MASTER

Model past estimation using a two point boundary value problem solver

Koopmans, H.T.G.

Award date:
1996

Link to publication
Model Past Estimation
using a
two point boundary value problem solver

hugo Koopmans
M.sc.thesis
WF-W-report 96.095
Model Past Estimation
using a
two point boundary value problem solver
hugo Koopmans
Master of science thesis
Abstract

In this report a solver for two point boundary value problems is developed using an artificial neural network, i.e. the Hopfield neural network. This solver is tested for an optimal estimation problem in system identification. Commonly used solver routines suffer from the Hamilton character of the system matrix, which causes numerical instability. With the network approach it is possible to build the Neuro Morphic Solver (NMS) for nonlinear two point boundary value problems (TPBVP) as they occur in optimal estimation and optimal control. It is shown here that the NMS suffers from slow convergence in large system identification problems. From this solver a new Direct Least Squares (DLS) solver is derived and tested successfully. The DLS solver is used in a new estimation technique called Model Past Estimation (MPE).

The MPE algorithm is tested off-line on experimental data.
to D.
for her patience
with me
to travel the globe.

to my parents
teaching values of life.
## Contents

1 Introduction  ........................................ ........................................ 3  

2 System Identification  .......................... ........................................ 5  
2.1 Problem definition  ........................................ ........................................ 5  
2.2 The estimation model  ........................................ ........................................ 6  
2.3 Optimal estimation  ........................................ ........................................ 6  
  2.3.1 The weighted least square norm  ........................................ ........................................ 7  
  2.3.2 Variation of the cost function  ........................................ ........................................ 7  
  2.3.3 The nonlinear Two Point Boundary Value Problem  ........................................ ........................................ 8  
  2.3.4 Alternative formulation of the Lagrange multiplier approach  ........................................ ........................................ 8  

3 The Neuro Morphic Solver for TPBVPs  ........................................ 11  
3.1 Introduction  ........................................ ........................................ 11  
3.2 The Hopfield network  ........................................ ........................................ 11  
3.3 Network design  ........................................ ........................................ 13  
  3.3.1 The Two Point Boundary Value Problem  ........................................ ........................................ 13  
  3.3.2 Error equations  ........................................ ........................................ 13  
  3.3.3 The structured Hopfield network  ........................................ ........................................ 14  
3.4 Verifying the Hopfield network approach  ........................................ ........................................ 16  
3.5 Constraints  ........................................ ........................................ 17  
  3.5.1 Constraints on the TPBVP for parameter estimation  ........................................ ........................................ 17  
  3.5.2 Constraints on the network states  ........................................ ........................................ 17  
  3.5.3 The Neuro Morphic Solver  ........................................ ........................................ 18  

4 System identification using the Neuro Morphic Solver  ........................................ 19  
4.1 Introduction  ........................................ ........................................ 19  
4.2 A mechanical system  ........................................ ........................................ 19  
4.3 Network simulations  ........................................ ........................................ 21  
  4.3.1 Estimating mass when measuring both position and velocity  ........................................ ........................................ 21  
  4.3.2 Estimating mass when measuring only position  ........................................ ........................................ 23  
  4.3.3 Estimating mass when measuring position with parameter constraints  ........................................ ........................................ 24  
4.4 Experimental Setup  ........................................ ........................................ 25  
4.5 Direct Least Squares solution  ........................................ ........................................ 27  
4.6 Experiments  ........................................ ........................................ 31  
4.7 Conclusion  ........................................ ........................................ 35
Chapter 1

Introduction

Background In general, the goal in system control is to let a system behave as desired. This can be done by developing a controller that controls input variables such that output variables describe the desired trajectory in space. Starting point in controller design is a mathematical model, that describes the system dynamics. In the model one finds state variables to deal with position, velocity and acceleration information and system parameters describing mechanical aspects of the system, such as for instance spring, damper or friction constants, mass, and inertia. System identification is about finding estimates of these states and parameters, using measurements of the system. These estimates must be accurate enough to be used in a controller.

Specific problem System identification plays an important role in control of mechanical systems. Here it is assumed that the model structure can be derived from the physical knowledge of the system. If we want to estimate parameters and at the same time reconstruct all state variables from measurements we can treat parameters in a likewise manner as we treat state variables. We can extend the state vector with the parameters to be estimated into an augmented state. Goal is to reconstruct this augmented state vector over a desired time span. Reconstruction of the augmented state can be done using a filter or a smoother algorithm. In a filter algorithm the augmented state at time \( t \) is calculated using only information from the past, whereas in a smoother algorithm information over the entire time span is used.

Problem definition:
Reconstruct the augmented state over a predefined time span using nonlinear model equations and measurement data.

Related work In [7] an introduction into system identification is given. It is shown that optimal estimation can be described as a nonlinear Two Point Boundary Value Problem (TPBVP) [7] [12] [16] [15].
Recent work in the field of artificial neural networks has shown that it is possible to solve such a TPBVP using a structured Hopfield neural network [1] [2] [3] [4]. This network can be shown to be a special case of a nonlinear programming circuit [3]. In [15] and [16] theory about optimal control can be found.

Motivation Due to the nonlinear nature of the combined parameter/state estimation problem it is desirable that the estimation method used expects a nonlinear estimation model. Optimal
estimation can be defined in terms of a nonlinear TPBVP, which is similar to optimal control that also can be defined as a TPBVP. So developing a solver for general nonlinear TPBVP can be useful in nonlinear Model Predictive Control (MPC) and in a new estimation algorithm called Model Past Estimation (MPE).

**Contribution** In this report a solver for the nonlinear TPBVP in optimal estimation is developed i.e. the Neuro Morphic Solver. This solver suffers from slow convergence speed due to the conjugate gradient method used in the optimization algorithm. A new solver is developed using the Direct Least Squares (DLS) method. This DLS solver is tested in a nonlinear estimation problem. The DLS solver is used in a new estimation technique called Model Past Estimation.

**Report overview** In general the system description will not correspond exactly to the real system due to incomplete modeling, simplifying assumptions and inaccurate parameter estimates. So a residue has to be taken into account. This leads to a nonlinear estimation model. After the introduction in Chapter 1, Chapter 2 starts from the mathematical model for nonlinear dynamical systems. A nonlinear estimation model is derived. Problem in estimation is minimizing the residuals that occur in the estimation model. By defining a measure that is to be minimized, the problem can be described as a nonlinear two point boundary value problem (TPBVP). In a likewise manner a TPBVP for optimal control can be formulated.

Developments in the field of artificial neural networks have shown that it is possible to solve such a TPBVP using the Hopfield network structure. In Chapter 3 this leads to the development of the Neuro Morphic Solver (NMS). The NMS is tested in Chapter 4 to solve a nonlinear estimation problem. Problems occur with convergence speed. That is why a new solver based on the Direct Least Squares optimization algorithm is developed. This DLS solver is tested successfully on simulation data. The DLS solver is then used to identify several parameters in an experimental setup.

In Chapter 5 a new estimation algorithm, Model Past Estimation (MPE) is presented. MPE is based on a solver for TPBVP. Here the DLS solver is used. In Model Past Estimation a memory window is defined in which the TPBVP for optimal estimation is solved. When new measurement data comes available the memory window is shifted ahead in time and the TPBVP is solved again. This loop is the essence of MPE. The MPE algorithm is tested off-line in system identification of an experimental setup at TUE.
Chapter 2
System Identification

2.1 Problem definition

Starting point in this report is a mathematical model of dynamic systems, which can be used for control purposes. Let us define the vector \( u \in \mathbb{R}^m \) to be the input of the system. The quantities to be controlled are elements of the vector \( z \in \mathbb{R}^l \).

We restrict ourselves to models which describe how the quantity to be controlled, \( z \), can be influenced by the input quantity \( u \). Furthermore we assume that the relationships in the model are given, so the model structure is known.

The unknown parameters in the model structure are put in a vector \( \theta \in \mathbb{R}^p \). The quantities to be measured are represented in the vector \( y \in \mathbb{R}^k \), which are the output variables of the system. Now a general mathematical model can be defined as:

\[
\begin{align*}
\dot{q}(t) &= \chi(q(t), \theta(t), u(t), t) \\
\dot{\theta} &= 0 \\
y(t) &= \gamma(q(t), \theta(t), u(t), t) \\
z(t) &= \varphi(q(t), \theta(t), u(t), t)
\end{align*}
\]

In which \( q \in \mathbb{R}^n \) is the state vector. The components of \( q \) are the state quantities which are internal model quantities necessary to describe the connection between, on the one hand, input \( u \) and, on the other hand, the quantities to be measured \( y \) and the quantities to be controlled \( z \).

Goal in control theory is to find an input \( u(t) \) that will make the quantities to be controlled \( z(t) \) follow a desired trajectory \( z_d(t) \). The literature on control theory provides us with information on how this can be done. Here the only thing of interest is that these methods often lead to a control law of the form:

\[
u(t) = -l(q(t), \theta(t), t)
\]

To apply such a law, \( \theta(t) \) and \( q(t) \) must be known at all time. In general not all of the quantities are measured and the measurements suffer from measurement errors.

The problem is how to find sufficiently accurate estimations \( \hat{\theta}(t) \) and \( \hat{q}(t) \) of \( \theta(t) \) and \( q(t) \) from measurements \( m(\tau) \) of output \( y(\tau) \) with \( \tau \in [t_0, t_e] \).

When estimating off-line, all measurements in \( \tau \in [t_0, t_e] \) are available, when estimating on-line only measurement data from the past is known and the algorithm must be fast enough to solve the estimation problem in real-time. Here we consider off-line estimation.
2.2 The estimation model

Assuming it is possible to find such estimates of \( \theta(t) \) and \( q(t) \) an estimation model can be introduced as:

\[
\begin{align*}
\xi(t) & = \dot{q}(t) - \chi(q(t), \hat{\theta}(t), u(t), t) \\
\eta(t) & = \dot{\hat{\theta}}(t) \\
\zeta(t) & = m(t) - \hat{y}(t) \\
\hat{y}(t) & = \gamma(q(t), \hat{\theta}(t), u(t), t)
\end{align*}
\]

(2.6) (2.7) (2.8) (2.9)

The residuals \( \xi, \eta \) and \( \zeta \) represent the difference between the model equations and their estimates. Here, \( \hat{y}(t) \) is an estimate of \( y(t) \) found if, at time \( t \), the available estimates \( \hat{\theta}(t) \) and \( q(t) \) and the input \( u(t) \) are substituted in Eq.(2.3).

The parameters \( \theta \) can be interpreted in a mathematical sense as state quantities, which are constant in time. We take \( q \in \mathbb{R}^n \) and \( \theta \in \mathbb{R}^p \) to be part of the augmented state \( x \in \mathbb{R}^{(n+p)} \):

\[
x(t) = \begin{bmatrix} q(t) \\ \theta(t) \end{bmatrix}
\]

(2.11)

With the augmented state the state equation Eq.(2.1) and the parameter equation Eq.(2.2) can be combined to

\[
\dot{x}(t) = \chi(x(t), u(t), t)
\]

(2.12)

\[
\chi(x(t), u(t), t) = \begin{bmatrix} \chi(x(t), u(t), t) \\ 0 \end{bmatrix}
\]

(2.13)

For clarity, time dependence of the variables \( x \) and \( u \) will not be explicitly written unless it causes indistinctness.

Then

\[
\begin{align*}
\dot{x} & = \chi(x, u, t) \\
y & = \gamma(x, u, t)
\end{align*}
\]

(2.14) (2.15)

and the estimation model in the augmented state form is

\[
\begin{align*}
\xi & = \dot{x} - \chi(x, u, t) \\
\zeta & = m - \hat{y}
\end{align*}
\]

(2.16) (2.17)

where \( \hat{y} = \gamma(x, u, t) \) and \( \hat{x} = \begin{bmatrix} \dot{q} \\ \dot{\theta} \end{bmatrix} \).

Goal is to minimize the residuals \( \xi \) and \( \zeta \), which leads to the optimal estimate \( \hat{x} \).

2.3 Optimal estimation

The optimal estimation problem can now be defined as:

Find the optimal estimate \( \hat{x} \) that minimizes the residuals \( \xi \) and \( \zeta \) where \( \hat{x} \) is subjected to the constraints defined in the estimation model.
2.3.1 The weighted least square norm

To find minimal values of $\xi$ and $\zeta$ we define a cost function $J$ in which the residuals occur in quadratic form, weighted over the time interval $[t_0, t_e]$:

$$J(\hat{x}) = \frac{1}{2} \int_{t_0}^{t_e} \xi(\hat{x}, t)^T W \xi(\hat{x}, t) + \zeta(\hat{x}, t)^T V \zeta(\hat{x}, t) \, dt$$

(2.18)

Here, weighting matrix $W$ is a positive definite matrix expressing confidence in the state equation 2.14 and weighting matrix $V$ is a semi positive definite matrix expressing confidence in the measurement equation 2.15.

The constraint defined by the estimation model 2.16 has to be taken into account when minimizing $J$. A method for solving an optimization problem with equality constrains, is the Lagrange Multiplier method. In this method the constraint is multiplied by a vector of variables $\lambda(t)$, the so called Lagrange multipliers. The cost function is extended with this extra term:

$$J(\xi, \hat{x}, \dot{\hat{x}}, \lambda) = \int_{t_0}^{t_e} \frac{1}{2} \xi^T W \xi + \frac{1}{2} \zeta^T V \zeta + \lambda^T (\dot{x} - \chi(\hat{x}, u, t) - \xi) \, dt + \frac{1}{2} \eta^T R \eta$$

(2.19)

Where $\zeta = m - \hat{y}$. The cost function is a function of $\xi, \hat{x}, \dot{\hat{x}}, \lambda$ because by introducing the Lagrange multipliers the augmented state variables and residuals have become independent.

Confidence in the initial estimation $\hat{x}(t_0) = x_0$ can be taken into account by adding an extra term to the cost function. Where $\eta = \hat{x}(t_0) - \hat{x}_0$ defined as the difference between $\hat{x}$ at time $t$ and the initial estimation $\hat{x}_0$.

2.3.2 Variation of the cost function

A necessary, but not sufficient condition for $J$ to be minimal is:

$$\delta J = 0 \quad \forall \quad \delta \xi, \delta \lambda, \delta \hat{x}$$

(2.20)

Variation with respect to $\xi$:

$$\int_{t_0}^{t_e} \delta \xi^T (W \xi - \lambda) \, dt = 0 \quad \forall \quad \delta \xi \rightarrow \lambda = W \xi$$

(2.21)

Variation with respect to $\lambda$:

$$\int_{t_0}^{t_e} \delta \lambda^T (\dot{x} - \chi(\hat{x}, u, t) - \xi) \, dt = 0 \quad \forall \quad \delta \lambda \rightarrow \dot{\hat{x}} = \chi(\hat{x}, u, t) + \xi$$

(2.22)

Variation with respect to $\hat{x}$:

In Eq.(2.19) $\hat{x}$ as well as $\dot{\hat{x}}$ appear. To variate $J$ with respect to $\hat{x}$ we have to loose $\dot{\hat{x}}$, which can be done by integrating $J$ by parts:

$$J(\xi, \hat{x}, \lambda) = \left[ \lambda^T \dot{x} \right]_{t_0}^{t_e} + \frac{1}{2} \left( \dot{x}(t_0) - x_0 \right)^T R (\dot{x}(t_0) - x_0)$$

$$+ \int_{t_0}^{t_e} \left( \xi^T W \xi + \frac{1}{2} (m - \gamma)^T V (m - \gamma) - \lambda^T \dot{x} + \lambda^T \dot{\hat{x}} \right) \, dt$$

(2.23)
System Identification

Now consider the variation of \( J \) due to variations in the state vector \( \delta \hat{x} \).

\[
\delta J = \lambda(t_e)^T \delta \hat{x}(t_e) - \lambda(t_0)^T \delta \hat{x}(t_0) + (\hat{x}(t_0) - x_0)^T R \delta \hat{x}
\]

\[
+ \int_{t_0}^{t_e} \left( [m - \gamma]^T V [-\frac{\partial \gamma}{\partial \hat{x}} - \hat{\lambda}^T - \lambda^T \frac{\partial \gamma}{\partial \hat{x}}] \right) \delta \hat{x} dt = 0 \quad \forall \quad \delta \hat{x}
\]

(2.24)

The integral term results in:

\[
\dot{\lambda} = -\frac{\partial \gamma^T}{\partial \hat{x}} \lambda - \frac{\partial \gamma^T}{\partial \hat{x}} V(m - \hat{y})
\]

(2.25)

and the other terms in 2.24 lead to boundary conditions

\[
\lambda(t_0) = R(\dot{x}(t_0) - x_0)
\]

(2.26)

\[
\lambda(t_e) = 0
\]

(2.27)

where \( R \) is a weighting matrix expressing confidence in the initial estimation \( x_0 \). If \( R=0 \) no a-priori knowledge about the initial estimation of the state vector is taken into account.

2.3.3 The nonlinear Two Point Boundary Value Problem

Substituting Eq.(2.21) in Eq.(2.22) together with Eq.(2.25) yields the nonlinear TPBVP:

\[
\dot{x} = \chi(\hat{x}, u, t) + W^{-1} \lambda
\]

(2.28)

\[
\dot{\lambda} = -\frac{\partial \chi^T}{\partial \hat{x}} \lambda - \frac{\partial \gamma^T}{\partial \hat{x}} V(m - \hat{y})
\]

(2.29)

with boundary conditions:

\[
\lambda(t_0) = 0
\]

(2.30)

\[
\lambda(t_e) = 0
\]

(2.31)

if no a-priori knowledge on \( x_0 \) is known (\( R=0 \)), or

\[
\lambda(t_0) = R(\dot{x}(t_0) - x_0)
\]

(2.32)

\[
\lambda(t_e) = 0
\]

(2.33)

if an initial estimate \( x_0 \) is known. Confidence in this initial estimate can be expressed by \( R \). In the next section we will show a method to solve this nonlinear TPBVP.

2.3.4 Alternative formulation of the Lagrange multiplier approach

Rearranging Eq.(2.28) gives

\[
\lambda = W \left( \hat{x} - \chi(\hat{x}, u, t) \right)
\]

(2.34)
When Eq.(2.34) is substituted in Eq.(2.29) we obtain:

\[ W\dot{\xi} - W\dot{\chi} + \frac{\partial \chi^T}{\partial \xi} W\dot{\xi} - \frac{\partial \chi^T}{\partial \xi} W\chi + \frac{\partial \gamma^T}{\partial \xi} V(m - \tilde{y}) = 0 \]  \hspace{1cm} (2.35)

With boundary conditions

\[ \dot{\xi}(t_0) = \chi(\tilde{x}(t_0), u(t_0), t_0) \]  \hspace{1cm} (2.36)
\[ \dot{\xi}(t_e) = \chi(\tilde{x}(t_e), u(t_e), t_e) \]  \hspace{1cm} (2.37)

On the other hand, variation of Eq.(2.18) to \( \delta \xi \) and \( \delta \zeta \) gives:

\[ W\dot{\xi} + \frac{\partial \chi^T}{\partial \xi} W\xi + \frac{\partial \gamma^T}{\partial \xi} V(m - \tilde{y}) = 0 \]  \hspace{1cm} (2.38)

Substitute Eq.(2.16) and one obtains the same result as Eq.(2.35).
System Identification
Chapter 3

The Neuro Morphic Solver for Nonlinear Two Point Boundary Value Problems

3.1 Introduction

In this chapter the method proposed by S.K. Biswas and J.J. Helferty is used for solving a nonlinear TPBVP [1]. The method uses the Hopfield network structure proposed by W. Tank and J. Hopfield [4]. First, it is shown that the network originates from an electronic circuit, so it can be implemented in hardware as an analog computer [4],[5],[6]. The Hopfield network can be unified to the canonical nonlinear programming circuit of Chua and Lin [3]. This is used here to extend the solver for TPBVP to deal with inequality constraints.

3.2 The Hopfield network

In [4] Tank&Hopfield propose a continuous network for Linear Programming. This network is shown to be a special case of the canonical nonlinear programming circuit proposed by Chua and Lin [2].

It can be seen from figure 3.1 that by using Kirchhoff's current law, the sum of the currents entering a point in an electrical circuit must equal the sum of currents leaving this point, the dynamics of the circuit can be written as

$$C_i \frac{du_i}{dt} = -\frac{ui}{R_i} - \sum_{j=1}^{N} w_{ij} h_i^{-1}(v_j) - \theta_i - g_i(\phi(v))$$

(3.1)

Where $\theta_i$ is the input bias current, $u_i$ is the internal voltage, $v_i$ is the output voltage and $w_{ij}$ are resistors to weight the output feedback. $C_i$ is a capacitor and $R_i$ is a resistor at each node to implement the network dynamics. $N$ is the number of network nodes and $h$ is the output function, $v = h(u)$. The function $g_i(\cdot)$ is a nonlinear penalty function for linear constraints $\phi(v)$ that are violated.
This network can be built as an electronic circuit [5], [6]. Networks of analog processors with this basic organization can be used to compute solutions to specific optimization problems by relating the minimization of the problem cost function to the minimization of the network energy function. To map the energy function to our optimization problem we take $R_i = \infty$ and $C_i = 1$.

Furthermore, $h(\cdot)$ has to be monotone non-increasing because of stability requirements. Here we take a proportional output map $v = h(u) = u$ then the network becomes

$$\frac{du_i}{dt} = - \sum_{j=1}^{N} w_{ij} u_j - \theta_i - g_i(\phi(u_i))$$

(3.2)

To prove the stability of the network Lyapunov's second method is used. Consider the candidate Lyapunov function

$$E = \frac{1}{2} \sum_{ij} w_{ij} u_i u_j + \sum_{i=1}^{n} \theta u_i + \sum_{i=1}^{n} g_i(\phi(u_i))$$

(3.3)

Taking the time derivative and using the fact that $\frac{\partial E}{\partial u_i} = 1$

$$\dot{E} = \left( \sum_{j} w_{ij} u_j + \theta_i + g_i(\phi(u)) \right) \dot{u}$$

(3.4)

$$\dot{E} = -(\dot{u})^2$$

(3.5)

$$\dot{E} \leq 0$$

(3.6)

the energy function is non-increasing, as required. The proof is independent of the nonlinear penalty function $g(\cdot)$, in [3] it is shown that a suitable choice of $g(\cdot)$ is essential to guarantee a bounded solution.
3.3 Network design

In [1] is explained how a network structure can be developed to solve a general TPBVP. The method consists of the following five steps.

- Definition of the original TPBVP
- Transformation into an integral error equation
- Transformation into a discrete error equation
- Transformation into the Hopfield network energy function
- Determination of the steady state of the network as the solution to the original TPBVP

First we recast the TPBVP into one of minimization of an integral error function over the continuous time space. Through a discrete numerical approximation, the integral error function is transformed into an error function over a finite number of discrete points. This discrete error function serves as the energy function of the Hopfield neural network. This leads to a neural network configuration with interconnection weights and input bias currents adapted to the specific problem to be solved. The rest state of the neural network yields the solution of the original TPBVP.

3.3.1 The Two Point Boundary Value Problem

A general Two Point Boundary Value Problem can be defined as

\[
\begin{align*}
\dot{y}_1 &= f_1(y_1, y_2), \quad y_1(0) = y_{10} \\
\dot{y}_2 &= f_2(y_1, y_2), \quad y_2(T) = y_{2T}
\end{align*}
\]

where \( y_1 \in \mathbb{R}^m, y_2 \in \mathbb{R}^{n-m} \) are the system state variables, and \( f_1 : \mathbb{R}^m \times \mathbb{R}^{n-m} \rightarrow \mathbb{R}^m, f_2 : \mathbb{R}^m \times \mathbb{R}^{n-m} \rightarrow \mathbb{R}^{n-m} \) are nonlinear functions describing the system dynamics. \( y_{10} \) and \( y_{2T} \) are the split boundary conditions. We assume that \( f_1 \) and \( f_2 \) satisfy appropriate conditions so that a solution is guaranteed. In [1] it is shown how we can transform this problem into that of minimizing an error function suitable for network processing.

3.3.2 Error equations

The integral error function to be minimized can be defined as

\[
J(y_1, y_2) = \int_0^T \|\dot{y}_1 - f_1(y_1, y_2)\|^2 dt + \int_0^T \|\dot{y}_2 - f_2(y_1, y_2)\|^2 dt + \|y_1(0) - y_{10}\|^2 + \|y_2(T) - y_{2T}\|^2
\]

where \( \| \cdot \| \) is the Euclidean norm.

Clearly, if the pair \( y_1(t), y_2(t) \) is a solution to the TPBVP, the integral error function 3.8 will have the value zero.

Minimizing Eq.(3.8) is essentially a problem of minimization of a functional over the space of
continuous differentiable functions. This problem can be transformed into an equivalent problem over a finite number of points. This can be accomplished if we discretize the time interval \([0,T]\) into a set of discrete points and represent the solution \(y_1(t), y_2(t)\) as a set of vectors in these discrete points. Let \(N\) be the number of discrete points over the time interval \([0,T]\) such that the time step between any pair of nodes is \(\Delta = T/(N - 1)\). Throughout the rest of this thesis, we shall use the superscript \(i\) to indicate the discrete time points and denote \(y^i\) as the solution \(y_1(t)\) at the \(i\)-th node point, i.e. at time \(t = (i-1) \Delta\). In order to transform Eq. (3.8) into a discrete error function we use the trapezium rule to approximate the integral error function. The discrete error function becomes:

\[
J(y_1^i, y_2^i, i = 1, 2, \ldots, N) = \Delta \sum_{i=2}^{N-1} ||\dot{y}_1^i - f_1(y_1^i, y_2^i)||^2 + \Delta \sum_{i=2}^{N-1} ||\dot{y}_2^i - f_2(y_1^i, y_2^i)||^2
+ \frac{\Delta}{2} ||\dot{y}_1^1 - f_1(y_1^1, y_2^1)||^2 + \frac{\Delta}{2} ||\dot{y}_2^1 - f_2(y_1^1, y_2^1)||^2
+ \frac{\Delta}{2} ||\dot{y}_1^N - f_1(y_1^N, y_2^N)||^2 + \frac{\Delta}{2} ||\dot{y}_2^N - f_2(y_1^N, y_2^N)||^2
+ ||y_1^1 - y_1^0||^2 + ||y_2^N - y_2^0||^2
\]  

(3.9)

Here, \(\dot{y}^i\) can be approximated by an appropriate difference formula. Due to this approximation the solution of the TPBVP will also be an approximation of which the accuracy will depend on the order of difference formulas used for \(\dot{y}^i\) and the number of discrete points \(N\) in the time interval \(T\).

### 3.3.3 The structured Hopfield network

It is possible to represent the discrete error function Eq. (3.9) as an energy function for the Hopfield network. Let \(N\) be the number of discrete points in time \(T\) and let \(n\) be the number of equations to be solved simultaneously. For minimization of the discrete error function Eq. (3.9) we can develop a network of \(N\) packets of neurons consisting of \(n\) processing elements in each packet. The output of each packet would give the solution of \(n\) state variables of the TPBVP at the corresponding discrete time point. In particular, the output of the \(i\)-th packet would give the solution to the \(n\) state variables of \(y_1(t)\) and \(y_2(t)\) at time \(t = (i-1) \Delta\). Consider the discrete error function Eq. (3.9) as an energy function and map the energy into the interconnection weights and bias currents of the network.

Then the network dynamics are described by

\[
\frac{dy^i}{ds} = \sum_{j=1}^{N} T_{ij} y^j + b^i
\]  

(3.10)

where \(y^i \equiv [y_1^i, y_2^i]^T\) is the output of the \(i\)-th packet of a collection of \(n\) neurons, \(b^i\) is a \(n\) vector with bias currents and \(T_{ij}\) is a \(n \times n\) matrix, consisting the interconnection weights \(w_{ij}\). Note that each packet of neurons has its own interconnection matrix \(T_{ii}\) and is connected to other packets through \(T_{ij}\). In Eq. (3.10) \(s\) is the network simulation time. The network has a total of \(N\) packets so the total number of neurons is \(n \times N\). It can be verified that the rest state of the network represents the minimum of the energy function:

\[
E = -\frac{1}{2} \sum_{ij} y^T T_{ij} y^j - \sum_i b^i y^i
\]  

(3.11)
The interconnection weights $T^{ij}$ and the bias currents $b^i$ are defined by the gradient of the discrete error function Eq.(3.9).

$$\nabla J = \left( \frac{\partial J}{\partial y_1}, \frac{\partial J}{\partial y_2}, \ldots, \frac{\partial J}{\partial y_i}, \ldots, \frac{\partial J}{\partial y_N} \right)$$

(3.12)

![Network structure for internal node i, n=2.](image)

When using second order difference formulas, for the internal nodes ($i=3,4,\ldots,N-2$), the following relations are obtained:

$$T^{i,i} = -\frac{1}{\Delta} I_n$$

(3.13)

$$T^{i,i+1} = \left[ \frac{\partial f}{\partial y_1}(y^i) \right]^T$$

(3.14)

$$T^{i,i-1} = -\left[ \frac{\partial f}{\partial y_1}(y^i) \right]^T$$

(3.15)

$$T^{i,i+2} = \frac{1}{2\Delta} I_n$$

(3.16)

$$T^{i,i-2} = -\frac{1}{2\Delta} I_n$$

(3.17)

$$T^{i,j} = 0 \quad \text{for any other } j$$

(3.18)

and for the bias

$$b^i = f(y^{i-1}) - f(y^{i+1}) - 2\Delta \left[ \frac{\partial f}{\partial y_1}(y^i) \right]^T f(y^i)$$

(3.19)

where $f \equiv [f_1, f_2]^T$, $y^i \equiv [y_1^i, y_2^i]^T$ and $\frac{\partial f}{\partial y}$ is the Jacobian matrix. $I_n$ is the unity matrix of dimension $n$. 

15
The network structure for the internal nodes is presented in Fig(3.2). The arrows coming from each discrete node represent the corresponding part of matrix $T$. The boundary conditions cause a few extra terms in the equations for $i=1$ and $i=N$, and due to different approximation of $\frac{dY_i}{ds}$ the weight and bias matrices at the boundaries are different.

3.4 Verifying the Hopfield network approach

To verify the network approach consider the linear TPBVP:

\[
\begin{align*}
\dot{y}_1 &= y_2, & y_1(0) &= 0 \\
\dot{y}_2 &= -\pi y_1, & y_2(1) &= -\pi
\end{align*}
\] (3.20) (3.21)

This problem can be solved analytically. The solution is given by $y_1(t) = \sin(\pi t)$ and $y_2(t) = \pi \cos(\pi t)$.

**Jacobian** For this problem the Jacobian matrix is:

\[
J = \begin{bmatrix} 0 & 1 \\ -\pi & 0 \end{bmatrix}
\] (3.22)

![Graphs showing the solution](image)

Figure 3.3: $y$ with $N=21$ nodes and $s=1$, $s=3$ and $s=5$ seconds

The network is initialized by setting $y_i = 0$ for $i=2$ to $N$ and $y_1^N = 0$ for $i=1$ to $N-1$. The boundary conditions are $y_i^L = 0$ and $y_i^N = -\pi$. Fig(3.3) shows the results at $s=1$, $s=3$ and $s=5$ seconds, were $s$ is the simulation time of the Hopfield network. After 5 seconds of simulation time the error is decreased to $10^{-4}$. Convergence is better then in [1] because the Conjugate Gradient method is used in a more efficient way. Here, instead of taking a fixed step size, the optimal step size in each search direction is calculated, before making a step.
3.5 Constraints

There are two possible ways to deal with inequality constraints on the state variables:

- 1. Redesign the estimation problem and extend the cost function to deal with constraints.
- 2. Use the network to deal with linear constraints directly.

3.5.1 Constraints on the TPBVP for parameter estimation

We can put constraints on certain parameters to bound their value. For instance a negative mass estimate has no physical meaning, thus the mass can be bounded to be positive. In general, any constraint $\phi(\tilde{x})$ can be implemented. This constraint can be added to the cost function by a nonlinear penalty function $g(\cdot)$ weighted by $P$.

$$J(\xi, \tilde{x}, \hat{x}, \lambda) = \int_{t_0}^{t_f} \frac{1}{2} \xi^T W \xi + \frac{1}{2} \xi^T V \xi + \frac{1}{2} \xi^T P g(\phi(\tilde{x})) dt + \lambda^T (\tilde{x} - \chi(\tilde{x}, u, t) - \xi) dt + \frac{1}{2} \eta^T R \eta$$

This results in an extra term in the TPBVP

$$\dot{\hat{x}} = \chi(\tilde{x}, u, t) + W^{-1} \lambda$$
$$\dot{\lambda} = -\frac{\partial \chi}{\partial \tilde{x}}^T \lambda - \frac{\partial \gamma}{\partial \tilde{x}}^T V (m - \hat{y}) + \frac{\partial g(\phi(\tilde{x}))}{\partial \tilde{x}}^T P g(\phi(\tilde{x}))$$

with appropriate boundary conditions.

3.5.2 Constraints on the network states

In this section it is shown how the Hopfield network can deal with linear inequality constraints, this method is used here to bound parameters. Various choices for the nonlinear penalty function $g(\cdot)$ can be made. It is important that by the choice of the penalty function, the energy function will stay bounded from below, see [3]. Because the solution to the nonlinear problem will not necessarily lie on a constraint, we choose a penalty function as:

$$g(x) = \begin{cases} 0 & \text{if } x > 0 \\ \frac{1}{R} x & \text{if } x < 0 \end{cases}$$

Where $x$ is the constraint to be satisfied. When the constraint is satisfied it has no influence on the accuracy of the solution, so $g(x) = 0$. Variable $\frac{1}{R}$ characterizes the 'hardness' of the constraint, see figure 3.4. When $R \to 0$ the constraint is exactly met. In simulations $R$ cannot be chosen close to zero because of numerical instability. When taking for instance $R = 0.1$ some overshoot on the constraint has to be taken into account.
3.5.3 The Neuro Morphic Solver

In this chapter a structured Hopfield neural network was presented. The energy function of the network system dynamics was shown to be a Lyapunov function of the network. So stability is guaranteed. An error function for a nonlinear TPBVP was defined. This error function can be mapped onto an energy function of the network. By simulating the network the energy in the network is minimized and so is the corresponding discrete error function for the TPBVP. The optimization routine uses a conjugate gradient method. Constraints can be dealt with in two different ways, they can be assimilated into the TPBVP, or they can be taken as constraints on the network states. Here the latter method is used because of simple implementation into the algorithm.

Figure 3.4: Nonlinear penalty function $g(\cdot)$
Chapter 4

System identification using the Neuro Morphic Solver

4.1 Introduction

In this chapter identification on simulated and experimental data is presented to test the theory from the previous chapter. The NMS is tested with several different set-ups in two simple system identification problems. In the first problem a mass is identified in a simple mechanical system. Parameter constraints on the mass are implemented. In the next problem a simulation model of an experimental setup at TUE is used. Goal in this experimental setup is identification of the inertia and a Coulomb friction constant. With this identification model an elementary problem in the NMS is uncovered. The optimization routine used is a conjugate gradient method which is apparently too slow for these kind of optimization problems.

The optimization routine in the solver is adapted and identification with this improved optimization routine is performed to show that the principal of optimization of the discrete error function is useful in optimal estimation. Experimental data are used to identify the inertia and friction in the experimental setup. The simulations and experiments are done in MatLab 4.2c. The MatLab software was run mostly on a Silicon Graphics Indy computer.

4.2 A mechanical system

Figure 4.1: mass spring damper system

The mechanical system used in the first simulation is a one degree of freedom (d.o.f.) mass-spring-damper system (see Fig(4.1)). The mass \( m \), the spring constant \( k \) and the damper constant \( b \)
are the system parameters. The force $F$ is the input and the d.o.f. $q$ is the mass position. For identification we apply an input signal $u = F$ and measure position $q$. The equation of motion, using Newton's second law, becomes

$$\ddot{q} = \frac{1}{m} (F - b\dot{q} - kq) \tag{4.1}$$

Assume the mass $m$ is the parameter to be estimated and $b$ and $k$ are both known. For identification an input signal $u$ is applied and position $q$ is measured. Using the augmented state:

$$x = \begin{bmatrix} q \\ \dot{q} \\ \theta \end{bmatrix} \text{ with } \theta = m,$$

the system can be represented in state space by

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{m} (F - b\dot{q} - kq) \\ 0 \end{bmatrix} \tag{4.2}$$

With this model the NMS is tested to solve the TPBVP for optimal estimation for the augmented state.
4.3 Network simulations

In this section the Neuro Morphic Solver is tested with the estimation model described in the previous section. As we have seen in the previous chapter, the NMS consists of a structured neural network. The TPBVP is mapped onto the energy function of this network. When simulating the neural network it will minimize its energy function. The rest state of the network is the solution to the discrete error function which can be interpreted as a discrete solution to the TPBVP for optimal estimation.

In this section the NMS is tested for three different situations. In the first network simulation the situation is ideal, both position and velocity are measured and the estimation model is exact. The next simulation is performed when only position information is measured and in the third simulation the problem is solved with a constraint on the estimated parameter while measuring only position.

4.3.1 Estimating mass when measuring both position and velocity

The NMS is set up according to:

**Weighting matrices.**

The model is exact and both position and velocity are measured exactly, so W and V are chosen identity matrices of appropriate size. In the initial estimate only $\hat{m}$ is likely to be wrong so the corresponding element in R is chosen 0.

The matrices are chosen as:

\[
W = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

(4.3)

\[
V = \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]

(4.4)

\[
R = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

(4.5)

**Number of network nodes.**

When choosing a network size, a compromise has to be made between accuracy and problem size. With a small discrete step the problem can be solved more accurate but the network size becomes large, so calculation time increases. We have to choose the smallest possible network size to limit calculation time. The minimal network size is determined by hand by a few trial and error approaches. The network size chosen in this simulation is N=64. The discrete step size $\Delta = 0.16[s]$.

**Initial conditions.**

The initial conditions for the network are obtained by simulating the mechanical system with the initial parameters for the desired time span. Here the initial mass estimation is 0.6 [kg]. The exact system parameters are $m=1[kg]$, $b=0.1[Ns/m]$ and $k=10 [N/m]$. 

\[21\]
Input signal

As input signal a combination of two sine waves with different frequencies is chosen:
\[ u = A_1 \sin(\omega_1 t) + A_2 \sin(\omega_2 t) \] (see Fig. (4.2)).
To excite the system adequately, \( \omega_1 \) and \( \omega_2 \) are chosen close to the eigen-frequency \( \omega_0 = \sqrt{\frac{k}{m}} \) of the system, i.e. \( \omega_1 = 0.8\pi, \omega_2 = 1.1\pi \), \( A_1 = A_2 = 1 \).

\[ \text{Figure 4.2: Input signal} \]

\[ \text{Figure 4.3: State variables} \]
\[ \text{Figure 4.4: Lagrange multipliers} \]

In Figs. (4.3) and (4.4) the results of a simulation with the above settings are shown. The solver is stopped at an accuracy of \( 1.1 \times 10^{-6} \). When measuring both position and velocity the solver has no problem finding the right estimate for \( m \). In Fig. (4.3) the measurements for position and velocity can not be distinguished from the smoothed states, this is good and it was expected because the model as well as the measurement was exact. The noise on the estimated parameter can be explained by discretization inaccuracy. Looking at the scale in the figure for the estimated mass, the parameter is constant with a deviation of less then 1 promille. The Lagrange multipliers in Fig. (4.4) are small and they should be, considered that they are related to the model error.
4.3.2 Estimating mass when measuring only position

For the next simulation the previous settings are kept except that only the position is measured. To increase accuracy the number of nodes is doubled to 128, so $\Delta = 0.08$[s]. Because the model used in the solver is the same model as used in the simulation, the model equations are exact. Theoretically the weight matrices can still be chosen identity matrices, but problem stiffness causes slow convergence. To deal with this problem stiffness, weight matrices can be chosen in such a way that the stiffness is minimal. Choosing $V$ and $W$ in such a way that the TPBVP has minimal stiffness is done by looking at the poles of the Jacobian of the TPBVP. $W$ and $V$ are chosen in such a way that the difference between the poles on the real axis is as small as possible. This can be done by a root locus on $J$ with respect to the elements of $V$ and $W$. To compromise for the problem stiffness, $V$ is set to 100. Because of the fact that the model is still exact, $W$ is chosen in the same order as $V$. To express that the mass is constant here a larger weighting factor $W_{33}$ is chosen.

\[
W = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 10^4 \end{bmatrix}
\]  
(4.6)
\[
V = \begin{bmatrix} 100 \end{bmatrix}
\]  
(4.7)
\[
R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\]  
(4.8)

With $V=100$ the measurement is weighted sufficiently heavy. In the figure for position the measurement and the smoothed state can not be distinguished from each other. Figs.(4.5) and (4.6) show the results. In the figure for position the measured and the smoothed state can not be distinguished from each other. The network simulation was stopped after the solver had minimized
The discrete error to $2.7 \times 10^{-3}$. The initial estimate for the mass was also 0.6[kg]. The Lagrange multiplier for position is a factor 100 larger then in the previous simulation. This can be explained by matrix $W$ which is chosen a factor 100 larger. In the TPBVP the inverse of the matrix $W$ occurs which explains why the corresponding Lagrange multiplier is larger then in the previous simulation. Because the velocity is not measured, this is not the case for the Lagrange multiplier for velocity. This Lagrange multiplier is a factor 50 larger then in the previous simulation. Even when the weighting factor for the mass is chosen $10^4$ it is not as constant as when both position and velocity were measured.

When measuring both position and velocity the solver reaches an accuracy of $1.1 \times 10^{-6}$ and finds an estimate for the mass parameter of $1.005[kg]$. When measuring only position the solver reaches an accuracy of $2.7 \times 10^{-3}$, then the mean value of the mass parameter is $1.015[kg]$. These are fairly good results.

4.3.3 Estimating mass when measuring position with parameter constraints

Look at the initial dynamics in the estimated parameter when only position is measured in Fig.(4.7). The dynamics in the initial parameter estimates are large. The model equation expects a constant parameter, but in the first few iterations this is definitely not the case. Because this dynamic behavior only exists in the first iteration it can be explained by numerical inaccuracies when initializing the solver.

We want to implement some constraints on this parameter so it will be bounded between $0.5 < m < 1.2$. Therefore these bounds were split into two separate constraints: $m - 0.5 > 0$ and $1.2 - m > 0$. In figure 4.8 the effect of the constraints on the parameter can be seen. The lower bound is never violated but the upper bound constrains the parameter to the desired value of 1.2. At the start, the constraint is violated due to the fact that in the penalty function $g(\cdot)$ the variable...
$R = 0.01$ and that is $\neq 0$, which makes $g(\cdot)$ a soft constraint.

With this constraint on the parameter the solver reaches an accuracy of $3.83 \times 10^{-4}$ and the mean value of the estimated parameter is $1.0022 \text{ [kg]}$. With the constraint problem, the solver finds a more accurate solution than in the unconstrained problem.

To explain this we have to go into detail about the stop criterion used. In the solver a variable step size was implemented. The solver was stopped when it was not longer possible to decrease the cost function with a minimal step size of $10^{-8}$. The overshoot on the estimated parameter is not as strong as in the unconstrained problem, of cause this is why we implemented the constraint, so the solver is much quicker near a better estimate. Then the solver has more iterations available to reach a more accurate solution. This stop criterion, which is in fact a norm on the cost function, can be argued about. Here, it is used to limit calculation time, but it would be better to implement a criterion by which the solver reaches a prepared accuracy, which is weighted over the number of discrete points used.

### 4.4 Experimental Setup

The NMS is tested in an experiment. Measurements were taken from an experimental setup as in Fig. (4.9).

![Figure 4.9: Experimental setup](image)

This is a rotational servo system where the load is linked to the actuator by an flexible joint which is adjustable. The actuator is a DC motor which generates an input torque on the actuator mass. Interfacing with the MatLab simulation environment is done by dSPACE [18]. Position measurements were taken by means of an incremental encoder with a resolution of $4.2 \times 10^{-4}\text{[rad]}$. In the experiment the flexible joint is blocked, so actuator and load can be modeled as one mass (see Fig. (4.10)). Goal is to identify this total mass and the friction for rotational movement. The
System identification using the Neuro Morphic Solver

equation of motion is given by:

\[ \ddot{\phi} = \frac{1}{J}(T - T_w) \]  \hspace{1cm} (4.9)

Input signal \( u = T \) and Coulomb friction is assumed. To deal with numerical problems, the Coulomb friction is modeled by

\[ T_w = f_w \frac{e^{s\dot{\phi}} - 1}{e^{s\dot{\phi}} + 1} \]  \hspace{1cm} (4.10)

Were the steepness of the discontinuity can be influenced by \( s \). With \( s = 50 \) no numerical problems occur. So, this is an acceptable approximation of the sign function (see Fig.(4.12)).

**Input signal**

The input signal is a filtered band limited white noise to excite the system in a relevant area of the state space. Because the model used, expects only one inertia and because the fact that the joint which is fixed, is still relatively flexible. The input signal should not excite the system in high frequencies. That is way the white noise is filtered with a low pass filter. The white noise has a frequency of 20[Hz] and is filtered by a low pass filter with cut-off frequency of 80[Hz] to eliminate high order dynamics (see Fig.(4.11)).

![Figure 4.11: Input signal](image1)

![Figure 4.12: friction model](image2)

**Simulating the experimental setup.**

From the equation of motion an estimation model is derived, this model is used to perform a simulation with parameters exactly known. Previous experiments have shown that these parameters in the actual system are approximately \( J = 0.05[kgm^2] \) and \( f_w = 0.1[Nm] \) so these parameters were used in the simulation.

To get a reference accuracy for the solver, a simulation is done in which the initial estimates correspond with the parameters used in the simulated data. When using a discrete step size \( \Delta = 0.008[s] \)
Direct Least Squares solution

the solver can minimize the discrete error to $7.78 \times 10^{-5}$.

When using wrong initial estimates for the parameters, it shows that convergence is very slow. The NMS is stopped because of excessive calculation time. The NMS reduces the discrete error to $2.9 \times 10^{-3}$ in $10^4$ iterations. The solution found by the NMS shows little parameter convergence.

**About accuracy and convergence**

Theoretically the solver should come close to the accuracy reached with exact initial parameters. This is not the case and that can have a number of causes. The discrete error function may have local minima in which the solver gets trapped. Because convergence is slow, the solver stopped as a result of prematurely activation of the stop criterion on reaching a minimum. This stop criterion is a minimal step size that has to be made with the calculated gradient to achieve a smaller discrete error function in the next iteration. This minimal step size was set to $10^{-5}$ which was chosen to limit calculation time. A better stop criterion would be a norm on the gradient but as stated before this would mean an unacceptable use of computer time.

The NMS has two main problems. First, due to the stiff character of the estimation problem, convergence on minimizing the discrete error function is very slow. Due to this a stop criterion was chosen which was not adequate.

Second, parameter convergence is small. It is shown in the next section that this is due to the discrete error formulation.

### 4.5 Direct Least Squares solution

In [1] the solver was evolved from a structured neural network. In this structured network economic implementation of the conjugate gradient method is possible. It was shown empirically that this optimization method is too slow to be useful in optimal estimation problems. Therefore we let go of the structured Hopfield neural network and solve the least squares discrete error function Eq.(3.9) directly.

The least squares optimization routine from the MatLab Optimization Toolbox can be used. This has several advantages:

- optimal line search routine is selected automatically,
- no analytical Jacobian needed,
- standard semi second order optimization routines available, such as Levenberg-Marquardt and Gauss-Newton.

These advantages reduce calculation time substantially (minutes instead of hours). However constraints as applied in the structured Hopfield network are not possible here. Constraints are implemented directly into the TPBVP formulation as described previously in section 3.5.1, they introduce an extra matrix $P$. When choosing the magnitude of matrix $P$ equal to that of matrix $W$, the constraints are weighted as heavy as the error on the model equations. Depending on the 'hardness' of the constraints, elements in matrix $P$ can be chosen differently.
System identification using the Neuro Morphic Solver

In a simulation with this improved optimization algorithm the solver shows convergence of the parameters (see Figs. (4.13) to (4.16)). Ideal measurements were assumed. The parameters used in the simulation are $J = 0.05[kgm^2]$ and $fw = 0.1[Nm]$. With a discrete step size $\Delta = 0.008[s]$ the discrete error function can be reduced to $8.87 \times 10^{-3}$. The solver finds $J = 0.0500[kgm^2]$ and $fw = 0.0996[Nm]$. The initial parameters can be chosen anywhere between 0 and 0.5. When choosing large initial parameters the constraints on the parameters to remain positive become active (the value of 0.5 is the largest value tested). Here we take initial parameters $J = 0.01[kgm^2]$ and $fw = 0.05[Nm]$.

When using identity matrices $W$, $V$, $P$ and $R$ the discrete error function is minimized without parameter convergence. There is too much freedom in the weighted least squares norm defined by the discrete error formulation. The inaccuracy introduced by the discretization of the continuous problem and the set of Lagrange multipliers cause the discrete error function to be minimized without finding the right parameters. To put more emphasis on the part of the set of equations that is of interest (namely the model equations) the TPBVP can be scaled such that model errors are weighted heavily. Matrices $W$, $V$, $P$ and $R$ have to be chosen large to scale the TPBVP such that the discrete error function can be reduced only by adapting the parameters. This can be done by choosing matrices of magnitude $10^8$, $10^{18}$.

When simulating also quantization noise as introduced by the encoder, the measurement equation is not exact. This is expressed by matrix $V$ chosen 100 times smaller than $W$. The discrete error function is reduced to $10^{-4}$. Results are shown in Figs.(4.17) to (4.20). The solver finds $J = 0.0499[kgm^2]$ and $fw = 0.1014[Nm]$. The initial estimates for $J$ and $fw$ are $J = 0.01[kgm^2]$ and $fw = 0.05[Nm]$. The Lagrange multipliers for position is substantial larger due to the scaling of the quantization noise on the measurement. Thus a more noisy signal then can be expected. The large value of matrices $V$ and $W$ cause the large value of the Lagrange multiplier on position.
Direct Least Squares solution

Figure 4.13: position and velocity

Figure 4.14: inertia J and friction f

Figure 4.15: Lagrange multipliers for position and velocity

Figure 4.16: Lagrange multipliers for estimated inertia J and friction f
System identification using the Neuro Morphic Solver

Figure 4.17: position and velocity

Figure 4.18: inertia J and friction f

Figure 4.19: Lagrange multipliers for position and velocity

Figure 4.20: Lagrange multipliers for estimated inertia J and friction f
4.6 Experiments

With the new and improved solver off-line estimation on experimental data is performed. The estimation problem is solved over a time span of 0.5 seconds with discrete time step $\Delta = 0.010[s]$. This defines 50 discrete points which is an acceptable problem size for the DLS solver. Weighting matrices are chosen as in the simulation with quantization noise. The matrices are:

$$W = \begin{bmatrix} 10^8 & 0 & 0 & 0 \\ 0 & 10^8 & 0 & 0 \\ 0 & 0 & 10^8 & 0 \\ 0 & 0 & 0 & 10^8 \end{bmatrix}$$ (4.11)$$

$$V = \begin{bmatrix} 10^6 \end{bmatrix}$$ (4.12)$$

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$ (4.13)

In order to gain information on the friction phenomena, the time span is chosen such that the velocity is switching sign in the time span covered by the solver. The solver is stopped at an accuracy of $9 \times 10^{-3}$, which is of the same magnitude as reached with exact initial parameters in simulation data. Results are shown in Fig.(4.21) to (4.24).

With initial estimates $J = 0.05[kgm^2]$ and $f_w = 0.1[Nm]$, the estimated parameters become $J = 0.061[kgm^2]$ and $f_w = 0.1246[Nm]$. When choosing initial estimates $J = 0.1[kgm^2]$ and $f_w = 0.15[Nm]$ the estimated parameters are $J = 0.061[kgm^2]$ and $f_w = 0.1245[Nm]$, so parameter convergence is good when choosing initial parameter estimates both above and below the convergated parameter estimates.
System identification using the Neuro Morphic Solver

Figure 4.21: position and velocity

Figure 4.22: inertia $J$ and friction $f$

Figure 4.23: Lagrange multipliers for position and velocity

Figure 4.24: Lagrange multipliers for estimated inertia $J$ and friction $f$
A simulation is performed with initial parameters and estimated parameters over a time span of 4 seconds. To show that parameter estimates are not the same in two different time spans the parameters are estimated over two different spans in these 4 seconds. In Fig. (4.25) the results are shown together with the actual measurement for the position and the reconstructed velocity. The method used to reconstruct the velocity from the measure position is described in [17]. With the estimated parameters obtained in the time span from 1.5 [s] to 2 [s], so $J = 0.061 [kgm^2]$ and $f_w = 0.1245 [Nm]$, the simulation is better then with initial parameters $J = 0.05 [kgm^2]$ and $f_w = 0.1 [Nm]$. When looking at a time span with larger velocities, for instance between 0.75 [s] and 1.25 [s] the estimate for friction becomes larger $f_w = 0.1759 [Nm]$, inertia $J = 0.0628 [kgm^2]$. With these parameters the simulation is even better. From this simulation it becomes clear that the time span over which the TPBVP is solved is important for the value of the estimated parameters. This means that the model used to identify the system is not correct. Problems with the model were anticipated by choosing an input signal which was filtered with a low-pass filter such that the joint which connects the two masses can be considered fixed. An explanation for the experimental results is a wrong friction model. If a time span is covered in which alarge velocities occur, the friction constant becomes larger. This indicates that in the friction torque a viscous component can be present.
System identification using the Neuro Morphic Solver

Figure 4.25: simulation
4.7 Conclusion

In this chapter it is shown that the NMS is capable of solving the TPBVP for optimal estimation in a simple mechanical system. Only one parameter had to be estimated. When using the solver in a larger problem, that is with more parameters to be estimated, the solver shows poor convergence speed. Optimization with the NMS using a conjugate gradient method, is too slow to be used in estimation problems of some size. This was not trivial, all the more because the use of a structured network assumed an economic use of the band symmetric property of the Jacobian matrix.

Higher order convergence algorithms had to be implemented, which could not be done in the network structure. For implementation of higher order convergence methods the Jacobian has to be in a vector form so it can be used to approximate the Hessian matrix. This is why we left the structured network approach and switched to the Direct Least Squares method, which is available as a standard routine in the MatLab optimization toolbox. This has several advantages which all come forward in the convergence speed of the algorithm.

With the Direct Least Squares (DLS) solver it is shown that the discrete error formulation leaves much freedom to be minimized without convergence of parameters. To deal with this problem the TPBVP is scaled by choosing large weighting matrices.

The Direct Least Squares (DLS) solver is robust for wrong initial parameters. Constraints on the parameters to remain positive were implemented successfully. Convergence speed is much faster than in the Neuro Morphic Solver. The DLS solver will be used in exploring the possibilities of Model Past Estimation in the next chapter.

From the experiments performed in this chapter, it was shown that parameters that were presumed to be constant in the model were estimated differently depending on the time span over which the TPBVP was solved. This can be explained by assumptions made when modeling the friction phenomena. From the experiments it was seen that the friction model used was not complete. Besides a Coulomb friction also a viscous component is likely to be present. Nevertheless, in the next chapter, this simple model is used to test Model Past Estimation (MPE). MPE is developed as an on-line estimation technique using a nonlinear estimation model.
Chapter 5

Model Past Estimation

5.1 Introduction

In this chapter a new estimation technique is presented. The technique can use any solver for a nonlinear TPBVP, here the Direct Least Squares (DLS) solver is used. The DLS solver can be used in on-line state reconstruction in processes which have sufficiently large time constants for the calculation to keep up with the process. Both optimal estimation and optimal control can be expressed in a TPBVP [20],[21],[7]. So the DLS solver can be used in nonlinear Model Predictive Control (MPC) as well as in Model Past Estimation (MPE), which is nonlinear by nature. Here emphasis is put on estimation. This estimation technique extends the analogy between optimal estimation and optimal control, described in [7]. As in MPC, Model Past Estimation uses a time span over which a criterion is optimized.

Figure 5.1: the MPE memory window
Model Past Estimation

5.2 What is MPE?

Model Past Estimation is an estimation technique which is based on solving a TPBVP for optimal estimation. A TPBVP is defined over a time span $t_w$ from the past to the present, the so called memory window. The TPBVP for optimal estimation, defined over the memory window, is solved in the time available between two successive memory windows (see Fig.(5.1)). Depending on the sampling time needed on-line, the window is shifted a desired time span ahead to the most recently available measurement data and the TPBVP is solved again. From the smoothed estimated $\hat{x}(\tau | \tau \in [t - t_w, t])$ available over the entire memory window only the reconstructed state at the present time is used as a filtered estimate $\hat{x}(\tau | \tau \in [t - t_w, t])$. So the first filtered estimate comes available after one memory window of time $(t_0 + t_w)$.

After the MPE memory window is shifted, the TPBVP is defined over the new window. The smoothed estimates calculated in the previous time span can be used as an initial estimate in the next window. For the initial estimates over the time span that was not in the previous window we have freedom of choice. For measured states, these measurements are taken as initial estimates over the new time span. For the states not measured we take a zero order hold of the old states (a higher order extrapolation can be used also). The Lagrange multipliers over the entire time span are take zero in each new window. The iterative process of solving the TPBVP and shifting the memory window is the essence of MPE.

In the MPE algorithm there are in fact two separate problems to deal with. The first has to do with identification, while the second is of more numerical nature. The identification problem depends on observability of the system and on a persistently exciting input signal. The system described by the TPBVP has to be observable, which means that from measurement data it has to be possible to reconstruct the entire augmented state vector over the desired time span. Also, the input signal has to excite the system in such a way that the information needed to identify the system can be revealed.

In MPE the second, numerical problem is that of finding a solution to the optimal estimation problem. Here, this optimal estimation problem is defined in a discrete error function which, when minimized, leads to a solution of the TPBVP for optimal estimation. This discrete optimization problem is solved using the DLS solver. Because a discrete error formulation is used, the solver suffers from inaccuracy due to time discretization of the continuous time problem. The nonlinear nature of the estimation problem causes long computation time which is a well-known problem in nonlinear optimization routines.

In the following, several aspects of the MPE algorithm are discussed with respect to both identification and numerical aspects.

Window size
By window size we mean the time span covered by the memory window defined as $t_w$. Important from identification point of view is the amount of information that is put in the window by the input signal. There has to be enough information in the memory window to excite the system adequately so parameters can converge. We have to keep this in mind when implementing the MPE algorithm into an on-line control system. It could mean that when a controlled system is in a steady-state and the input signal is a constant or zero, the estimated parameters can drift.
What is MPE?

because of leak of information.

**Sampling time**

An other critical item in designing MPE is the sampling time. The measured state(s) must be sampled such that all necessary information from the system is available in the measurement signals. So, the sampling time has to be small enough to define a problem that can be solved with the needed accuracy. Too few data points lead to a less accurate solution, it is possible that the accuracy asked from the solver can never be reached.

It can be thought of the possibility to interpolate between measured data points to increase accuracy. Or, if accuracy allows it, to skip measured data points to minimize the optimization problem. The discretization used in the solver is not necessarily equal to the sampling time of the measured signal or to the sampling time needed by the controller.

**Window shift time** The window shift time $t_w$ is defined as the time between two successive memory windows. When using the MPE filtered estimate of the augmented state in a controller, the window shift time is determined by the update rate needed by the controller.

**Solver accuracy and calculation time**

The MPE algorithm needs a solver for the TPBVP for optimal estimation defined over the memory window. The DLS solver used here, uses a discrete time error formulation which defines an optimization problem over a number of discrete time points. Window size and sampling time together define this optimization problem size. A long memory window together with a small sampling time defines a large number of discrete points, while a short memory window with a large sampling time defines a small number of discrete points. Calculation time depends on the optimization problem size, thus it depends on the number of discrete points.

Accuracy of the solver depends on the discrete step size and thus on the sampling time used in the memory window.

**Initializing MPE**

The solver needs knowledge of a reasonable initial guess for the state variables. However, when starting MPE no information is available from any previous memory window. To get an initial guess for the state variables in the initial window, the measured state(s) can be used to reconstruct not measured state variables. A more elegant way to obtain an initial guess is to perform a simulation with the identification model and the initial parameters.
5.3 Input signal

The input signal used in the simulation and the experiment is a filtered band limited white noise superposed on a sine wave with a frequency of 0.5 [Hz] and an amplitude of 0.8 [Nm] (see figure 5.2). The band limited white noise with power density of 0.01 and a sample time of 0.05 [s] was filtered with a low pass filter (cut-off frequency at 80 [Hz]) to suppress high order dynamics.

![Filtered band limited white noise on a sine wave](image)

Figure 5.2: Input signal used in MPE

5.4 Simulation

The simulation model is taken from an experimental setup at TUE described in the previous chapter. Measurements are simulated as if coming from the encoder used in the experimental setup. The position $\phi$ is measured. For an initial estimate in the first window, the velocity $\dot{\phi}$ is reconstructed using a data reconstruction method described in [17]. Weight matrices are chosen $W = 10^6 I$ because the problem must be scaled and $V = 10^6$, a factor 100 smaller because of simulated quantization noise. The MPE algorithm is started with a discrete step size $\Delta = 0.010$ [s] and a memory window of 0.5 [s]. Initial parameter estimates are $J = 0.1$ and $f_w = 0.2$. Filtered estimates are shown in figure 5.3 to 5.6, the actual system parameters are $J = 0.050$ and $T_w = 0.100$. The accuracy of the DLS solver is set to $10^{-2}$.

When looking at the Lagrange multipliers for position in figure 5.5 it can be seen that some relative high peaks occur which correspond with an inaccurate parameter estimation. These peaks result from the fact that in these points the solver does not reach the desired accuracy within the maximum number of iterations. The optimization process is terminated at a preset maximum number.
of iterations which is here $1.5 \times 10^4$.

In Fig. (5.3) the filtered position estimate and the measured position are shown. It seems like only a dotted line can be seen but they cannot be distinguished from each other. Besides the peaks explained before the Lagrange multipliers are small, at least a factor 100 smaller then the corresponding states.
In Fig. (5.7) the differences between the filtered states and the simulated states are shown. As we can see the error on the measured position is very small. The error on the not measured velocity is larger but when not looking at the peaks caused by the solver problem the filtered velocity is also good.

![Figure 5.7: difference between filtered and simulated position and velocity in simulation](image)

This simulation shows that the MPE algorithm can deal with the quantization noise introduced by the encoder used in the experimental setup. Now we are ready to start MPE on experimental data.
5.5 Experiment

The MPE algorithm is tested on data from the experimental setup. The DLS solver is setup as in the simulation with measurement noise, that is with $W = 10^8 I$, $V = 10^6$. In this experiment the solver is set to an accuracy of 0.1. The memory window covers 0.5 [s] with a discrete step size $\Delta = 0.01 [s]$. The memory window is moved over 0.05 [s] when shifted. The MPE algorithm is run for 69 window shifts covering a time span of 3.5 seconds. The initial parameter estimates for the MPE process are obtained from the experiments done in the previous chapter. They are taken $J = 0.06 [kgm^2]$ and $f_w = 0.12 [Nm]$.

The filtered estimates are shown in Figs.(5.8) to (5.11). The inertia is more or less constant with a mean value of 0.0647[kg] and a standard deviation of 0.0026[kg]. The trend for the friction constant shows dependency on the velocity sign. When velocity is negative the friction constant is about 40 % lower then with a positive velocity. There is a possibility that, for instance because of excessive use of the system in one particular direction, the friction is dependent on the rotation direction.

The high peaks in the Lagrange multipliers are again explained by the fact that at these points the solver was not able to reach the desired accuracy within the preset maximum number of iterations.

When looking at Fig(5.8) the measured and the filtered position can not be distinguished from each other. This was also the case in the simulation were the filtered states proved to be a good estimates of the simulated state variables.
Figure 5.8: position and velocity

Figure 5.9: estimated mass and friction

Figure 5.10: Lagrange multipliers for position and velocity

Figure 5.11: Lagrange multipliers for estimated mass and friction
To investigate this further, the measured position was used to reconstruct the velocity. The measured position and the reconstructed velocity were compared with the filtered estimates. In Fig. (5.12) it can be seen that the filtered estimates are still fairly good. The error on position is small and the error on velocity is acceptable. The error on velocity can be explained by a not modeled, viscous friction component. The error grows larger as the velocity increases.

Figure 5.12: difference between filtered estimates and measured position and reconstructed velocity
5.6 Future of MPE

The algorithm of Model Past Estimation is based on a suitable solver for the nonlinear TPBVP defined by the optimal estimation problem. The Direct Least Squares solver developed in this thesis can solve such a nonlinear TPBVP in a calculation time of several minutes. This restricts the use of the MPE algorithm to relative slow processes.

Variable window size.

To reduce calculation time we have to minimize the problem size. As discussed previously, this can be done by editing the window size or the sampling time. The latter is fixed by the accuracy needed by the solver. But the first can be optimized by choosing the smallest possible memory window which still guarantees parameter convergence. At present this window size is fixed and determined by trial and error. In the future an on-line algorithm can be developed that investigates the input signal on a measure of excitement. From this measure of excitement an optimal window size can be calculated for the present input signal. In this way an optimal window size can be maintained depending on the applied input signal.

Recursive formulation.

An alternative would be a recursive formulation for the TPBVP for optimal estimation. When using a recursive formulation to calculate the next estimate only the estimate at the present is used. With this formulation all information from the past is recursively present in the last calculated estimate. The solver will still need an observable system and a persistingly exciting input signal. But the size of the memory window will only depend on numerical aspects of the discrete error formulation, such as the number of discrete points necessary to approximate the state’s time derivative in the discrete error formulation.
Combining MPE with MPC.

It must be possible to combine MPE with nonlinear MPC. To do this, two consecutive windows have to be defined, one in the past and one in the future. The window in the future contains a TPBVP for optimal control and in the window in the past the TPBVP for optimal estimation is defined.

In Fig. (5.13) it is shown how measurement information available up to the present time is used to reconstruct the augmented state in the memory window. The filtered augmented state at the present time is used together with information about the desired trajectory, to calculate the optimal input signal over a finite prediction horizon to obtain optimal control.

The two TPBVPs can be solved separately or the boundary conditions at the present time can be combined to merge the problem into one optimal estimation/control problem.

![Figure 5.13: Combination of Model Past Estimation and nonlinear Model Predictive Control](image-url)
Chapter 6

Conclusion and Recommendations

The Neuro Morphic Solver (NMS) can solve a nonlinear Two Point Boundary Value Problem. When using the NMS in system identification problems occur with convergence speed. Due to the stiff character of such problems and due to the optimization algorithm used in the Neuro Morphic Solver the convergence to a solution is slow. Because of the structure of the NMS it can not be adapted to be used with any higher order optimization algorithm.

From the NMS a Direct Least Squares (DLS) solver is developed. This DLS solver performs better than the NMS. The DLS solver uses the Levenbergh-Marquardt optimization algorithm which is of semi-second order. Constraints can be implemented directly in the TPBVP. Linear constraints have been implemented and tested successfully.

With the DLS solver it was shown that a useful solution to the TPBVP for optimal estimation can only be derived when scaling the TPBVP. This was done by choosing large weighting matrices $W$ and $V$.

The DLS solver was tested with a simulation model based on an experimental setup at TUE. The solver shows to be robust for wrong initial parameters and quantization noise on measurement data. The results turned out to be good. The simulation model is for identification of an inertia and friction in an experimental setup at TUE. Results have shown that the friction model used in the model is not complete. Besides Coulomb friction, a viscous component is likely to be part of the occurring friction phenomena.

Model Past Estimation is a new estimation technique based on a solver for the optimal estimation problem, which can be defined as a nonlinear TPBVP. Any good solver for a TPBVP can be used in MPE. Here, the Direct Least Squares solver is used. MPE is developed as the counterpart of MPC in optimal control. Important aspects in MPE are window size, the sampling time, window shift time, solver accuracy and available calculation time. These aspects were discussed with respect to system identification and numerical aspects for the TPBVP solver used.

Simulations and experiments were performed and it is shown that MPE works good. A stable filtered estimate can be calculated using the principals of MPE. The usability of the MPE algorithm is restricted by the calculation time used by the solver for the TPBVP.
Conclusion and Recommendations

Further research should be done on:

- Extensive tests of the DLS solver in system identification problems. This should imply robustness for wrong initial parameter estimates and convergence speed.

- A selection has to be made in potentially usable solvers. They have to be tested on several identification problems such that they can be compared in accuracy and calculation speed.

- The DLS solver should be tested with nonlinear constraints.

- The effects of scaling the TPBVP should be further investigated. It is proven to be a useful tool in minimizing problem stiffness and thus calculation time.

- In Model Past Estimation a number of aspects have to be investigated such as:
  - Variable window length.
  - The best way to use previous window information in the next window. Especially in the situation when the maximum number of iterations is reached.
  - Optimal window size and discrete step size. Now this is done by trial and error.

- Exploring the possibilities of a combination of both Model Past Estimation and nonlinear Model Predictive Control.
Bibliography


[14] Bosch van den P.P.J., Klauw van der A.C., *Stochastische Systeemtheorie* syllabus TUE, 1994


51


Appendix A

TheSolver Toolbox

A.1 Intro

The solver as described in this thesis is available in the TheSolver Toolbox for MatLab. It is setup in such a manner that any general nonlinear Two Point Boundary Value Problem with nonlinear constraints can be solved. In this appendix the files in the toolbox are described briefly, this is not a user manual, but all files are supplied with a help text.

A.2 Contents

The NMS toolbox contains the following files:

fct.m The actual TPBVP.

dis.m This is the solver were parameters as window size, discrete stepsize, maximal number of iterations and the solver accuracy are needed.

mpe.m The Model Past Estimation algorithm.

show.m Shows results of the solver.
A.3 MatLab files

In this appendix the MatLab files for the rotational mass with friction is given as an example of how to implement the Neuro Morphic Solver.

A.3.1 fct.m

\%*** Two Point Boundary Value Problem for Optimal Estimation ****
\% date : 160596
\% name : rotational mass with coulomb friction
\% function call : fct4(Y,U,M,i,W,V,C,dt)
\% U : input
\% M : measurement
\% W, V, R : weight matrices
\% Y : augmented state vector + lagrange multipliers
\% C
\% dt : discrete time step
\%******************************************************************************

function fun=fct4(Y,U,M,i,W,V,C,dt);

x=Y([1:4],i);
l=Y([5:8],i);

a=50;
ts=(exp(a*x(2))-1)/(exp(a*x(2))+1);
dts=(2*a*exp(a*x(2)))/(exp(a*x(2))+1)\^-2;

Chi=[ x(2);
(U(i)-x(4)*ts)/x(3);
0;
0];

dChidx=[0 1 0 0;
0 -x(4)*dts/x(3)-(U(i)-x(4)*ts)/(x(3)^2)-ts/x(3);
0 0 0 0;
0 0 0];

iW=[1/W(1,1) 0 0 0; 0 1/W(2,2) 0 0; 0 0 1/W(3,3) 0; 0 0 0 1/W(4,4)];

%constraints

g=x-n;
I=find(g>=0);
g(I)=zeros(size(I));
fun=[ Chi+iW*1;
-dChidx'*1-C'*V*(M(:,i)-C*x)-g'*P*g];
A.3.2 dls.m

%% ***** The Direct Least Squares solver for Optimal Estimation *****
%% optimization by least squares routine from matlab toolbox
%% uses: lsq4.m (least squares formulation)
%% fct4.m (TPBVP)
%% show4.m(shows results)
%% problem: rotational mass with Coulomb friction

K=menu('The Direct Least Squares solver','Initialize ...','Reactivate ...');
if K==1
    \%
    load data
    load RotSysWNP.mat;
    \%
y=Mq(:,[1,2]);
    clear y;
    \%
    parameter schatting
    J = 0.05;
    fw= 0.1;
    \%
    aantal states n, discrete stapjes N
    dI=16;
    Imax=1024;
    I=1:dI:Imax;
    n=8;N=length(I);
    X0=zeros(n,N);
    \%
    weegmatrices geven vertrouwen aan
    \%
    W voor modelvergelijking
    \%
    V voor meetvergelijking
    \%
    R voor beginschatting
    W=eye(n/2);
    V=1;
    R=eye(n/2); R(3,3)=0;R(4,4)=0;
    \%
    sample data
    disp('Get data and sample it to fit into network');
    sd2;
    X0([1,2],:)=M;
    M=M(1,:);
MatLab files

\% Startwaarden voor Y
X0([3,4],:)=q0*ones(1,N);

\% begin schatting q0:
q0=[X0([1:2],1);q0];
    end;
X0=X0(:,);
if K==2
X0=X;
end;

options(1)=1;
options(2)=1e-4;
options(3)=1e-4;
X=leastsq('lsq4',X0,options,[],T,N,n,W,V,R,q0,U,M,C);

makey;
Y=y;
Tijd=T;
show4;
A.3.3 mpe.m

\%----------------------------------------------------- Model Past Estimation -------------------------------
\% using the Direct Least Squares Solver for TwoPointBoundaryValueProblems
\%----------------------------------------------------- softogh =-------------------------------------------------

K=menu('MPE','Start MPE','Reactivate');
if K==1

\% Goal error criterium
GoalError=5e-1;
Error=5e-1;
maxit=15e3;
wsf=5;

\% load data RotSysWNP
load dataRotSysQ4.mat;

\% sample data according to I
dI=10;
I=1:dI:500;
n=8;
N=length(I);
Y=zeros(n,N);

\% weegmatrices geven vertrouwen aan
\% W voor modelvergelijking
\% V voor meetvergelijking
\% R voor beginschatting
W=1e6*eye(n/2);
V=1e6;
P=W; P(1,1)=0; P(2,2)=0;
R=1*eye(n/2); R(3,3)=0; R(4,4)=0;

\% sample data
disp('Get data and sample it to fit into network');
sd3;
T=T';
\% Y(:,2)=M;
Y([1:2],:)=M([1,2],:);
M=M(:,:);
Tijd=T;
\% schatting parameters
\% massaatragheid en droge wrijving
Je = .05;
fwe = 1;

\% Startwaarden voor Y
Y(3,:) = (Je) * ones(1,N);
Y(4,:) = (fwe) * ones(1,N);

w = 0; State = [] ; I t = [] ; Time = [] ; X t = [] ; shiftwindow = 1;

disp('Least squares optimization routine activated ... ');
Y = nms(Y, U, M, T, W, V, P, R, Error, maxit, 1);
dI = wsf * dI;
end; \% menu option 1

if K = 2
w = w - 1;
I = I - dI;
Y = Yold; end;

\% start MPE

while shiftwindow;

disp(['window: %3.0f ', w], ... 
sprintf('| Goal : %5.3e', GoalError), ... 
sprintf('| Error: %5.3e', Error), ... 
sprintf('| N : %2.0f', N), ... 
sprintf('| I : %4.3f', mean(Y(3,:)) ... 
sprintf('| fw: %4.3f', mean(Y(4,:)))]);
disp('MPE in progress ... ');
\% increase accuracy by reducing error criterium
if Error > GoalError
    Error = Error / 2;
end;
if Error < GoalError
    Error = GoalError;
end;
w = w + 1;
I = I + dI;

\% shift state
Y(:, [I:N-wsf]) = Y(:, [wsf+1:N]);
Y(:, [N-wsf:N]) = Y(:, N-wsf) * ones(size(Y(1,[N-wsf:N])));
\% lagrange multipliers should be zero
Y([3:8],:)=zeros(4,N);
\% parameters should be constant
Y(3,:)=mean(Y(3,:))*ones(size(Y(3,:)));
Y(4,:)=mean(Y(4,:))*ones(size(Y(4,:)));

\% sample data
sd3;
M=M(:,:);
Y(:,1)=M;
\%Y([1:2],:)=M;

[Y,numit,Jnew]=nms(Y,U,M,T,W,V,P,R,Error,maxit,1);
\% update stats
It=[it numit];
E=[E Jnew];
State=[State Y(:,N)];
Xt=[Xt M(:,N)];
Time=[Time T(N)];
Yold=Y;
\% stop shift
if w==40 shiftwindow=0;
end;
end;
save mpedata.mat;