Adaptive control of a RRR-manipulator

Arjen den Hamer

Report No. DCT 2003.68

Report of internal traineeship

Coaches: ir. D. Kostic
dr.ir. B. de Jager

Supervisor: prof.dr.ir. M. Steinbuch

Technische Universiteit Eindhoven (TU/e)
Department of Mechanical Engineering
Control Systems Technology Group

Eindhoven, July 5, 2003
Contents

1 Manipulator Description ...................................................... 3

2 Feedback-linearization .................................................. 9
  2.1 Adaptive case .................................................................. 10
    2.1.1 Comments on adaptive feedback linearisation ............ 11

3 Passivity based controller .................................................. 12
  3.1 Passivity based regulators .............................................. 14
  3.2 Passivity based tracking controllers ................................. 16
    3.2.1 Slotine and Li ......................................................... 16

4 Adaptive case ....................................................................... 18
  4.1 Slotine and Li ................................................................ 18
    4.1.1 Comments on the controller of Slotine and Li ............ 19
  4.2 Berghuis, Ortega and Nijmeijer ...................................... 20
    4.2.1 Comments on the controller of Berghuis, Ortega and Nijmeijer .... 22

5 Implementation ................................................................. 23
  5.1 Tuning of parameters ..................................................... 23
    5.1.1 Choosing $K_p$ and $K_d$ ............................................ 23
    5.1.2 Choice of $\Gamma$ and $\Gamma_{fric}$ ................................ 24
    5.1.3 Choice of $\lambda$ and $\lambda_0$ .................................... 24
  5.2 Implementation of the regressor ....................................... 25
  5.3 Reference trajectory ..................................................... 26
  5.4 Simulations .................................................................... 27
    5.4.1 Simulations without friction ....................................... 29
    5.4.2 Simulation with friction ............................................. 30
  5.5 Experiments ................................................................. 31
    5.5.1 Experimental results: controller of Slotine and Li ........ 33
    5.5.2 Experimental results: controller of Berghuis, Ortega and Nijmeijer .. 34
    5.5.3 High adaptation gains on the friction parameters ............ 38
    5.5.4 Comparison with previous work ................................. 39

6 Conclusion .......................................................................... 41

A Matrix $M$ and vector $h$ .................................................. 45
CONTENTS

B Regressor implemented in C 47
Introduction

In model based control, the performance of a controller is mainly determined by the accuracy of the model describing the system. Mostly this model is not able to perfectly describe the real system. This is partly due to simplified structure of the model. High frequent dynamics, like amplifier dynamics or flexibilities in manipulator-links, are often neglected because they do not really contribute to describing the dynamics of interest and make the model too complex for real-time implementation. On the other hand, there are some effects like friction which are hard to describe in a mathematical way so that they have to be approximated.

Another reason for model-errors is the fact that model-parameters are not exactly known or even change slightly. Parameters can vary due to several reasons. Think of manipulators which pick up masses, so that their inertia behaviour changes or wear of bearings which causes different friction properties.

These model uncertainties leads to performance degradation or even instable systems. One way to deal with uncertainties is by using robust controllers. The price which has to be paid for robustness is performance. In robust control there is no ability to learn in order to make the parameter uncertainties smaller.

Since the eighties much attention is paid to adaptive control based on mathematical stability proofs. These algorithms use error information and knowledge of the system dynamics to reduce the parameter uncertainties. It has to be mentioned that adaptive controllers are not able to compensate for errors in the model-structure. Another condition for adaptive control is that the speed of parameter variation has to be much smaller than the dynamics of the system. These two points make adaptive controllers rather sensitive to unmodelled dynamics, which can result in instability of the closed-loop system. This will come up for discussion during the experimental results.

Although the theory of adaptive control is pretty well worked out in literature, there seems to be less information about implementing adaptive controller on manipulators with many degrees of freedom and many uncertain parameters. This report will handle adaptive-control applied to a 3-dof robot with 22 unknown parameters and can therefore be seen as an extension to the usual set-up in papers.

The report has the following set-up:

- First the modelling of the manipulator will be described in chapter 1. Some properties which are needed in adaptive control will be described.

- After that, chapter 2 and 3 discusses some important adaptive schemes from literature with their stability-proofs. Drawbacks and advantages will be compared in order to choose two algorithms to implement. These algorithms are further worked out in chapter 4.

- These two algorithms are tested in simulation and in practice and the results are presented and compared in chapter 5.

- At last, conclusions and recommendations on adaptive control are given.

During the report, some tuning rules for the controller will be given. Problems which can occur during simulations and experiments will be described and explained from a mathematical point of view. As a consequence, much attention is paid to mathematical derivations of the closed-loop system and stability proofs.
Chapter 1

Manipulator Description

In literature many articles on adaptive control can be found, however often applied on manipulators with one or two degrees of freedom. This low number of degrees of freedom (DOF) leads to relative simple systems with few parameters which is fine from an educational point of view. Contrary to most of these articles, the purpose of this report is to test adaptive controllers for a manipulator with more than two degrees of freedom and many unknown parameters. This lead to a more challenging and realistic situation as it can be found in industrial environments.

Such a manipulator which has 3 revolute DOF’s and many unknown parameters is situated in the lab of the section Dynamics and Control Technology at the TUE. It is chosen to use this manipulator for simulations and experiments as described in this report. Figure 1.1 and 1.2 depict respectively a photo and a schematic overview of the manipulator.

It can be seen that the first joint-axis is aligned perpendicular to the ground. Therefore there is no need for compensation of gravitational forces acting on the first joint. The second and third joint-axis lie in the horizontal plane. This gives some extra complexities due to compensation for gravity. The schematic overview of the manipulator, as depicted in figure 1.2, also shows how the zero-positions of the different joints are defined. A more extended kinematic and dynamic description of the manipulator, inclusive Denavit Hartenberg parameters, is given in the paper [1].

Under assumption of rigid links, the dynamics of the manipulator can be modelled using multi-body-dynamics techniques. This results in the following model as derived in [1].

\[ M(q)\ddot{q} + h(q, \dot{q}) + f_{fric}(\dot{q}) = \tau \]  \hspace{1cm} (1.1)

Where:
- \( q = [q_1, q_2, q_3]^T \) are the angular positions of the different joints in as defined in figure 1.2
- \( \tau = [\tau_1, \tau_2, \tau_3]^T \) represents the applied torques generated in the different joints.

The matrix \( M(q) \) represent inertia terms, \( h(q, \dot{q}) \) represents a vector with centripetal, Coriolls and gravitational forces and \( f_{fric} \) represent the friction forces working in the different joints. A symbolic expressions for \( M(q) \) and \( h(q, \dot{q}) \) can be found in appendix A. Because the correctness of the dynamic representation of the manipulator in equation (1.1) is already proved during previous work, this model will be used as starting point of further work in this report.

In order to keep the order of the model low, the dynamics of the motors and amplifier are neglected. This can be justified because electronic/magnetic dynamics occurring in amplifiers
and electronic motors are much faster than the mechanical dynamics of the manipulator. The following static relation is assumed for the relation between the torques generated in the joints and the applied voltage.

\[ \tau = Ku \]  

\[ (1.2) \]

The vector \( u = [u_1, u_2, u_3]^T \) represents the applied voltages in the different joints. The diagonal matrix \( K \) contains the motor-constants and is therefore defined as:

\[
K = \begin{bmatrix}
12 & 0 & 0 \\
0 & 6 & 0 \\
0 & 0 & 3
\end{bmatrix} \quad [Nm/V] \tag{1.3}
\]

The most common adaptive algorithms described in literature exploit the structure of a C-matrix. The structure of the C-matrix was not used before so it has to be derived. In order to get this C-matrix, the vector \( h(q, \dot{q}) \) from equation (1.1) has to be split up in a part \( C(q, \dot{q})\dot{q} \) which represents centripetal and Coriolis forces and a vector \( G(q) \)\(^1\) representing the gravitational forces. With this split-up, the dynamics of equation (1.1) can be written as:

\[
M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + f_{fric} = \tau
\]

\[ (1.4) \]

Because the \( C \) matrix is not unique, there are several ways to do this split-up. Due to further derivations the following property has to be fulfilled.

\(^1\)Contrary to the common representation of vectors, the vector with gravitational forces is represented by the capital symbol \( G(q) \). This is done in order to prevent confusion with the gravitation constant \( g \).
property 1:

\[ q^T (\dot{M}(q) - 2C(q, \dot{q}))q = 0 \]  

(1.5)

In other words: find a matrix \( C(q, \dot{q}) \) such that \( \dot{M}(q) - 2C(q, \dot{q}) \) is skew-symmetric. This property can be found mathematically with the definition of Christoffel coefficients [2]. A physical interpretation for this property is the following: The change of the manipulator’s kinetic energy \( q^T \dot{M} \dot{q} \) must equal the power input provided by the actuators and torques caused by gravitational forces.

\[
\frac{d}{dt} \left[ \frac{1}{2} q^T M(q) \dot{q} \right] = [\tau - G(q)] \dot{q}
\]

(1.6)

When we assume that \( \tau_{fric} \) in equation (1.4) equals zero and using the fact that \( M(q) \) is symmetric, it can be derived that:

\[
q^T M(q) \dot{q} + \frac{1}{2} \dot{q} M(q) = [\tau - G(q)] \dot{q}
\]

\[
[\tau - C(q, \dot{q})] \dot{q} - G(q) \dot{q} + \frac{1}{2} \dot{q} M(q) = [\tau - G(q)] \dot{q}
\]

\[
q^T \left( \frac{1}{2} \dot{M}(q) - C(q, \dot{q}) \right) \dot{q} = 0
\]

(1.7)

It is chosen to split-up the vector \( h(q, \dot{q}) \) based on this skew-symmetric property, to determine \( C(q, \dot{q}) \) and \( G(q) \). So the following conditions has to hold:

\[
C(q, \dot{q}) \dot{q} + G(q) = h(q, \dot{q})
\]

(1.8)

\[
\dot{M}(q) - 2C(q, \dot{q}) \text{ is skew symmetric}
\]

(1.9)

Because the matrix \( C(q, \dot{q}) \) contains only centripetal and Coriolis forces, it does not depend on the gravity. So every term in \( h(q, \dot{q}) \) which contains the gravitation constant, has to be an element of \( G(q) \). The gravitational constant appear only linearly in \( h(q, \dot{q}) \) so the following equation can be used to extract \( G(q) \) from \( h(q, \dot{q}) \) by symbolic computing with, for example, the Symbolic Toolbox of Matlab:

\[
G(q) = \frac{\partial h \quad g}{\partial q}
\]

(1.10)

Consequently \( C(q, \dot{q}) \dot{q} \) is also known from equation (1.8).

\[
C(q, \dot{q}) \dot{q} = h(q, \dot{q}) - G(q)
\]

(1.11)

Still it is not known which velocity-terms of \( C(q, \dot{q}) \dot{q} \) belongs to \( C(q, \dot{q}) \). Therefore the following set of equations has to be solved using the skew-symmetric property:

\[
\dot{M}(1, 2) = C(2, 1) + C(1, 2)
\]

\[
\dot{M}(1, 3) = C(1, 3) + C(3, 1)
\]

\[
\dot{M}(2, 3) = C(2, 3) + C(3, 2)
\]

(1.12)

Notice that the centripetal terms in the vector \( h(q, \dot{q}) \) are already known. For these quadratic terms, it doesn’t matter which \( \dot{q}_i \)-term is placed in the \( C \)-matrix and which \( q_i \)-term is placed in \( \dot{q} \) because they are both the same. The terms in \( h(q, \dot{q}) \) which can not be fit in
the set of equations above has to be diagonal elements of the matrix $C$. Solving this set of equations leads to the following expressions for $C(q, \dot{q})$ and $G(q)$:

\begin{align}
C(1, 1) &= -\frac{1}{2} \sin(2q_2)\dot{q}_2\theta_1 + \frac{1}{2} \cos(2q_2)\dot{q}_2\theta_2 \\
&\quad - a_2 \sin(q_2 + q_3)\dot{q}_3\theta_3 - a_2 \sin(2q_2 + q_3)\dot{q}_2\theta_3 \\
&\quad - a_2 \cos(q_2 + q_3)\dot{q}_3\theta_4 - a_2 \cos(2q_2 + q_3)\dot{q}_2\theta_4 \\
&\quad - \frac{1}{2} \sin(2q_3 + 2q_2)\dot{q}_2\theta_5 - \frac{1}{2} \sin(2q_3 + 2q_2)\dot{q}_3\theta_5 \\
&\quad - \frac{1}{2} \cos(2q_3 + 2q_2)(\dot{q}_2 + \dot{q}_3)\theta_6 \\
C(1, 2) &= - \sin(q_2)\dot{q}_2\theta_1 0 + \cos(q_2)\dot{q}_2\theta_1 1 - \frac{1}{2} \sin(2q_2)\dot{q}_1\theta_1 \\
&\quad + \frac{1}{2} \cos(2q_2)\dot{q}_1\theta_2 - a_2 \sin(2q_2 + q_3)\dot{q}_1\theta_3 - a_2 \cos(2q_2 + q_3)\dot{q}_1\theta_4 \\
&\quad - \frac{1}{2} \sin(2q_3 + 2q_2)\dot{q}_1\theta_5 - \frac{1}{2} \cos(2q_3 + 2q_2)\dot{q}_1\theta_6 - (q_2 + q_3)\sin(q_2 + q_3)\theta_8 \\
&\quad + (\dot{q}_2 + \dot{q}_3)\cos(q_2 + q_3)\theta_9 \\
C(1, 3) &= - a_2 \sin(q_2 + q_3)\dot{q}_1\theta_3 + \cos(q_2 + q_3)\dot{q}_1\cos(q_2)\theta_4 \\
&\quad - \frac{1}{2} \sin(2q_2 + 2q_3)\dot{q}_1\theta_5 - \frac{1}{2} \cos(2q_3 + 2q_2)\dot{q}_1\theta_6 \\
&\quad - (q_2 + q_3)\sin(q_2 + q_3)\theta_8 + (\dot{q}_2 + \dot{q}_3)\cos(q_2 + q_3)\theta_9 \\
C(2, 1) &= \frac{1}{2} \sin(2q_2)\dot{q}_1\theta_1 - \frac{1}{2} \cos(2q_2)\dot{q}_2\theta_1 \\
&\quad + a_2 \sin(2q_2 + q_3)\dot{q}_1\theta_3 + a_2 \cos(2q_2 + q_3)\dot{q}_1\theta_4 \\
&\quad + \frac{1}{2} \sin(2q_3 + 2q_2)\dot{q}_1\theta_5 + \frac{1}{2} \cos(2q_3 + 2q_2)\dot{q}_1\theta_6 \\
C(2, 2) &= - a_2 \dot{q}_3\sin(q_3)\theta_3 - a_2 \dot{q}_3\cos(q_3)\theta_4 \\
C(2, 3) &= - a_2 \sin(q_3)\dot{q}_3\theta_3 + \cos(q_3)\dot{q}_3\theta_4 - a_2 \dot{q}_2\sin(q_3)\theta_3 - a_2 \dot{q}_2\cos(q_3)\theta_4 \\
C(3, 1) &= \frac{1}{2} a_2 \sin(q_3)\dot{q}_1\theta_3 + \frac{1}{2} \sin(2q_2 + q_3)\dot{q}_1\theta_3 \\
&\quad + \frac{1}{2} a_2 \dot{q}_1\cos(q_3)\theta_4 + \frac{1}{2} a_2 \cos(2q_2 + q_3)\dot{q}_1\theta_4 \\
&\quad + \frac{1}{2} \sin(2q_3 + 2q_2)\dot{q}_1\theta_5 + \frac{1}{2} \cos(2q_3 + 2q_2)\dot{q}_1\theta_6 \\
C(3, 2) &= a_2 \dot{q}_2\sin(q_3)\theta_3 + a_2 \dot{q}_2\cos(q_3)\theta_4 \\
C(3, 3) &= 0
\end{align}

\begin{align}
G_1 &= 0 \\
G_2 &= g \cos(q_2 + q_3)\theta_3 - g \sin(q_2 + q_3)\theta_4 + g \cos(q_2)\theta_15 - g \sin(q_2)\theta_16 \\
G_3 &= g \cos(q_2 + q_3)\theta_3 - g \sin(q_2 + q_3)\theta_4
\end{align}
CHAPTER 1. MANIPULATOR DESCRIPTION

Property 2:

An important property of manipulators is that the dynamics, except for complex friction models, can be written linearly in the parameters. This is an important property which is exploited for the derivation of adaptive controllers. The dynamics of equation (1.4) can therefore be written in the form:

\[ Y\theta + f_{fric} = \tau \]  

(1.25)

Where the vector \( \theta = [\theta_1, \theta_2, \ldots, \theta_k]^T \) contains the unknown parameters.

\( Y \) represents the so-called regressor matrix. A symbolic expression of this regressor is given in appendix B.

The smallest set of parameters, in which the rigid-body dynamics can be written linearly, is called the Base Parameter Set (BPS). The values for this base parameter set were identified by a least-squares approach during previous work [7]. During simulations, these identified values will be used to model the manipulator in order to test and tune the controller for implementation on the real system. During experiments, the values are used to check the results of the adaptation algorithm.

In this report the BPS is defined as the vector of parameters \( \theta \) which is needed to describe the dynamics of equation (1.4) except for friction. The parameters needed to describe the friction are not seen as Base Parameters in this report, and are stored in the vector \( \psi \). This separation between friction and other dynamics is made because it is likely that the friction modelling will be changed in the future because there exist many ways of approximating it.

In order to simulate the robot more realistically, it is chosen to use a neural network approximation to model the friction. This model also describes the Stribeck effect which is not linear in the parameters. In this way, it can be simulated how adaptation algorithms deal with dynamics which can not be fit exactly on the model.

The neural-friction model is chosen as:

\[ f_{fric} = \sum_{k=1}^{3} f_k(1 - \frac{2}{e^{2w_k q_i} + 1}) + b_i q_i \]  

(1.26)

The parameters in this models are also obtained by least-squares estimation. The numerical values used during simulations are:

\[
\begin{align*}
  f_1 &= [1.179 \ 1.285 \ 1.078]^T \\
  f_2 &= [0.650 \ 1.177 \ 1.152]^T \\
  f_3 &= [1.023 \ -0.010 \ 1.152]^T \\
  b &= [0.321 \ 0.577 \ 0.486]^T \\
  w_1 &= [100.29 \ 9.876 \ 1.052]^T \\
  w_2 &= [104.72 \ 19.40 \ -0.214]^T \\
  w_3 &= [11.942 \ -2.922 \ 0.321]^T
\end{align*}
\]

As already stated, the adaptation algorithm can only deal with parameters which occur linear in the dynamics. A approximation for friction which is linear in it’s parameters, can
be obtained by combining viscous and Coulomb friction models.

\[ f_{\text{fric}} \approx S\psi_{\text{Coulomb}} + Q\psi_{\text{vis}} \]  \hspace{1cm} (1.27)

The vectors \( \psi_{\text{Coulomb}} \) and \( \psi_{\text{vis}} \) contain respectively the unknown Coulomb and viscous friction constants.

\( S \) and \( Q \) are defined as:

\[
S = \begin{bmatrix}
\text{sign}(\dot{q}_1) & 0 & 0 \\
0 & \text{sign}(\dot{q}_2) & 0 \\
0 & 0 & \text{sign}(\dot{q}_3)
\end{bmatrix}
\quad Q = \begin{bmatrix}
\dot{q}_1 & 0 & 0 \\
0 & \dot{q}_2 & 0 \\
0 & 0 & \dot{q}_3
\end{bmatrix}
\]  \hspace{1cm} (1.28)

Together with equation (1.25) and equation (1.27), the following expression approximates the dynamic behavior of the system:

\[
M(q)\ddot{q} + C(q, \dot{q}_a)\ddot{q}_b + G(q) + S\psi_{\text{Coulomb}} + Q\psi_{\text{vis}} = \tau
\]

\[
Y(q, \dot{q}_a, \dot{q}_b, \dot{q})\theta + X(\dot{q})\psi = \tau
\]  \hspace{1cm} (1.29) \hspace{1cm} (1.30)

The regressor matrix \( X(\dot{q}) \), used for friction, is defined as:

\[
X(\dot{q}) = \begin{bmatrix}
\text{sign}(\dot{q}_1) & 0 & 0 & \dot{q}_1 & 0 & 0 \\
0 & \text{sign}(\dot{q}_2) & 0 & 0 & \dot{q}_2 & 0 \\
0 & 0 & \text{sign}(\dot{q}_3) & 0 & 0 & \dot{q}_3
\end{bmatrix}
\]  \hspace{1cm} (1.31)

and

\[
\psi = \begin{bmatrix}
\psi_{\text{Coulomb}} \\
\psi_{\text{vis}}
\end{bmatrix}
\]  \hspace{1cm} (1.32)

The notation \( \dot{q}_a \) and \( \dot{q}_b \) in equation (1.29) are introduced to distinguish the \( \dot{q} \)-term in the matrix \( C \) and \( \dot{q} \)-term multiplied with the matrix \( C \). From equation (1.9) it can be seen that \( \dot{q}_a \neq \dot{q}_b \), and can not be exchanged arbitrarily.
Chapter 2

Feedback-linearization

The basic concept to control a mechanical system is by PD-control, but using a PD-controller on a highly non-linear system is not preferable because of the hard tuning for performance. The most common way to deal with these non-linearities is by feedback-linearization. By feedback-linearization the non-linear system is transformed into a linear decoupled system by cancelling the non-linearities via the control law. By applying loop-shaping techniques on this "new" linear system, PD-gains can be tuned much easier than in the non-linear case to get the desired robustness and performance.

When it is assumed that the parameters are exactly known, feedback linearization can be obtained by the following control-law:

\[ \tau = M(q)a + C(q, \dot{q})\dot{q} + G(q) \quad (2.1) \]

\[ a = \ddot{q}_d - K_d \dot{e} - K_p e \quad (2.2) \]

where:
\[ q_d \] is the vector with the desired acceleration in the joints.
\[ e \] is defined as \[ e = q - q_d \]

The first equation (2.1) cancels the non-linearities and couplings so that the system is feedback-linearized and decoupled. Together with the dynamic model of equation (1.4) and the assumption that \( \tau_{fri} = 0 \), it can be derived that the input-output behaviour becomes:

\[ \ddot{q} = a. \]

The second equation (2.2) makes that this unstable system of pure integrators is stabilized by adding a PD-action. The following error-equation can be derived for the closed-loop system:

\[ \ddot{e} + K_d \dot{e} + K_p e = 0 \quad (2.3) \]

Which is stable if the matrices \( K_p \) and \( K_d \) are positive definite. The dimensionless damping \( \xi \) and the natural eigenfrequency \( \omega_n \) of the error dynamics of each degree of freedom, can be obtained using the characteristics of a second-order differential equation.

\[ \ddot{e} + 2\xi \omega_n \dot{e} + \omega_n^2 e = 0 \quad (2.4) \]
2.1 Adaptive case

As mentioned before, the parameters used in the controller are not exactly known so estimates has to be used. The use of estimates is represented by the \( \hat{\theta} \) symbol. The matrices \( \hat{M}, \hat{C} \) and \( \hat{G} \) has therefore the same functionality as \( M, C \) and \( G \) but estimated parameters \( \hat{\theta} = \hat{\theta}_1, \ldots, \hat{\theta}_i \) are used.

The following adaptive controller using parameter estimation, is proposed by [2] Craig (1986):

\[
\tau = \hat{M}(q)a + \hat{C}(q,q)\dot{q} + \hat{G}(q) \\
a = \ddot{q}_d - K_v\dot{e} - K_p\varepsilon
\]  

(2.5)  

(2.6)

Using the regressor representation of equation (1.30), the error-dynamics can be derived for the case of parameter uncertainties.

\[ \ddot{e} + K_v\dot{e} + K_p\varepsilon = \hat{M}^{-1}Y\hat{\Theta} =: \Phi\hat{\Theta} \]  

(2.7)

Where the tilde symbol is defined as the difference between the estimation and the real value, defined as: \( \tilde{\theta} = \hat{\theta} - \theta \)

The closed loop error-dynamics can be written as a linear system:

\[ \dot{x} = Ax + B\Phi\hat{\Theta} \]  

(2.8)

where:

\[ A = \begin{bmatrix} 0 & I \\ -K_p & -K_d \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ I \end{bmatrix} \quad \text{and} \quad x = \begin{bmatrix} e \\ \dot{e} \end{bmatrix} \]

This expression for \( \dot{x} \) is needed in the Lyapunov stability criterion.

Choose Lyapunov function:

\[ V = x^TPx + \tilde{\Theta}^T\Gamma\tilde{\Theta} \]  

(2.9)

\[ \dot{V} = x^TP(Ax + B\Phi\tilde{\Theta}) + (x^TA^T + \tilde{\Theta}^T\Phi^T B^T)Px + 2\tilde{\Theta}^T\Gamma\tilde{\Theta} \]

\[ = x^TPAx + x^TA^TPx + x^TPB\Phi\tilde{\Theta} + \tilde{\Theta}^T\Phi^TB^TPx + 2\tilde{\Theta}^T\Gamma\tilde{\Theta} \]

\[ = x^T(PA + A^TP)x + 2\tilde{\Theta}^T[\Phi^TB^TPx + \Gamma\tilde{\Theta}] \]  

(2.10)

When \( P \) is chosen as a solution to the Lyapunov-equation:

\[ A^TP + AP + Q = 0 \]  

(2.11)

\( \dot{V} \) can be written as:

\[ \dot{V} = -x^TQx + 2\tilde{\Theta}^T[\Phi^TB^TPx + \Gamma\tilde{\Theta}] \]  

(2.12)

If we choose the following update law:

\[ \dot{\hat{\Theta}} = -\Gamma^{-1}\Phi^TB^TPx \]  

(2.13)
where $\Gamma = \Gamma^T > 0$, the following expression is found for $V$:

$$V = -x^T Q x$$

(2.14)

The Lyapunov function is asymptotically stable if the matrix $Q$, as defined in the Lyapunov equation (2.11), is positive definite. From the definition of the Lyapunov-function (2.9), it is proven that $\dot{\theta}$ and $x$ goes to zero for $t \to 0$. By definition, this also means that $e$ and $\dot{e}$ goes to zero. The error-dynamics of the closed-loop system, equation (2.7), shows that also $\ddot{e}$ goes to zero. This completes the prove of the adaptive controller as proposed by Craig [2].

### 2.1.1 Comments on adaptive feedback linearisation

Because the derivative of the Lyapunov function (2.14) and $\dot{\theta}$ (2.13) is finite, it can be seen that adaptation of the parameters needs time. Only if the dynamics of the changing parameters is much slower than the dynamics of the parameter adaptation, proper parameter adaptation will be achieved.

From equation (2.13) it can be seen that if $\Phi$ contains much zeros, proper parameter convergence can not be achieved. This leads to the so called persistently exiting condition. If the system is not excited the regressor will contain zeros so some $\theta_i = 0$. But even in this case, $x$ will go to zero as can be seen from equation (2.14) and the definition of the Lyapunov function (2.9). The resulting parameter estimations are a local fit which is further improved when the system is persistently exited again.

As can be seen from equation (2.7) and (2.13), the inverse of the inertia matrix is needed for updating $\theta$. Calculation of an inverse of a matrix needs much computation effort and because this has to be found for every new sample due to it dependency on $q$, this is not desirable. Another problem arises when calculating the controller output from equation (2.5). Position, velocity and acceleration measurements are needed, although most manipulators only measure the position so acceleration and velocities has to be estimated by for example a Kalman filter. These two drawback can be evaded but this leads to more complicated algorithms [2]. Therefore it seems to be more attractive to choose an algorithm based on the passivity concept for implementation.

In this passivity approach there is no need for the inverse of the inertia matrix or acceleration measurements. A drawback is that the closed-loop system is not feedback linearized anymore. This makes the tuning of the PD-gains harder because the system behaviour is still dependent from the state of the manipulator and is therefore highly non-linear.
Chapter 3

Passivity based controller

The derivations in this chapter and the next will be presented extensively. It is chosen to work out these derivations in more details than is done in the articles [2], [3] and [4] for the following reasons.

Adaptive control is hard to understand without the mathematical derivations. In these derivations, often an explanation can be found for phenomena which can not be explained intuitively. Showing every step of the theory makes it easier to understand and to apply the theory in practice.

Because the regressor can be defined in different ways, it is more important to be familiar with basic procedure of deriving a passive adaptive controller instead of concentrating just on the stability proof of one control-law. During small steps, needed to prove stability, tricks are shown which can also be used for stability proofs of adaptive controllers with an other choice of the regressor.

In this chapter the passivity based computed torque controller will be discussed as described by Berghuis [5], but the derivation will be more detailed. Using this controller will not lead to a system which is feedback-linearized, but the advance is that the adaptation law is less complicated because the natural passivity structure of the manipulator is used. In this way there is no need for the inverse of the inertia matrix or measurement of $\dot{q}$ as mentioned in the previous chapter. The basic idea is that by adding a passive loop to the system, the system becomes stable to the input/output of that loop. As can be seen from figure 3.1, the adaptation-law can be seen as a kind of feedback loop because the error influences the computed-torque via the estimated parameters.

By reshaping the natural energy of a manipulator, the energy-minimum of the closed-loop system can be shifted to a desired position and speed. The first useful adaptive controller for tracking purpose, based on this passivity concept, was proposed by Slotine and Li [4]. With their innovative idea to use some kind of sliding variable it can be proven that not only the derivative of the tracking-error but also the tracking-error itself converges to zero.

For simplicity, first the basic idea of passivity based computed-torque will be discussed without adaptation. The passivity concept will be further worked out for the adaptive case in chapter 4. In this chapter first some basic properties are shown, then the simple case for positioning purpose is derived. After that, a passivity controller for tracking purposes will be proposed and stability will be proven.
CHAPTER 3. PASSIVITY BASED CONTROLLER

Figure 3.1: Feed-back loops of adaptive controller
3.1 Passivity based regulators

A common way to derive the dynamics of the system is by the Euler-Lagrange equation of motion.

\[
\frac{d}{dt} \left( \frac{\partial L(q,\dot{q})}{\partial \dot{q}} \right) - \frac{\partial L(q,\dot{q})}{\partial q} = \tau
\]  

(3.1)

Where:

- the Lagrangian \( L \) is defined as: \( L = T_\infty - P \)
- \( P(q) \) represents the potential energy.
- \( T_\infty(q,\dot{q}) \) the kinetic (co)energy defined as: \( \frac{1}{2} \dot{q}^T M(q) \dot{q} \)

With the Legendre transformation the Hamiltonian of the system can be obtained.

\[
H(q,p) = p^T(q) \dot{q} - L(q,\dot{q})
\]  

(3.2)

\( p \) is defined as the generalized momentum: \( p = \frac{\partial L}{\partial \dot{q}} \)

For the revolute type of manipulators \( p \) is defined as: \( p = M(q) \dot{q} \). With definition of \( L \) and \( p \), the expression for the Hamiltonian can be written as:

\[
H(q,p) = \frac{1}{2} \dot{q}^T M(q) \dot{q} + P(q)
\]  

(3.3)

\[
= \frac{1}{2} p^T M^{-1}(q) p + P(q)
\]  

(3.4)

where \( T(q,p) \) represents the kinetic energy and \( P(q) \) the potential energy.

Equation (3.1) can now also be written as Hamiltonian equations of motion:

\[
\dot{q} = \frac{\partial H(q,p)}{\partial p}
\]  

(3.5)

\[
\dot{p} = - \frac{\partial H(q,p)}{\partial q} + \tau
\]  

(3.6)

With equation (3.4) inserted in equation (3.5) and (3.6) we obtain:

\[
\dot{q} = \frac{\partial T(q,p)}{\partial p}
\]  

(3.7)

\[
\dot{p} = - \frac{\partial T(q,p)}{\partial q} - \frac{\partial P(q)}{\partial q} + \tau
\]  

(3.8)

Now \( \tau \) can be used to reshape the potential energy. Choose:

\[
\tau = \frac{\partial P(q)}{\partial q} - \frac{\partial P_0(q)}{\partial q} + v
\]  

(3.9)

When we insert equation (3.9) into (3.8), it can be seen the Hamiltonian has a new minimum in the potential energy at \( P_0 = 0 \).

\[
H_0(q,p) = T(q,p) + P_0(q)
\]  

(3.10)
CHAPTER 3. PASSIVITY BASED CONTROLLER

For stability reasons it has to hold that: \( \dot{H}_0(q, p) \leq 0 \) which means that the manipulator is moving to the desired point where \( P_0 \) equals zero. With equations (3.7) and (3.8), the expression for \( \dot{H}_0(q, p) \) can be derived:

\[
\dot{H}_0(q, p) = \frac{\partial T}{\partial p} \frac{dp}{dt} + \frac{\partial T}{\partial q} \frac{dq}{dt} + \frac{\partial P_0}{\partial q} \frac{dq}{dt}
\]

\[
= \dot{q}^T v
\]

If \( v \) and \( P_0 \) are chosen as:

\[
v = -K_d \dot{q}
\]

\[
P_0(q) = \frac{1}{2} \varepsilon^T K_p \varepsilon
\]

\( \tau \) can be calculated from equation (3.9):

\[
\tau = G(q) - K_p \varepsilon - K_d \dot{q}
\]

Choosing \( K_d \) positive-definite gives \( \varepsilon = 0 \) for \( t \to \infty \) as can be seen from the new definition of the Hamiltonian \( H_0 \) and its derivative. Via the Lyapunov stability theorem, it is proved that this controller leads to a stable closed-loop system with an new minimum in the potential energy as defined in equation (3.10).

Remark

In the area of adaptive control, stability of the closed-loop system is often proved with the passivity-integral. It is proved that a system is asymptotically stable from input \( u \) to output \( y \) if:

\[
<y, u> = \int_0^T y^T(t) u(t) \, dt \geq -\beta \tag{3.16}
\]

with \( \beta \geq 0 \)

It can be seen that this stability-proof is close to the Lyapunov prove above. We can get equation (3.16) by integration of equation (3.12).

\[
\dot{H} = \dot{q}^T v \tag{3.17}
\]

\[
H \geq 0 \tag{3.18}
\]

so:

\[
\int_0^T \dot{q}^T v \, dt + c \geq 0 \tag{3.19}
\]

with:

\[
c = H(t_0) = \beta \tag{3.20}
\]

\[
\int_0^T \dot{q}^T v \, dt + \beta \geq 0 \tag{3.21}
\]

With \( \beta = H(t_0) \), it is proven that \( \beta \geq 0 \). It can therefore be concluded that the use of this passivity integral for stability proofs, gives the same results as the Lyapunov stability proof. During further derivations, stability will be proved with the Lyapunov stability theorem.
3.2 Passivity based tracking controllers

For tracking purpose the energy minimum of the system should be shifted from \((q, \dot{q}) = (0, 0)\) to \((e, \dot{e}) = (0, 0)\). Now we are also dealing with \(\dot{e}\) so it is obvious that not only the potential energy but also the kinetic energy of the closed loop system has to be changed. This is for instance done by the controller proposed by Paden and Panja (1988) [2].

\[
\tau = M(q)\ddot{q}_d + C(q, \dot{q})\dot{q}_d + G(q) - K_p e + v \tag{3.22}
\]

Together with equation (3.4) the Hamiltonian of the closed-loop system becomes:

\[
H(e, \dot{e}) = \frac{1}{2} e^T M(q) \dot{e} + \frac{1}{2} e^T K_p e \tag{3.23}
\]

Again with equation (3.7) and (3.8) the expression can be obtained for \(\dot{H}\):

\[
\dot{H}(e, \dot{e}) = \dot{e}^T v \tag{3.24}
\]

For damping purpose \(v\) is chosen as:

\[
v = -K_d \dot{e} \tag{3.25}
\]

This assures that \(\dot{H} \leq 0\) for \(H > 0\).

As can be seen from equation (3.24) and (3.25), a difficulty arises when \(\dot{e} = 0\). When \(\dot{e} = 0\), \(\dot{H} = 0\). In this case it can not be proven that \(e\) converges asymptotically to 0. A solution for this problem was proposed by Slotine and Li.

3.2.1 Slotine and Li

With the controller of Paden and Panja, it appeared that asymptotic stability of the tracking error can not be proven. This is due to the fact that if \(\dot{e} = 0\), the derivative of the Hamiltonian (3.24) becomes zero. The solution to this problem is found by introducing a new variable based on sliding-mode techniques \(^1\). When this sliding variable converges to zero the remaining error-dynamics are asymptotically stable. (So there is no need to explicitly introduce some \(K_p\) gain.) In essence this new variable introduces a integration-action via the reference which can be seen from equation (3.26) and (3.27).

The proposed tracking controller is:

\[
\tau = M(q)\ddot{q}_r + C(q, \dot{q})\dot{q}_r + G(q) + v \tag{3.26}
\]

Where

\[
q_r = q_d - \lambda \int_0^t e \, dt \tag{3.27}
\]

\[
\dot{q}_r = \dot{q}_d - \lambda e \tag{3.28}
\]

where \(\lambda\) is defined as a constant vector.

\(^1\)The term sliding-mode is commonly used for other techniques but according to the terminology of Slotine and Li [4], the terms sliding surface and sliding variable will be used in this context.
CHAPTER 3. PASSIVITY BASED CONTROLLER

Combining equation (1.4) and (3.26), the closed-loop dynamics can be derived. Due to the definition of \( q_r \), the sliding variable \( s \), appears in the closed-loop dynamics as:

\[
M(q)\dot{s} + C(q, \dot{q})s = v
\]  
(3.29)

Where \( s \) is defined as:

\[
s = \dot{q} - q_r = \dot{e} + \Lambda e
\]  
(3.30)

Because there are no gravity terms in equation (3.29), there is no potential-energy term in the Hamiltonian of the closed-loop system.

\[
H(s) = \frac{1}{2} s^T M s
\]  
(3.31)

With equation (3.7) and (3.8) the time derivative of the Hamiltonian becomes.

\[
\dot{H}(s) = s^T v
\]  
(3.32)

Choosing:

\[
v = -K_ds
\]  
(3.33)

proves that \( \dot{H} < 0 \) for \( H > 0 \). It is now proved that the closed-loop system asymptotically converges to \( s = 0 \). This gives the error-dynamics:

\[
\dot{e} = -\Lambda e
\]  
(3.34)

When \( \Lambda \) is chosen larger than zero, \( (e, \dot{e}) \) converges to \( (0,0) \). This proves that the controller of Slotine and Li is indeed able to track a reference trajectory even when \( \dot{e} = 0 \).

In the recent years many new passivity based adaptive controller are proposed. Most of them are variations on the proposed controllers above but with improvements with respect to the influence of measurement noise, transient behavior or preventing excitation of unmodelled dynamics. Together with the controller of Slotine and Li, one of these improved controllers as proposed by Berghuis, Ortega and Nijmeijer, will be used for simulations and experiments.
Chapter 4

Adaptive case

An advantage of a passivity based controller is that it can be easily extended to an adaptive controller. The coupling between the error and the computed-torque through the adaptation-law, can be seen as an extra feedback loop which results in the system of figure 3.1. When the adaptation-loop remains passive from \( e \) to \( \dot{\theta} \), it is proven that \( \dot{\theta} \) remains bounded and converge to zero for \( t \to \infty \). Again the Lyapunov stability theorem is used to prove convergence of the tracking-error \((e, \dot{e})\) and parameters estimation.

Because the derivations in this chapter are more complex, the friction is neglected for the moment. In a similar way, stability can be proved for the case with friction.

4.1 Slotine and Li

This section expands the controller of section 3.2.1 for parameter uncertainties. The adaptation law can be derived via the Lyapunov stability theorem. The estimation error \( \ddot{\theta} \) is now also taken into account so the following Lyapunov function is chosen:

\[
V(s, \dot{\theta}) = \frac{1}{2}s^T M(q)s + \frac{1}{2}\dot{\theta}^T \Gamma \dot{\theta}
\]

(4.1)

\[
\dot{V}(s, \dot{\theta}) = s^T M(q)s + \frac{1}{2}\dot{s}^T \dot{M}(q)s + \dot{\theta}^T \Gamma \dot{\theta}
\]

(4.2)

Because the exact values for \( \theta \) are not known, the parameter values in the control-law of Slotine and Li, derived in equation (3.26), are replaced by estimated values \( \dot{\theta} \). The matrices \( M, C \) and \( G \) of equation (3.26) are replaced by \( \dot{M}, \dot{C} \) and \( \dot{G} \).

\[
\tau = \dot{M}\dot{q} + \dot{C}q + \dot{G} + v
\]

(4.3)

Combining equation (1.4) and (4.3) gives the dynamics of the closed-loop system with parameter uncertainty. A new symbol \( \Delta \) is introduced, which is defined as: \( \Delta = \ddot{\cdot} - \cdot \).

\[
M(q)(\ddot{q} - \dot{q}_r) + C(q, \dot{q})(\dot{q} - \dot{q}_r) = \Delta M(q)\dot{q}_r + \Delta C(q, \dot{q})\dot{q}_r + \Delta G(q) + v
\]

(4.4)

\[
M\ddot{s} + Cs = Y(q, \dot{q}, \dot{q}_r, \dot{q}_r)\ddot{\theta} + v
\]

(4.5)
CHAPTER 4. ADAPTIVE CASE

With this equation, the equation for $\dot{V}$ can be further worked out.

$$\dot{V}(s, \dot{\theta}) = s^T(Y \dot{\theta} - Cs + v + \frac{1}{2} \dot{M} s) + \dot{\theta}^T \dot{\theta}$$

$$= s^T(Y \dot{\theta} + v) + \dot{\theta}^T \dot{\theta}$$

During this derivation we made use of the following property: $s^T(M - 2C)s = 0$.

It is guaranteed that $\dot{V} \leq 0$ for $V \geq 0$ if we choose:

$$v = -K_d s$$

and:

$$s^T Y \dot{\theta} = -\dot{\theta}^T \dot{\theta}$$

This gives the expression for the adaptation- and control-law:

$$\dot{\theta} = -\Gamma^{-1} Y s$$

$$\tau = \dot{M} \dot{q}_r + \dot{C} \dot{q}_r + \dot{G} - K_d s$$

With these laws it is assured that the Lyapunov function $V$ becomes zero so also the sliding variable $s$ and $\dot{\theta}$ converges to zero.

4.1.1 Comments on the controller of Slotine and Li

During experiments it appeared that the closed-loop system could become unstable due to, at first sight, unknown reasons. An explanation for this is found in the equation of the computed-torque controller of equation (4.3) and the definition of $q_r = \dot{q}_d - \lambda(q - q_d)$. When the control-law is written out, it can be seen that additional PD-gains appear.

$$\tau = \dot{M}(q)(\dot{q}_d - \lambda \dot{e}) + \dot{C}(q, \dot{q})(\dot{q}_d - \lambda \dot{e}) + \dot{G}(q) - K_d \dot{e} - K_p e$$

$$= \dot{M}(q)\dot{q}_d + \dot{C}(q, \dot{q})\dot{q}_d + \dot{G}(q) - (\dot{M}(q)\lambda + K_d) \dot{e} - (\dot{C} \lambda + K_p) e$$

with: $K_p = K_d \lambda e$.

As can be seen, extra PD gains are added by the computed-torque part of the controller. These PD-gains depend on $\dot{\theta}$ via $\dot{M}$ and $\dot{C}$, and can therefore cause instability. Disturbances cause errors which leads to variation in the estimation of the parameters. Due to these disturbances, $\dot{\theta}$ could be changed in such a way that the feedback gains of the computed-torque part of the controller makes the closed-loop system unstable. Therefore, much attention has to be paid to the influence of variation in $\dot{\theta}$ on the additional $K_p$ and $K_d$ gains of the computed torque controller. This can be done by choosing the PD-gains more conservative during the tuning of the PD-part of the controller.

A problem of the controller of Slotine and Li is that it suffers from parameter-drift due to noise in the measurement of the velocity. From equation (4.10) and the definition of the regressor (4.5), it can be seen that there appears a quadratic term with $\dot{q}$ in parameter-update law. Due to this quadratic term, the measurement-noise is not middled out. A solution for this problem is to substitute the term $\dot{q}$ in the regressor by $\dot{q}_d$. The stability of this new controller is proved in the next section.
CHAPTER 4. ADAPTIVE CASE

4.2 Berghuis, Ortega and Nijmeijer

As mentioned in the previous subsection, the adaptive controller proposed by Slotine and Li suffers from parameter-drift due to noisy velocity measurements/estimations. New controllers are proposed which solve this problem by replacing one $\dot{q}$ in the quadratic term by $\dot{q}_d$. This is for example done by Berghuis, Ortega and Nijmeijer [3]. Stability of this controller will be proved in this section.

The proposed controller is:

$$\tau = \dot{M}(q)\dot{q}_d + \hat{C}(q, \dot{q} - \lambda e)\dot{q}_d + \hat{G}(q) - K_d\dot{e} - K_pe$$ (4.13)

Where $\lambda$ is defined as:

$$\lambda = \frac{\lambda_0}{1 + \|e\|}$$ (4.14)

Similar to the derivation of Slotine and Li, the closed-loop dynamics with unknown parameters can be derived using equation (1.29) and (1.30):

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \dot{M}(q)\dot{q}_d + \hat{C}(q, \dot{q})\dot{q}_d + \lambda\hat{C}(q, \dot{e})\dot{q}_d + \hat{G}(q) - K_d\dot{e} - K_pe$$ (4.15)

$$M(q)\ddot{e} + C(q, \dot{q})\dot{e} + \lambda C(q, \dot{e})\dot{e}_d = \Delta M(q)\ddot{q}_d + \Delta C(q, \dot{q})\dot{q}_d + \Delta C(q, \dot{q} - \lambda e)\dot{q}_d - K_d\dot{e} - K_pe$$ (4.16)

$$= Y(q, \dot{q}, \dot{q}_d, \ddot{q}_d)\ddot{\theta} - K_d\dot{e} - K_pe$$ (4.17)

During this derivation we use the properties:

$$C(q, x + \alpha y) = C(q, x) + \alpha C(q, y)$$ (4.18)

$$C(q, x)y = C(q, y)x$$ (4.19)

The Lyapnov function is defined as.

$$V = \frac{1}{2}s^T M(q)s + \frac{1}{2}\dot{e}T\dot{\theta} + \frac{1}{2}e^TK_pe$$ (4.20)

$$\dot{V} = s^T M(q)\dot{s} + \frac{1}{2}s^T\dot{M}(q)s + \dot{\theta}T\dot{\theta} + e^TK_pe$$ (4.21)

The extra term $\frac{1}{2}e^TK_pe$ is not strictly needed to prove asymptotic stability of the system because the error already appears in the definition of $s$. But to get more freedom in tuning the controller, the $K_pe$ term is added to the Lyapnov function and consequently to the controller.

The expression of equation (4.21) can be further worked out. With the definition of $s = \dot{e} + \lambda e$, the following expression is found for $M(q)\dot{s}$.

$$M(q)\dot{s} = M(q)\ddot{e} + M(q)\lambda e + M(q)\dot{\lambda}e$$ (4.22)

With the equation of the closed-loop system (4.17), the term $M(q)\ddot{e}$ can be substituted. This gives:

$$\dot{V} = s^T[Y\ddot{\theta} - C(q, \dot{q})\dot{e} - \lambda C(q, e)q_d - K_d\dot{e} - K_pe + \lambda M(q)e + \dot{\lambda}M(q)e + \frac{1}{2}\dot{M}(q)s] + \dot{\theta}T\dot{\theta} + e^TK_pe$$ (4.23)
CHAPTER 4. ADAPTIVE CASE

Notice that the matrix $C$ can depend on $(q, \dot{q})$ or on $(q, e)$.

From the skew-symmetric property, it can be seen that:

$$s^T \frac{1}{2} \dot{M}(q)s = s^T C(q, \dot{q})s$$  \hspace{1cm} (4.24)

The derivative of the Lyapunov function becomes:

$$\dot{V} = s^T \left[ Y \ddot{\theta} - \lambda C(q, e) \dot{q}_d - K_d \dot{e} - K_p e + \dot{\lambda} M(q)e + \lambda M(q) \dot{e} + \dot{\lambda} C(q, \dot{q})e \right]$$

$$+ \dot{\theta} \Gamma \dot{\theta} + e^T K_p \dot{e} + s^T K_p e$$  \hspace{1cm} (4.25)

Some terms can be rewritten:

$$e^T K_p \dot{e} + s^T K_p e = \lambda e^T K_p e$$  \hspace{1cm} (4.26)

and

$$\lambda C(q, \dot{q})e - \lambda C(q, e) \dot{q}_d = \lambda s^T C(q, \dot{e})e$$  \hspace{1cm} (4.27)

so $\dot{V}$ becomes:

$$\dot{V} = s^T \left[ Y \ddot{\theta} - K_d \dot{e} - \lambda s^T C(q, e)e + \dot{\lambda} M(q)e + \lambda M(q) \dot{e} \right] + \dot{\theta} \Gamma \dot{\theta} - \lambda e^T K_p e$$  \hspace{1cm} (4.28)

For simplicity a new variable will be introduced:

$$s_1 = \dot{e} + \frac{\lambda}{2} e$$  \hspace{1cm} (4.29)

We can rewrite the equation (4.28) as:

$$\dot{V} = -s_1^T \left( K_d - \lambda M(q) \right) s_1$$

$$+ \left( \frac{\lambda}{2} e \right)^T \left[ 4 \lambda^{-1} K_p - \left( K_d - \lambda M(q) \right) \right] \left( \frac{\lambda}{2} e \right)$$

$$+ \dot{\lambda} s^T M(q)e + \lambda s^T C(q, e)e + s^T Y \ddot{\theta} + \dot{\theta} \Gamma \dot{\theta}$$

It can be proved that:

$$\dot{\lambda} s^T M(q)e \leq 2 \lambda_0 M_m (\|s_1\|^2 + \|\frac{\lambda}{2} e\|^2)$$  \hspace{1cm} (4.31)

$$\lambda s^T C(q, \dot{e})e \leq 2 \lambda_0 C_m (\|s_1\|^2 + \|\frac{\lambda}{2} e\|^2)$$  \hspace{1cm} (4.32)

So equation (4.30) becomes:

$$\dot{V} \leq -k_1 \|s_1\|^2 - k_2 \|\frac{\lambda}{2} e\|^2 + s^T Y \ddot{\theta} + \dot{\theta} \Gamma \dot{\theta}$$  \hspace{1cm} (4.33)

where

$$k_1 = K_{d,m} - 3 \lambda_0 M_m - 2 \lambda_0 C_M$$  \hspace{1cm} (4.34)

$$k_2 = \frac{4 K_{p,m}}{\lambda_0} - K_{d,M} - 2 \lambda_0 M_M - 2 \lambda_0 C_M$$  \hspace{1cm} (4.35)
CHAPTER 4. ADAPTIVE CASE

\[ X_m = \sigma_{\text{min}}(X), \quad X_M = \sigma_{\text{max}}(X) \]  \hspace{1cm} (4.36)

with \( \sigma_{\text{max}} \) and \( \sigma_{\text{min}} \) the maximum and minimum singular values of the dummy matrix \( X \).

\( s, e \) and \( \Theta \) converges to zero if the Lyapunov function goes to zero as can be seen from equation (4.20). From equation (4.33) it can be seen that \( \dot{V} \leq 0 \) for \( V > 0 \) when:

\[ s^T Y \dot{\theta} + \tilde{\theta} \Gamma \dot{\theta} = 0 \]  \hspace{1cm} (4.37)

and:

\[ \lambda_0 < \min \left\{ \frac{K_{d,m}}{3M_M + 2C_M}, \frac{4K_{p,m}}{K_{d,M} + K_{d,m}} \right\} \]  \hspace{1cm} (4.38)

From equation (4.37) the parameter update-law can be derived:

\[ \dot{\theta} = -\Gamma^{-1} Y^T s \]  \hspace{1cm} (4.39)

where \( \Gamma \) contains the adaptation gains.

By the definition of the Lyapunov function it is now proved that \( \dot{\theta}, s \) and therefore \( \dot{e} \) and \( e \) converges to zero.

In that case it can be seen from the equation of the closed-loop system (4.16) and the equation below:

\[ \lambda C(q, e) q_d = \lambda C(q, \dot{q}_d) e \]  \hspace{1cm} (4.40)

that also \( \dot{e} \) goes to zero.

4.2.1 Comments on the controller of Berghuis, Ortega and Nijmeijer

The term \( q \) in the regressor is replaced by \( \dot{q}_d \). Therefore the problem of parameter-drift due to measurement noise is solved.

From the equation (4.13), it can be seen that the computed-torque part of the controller doesn’t introduce extra P\(D\) gains. This makes the closed-loop system more robust because disturbances can not influence the feed-back-gains via the estimated parameters anymore as is the case with the controller of Slotine and Li. This effect can be clearly recognized during the experiments as described in the next chapter.

From equation (4.33) it can be seen that the need for persistently exciting is only needed for parameters convergence. When the system is not excited, \( s \) still converges to zero. The values for \( \dot{\theta} \) only holds for that moment. When further exciting the system, \( \dot{\theta} \) further converges to zero.

For the determination of \( \lambda_0 \), the singular values are needed. But as described by Berghuis [5], this condition is sufficient but not necessary to have a stable system, so \( \lambda_0 \) is tuned emperical during experiments.
Chapter 5

Implementation

In chapter 4, stability of the controller of Slotine and Li and the controller of Ortega, Berhuis and Nijmeijer is proved. Robustness, performance and parameter convergence of these two controllers will be further investigated by simulations and experiments. Performance in this chapter will be defined as small position errors and not as the rate of parameter adaptation.

In the previous derivations of section 4.1 and 4.2, stability was guaranteed as long as the values for $\Gamma$, $\lambda$, $K_p$ and $K_d$ were positive definite. But during simulations and experiments it appeared that these controller parameters could not be chosen arbitrarily but has to lie within certain bounds.

5.1 Tuning of parameters

5.1.1 Choosing $K_p$ and $K_d$

The RRR-manipulator is approximated by a multi-body model which consists of pure inertia’s exposed to gravitation-forces and motor torques, there is no flexibility or damping modelled of the links. When we do not take into account friction for the moment, the system always has 180 degrees phase-lag. From the Nyquist stability criterium it can be seen that every $K_p$ and $K_d$ which is positive definite, leads to passing the critical point $-1$ at the right-hand side. Choosing a larger $K_d$ leads to extra phase-margin which makes the system more robust but also leads to a lower feed-back gain at low frequencies. Loop-shaping techniques can be used to tune the PD-gains around some linearized point of manipulator-state.

With the characteristics of a second-order differential equation, the eigenfrequency $\omega_n$ and dimensionless damping $\xi$ of the error-dynamics for each degree of freedom can be obtained by choosing a certain $K_p$ and $K_d$ and apply it to linearized dynamics around a desired point.

$$\ddot{e} + 2\xi\omega_n \dot{e} + \omega_n^2 e = 0$$ (5.1)

The real system contains friction and dynamics which are not represented in the model. Unmodelled dynamics are for example the flexibilities of the links and motor/amplifier-dynamics. Also phase-lag caused by computational time, can cause problems with respect to stability. This leads to bound for the choice of $K_p$ and $K_d$. A way to investigate these bounds, is by measuring Frequency Response Functions (FRF). Due to the nonlinearity of the system, this has to be done in different positions in the robot-space. Some states of the robot will appear to be critical with respect to stability of the closed-loop system. The FRF's
measured in these critical points are used for loop-shaping. It is chosen to take PD-gains which were obtained during previous research. Based on empirical experience, these values are changed during the experiments and simulations.

During initial parameter estimation of a manipulator, the adaptation of the parameters has priority above performance. In this case PD-gains can be chosen low which leads to poor performance. The resulting errors are big but contain much information about the low-frequent modelled system behaviour. Due to these low PD-gains, less high frequent dynamics will be excited so the noise to model-information ratio will be much higher. The price paid for this fast adaptation is bad performance which is not needed during identification.

Note that due to these larger errors and consequently a large $s$, the ideal adaptation gains has to be lower as in the case of small errors. Otherwise the derivatives of the estimated parameters will be high which causes large oscillations in the parameter estimations. This can be seen from equation (4.39).

5.1.2 Choice of $\Gamma$ and $\Gamma_{fric}$

The choice of the adaptation gain $\Gamma$ is again a trade-off between robustness and fast convergence. $\Gamma$ determines the rate of parameter adaptation. When this gain is chosen too high, small disturbances, which shortly occur, are gained to big parameter variations. These big parameter variations effect the computed-torque part of the controller in a negative way. So a big $\Gamma$ does not necessary leads to better performance, in the worst case it can even lead to instability of the adaptation-loop. This upper bound on $\Gamma$ can cause problems in cases where parameters vary significantly, think of picking up masses. Choosing $\Gamma$ to low, gives a slow adaptation of the parameters.

When there are few parameters, it is possible to tune $\Gamma$ by hand. To make this proces more easy, $\Gamma$ can chosen to be a diagonal matrix. In case of a diagonal $\Gamma$ matrix, it can be seen from equation (4.39) that the rate of change of $\theta_i$ due to a certain error, represented by $s_i$, is determined by element $\Gamma(i,i)$.

A more profound method is to use the information of the regressor during the trajectory in order to calculate $\Gamma$. This is based on the least-squares method and is also used in other identification techniques.

$$\Gamma^{-1} = \int_0^T Y^T Y \, dt \tag{5.2}$$

Because the reference trajectory is repeating every 10 seconds, the integration-time is also set to 10 seconds. In this way the information of the total trajectory is used to determine $\Gamma$. The integration of equation (5.2) is solved by Simulink. Because the choice of the integration time $T$ influences the scaling of $\Gamma$, the regressor matrix has to be scaled by a scalar in order to get proper convergence of $\dot{\theta}$.

5.1.3 Choice of $\lambda$ and $\lambda_0$

Contrary to the controller parameters above, the effect of $\lambda$ on the controller performance and adaptation can not be seen so easily in the resulting tracking-error. $\lambda$ determines the speed of error-dynamics when $s = 0$.

$$s = \dot{e} + \lambda e \tag{5.3}$$

when:

$$\begin{align*}
\text{when: } s &= 0 \\
\dot{e} &= -\lambda e \tag{5.4}
\end{align*}$$
When $\lambda$ is chosen too high, the error-dynamics can excite unmodelled dynamics. Choosing $\lambda$ too low, the error-dynamics are too slow to get small tracking-errors.

During the tuning of the controller, the gain-factor $\lambda$ or $\lambda_0$ is increased till a negative effect on the tracking-error and parameter-estimation is noticed. This means that the error-dynamics are as fast as possible without excitation of unmodelled dynamics. Also choosing a high $K_d$, which forces $s$ to zero (see damping of Lyapunov function), and $\Gamma$ as high as possible lead to fast error- and parameter-convergence.

5.2 Implementation of the regressor

As already mentioned in the description of the manipulator, an important property of manipulators is that the dynamics, with exception of complex friction models, are linear in the unknown parameters.

$$M(q)\ddot{q} + C(q, \dot{q}_s)\dot{q} + G(q) = Y(q, \dot{q}_s, \dot{\theta}, \ddot{\theta}) \theta$$  \hfill (5.5)

This property was used to derive and prove stability of the adaptive controllers in chapter 3 and 4.

As a consequence, we can only use a friction model which is also linear in its parameters.

$$S(\text{sign}(\dot{\psi}))\psi_{\text{Coulomb}} + Q(\dot{\psi})\psi_{\text{vis}} = X\psi$$  \hfill (5.6)

Where $S$ and $Q$ are defined as in equation (1.28).

The regressor of the total system is split in a part which represents the Base Parameter Set and a part which is used for the friction parameters. The reason that this is possible without harming the stability proofs lies in the fact that the regressor for the BPS is completely independent of the friction parameters. The regressor representing the total system has the following form:

$$Y\theta + X\psi = Z\Upsilon = \begin{bmatrix} Y & 0 \\ 0 & X \end{bmatrix} \Upsilon$$  \hfill (5.7)

where: $\Upsilon = \begin{bmatrix} \theta \\ \psi \end{bmatrix}$

It can be seen that due to the zeros in the matrix $Z$, the expressions for the parameter adaptation and the expression for the feed-forward can be split-up into representing the BPS and a part representing the friction without changing the adaptive controller.

The reason for splitting up the regressor in a BPS part and a friction part has several advantages:

- By separate regressors, it is more easy to check correctness of each regressor apart. During simulations the friction of the robot-model can be turned off. This enables us to check the regressor $Y$ without using $X$. Because the model used for the regressor $Y$ exactly fits the robot-model used in simulations, the error can become infinitely small. In this case, correctness of the regressor $Y$ is proven.

- It is possible to use different scalings for friction and for the BPS. It might be useful to change the adaptation gains for the BPS independent of the friction parameters. This could also be achieved by changing the total regressor matrix but it is more easy to scale the two regressor-matrices by two separate scalars.
Because friction is hard to model as a linear function, it is imaginable that the model for the friction will change in the future. This is easier when only a small regressor has to be changed instead of one big regressor containing all the dynamics.

The regressor for the BPS uses 16 unknown parameters so it is large and very complex. In order to reduce the possibility of making errors it is preferable to derive the regressor with the use of a symbolic mathematical program like the Matlab Symbolic Toolbox.

In order to make the regressor formulations compatible for other adaptation schemes, it is chosen to make the regressor as universal as possible. For universality, the regressor is written as function of six inputs. The six entries correspond to \( u, v, w, x, y, z \) in the equation below. From these six entries \( u, w, z \) are mostly the same. They all represent \( q \) in the matrices \( M(q) \), \( C(q, \dot{q}) \) and vector \( G(q) \).

\[
M(u)v + C(w, x)y + G(z) = Y(u, v, w, x, y, z)\theta
\]  
\hspace{1cm} (5.8)

The matrix \( M(q) \) was derived during previous work [1], the matrix \( C(q, \dot{q}) \) and the vector \( G(q) \) are derived from the vector \( h(q, \dot{q}) \) in chapter 1. When \( M(q) \), \( C(q, \dot{q}) \) and \( G(q) \) are known in symbolic form, it is easy to derive the regressor \( Y \). Substitute \( q, \dot{q} \) and \( \ddot{q} \) by \( u, v, w, x, y, z \) using a symbolic toolbox gives:

\[
\begin{align*}
M(q) & \rightarrow \ M(u) \\
C(q, \dot{q}) & \rightarrow \ C(w, x) \\
G(q) & \rightarrow \ G(z) \\
\dot{q} & \rightarrow \ y \\
\ddot{q} & \rightarrow \ v
\end{align*}
\]  
\hspace{1cm} (5.9) - (5.13)

We can now calculate \( Y(u, v, w, x, y, z) \) in an symbolic form using equation (5.8). \( Y \) can be calculated by using the property of linearity in the parameters. Using a symbolic toolbox, the regressor can be calculated by partial derivatives in the following way.

\[
Y(i, j) = \frac{\partial a_i}{\partial \theta_j}
\]  
\hspace{1cm} (5.14)

where the vector \( a \) is defined as:

\[
a = M(u)v + C(w, x)y + G(z)
\]  
\hspace{1cm} (5.15)

The derivation of the friction regressor is much more simple and can directly be found from equation (1.28).

\[
X = \begin{bmatrix} S & 0 \\ 0 & Q \end{bmatrix}
\]  
\hspace{1cm} (5.16)

5.3 Reference trajectory

As already mentioned during the derivation of the controller, the system needs to be persistently excited in order to get proper parameter convergence. As a rule of thumb from identification techniques on linear systems, it can be proposed that two unknown parameters can be fitted from the error information of one sinusoid at the input. Because were are dealing with a non-linear system this rule of thumb is not necessary because the non-linearities
generates new frequencies itself. Still a summation of different frequencies is used in order to get a good fit of the parameters over the whole frequency region.

It has to be prevented to use a high frequent components in reference signal because they excites high frequent unmodelled dynamics or causes actuator saturation due to high torques needed for high accelerations. This causes error-signals which can not be fit on the rigid-body model. It is chosen to take a reference trajectory which is a summation of sinusoids up to a frequency of ten Hertz. In cases where a high frequent reference is unavoidable, a second order low pass-filter can be used to filter out the high frequent components. The used reference trajectory is depicted in figure 5.1. Note that \( \dot{q}_d \) and \( \ddot{q}_d \) are also inputs for the controller, but are not depicted.

5.4 Simulations

Now the regressor is derived and the influence of the controller settings is discussed, the adaptation algorithm of Slotine and Li and the algorithm of Berghuis, Ortega and Nijmeijer can be implemented. These adaptation schemes are implemented in Matlab Simulink R12. An advance of this Matlab version with respect to previous versions is the ability to work with matrices. An overview of the Simulink scheme is depicted in figure 5.2. The implementation is split into blocks which represent a certain task/equation in the adaptive controller. The task which is fulfilled by a certain block will be shortly described together with the needed equations. Although most of these equations are already mentioned in the derivations of chapter 3 and 4, they will be repeated as a short summary.

- **Error signals**
  This block calculated the error signals which are needed in the regressor and the adaptation law.
Figure 5.2: Control scheme implemented in Simulink

The vector \( \lambda \) is chosen to be a scalar to simplify the implementation and tuning of the controller of Slotine and Li. In the controller of Berghuis, Ortega and Nijmeijer, \( \lambda \) is not defined as a constant scalar but as a function of the error:

\[
\lambda = \frac{\lambda_0}{1 + \|e\|}
\]

(5.20)

- **Regressor**
  This block calculates the regressor from the position-signals and error-signals, as defined in equation (5.18). In the controller of Slotine and Li, the regressors represent the dynamics:

\[
Y \theta = M \ddot{q}_r + C(q, \dot{q}) \dot{q}_r + G(q)
\]

(5.21)

and from the controller of Berghuis, Ortega and Nijmeijer it follows that:

\[
Y \theta = M(q) \ddot{q}_d + C(q, \dot{q} - \lambda e) \dot{q}_d + G(q)
\]

(5.22)

- **Computed-torque**
  The computed-torque part calculates the input needed to compensate for the modelled dynamics of the system. The computed-torque is based on the parameters which are...
estimated by the parameter update-law in the adaptation-block and the regressor which is also needed for the parameter adaptation.

\[ \tau = Y \theta + X \psi \]  \hspace{1cm} (5.23)

- **Adaptation**

This part performs the adaptation of the unknown parameters. As derived in chapter 4, it can be proved that the following parameter update-law leads to asymptotic convergence of \( e, \dot{e}, e, \) and \( \dot{\theta} \) to zero.

\[ \dot{\theta} = -\Gamma^{-1} Y^T s \]  \hspace{1cm} (5.24)

Note that proper parameter-convergence is only achieved under the condition of persistently exciting the system.

- **PD-gains**

Even when the parameters are estimated correctly, the computed-torque is only able to compensate for modelled dynamics. An extra PD-controller is used to reduce the errors caused by unmodelled dynamics. Remark that the controller of Slotine and Li has already a PD-action in itself as discussed in section 4.1.1, but these gains also depends on the choice of \( \lambda \) so to obtain more freedom in tuning the PD-gains, an extra PD-action is added in this controller.

In the case where the estimated parameters are not exactly the right ones, the PD-action must also compensate for modelled dynamics which are not totally cancelled by the computed torque. This can be seen from equation (4.5) and (4.17).

### 5.4.1 Simulations without friction

Because the derivation of the regressor \( Y \) is complex it will be necessary to check the correctness \( Y \). This is done by running the adaptation in simulation without friction using the manipulator model used by previous work.

Because of the absence of unmodelled dynamics, or noisy measurements, there exist a perfect match for the parameters \( (\dot{\theta} = \theta) \). Using this \( \dot{\theta} \) should perfectly compensate the dynamics of the system and lead to infinite small tracking errors.

For this simulation the adaptive controller of Slotine and Li is used with the following controller settings:

\[
K_p = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 50 \end{bmatrix}, \quad K_d = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]  \hspace{1cm} (5.25)

\( \lambda = 50, \quad \Gamma_{fric} = 0, \quad Gain_T = 20 \) and \( \theta(t_0) = 0 \)

where \( Gain_T \) is the scaling of the matrix \( \Gamma \) as described in section 5.1.2.

The results are plotted in figure 5.3, 5.4 and 5.5. It seems that the parameters do not convergence anymore after 18 seconds. In fact the signal \( s \), which drives the adaptation, became much smaller. As a consequence, the derivative of \( \dot{\theta} \) becomes smaller when the parameter fit becomes better. In order to speed up further adaptation, the adaptation gains can be made larger during the adaptation. It is not interesting to further investigate this
effect because unmodelled dynamics will prevent the errors to become that small in practice. Besides, when the error becomes smaller, the adaptation has to be lower because the error contains less information about modelled dynamics and more about unmodelled behaviour.

As can be seen from figure 5.5, the error in the parameters estimation converges to zero so it can be concluded that the regressor \( Y \) is errorless. In the controller of Berghuis, Ortega and Nijmeijer, the same regressor structure as Slotine and Li is used so correctness of the regressor is also proven for this controller.

![Figure 5.3: e during simulation of adaptation, without friction](image1)

![Figure 5.4: s during simulation of adaptation, without friction](image2)

### 5.4.2 Simulation with friction

Once the adaptation without friction appeared to work properly, friction can be added to the model of the manipulator. We directly apply the adaptation to the model with the neural network friction model. This can be justified because the structure of the friction-regressor \( X \) is much more simple than the regressor \( Y \) for adaptation on the BPS. Therefore there is no need to check this regressor by applying the adaptive controller to the manipulator model with linear friction. The neural-network representing the friction is approximated by a Coulomb-viscous friction model as described in equation (1.27). Due to the fact that there exists no exact fit to the neural friction model, the computed-torque is not able to completely compensate the friction. The resulting dynamics keeps producing errors which cause variations in the parameter estimation. Notice that these errors cause not only variations in \( \dot{\theta} \) but also in \( \ddot{\theta} \). This can be seen from the parameter-update laws (4.39) and (4.10) where the same error signal \( s \) is used to drive the adaptation for \( \dot{\theta} \) and \( \ddot{\theta} \).

During the simulation, the same controller setting were used as during the simulation without friction. The result during simulation with the controller of Slotine and Li were very similar to the result of the controller of Berghuis, Ortega and Nijmeijer. Therefore only the results of Slotine and Li are plotted.

From figure 5.8 and 5.9, it can be seen that parameters do not totally converge, but keep varying around some value. As mentioned before, this is due to the fact that there exists no exact fit for the parameters in order to cancelate for the dynamics of the model. In order to reduce the drift of the friction parameters, one could try to reduce the adaptation gain when the parameters seem to be converged. From simulations it appeared that lowering the
adaptation-gain after 20 seconds did lead to smaller parameters variations but did not lead to significant larger or smaller errors, as can be seen from figure 5.12 and 5.13.

5.5 Experiments

As the adaptive controller works properly on the simulation level, and experience on tuning the controller is obtained, we seem to be ready for real-life implementation. As expected, it appeared that the controller setting during the experiments has to be chosen more conservative than the setting for optimal performance during simulations. This is due to unmodelled dynamics and disturbances working on the real system.

The following strategy was used for controller tuning during the experiments. This strategy is a consequence of the trade-off between small tracking errors which results in excitation of unmodelled dynamics and fast adaptation which can only occur when the tracking error contains much information about the low frequent modelled dynamics and less high frequent unmodelled dynamics.

- Start with low PD gains. Large errors are allowed during the initial adaptation. The errors contain much low-frequent information and less high-frequent unmodelled dynamic behaviour. This is the parameters estimation phase where no performance is expected from the closed-loop system.
CHAPTER 5. IMPLEMENTATION

Figure 5.6: $e$ during simulation of adaptation

Figure 5.7: $s$ during simulation of adaptation

Figure 5.8: $\theta$ during simulation of adaptation

Figure 5.9: $\psi$ during simulation of adaptation

Figure 5.10: $\theta$ during adaptation with varying $\Gamma_{fric}$

Figure 5.11: $\psi$ during adaptation with varying $\Gamma_{fric}$
Figure 5.12: $e$ during adaptation with constant $\Gamma_{fric}$

Figure 5.13: $e$ during adaptation with varying $\Gamma_{fric}$

- After some time, the parameters seem to be converged to some value. Now the adaptation gains can be lowered because we do not want to adapt on unmodelled dynamics which occur during higher performance settings of the PD-controller. Lowering the adaptation gain is justified because most of the parameters of the manipulator will only change slowly.

- As the adaptation gains are reduced, the PD-gains can be made larger in order to improve the performance. The error has a large content of unmodelled dynamics. The controller now only adapt slow on varying parameters.

The controller-gains in the experiments are chosen as large as possible. This means that larger values for $\text{Gain}_r$, $\text{Gain}_{fric}$ would lead to instability with the current settings for $K_p$, $K_d$ and $\lambda$.

5.5.1 Experimental results: controller of Slotine and Li

First we will only apply the first step of this strategy to see how the adaptive algorithm behaves on a real system. The following settings are used for the adaptive scheme of Slotine and Li:

$\text{Gain}_r = 20$, $\text{Gain}_{fric} = 10$, $\theta_0 = 0$, $\psi_0 = 0$, $\Lambda = 20$

\[
K_p = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 15 \end{bmatrix}, \quad K_d = \begin{bmatrix} 0.8 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 0.4 \end{bmatrix}
\]  \hspace{1cm} (5.26)

The results are depicted in figure 5.14, 5.15, 5.16 and 5.17. Because of the large number of lines, no legend is plotted in figure 5.17. A table with the mean converged parameter values can be found in table 5.1.

As depicted in figure 5.14, the error decreases significantly to an order of $1 \times 10^{-2}$ radians during the adaptation. Because the friction parameters are not converged yet, it seems that the friction parameters have no significant influence on the performance of the feed-forward.
This can be explained from the fact that the computed-torque generated by $Y \dot{\theta}$ is much larger than the friction-compensation represented by $X \ddot{\psi}$.

Further increasing the PD-gains makes the system instable at arbitrarily moments because parameters estimation influences the the feed-back gains as discussed in section 4.1.

5.5.2 Experimental results: controller of Berghuis, Ortega and Nijmeijer

During the implementation of this controller, we change the controller setting when parameter convergence is reached according the the proposed strategy.

It appeared that the adaptation gains can be chosen much larger than with the controller of Slotine and Li. This can be explained by the fact that no additional PD-gains are introduced. As can be seen from figure 5.14 the errors are indeed larger due to the lower PD-gains but the adaptation-gains can also be faster. This shows again the trade-off between performance and adaptation.
Figure 5.17: \( \dot{\theta} \) Slotine and Li
The controller settings before parameter convergence are: $Gain_\Gamma = 30$, $Gain_{\Gamma_{re}} = 20$, $\theta_0 = 0$, $\psi_0 = 0$

$$K_p = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 15 \end{bmatrix}, \quad K_d = \begin{bmatrix} 0.8 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 0.4 \end{bmatrix}$$ (5.27)

The results are depicted in figure 5.14, 5.19 and 5.22. The values for $\hat{\theta}$ are again given in table 5.1.

In figure 5.20 and 5.21, the control output of the PD-part and the computed-torque part is given. As can be seen, the input-voltage needed to follow the trajectory moves up from the PD-part to the computed-torque part. The voltage supplied by the PD-part after parameter convergence is needed to reject disturbances and compensate unmodelled dynamics.

After the parameter estimates are converged, the adaptation gains can be lowered and
Figure 5.22: $\dot{\theta}$ during adaptation phase, Berghuis, Ortega, Nijmeijer
the PD-gains can be made higher. The controller-settings after parameter convergence are: 
\[ \text{Gain}_\text{r} = 10, \text{Gain}_\text{r,fric} = 10, \Lambda = 20 \]

\[
K_p = \begin{bmatrix} 170 & 0 & 0 \\ 0 & 150 & 0 \\ 0 & 0 & 70 \end{bmatrix}, \quad K_d = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]  
(5.28)

Notice that \( \text{Gain}_\text{r} \) and \( \text{Gain}_\text{r,fric} \) are smaller but adaptation is still active.

![Figure 5.23: Position-error [rad] after changing \( \Gamma \)](image)

From the resulting position errors depicted in figure 5.23 it can be concluded that position errors become much smaller with large PD-gains and smaller adaptation gains than visa versa. So it can be concluded that a large part of the position-error can not be fitted on the model, it is therefore better to suppress these errors with a PD-controller instead of compensating for them via the computed-torque with high adaptation gains. The adaptation is still active but the gains are lower so adapting on parameters variations will take more time.

### 5.5.3 High adaptation gains on the friction parameters

During the experiments it appeared that a high adaptation gain for the Coulomb-friction could make the system less robust with respect to stability. Oscillations are caused by Coulomb-friction compensation when high adaptation gains are chosen for the friction-parameters. This leads to overcompensating the Coulomb friction. This effect can be explained from the closed-loop dynamics of equation (4.5) and (5.29). Because the Coulomb-friction compensation is a sign-function, the \( K_d s \) term is not large enough to compensate the error in the Coulomb-friction compensation till \( s \) is big enough.

\[
M(q) \ddot{s} + C(q, \dot{q}) s = Y \ddot{\theta} + S \dot{\psi}_{\text{vis}} + Q \dot{\psi}_{\text{coulomb}} + K_d s
\]  
(5.29)

Remember that contrary to equation (4.5), we are now also dealing with friction so \( S \dot{\psi}_{\text{vis}} \) means the overcompensation for the viscous friction and \( Q \dot{\theta}_{\text{coulomb}} \) the overcompensation for the Coulomb-friction.
When \( s \approx 0 \), all the terms in the equation are approximate zero, except \( Q \dot{\psi}_{\text{Coulomb}} \). This is no problem as long as \( Q \dot{\psi}_{\text{Coulomb}} \leq 0 \). But when \( Q \dot{\psi}_{\text{Coulomb}} \geq 0 \), \( s \) becomes larger till \( s \) is big enough to compensate for this term so that \( Q \dot{\psi}_{\text{Coulomb}} \leq -K_d s + \ldots \). We would expect that the system goes to a stable limit cycle but this is not the case because the sign-function at the input causes excitation of high frequent dynamics. This causes problems in adaptation process which leads to errors in the computed-torques. This effect makes the system unstable.

This oscillation problem can be solved by feeding the computed-torque for Coulomb-friction compensation by \( \dot{q}_d \):

\[
\tau_{\text{friction}} = S(\dot{q}_d)\dot{\psi}_{\text{Coulomb}} + Q(\dot{q})\tilde{\psi}_{\text{vis}}
\]

(5.30)

In this case, there is no feed-back via the parameter estimation for Coulomb-friction so the adaptation gain \( \text{Gain}_{\text{fric}} \) can be chosen much larger. This is done during the experiments as described before.

### 5.5.4 Comparison with previous work

As a check it will be interesting to compare the estimated parameters with values found by previous work. During previous work [7], the unknown parameters are estimated by a least-squares method. The values of adaptive control and of least-squares are given below\(^1\). The value given for adaptive control are the converged values middled over the 10 second.

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \theta_{\text{Slotine}} )</th>
<th>( \theta_{\text{Bergbuis}} )</th>
<th>( \theta_{\text{least-squares}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 2.9 \times 10^{-1} )</td>
<td>( 2.8 \times 10^{-1} )</td>
<td>( 3.4 \times 10^{-1} )</td>
</tr>
<tr>
<td>2</td>
<td>( 1.4 \times 10^{-2} )</td>
<td>( 2.1 \times 10^{-2} )</td>
<td>( 4.6 \times 10^{-3} )</td>
</tr>
<tr>
<td>3</td>
<td>( 1.9 \times 10^{-1} )</td>
<td>( 2.2 \times 10^{-1} )</td>
<td>( 1.9 \times 10^{-1} )</td>
</tr>
<tr>
<td>4</td>
<td>( -3.0 \times 10^{-3} )</td>
<td>( -7.5 \times 10^{-3} )</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>( 7.1 \times 10^{-2} )</td>
<td>( 6.3 \times 10^{-2} )</td>
<td>( 5.5 \times 10^{-2} )</td>
</tr>
<tr>
<td>6</td>
<td>( -8.2 \times 10^{-3} )</td>
<td>( -6.3 \times 10^{-2} )</td>
<td>( -1.2 \times 10^{-2} )</td>
</tr>
<tr>
<td>7</td>
<td>( 7.7 \times 10^{-1} )</td>
<td>( 8.4 \times 10^{-1} )</td>
<td>( 6.9 \times 10^{-2} )</td>
</tr>
<tr>
<td>8</td>
<td>( -3.4 \times 10^{-3} )</td>
<td>( -2.1 \times 10^{-2} )</td>
<td>( -9.3 \times 10^{-3} )</td>
</tr>
<tr>
<td>9</td>
<td>( -4.5 \times 10^{-2} )</td>
<td>( -4.7 \times 10^{-2} )</td>
<td>( -4.2 \times 10^{-2} )</td>
</tr>
<tr>
<td>10</td>
<td>( -3.2 \times 10^{-3} )</td>
<td>( 1.2 \times 10^{-2} )</td>
<td>( -5.2 \times 10^{-3} )</td>
</tr>
<tr>
<td>11</td>
<td>( -3.1 \times 10^{-1} )</td>
<td>( -3.7 \times 10^{-1} )</td>
<td>( -3.0 \times 10^{-1} )</td>
</tr>
<tr>
<td>12</td>
<td>( 4.8 \times 10^{-1} )</td>
<td>( 5.7 \times 10^{-1} )</td>
<td>( 4.3 \times 10^{-1} )</td>
</tr>
<tr>
<td>13</td>
<td>( 7.2 \times 10^{-2} )</td>
<td>( 7.7 \times 10^{-2} )</td>
<td>( 7.6 \times 10^{-2} )</td>
</tr>
<tr>
<td>14</td>
<td>( 8.8 \times 10^{-2} )</td>
<td>( 9.3 \times 10^{-2} )</td>
<td>( 4.1 \times 10^{-2} )</td>
</tr>
<tr>
<td>15</td>
<td>( 1.6 )</td>
<td>( 1.6 )</td>
<td>( 1.5 )</td>
</tr>
<tr>
<td>16</td>
<td>( -3.7 \times 10^{-3} )</td>
<td>( -1.8 \times 10^{-2} )</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.1: Estimated values for \( \theta \)

\(^{1}\) The parameters below do not totally agree with the parameters of the article but these are new ones as found by the same approach.
The values found by adaptive control and the values found by previous work often lies in
the same region but can also differ significantly. Two explanations can be given for this:

- Because we are dealing with a highly non-linear system, observability of the parameters
  is hard to prove. It is possible that the measurements contains less information about
  certain parameters so that it is hard to estimate them.

- Another reason can be some singularity in the parameters. In other word, the system
  could also be represented by less parameters. This degree of freedom can cause the
  difference between the estimation techniques\(^2\).

The friction parameters can not be compared because the used models are not the same.
Using least-squares, the parameters of a neural network are fitted while with adaptive control
we estimated viscous and Coulomb friction parameters.

The position-error of the adaptive controller of Berghuis, Ortega and Nijmeijer and the
passive computed-torque controller with constant parameters, as implemented by previous
work [5], both peaks to \(5 \cdot 10^{-8}\). Note that the reference trajectory is not the same for these
two cases but because output voltages of both controllers peaks to the same values, the applied
torques and therefore the excitation of the system during these two experiments will be more
or less comparable. This justifies a comparison but for a better comparison, new experiments
should be done with the same trajectory.

Because the parameters are converged, no better parameters fit can be expected so further
performance improvement has to be obtained via the PD-part of the controller. It is expected
that better performance can be achieved when the PD-part of the controller is better tuned
using for example more complex filters.

\(^2\)By other research it is indeed found that the system can also be represented by 15 parameters
Chapter 6

Conclusion

In the report it is shown that adaptive controllers applied on systems with one or two degrees of freedom as often described in literature, are also suited for systems with higher degrees of freedom and many unknown parameters. As a consequence, the regressor matrix needed for the adaptation, is much more complex. Possibilities are found to derive this regressor matrix by symbolic programming in a universal way, which is suited for different definitions of the regressor and therefore different kinds of adaptive controllers. Much attention is paid to prevent errors by using a symbolic toolbox and proving correctness of this regressor by doing simulations on a robot-model for which correctness is proved by previous work.

Two well known algorithms from literature are studied and implemented. The first controller was proposed by Slotine and Li. This controller, based on the passivity concept, appeared to suffer from parameter drift due to noise in the velocity measurement. To overcome this problem, a second controller proposed by Berghuis, Ortega and Nijmeijer is chosen to implement. During derivations and simulations the influence of controller setting are investigated. Finally the controllers are tested on the real system. As expected, the controller settings appear to be more conservative in practice than during simulations because of the unmodelled dynamics and the presence of disturbances.

The fact that adaptive controllers are often avoided because of their unpredictable stability behavior seems to be ungrounded. During simulations and experiments, several phenomena occurred which seem to be unpredictable at first sight. This report tried to give the insight that many of these phenomena can be declared from the dynamics of the closed-loop system. During experiments, the following problems are experienced and explained with the closed loop dynamics:

- During experiments, the adaptive controller proposed by Berghuis, Ortega and Nijmeijer appeared to be more robust than the controller of Slotine and Li. This can be explained by the fact that the controller of Slotine and Li contains PD-terms which depend on the estimated parameters. This can result in an unstable system because disturbances can influence the estimated parameters in such a way that the PD-gains makes the system unstable. This is worked out in section 4.1.1.

- High adaptation gains for estimation of Coulomb friction parameters can also make a system less robust. This can be seen from equation (5.29). By calculating the computed-torque for Coulomb-friction compensation with the reference speed $q_d$ instead of the actual speed $\dot{q}$, the feed-back is removed and the problem is solved.
An important property during the tuning of an adaptive controller appeared to be the trade-off between small tracking errors, which needs high PD-gains, and fast adaptation, represented by a large $\Gamma$. High performance generates errors with a large content of high frequent unmodelled behaviour which can not be fit on the model, while low performance speeds up the adaptation due to much low-frequent modelled information from the system. Adaptation on large/fast parameter variations, which needs fast adaptation, will lead to larger tracking errors. This is on the one hand due to the fact that variations in the system behaviour needs a more robust tuning of the PD-controller. On the other hand, making PD-gains high, excites unmodelled dynamics which leads to lower adaptation gains. This leads to slower adaptation.

During the performed experiments on the RRR-manipulator the parameters are constant. As expected, it appeared that better results are obtained when PD-actions are made higher at expense of adaptation gain. This means that the error is mainly caused by unmodelled dynamics which can not be fitted on the model of the robot.

By tuning the controllers for best performance, it appeared that the performance is not better than a passive computed-torques controller with well chosen constant parameters. The tracking-error during a repetitive reference trajectory with frequencies till 10 Hertz as given in figure 5.1 are peaking to $5 \cdot 10^{-3}$ radians. Of course the power of adaptive controllers is not to get ultimate performance but the ability to learn unknown parameters during their work. Thinking of wear in bearings or cutting tools, adaptive controllers seem to be superior to computed-torque controllers with constant parameters.
CHAPTER 6. CONCLUSION

Recommendations

- As mentioned in this report, adaptive control is not able to handle unmodelled dynamics and disturbances. Sliding-mode is a way to suppress these effects, so it is likely that this method can further improve the performance. Although, by previous research, good results are experienced with sliding-mode control applied on one degree of freedom, applying a sliding-mode controller on three DOF's didn't give the performance which was expected. During further research, it will be interesting to investigate if a sliding-mode controller can increase the performance.

- Another way of reducing the errors is to better tune the PD-controller. In this report we didn't use more sophisticated PD-controller techniques like band-pass filters or lead-lag filters. The results of these techniques has to be investigated with respect to excitation of unmodelled dynamics and performance.

- It seems to be possible to represent the dynamics of the BPS with 15 parameters. Using a regressor with less parameters will lead to a compact representation and maybe less computational effort. Also a better comparison can be made between several estimation techniques because the BPS is then unique. When the matrixes $M(\theta)$, $C(\theta)$ and $G(\theta)$ are known in 15 parameters, a regressor using 15 parameters can easily be derived with the techniques represented in this report.

- During simulations and experiments is appeared that a steep arc-tan gives problems with implementation. So it is unavoidable to use a sign-function which excites unmodelled dynamics and causes problems in the computed-torque for the Coulomb friction. Smoothing the step-function can result in better performance.
Bibliography


Appendix A

Matrix $M$ and vector $h$

\[
M(1, 1) = 2 a_2 \cos(q_2 + q_3) \cos(q_2) \theta_3 \sin(q_2 + q_3) \cos(q_2) \theta_4 + \theta_7 \\
+ \cos(q_2 + q_3)^2 \theta_5 + \cos(q_2)^2 \theta_1 - \sin(q_2 + q_3) \cos(q_2 + q_3) \theta_6 \\
+ \sin(q_2) \cos(q_2) \theta_2 \\
M(1, 2) = \sin(q_2 + q_3) \theta_9 + \cos(q_2) \theta_{10} + \cos(q_2 + q_3) \theta_8 + \sin(q_2) \theta_{11} \\
M(1, 3) = \cos(q_2 + q_3) \theta_8 + \sin(q_2 + q_3) \theta_9 \\
M(2, 1) = M(1, 2) \\
M(2, 2) = \theta_{12} - 2 a_2 \sin(q_3) \theta_4 + 2 a_2 \cos(q_3) \theta_3 \\
M(2, 3) = a_2 \cos(q_3) \theta_3 + \theta_{13} - a_2 \sin(q_3) \theta_4 \\
M(3, 1) = M(1, 3) \\
M(3, 2) = M(2, 3) \\
M(3, 3) = \theta_{14};
\]
\[ h(1, 1) = -\theta_1 \sin(2 q_2) q_{d1} q_{d2} + (q_{d2} + q_{d3})^2 \cos(q_2 + q_3) \theta_3 \]
\[-2 a_2 \cos(q_2 + q_3) q_{d1} q_{d3} \cos(q_2) + \cos(2 q_2 + q_3) q_{d1} q_{d2} \theta_4 \]
\[+ 2 a_2 (-\sin(q_2 + q_3) q_{d1} q_{d3} \cos(q_2) - \sin(2 q_2 + q_3) q_{d1} q_{d2}) \theta_3 \]
\[+ \theta_2 \cos(2 q_2) q_{d1} q_{d2} + \cos(q_2) q_{d1}^2 \theta_{11} \]
\[-(2 \cos(q_2 + q_3) \sin(q_2 + q_3) q_{d1} q_{d3} + \sin(2 q_3 + 2 q_2) q_{d1} q_{d2}) \theta_5 \]
\[-\sin(q_2) q_{d1}^2 \theta_{10} - (q_{d2} + q_{d3})^2 \sin(q_2 + q_3) \theta_8 \]
\[-\cos(2 q_3 + 2 q_2) q_{d1} (q_{d2} + q_{d3}) \theta_6 \]

\[ h(2, 1) = \frac{1}{2} \sin(2 q_2) q_{d1}^2 \theta_1 - \frac{1}{2} \cos(2 q_2) q_{d1}^2 \theta_2 + (2 a_2 (-\frac{1}{2} q_{d3}^2 + q_{d2} q_{d3}) \sin(q_3) \]
\[+ \frac{1}{2} \sin(2 q_2 + q_3) q_{d1}^2 + g \cos(q_2 + q_3) \theta_3 \]
\[+ \frac{1}{2} \cos(2 q_3 + 2 q_2) q_{d1}^2 \theta_6 + (2 a_2 (-\frac{1}{2} q_{d3}^2 + q_{d2} q_{d3}) \cos(q_3) \]
\[+ \frac{1}{2} \cos(2 q_2 + q_3) q_{d1}^2 \theta_5 + g \sin(q_2 + q_3) \theta_4 + g \cos(q_2) \theta_{15} \]
\[+ \frac{1}{2} \sin(2 q_3 + 2 q_2) q_{d1}^2 \theta_5 - g \sin(q_2) \theta_{16} \]

\[ h(3, 1) = (2 a_2 (-\frac{1}{2} q_{d2}^2 + \frac{1}{2} q_{d1}^2) \sin(q_3) + \frac{1}{4} \sin(2 q_2 + q_3) q_{d1}^2 \]
\[+ g \cos(q_2 + q_3)) \theta_3 + \frac{1}{2} \cos(2 q_3 + 2 q_2) q_{d1} \theta_6 \]
\[+ (2 a_2 (-\frac{1}{2} q_{d2}^2 + \frac{1}{2} q_{d1}^2) \cos(q_3) + \frac{1}{4} \cos(2 q_2 + q_3) q_{d1}^2 \]
\[\sin(q_2 + q_3) \theta_4 + \frac{1}{2} \sin(2 q_3 + 2 q_2) q_{d1}^2 \theta_5 \]
Appendix B

Regressor implemented in C

The regressor is implemented in C code using the template of Matlab Simulink.

The meaning of:

\( q_{m}, \quad qdd_{multm}, \quad q_{c}, \quad qd_{c}, \quad qd_{multc}, \quad q_{g} \)

can be seen from the equation below:

\[
M(q_{m})qdd_{multm} + C(q_{c}, qd_{c})qd_{multc} + G(q_{g}) = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q)
\]

/* calculation of tau as function of regressor with information about C matrix *
* used to check regressor matrix */

#define S_FUNCTION_NAME regressor
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include "math.h"

/***************************************************************************/

/* Function: mdlInitializeSizes = analytic of target */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
    {
        return; /* Parameter mismatch will be reported by Simulink */
    }
    if (!ssSetNumInputPorts(S, 6)) return;

    ssSetInputPortWidth(S, 0, 3); // q_{m1}, q_{m2}, q_{m3}
    ssSetInputPortWidth(S, 1, 3); // qdd_{multm1}, qdd_{multm2}, qdd_{multm3}
    ssSetInputPortWidth(S, 2, 3); // q_{c1}, q_{c2}, q_{c3}
    ssSetInputPortWidth(S, 3, 3); // qd_{c1}, qd_{c2}, qd_{c3}
    ssSetInputPortWidth(S, 4, 3); // qd_{multc1}, qd_{multc2}, qd_{multc3}
    ssSetInputPortWidth(S, 5, 3); // q_{g1}, q_{g2}, q_{g3}
    ssSetInputPortWidth(S, 6, 3); // qdd_{multm1}, qdd_{multm2}, qdd_{multm3}

    /* Place your code here */
    return;
}
APPENDIX B. REGRESSOR IMPLEMENTED IN C

```c
ssSetInputPortWidth(S, 2, 3); // q_c1, q_c2, q_c3
ssSetInputPortWidth(S, 3, 3); // qd_c1, qd_c2, qd_c3
ssSetInputPortWidth(S, 4, 3); // qd_multc1, qd_multc2, qd_multc3
ssSetInputPortWidth(S, 5, 3); // q_g1, q_g2, q_g3

ssSetInputPortDirectFeedThrough(S, 0, 1); // feedthrough for input 1
ssSetInputPortDirectFeedThrough(S, 1, 1); // feedthrough for input 2
ssSetInputPortDirectFeedThrough(S, 2, 1); // feedthrough for input 3
ssSetInputPortDirectFeedThrough(S, 3, 1); // feedthrough for input 4
ssSetInputPortDirectFeedThrough(S, 4, 1); // feedthrough for input 5
ssSetInputPortDirectFeedThrough(S, 5, 1); // feedthrough for input 6

if (!ssSetNumOutputPorts(S, 3)) return;

ssSetOutputPortWidth(S, 0, 16);
ssSetOutputPortWidth(S, 1, 16);
ssSetOutputPortWidth(S, 2, 16);

ssSetNumSampleTimes(S, 1);

ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}

/* Function: mdlInitializeSampleTimes ------------------------------------------ */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, 0.001); // this could also be CONTINUOUS_SAMPLE_TIME
    ssSetOffsetTime(S, 0, 0.0);
}

/* Function: mdlOutputs ----------------------------------------------------- */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T *pointer_y1 = ssGetOutputPortRealSignal(S,0);
    real_T *pointer_y2 = ssGetOutputPortRealSignal(S,1);
    real_T *pointer_y3 = ssGetOutputPortRealSignal(S,2);

    InputRealPtrsType uPtrs_1 = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType uPtrs_2 = ssGetInputPortRealSignalPtrs(S,1);
    InputRealPtrsType uPtrs_3 = ssGetInputPortRealSignalPtrs(S,2);
    InputRealPtrsType uPtrs_4 = ssGetInputPortRealSignalPtrs(S,3);
    InputRealPtrsType uPtrs_5 = ssGetInputPortRealSignalPtrs(S,4);
    InputRealPtrsType uPtrs_6 = ssGetInputPortRealSignalPtrs(S,5);

    real_T q_m1, q_m2, q_m3, qdd_multm1, qdd_multm2, qdd_multm3;
    real_T q_c1, q_c2, q_c3, qd_c1, qd_c2, qd_c3, qd_multc1, qd_multc2, qd_multc3;
    real_T q_g1, q_g2, q_g3;

    real_T a2 = 0.2;
    real_T g = 9.81;
```
APPENDIX B. REGRESSOR IMPLEMENTED IN C

```c
real_T a[3][16];
int_T i;
//int_T width_tau = ssGetOutputPortWidth(S,0);

q_m1 = *uPtrs_1[0]; q_m2 = *uPtrs_1[1]; q_m3 = *uPtrs_1[2];
qdd_multm1 = *uPtrs_2[0]; qdd_multm2 = *uPtrs_2[1]; qdd_multm3 = *uPtrs_2[2];
q_c1 = *uPtrs_3[0]; q_c2 = *uPtrs_3[1]; q_c3 = *uPtrs_3[2];
q_c1 = *uPtrs_4[0]; q_c2 = *uPtrs_4[1]; q_c3 = *uPtrs_4[2];
q_multc1 = *uPtrs_5[0]; qd_multc2 = *uPtrs_5[1]; qd_multc3 = *uPtrs_5[2];
q_gl = *uPtrs_6[0]; q_g2 = *uPtrs_6[1]; q_g3 = *uPtrs_6[2];

/* definition of regressor matrix */
a[0][0] = cos(q_m2)*cos(q_m2)*qdd_multm1-0.5*sin(2*q_c2)*qdd_multc1
    -0.5*sin(2*q_c2)*qdd_multc2;
a[0][1] = sin(q_m2)*cos(q_m2)*qdd_multm1+0.5*cos(2*q_c2)*qdd_multc1
    +0.5*cos(2*q_c2)*qdd_multc2;
a[0][2] = 2*a2*cos(q_m2+q_m3)*cos(q_m2)*qdd_multm1+(-a2*sin(q_c2+q_c3)*qd_c3*cos(q_c2)
    -a2*sin(2*q_c2+q_c3)*qd_c2)*qd_multc1-a2*sin(2*q_c2+q_c3)*qd_c1*qd_multc2
    -a2*sin(q_c2+q_c3)*qd_c2*qd_multc3;
a[0][3] = -2*a2*sin(q_m2+q_m3)*cos(q_m2)*qdd_multm1+(-a2*cos(q_c2+q_c3)*qd_c3*cos(q_c2)
    -a2*cos(2*q_c2+q_c3)*qd_c2)*qd_multc1-a2*cos(2*q_c2+q_c3)*qd_c1*qd_multc2
    -a2*cos(q_c2+q_c3)*qd_c2*qd_multc3;
a[0][4] = cos(q_m2+q_m3)*cos(q_m2+q_m3)*qdd_multm1+(-0.5*sin(2*q_c3+2*q_c2)*qd_c2
    -0.5*sin(2*q_c3+2*q_c2)*qd_c3)*qd_multc1-0.5*sin(2*q_c3+2*q_c2)*qd_c1*qd_multc2
    -0.5*sin(2*q_c3+2*q_c2)*qd_c1*qd_multc3;
a[0][5] = -sin(q_m2+q_m3)*cos(q_m2+q_m3)*qdd_multm1-0.5*cos(2*q_c3+2*q_c2)*(q_c2*qd_c3)
    *qd_multc1
    -0.5*cos(2*q_c3+2*q_c2)*qd_c1*qd_multc2-0.5*cos(2*q_c3+2*q_c2)*qd_c1*qd_multc3;
a[0][6] = qdd_multm1;
a[0][7] = cos(q_m2+q_m3)*qdd_multm2+cos(q_m2+q_m3)*qdd_multm3
    -(q_c2*qd_c3)*sin(q_c2+q_c3)*qd_multc2-(q_c2*qd_c3)*sin(q_c2+q_c3)*qd_multc3;
a[0][8] = sin(q_m2+q_m3)*qdd_multm2+sin(q_m2+q_m3)*qdd_multm3
    +(q_c2*qd_c3)*cos(q_c2+q_c3)*qd_multc2+(q_c2*qd_c3)*cos(q_c2+q_c3)*qd_multc3;
a[0][9] = cos(q_m2)*qdd_multm2-sin(q_c2)*qd_c2*qd_multc2;
a[0][10] = sin(q_m2)*qdd_multm2+cos(q_c2)*qd_c2*qd_multc2;
a[0][11] = 0;
a[0][12] = 0;
a[0][13] = 0;
a[0][14] = 0;
a[0][15] = 0;

a[1][0] = 0.5*sin(2*q_c2)*qd_c1*qd_multc1;
a[1][1] = -0.5*cos(2*q_c2)*qd_c1*qd_multc1;
a[1][2] = 2*a2*cos(q_m3)*qdd_multm2+a2*cos(q_m3)*qdd_multm3+a2*sin(2*q_c2+q_c3)
    *qd_c1*qd_multc1
    -(2*a2*qd_c3)*sin(q_c3)*qd_multc2+(-a2*qd_c3)*sin(q_c3)++
    (2*a2*qd_c2+q_c3) *qd_multc3
    *(2*a2*qd_c2+q_c3) *qd_multc3
```
APPENDIX B. REGRESSOR IMPLEMENTED IN C

\[ +g \cos(q_{g2}+q_{g3}); \]
\[ a[1][3] = -2a_2\sin(q_{m3})q_{dd_multm2} - a_2\sin(q_{m3})q_{dd_multm3} + a_2\cos(2q_{c2}+q_{c3})q_{dd_multc1} - a_2q_{c3}\cos(q_{c3})q_{dd_multc2} + a_2q_{c3}\cos(q_{c3})q_{dd_multc3} - g\sin(q_{g2}+q_{g3}); \]
\[ a[1][4] = 0.5\sin(2q_{c3}+2q_{c2})q_{dd_multc1}; \]
\[ a[1][5] = 0.5\cos(2q_{c3}+2q_{c2})q_{dd_multc1}; \]
\[ a[1][6] = 0; \]
\[ a[1][7] = \cos(q_{m2}+q_{m3})q_{dd_multm1}; \]
\[ a[1][8] = \sin(q_{m2}+q_{m3})q_{dd_multm1}; \]
\[ a[1][9] = \cos(q_{m2})q_{dd_multm1}; \]
\[ a[1][10] = \sin(q_{m2})q_{dd_multm1}; \]
\[ a[1][11] = q_{dd_multm2}; \]
\[ a[1][12] = q_{dd_multm3}; \]
\[ a[1][13] = 0; \]
\[ a[1][14] = g\cos(q_{g2}); \]
\[ a[1][15] = -g\sin(q_{g2}); \]
\[ a[2][0] = 0; \]
\[ a[2][1] = 0; \]
\[ a[2][2] = a_2\cos(q_{c3})q_{dd_multm2} + (0.5a_2\sin(q_{c3})q_{dd_c1} + 0.5a_2q_{dd_c2}\sin(q_{c3})q_{dd_multc2} + g\cos(q_{g2})q_{dd_m}); \]
\[ a[2][3] = -a_2\sin(q_{m3})q_{dd_multm2} + (0.5a_2q_{dd_c1}\cos(q_{c3}) + 0.5a_2q_{dd_c2}\cos(q_{c3})q_{dd_multc2} - g\sin(q_{g2})q_{dd_m}); \]
\[ a[2][4] = 0.5\sin(2q_{c3}+2q_{c2})q_{dd_multc1}; \]
\[ a[2][5] = 0.5\cos(2q_{c3}+2q_{c2})q_{dd_multc1}; \]
\[ a[2][6] = 0; \]
\[ a[2][7] = \cos(q_{c2}+q_{c3})q_{dd_multm1}; \]
\[ a[2][8] = \sin(q_{c2}+q_{c3})q_{dd_multm1}; \]
\[ a[2][9] = 0; \]
\[ a[2][10] = 0; \]
\[ a[2][11] = 0; \]
\[ a[2][12] = q_{dd_multm2}; \]
\[ a[2][13] = q_{dd_multm3}; \]
\[ a[2][14] = 0; \]
\[ a[2][15] = 0; \]

/* output */

for (i=0; i<=16-1; i++)
{
   pointer_y1[i] = a[0][i];
   pointer_y2[i] = a[1][i];
   pointer_y3[i] = a[2][i];
}

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

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