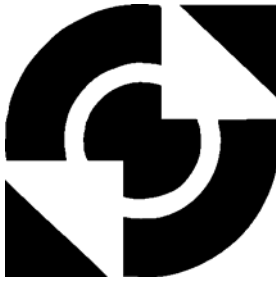
University of Twente

EEMCS / Electrical Engineering
Control Engineering



An optimal controller for Desdemona for an optimal feeling

Pieter Schutyser

M.Sc. Thesis

Supervisors
prof.dr.ir. J. van Amerongen
dr.ir. S. Stramigioli
dr.ir. M.Wentink, ir. V. Duindam

March 2005

Report nr. 010CE2005
Control Engineering
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Public version

In this report confidential figures and tables are not displayed. For more information see [2].

Summary

Desdemona is a new prototype disorientation simulator which can generate acceleration profiles. In this report the design of an optimal controller for Desdemona is described. An optimal controller calculates the optimal movements of Desdemona within the limits of Desdemona. The limitations are caused by maximum displacement of the joints and a maximum in the power/torque of the motors, where a dynamic model is used to calculate the power and torque of the motors. The optimal controller optimizes the complete trajectory if it is known beforehand or until a limited time horizon if the trajectory can only be estimated for a limited time horizon.

The optimal controller is realized and tested for Desdemona with one degree of freedom (d.o.f.), called *dynamic flight simulator* also with two d.o.f. where the central yaw and the vertical sledge are controlled. And finally with three d.o.f where yaw of the cabin is also included. If the d.o.f. is one then the optimization can be solved for the case the trajectory can only be estimated for a certain time horizon. This is the case for example with *man in the loop*.

The results show that the optimization works especially for limited d.o.f. An increased number of d.o.f. makes the problem much harder to solve, which causes a decrease in performance. The performance can be increased by choosing smart initial values. Initial values for the optimization can be generated with the existing controllers. The results of the optimization will be better or equal to the results of the initial values from the controller. This makes the optimization method an interesting extension for the existing controllers. For Desdemona as *dynamic flight simulator* this is shown.

The off-line optimization is ready to be implemented for simulations. Some work has to be done for realizing the on-line optimization. The path of the simulated model has to be estimated in the future and the calculation time has to be reduced.

Preface

After a couple of studies I thought I was ready and went to the University. Now almost six yeas later Im writing the last pages of my final thesis for completing my Master of Science study in the field of Electrical Engineering at the University of Twente. The thesis is about a subject that has my interest from the beginning of my technical study. It is about a moving object which is controlled by a controller which looks complex but is, if you look to the mathematics, quite simple. I have no idea why I'm interested in theoretical mathematics, moving objects and the complete area between. But it is for me fascinating to see that all these things together can result in a piece of advanced technology.

I would like to thank some people for their support during this assignment. First I would like to thank dr.ir. Stefano Stramigioli, he deserves a lot of appreciation for all the advice he gave during my work. I also would like thank dr.ir. Mark Wentink for all the enthusiastic support he has given me by phone and 'live' discussions. Also I would like to thank him for introducing me in the fascinating world of disorientation simulators.

The last but not the least, I also would like to thank my family, who always give support and courage during my studies. And my friends who make life much better in and outside Enschede. Thanks to all of you.

Nomenclature

Symbols

$\ddot{p}$	Acceleration in the origin of the cabin , see equation (3.2)
$\ddot{p}_d$	Desired acceleration in the cabin , see equation (3.1)
τ	Torque of the motors
τ_{max}	Maximum torque of the motors
$\tilde{T}_k^{i,j}$	Twist from k to j seen from coordinates i
$C(q, \dot{q})$	Coriolis matrix
e_n	n-axis of the coordination frame. With n as x,y,z
f_0	Cost function in the analytical optimization
f_{feas}	Feasibility function, the part in the cost function that represent the constraints
G	Gravity , see equation (3.2)
g	Gravity constant
H_m^n	Matrix that expresses a general change of Cartesian coordinates from coordinates m to n
h_n	Step-size (sample time) with n as the sampled item (S_d,q...)
$I^{0,i}$	Generalized inertia matrix of body i expressed in coordinate frame 0
$i_{e_n^m}$	Inertia in direction n of element m
J	Cost function
J^n	Jacobian matrix (linear relationship between the joint velocities and the end-effector n twist)
k	Discrete time step
$K()$	End cost
M	Modulus of the path of Desdemona , page 46
$M(q)$	Mass matrix
M_e	Modulus of the estimated path in the future , page 46
M_n	Mass of element n
N	Conservative force
O_n	n^{th} Coordinates frame

p	Translation vector
P_{max}	Maximum power of the motors
Q	(Weighing matrix)Matrix which give to some extent the importance of the optimization per d.o.f. of the path
q	Joints of Desdemona
$q_n max$	Maximum position q_n can have
R	Rotation matrix
S	Path of Desdemona , see equation (3.2)
S_d	Desired path , page 7
$S_{d,perc}$	Perceived S_d
S_{perc}	Perceived S
T_c	Calculation time
T_h	Horizon Time
V	Potential energy
n_s	Sampled n

Abbreviations and used terms

d.o.f.	degree of freedom
dom	Domain
ips	Neural discharge rate (impulses per second)
Iteration	A iteration is one step in the optimization routine
MPC	Model Predictive Control
RMS	Root Mean Square
Run	A run is a complete optimization, which includes a number of iterations

Contents

Chapter 1

Introduction

Human beings sense movements of their bodies. The interpretation of the sensed movements is a complex process and a broad research topic. For new research topics Desdemona is developed, it is a new kind of motion platform. With Desdemona it is possible to do research on the human perception model and to use the gained knowledge for applied research and applications. Some examples are the training for flying a plane, driving a car and steering a ship.

Figure 1.1: Desdemona at TNO

Current motion platforms are limited in their simulation of sustained acceleration. Desdemona has an innovative kinematic design (see figure 1.1) so that it is less limited than conventional simulators. Because the control system of existing motion platforms are not optimal for Desdemona, a new kind of controller is discussed in this report.

1.1 Desdemona in comparison with other disorientation simulators

The [1] most common motion platform is the Stewart six-pod platform, which allows all six degrees of freedom. Despite the fact that accelerations can be induced only for a short time, for many applications such as training for transport aircraft this platform is sufficient. However, for fighter aircraft with pulling high sustained G-loads, this platform is not sufficient for realistic mission rehearsal. Present developments show that for flight simulation with higher G-loads, centrifuges are built with roll and pitch options of the gondola. Problem with these simulators is that they are able to simulate the G-load, but that the corresponding angular accelerations during the onset, introduce conflicting motion information to the pilot's equilibrium system. This may lead to unwanted side effects such as disorientation and motion sickness. Desdemona is a research simulator, which combines the possibilities of common Stewart platform with the possibility of the sustained G-load, however, without the co-varying rotational accelerations.

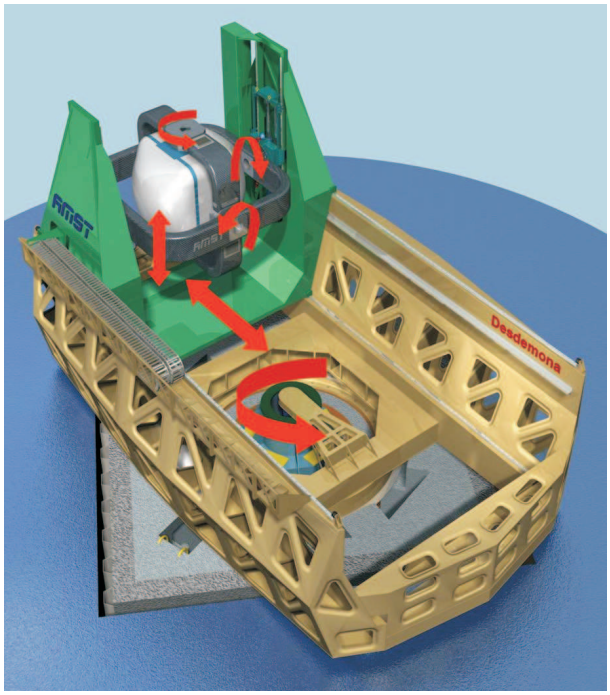


Figure 1.2: Desdemona movements

The Desdemona concept was developed by TNO Human Factors in co-operation with AMST Systemtechnik [2]. Figure 1.2 shows the concept. It consists of a fully gimbaled cockpit, which allows for unlimited rotation in all directions. The pilot with his head in the center of rotation, controls the (flight) instruments and has a view outside via up-to-date visuals. For simulation purposes different models and databases are available. The cockpit can move along an 8 meter horizontal track. The cockpit can also move vertically over 2 meter. Finally, the horizontal track is rotated around a vertical axis, which means that centrifuging is also possible. The working principle of a normal centrifuge is a variation on the angular velocity with constant eccentricity resulting in a varying G-load. However, keeping the angular velocity constant and varying the amount of eccentricity is another way of varying the G-load. Desdemona applies both with a maximal G-load of 3g. The main characteristic of the concept, i.e. the variation of the G-load by varying the eccentricity, has consequences for simulation. It means that the onset of a sustained G-load is not necessarily accompanied with a strong angular acceleration, as is the case in the conventional centrifuge. It is possible to start the rotator subliminal up to the desired speed with the cockpit in the center position, and subsequently move the cockpit away from the axis without varying the angular velocity. It is obvious that one should take into account the linear Coriolis accelerations during the movement of the cockpit on the track. This

[1] This part is mainly based on [2]

example also shows the difficulty of controlling Desdemona. It is possible to simulate a transversal acceleration by moving the cockpit on the horizontal track while possibly rotating the horizontal track. The choice depends on the magnitude and the duration of the desired acceleration.

1.2 Description of the problem

The control problem of the Desdemona is significantly different from that of the Steward platform, because the directions of movement of the Steward platform are in general the same as the model movements. Desdemona on the other hand can also use rotation to simulate a linear acceleration.

To find the optimal acceleration and rotation path, an optimal controller is needed. The optimal path is defined as *the path that is as close as possible to the desired path.*The optimal controller decides which movements of Desdemona give the best simulation of the desired path with taking constraints such as maximum power in account.

In this report an optimal controller is designed for the cases in which the acceleration trajectory is known and unknown beforehand.

1.3 Outline

The outline of this report is as follows. In chapter 2, the objectives of this report are given. Then in chapter 3 the optimization problem is defined which forms the basis of the optimization. In chapter 4 the numerical optimization methods are discussed and the problem is solved for the most basic off-line case. Then in chapter 5 the numerical optimization is redefined to make it more practically applicable. In chapter 6 the human perception model is included. In chapter 7 an on-line optimization for Desdemona as dynamic flight simulator is done. Finally, in chapter 8 conclusions and recommendations of this research are drawn.

Chapter 2

Objectives

Desdemona combines the degrees of freedom (d.o.fs) of a centrifuge simulator, used for sustained G-load simulation, with additional d.o.f.'s to give it the full 6 d.o.f. motion cueing possibilities of a standard hexapod simulator. In centrifuge mode, Desdemona rotates around its central yaw axis while in hexapod mode this axis, in combination with the main arm, is used to simulate linear accelerations. In both modes, attitude is simulated with the gimbaled system. The combination of a centrifuge and hexapod simulation mode provides huge advantages in simulations of highly agile maneuvers with sustained G-load (e.g.: F16 combat simulation, unusual attitude recoveries, etc.). The main objective of this report is to design an optimal motion cueing and control algorithm that takes full advantage of the available d.o.f's of Desdemona.

Goal

The simulation of high frequency, agile motion in hexapod mode in combination with, or followed by, sustained G-loads in centrifuge mode requires advanced motion cueing and control algorithms. Although algorithms for the centrifuge and hexapod mode are developed by AMST separately, an algorithm that combines the two has not yet been developed. The goal of this work is to develop an optimal control algorithm that combines the centrifuge and hexapod capabilities of Desdemona during the simulation of highly agile maneuvers (high frequency accelerations) in combination with, or followed by, sustained G-loads (low frequency accelerations). In this scope, optimal is defined as:

- minimization of the perception of differences in the simulator
- minimization of false cues,
- within the limits of Desdemona (structural limits and drive limits).

Although pre-defined maneuvers will be used first, the ultimate, long-term goal is an optimal control algorithm that can be implemented in pilot-in-the-loop simulations (which will result in maneuvers that are hard to predict beforehand).

Methods

In order to reach the stated goal, the following sub-goals are defined:

- Study of literature on: modeling and simulation of robotic dynamics, optimal control concepts, Desdemona specifications, motion cueing in conventional flight simulation,
- Developing a dynamic simulation model of Desdemona,
- evaluating suitable optimal control concepts,

- defining an optimization cost function (incorporating perception of difference, false cues, drive performance parameters, etc.)
- developing & implementing of optimal control algorithms in the dynamic simulation model
- evaluating the optimal control algorithm, especially the 'optimal' motion characteristics during the transition from hexapod motion types to centrifuge motion types (and vice versa).
- Working towards a real-time optimal control algorithm that can be used during *man in the loop* simulations.

Chapter 3

Problem definition for an optimal controller

The main goal of the optimal controller is to find optimal movements for the joints $q(t)$ of Desdemona within the physical constraints such as motor power, dimensions etc.

To define the optimal control problem, the path and the error will be defined first and then the physical constraints. At the end of this chapter a simplified model of Desdemona will be outlined.

3.1 Defining the path to be followed

Desdemona is a simulator, in which one feels the acceleration from the simulated model. The model gives output information on the rotation speed $\omega(t)$ and the translation acceleration $\ddot{p}(t)$ and gravity $G(t)$. The calculated path from the model is the desired path for Desdemona and is indicated with:

$$S_d(t) = \begin{pmatrix} \omega_d(t) \\ \ddot{p}_d(t) + G(t) \end{pmatrix} \tag{3.1}$$

The mechanical part of Desdemona has to follow that path. The movement of Desdemona can be written in the same way as the movement of the model, but now as function of the joints and it derivatives $q(t), \dot{q}(t), \ddot{q}(t)$.

$$S(t) = \begin{pmatrix} \omega(q(t), \dot{q}(t)) \\ \ddot{p}(q(t), \dot{q}(t), \ddot{q}(t)) + G(q(t)) \end{pmatrix} \tag{3.2}$$

Where omega can be extracted from the twist (see also A.6): $T_{head}^{head,earth} = \begin{pmatrix} \omega \\ v \end{pmatrix}$ and $\ddot{p}$ can be calculated with equation A.18. The gravity vector $G(q(t))$ can be calculated with the help of the rotation matrix (see A.19).

The goal of Desdemona is to follow the path of the model. The error that Desdemona makes can be written as: $\Delta S(q(t), \dot{q}(t), \ddot{q}(t), t) = S_d(t) - S(q(t), \dot{q}(t), \ddot{q}(t))$. The error that the pilot will observe is not $\Delta S(q(t), \dot{q}(t), \ddot{q}(t), t)$ because the vestibular nerve system in combination with the visual input does not give an exact impression of $\Delta S(q(t), \dot{q}(t), \ddot{q}(t), t)$. It is important to keep this in mind designing the optimum controller. This fact will be discussed later (chapter 6). But at this point it will be assumed that the pilot is observing $\Delta S(q(t), \dot{q}(t), \ddot{q}(t), t)$ as fault in the path.

3.2 Physical bounds of Desdemona

Some of the movements of Desdemona are physically limited because of the finite stroke of the joints. This limitation is given in equation (3.3). The joints of the vertical sledge and heave, respectively $q_2(t)$ and $q_3(t)$, are

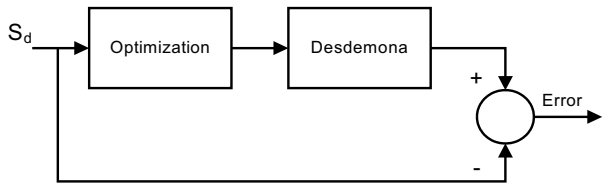


Figure 3.1: Optimal control problem

limited because they are the only two translation joints. The others are unbounded joint rotation movements.

$$\begin{aligned} -q_2\,max \leq q_2(t) \leq q_2\,max \\ -q_3\,max \leq q_3(t) \leq q_3\,max \end{aligned} \tag{3.3}$$

Other bounds on the velocities and accelerations prevent large mechanical stress and vibrations to occur in the system.

$$\begin{aligned} -\dot{q}_{max} \leq \dot{q}(t) \leq \dot{q}_{max} \\ -\ddot{q}_{max} \leq \ddot{q}(t) \leq \ddot{q}_{max} \end{aligned} \tag{3.4}$$

Desdemona is actuated by a number of motors and every motor has limited power and torque.

$$\begin{aligned} -P_{i,max} \leq \tau_i(t)\dot{q}_i(t) \leq P_{i,max} \\ -\tau_{max} \leq \tau(t) \leq \tau_{max} \end{aligned} \tag{3.5}$$

The power and torque relations are calculated in section A.4

3.3 Time horizon

Time horizon (T_h) is defined as the amount of time in the future for which the desired path, $S_d(t)$, is known.
The time horizons depend on the purpose the Desdemona is used. They can be grouped in two categories: *man in the loop* and *no man in the loop* (for example a roller coaster). If Desdemona works with *man in the loop*, the time horizon will be short although it is possible to predict a few seconds. This depends of course on the model which Desdemona is simulating. For example the path of a car which is driving on a road with no side lanes is easier to predict then a car on a road with side lanes. If there is no *man in the loop* then the time horizon runs till the end of the simulation, because the path is predefined.

3.4 Mathematical formulation of the optimization problem

Figure 3.1 shows the optimization problem. The problem can be calculated off-line for cases where there is no *man in the loop*. In case there is a *man in the loop* simulation, then the state at the end of the time horizon (final state) of Desdemona is important, since future simulation errors need to be avoided[1]. Because the path S_d can not be calculated in advance, the optimization should be done online.

The definition of the problem is divided an on-line and off-line optimization.

[1]With *man in the loop* the simulation time is longer than the time horizon and the path of Desdemona is optimized with several optimizations which start after each other. An end cost avoids that at the end of an optimization Desdemona comes in a certain position which limits the accelerations that can be simulated. This can happen if a gimble lock occurs or if the vertical sledge reach the end of the vertical track. In chapter 7 this is discussed

The off-line optimization problem can be written as:

$$\min_{\ddot{q}} \; J = \int_{t_0}^{t_e} \Delta S(q(t), \dot{q}(t), \ddot{q}(t), t)^T Q \Delta S(q(t), \dot{q}(t), \ddot{q}(t), t) \, \mathrm{d}t \tag{3.6}$$

subject to the Dynamical system: $\ddot{q}(t) = f(q(t), \dot{q}(t), \tau(q(t), \dot{q}(t), \ddot{q}(t)))$ (see section A.4), and the constraints:

$$\begin{aligned} |q_2(t)| - \quad & q_{2\,max} \quad \leq 0 \\ |q_3(t)| - \quad & q_{3\,max} \quad \leq 0 \\ |\dot{q}(t)| - \quad & \dot{q}_{max} \quad \leq 0 \\ |\ddot{q}(t)| - \quad & \ddot{q}_{max} \quad \leq 0 \\ |\tau(t)| - \quad & \tau_{max} \quad \leq 0 \\ \left|\tau_i(t)\dot{q}_i(t)^T\right| - \quad & P_{i,max} \quad \leq 0 \end{aligned} \tag{3.7}$$

where Q is a semi-positive definite weighing matrix. The elements of the matrix give to some extent the importance of the optimization per direction of the path. $\ddot{q}$ and τ are calculated in A.27 and A.26

The on-line (real time) optimization problem can be written as:

$$\min_{\ddot{q}} \; J = K(q(t_e), \dot{q}(t_e), t_e) + \int_{t_0}^{t_e} \Delta S(q(s), \dot{q}(s), \ddot{q}(s), s)^T Q \Delta S(q(s), \dot{q}(s), \ddot{q}(s), s) \, \mathrm{d}s \tag{3.8}$$

And is subjected to the same constants as in the off-line optimization. Where $K(q, \dot{q}, t_e)$ gives the inverse of the quality of the state of Desdemona at t_e. Quality of the state is defined as the possible accelerations Desdemona can simulate at that point. If Desdemona is in a singular point then K will be large and if Desdemona can simulate all accelerations then K will be small.

3.5 Example of the optimization

3.5.1 Introduction

The problem of finding the right path for Desdemona can be made a bit easier by reducing it to a problem with two degrees of freedom. The problem itself will be the same ”find the optimal path”.

In this example only joints q_1 and q_2 are considered. The other joints are fixed and for the fixed parts the inertia is zero. The acceleration that the pilot should feel is taken as a cosine in the y direction of the cockpit. At this point the other accelerations (false cues) are not include in the optimization. The path S_d can be defined by:

$$\ddot{y} = \cos(wt) \tag{3.9}$$

$$S_d(t) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \ddot{y} \\ 0 \end{pmatrix} \tag{3.10}$$

For the error only the acceleration in the e_y direction is taken into account. Thus

$$Q = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The constraints which are mentioned in paragraph 3.2 are valid.

The optimal controller should calculate the movements of the joints. The expectation is that if the frequency of $\ddot{y}$ is high enough, the linear sled will make sine movements and the rotation of the central yaw will remain zero. But for low frequencies the linear sled will not have enough space to move. It will reach the end of the vertical axis. Desdemona will use the central yaw to generate an acceleration in the y direction of the cockpit by centrifugal acceleration.

3.5.2 Mathematical formulation

The problem can be formulated as:

$$\min_{\ddot{q}} \; J = \int_{t_0}^{t_e} \Delta S(q(t), \dot{q}(t), \ddot{q}(t), t)^T Q \Delta S(q(t), \dot{q}(t), \ddot{q}(t), t) \; dt \tag{3.11}$$

$$\begin{aligned} |q_2(t)| - \quad & q_{2\,max} \quad \leq 0 \\ |\dot{q}(t)| - \quad & \dot{q}_{max} \quad \leq 0 \\ |\ddot{q}(t)| - \quad & \ddot{q}_{max} \quad \leq 0 \\ |\tau(t)| - \quad & \tau_{max} \quad \leq 0 \\ \left|\tau(t)\dot{q}(t)^T\right| - \quad & P_{max} \quad \leq 0 \end{aligned}$$

The S and the τ vector are calculated in appendix B

3.5.3 Outline of analytical solution

One of the basic methods to solve optimal problems is the *Minimum Principle of Pontryagin* [18]. To solve the optimal problem with the minimum *principle of Pontryagin*, the dynamic equation has to be rewritten as a first order differential equation and the constraints are rewritten as extra cost. The solution of the problem can then be rewritten as 8 ODE's (ordinary differential equations) where 4 ODE's have an initial value and 4 a final value. This is done in Appendix B.1.

The ODE's can not be solved analytically with symbolic computer programs like Maple. It is possible to solve the equations numerically with the Matlab function *ode45* which is based on an explicit *Runge-Kutta* [7]. The difficulty is that the method can only handle initial values. So the initial values have to be chosen in such a way that p (the co-state, see section B.1) is zero at t_e. For some x_0 and some trivial weight functions of the cost function this is possible. With the *Newton* method [11] a solution could be found. This is one of the numerical methods to solve an optimal problem, that will be described in the next chapter.

Chapter 4

Numerical optimization

4.1 Introduction to numerical optimization

In this section the analytical optimization problem will be redefined as a numerical optimization problem. This numerical problem will be compared to other standard optimization problems to determine the right numerical optimization method. After that, the optimization method and their limitations will be discussed. At the end of this chapter the example from section 3.5.3 is solved using numerical optimization.

4.2 Redefining the analytical optimization to a numerical problem

The off-line optimization problem is given in equation 3.6. For the numerical case the cost function has to be written in a discrete form because integrating and differentiating are done in continues time and discrete time differently.
The integral of the joint vectors $q(k)$ is calculated with the discrete method Runge-kutta [7], where $\ddot{q}(k)$ is taken as input.
Thereafter the cost function and the constraints are be calculated. The integral in the cost function can be replaced with a sum operator, since the integral only integrates discrete values. This is elaborated in section D.2.

The input for the discrete cost function is $\ddot{q} \in \mathbb{R}^{M \times N}$, where M is the number of d.o.f of q, and N the number of time steps. All the time steps have to be optimized, therefore the number of parameters which have to be optimized is $M \times N$.

The discrete optimization problem can now be written as:

$$\min_{\ddot{q}} \; J = \sum_{k=t_0}^{t_e} \Delta S(q(k), \dot{q}(k), \ddot{q}(k), t(k))^T Q \Delta S(q(k), \dot{q}(k), \ddot{q}(k), k) \tag{4.1}$$

subject to

$$\begin{aligned}
|q_2(k)| - \quad & q_{2\,max} \quad \leq 0 \\
|q_3(k)| - \quad & q_{3\,max} \quad \leq 0 \\
|\dot{q}(k)| - \quad & \dot{q}_{max} \quad \leq 0 \\
|\ddot{q}(k)| - \quad & \ddot{q}_{max} \quad \leq 0 \\
|\tau(q(k), \dot{q}(k), \ddot{q}(k))| - \quad & \tau_{max} \quad \leq 0 \\
\left|\tau_i(q(k), \dot{q}(k), \ddot{q}(k))\dot{q}_i(k)^T\right| - \quad & P_{i,max} \quad \leq 0
\end{aligned} \tag{4.2}$$

Another way of writing J is

$$J = \sum_{k=t_0}^{t_e} \|f(t(k), \ddot{q}(k)\|^2 \tag{4.3}$$

with

$$f(t(k), \ddot{q}(k)) = \sqrt{Q}\Delta S(q(k), \dot{q}(k), \ddot{q}(k), k) \tag{4.4}$$

and $\sqrt{Q}$ such that $(\sqrt{Q})^T\sqrt{Q} = Q$

4.3 Classifying the numerical optimization problem

The optimization problem can be grouped in two differed classes [17]. The first group is where the cost function depends on integer values. Examples where the cost function depends on an integer are routing in batch processes or path-length problems. The second group, is where the cost function depends on parameters with a real value, can be separated in the following subgroups.

Linear programming problem The cost function is linear or affine[1]

Quadratic programming problem The cost function is quadratic and the constraints are linear or affine.

Nonlinear optimization problem The cost function is nonlinear, non-convex.

Convex optimization The cost function is a convex function. (A convex function has one local optimum (the global minimum), see section C.1.3)

The numerical Desdemona optimization problem can be placed in the group nonlinear optimization problems. The numerical cost function defined in 4.1 is nonlinear and not quadratic because ΔS is not linear and the cost function is not a quadratic function of the inputs. Another reason is that the cost function is non-convex and we expect local minima. A nonlinear optimization problems can be grouped in several classes. From equation 4.3 it is obvious that the Desdemona problem has the form which is called a least-squares (data-fitting) problem. The least-squares (data-fitting) problem has the from of:

$$\min_x \; J(x) = f_1(x) + f_2(x) + \ldots + f_n(x)$$

4.4 Methods for solving nonlinear least-squares optimization problems

Most of the numerical optimization techniques work in combination with the *Newton* method (see appendix C.2). The *Newton* method in combination with the line search algorithm is selected for nonlinear least-squares optimization problems because it is used in [17] and [8], in [9] this method is also mentioned as option. In [4] this method is described for solving a Model Predictive Control (MPC) problem. A MPC problem is a online optimization problem. A global overview of the complete optimization method can be found in section D.1.1.

4.4.1 Methods with direction determination and line search

Each iteration step of these methods consists of a direction determination followed by a line search, i.e., an optimization along this determined direction. The basic idea is to choose a search direction d_k at an initial point x_k, and then to minimize the objective function J(x) along a line

$$x_{k+1} = x_k + d_k s \quad s \in \mathbb{R}$$

Where x_k and d_s are vector and s is a scalar.
Now a one dimensional minimization problem can be obtained.

$$\min_s J(x(k) + d(k)s)$$

[1]Example: the function $f(x) = Ax + B$ is linear if $B = 0$ and affine if $B \neq 0$

The starting point for the next iteration is then given by

$$x(k+1) = x(k) + d(k)s_{opt}(k)$$

where

$$s_{opt}(k) = \min_{s} J(x(k) + d(k)s)$$

The search direction can be calculated with the *Newton* method $d(k) = -H^{-1}(x(k))\nabla^T J(x(k))$, or any other *quasi-Newton* method.

For the determination of the step size of s are several methods available [17].

4.5 Limitations to the numerical optimization

The goal of every optimization is to find fast the global minimum of the optimization problem. This is a difficult problem because as stated before our optimization problem is a nonlinear and non-convex problem. This is also mentioned in [4] for solving a nonlinear MPC problem (which is comparable to our problem): "*The solution of this problem requires the consideration (and at least partial solution) of a non-convex, nonlinear problem (NLP) which gives rise to a lot of computational difficulties related to the expense and reliability of solving the NLP online*". Reference [13] shows that the minimum number of computations required to compute the global minimum of a general differentiable function in n variables, within an accuracy of ε, grows like,

$$\left(\frac{1}{\varepsilon}\right)^n \tag{4.5}$$

roughly speaking, the complexity of a search over an n-dimensional ε-grid. If the function is convex, then the function becomes

$$n \ \log\left(\frac{1}{\varepsilon}\right) \tag{4.6}$$

Taking e.g. ε equal to 10^{-3} and $n = 10$, then we have the following complexity measures: for the non-convex function 10^{30} and for convex function equal to 30. This result shows that a non-convex problem is hard to solve.

In a nonlinear and non-convex problem the used *Newton* method always finds local minima instead of global minima; this is unavoidable [17][9][3]. The only method to avoid this is to start optimizing with different initial values so that several local minima will be found, and hopefully (certainty is never present) the lowest is the global minimum.

The duration of the optimization is something which should stay small. This is specially imported for real time applications (like model predictive control). The main time consuming operation is the calculation of derivative of the cost function which has to be calculated in the *Newton* method. The calculations can be done in two ways. The best (fastest) method is to find an expression for the derivative, but this is not always possible, in which the derivative has to be determined by numerically. It can be done by varying the input of the cost function and to calculate the output. This is a time consuming operation but the only way to find the derivative of our optimization problem[2].

4.6 Converting a constrained to an unconstrained optimal problem

Nonlinear inequality constraints usually make the optimization problem more difficult. One common way to tackle the nonlinear inequality problem is to incorporate the constraints function in the objective function, using a barrier function.

[2]In the case of the dynamic flight simulator it is possible to calculate the derivative symbolical because of some trivial simplifications (see section 7)

If the optimization function has the constraint $g(x) \leq 0$ then the basic idea of the barrier function is to introduce a feasibility function J_{feas} of the form

$$J_{feas}(x) = \begin{cases} 0 & g(x) \ll 0 \\ \infty & g(x) \to 0 \end{cases}$$

and add this function to the original cost function

$$\min_x \left[J(x) + J_{feas}(x) \right]$$

In this way the cost function will increase significantly if the inequality constraints is not met, an example is:

$$J_{feas} = -\frac{1}{\beta g(x)}$$

with $\beta > 1$

4.7 Example of numerical optimization

The example which was drawn up analytically (see section 3.5.3) is now solved numerically. The path S_d (eq. 3.9) will be redefined, namely $\ddot{y} = 5\cos(2t)$
The constraints will be included using a feasibility function. It is taken the same as in the analytical optimization (eq. B.13).

The used parameters are: $M_1 = 1$ $i_{e_z^1} = 1$ $M_2 = 2$ $i_{e_z^2} = 2$ (the other elements of Desdemona are assumed weightless) $a = 0.1$ $b = 1e^{-3}$ $c = 1e^{-5}$ (These parameters are taken as example, they are not realistic). The step-size is taken 0.1 sec and the total time is 5 sec.

The initials $q(t_0), \dot{q}(t_0))$ are taken random just as $\ddot{q}(t_0), \ddot{q}(t_1), \ldots, \ddot{q}(t_e)$. Now the optimization can be done, using 40 iterations in combination with the line search method.

This optimization has been done 500 times (runs), every run with different initial values. This is done because the line search method finds a local optimum and this way it was tried to find the global minimum. These runs are sorted on performance (see figure 4.1), where the performance is the cost function without feasibility function . It can be seen that the initial values have an influence on the final results of the optimization.
In figure 4.2 the results are given for run 300, so there are thus 299 runs with a better performance. The movement Desdemona is making is rotating and making small sinus movements around the origin with the vertical sledge. It can be seen that the accelerations S are almost the same as S_d. The values of joint vector q_2 are bounded. The runs with a better performance have a similar behavior, the ΔS becomes smaller and q_2 remains small. The main difference is that the size of ΔS is smaller in the beginning of a run. This is due to better initial values.
The numerical optimal control is thus working properly.

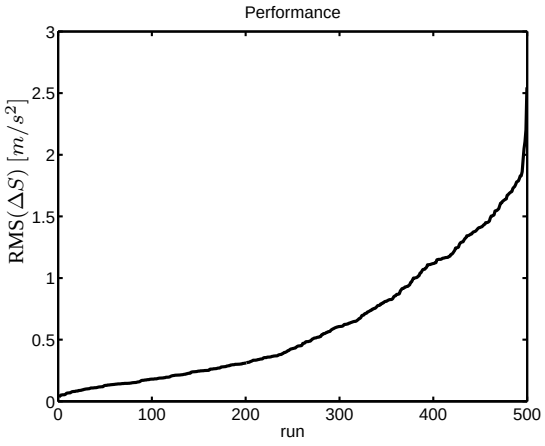


Figure 4.1: This is a plot of the 500 optimization runs. (A run is the final result of an optimization which consists of 40 iterations) These runs are sorted on their performance, where the performance is given as the root mean square (RMS) of the cost function without feasibility function. This is plotted on the y-axis). The initial values are taken randomly every run.

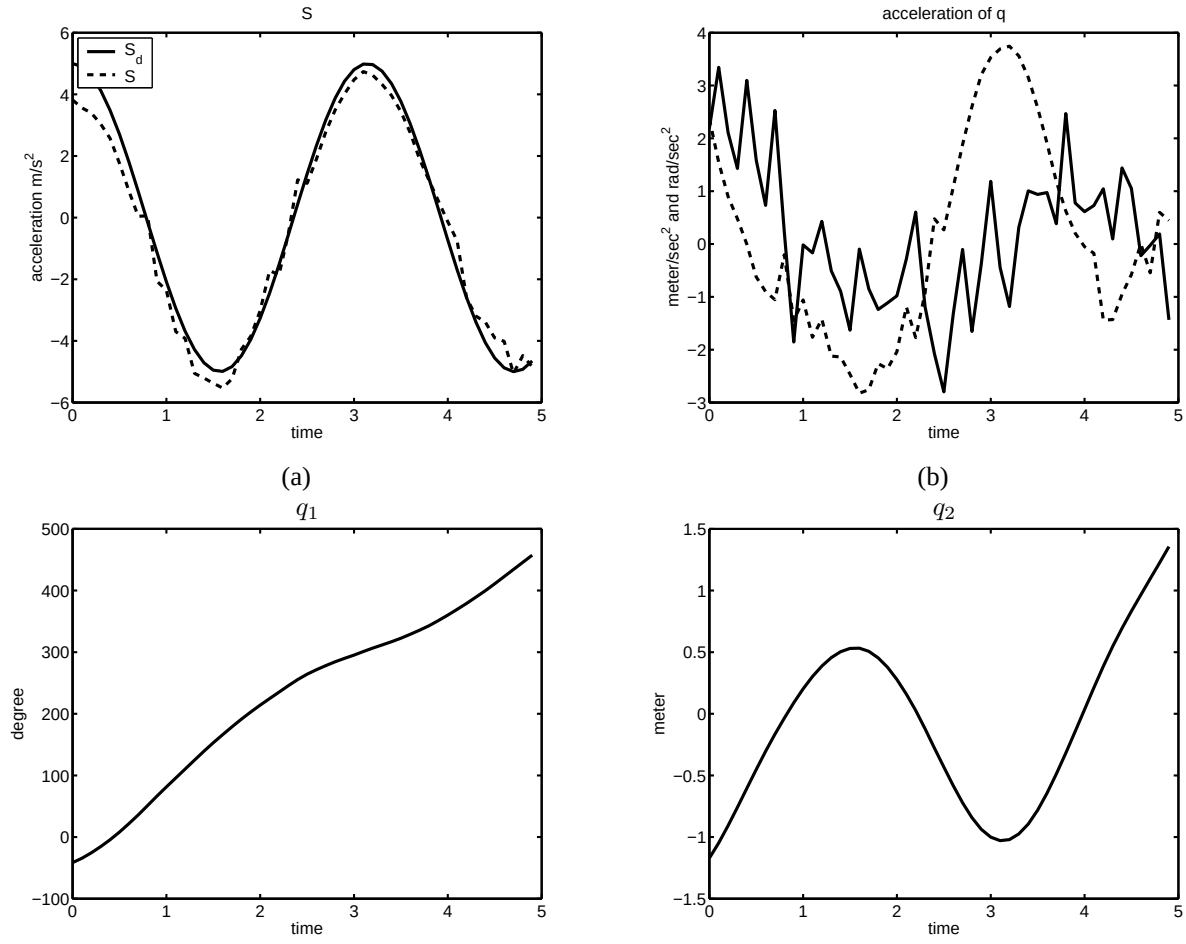


Figure 4.2: In (a) the S (doted line) and S_d (Solid line) are plotted. In (b) the vector $\ddot{q}_1$ (solid line) and $\ddot{q}_2$ the (doted line) are plotted. In (c) and (d) the vector q_1 and respectively q_2 are plotted. The initial velocity is $\dot{q} = [63.6\, degree/s\ 1.13m/s]$ (These results are not based on real parameters).

Chapter 5

Practical extensions of the numerical optimization

In the previous chapter the optimization problem is solved for the most basic case. In this chapter the optimization problem is redefined to make it a more practical applicable optimization problem. Therefore the feasibility function is changed and the initial values of Desdemona ($q(t_0), \dot{q}(t_0)$) are included as optimization parameter. The sample times and stop criteria are chosen more carefully. At the end of this chapter some more realistic paths are defined and Desdemona is optimized with real parameters for the two and three d.o.f. cases.

In this chapter the following terminology is used: with *run*, a complete optimization. A complete optimization consists of a number of iterations. The performance of a run is given as the root mean Square (RMS) of ΔS.

$$RMS(\Delta S) = \sqrt{\frac{\sum_{i=1}^{n} \Delta S_i^2}{n}} \tag{5.1}$$

These RMS sorted with the best run first is plotted in the next sections for a number of runs to compare several different optimization configurations.

5.1 Barrier functions

The barrier function (see section 4.6) is taken quadratic in the example (section 3.5 or 4.7). This is not a realistic barrier function because it results in costs when there is no reason for. This results in unwanted restriction of the joins and motor. This can be seen for q_2 in figure 5.1.a. if $q_2 = 2$ then it is not near the end of the track but there is already a cost.

A good cost function is a cost function where the cost is low and constant if the constraints are not violated, if the constraints are (almost) violated then the cost function should increase significantly. The function should be continuous which is important when taking the derivative in the *Newton* method.

An example of a good barrier function is given in eq. 5.2 and can be seen in figure 5.1.b.

$$J_{feas}(q_2) = \begin{cases} ((q_2)^2 - 3.7^2)^2 & |q_2| > 3.7 \\ 0 & |q_2| \leq 3.7 \end{cases} \tag{5.2}$$

To evaluate the cost function the experiment from section 4.7 is repeated, but now with the barrier function which is given in equation 5.2. The results are given in figure 5.2. The number of runs is chosen in such away the calculation time is reasonable and the results can be justified statistically and the number of iterations is taken big enough so that the results are satisfying. Further in this chapter there is discussed how to determine the right number.

In figure 5.2 it can be seen that the cost of the optimization with a 'if' barrier function is lower then the cost of the optimization with a quadratic barrier function. So there can be concluded that the 'if' barrier function gives better results then the quadratic barrier function.

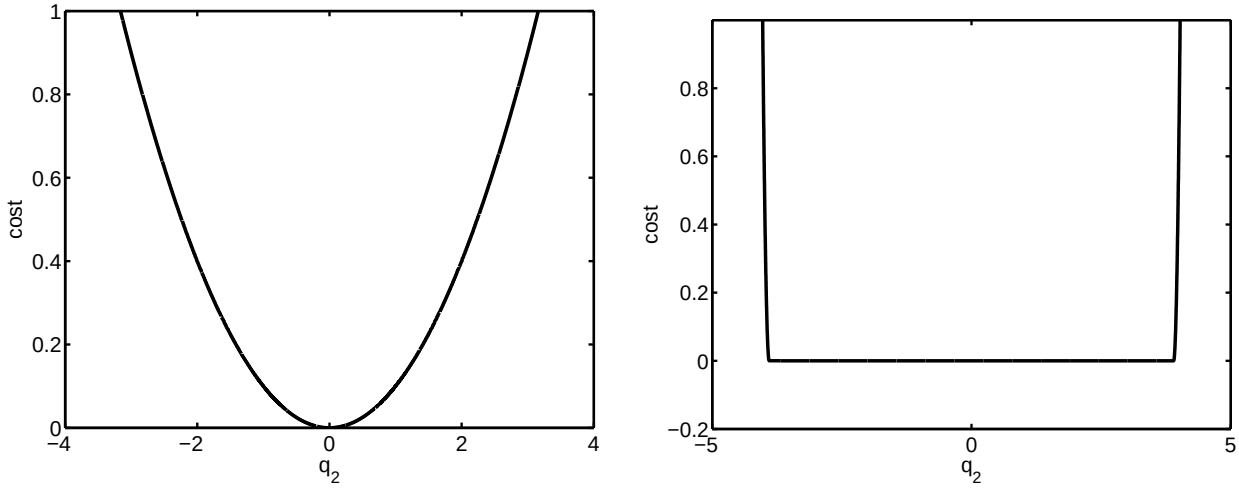


Figure 5.1: Two differen types of barrier functions. (a) is a quadratic function and (b) is a combination of a quadratic function and an 'if' statement.

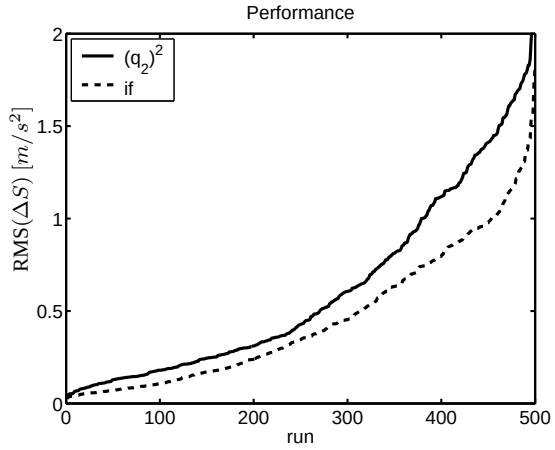


Figure 5.2: Two experiments with different barrier functions. One run consists of an optimization with 40 iterations. The result (y-axis) of one run is the root mean square (RMS) of (ΔS). In total 500 runs have been done which are sorted with the lowest cost first. The solid line is the result of the optimization with a quadratic feasibility function, this experiment is the same as in figure 4.1. The other line gives the result for an optimization with a feasibility function which is given in equation 5.2.

5.2 Optimizing initial position and speed of Desdemona

Every profile has an optimal for the initial position and speed. This can be seen in the example (figure 4.2) where the initial position and speed aren't optimal which results in a relatively large difference (ΔS) in the beginning of the path.

The initial position and speed can be optimized by including it in the optimization problem. Therefore the numerical optimization problem from section 4.2 is redefined. The original the cost function J *only* depends on $\ddot{q}(\cdot)$. The new cost function includes $q(t_0), \dot{q}(t_0)$, the new optimization problem is thus: $J(q(t_0), \dot{q}(t_0), \ddot{q}(\cdot))$.

To evaluate the new optimization problem, the original and new optimization problem (section 4.7) are solved again. The original optimization problem was solved with 40 iterations (number of iteration steps which are done in one run, see section 4.4.1), the new problem is solved for two cases: 40 and 50 iterations.

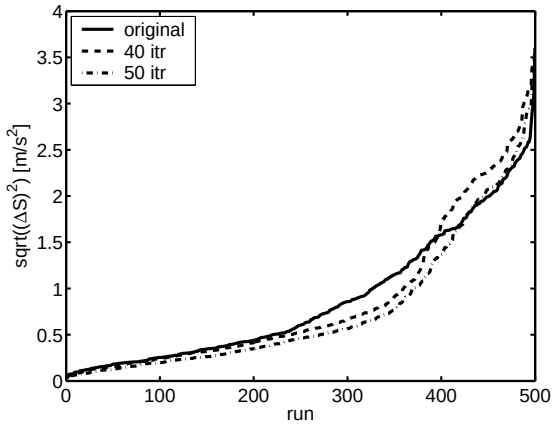


Figure 5.3: Three experiments are plotted, the original problem and the optimization which include $(q(t_0), \dot{q}(t_0))$. The optimization is done with 40 iterations (40 itr) and with 50 iterations (50 itr). The results (y-axis) of one run is the root mean square (RMS) of (ΔS).

The results can be found in figure 5.3. It can be seen that the solutions which optimize the initial $q(0), \dot{q}(0)$ give better results than the original solution. If the cost is larger then 1.5 m/s^2 then the new optimization problem need 50 iterations for getting the same performance. It can be understood intuitively that the new optimization problem needs more iterations for some cases, because the larger the number of variables will make the optimization problem harder. This is also shown in the next section.

5.3 Optimizing with different step size

The numerical optimization problem works with a sampled version of $S_d(t)$ and $q(t)$. The numerical integration method also works with a certain sample frequency for the continuous time ODE. These three sample frequencies can be chosen independently of each other. This is because the sample frequency of the iteration method (fs_{itr}) should be taken carefully to ensure numerical stability. The sample frequency for $S_d(t)$ is fs_{S_d} and for $q(t)$ is fs_q. They should be at least twice as big as bandwidth of respectively $S_d(t)$ and $q(t)$ because of the so called *Nyquist criteria* [5]. Another reason for choosing the step size of $q(k)$ carefully, is that the larger the step size (the lower fs_q) the smaller the number of parameters which have to be optimized. This results in easier problems.

The optimization problem is rewritten so that it is possible to optimize the problem with three different sample times. For the conversion between two sample times some values have to be re-sampled in a way such that there is no information lost. Therefore the new samples should be accurately interpolated. The conversion to a smaller step size is done by adding samples with value 0 and applying an anti-aliasing (lowpass) filter to the re-sampled value (with compensation for the filter's delay). For the conversion to a larger step size first a anti-aliasing filter is applied and after that some sample are deleted. This technique is described in [5]. The new cost function is given schematically in section D.2.

$h_u[s]$	computing time [s]
0.1	101.6
0.2	52.1
0.5	22.3
1.0	12.5

Table 5.1: The average computing time (simulated with a AMD Athlon 1800) that is needed for one optimization run is given categorized per sample time.

To get an idea of the influence of different sampling frequencies two experiments are done. The first experiment

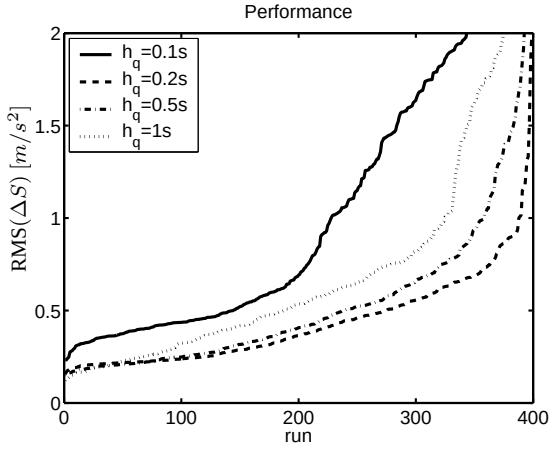


Figure 5.4: Four experiments optimizing with different step sizes, in the legends the sample time is given. The results (y-axis) of one run is the root mean square (RMS) of (ΔS).

is the same as the original experiment (section 4.7) but now with different step size and number of iterations (see for the definition of iteration section 4.4.1). The step size of $S_d(k)$ is taken as $h_{S_d} = 0.1s$ and the step size of the numerical integration method is changed to $h_{itr} = 0.05s$. The step-size of the numerical integration is not changed during the experiment. The step size of $\ddot{q}(k)$ is changed during the experiment between $h_{\ddot{q}} = 1s$ and $h_{\ddot{q}} = 0.1s$ (original optimization problem). The optimization is done with 20 iterations.

In figure 5.4 and table 5.1 the results are given. The original optimization problem with sample time $h_{S_d} = 0.1s$ has the worst performance. This can be seen from the high cost and computing time. This is due to the number of variables which have to be optimized. The fewer variables (the larger the step size) that have to be optimized, the easier (faster) it is to optimize. The is that the sample frequency of q can not be taken to low because then it is not possible to include the high frequency accelerations. This will be shown in the next experiment.

From the previous paragraph it is clear that for a last optimization the lowest possible sample frequency of $q(t)$ should be determined prior to the optimization. When the path of Desdemona is a linear function of the states of the joints $S(q(t), \dot{q}(t), \ddot{q}(t), t)$, then the sample frequencies of S and q can be taken the same. But because Desdemona is nonlinear this is not the case[1]. The only relation is that the higher the frequencies in $S_d(t)$ the higher the sample frequency has to be for $q(t)$.

In the following experiment it will be shown that there is a relationship between the step-size of $q(k)$ and the frequency in $S_d(k)$. The following step sizes are used: $h_{S_d} = 0.1s$, $h_{itr} = 0.05s$ and $h_{\ddot{q}} = 1s$. The frequency of the accelerations in $S_d(k)$ is changed from 2 to 5 rad/s, the results are shown in figure 5.5. If the frequency of S_d is above 3 rad/s the cost is increasing enormously. This is due to a too low sample frequency for $q(k)$. So it can be concluded that the frequency of $S_d(k)$ has an influence on the needed fs_q.

5.4 Different stopping criterions

In previous experiments the number of iterations in a optimization run is fixed. This is not a logic stop criteria because if the optimization method finds the (local) optimum in fewer steps, the additional steps do not doesn't benefit the performance. Therefore the optimization should be stopped if the optimum is reached. If the optimization has reached its optimum then the change in the cost function will remain small in the next iteration steps. So, a good extension of the stop criteria is adding a minimum for the change of the cost function. If the cost function is changing less then the minimum, the optimization will be stopped.

This extended stop criterium has been tested in the following experiment. The minimum change for the stop criteria is taken 0.1. The results are given in figure 5.6. The performance is almost the same hence the extra stop

[1]It is possible for some nonlinear filter cases to calculate the output frequency for a given input frequency. But for Desdemona this is not possible, due to used sinuses and limiters in the transfer function

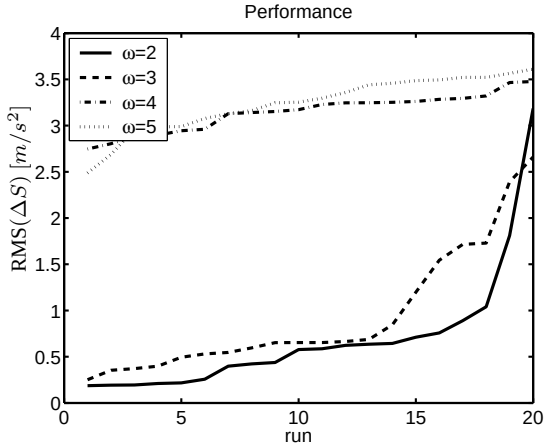


Figure 5.5: Four times the experiment of section 4.7 is done with four different $S_d(k)$. The $S_d(k)$ in equation 3.9 is defined as $\ddot{y} = 5\cos(\omega k)$ with ω changing between 2 and 5 rad/s. The results (y-axis) of one run is the root mean square (RMS) of (ΔS).

criterium is not influencing the results. But there is a 30% saving of computing time, hence the extended stop criterium is useful.

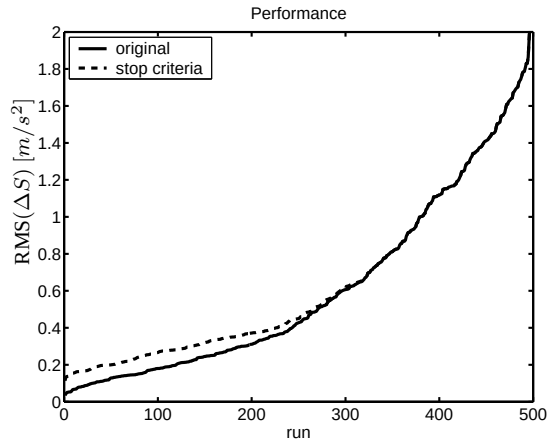


Figure 5.6: Different stopping criterions, the results of the original experiment (section 4.7) with 40 iterations and the same experiment with extra stop criterium. The extra stop criterium will stop the optimization if the changing of cost function(inclusive feasibility function) is below 0.1. The results (y-axis) of one run is the root mean square (RMS) of (ΔS).

5.5 Evaluating different profiles

The final step in the two degrees optimization problem is to combine the new techniques described in this chapter. Another step is to use the real parameters (see section A.5) for dimensioning Desdemona and the motors. This optimization problem is tested with some realistic paths.

To get an idea of the performance of Desdemona the two degree Dynamic model assumes that the degrees of freedom which are not used have become rigid connections. It means that the mass and inertia (see section A.5) of the whole Desdemona are taken into account.

In this optimization problem all the constraints (see section 3.4) will be taken into account. For the feasibility function the barrier function in the form of equation 5.2 is chosen. The parameters which determine the bounds of the constraints can be found in section A.5.

5.5.1 Cosine path

To be able to compare the new realistic system with previous experiments the original cosine path will be used for the first experiment. The initial position is also optimized, and the stop criterium is taking the change of the cost function in account. The results for the best run is given in figure 5.7. The result is good because the constraints are met and S is very close to S_d

5.5.2 Catapult start

This profile describes the start of an airplane from a aircraft carrier. It is a very strong acceleration in one (e_y) direction. The optimization also optimized the initial position as well. The results for the best run is given in figure 5.8. As can be seen there is a difference between S and S_d although the flanks of S are steep.

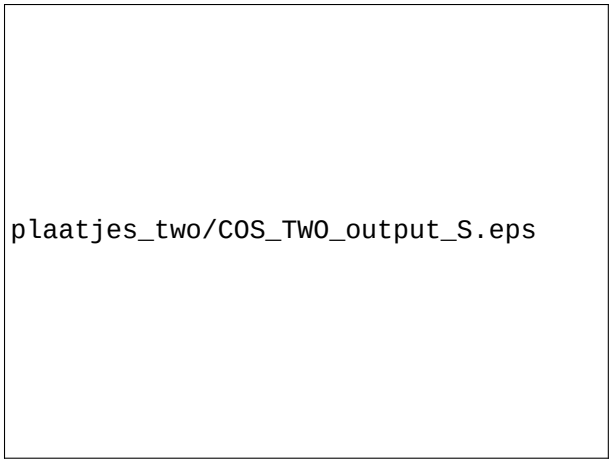
5.5.3 Triple pulse

The initial position is also optimized, and the stop criterium is taking the change of the cost function in account. The results for the best run (figure 5.9) are the same as the results of the catapult start, there is a difference between S and S_d but the flanks are steep.

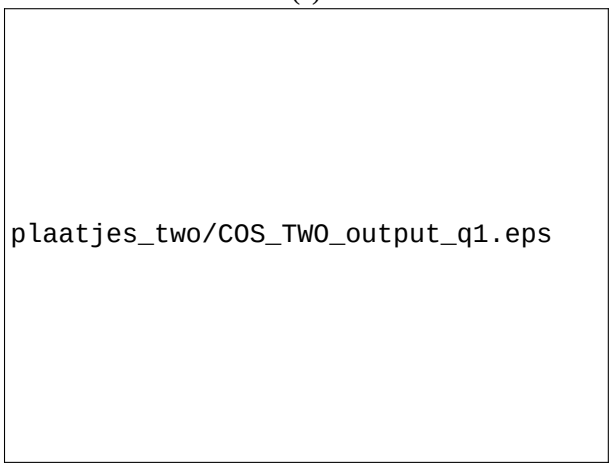
5.5.4 Fast care

In this experiment the data from a real car is used. The accelerations are measured during 35 second while the car is doing the track from figure 5.10.

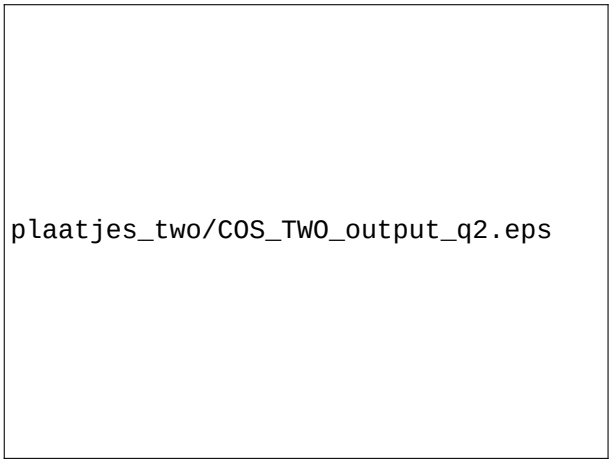
The best run is given in figure 5.11. As can be seen the results are good.



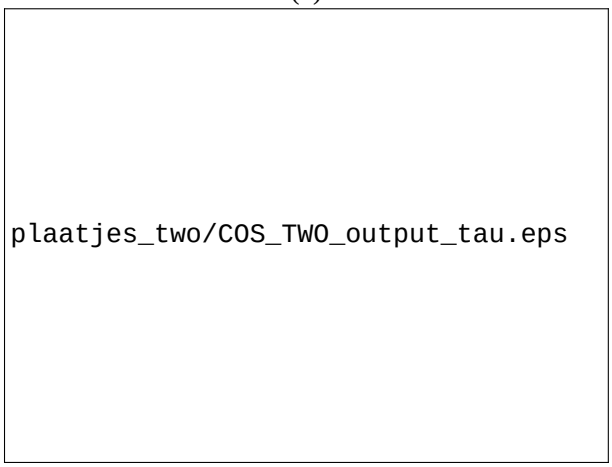
(a)



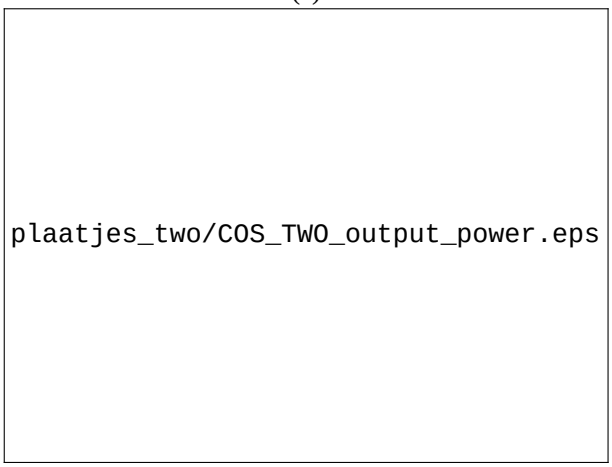
(b)



(c)

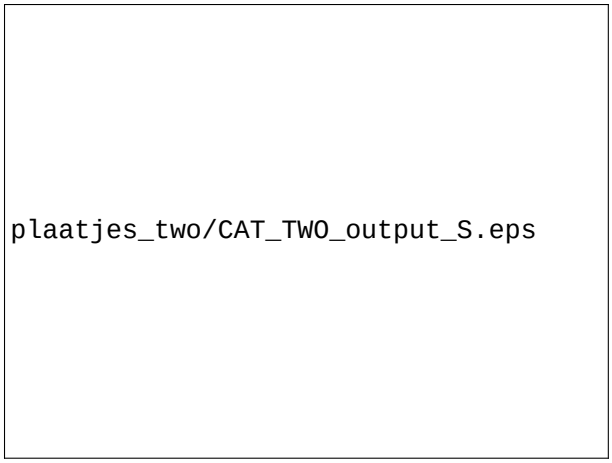


(d)

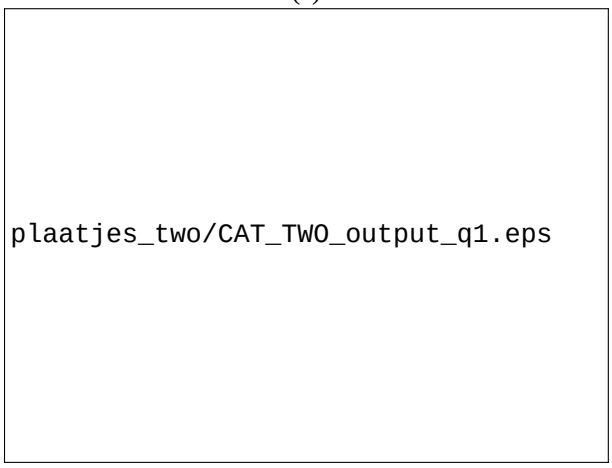


(e)

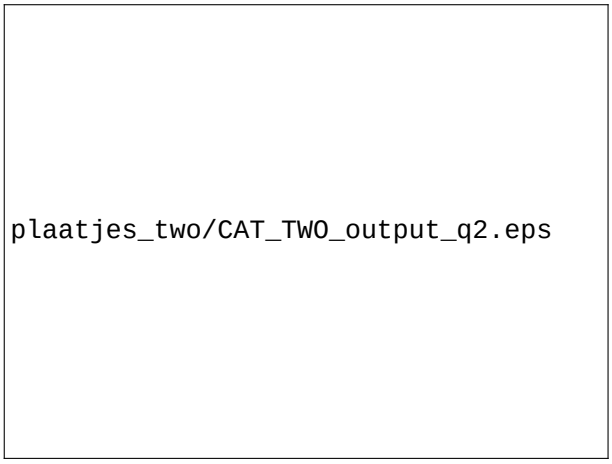
Figure 5.7: The best results for the cosines experiment. In (a) the S (doted line) and S_d (Solid line) are plotted. (b) and (c) are the positions of the joints q_1 and q_2 respectively are plotted. In (d) and (e) the normalized force and power respectively are plotted. They are normalized by dividing the force and power by its maximum allowed value. If the normalized force or power is larger than one, then one of of the motors will be damaged. The initial velocity is $\dot{q} = [109.5\, degree/s\ 0.18\, m/s]$



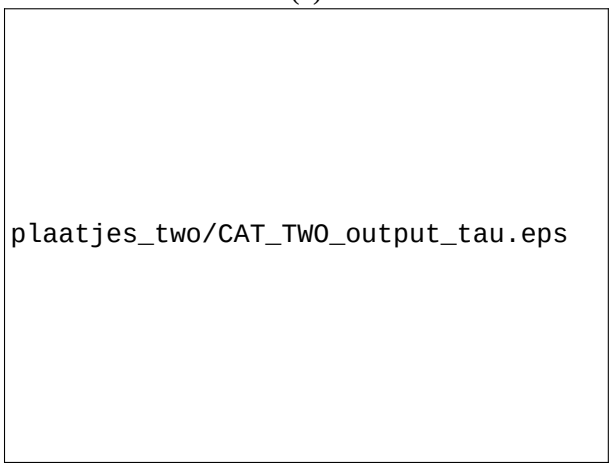
(a)



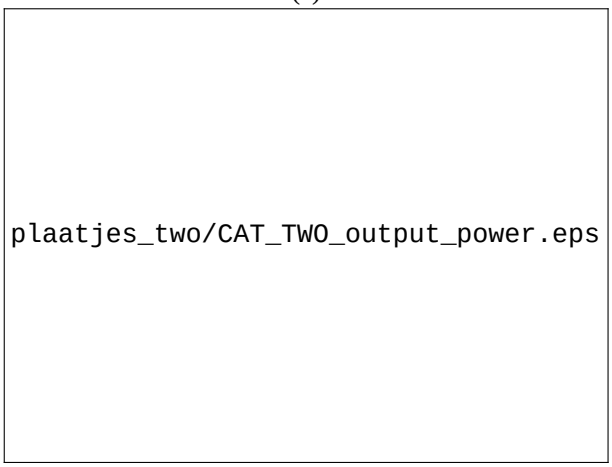
(b)



(c)

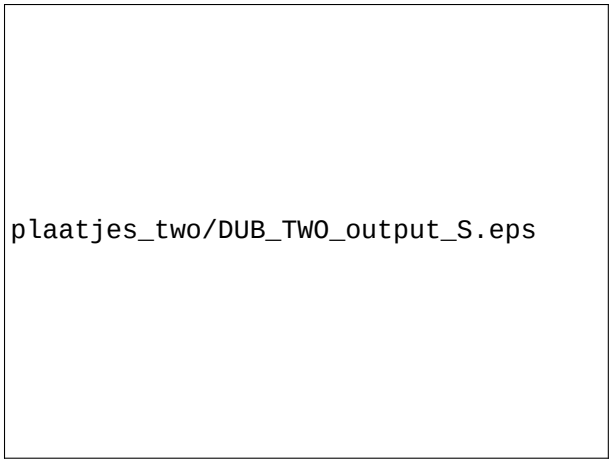


(d)

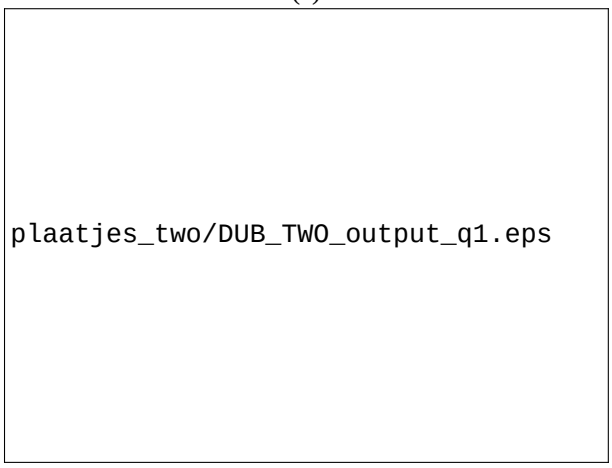


(e)

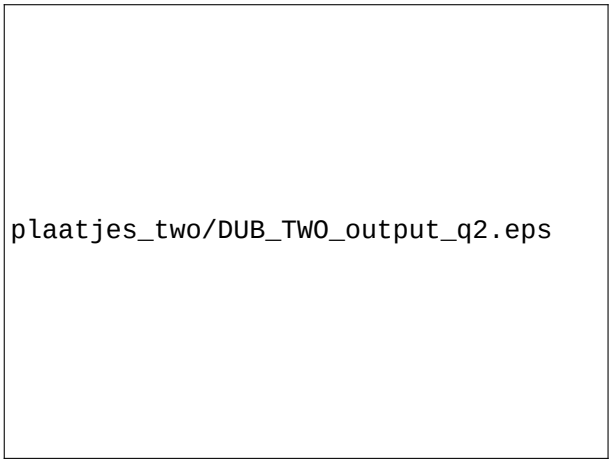
Figure 5.8: The best results for the catapult start. In (a) the S (doted line) and S_d (Solid line) are plotted. (b) and (c) are the positions of the joints are plotted. In (d) and (e) the normalized force and power respectively are plotted. It is normalized by dividing the force and power respectively by its maximum allowed value. If the normalized force or power is larger, then one, than one of the motors will be damaged. The initial velocity is $\dot{q} = [-82.6\, degree/s \;\; -0.04\, m/s]$



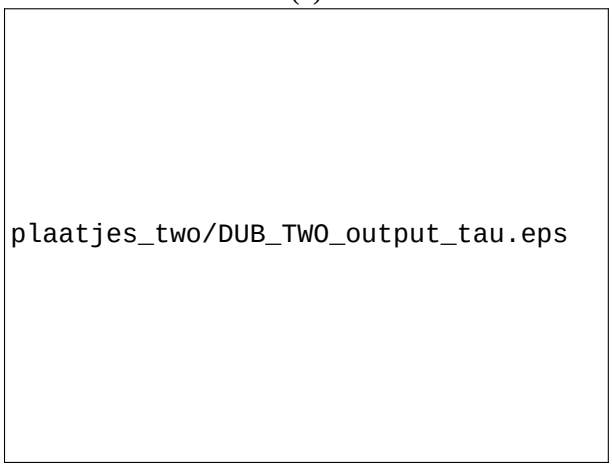
(a)



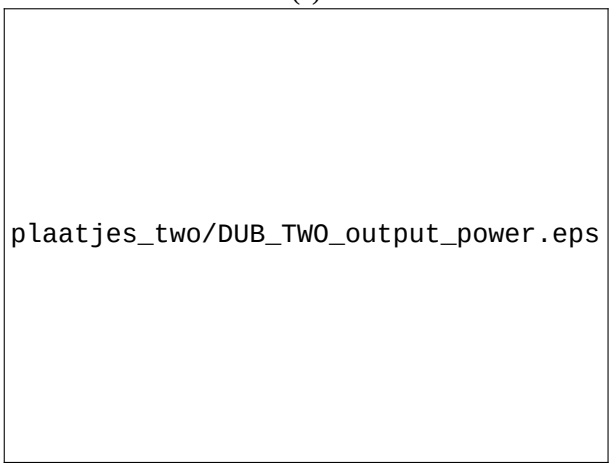
(b)



(c)



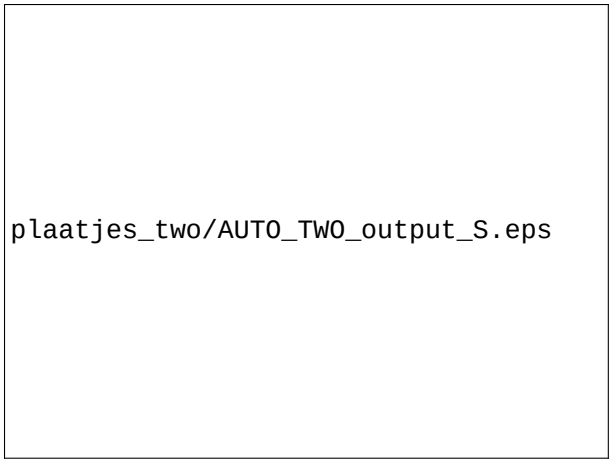
(d)



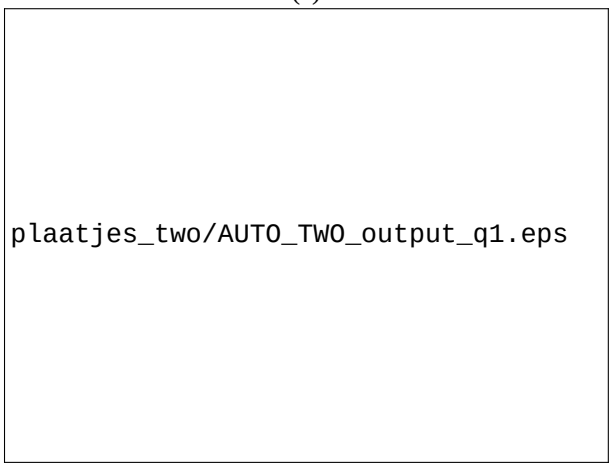
(e)

Figure 5.9: The best results for the double pulse experiment. In (a) the S (doted line) and S_d (Solid line) are plotted. (b) and (c) the positions of the joints are plotted. In (d) and (e) the normalized force and power respectively are plotted. It is normalized by dividing the force and power respectively by its maximum allowed value. If the normalized force or power is larger then one, than the motors will be damaged. The initial velocities are is $\dot{q} = [2.82\,rad/s \;\; -0.11\,m/s]$

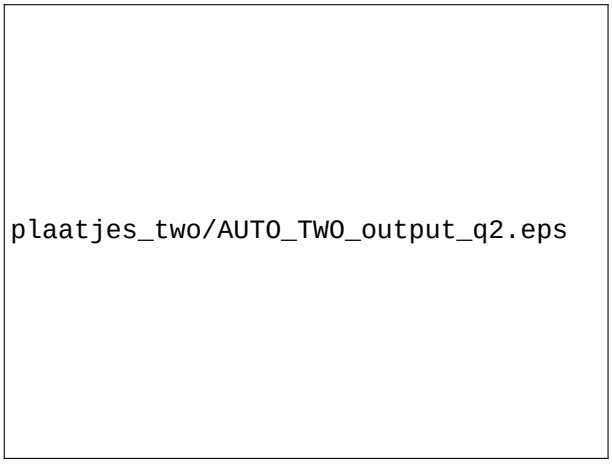
Figure 5.10: The track of the fast care, done in 35 seconds



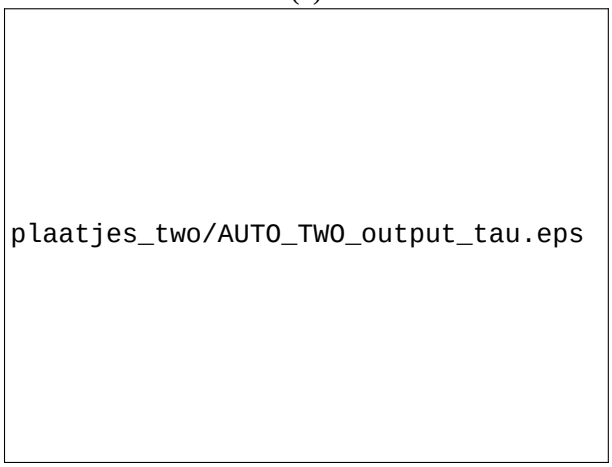
(a)



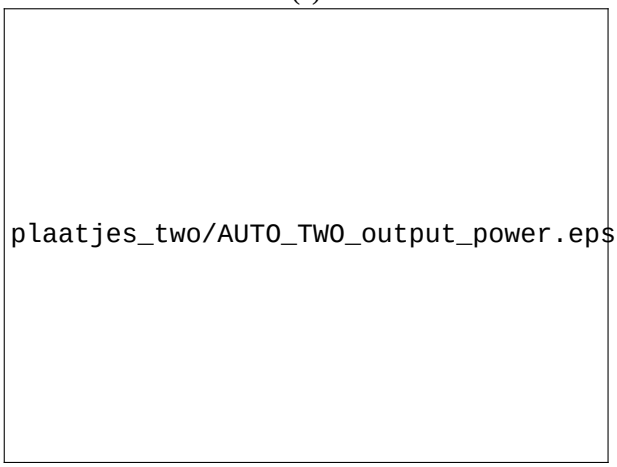
(b)



(c)



(d)



(e)

Figure 5.11: The best results for the fast care experiment. In (a) the S (doted line) and S_d (Solid line) are plotted. (b) and (c) the positions of the joints are plotted. In (d) and (e) the normalized force and power respectively are plotted. It is normalized by dividing the the force and power respectively by its maximum allowed value. If the normalized force or power is larger then one, than the motors will be damaged. The initial velocities are is $\dot{q} = [-9.4\, degree/s \quad -0.4 m/s]$

5.6 Desdemona with three degrees of freedom

A logical step to make the optimization problem more realistic is to increase the number of d.o.f. Therefore q_5 the yaw of the cabin is included. With this extra d.o.f. it is possible to place the cabin in every position in the e_x e_y plane of the fixed world and to simulate any desired acceleration profile (that is within bounds). In this section the same experiments are done as in section 5.5. All the experiments optimize also the initial position, and the stop criterium is taking the change of the cost function in account. In the path S_d the linear translation in e_x e_y direction are defined, the rotation is not taken into account. The rotation is not taken into account because for example the catapult start path demands for almost 2 seconds an acceleration of $20m/s^2$ which result in a linear 40 meter displacement or a combination of a small displacement and rotation. Desdemona isn't 40 meter long, so we expect the optimal controller to force Desdemona to start rotating. Q is defined so that the rotation isn't included in the optimization.

The results of the experiments are discuss in section 5.8.

5.6.1 Cosine path

To be able to compare the new realistic system with previous experiments the original cosines path will be used for the first experiment. The results for the best run is given in figure 5.12. The result is good because the constraints are met and S is very close to S_d.

5.6.2 Catapult Start

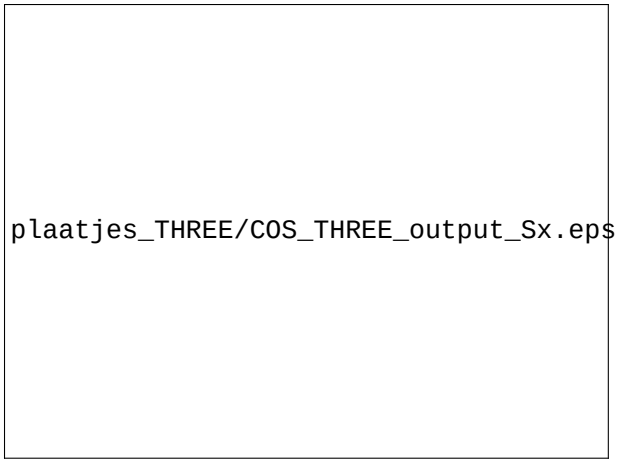
This profile is from the start of an airplane from a aircraft carrier. It is very strong acceleration in one (e_y) direction. The optimization also optimized the initial position. The weight matrix Q (see section 3.4) is taken so that an error in e_x direction result in a three times larger cost than in e_y direction. The results for the best run is given in figure 5.13. It can be seen that the results aren't so good as the two degree problem, this is to be expected because it is a much harder problem. The false cue (S_x) is smaller then the wanted acceleration (S_y) which is one of the basic requirements to make the simulation realistic.

5.6.3 Double pulse

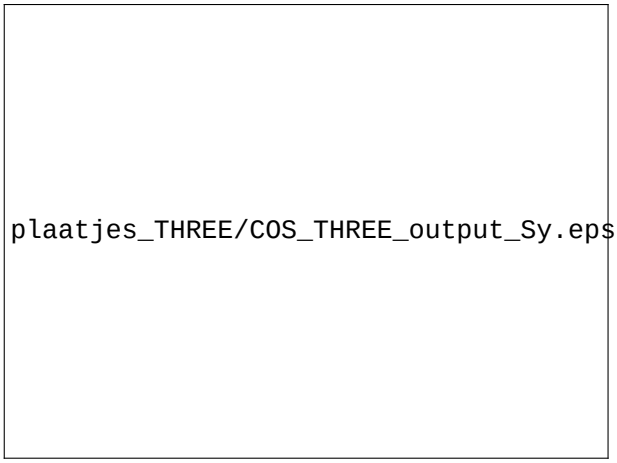
This is just a test profile with twice an step acceleration in the e_y direction. The results for the best run is given in figure 5.14. The weight matrix Q (see section 3.4) is taken so that an error in e_x direction result in a three times larger cost then in e_y direction.

5.6.4 Fast car

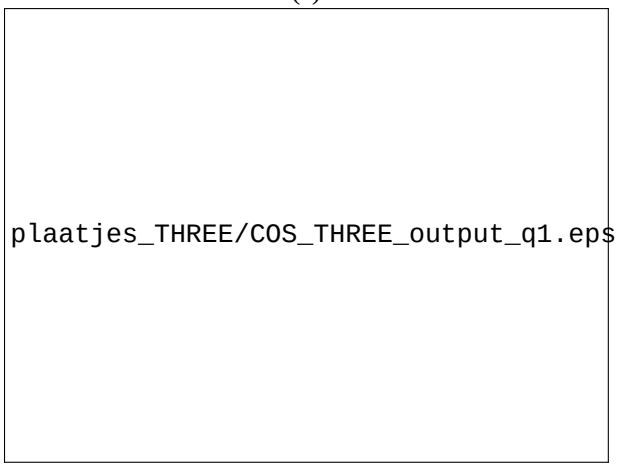
In this experiment the data from a real care is used (the car is doing the track from figure 5.10). The results for the best run is given in figure 5.15. It can be seen there is a large difference between S and S_d. But on the other hand S_d is high frequent, and hence difficult to follow.



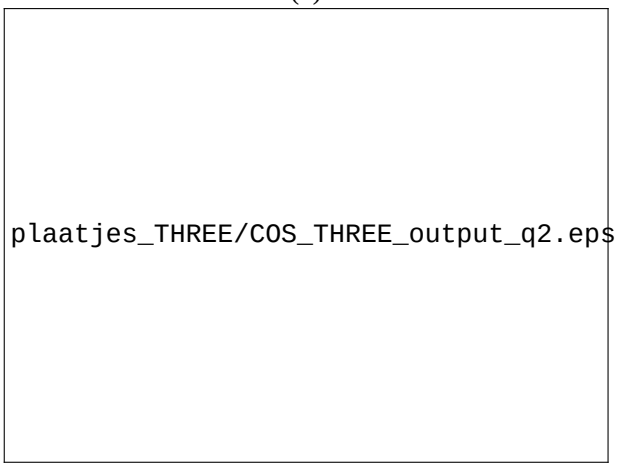
(a)



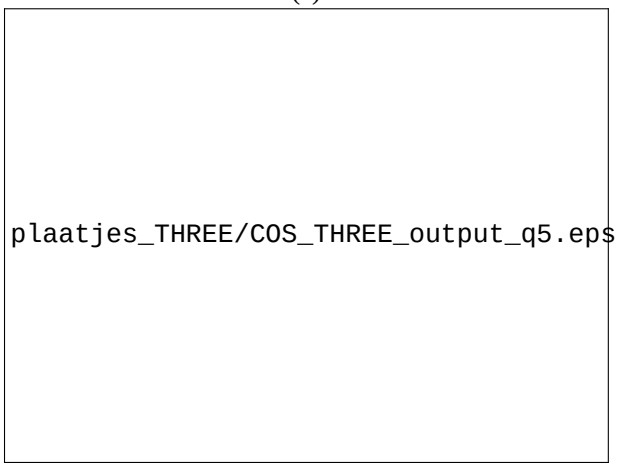
(b)



(c)

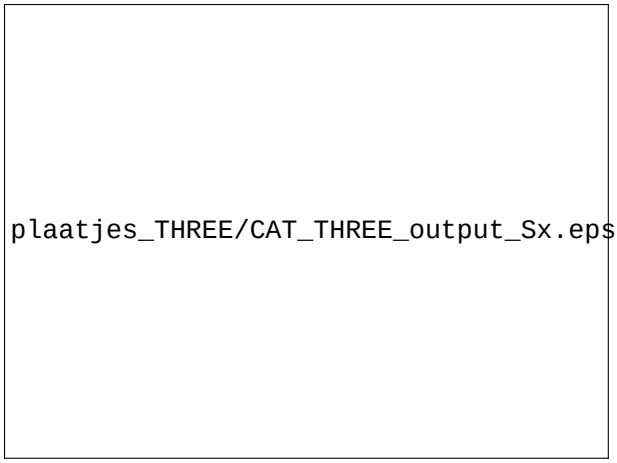


(d)

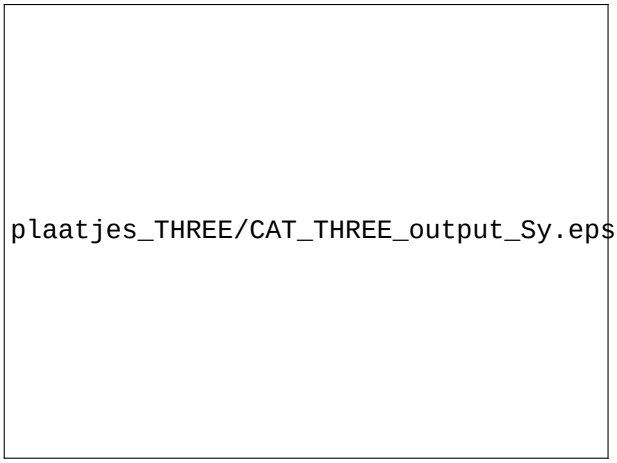


(e)

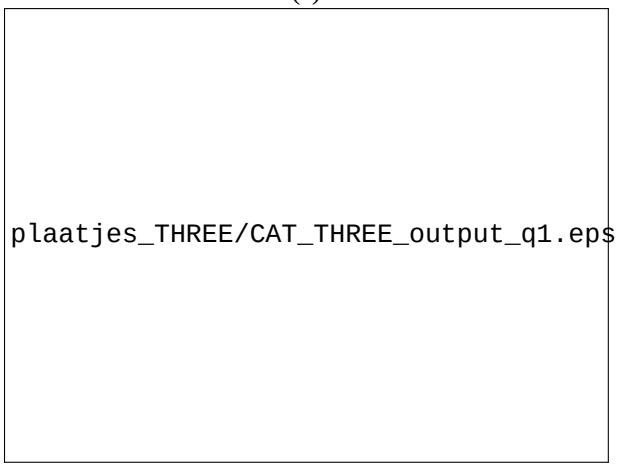
Figure 5.12: The best results for the three d.o.f. cosines experiment. In (a) and (b) the S_x and respectively S_y are plotted. In (c), (d) and (e) the positions of the joints q_1, q_2 respectively q_5 are plotted. The constraints are not plotted but they are all met. The initial velocities are $[\dot{q_1}\ \dot{q_2}\ \dot{q_5}] = [37.3\, degree/s\ \ -1.0\, m/s\ 0.5\, degree/s]$



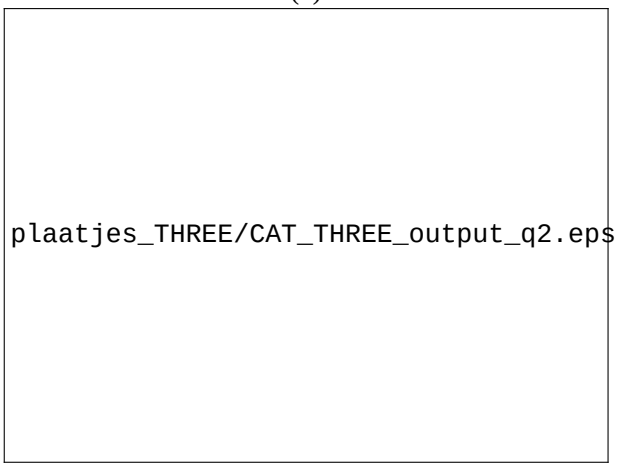
(a)



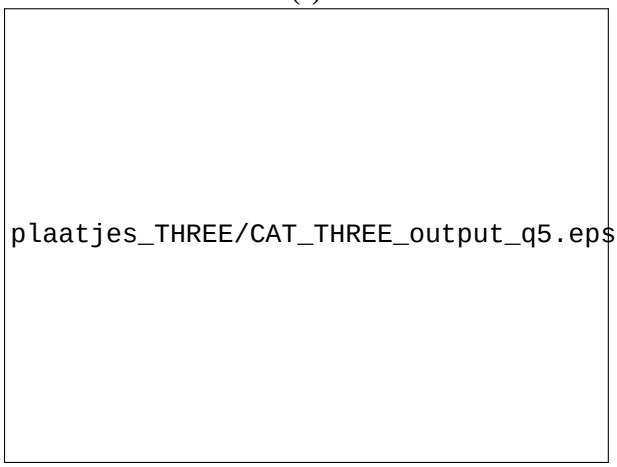
(b)



(c)

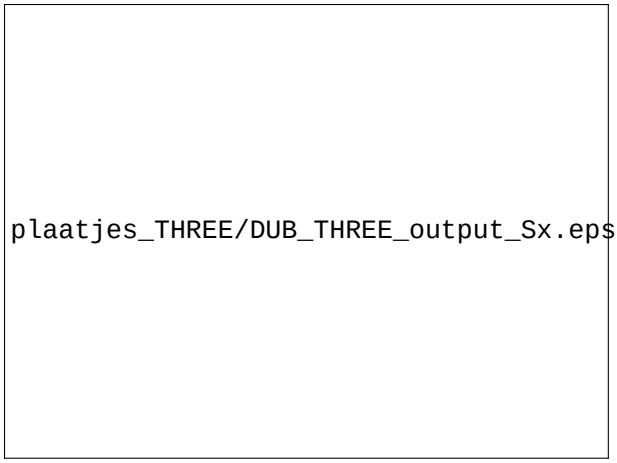


(d)

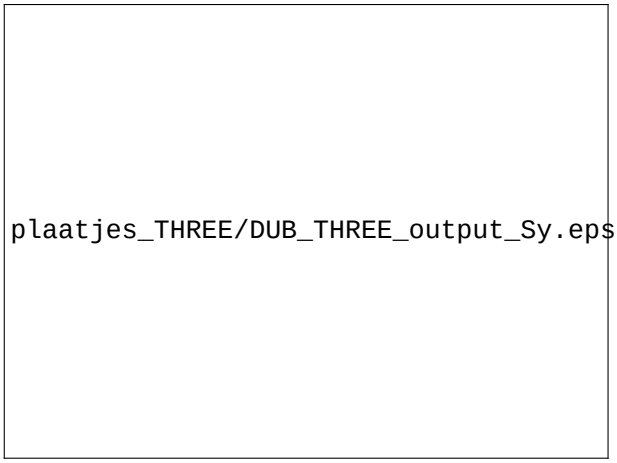


(e)

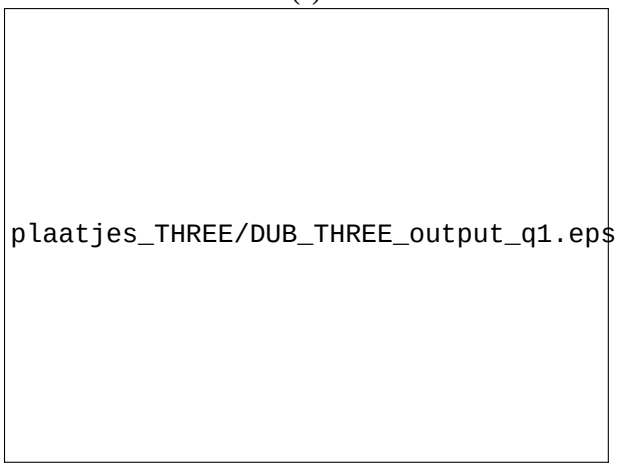
Figure 5.13: The best results for the three degree catapult start experiment. In (a) and (b) the S_x and S_y respectively are plotted. In (c), (d) and (e) the positions of the joints q_1, q_2 and q_5 respectively are plotted. The constraints are not plotted but they are all met. The initial velocities are $[\dot{q_1}\ \dot{q_2}\ \dot{q_5}] = [-74.9\,degree/s\ \ -1.5\,m/s\ 10.0\,degree/s]$



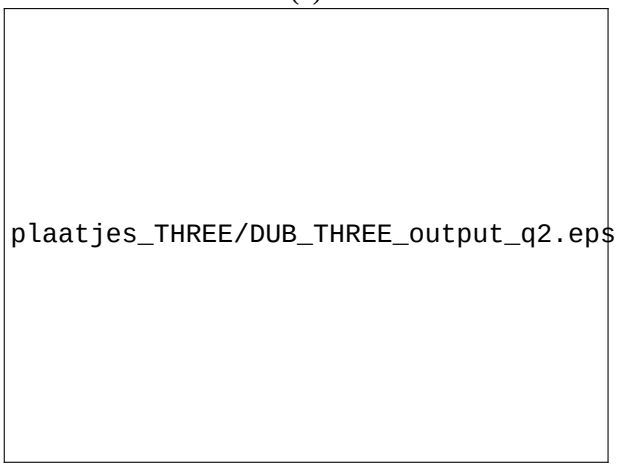
(a)



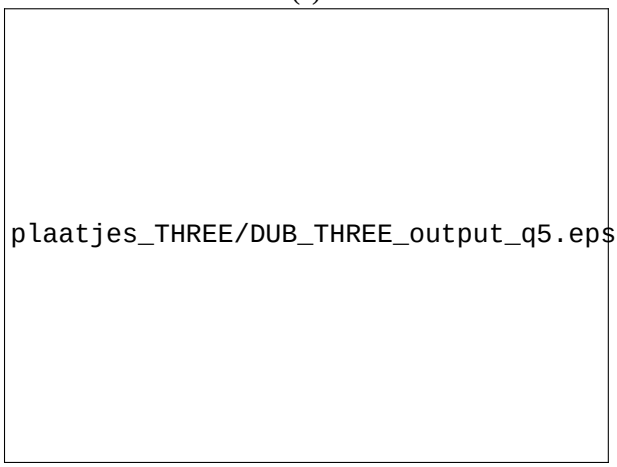
(b)



(c)

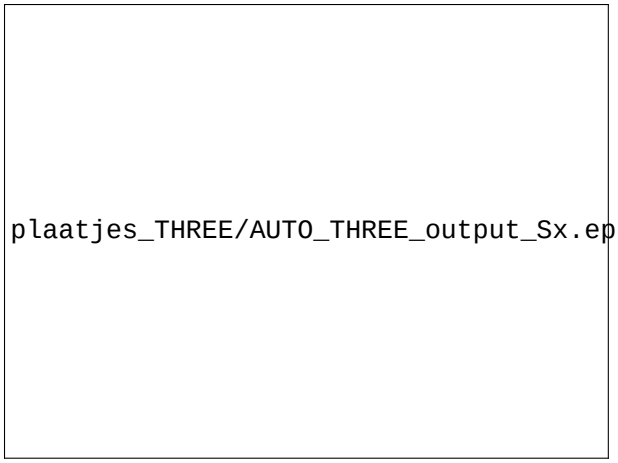


(d)

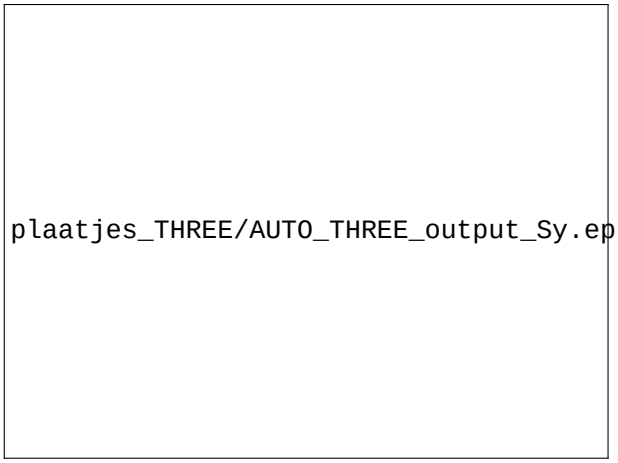


(e)

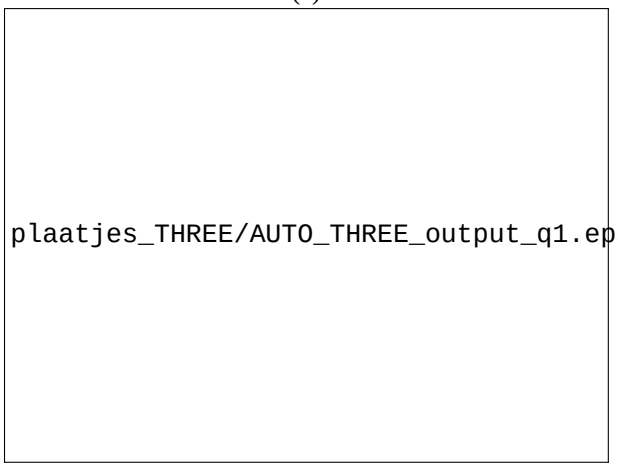
Figure 5.14: The best results for the three degree catapult start experiment. In (a) and (b) the S_x and S_y respectively are plotted. In (c), (d) and (e) the positions of the joints q_1, q_2 and q_5 respectively are plotted. The constraints are not plotted but they are all met. The initial velocities are $[\dot{q_1}\ \dot{q_2}\ \dot{q_5}] = [-52.9\,degree/s\ \ -1.5\,m/s\ \ -39.2\,degree/s]$



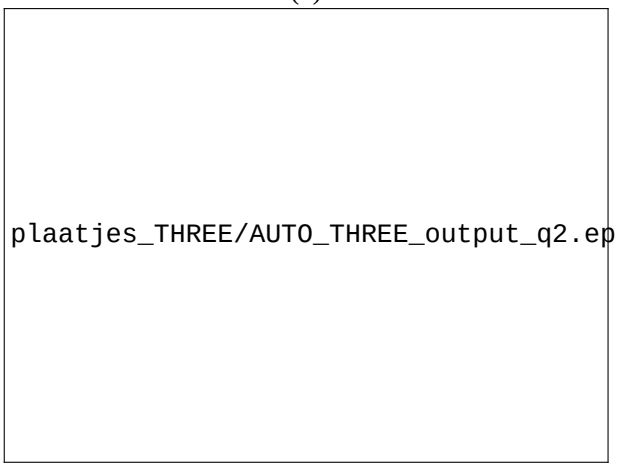
(a)



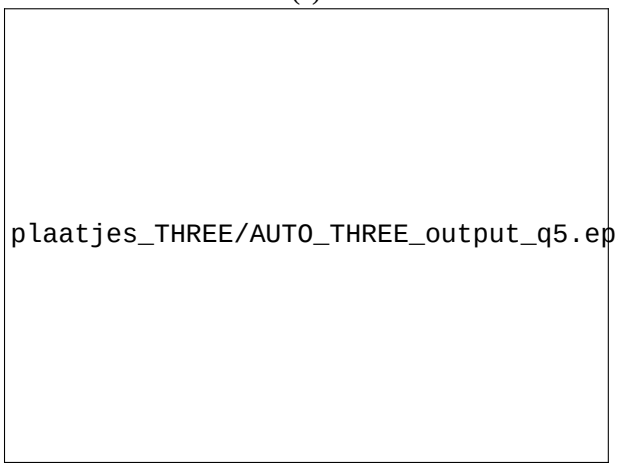
(b)



(c)



(d)



(e)

Figure 5.15: The best results for the fast care experiment. In (a) and (b) the S_x and S_y respectively are plotted. In (c), (d) and (e) the positions of the joints q_1, q_2 and q_5 respectively are plotted. The constraints are not plotted but they are all met. The initial velocities are $[\dot{q_1}\ \dot{q_2}\ \dot{q_5}] = [7.8\,degree/s\ -1.7\,m/s\ -6.5\,degree/s]$

5.7 Improving the performance

In previous section Desdemona simulated with three d.o.f. The results can be increased by simplifying the dynamic model of Desdemona and by taking initial values not randomly but by selecting promising values. This is explained in this section.

Simplifying the dynamic model

The parameters of the dynamic are given in appendix A.5. The center of mass is not always in the origin of Desdemona coordinates. This makes the dynamic model more complex because it increases the number of joints that are influencing each other torques. By assuming the the center of the mass is always in the origin, some small errors are introduced in the optimization. The result is that the calculation time is shorter. The fact that some small errors are introduced can be eliminated by an extra optimization with the full dynamic model at the end. Thus this method is increasing the performance.

Smart Guess

All the experiments are until now done with random initial values (random $q(0), \dot{q}(0), \ddot{q}(\cdot)$). But due to other research[2] about Desdemona there are ideas of good performing inial values. Some experiments are done with these inial values. The idea is that a initial values that are chosen with prior knowledge give better results.

For example, if we expect an S_d in the e_y direction, where at the begin no accelerations and later some high acceleration occur, then three groups of sensible initial parameters can be chosen, see table 5.2.

q_2	$\dot{q}_1$	$\dot{q}_2$
0	$\uparrow$	-
$\uparrow$	0	0
small	$\uparrow$	$\downarrow$

Table 5.2: The three possible initial parameters. The $\uparrow$ means a large value. The acceleration will be low for a limited time and thereafter high. This can be found from equation B.5

5.7.1 Catapult start

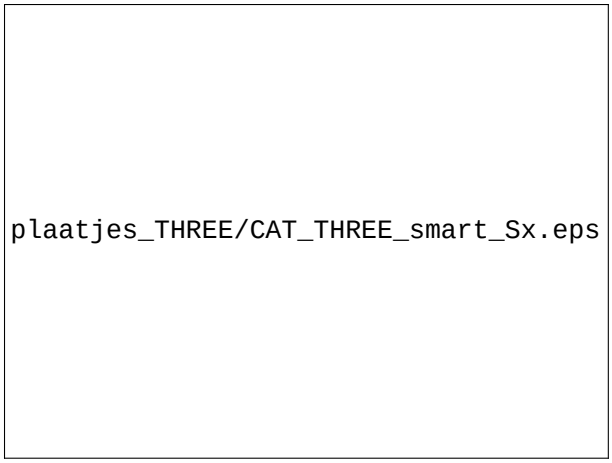
The catapult start is repeated but now with a simplified dynamics and smartly guessed initials values. $q(0), \dot{q}(0)$:

$$[q_1\ q_2\ q_5] = [0\,degree\ -0.01\,m\ 0\,degree]$$

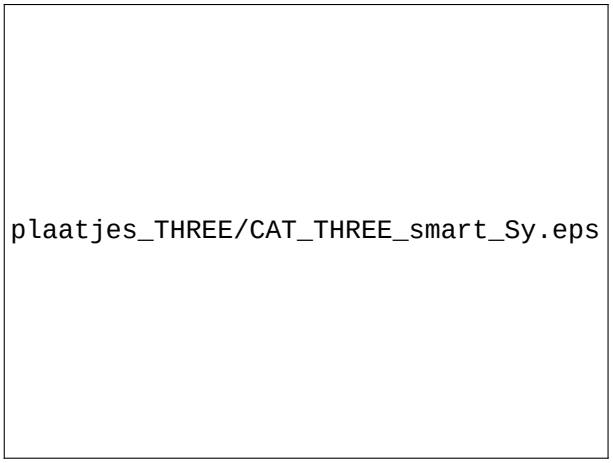
$$[\dot{q_1}\ \dot{q_2}\ \dot{q_5}] = [86\,degree/s\ -1.4\,m/s\ 0\,degree/s]$$

The results are given in figure 5.16. The optimization with the simplified dynamics is done first and thereafter an optimization with the real dynamic is done to ensure the results are applicable to the Desdemona.

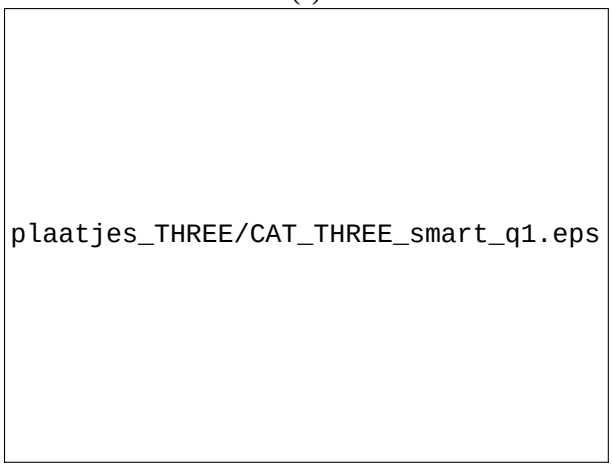
[2]Mark, heb je nog referencies??todo



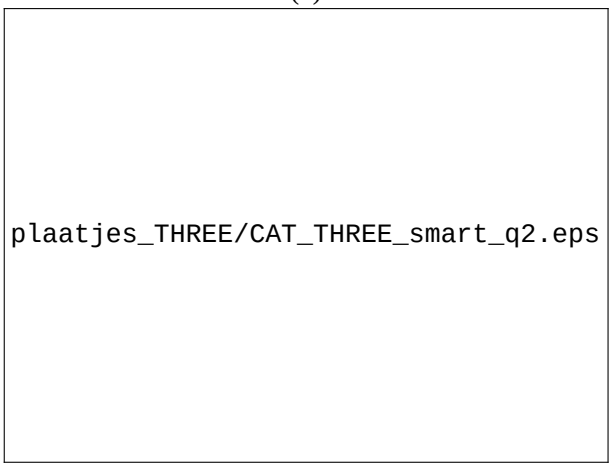
(a)



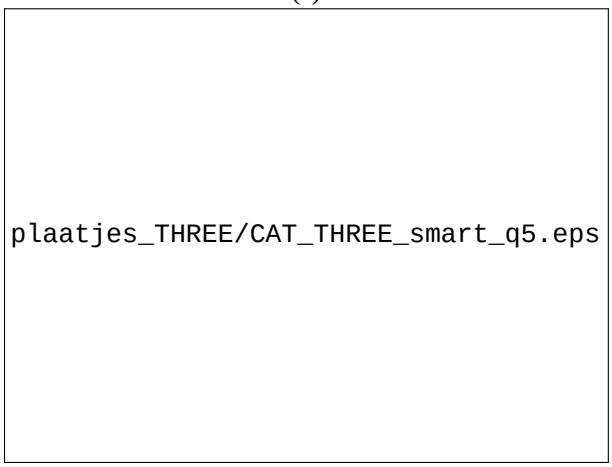
(b)



(c)



(d)



(e)

Figure 5.16: The best results for the smart catapult experiment. In (a) and (b) the S_x and S_y respectively are plotted. In (c), (d) and (e) the positions of the joints q_1, q_2 andq_5 respectively are plotted. The constraints are not plotted but they are all met. The initial velocities are $[\dot{q_1}\ \dot{q_2}\ \dot{q_5}] = [95.2\,degree/s\ \ -0.4\,m/s\ \ -21.9\,degree/s]$

5.8 Discussion three d.o.f. results

The results are good for the cosines experiment (figure 5.12). Thus the method is working. For the other experiments (figure 5.13, 5.14 and 5.15) the result are not as good. This is because the cost function is non-linear and non-convex, which result in a large number of local minima. In the two degree optimizations cases the problem is less nonlinear and therefore there are fewer local optimums.

For finding a low local minimum you need a huge number of trials. This number can be reduced by using promising initial values and by simplifying the dynamic model. But then you need a way to find these values. An example can can be found in section 5.7.

The cosines experiment has the best result, this is because the lower the existing frequencies in $S_d(t)$ are, the easier to solve the optimization problem. One reason is you can resample the problem, and solve easier (see section 5.3). Or in other words, a change at a certain time of the joints of Desdemona will not only influence the error (ΔS) at the same time interval but also in later intervals. This is particular the case if $S_d(t)$ contains only lower frequencies since there is more correlation between two sample next to each other.

Another difficulty is that the optimization has to deal with a huge number of input variables, for the fast car experiment (figure 5.15) 450 variables, which cause a huge complexity for the problem (see section 4.5).

Thus the results could be expected because of the known limitations and complexity (see section 4.5). Some options give better results by choosing better initial values, and to avoid high frequencies in $S_d(t)$

If the optimization find a local minima, a method that possibly can be used to find other local minima in the neighborhood can be applied. The method redefines the $S_d(\cdot)$ as

$$S_{dnew}(\cdot) = \lambda S_d(\cdot) + (1 - \lambda)S(\cdot)$$

with $\lambda \in < 0 \ldots 1 >$ and $S(t)$ is the path from the previous found optimum. Then with a new optimization run the $S_{dnew}(t)$ can be used to find other local minima. This method has been implemented with several λ and tested with the three d.o.f. catapult start experiment (section 5.6.2). The results is that the average performance decreases. Thus there can be concluded that the method is in our case not useful for improving the optimization performance.

Chapter 6

Including human perception model

6.1 Introduction

As stated before, the main goal of Desdemona is to let the person in Desdemona observe the same movements as in the simulated vehicle. Sometimes a human observes different movements than the human body it is making due to measurement faults. Therefore it is not always necessary that Desdemona makes the same movements as the simulated vehicle. By taking this into account it is possible to filter out irrelevant movements (movements a human doesn't observe) and concentrate on the movements that a human does percept. To achieve this, a human perception model is included in the optimization problem, this is given in figure 6.1.

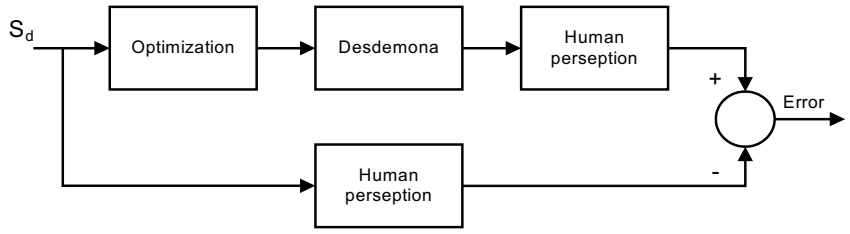


Figure 6.1: A schematic model of the optimization problem with human perception model

6.2 Human perception

There are three main sensory systems involved in spatial orientation and disorientation: the visual system, the vestibular system (inner ear), the somatosensory system (”seat of the pants”)[1]. To a smaller degree, the auditory system is also involved. Of these, vision is by far the most important sensory system providing spatial orientation during flight. In the absence of vision, orientation must be derived solely from the vestibular or somatosensory systems, and these systems do not always provide accurate motion and position cues.

The primary role of the vestibular system is to enhance vision. It provides angular and linear acceleration information to stabilize the eyes when motion of the head and body would otherwise result in blurred vision. It's secondary role, in the absence of vision, is to provide a sense of position and motion.

The nerve system and brain are interpreting and combining the information from the sensors and let a person observe the movements which the person is making. But sometimes a person observe another movements that it is making. This is due to measurement and interpretation faults. This and other effects from the human perception model are still a subject for ongoing research. An example is that you don't feel constant rotations, or another

example, if you sit in the train, you think it is leaving but instead it is nearby a train that is leaving

A fixed simulator involves virtual and auditory simulation and a moving simulator also involves the vestibular and somatosensory simulation. This work only deals with the moving part of Desdemona and in dialogue with TNO there has been decided to include the vestibular system in the optimization problem.

6.3 Vestibular system

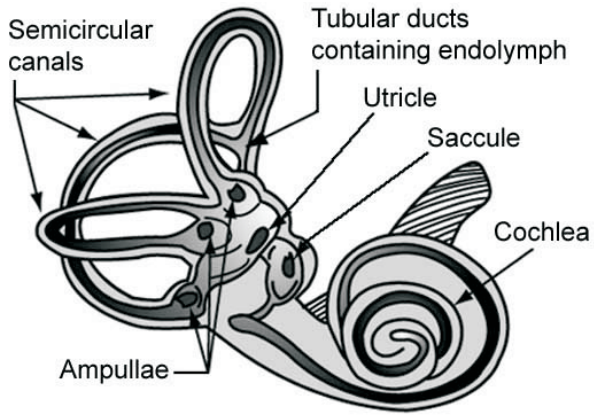


Figure 6.2: The Vestibular system (from [12])

The vestibular system can be divided in the semi-circular canals and otoliths see 6.2. With these two perception models it is possible to measure rotation and linear acceleration. These models will be included in the cost function of the optimization and the new optimization will minimize the error in perception.

6.3.1 The semi-circular canals

The semicircular canals are part of the vestibular system. There are three semicircular canals located in the inner ear on each side of the head. Fluid within the canals moves relative to the canal walls when angular accelerations are applied to the head. This fluid movement bends sensory hair filaments in specialized portions of the canals, which send nerve impulses to the brain resulting in the perception of rotary motion. Each canal is oriented such that it responds differently from the others to angular accelerations in the pitch, roll and yaw spatial planes (see figure 6.3).

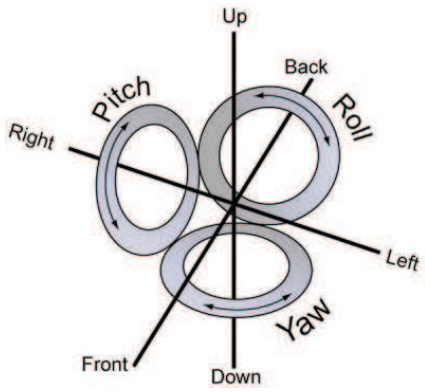


Figure 6.3: The pitch, roll and yaw canals (from [12])

The relative responses from all three canals are integrated by the brain to determine the actual plane in which the motion is occurring.

Dynamic model of the semi-circular canals

The model of the semi-circular canals are based on an overdamped-torsion-pendulum with transfer function

$$H_{scc}(\omega) = \frac{neural\ discharge\ rate}{angular\ acceleration} = \frac{K}{(1+\tau_1 j\omega)(1+\tau_2 j\omega)} \tag{6.1}$$

The technique to measure the neural discharge rate (impulses per second, ips) of the primary neurons in the vestibular nerve, made the determination of the transfer function of the semi canals possible [6]. The transfer function becomes:

$$H_{scc}(\omega) = \frac{K(1+\tau_L j\omega)}{(1+\tau_1 j\omega)(1+\tau_2 j\omega)} \tag{6.2}$$

With the following parameters given in the table 6.1.

K	2 $ips/m/s^2$
τ_n	0.11s
τ_1	5.9s
τ_2	0.005s

Table 6.1: The parameters of the model (eq. 6.2) of the semi-circular canals. (ips stands for the Neural discharge rate (impulses per second)

Because in the used path S the angular displacement is given as omega and not as acceleration the transfer function H_{scc} has to be differentiated. This is done with a lead filter [14] because a pure differentiator is difficult to implement. The transfer function of the lead filter is $\frac{j\omega}{\frac{j\omega}{\alpha}+1}$ where α is chosen larger than the bandwidth of Desdemona (see section A.5) so the lead compensation doesn't have influence in the relevant frequency domain. The final transfer function can be found in eq. 6.3 and is plotted in figure 6.4.

$$H_{scc(velocity)}(\omega) = \frac{K(1+\tau_L j\omega)}{(1+\tau_1 j\omega)(1+\tau_2 j\omega)} \frac{j\omega}{\frac{j\omega}{\alpha}+1} \tag{6.3}$$

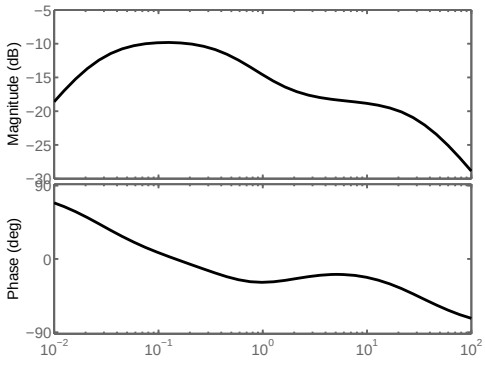


Figure 6.4: The frequent response of the semi-circular canals (from [12])

6.3.2 The otoliths

The otolith organs are one of the sensory components of the vestibular system. They translate gravitational and inertial forces into spatial orientation information - specifically, information about angular position (tilt) and linear motion of the head. The otolith organs (see figure 6.5) contain top-heavy hair-like cells that bend in response to gravitational or linear acceleration forces. The bending of the sensory hairs relative to the membranes / structures to which they are attached are transformed into orientation signals to the brain. Inertial forces associated with the movement of the human body, combined with gravity, act upon the otolith organs.

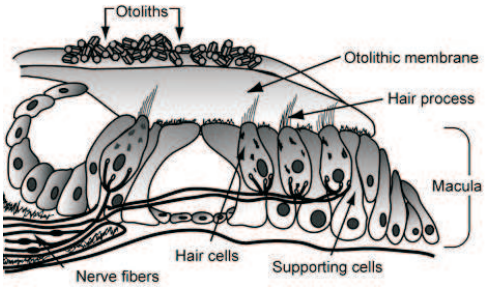


Figure 6.5: The otholits (from [12])

Dynamic model of the otoliths

Considering their physical structure, each otolith can be modelled as an acceleration meter with over-damped mass-spring-dashpot characteristics [6]. The transfer function is given as:

$$H_{oto}(\omega) = \frac{neural\ discharge\ rate}{linear\ acceleration} = \frac{K}{(1+\tau_1 j\omega)(1+\tau_2 j\omega)} \tag{6.4}$$

It has been very difficult to identify the time constants. The main reason is the lack of adequate experimental installations to generate necessary linear acceleration stimuli. Based on different arguments, researchers concluded that the numerator of equation (6.4) should be extended with a first-order term [6] to:

$$H_{oto}(\omega) = \frac{K(1+\tau_n j\omega)}{(1+\tau_1 j\omega)(1+\tau_2 j\omega)} \tag{6.5}$$

with the following parameters given in table 6.2. The transfer function is plotted in figure 6.6. It can be seen that the otolith is most sensitive to movements between 0.1 and 10 hertz.

K	3.4 $ips/m/s^2$
τ_n	1s
τ_1	0.5s
τ_2	0.016s

Table 6.2: The parameters of the model (eq. 6.5) of the otoliths. (ips stand for the Neural discharge rate (impulses per second, ips)

6.4 Including model of the semi-circular canals and the otoliths

Both the transfer functions are scaled so that $|H(\omega \to 0)| = 1$ to make it easier to interpret the results. The transfer functions are rewritten as a state space model and are included in the cost function. The new calculation scheme is given in section D.2.

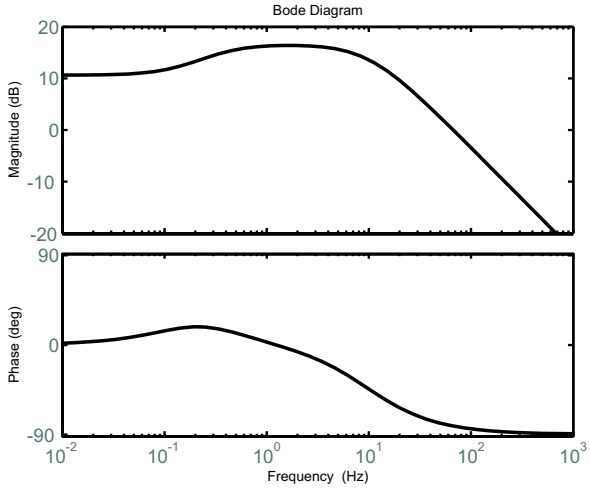


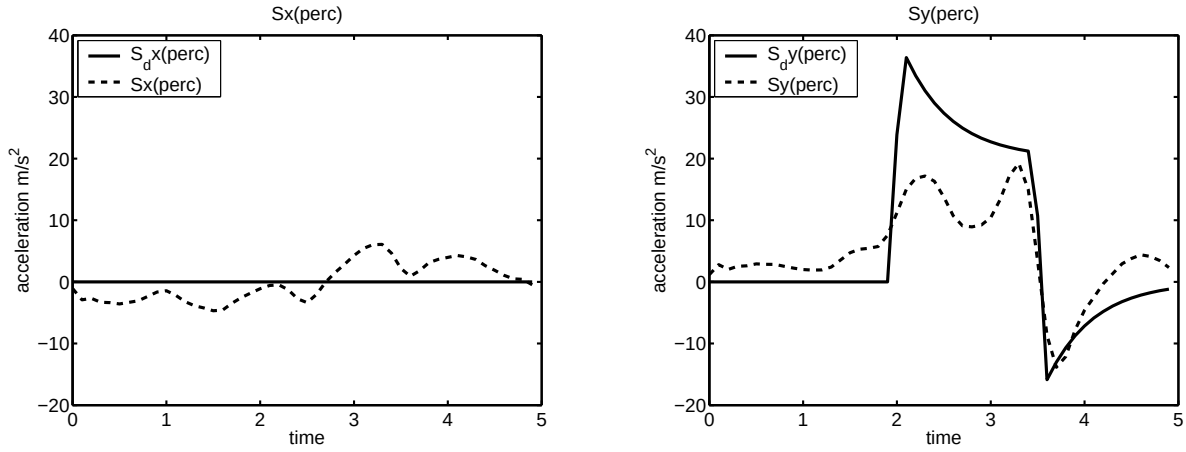
Figure 6.6: The bode plot of the otoliths transfer function eq. 6.5.

6.5 Results

6.5.1 Catapult Start

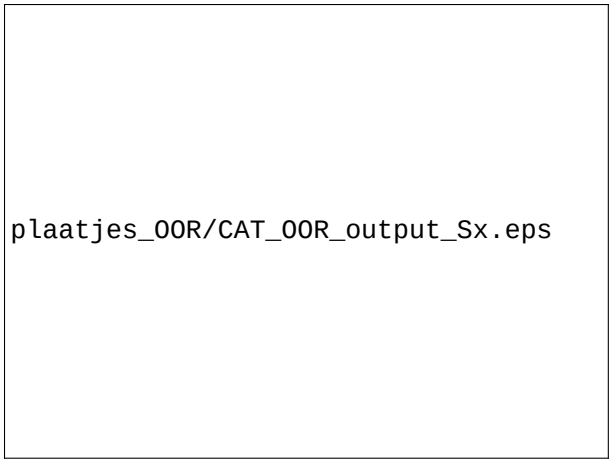
This experiment is an extension from the experiment which has been presented in section 5.6.2. The extension is that the effect of the otoliths are now included, the semi-circular canals are not included because in the original experiment the rotation is also not taken in to account. The result is given in figure 6.7 and 6.8. In figure 6.7 the output of the otoliths are given. It can be seen that a the otoliths are sensitive for an abrupt change in the acceleration (called jerk motion).

The results are better in comparison with the results from the same experiment without human perception model (see section 5.6.2). This is because the perception model is sensitive for acceleration changing which results in a steep acceleration pulse.

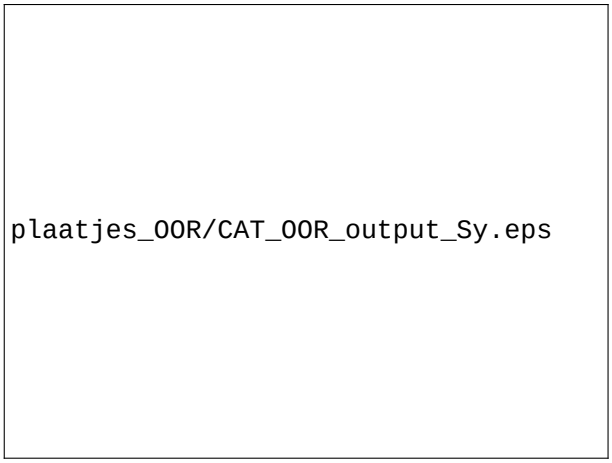


(a) (b)

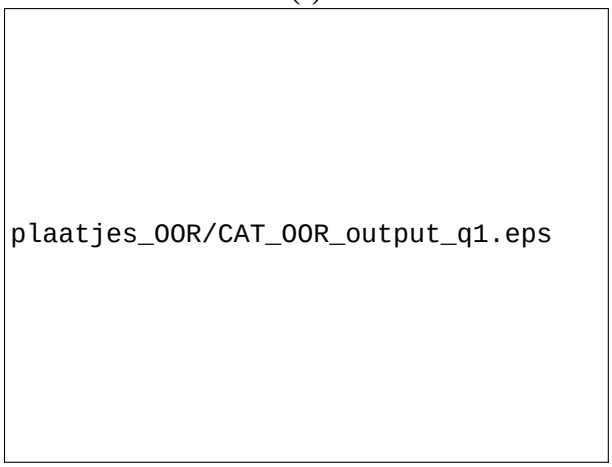
Figure 6.7: The best results for the three d.o.f. catapult start experiment. In (a) and (b) the $S_{x,perc}$ (perception, perc) and respectively $S_{y,perc}$ are plotted. This is the measured acceleration by the otoliths.

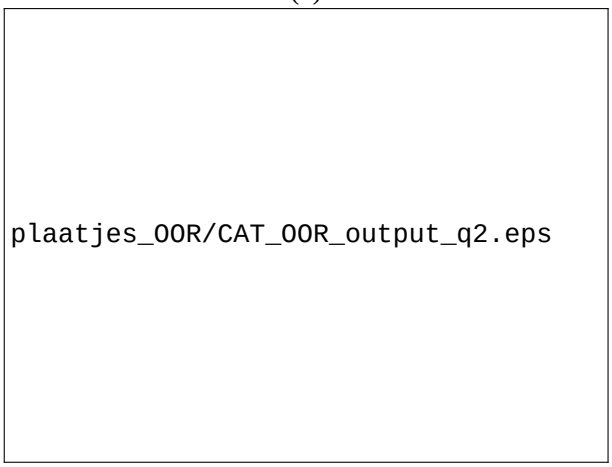


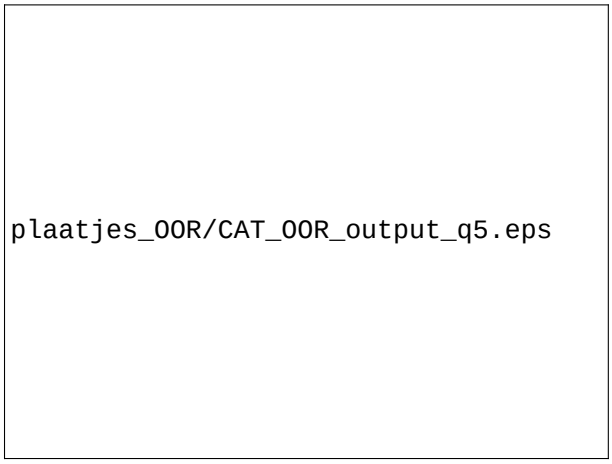
(a)



(b)







(e)

Figure 6.8: The best results for the three d.o.f. catapult start experiment. In (a) and (b) respectively S_x and S_y are plotted. In (c),(d) and (e) the positions of the joints respectively q_1, q_2 and q_5 are given. The constraints are not plotted but they are all met. The initial velocities are $[\dot{q_1}\ \dot{q_2}\ \dot{q_5}] = [-37.8\,degree/s\ 0.1\,m/s\ -12.7\,degree/s]$.

Chapter 7

Predictive optimal control

7.1 The principle of on-line optimization

In some cases there is a need for on-line optimization. An example is *man in the loop* (see figure 7.1). Since human behavior can not be predicted, S_d can not be determined prior to the optimization. Therefore the optimization has to be done online (see also 3.4).

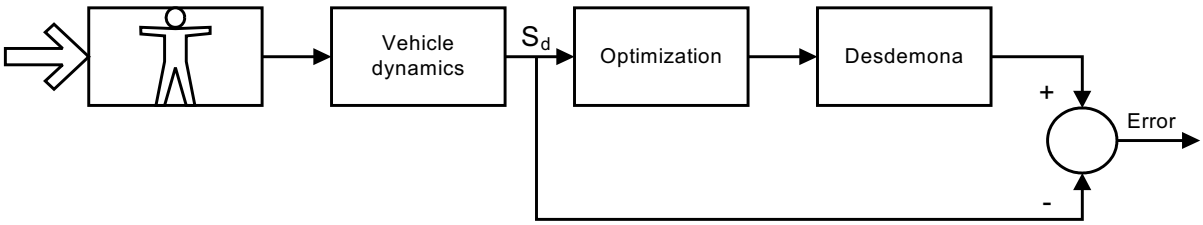


Figure 7.1: The calculation and horizon time.

The on-line optimization needs an estimated path with a certain time horizon T_h to calculate the path of the joints of Desdemona. The time necessary to calculate the optimization is the calculation time T_c (see figure 7.2). The determined path of the joints of Desdemona will be used as input during $t = T_c \ldots 2T_c$ and meanwhile the on-line optimization will determine the path again for the next time step.

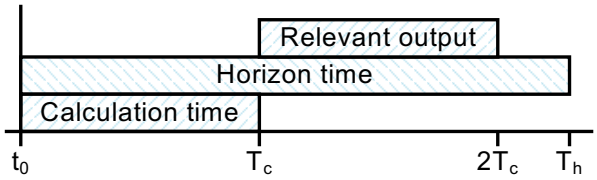


Figure 7.2: The calculation and horizon time.

The time of calculation should be short, at most half of T_h. They only way to make an on-line optimization is to configure Desdemona as a dynamic flight simulator, because with this configuration it is possible to realize a calculation that is short enough .

7.2 Desdemona as a dynamic flight simulator

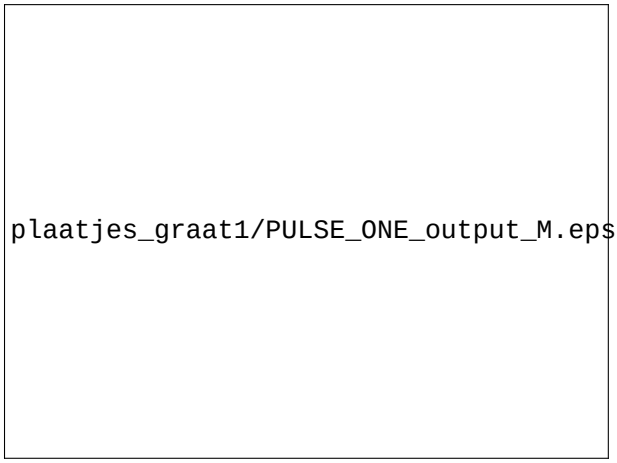
In the following part of this chapter, Desdemona is considered as a dynamic flight simulator. This means the vertical sledge is fixed in its outer position and the horizontal sledge is fixed in its lowest position. Desdemona works like a centrifuge-based simulator. The main benefit of the dynamic flight simulator configuration is that the simulated acceleration can be higher than in other configurations of Desdemona [2].

The angle of the acceleration vector working on the pilot can be changed with the gimbles. The optimization problem has to be redefined because the new goal is to optimize the path of the acceleration *modulus*.

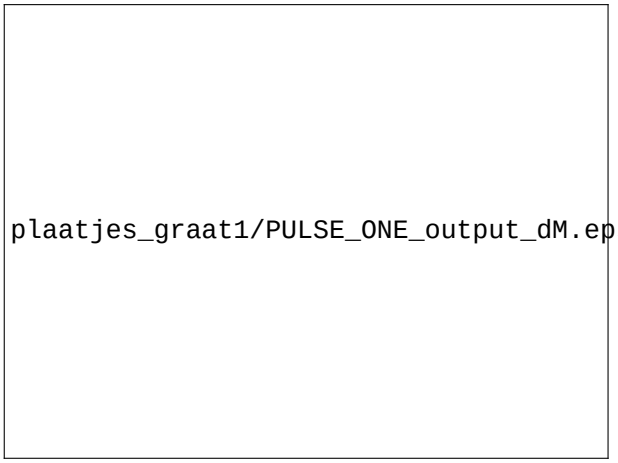
$$M_d = \sqrt{S_{dx}^2 + S_{dy}^2 + S_{dz}^2} \tag{7.1}$$

$$M = \sqrt{S_x^2 + S_y^2 + S_z^2} \tag{7.2}$$

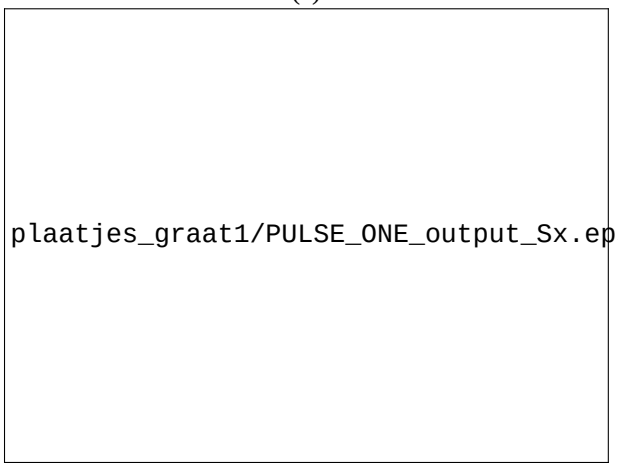
To give an impression of the possibilities of the dynamic flight simulator an off-line optimization is made for a catapult start. The results are given in figure 7.3.



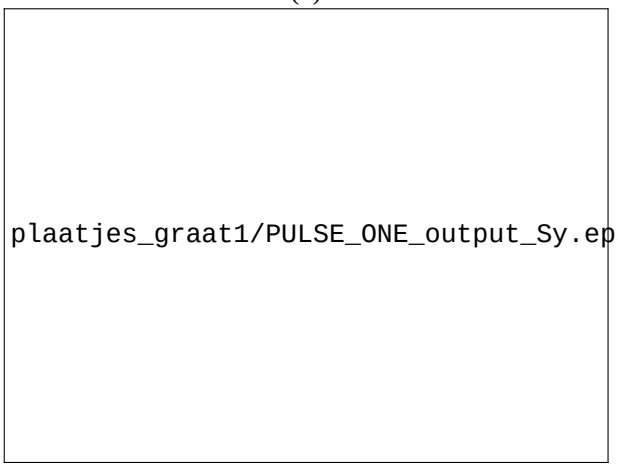
(a)



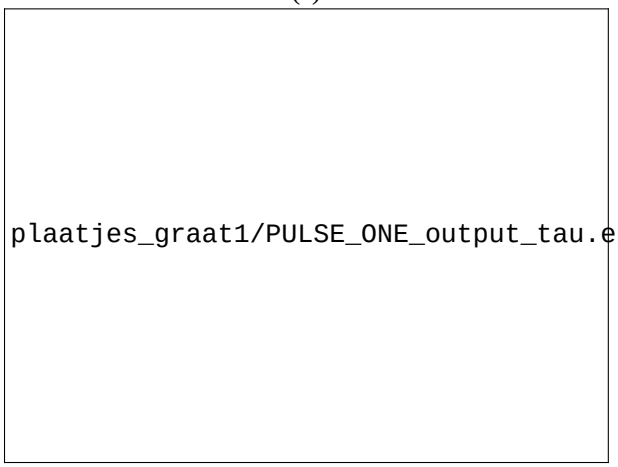
(b)



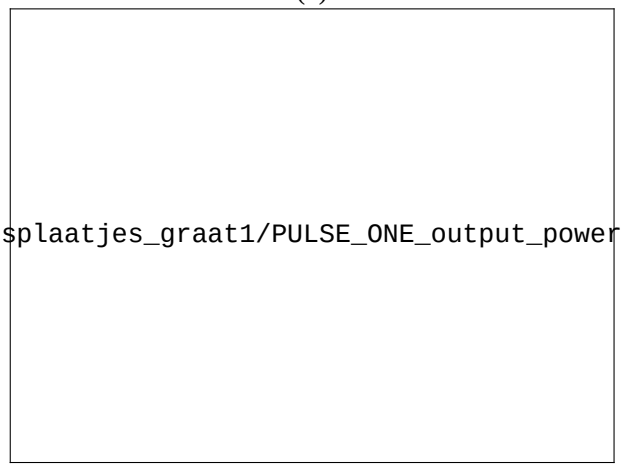
(c)



(d)



(e)



(f)

Figure 7.3: The results of the off-line catapult start. In (a) modulus M_d, M and M_{org} are given. M_{org} represents the original results of a non-optimized method (see section 7.4.2). (b) shows the error between M_d and M, and between M_d and M_{org}. In (c) and (d) respectively S_x and S_y are plotted. (e) and (f) show the motor force and power.

7.3 The on-line control method

The goal of the on-line control method is to generate in real time the path of the joints by using an estimated path (M_e). Since the estimated path is used as a criteria in the optimization it is important that the path is realistic. The method to estimate the path depends on the vehicle to be simulated and the mission of the pilot. In this report it is assumed that the path can be fully estimated. An overview of the optimization method is found in section D.1.2.

In equation 3.8 an online optimization algorithm with end cost is given. For the end cost of the dynamic flight simulator value zero can be taken, because the movements of the simulator are not limited. An example of limited movements of Desdemona are its horizontal movements.

The cost function of the dynamic flight simulator is almost the same as in equation 4.1. The discrete optimization problem is written as:

$$\min_{\ddot{q}} \; J = \sum_{k=t_0}^{t_e} \Delta M(\dot{q}(k), \ddot{q}(k), k)^2 \tag{7.3}$$

where $\Delta M(\dot{q}(k), \ddot{q}(k), k) = M_e(k) - M(\dot{q}(k), \ddot{q}(k))$ and with constrains:

$$\begin{aligned} |\dot{q}_1(k)| - \quad & \dot{q}_{1\,max} \quad \leq 0 \\ |\ddot{q}_1(k)| - \quad & \ddot{q}_{1\,max} \quad \leq 0 \\ |\tau(\ddot{q}_1(k))| - \quad & \tau_{max} \quad \leq 0 \\ |\tau(\ddot{q}_1(k))\dot{q}_1(k)| - \quad & P_{max} \quad \leq 0 \end{aligned} \tag{7.4}$$

The torque depends on $\ddot{q}_2$, so the constraint ($\ddot{q}$) and $\tau(\ddot{q}_1(k))$ can be rewritten as one constraint.

Another way of writing J is

$$J = \sum_{k=t_0}^{t_e} f(\dot{q}_1(k), \ddot{q}_1(k), k)^2 \tag{7.5}$$

with

$$f(\dot{q}_1(k), \ddot{q}_1(k), k) = \Delta M(\dot{q}_1(k), \ddot{q}(k), k) \tag{7.6}$$

$M(\dot{q}_1(k), \ddot{q}(k))$ can be calculated by combining equation B.5 and 7.2 and the parameters $q_2 = -4$, $\dot{q}_2 = 0$ and $\ddot{q}_2 = 0$. Then

$$M(\dot{q}_1(k), \ddot{q}(k)) = \sqrt{(4\ddot{q}_1(k))^2 + (4\dot{q}_1(k)^2)^2 + g^2} \tag{7.7}$$

The nature of the optimization problem described in the previous section is the same as the one described in chapter 4. This is because both problems have the form of a least-squares (data-fitting) problem and are nonlinear and non-convex. Therefore, the same optimization algorithm as in section 4.3 can be selected.

7.4 Reducing the optimization time

The calculation time can be reduced by speeding up the optimization and by choosing the initial values smartly. First a fast method of calculating the Jacobian is introduced. Thereafter, the method to choose the initial values is discussed.

7.4.1 Calculation of the Jacobian

The optimization method iterates to its optimum as explained in section C.2. Therefore the Jacobian has to be calculated at every iteration. This can be done numerically, but in this case it is faster to calculate the Jacobian analytically.

The Jacobian can be calculated by integrating the $\dot{q}(k)$ with the explicit Euler method and $\Delta M(k)$ with equation 7.7

$$\dot{q}(k) = \dot{q}(0) + h \sum_{n=1}^{k} \ddot{q}(n)$$

$$\Delta M(\dot{q}(k), \ddot{q}(k), k) = M_e(k) - M(\dot{q}(k), \ddot{q}(k))$$

Then the cost function $f(k) = (\Delta M(\dot{q}(k), \ddot{q}(k), k))^2$ can be calculated analytically. Thus the Jacobian is: $Jacobian(i,j) = \frac{\partial f(i)}{\partial \ddot{q}(j)}$, which results in:

$$Jacobian(i,j) = \begin{cases} 0 & i < j \\ \frac{\Delta M(i) R^2 2[2h\dot{q}(i)^3 + \ddot{q}(j)]}{M(i)} & i = j \\ \frac{\Delta M(i) R^2 4h\dot{q}(i)^3}{M(i)} & i > j \end{cases} \tag{7.8}$$

Thus for a $N \times N$ Jacobian matrix the number of calculations is $2N$.

7.4.2 Smart guessing of the initial values

The optimization time can be reduced by choosing the initial values closely to the optimal values. Therefore a small number of optimization (iteration) steps can be taken without having a worse performance. The new initial values ($\ddot{q}_1(k)$) can be calculated with a small error by neglecting the acceleration $\ddot{q}_1$ in equation 7.7. This can be done because $M(\dot{q}(k), \ddot{q}(k))$ mainly depends on $\dot{q}$ and g. The new equation is $M(\dot{q}(k)) = \sqrt{(4\dot{q}(k)_1^2)^2 + g^2}$ or rewritten[1] $\dot{q}_1(k) = \sqrt{\frac{\sqrt{M(k)^2 - g^2}}{4}}$. The estimated path can be used to calculate $\dot{q}_1$ thus $\dot{q}_1(k) = \sqrt{\frac{\sqrt{M_e(k)^2 - g^2}}{4}}$. With a control loop it is possible to calculate the new input ($\ddot{q}_1$) for the optimization and taking care of the constraint $|\ddot{q}_1(k)| - \ddot{q}_{1\,max} \leq 0$ (see figure 7.4).This method is a common method to control dynamic flight simulators.

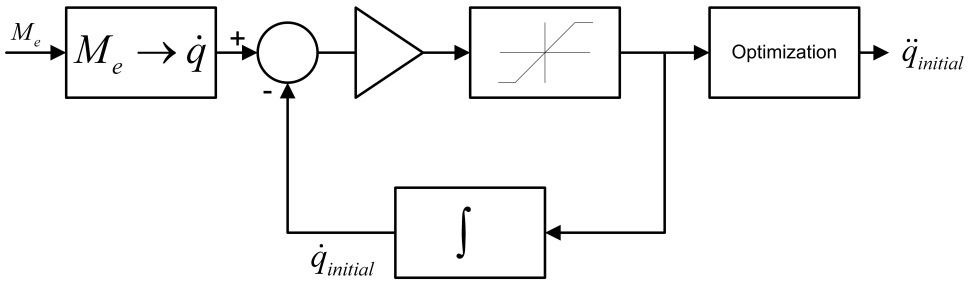


Figure 7.4: The schematic representation of calculating $\ddot{q}_1$ that is used for the initial value of the optimization

7.5 On-line examples

The catapult start and roller coaster experiments discussed in this section both have the same calculation $T_c = 0.4\ sec$, horizon $T_h = 1\ sec$, and sample time $h_{\ddot{q}} = 0.1\ sec$.

As can be understood intuitively, larger horizons give better results. Experiments also verify this, but the results of these are not displayed here.

The calculation time of the optimization algorithm has to be smaller or equal T_c to be useful in practical applications. When using an *AMD Athlon 1800* the calculation time of the optimization algorithm is smaller than T_c. So, the *AMD Athlon 1800* is suitable to do this optimization in practice.

The control loop method from section 7.4.2 (see figure 7.4) is originally used in dynamic flight simulators for calculating the central yaw [15]. To be able to compare the optimized results with the original results, the latter are also given.

[1] Without influencing the optimization results, $\dot{q}$ can be assumed positive.

7.5.1 Catapult start

For the on-line optimization example a path (M_d) is chosen that consists of a pulse superposition at a constant acceleration. The results can be found in figure 7.5. The average calculation time an *AMD Athlon 1800* needs is 0.23 second.

7.5.2 Roller coaster

The roller coaster experiment is given in figure 7.6. The M_d is specified by measured data from a real roller coaster to which halve of the gravity is added. This gravity is added to be able to simulate accelerations that are smaller then gravity[2]. Thereafter M_d is limited by the limits of Desdemona and filtered with a low pass filter of 5hz. The average calculation time an *AMD Athlon 1800* needs is 0.23 second.

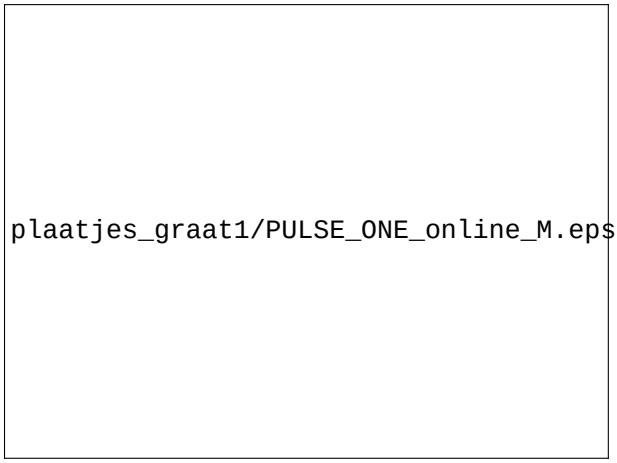
7.6 Discussion of the results

To evaluate the results two criteria will be considered: the performance and the calculation time. Both criteria have to be satisfied to be useful in a real time situation.

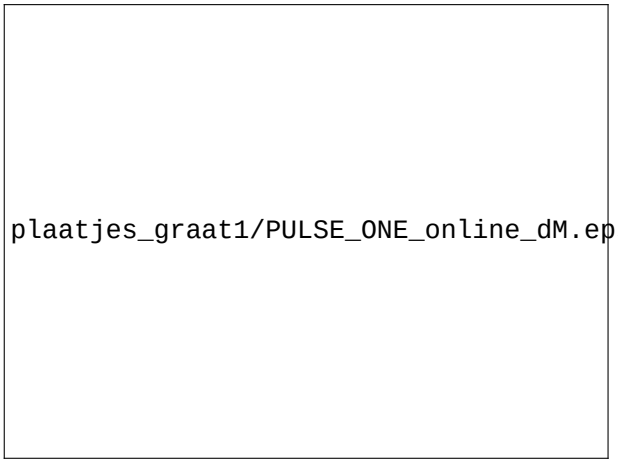
The performance of the optimal control algorithm is better then the original control loop method (see section 7.4.2). This is due to the fact that the optimization method is anticipating on the future. This results in a smaller delay between M_d and M.

The calculation time is small enough to make it applicable in practical situations. To conclude, the optimization method is a useful extension on current systems. When the path can be estimated for limited time (T_h) in the future then the method can be implemented. If the calculation time T_c is reduced then T_h can be reduced without a loss of performance. This can be useful for cases in which it is difficult to estimate the path.

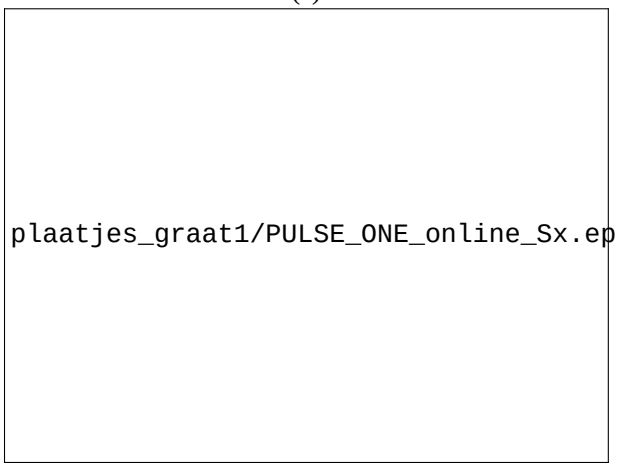
[2]This way also allows for steeper acceleration to be made. [15]



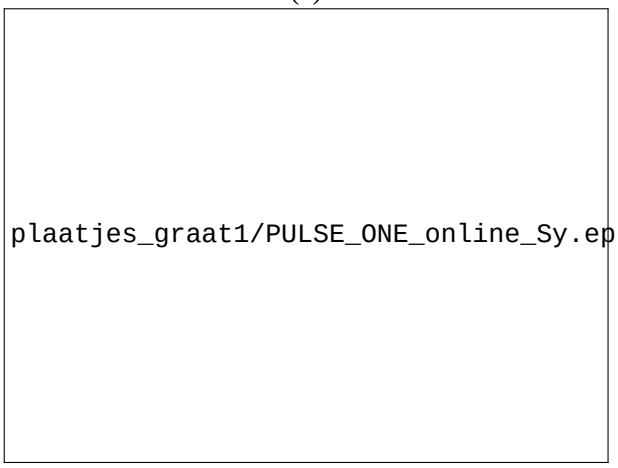
(a)



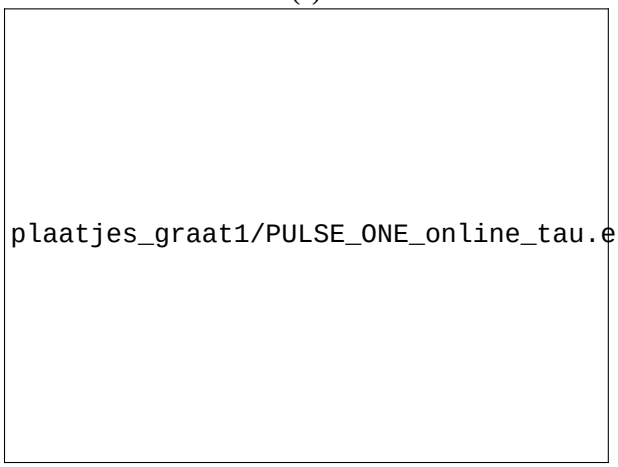
(b)



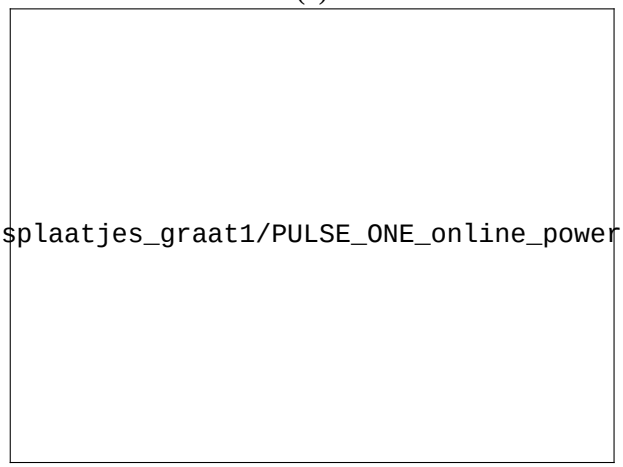
(c)



(d)



(e)



(f)

Figure 7.5: The results of the catapult start. In (a) modulus M_d, M and M_{org} are given. M_{org} represents the original results of a non-optimized method (see section 7.4.2). (b) shows the error between M_d and M, and between M_d and M_{org}. In (c) and (d) respectively S_x and S_y are plotted. (e) and (f) show the motor force and power.

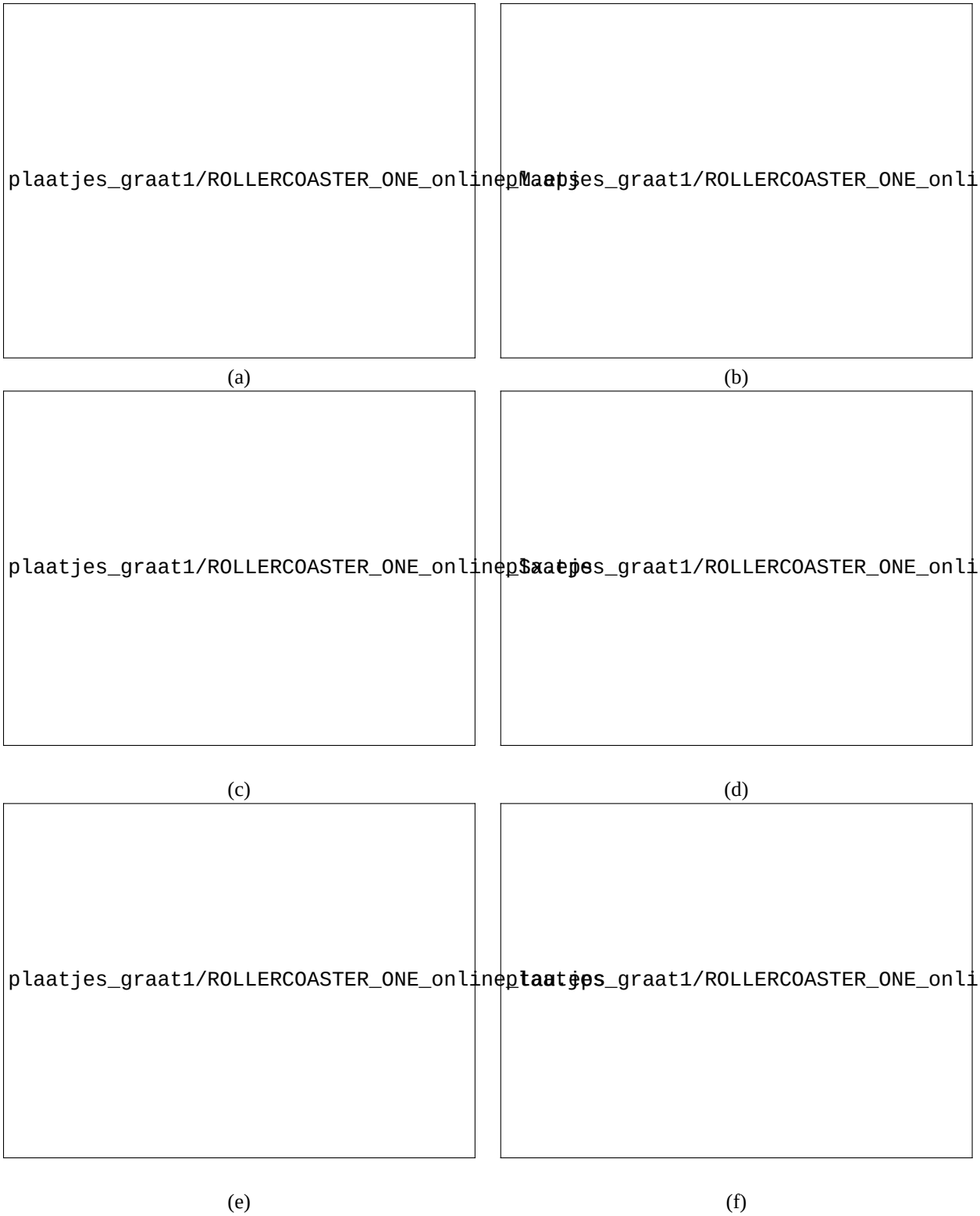


Figure 7.6: The results of the roller coaster experiment.In (a) modulus M_d, M and M_{org} are given. M_{org} represents the original results of a non-optimized method (see section 7.4.2). (b) shows the error between M_d and M, and between M_d and M_{org}. In (c) and (d) respectively S_x and S_y are plotted. (e) and (f) show the motor force and power.

Chapter 8

Conclusion and recommendations

8.1 Conclusions

For this research, first a literature study on modeling and simulation of robotic dynamics, optimal control concepts and Desdemona specifications and motion cueing methods has been conducted. Taking in regard the literature, an optimization method has been implemented which includes the differences in perception of the simulator, the false cues (false cues are wrongly simulated accelerations), and the structural and drive limits of Desdemona. Different suitable optimal control concepts and cost functions have been evaluated. The optimization method has been implemented for Desdemona with one, two or three d.o.f.. Finally, a real time optimal control algorithm which can be used during *man in the loop* simulations has been developed for Desdemona in the configuration of a dynamic flight simulator.

We can draw the following Conclusions:

- An analytical optimization is not possible for Desdemona with two d.o.f. without a numerical search algorithm to the co-states. In practice numerical optimizations instead of analytical optimizations are used for complex systems.
- The optimization includes the transitions from hexapod motion types to centrifuge motion types, because the optimization method only has one model for both kind of motions.
- The cost function is non-convex and therefore more than one local optimum may exist. There is no general rule that can be used to determine whether a local optimum is also a global optimum.
- It is time consuming to solve the optimization problem, because it is a non-convex, non-linear problem. Therefore huge calculations need to be done.
- It is easier to optimize a low frequency reference signal than a high frequency signal.
- Smart guessing of the initial values makes the performance better, because the optimization algorithm starts iterating from the initial values towards a local optimum. Since the initial values are realistic, it is more likely that the found local optimum has a good performance.
- By assuming that the center of mass is located in the origin, the calculation time will decrease and the optimization converges to an optimum faster. It also results in a small error in the dynamics, but this can be corrected.
- Human perception is sensitive to certain frequencies of linear acceleration. Including the human perception model in the cost function gives better results, because in that case the cost function pays more attention to those frequencies of the acceleration path.

- Optimization the Desdemona configured with one instead of three d.o.f. needs less computing power, because the number of optimization variables is smaller and the dynamics is less complex. The number of joints directly influence the number of variables: a one d.o.f. problem compared to a three d.o.f. problem has three times fewer input variables for the optimization. The dynamic is less complex because there are fewer joints which are coupled. Also the expression for the acceleration the pilot makes is less complex.
- If Desdemona is configured with only one d.o.f instead of more than one d.o.f. (see previous point), then it is easier to compute the optimization problem. In that case the optimization problem can be implemented as a real time application for Desdemona as dynamic flight simulator.

8.2 Recommendations

As stated before, the on-line optimal control algorithm which can be used during *man in the loop* simulations has been implemented for Desdemona configured as a dynamic flight simulator. To implement the algorithm, an estimator needs to be designed to predict the path in the future. The model for the estimator is dependent on the simulated model. The time that needs to be estimated can be reduced by reducing the time of calculation of the optimization algorithm. This could be done by optimizing the implementation code.

The recommendation for off-line optimization of Desdemona with more then one d.o.f is to apply an optimization method with smart chosen initial values. The results will be better than or equal to the results produced by the initial values. Implementing the optimization algorithm is especially recommended for cases in which the results of the initial values are not sufficient.

Including all the d.o.f. in the optimizing problem of the Desdemona will result in needs of huge computer power and probably with poor performance. It is not always necessary to include all the d.o.f. of Desdemona in the optimization, because by combining the optimized joints and the desired path (S_d), it is possible to calculate the positions of the other joints. This has been done when Desdemona configured as a dynamic flight simulator. Therefore, more research should be done to redefine the optimization problem for Desdemona was configured with (almost) all the d.o.f. to a configuration with a reduced number of d.o.f.

Bibliography

[1] M. B. Bannach. *Air Force Manual 11-217, Instrument Flight Procedures*. U. S. Air Force, volume 1 edition, December 2000.

[2] W. Bles, R. J. A. W. Hosman, and B. de Graaf. Desdemona: Advanced disorientation trainer and (sustained-g) flight simulator. volume AIAA-2000-4176 of *AIAA Modeling and Simulation Technologies Conference*, Soesterberg, Netherlands, August 2000. TNO Human Factors, American Institute of Aeronautics and Astronautics.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[4] E.F. Camacho. *Model predictive control*, volume 2. Springer-Verlag, 2003.

[5] S. Haykin. *An Introduction to Analog and Digital Communications*. John Wiley & Sons, 1989.

[6] R.J.A.W. Hosman. *Pilot's perception and control of aircraft motions*. PhD thesis, Delft University of Technology, 1996.

[7] E. Kreyszig. *Advanced Engineering Mathematics*. eighth edition (Special Indian Edition). John Wiley & Sons, 1999.

[8] F. Martinsen, L. T. Biegler, and B. A. Foss. A new optimization algorithm with application to nonlinear mpc. *Journal of Process Control 14 (2004) 853-865*, 2004.

[9] The Mathworks, Inc, 3 Apple Hill Drive Natick, MA 01760-2098. *Matlab optimization toolbox user's guide*, version 3 edition, Oktober 2004. online only, http://www.mathworks.com.

[10] Richard M. Murray, Zexiang Li, and S.Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[11] Y. Nakamura. *Advanced Robotics: Redundancy and optimization*. Addison-Wesely Publising Company, Inc., 1991.

[12] National Aeronautics and Space Administration (NASA). *Educational Brief, The Effects of Space Flight on the Human Vestibular System*, eb-2002-09-011-ksc edition, September 2002.

[13] A.S. Nemirovsky and D.B. Yudin. *Problem complexity and method efficiency in optimization*. Wiley-Interscience Series in Discrete Mathematics. John Wiley and Sons Ltd, 1983.

[14] N.S.Nise. *Control Systems Engineering*, volume 2. The Benjamin/Cummings Publishing Company, Inc., 1995.

[15] C. H. Spenny, B. S. Liebst, T. L. Chelette, C. Folescu, and J. Sigda. Development of a sustainable-g dynamic flight simulator. AIAA-2000-4075. American Institute of Aeronautics and Astronautics, 2000.

[16] S. Stramigioli. *Modeling and IPC control of interactive mechanical: a co-ordinate free approach*. LNCIS serie of Springer. Springer-Verlag London, 2001.

[17] T. van den Boom and B. Schutter. *Optimization in Systems and Constrol*. Delft University of Technology, Mekelweg 2 2628CD Delft, The Netherlands, August 2004.

[18] H.J. Zwart and J.W. Polderman. *Optimal Control (Systeem- en besturingstheorie)*. University of Twente, Drienerlolaan 5 7522NB Enschede, The Netherlands, August 2001.

Appendix A

Notation and Model

A.1 Notation

The notation which is used in this report is based on [16]. It is a geometrical and global method. In this appendix some of the notations are explained and the parameters of Desdemona are given.

A.1.1 The homogeneous transformation matrix

A general change Cartesian coordinates from coordinates 1 into 2 can be expressed with a matrix of the form:

$$H_1^2 = \begin{pmatrix} R_1^2 & p_1^2 \\ 0_3^T & 1 \end{pmatrix} \tag{A.1}$$

In this matrix R is the rotation matrix and p the translation vector.

With a serial robot like Desdemona, it is possible to first define the transformation matrices for each joint, and then combine them. For example $H_j^i = H_k^i H_j^k$.

Rotations around the x,y and z axis can be defined as followed.

$$H_{x\,axis}(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.2}$$

$$H_{y\,axis}(\beta) = \begin{pmatrix} cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.3}$$

$$H_{z\,axis}(\gamma) = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.4}$$

Translations can be written as:

$$H_{translation}(x, y, z) = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.5}$$

A.1.2 Twist

The twist is defined as $T = [\omega\ v]^T$ and can also be written in matrix notation.

$$\tilde{T}_i^{i,j} = \dot{H}_i^j H_j^i \tag{A.6}$$

The indices in $\tilde{T}_k^{i,j}$ mean a twist from k to j expressed in from coordinates i. The coordinates from the twist can also be changed with help from the adjoint matrix [16] $T_i^{j,j} = Ad_{H_i^j} T_i^{i,j}$.

A.2 Model

In figure A.1 the axis from Desdemona are defined. The numbering of the axis is done from earth to cabin with the coordinate system of earth as number zero, and the cabin number six. The angles q have the same order of numbering with q_1 as the angle between the earth and the central yaw. The transformation matrices are defined as:

$$\begin{array}{rcl} H_1^0 = & H_{z\,axis}(q_1) \\ H_2^1 = & H_{translation}(0, q_2, 0) \\ H_3^2 = & H_{translation}(0, 0, q_3) \\ H_4^3 = & H_{x\,axis}(q_4) \\ H_5^4 = & H_{z\,axis}(q_5) \\ H_6^5 = & H_{y\,axis}(q_6) \end{array} \tag{A.7}$$

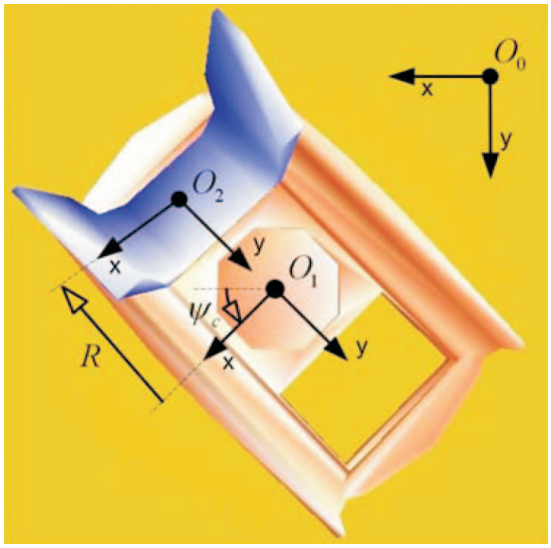
The twists are defined as:

$$\begin{array}{rcl} T_1^{0,0} & = & [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0]^T \\ T_2^{1,1} & = & [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0]^T \\ T_3^{2,2} & = & [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1]^T \\ T_4^{3,3} & = & [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T \\ T_5^{4,4} & = & [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0]^T \\ T_6^{5,5} & = & [0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]^T \end{array} \tag{A.8}$$

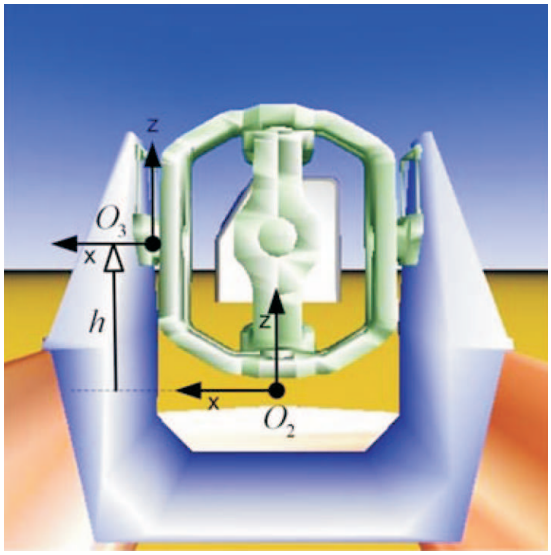
The Jacobian matrix J^n gives the linear relationship between the joint velocities and the end-effector n twist. This can be written as $T_1^{0,n} = J^n(q)\dot{q}$

The Jacobian matrix can be calculated with

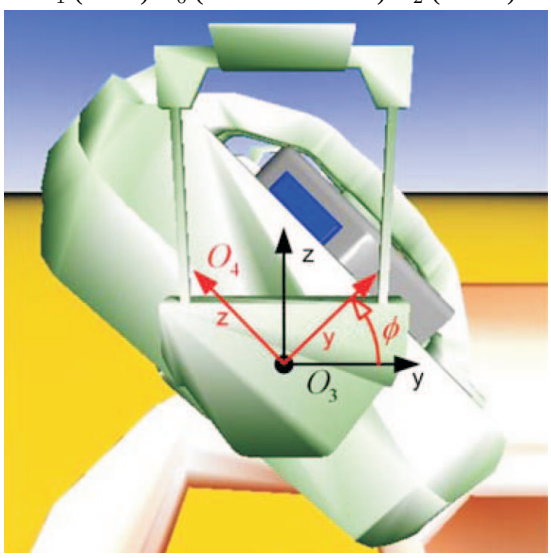
$$J^n(q) = [T_1^{0,0} \dots T_n^{0,n-1}] \tag{A.9}$$



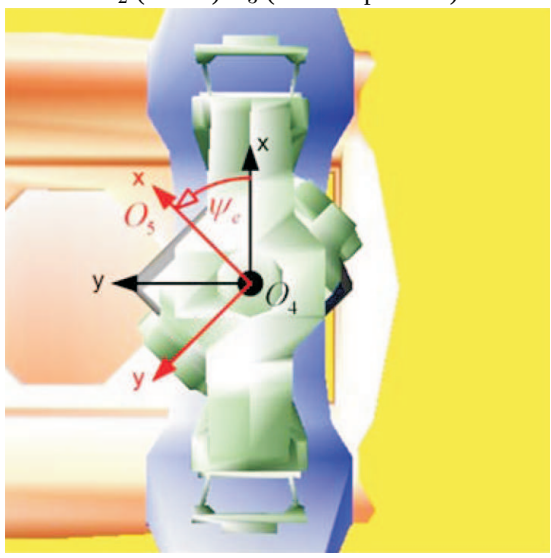
O_1 (Earth) O_0 (Horizontal track) O_2 (Heave)



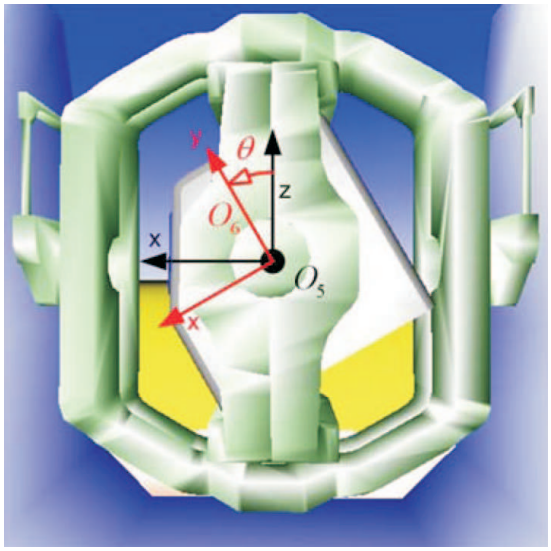
O_2 (Heave) O_3 (Vertical position)



O_3 (Vertical position) 0_4 Outer gimble



0_4 outer gimble O_5 (Inner gimble)



O_5 (Inner gimble) O_6 (Cabin)

Figure A.1: The coordinates frame (O) of Desdemona per part

A.3 Calculating the path of Desdemona.

The path which Desdemona has to follow is given in translation acceleration and rotation speeds in the origin of the head. It is also possible to express the trajectory of Desdemona in the same way. The expression is

$$S = \begin{pmatrix} \omega(t) \\ \ddot{P}(t) + G(t) \end{pmatrix} \tag{A.10}$$

where ω is the rotation of the head. P the acceleration and G the gravity.

To calculate P, first the twist and his derivative has to be calculated as follows:

$$\tilde{T}_6^{0,0} = \dot{H}_6^0 H_0^6 \tag{A.11}$$
$$T_6^{6,0} = Ad_{H_0^6} T_6^{0,0} \tag{A.12}$$
$$\dot{\tilde{T}}_6^{6,0}(q,\dot{q})_{ij} = \sum_{k=1}^{6} \left(\frac{\partial (T_6^{6,0})_{ij}}{\partial q_k} \dot{q}_k + \frac{\partial (T_6^{6,0})_{ij}}{\partial \dot{q}_k} \ddot{q}_k \right) \tag{A.13}$$

The $\ddot{p} \in \mathbb{R}^3$ consists of the three upper elements of $\ddot{P} \in \mathbb{R}^4$ and can be calculated with the help of the basic equation $P^0 = H_6^0 P^6$. The acceleration $\dot{P}^0$ can then be expressed as: $\ddot{P}^0 = \ddot{H}_6^0 P^6$ or rewritten as: $\ddot{P}^0 = \ddot{H}_6^0 H_0^6 P^0$

$$\ddot{H}_6^0 H_0^6 \leftrightarrow (\dot{\dot{H}}_6^0) H_0^6 \tag{A.14}$$
$$\leftrightarrow (\dot{\tilde{T}}_6^{0,\dot{0}} H_6^0) H_0^6 \tag{A.15}$$
$$\leftrightarrow \dot{\tilde{T}}_6^{0,0} + \tilde{T}_6^{0,0} \dot{H}_6^0 H_0^6 \tag{A.16}$$

hence

$$\ddot{P}^0 = \dot{\tilde{T}}_6^{0,0} P^0 + \tilde{T}_6^{0,0} \dot{P}^0 \tag{A.17}$$

For calculating the acceleration of point P^0 you need the twist $\tilde{T}_6^{0,0}$ and his derivative $\dot{\tilde{T}}_6^{0,0}$. If you want the acceleration seen from coordinate frame 6 then the equation becomes:

$$(\ddot{P}^0)^6 = \dot{\tilde{T}}_6^{6,0} P^0 + \tilde{T}_6^{6,0} \dot{P}^0 \tag{A.18}$$

The gravity vector can be calculated with:

$$G = \{ G \in \mathbb{R}^3 | \begin{pmatrix} G \\ 0 \end{pmatrix} = H_0^6 \begin{pmatrix} 0 \\ 0 \\ g \\ 0 \end{pmatrix} \} \tag{A.19}$$

where g the gravity constant is.

A.4 Euler Lagrangian equations and Dynamic model

Definitions of the necessary quantities

The manipulator inertia matrix [10] can be calculated with:

$$M(q) = \sum_{i=1}^{n} (J^i)^T I^{0,i} J^i \tag{A.20}$$

With $I^{0,i}$ the generalized inertia matrix of body i expressed in coordinate frame 0. The parameters can be found at the end of this section.
The potential energy in the system can be calculated with

$$V(q) = \sum_{i=1}^{n} m_i g h_i(q) \tag{A.21}$$

where m_i and $h_i(q)$ as respectively the mass and the height of the i^{th} link and g is the gravitational constant. Then the conservative force can be defined as

$$N_i(q) = \frac{\partial V(q)}{\partial q_i} \tag{A.22}$$

The Coriolis matrix $C(q, \dot{q})$ gives the Coriolis and centrifugal terms in the equation of motion.

$$C_{ij}(q, \dot{q}) = \frac{1}{2} \sum_{k=1}^{n} \left(\frac{\partial M_{ij}(q)}{\partial q_k} + \frac{\partial M_{ik}(q)}{\partial q_j} - \frac{\partial M_{kj}(q)}{\partial q_i} \right) \dot{q}_k \tag{A.23}$$

Euler Lagrangian equations

To make a dynamical model of Desdemona the Euler Lagrangian equations are used. The Lagrangian for the rigid body is written [10] in terms of the joint angles and velocities. The Lagrangian is given in equation A.24 .

$$L(q, \dot{q}) = \frac{1}{2} \dot{q}^T M(q) q - V(q) \tag{A.24}$$

The equation of motion can be acquired by using the Lagrange's equation. (eq A.25).

$$\tau_i = \frac{d}{dt} \frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} \tag{A.25}$$

The equation of motion can now be rewritten as:

$$\tau = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + N(q, \dot{q}) \tag{A.26}$$

where $M(q)$as the inertia matrix, $C(q, \dot{q})$ as the Coriolis matrix C, $N(q, \dot{q})$ as vector for the Conservative forces and friction.

The power that is exchanged between the actuating motor and the joint can be calculated with $P = \tau^T \dot{q}$

The inverse dynamic model

The equation of motion A.26 can be rewritten as $\ddot{q} = f(\dot{q}, q, \tau)$ which give the inverse dynamic equation as function of τ.

$$\begin{aligned} M^{-1}\tau &= M^{-1}M(q)\ddot{q} + M^{-1}C(q, \dot{q})\dot{q} + M^{-1}N(q, \dot{q}) \\ \ddot{q} &= M^{-1}\tau - M^{-1}C(q, \dot{q})\dot{q} - M^{-1}N(q, \dot{q}) \end{aligned} \tag{A.27}$$

A.5 Parameters of Desdemona

All the parameters that are used in this report are mentioned in this section. In table A.1 the motor limits are given. In table A.2 and A.3 the mass and respectively inertia are given. Table A.4 gives the limits of the body of Desdemona.

This report is a public version, therefor this table is not displayed. For more information [2].

Table A.1: This are the effective motor limits.

This report is a public version, therefor this table is not displayed. For more information [2].

Table A.2: This are the mass and the center of gravity in Desdemona coordinates ('-' is not important in the dynamic model)

This report is a public version, therefor this table is not displayed. For more information [2].

Table A.3: The moment of inertia in the origin of Desdemona coordinate system. ('-' is not known or not important in the dynamic model, and taken as zero). The units that are used are $[kg\ m^2]$

The center of gravity and the origin of Desdemona coordinate system are not the same in case of the Inner gimble and cabin. For taking this effect into account the mass is rewritten as a generalized mass ($(I_{mass})^m = \begin{pmatrix} 0 & 0 \\ 0 & mI \end{pmatrix}$) and the coordinates are transfer to the same coordinates as the moment of inertia. $(I_{mass})^0 = \left(Ad_{H_m^0}\right)^T I^m Ad_{H_m^0}$. Now the generalized inertia of the total body in the origin of Desdemona coordinates is $I = (I_{mass})^0 + \begin{pmatrix} J & 0 \\ 0 & 0 \end{pmatrix}$ Where J is the moment of inertia of the body

This report is a public version, therefor this table is not displayed. For more information [2].

Table A.4: The physical limits, due to limited dimensions, stiffness and forces of Desdemona body. *For Desdemona as dynamic flight simulator (section 7) another maximum is valid:* $\dot{q}_{1max} = 155.4°/s$

This report is a public version, therefor the bandwidth of the cockpit is not given.

Appendix B

Model with two degrees of freedom

In this appendix a model of Desdemona configured with only the joints q_1 and q_2 (the central yaw respectively vertical sledge) will be defined. The other joints are fixed. For this system it is possible to define a problem which is similar to the full Desdemona, but because it has fewer decrees of freedom, it is easier (faster) to solve the problem. The reason for choosing q_1 and q_2 as joints is that both joins can generate an acceleration in the e_y direction in the cockpit.

B.0.1 Basic equations

The equation of motion (eq. A.26) is for this case

$$\tau = \begin{pmatrix} i_{e_z^1} + i_{e_z^2} + m_2(q_2)^2 & 0 \\ 0 & m_2 \end{pmatrix} \ddot{q} + \begin{pmatrix} q_2 m_2 \dot{q}_2 & q_2 m_2 \dot{q}_1 \\ -q_2 m_2 \dot{q}_1 & 0 \end{pmatrix} \dot{q} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{B.1}$$

The twist vector $T_2^{2,0}$ (eq. A.8)is.

$$T_2^{2,0} = \begin{pmatrix} 0 \\ 0 \\ \dot{q}_1 \\ -q_2 \dot{q}_1 \\ \dot{q}_2 \\ 0 \end{pmatrix} \tag{B.2}$$

The acceleration vector $\ddot{P}$ (eq. A.18)is.

$$\ddot{P} = \begin{pmatrix} -q_2 \ddot{q}_1 - 2\dot{q}_1 \dot{q}_2 \\ \ddot{q}_2 - \dot{q}_1^2 q_2 \\ 0 \\ 0 \end{pmatrix} \tag{B.3}$$

The gravity vector G (eq. A.18)is.

$$G = \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} \tag{B.4}$$

The path vector S is.

$$S = \begin{pmatrix} 0 \\ 0 \\ \dot{q}_1 \\ -q_2 \ddot{q}_1 - 2\dot{q}_1 \dot{q}_2 \\ \ddot{q}_2 - \dot{q}_1^2 q_2 \\ g \end{pmatrix} \tag{B.5}$$

B.1 Analytical optimization

B.1.1 The used theory

The analytical optimization is based on the *Minimum Principle of Pontryagin* [18]. It solves the following optimization problem:

$$\min_{\tau} \; J = K(x(t_e)) + \int_{t_0}^{t_e} f_0(x(t, t_0, x_0, u), u(t), t)\, dt \tag{B.6}$$

subject to
$\dot{x} = f(x(t), u(t))$ and $x(t_0) = x_0$
With the hamiltonian the condition for optimality can be deduced. An hamiltonian is defined as: $H = p(t)^T f(x(t), u(t)) + f_0(t, x(t), u(t))$ (with $p(x(t), u(t))$ as co-state, $f(x(t), u(t))$ the dynamic system and $f_0(t, x(t), u(t))$ the cost function)

Then with the *Minimum Principle of Pontryagin* can be found that for an optimal control minimizing equation B.6, the following equations hold.

$$\dot{x}_{opt}(t) = \frac{\partial H}{\partial p}(p_{opt}(t), x_{opt}(t), u_{opt}(t))^T \tag{B.7}$$

$$x_{opt}(0) = x_0 \tag{B.8}$$

$$\dot{p}_{opt}(t) = -\frac{\partial H}{\partial x}(p_{opt}(t), x_{opt}(t), u_{opt}(t))^T \tag{B.9}$$

$$p_{opt}(t_e) = \frac{\partial K}{\partial x}(x_{opt}(t_e))^T \tag{B.10}$$

If there exists an optimal solution then

$$\frac{\partial H}{\partial u}(p_{opt}(t), x_{opt}(t), u_{opt}(t)) = 0$$

B.1.2 Solving the example with the Minimum Principle of Pontryagin

The problem statement is given in section 3.5.

For getting the optimal problem in the same form as the Minimum Principle of Pontryagin, it has to be rewritten as a first order problem.
Therefore it is necessary to introduce $x = \left\{ x \in \mathbb{R}^4, q \in \mathbb{R}^2, \dot{q} \in \mathbb{R}^2 \,|\, x = \begin{pmatrix} q \\ \dot{q} \end{pmatrix} \right\}$. With x the dynamic of Desdemona can be rewritten as $\ddot{q}(t) = f(x(t), \tau(t))$ and finally as first order differential equation.

$$\dot{x}(t) = \begin{pmatrix} x_3(t) \\ x_4(t) \\ f(x(t), \tau(t)) \end{pmatrix} \tag{B.11}$$

$$= R((x(t), u(t)) \tag{B.12}$$

where $u = \tau$.

To avoid violation of the constraints, the error function f_0 is not only including ΔS but also the constraints. The extra addition in the cost function is called a feasibility function, which is explained for the numerical part in section 4.6. To ensure the torque will be constrained a feasibility function is taken as $b\tau_1^2$ and $c\tau_2^2$, where b and c are weight coefficients. The position of q_2 will be included as aq_2^2 (with a as weight coefficient). The rest of the constraints are not taken into account. The total error function becomes:

$$f_{feas} = aq_2^2 + b\tau_1^2 + c\tau_2^2 \tag{B.13}$$

$$f_0 = \left(\cos(wt) - R_4((x(t), u(t)) - (x_3)^2 x_2\right)^2 + f_{feas} \tag{B.14}$$

There are no final costs defined, thus $K(x(t_e)) = 0$

Now it is possible to calculate the ODE's for the example:

$$\dot{x} = \begin{pmatrix} \dot{q} \\ \ddot{q} \end{pmatrix} = \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ -\frac{p_3}{2(iz\_1+iz\_2+{q_2}^2 M_2)^2 b} - 2\frac{q_2 M\_2 \dot{q}_2 \dot{q}_1}{iz\_1+iz\_2+{q_2}^2 M\_2} \\ \frac{-1}{2}\frac{p_4 - 2\cos(wt) + 4\dot{q}_1^2 q_2}{2+c{M\_2}^2} + \dot{q}_1^2 q_2 \end{pmatrix}$$

$$\dot{p} = \begin{pmatrix} 0 \\ \begin{bmatrix} \left(2\frac{\tau_1 q_2 M_2}{(iz\_1+iz\_2+{q_2}^2 M_2)^2} + 4\frac{{q_2}^2 M\_2^2 \dot{q}_2 \dot{q}_1}{(iz\_1+iz\_2+{q_2}^2 M_2)^2} - 2\frac{M_2 \dot{q}_2 \dot{q}_1}{iz\_1+iz\_2+{q_2}^2 M_2}\right) p_3 \dots \\ \dots - \dot{q}_1^2 p_4 + 4\left(\cos(wt) - \frac{\tau_2}{M_2} - 2\dot{q}_1^2 q_2\right)\dot{q}_1^2 - 2aq_2 \end{bmatrix} \\ -p_1 + 2\frac{q_2 M_2 \dot{q}_2 p_3}{iz\_1+iz\_2+{q_2}^2 M_2} - 2q_2 \dot{q}_1 p_4 + 8\left(\cos(wt) - \frac{\tau_2}{M_2} - 2\dot{q}_1^2 q_2\right) q_2 \dot{q}_1 \\ -p_2 + 2\frac{q_2 M_2 \dot{q}_1 p_3}{iz\_1+iz\_2+{q_2}^2 M_2} \end{pmatrix}$$

and

$$\tau = \begin{pmatrix} -1/2\frac{p_3}{(iz\_1+iz\_2+{q_2}^2 M_2)b} \\ -1/2\frac{M_2\left(p_4 - 2\cos(wt) + 4\dot{q}_1^2 q_2\right)}{1+c{M_2}^2} \end{pmatrix}$$

The begin value is given by $x_{opt}(t_0) = x_0$ and the end value by $p_{opt}(t_e)$=0.

Hence to solve the 8 nonlinear differential equations with 4 end and 4 begin condition analytical is very difficult. Therefore the the problem is rewritten as a numerical problem. This has been done in section 4.2.

Appendix C

Optimization methods and definitions

In this appendix the properties of a convex function are given and the *Newton* optimization method is explained.

C.1 Convex

C.1.1 Convex set

A set $\mathcal{C}$ in $\mathbb{R}$ is convex if for each pair $x, y \in \mathcal{C}$ and for all $\Theta \in [0, 1]$ the next property holds:

$$(1 - \Theta)x + \Theta y \in \mathcal{C}$$

C.1.2 Convex function

A function J is convex if [3]

1. The domain dom(J) is a convex set
2. The next inequality holds for all $x, y \in dom(J)$ and $\Theta \in [0, 1]$

$$J\left((1 - \Theta)x + \Theta y\right) \leq (1 - \Theta)J(x) + \Theta J(y)$$

Geometrically, this inequality means that the line segment between [x, J(x)] and [y, J(y)], which is the chord from x to y, lies above the graph of f (see figure. C.1)

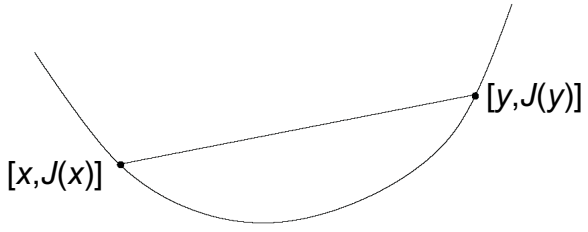


Figure C.1: 1 Graph of a convex function. The chord (i.e., line segment) between any two points on the graph lies above the graph.

C.1.3 First-order conditions

Suppose J is differentiable (i.e., its gradient ∇J exists at each point in dom(J)). Then as can be seen in [3] J is convex if and only if dom(J) is convex and

$$J(y) \geq J(x) + \nabla J(x)^T (y - x) \tag{C.1}$$

holds for all $x, y \in dom(J)$

As one simple example, the inequality C.1 shows that if $\nabla J(x) = 0$, then for all $y \in domJ$,$J(y) \geq J(x)$, i.e., x is a global minimizer of the function J. This example shows also that there is only one local minimum [3].

C.2 Newton and Quasi-Newton methods

The *Newton* method [17] is based on a second-order Taylor series expansion of the cost function J. If the cost function is $J(x)$ then the Taylor series around x_0 is

$$\widetilde{J}(x) = J(x_0) + \nabla^T J(x_0)(x - x_0) + \frac{1}{2}(x - x_0)^T H(x_0)(x - x_0)$$

The higher order terms are neglected and the Jacobian (gradient) is $\nabla^T J(x_0)$ and the Hessian is $H(x_0)$.

For the (local) minimum the derivative of the cost function is zero.

$$\min_x J(x) \Rightarrow \nabla J(x) = 0$$

Thus for the Taylor function $\nabla \widetilde{J}(x) = 0 \Rightarrow \nabla J(x_0) + H(x_0)(x - x_0) = 0$. An estimate for the optimum of the original problem is given by

$$x_{opt} = x_0 - H^{-1}(x_0)\nabla J(x_0).$$

To improve the results we can repeat this procedure and so we obtain *Newton*'s algorithm, which is characterized by the following iteration step:

$$x_{k+1} = x_k - H^{-1}(x_k)\nabla^T J(x_k)$$

The estimate x_{k+1} will be good if the higher order terms in the Taylor series are small. The closer we are to the optimum x_{opt}, the smaller the higher order terms will be. This means that , if the initial guess x_0 is close enough to x_{opt}, the approximation will become better each step and the algorithm converges.

The *Newton* algorithm does not always perform well, e.q. the inversion of the Hessian matrix may become numerical ill-conditioned if the Hessian matrix is almost singular. Another problem is that the computation of the Hessian matrix $H(x_k)$ is time consuming. Therefore there are several algorithms to avoid these problems [17].

Appendix D

Implementation of the optimization problem

In this chapter way of the optimization program works is described. First the off-line and on-line optimizations are described and thereafter the used cost functions. In the last section the reference list from the used Matlab files is given.

D.1 Optimization program

D.1.1 Off-line optimization

The implementation of the off-line optimization is given in the flow diagram of figure D.1.The program starts in the block 'start' where it loads some basic information like: mass, max motor power, S_d, etc. Then the program goes to the block 'initial' where it loads the initial values for the optimization. Thereafter the real optimization takes place. Then a choice is made (in block 'Another optimization?') if the number of optimizations is enough or if another run is required. If not, then the best optimization with the best result is displayed.

The most important part of the program is the optimizing part therefore some details are given (see the zoom in figure D.1). First the Jacobian (The Jacobian is explained in section C.2) is calculated from the cost function(The used cost functions can be found in section D.2). Then the jacobian is used for the line optimization (see section 4.4.1). Then one optimization step (iteration) is done. If the stop criteria is satisfied then the optimization is finished, or else another step is done.

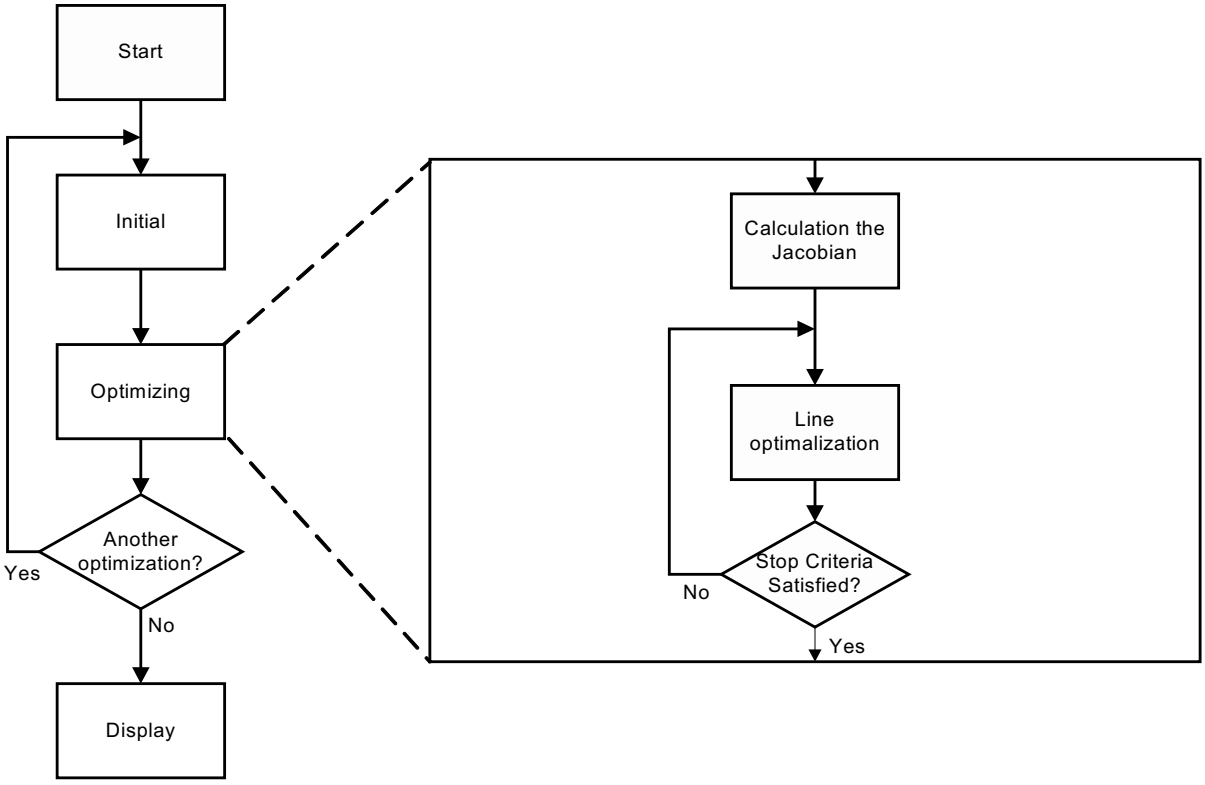


Figure D.1: The off-line optimization program

D.1.2 on-line optimization

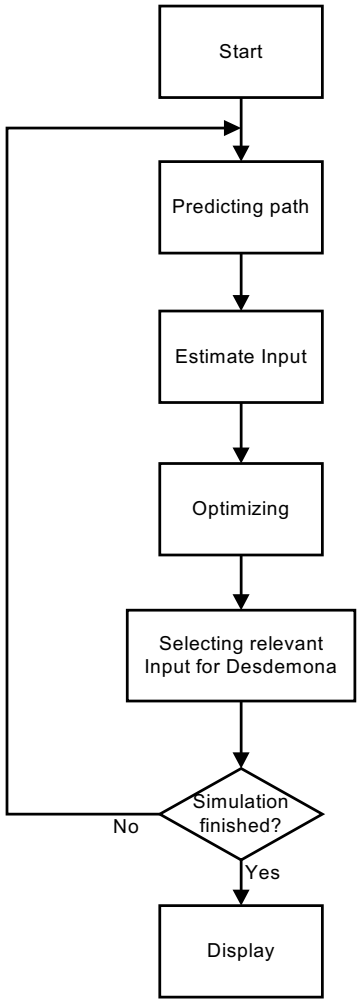


Figure D.2: The on-line control program

The implementation of the on-line optimization is plotted in a flow diagram see figure D.2.The program starts in the block 'start' where it loads some basic information like: mass, max motor power, S_d, etc. Then the program goes the block 'Predicting path' where the path is predicted T_h in the future. This path is used to estimate the input for the optimization. (The optimization is described in detail in the previous section.) After the optimization the relevant[1] input for Desdemona is selected. If the simulation is not finished then the hole process is repeated, or else the final results are displayed. One cycle costs T_c of time.

[1]With relevant is mentioned the input samples that are valid at the time the optimization is finished until T_c in the future

D.2 Different cost functions that are used in this report

An optimization problem is minimizing a cost function, in this section the implementation of used cost functions are given. The related Matlab files are given at the end of this chapter.

The Basic cost function algorithm. The following steps are done every time the cost function is calculated. First the discrete q and $\dot{q}$ are calculated. Then the path S is calculated and thereafter τ. As last the cost function (output) is calculated. Which is used by the optimization method to evaluate the input.
This can be written in a short form as:
Input: $q(0)$ $\dot{q}(0)$ $\ddot{q}$

1. $[q, \dot{q}]$=integration_method($q(0)$, $\dot{q}(0)$, $\ddot{q}$)
2. $[S]$=calculation_S(q, $\dot{q}$, $\ddot{q}$)
3. $[\tau]$=calculation_τ(q, $\dot{q}$, $\ddot{q}$)
4. $[J]$=calculation_cost(S_d, S, τ, q, $\dot{q}$, $\ddot{q}$)

Output: J

Extended cost function. The following steps which are done every time the cost function is calculated. First the discrete q is resampled to q_s(with the new sample frequency fs_{itr}). In the nexst step q and $\dot{q}$ are calculated. Then after resampeling every vector to the frequencyfs_{S_d} the path S is calculated and thereafter τ. As last the cost function (output) is calculated. Which is used by the optimization method to evaluate the input
This can be written in a short form as:
Input: $q(0)$ $\dot{q}(0)$ $\ddot{q}$

1. $[\ddot{q}_s]$=resample($\ddot{q}$)
2. $[q_s, \dot{q}_s]$=integration_method($q(0)$, $\dot{q}(0)$, $\ddot{q}_s$)
3. $[q_{ss},\dot{q}_{ss}, \ddot{q}_{ss}]$=resample($q_s$, $\dot{q}_s$, $\dot{q}_s$)
4. $[S]$=calculation_S(q_{ss}, $\dot{q}_{ss}$, $\ddot{q}_{ss}$)
5. $[\tau]$=calculation_τ(q_{ss}, $\dot{q}_{ss}$, $\ddot{q}_{ss}$)
6. $[J]$=calculation_cost(S_d, S, τ, q_{ss}, $\dot{q}_{ss}$, $\ddot{q}_{ss}$)

Output: J

The cost function with human perception model. The following are steps which are done every time the cost function is calculated. First the discrete q is resampled to q_s(with the new sample frequency fs_{itr}). In the next step q and $\dot{q}$ are calculated. Then after resampling every vector to the frequency fs_{S_d} the τ and the path S are calculated. Then the human perception model is included. To be able to calculate the human perception the step site is reduced by resampling S. After calculating the human perception the step size is changed back by again resampling and then the cost function is calculated (as function from the observed path).
This can be written in a short form as:
Input: $q(0)$ $\dot{q}(0)$ $\ddot{q}$

1. $[\ddot{q}_s]$=resample($\ddot{q}$)
2. $[q_s, \dot{q}_s]$=integration_method($q(0)$, $\dot{q}(0)$, $\ddot{q}_s$)
3. $[q_{ss},\dot{q}_{ss}, \ddot{q}_{ss}]$=resample($q_s$, $\dot{q}_s$, $\dot{q}_s$)
4. $[\tau]$=calculation_τ(q_{ss}, $\dot{q}_{ss}$, $\ddot{q}_{ss}$)
5. $[S]$=calculation_S(q_{ss}, $\dot{q}_{ss}$, $\ddot{q}_{ss}$)

6. $[S_s]$=resample(S)
7. $[S\_perc_s]$=human_perseption_model(S_s)
8. $[S\_perc]$=resample(S_pers_s)
9. $[J]$=calculation_cost(S_perc_d, S_perc, τ, q_{ss}, $\dot{q}_{ss}$, $\ddot{q}_{ss}$)

Output: J

The Online cost function and Jacobian. The following are steps which are done every time the cost function is calculated. First $\dot{q}$ is calculated. Thereafter the path S is calculated and thereafter the jacobian. As last the cost function (output) is calculated.
This can be written in a short form as:
Input: $\dot{q}(0)$ $\ddot{q}$

1. $[\dot{q}_1]$=integration_method($\dot{q}_1(0)$, $\ddot{q}_1$)
2. $[S]$=calculation_S(q_1, $\dot{q}_1$, $\ddot{q}_1$)
3. $[Jacobian]$=calculation_Jacobian(M, $\dot{q}_1$, $\ddot{q}_1$)
4. $[J]$=calculation_cost(M_e, M, τ, $\dot{q}_1$, $\ddot{q}_1$)

Output: J

D.3 Reference list from the used files

In this section the structure and the filenames are given for the optimization program. It is the same structure for on-line and off-line optimization but the code inside the files is different.

start `main_file.m`

Loading parameters `parameters.m`

Loading S_d `Load_Sd.m`

Prepare the optimizing `optimizing_ini.m`

- **Optimizing** `lsqnonlin`
- **cost function** `costfunction.m`
  - **resample** `resample.m`
  - **ODE45** `integration_method.m`
  - **S and** τ `Spath.m`
  - **(Possible) Human perception model** `Human_perseption_model.m`
  - **(Possible) Jacobian** `Jacobian.m`
  - **Calculation of the cost function** `The_cost.m`

Display Output.m