian of system:

$$\frac{\partial f}{\partial r} = \mu - 3r^2 \tag{6.6}$$

Since $r^2 \rightarrow \mu$, the system is contracting as shown in Fig. 1.

The jacobian in cartesian coordinates is also contracting. The equations of oscillator are as follows

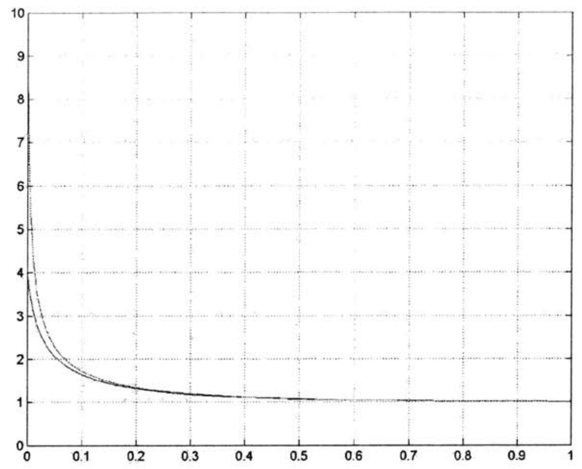


Figure 6-2: Time evolution of $r$ with initial conditions 4 and 10

$$\frac{\partial f}{\partial x} = \gamma(\mu - 3x^2 - y^2)$$
$$\frac{\partial f}{\partial y} = \gamma(\mu - 3y^2 - x^2)$$

Since $x^2 + y^2 \to \mu$, our equations contract and eventually become

$$\frac{\partial f}{\partial x} = -2\gamma x^2 \tag{6.7}$$
$$\frac{\partial f}{\partial y} = -2\gamma y^2 \tag{6.8}$$

**Theorem 1** *if we couple Eq.6.3 and Eq.6.4 as it is done in Hopf oscillators, then $x$ and $y$ will converge to each other exponentially with a phase difference $\pi/2$ regardless of initial conditions.*

**Proof**: It can be shown with the same method used in [14]. Given initial conditions

$x(0)$ and $y(0)$, we define

$$g(x,y,t) = \dot{x} - \frac{\partial f}{\partial x} + wy = \dot{y} - \frac{\partial f}{\partial y} - wx \tag{6.9}$$

and construct the auxiliary system

$$\dot{z} = \frac{\partial f}{\partial z} + g(x,y,t) \tag{6.10}$$

This system is contracting since the function $\frac{\partial f}{\partial z}$ is contracting and all solutions of $z$ converges into a single trajectory. Since $z = x$ and $z = y = xj$ are two particular solutions, this implies that $x$ and $y$ synchronize with phase difference $\pi/2$.

Even after coupling, we still preserve the Eq. (6.6) in polar coordinates and the system contracts. Hence, we can apply partial contraction theory for different tasks. In particular, we will use one-way synchronization and rotational transformation (see Keehong [52]) to lock the oscillators with phase differences as it is done in [51]. The equations can be represented by

$$\dot{x_1} = \gamma(\mu - r_1^2)x_1 \tag{6.11}$$

$$\dot{y_1} = \gamma(\mu - r_1^2)y_1 \tag{6.12}$$

$$\dot{x_2} = \gamma(\mu - r_2^2)x_2 - \beta(x_2 - \cos(\theta)x_1 + \sin(\theta)y_1) \tag{6.13}$$

$$\dot{y_2} = \gamma(\mu - r_2^2)y_2 - \beta(y_2 - \sin(\theta)x_1 - \cos(\theta)y_1) \tag{6.14}$$

where $r_i = \sqrt{x_i^2 + y_i^2}$, $i = 1, 2$ and $\theta$ is the phase difference desired between oscillators. The example of phase-locking can be seen in Fig.6-3.

## 6.2 Matsuoka's Oscillators

In [44], Matsuoka design mutual inhibition networks represented by a continuous-variable neuron model to generate oscillations (see details for [44]). The system can be represented by

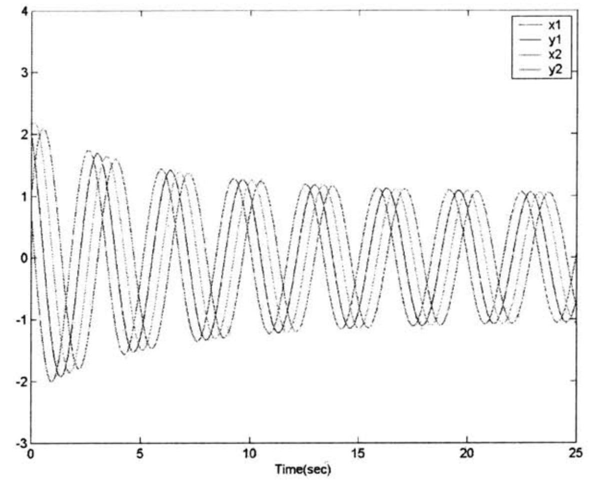


Figure 6-3: One-way coupling with a phase difference $\pi/4$

$$\dot{u_1} = -u_1 - wy_2 - \beta v_1 + u_0 \tag{6.15}$$

$$\dot{u_2} = -u_2 - wy_1 - \beta v_2 + u_0 \tag{6.16}$$

$$\tau \dot{v_1} = -v_1 + y_1 \tag{6.17}$$

$$\tau \dot{v_2} = -v_2 + y_2 \tag{6.18}$$

$$y_i = f(u_i) \quad f(u_i) = max(0, u_i) \quad (i = 1, 2)$$
$$u_0 > 0$$

where $u_i$ is the inner state of the neuron; $y_i$ is the output of the neuron; $v_i$ is the self inhibition variable; $u_0$ is the external output; w is the coupling constant; $\tau$ is the time constant. Since the system contains non-smooth functions ($y_i$) in it, we will investigate the Matsuoka's oscillators in four regions as it is done in [50].

**(i)** For $u_1 > 0$ and $u_2 > 0$, we can derive the equations below:

$$\tau \ddot{v_1} = -\dot{v_1} - (\tau \dot{v_1} + v_1) - w(\tau \dot{v_2} + v_2) - \beta v_1 + u_0 \tag{6.19}$$

$$\tau \ddot{v_2} = -\dot{v_2} - (\tau \dot{v_2} + v_2) - w(\tau \dot{v_1} + v1) - \beta v_2 + u_0 \tag{6.20}$$

From Eq. (6.19) and Eq. (6.20), we have

$$\tau(\ddot{v_1} + \ddot{v_2}) = -(\tau + 1 + \tau w)(\dot{v_1} + \dot{v_2}) - (\beta + 1 + w)(v_1 + v_2) + 2u_0 \tag{6.21}$$

and also

$$\dot{u_1} + \dot{u_2} = -(u_1 + u_2) - w(u_1 + u_2) - \beta(v_1 + v_2) + u_0 \tag{6.22}$$

From the equations above, it is easy to say that variables $u_1 + u_2$ and $v_1 + v_2$ will converge to equilibrium points, if the constants that multiplies the variables are positive. However, there are two solutions in terms of $u$'s and $v$'s. For $u$'s, first solution is $u_1 = 0, u_2 = 0$ and the second one, which is of our interest, is $u_1 = -u_2$. To have the second solution, our system should be unstable. Therefore, eigenvalue analysis is done to have proper variables. From analyzes, it is found that one of the conditions listed below should be accomplished.

$\beta < w - 1$, $\beta < -w - 1$, $w < -1 - \frac{1}{\tau}$ or $w > 1 + \frac{1}{\tau}$

**(ii)** For $u_1 > 0$ and $u_2 < 0$, we can derive the equations below:

$$\tau \ddot{v_1} = -(\tau + 1)\dot{v_1} - (\beta + 1)v_1 + u_0$$

$$\tau \dot{v_2} = -v_2$$

From ODE analysis, It is found that $v_1 \longrightarrow \frac{u_0}{\beta+1}$ and $v_2 \longrightarrow 0$. Then our equations for $u$'s become

$$\tau \dot{u_1} = -u_1 - \beta \frac{u_0}{\beta + 1} + u_0$$

$$\tau \dot{u_2} = -u_2 - w u_1 + u_0$$

Therefore, $u_1 \longrightarrow \frac{u_0}{\beta+1}$ and $u_2 \longrightarrow \frac{\beta - w + 1}{\beta+1}$ .

**(iii)** For $u_1 > 0$ and $u_2 < 0$ , solution is a symmetric version of the region (ii).

**(iv)** For $u_1 < 0$ and $u_2 < 0$, we can derive the equations below:

$$\dot{u_1} = -u_1 - \beta v_1 + u_0 \tag{6.23}$$

$$\dot{u_2} = -u_2 - \beta v_2 + u_0 \tag{6.24}$$

$$\tau \dot{v_1} = -v_1 \tag{6.25}$$

$$\tau \dot{v_2} = -v_2 \tag{6.26}$$

From ODE analysis, it can be concluded that $v_1, v_2 \longrightarrow 0$ and $u_1, u_2 \longrightarrow u_0$.

**Theorem 2** *System has periodic oscillations if and only if*

$\beta > w - 1$ and $w > 1 + 1/\tau$

**Proof:** Region(ii), region(iii) and region(iv) contract (see Fig. 6-4 and Fig.6-5) and converge to a virtual equilibrium point in region(i), if we have the following conditions: $\beta > -1$ and $w > 1 + 1/\tau$. In oscillations, there should be a continuous chain reaction among regions. To achieve such a property, parameters of the region(i) should have at least one of these conditions:$\beta < w - 1$, $\beta < -w - 1$, $w < -1 - \frac{1}{\tau}$ or $w > 1 + \frac{1}{\tau}$.

Therefore, overall conditions for parameters are $\beta > w - 1$ and $w > 1 + 1/\tau$. Since the system converges to a specific trajectory in the contracting regions , dynamic system in region(i) itself also generates a specific trajectory because of the same initial conditions. Moreover, in region (i) variables are forced to change signs . They have symmetry and receive the phase difference among each other as well.

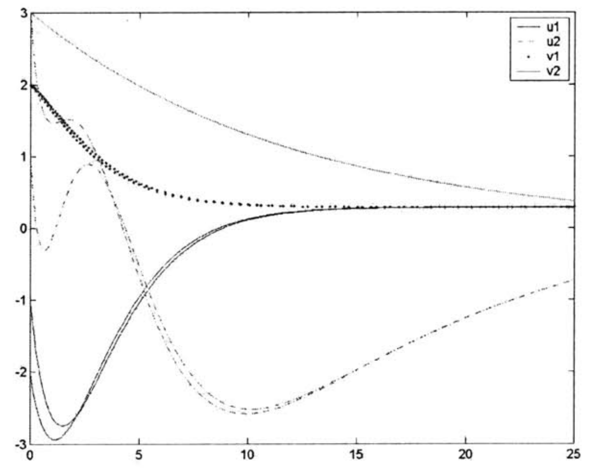


Figure 6-4: Contraction in region (ii), initial conditions of $u_1$ and $u_2$ are changed from -2,3 to -1,1 respectively

*Remark:*

As compared to two neurons, intuitively, we can derive the similar conditions for three-neuron ring in [44]. In this case, anti-synchronization in region (i) ,where $u_1, u_2, u_3 > 0$, can be achieved (see Fig.6-6) given proper parameters. If we force the other regions to converge to the region(i) with adjusted parameters then the desired variables can synchronize with a phase difference as shown in Fig.6-7.

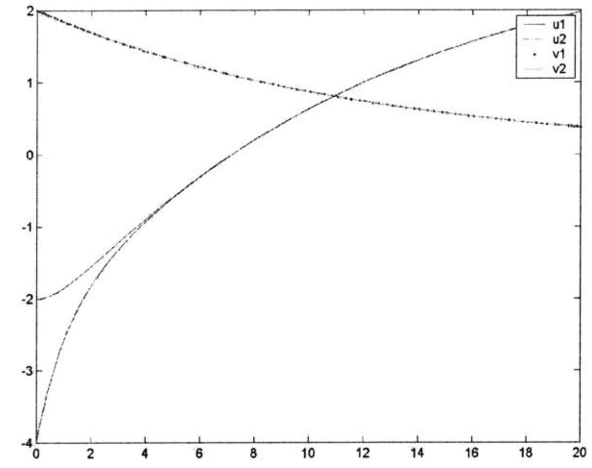


Figure 6-5: Contraction in region (iii), initial conditions of $u$'s are changed from -2 to -4

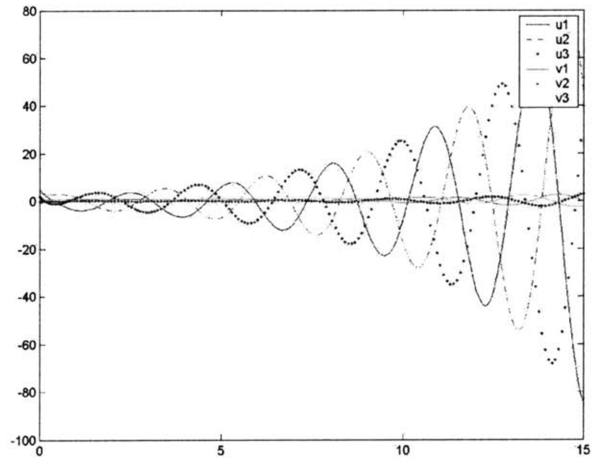


Figure 6-6: variables in region(i)

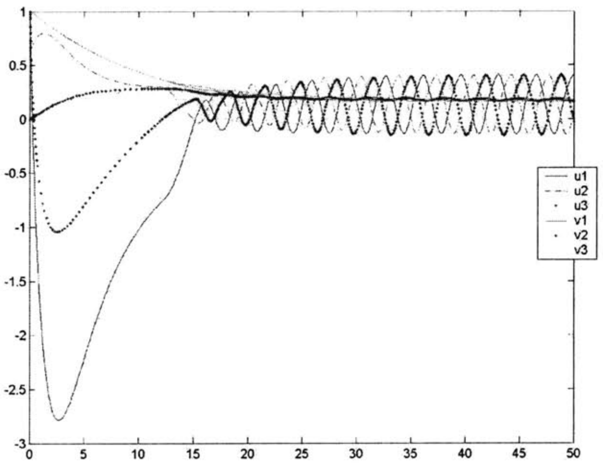


Figure 6-7: Three oscillators

# Chapter 7

# Extensions of DMP

## 7.1 Dynamical System with First-Order Filters

DMP algorithm can be improved by replacing the first system with the equations shown below:

$$\tau \dot{y} + a_1 y = a_1 x \tag{7.1}$$

$$\dot{x} + a_2 x = a_2 g + \frac{1}{a_1} f \tag{7.2}$$

which is equivalent of

$$\tau \ddot{y} + (a_1 + \tau a_2)\dot{y} + a_1 a_2 y = a_1 a_2 g + f \tag{7.3}$$

By introducing two first-order filters, we can guarantee the stability of the system against time varying parameters like $\tau(t)$ or $g(t)$ . Since the system is linear without $f$-function (Eq.3.11), we can achieve learning and modulation properties of DMP using the $f$ in either Eq.(7.1) or Eq.(7.2). For further applications we will use this model to generate primitives for time-varying goal points.

## 7.2 Two-way Synchronization of DMPs

Experiments on frog's spinal cord [30, 25, 48] suggest that movement primitives can be generated from linear combinations of vectorial force fields which lead the limb of a frog to the virtual equilibrium points. In [48], it is also pointed out that vectorial summation of two force fields with different equilibrium points generate a new force field whose equilibrium point is at intermediate location of the original equilibrium points. In this perspective, we will synchronize DMPs to generate a new primitive whose trajectory is a linear combination of synchronized trajectories. Consider a system

$$\ddot{y}_1 = f(y_1, t) + K(u(y_2) - u(y_1)) \tag{7.4}$$

$$\ddot{y}_2 = f(y_2, t) + K(u(y_1) - u(y_2)) \quad K > 0 \tag{7.5}$$

Where $y_1$ and $y_2$ represent the first and the second primitive respectively. From partial contraction theory, we can say that $y_1$ and $y_2$ converge together exponentially, if $f - 2Ku$ is contracting. Since DMPs are already contracting, we can achieve synchronization using contracting inputs. In Fig.7-1, new primitive is a linear combination of sine and cosine primitives.

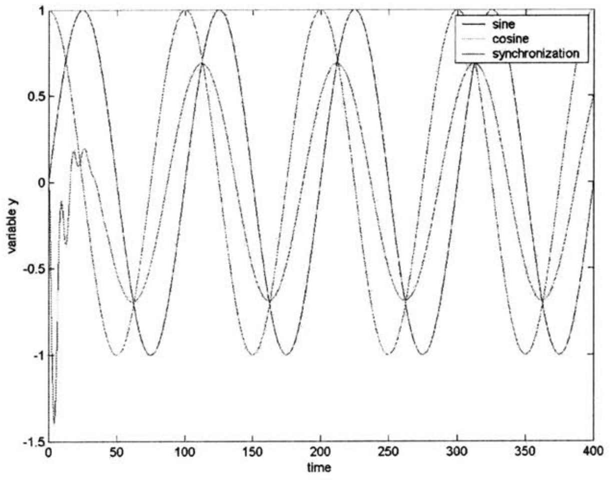


Figure 7-1: Synchronization of sine and cosine primitives

# Chapter 8

# Conclusion

In this thesis, we use a novel approach, inspired by biological experiments, which uses control primitives to imitate the data taken from human-performed obstacle avoidance maneuver. In our model, DMP computes the trajectory dynamics so that we can generate complex primitive trajectories for given different start and end points, while one-way coupling ensures smooth transition between primitives at the (possible unstable) equilibrium point. We demonstrate our algorithm with an experiment. We generate a complex, aggressive maneuver, which our helicopter could follow within a given error bound with a desired speed. We will conduct future research on different combinations of primitives using partial contraction theory.

# Appendix A

# Contraction Theory

The basic theorem of contraction analysis [13] can be stated as

**Theorem 1 (Contraction)** *Consider the deterministic system*

$$\dot{x} = f(x,t) \tag{A.1}$$

*where f is a smooth nonlinear function. If there exist a uniformly invertible matrix associated generalized Jacobian matrix*

$$F = (\dot{\Theta} + \Theta\frac{\partial f}{\partial x})\Theta^{-1} \tag{A.2}$$

is uniformly negative definite, then all system trajectories converge exponentially to a single trajectory, with convergence rate $|\lambda_{max}|$, where $\lambda_{max}$is the largest eigenvalue of the symmetric part of F. The system is said to be contracting.

It can be shown conversely that the existence of a uniformly positive definite metric

$$M(x,t) = \Theta(x,t)^T\Theta(x,t) \tag{A.3}$$

with respect to which the system is contracting is also a necessary condition for global exponential convergence of trajectories. Furthermore, all transformations $\Theta$

corresponding to the same $M$lead to the same eigenvalues for the symmetric part $F_s$ of $F$, and thus to the same contraction rate $|\lambda_{max}|$.

In the linear time-invariant case, a system is globally contracting if and only if it is strictly stable, with $F$ simply being a normal Jordan form of the system and $\Theta$the coordinate transformation to that form.

Contraction analysis can also be derived for discrete-time systems and for classes of hybrid systems.

A simple yet powerful extension to contraction theory is the concept of partial contraction, which was introduced in [14].

**Theorem 2 (Partial contraction)** *Consider a nonlinear system of the form*

$$\dot{x} = f(x, x, t)$$

*and assume that the auxiliary system*

$$\dot{y} = f(y, x, t)$$

*is contracting with respect to $y$. If a particular solution of the auxiliary $y$-system verifies a specific smooth property, then all trajectories of the original $x$-system verify this property exponentially. The original system is said to be partially contracting.*

Indeed, the virtual, observer-like $y$-system has two particular solutions, namely $y(t) = x(t)$ for all $t \geq 0$ and the solution with the specific property. Since all trajectories of the $y$-system converge exponentially to a single trajectory, this implies that $x(t)$ verifies the specific property exponentially.

# Appendix B

# Simulink Diagram

Figures show the Simulink block diagrams used in our hardware.

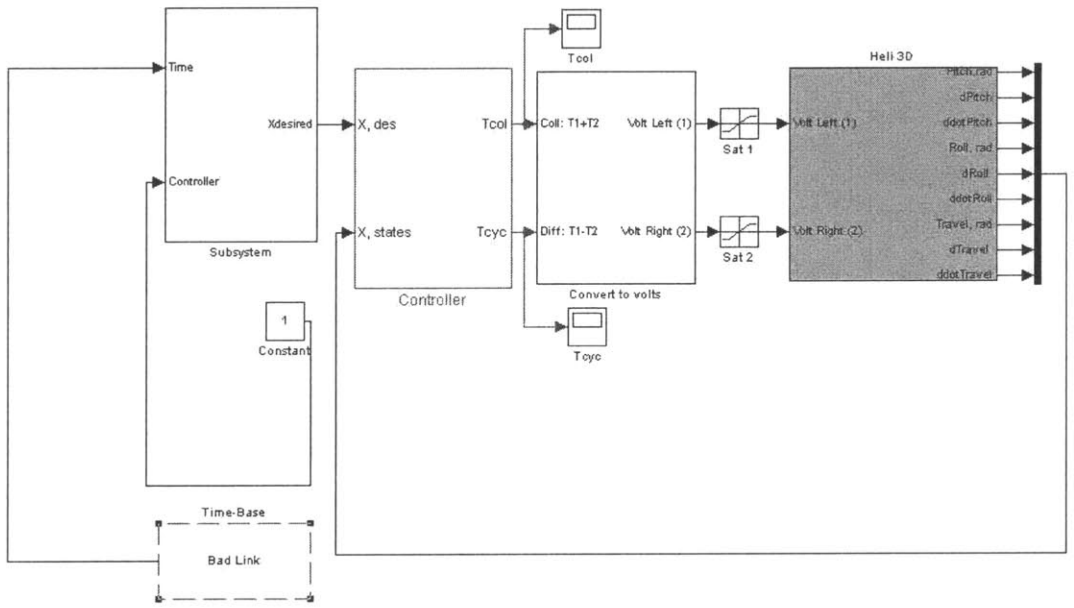


Figure B-1: Simulink block diagram of the feedback linearization controller

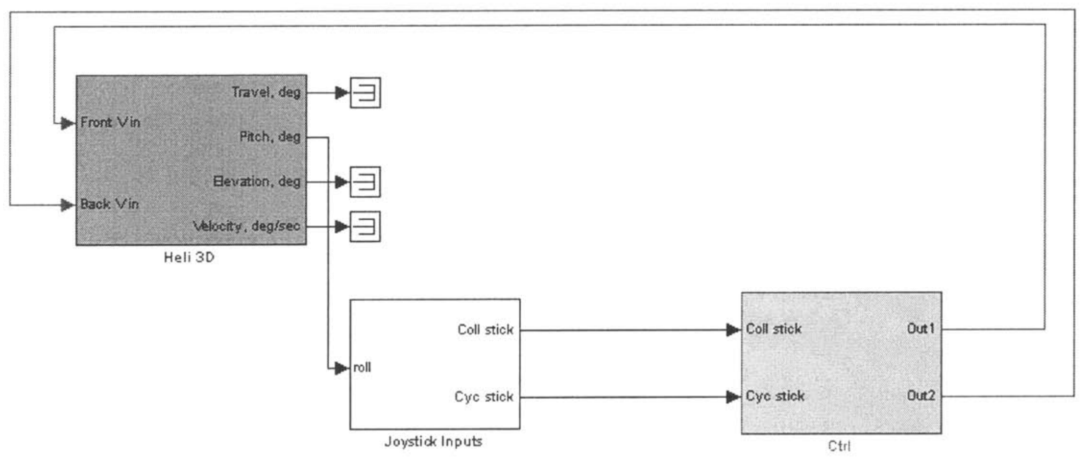


Figure B-2: Simulink block diagram of a joystick

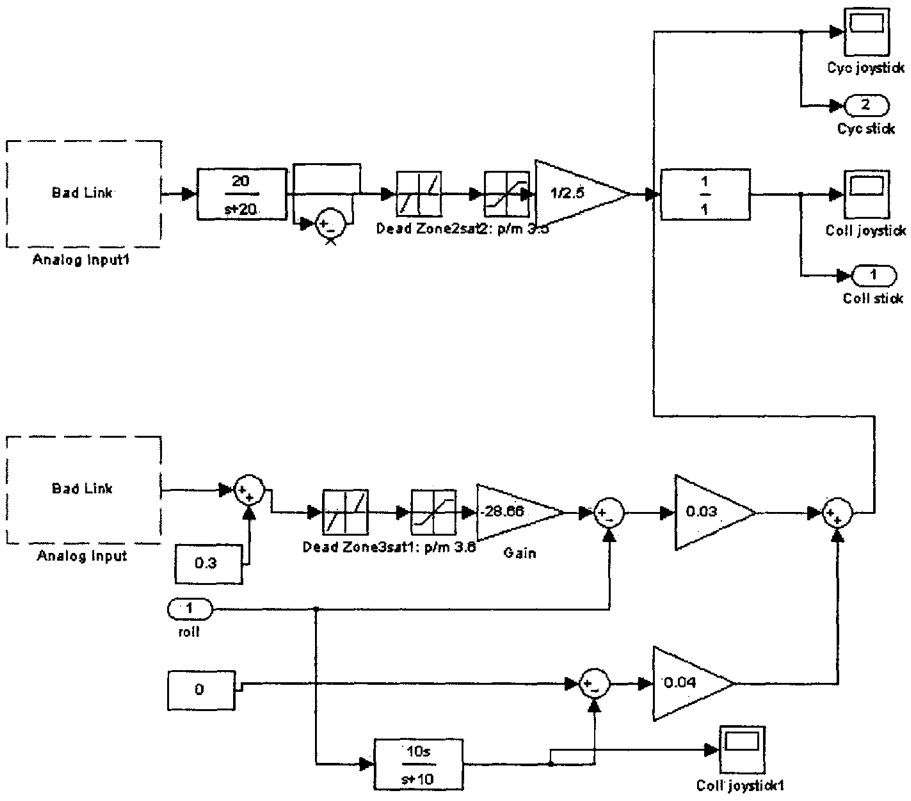


Figure B-3: Roll controller in the joystick simulink diagram

# Appendix C

# Codes of Trajectory Generator S function

```
/*=================================================================
* traj_generator.C
* The calling syntax is:
*  INPUTS:
*  OUTPUT
*  COMMENTS:
*=================================================================*/
```

% I want to thank Selcuk Bayraktar and Mariya A. Ishutkina for their contributions. This code is modified over their previous work.

```
#define S_FUNCTION_LEVEL 2 #define S_FUNCTION_NAME  traj_generator

#include "simstruc.h" #include <stdlib.h> #include <stdio.h>
#include <math.h>
```

```
/* #define Time 0 #define GlobalOn 0
*/



//#define ARRAY_SIZE 1797

/*====================*
* S-function methods *
*====================*/

/* Function: mdlInitializeSizes
===============================================
* Abstract:
*    The sizes information is used by Simulink to determine the S-function
*    block's characteristics (number of inputs, outputs, states, etc.).
*/
static void mdlInitializeSizes(SimStruct *S) {
    ssSetNumSFcnParams(S, 0);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        /* Return if number of expected != number of actual parameters */
        return;
    }

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);

    if (!ssSetNumInputPorts(S, 2)) return;
    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortWidth(S, 1, 1);
    ssSetInputPortDirectFeedThrough(S, 0, 1);
    ssSetInputPortDirectFeedThrough(S, 1, 1);



    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 10);

    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}



/* Function: mdlInitializeSampleTimes
=========================================
* Abstract:
*    This function is used to specify the sample time(s) for your
*    S-function. You must register the same number of sample times as
*    specified in ssSetNumSampleTimes.
*/
static void mdlInitializeSampleTimes(SimStruct *S) {
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S,0,FIXED_IN_MINOR_STEP_OFFSET);
} /* static void mdlInitializeSampleTimes(SimStruct *S) {
ssSetSampleTime(S, 0, STEP_SIZE); /* 0.01 is the sample time used in
```

```
the simulation */ /*       ssSetOffsetTime(S, 0, 0.0);  /* use
descrete sample
 */
/* }
*/


#undef MDL_INITIALIZE_CONDITIONS   /* Change to #undef/define to
remove function */ #if defined(MDL_INITIALIZE_CONDITIONS) /*
Function: mdlInitializeConditions
========================================
* Abstract:
*    In this function, you should initialize the continuous and discrete
*    states for your S-function block.  The initial states are placed
*    in the state vector, ssGetContStates(S) or ssGetRealDiscStates(S).
*    You can also perform any other initialization activities that your
*    S-function may require. Note, this routine will be called at the
*    start of simulation and if it is present in an enabled subsystem
*    configured to reset states, it will be call when the enabled subsystem
*    restarts execution to reset the states.
*/
static void mdlInitializeConditions(SimStruct *S) { } #endif /*
MDL_INITIALIZE_CONDITIONS */




#define MDL_START  /* Change to #undef to remove function */ #if
defined(MDL_START) /* Function: mdlStart
=======================================================
* Abstract:
*    This function is called once at start of model execution. If you
*    have states that should be initialized once, this is the place
*    to do it.
*/
static void mdlStart(SimStruct *S) {
    // controller_engaged = 0;
} #endif /*  MDL_START */

/* Function: mdlOutputs
=======================================================
* Abstract:
*    In this function, you compute the outputs of your S-function
*    block. Generally outputs are placed in the output vector, ssGetY(S).
*/
static void mdlOutputs(SimStruct *S, int_T tid) {
    const int_T ARRAY_SIZE=1700;
    /* inputs */
    InputRealPtrsType u0Ptrs = ssGetInputPortRealSignalPtrs(S,0); /* Time */
    InputRealPtrsType u1Ptrs = ssGetInputPortRealSignalPtrs(S,1); /* Controller*/

static real_T pitch_input1[1700] ={25.4039, 25.4039, 25.4039, ....};


static real_T pitch_input2[1700] ={0.0000, 0.0000, -0.0000, ...};


static real_T pitch_input3[1700] ={0.0000, -0.0000, ...};


static real_T roll_input1[1700] ={0.4395, 0.4395, 0.4395, ...};


static real_T roll_input2[1700] ={0.0000, 0.0000, 0.0030, ...};


static real_T roll_input3[1700] ={0.0000, 0.2997, 0.5955,...};
static real_T travel_input1[1700] ={-82.9989, -82.9989,...};

static real_T travel_input2[1700] ={0.0000, 0.0000,...};

static real_T travel_input3[1700] ={0.0000, -0.0003, ...};


real_T Time_current = *u0Ptrs[0];

int_T times=0;

/*real_T pitch pitch_input1[times]; real_T pitch_d
pitch_input2[times]; real_T pitch_dd pitch_input3[times]; real_T
travel travel_input1[times]; real_T travel_d travel_input2[times];
real_T travel_dd travel_input3[times]; real_T roll
roll_input1[times]; real_T roll_d roll_input2[times]; real_T roll_dd
roll_input3[times]; */

real_T GlobalOn  = *u1Ptrs[0]; static real_T Time1 =0;


/* outputs */
real_T *y0 = ssGetOutputPortRealSignal(S,0);  /* x_desired states */

/* code starts here */
if ((GlobalOn >= 1.0))
{// if GlobalOn

 real_T Time=Time_current-Time1;
```

```
times=Time/0.01;


if (times>1700)
{
y0[0] =28.5/57.29;
y0[1] =0;
y0[2] =0;
y0[3] =10/57.29;
y0[4] =0;
y0[5] =0;
y0[6] =300/57.29;
y0[7] =0;
y0[8] =0;
y0[9] =0;
}
else
{
y0[0] =pitch_input1[times]/57.29;
y0[1] =pitch_input2[times]/57.29;
y0[2] =pitch_input3[times]/57.29;
y0[3] =roll_input1[times]/57.29;
y0[4] =roll_input2[times]/57.29;
y0[5] =roll_input3[times]/57.29;
y0[6] =travel_input1[times]/57.29;
y0[7] =travel_input2[times]/57.29;
y0[8] =travel_input3[times]/57.29;
y0[9] =0;
}
```

```
 //if (travel_input1>0)
 //{
 //  y0[9]=2;
 //}
 }// if GlobalOn

 else
 {// if GlobalOn
 Time1   = Time_current; /* Define a variable to reset the time when you
 change the controller*/
 y0[0]   = 0.4434;
 y0[1]   = 0.0;
 y0[2]   = 0.0;
 y0[3]   = 0.0;
 y0[4]   = 0.0;
 y0[5]   = 0.0;
 y0[6]   = -1.449;
 y0[7]   = 0;
 y0[8]   = 0;
y0[9]   = 0;

 }// if GlobalOn

}//

#undef MDL_UPDATE  /* Change to #undef/define to remove function */
#if defined(MDL_UPDATE) /* Function: mdlUpdate
======================================================
* Abstract:
*    This function is called once for every major integration time step.
*    Discrete states are typically updated here, but this function is useful
*    for performing any tasks that should only take place once per
*    integration step.
*/
static void mdlUpdate(SimStruct *S, int_T tid) { } #endif /*
MDL_UPDATE */



#undef MDL_DERIVATIVES  /* Change to #undef to remove function */
#if defined(MDL_DERIVATIVES) /* Function: mdlDerivatives
=================================================
* Abstract:
*    In this function, you compute the S-function block's derivatives.
*    The derivatives are placed in the derivative vector, ssGetdX(S).
*/
static void mdlDerivatives(SimStruct *S) { } #endif /*
MDL_DERIVATIVES */



/* Function: mdlTerminate
=====================================================
* Abstract:
*    In this function, you should perform any actions that are necessary
*    at the termination of a simulation.  For example, if memory was
*    allocated in mdlStart, this is the place to free it.
*/
static void mdlTerminate(SimStruct *S) { }
```

```
/*=====================================================*
* See sfuntmpl.doc for the optional S-function methods *
*=====================================================*/

/*=============================*
* Required S-function trailer *
*=============================*/

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a
MEX-file? */ #include "simulink.c"      /* MEX-file interface
mechanism */ #else #include "cg_sfun.h"       /* Code generation
registration function */ #endif
```

# Bibliography

[1] A.Y. Ng, Daishi Harada and Shankar Sastry. Autonomous Helicopter Flight via Reinforcement Learning. In *Neural Information Processing Systems* 16, 2004

[2] J. Bagnell and J. Schneider. Autonomous Helicopter Control using Reinforcement Learning Policy Search Methods. In *International Conf. Robotics and Automation.* IEEE, 2001.

[3] T. Schouwenaars, B. Mettler, E. Feron, and J. How. Hybrid architecture for full-envelope autonomous rotorcraft guidance. In *American Helicopter Society 59th Annual Forum,* 2003.

[4] G. Taga, Y. Yamaguchi, H. Shimizu. Self-organized Control of Bipedal Locomotion by Neural Oscillators in Unpredictable Environment. In *Biological Cybernetics*, 1991.

[5] G. Taga. A Model of Neuro-musculo-skelatal System for Human Locomotion. In *Biological Cybernetics*, 1995.

[6] S. Schaal, D. Sternad, and C. Atkeson, One Handed Juggling : A Dynamical Approach to a Rhytmic Movement Task. In *Journal of Motor Behaviour*, 1996.

[7] J. A. Ijspert, J. Nakanishi, and S. Schaal. Learning Rhythmic Movements by Demonstration Using Nonlinear Oscillators. In *International Conf. Robotics and Automation.* IEEE, 2000.

[8] M. Ishutkina. Design and implimentation of a supervisory safety controller for a 3DOF helicopter. *Master's thesis, Massachusetts Institute of Technology*, 2004.

[9] S. Bayraktar. Aggressive Landing Maneuvers for Unmanned Aerial Vehicles. *Master's thesis, Massachusetts Institute of Technology*, 2004.

[10] P. Bakker and Y. Kuniyoshi. Robot See, Robot Do: An Overview of Robot Imitation. *AISB'96 Workshop on Learning in Robots and Animals*, 1996.

[11] S. Bayraktar, M. Ishutkina. Nonlinear Control of a 3 DOF Helicopter. *2.152, Nonlinear Control System Design, Project Report*, 2004.

[12] J.J. Slotine and W. Li. *Applied Nonlinear Control*, Prentice-Hall, 1991.

[13] W. Lohmiller, J.J. Slotine. On Contraction Analysis for Nonlinear Systems. *Automatica 34(6)*, 1998.

[14] W. Wang, J.J. Slotine. On Partial Contraction Analysis for Coupled Nonlinear Oscillators.*Biological Cybernetics*, 2004.

[15] W. Lohmiller, J.J.E. Slotine. Applications of Contraction Analysis. *Proc. of the 36th CDC San Diego, CA*, 1997.

[16] S. Schaal, J. Peters, J. Nakanishi, A. Ijspeert. Control, planning, learning, and imitation with dynamic movement primitives. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[17] A.J. Ijspeert, J. Nakanishi, S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. *ICRA*, 2002.

[18] F.A. Mussa-Ivaldi, S.A. Solla. Neural primitives for motion control. *IEEE Journal of Oceanic Engineering*, 2004.

[19] M.N. Nicolescu, M. Mataric. Task learning through imitation and human-robot interaction. *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 2003.

[20] S. Nakaoka, A. Nakazawa, K. Yokoi, K. Ikeuchi. Generating whole body motions for a biped humanoid robot from captured human dances. *ICRA*, 2003.

[21] R.R. Burridge, A.A. Rizzi, D.E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *International Journal of Robotics Research*, 1999.

[22] T. Inamura, I. Toshima, Y. Nakamura. Acquisition and embodiment of motion elements in closed mimesis loop. *ICRA*,2002.

[23] A. Polit, E. Bizzi. Characteristic of Motor Programs Underlying Arm Movements in Monkeys.*Journal of Neurophsiology*, 1979.

[24] E. Bizzi, F.A. Mussa-Ivaldi, N. Hogan. Regulation of multi-joint arm posture and movement. *Progress in Brain Research, Vol. 64*, 1986.

[25] F.A. Mussa-Ivaldi, S. Giszter, E. Bizzi. Motor Space Coding in the Central Nervous System. *Cold Spring Harb Symp Quant Biol Vol. 55* ., 1990.

[26] L. Berthouze, P. Bakker and Y. Kuniyoshi. Learning of Oculo-Motor Control: a Prelude to Robotic Imitation. *Proc. IROS*, 1996.

[27] B.G. Galef. Imitation in animals: History, definition and interpretation of data from the psychological laboratory. *Social Learning: Psychological and Biological Perspectives*, 1988.

[28] A. Billard, M. J. Mataric. Learning human arm movements by imitation: evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems*

[29] A. Fod, M.J Mataric, O.D. Jenkins. Automated Derivation of Primitives for Movement Classification. *Autonomous Robots*, 2002.

[30] E. Bizzi, F.A. Mussa-Ivaldi, S. Giszter. Computations underlying the execution of movement: a biological perpective. *Science 253*, 1991.

[31] F.A. Mussa-Ivaldi, E. Bizzi. Motor learning Through the Combination of Primitives. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 2000.

[32] F.A. Mussa-Ivaldi, J.L. Platton. Robots can teach people how to move their arm. *ICRA*, 2000.

[33] A. Billard, A.J. Ijspeert. Biologically inpired neural controllers for motor control in a quadruped robot. *IEEE-INNS-ENNS International Joint Conference on Neural Networks, Volume 6*, 2000.

[34] E. Frazzoli, M.A. Dahleh, E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Trans. on Robotics*, 2005.

[35] V. Gavrilets, E. Frazzoli, B. Mettler, M. Piedmonte, E. Feron. Aggressive maneuvering of small autonomous helicopters: a human centered approach. *Internatiaonal Journal of Robotics Research*, 2001.

[36] V. Gavrilets, I. Martinos, B. Mettler, E. Feron . Control logic for automated aerobatic flight of a miniature helicopter. *AIAA Guidance, Navigation, and Control Conference and Exhibit, Monterey, California*, 2002.

[37] R. Mahony, R. Lozano. (Almost) Exact path tracking control for an autonomous helicopter in hover manoeuvres. *ICRA*, 2000.

[38] O. Shakernia, Y. Ma, T. J. Koo, S. Sastry. *Asian Journal of Control, 1(3):128–145*, 1999.

[39] D.H. Shim, H.J. Kim, S. Sastry. Control system design for rotorcraft-based unmanned aerial vehicles using time-domain system identification. *Proceedings of the 2000 IEEE International Conference on Control Applications*, 2000.

[40] B. Mettler, E. Bachelder. Combining on- and offline optimization techniques for efficient autonomous vehicle's trajectory planning. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005.

[41] W. Ilg, G.H. Bakir, J. Mezger, M.A. Giese. On the representation, learning and transfer fo spatio-temporal movement characteristics. *International Journal of Humanoid Robotics*, 2004.

[42] J. Mezger, W. Ilg, M.A. Giese. Trajectory synthesis by hierarchial spatio-temporal correspondence: comparison of different methods. *Proceedings of the 2nd symposium on Applied perception in graphics and visualization*, 2005.

[43] L. Sentis, O. Khatib. Synthesis of whole-body behaviors through hierarchial control of behavioral primitives. *International Journal of Humanoid Robotics*, 2005.

[44] K. Matsuoka. Sustained oscillations generated by mutually inhibiting with adaptation. In *Biolagical Cybernetics*, 1985.

[45] F. A. Mussa-Ivaldi. Nonlinear force fields: a distributed system of control primitives for representing an learning movements. *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1997.

[46] L. Righetti, J. Buchli, A.J. Ijspeert. From dynamic hebbian learning for oscillators to adaptive central pattern generators. In *Proceedings of the Third International Symposium on Adaptive Motion in Animals and Machines AMAM2005*, 2005.

[47] A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *in Advances in Neural Information Processing Systems 15*, 2003.

[48] F.A. Mussa-Ivaldi, S.F. Giszter, and E. Bizzi. Linear combinations of primitives in vertebrate motor control. In *Proceedings of the National Academy of Sciences*, 1994.

[49] S. Grillner. Locomotion in vertebrates: central mechanisms and reflex interaction. In *Physiolaogical Reviews* ,1975.

[50] A.M. Arsenio. On stability and tuning of neural oscillators: application to rhytmic control of a humanoid robot. In *Neural Networks*, 2004.

[51] L. Righetti and A.J. Ijspeert. Programmable central pattern generators: an application to biped locomotion. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, 2006.

[52] K. Seo, J.J. Slotine. Models for global syncronization in fish and salamader locomotion. In *Nonlinear Systems Laboratory Report (MIT)*, 2006.