Development and Evaluation of a new Predictive Control Algorithm for the Control of Multivariable Systems

Roland Leslie John Bolton

A project report submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in fulfillment of the requirements for the degree of Master of Science in Engineering

Johannesburg, 1994

Declaration

I declare that this project report is my own unaided work. It is being submitted for the degree of Master of Science in Engineering at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other university.

Roland Leslie John Bolton

17 day of January, 1994

Abstract

A new predictive control algorithm is developed for the computer control of multivariable systems. In addition to the integral action inherent in the algorithm, constraints on the input variables of the system can be specified.

This project investigates the development and evaluation of the algorithm. Extensive MATLAB simulations are used to explain concepts and verify the functionality of the algorithm. The principles for achieving a stable and robust control system are addressed and simulated examples are given to emphasize the principles. A detailed qualitative approach for selecting the system parameters is suggested.

The work done in this project shows that it is possible to design a stable, robust multivariable control system, that allow for constraints on the process input variables.

Acknowledgements

I am deeply indebted to my Father for his constant encouragement and support. I would also like to thank my supervisor, Prof Ian MacLeod for his knowledge and experience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multi-Step Predictive Control, better known in the control literature as Generalised Predictive Control, deals with the idea of using a model of a plant to predict plant outputs based on past, present and future control inputs, and past outputs. Before predictive control is discussed, a brief overview of computer control should be presented in order to gain insight into how this technology will be implemented in the real world.

1.1 Background to Computer Control

For an indepth discussion on the background to computer control, the reader is referred to the book by Åström and Wittenmark [2].

The idea of using digital computers as components in control systems emerged around 1950. Applications in missile and aircraft control were investigated first. Studies showed that there was no potential for using the general-purpose digital computers that were available at that time. The computers were too big, they consumed too much power, and they were not sufficiently reliable. For this reason special-purpose computers – digital differential analysers (DDA) – were developed for the early aerospace applications. The major developments in computer control occurred in the process industries.

1.2 Why use computers ?

With the decrease in cost and increase in reliability of the digital computer over the years, there is a greater demand for better control systems.
The added benefits (to name a few) of computer control are:

- Computers can be used to design **adaptive controllers** which 'adapt' to the system. In other words the system predicts the model of the plant on-line, which is used in the control algorithm. This strategy is exploited in the Process Control Industry, which is renowned for its poorly modelled systems.
- Systems can be made **more flexible**, as decisions can be made on-line.

The achievements of modern control theory are well-known. Successful applications to aerospace guidance problems are remarkable. However, the implementations of such techniques for industrial control problems are not so successful. Industrial processes are quite different, they are highly multivariable systems, perturbations affect

the plant structure more often than the measurable variables. Modelling of such systems is often difficult and often the models available are inaccurate. Industrial processes also have the undesirable characteristic of transport delays. [2]

Hence, existing control techniques appear to be insufficient. Multi-Step Control will hopefully prove to be a superior control strategy, when applied to certain control problems.

1.3 Introduction to Multi-Step Predictive Control

Multi-Step Predictive Control (MSC) is a computer based control algorithm, which as its name suggests, is a predictive control technique. This control technique, as far as the control literature is concerned, is often called Model Predictive Control ([3],[7],[4],[9]).

The term 'predictive' is used because MSC makes use of a model of the plant to predict future outputs of the process based on assumed future control actions. Although this might seem potentially dangerous [1], feedback is incorporated into the structure of MSC, which will compensate for model mismatch.

Model Predictive Control:
Model Predictive Control is based on four key concepts:

- **Predictor Model**
  A model of the process which needs to be controlled is incorporated into a predictive algorithm. This model is used to predict the optimum control action that must be taken at each step.
- **Reference Trajectory**
  This trajectory, which is reconstructed at each step, describes the ideal closed loop response the system must track. In simpler terms, this trajectory describes the path the process output must follow, in order to return to the specified set-point. This path can be constructed in such a way that the rise time and set-point are both specified.
- **Prediction Horizon**
  This parameter describes the number of steps the predictive controller must use in its predictive algorithm.
- **Constraint**
  It is possible to specify the constraints on the input and output variables of the process. This facility is the most impressive feature of Model Predictive Control schemes.
- **Cost function**
  This function describes the extent of mismatch between the specified reference trajectory and the predicted close loop response. Ideally, this mismatch should be minimised, so that the predicted closed loop response corresponds as closely as possible to the reference trajectory.

[1] The reference model and actual process model may be different, leading to poor control

Multi-Step Control incorporates the above defined concepts into a computer based control algorithm. The exact details of how the predictive algorithm works will be covered in later chapters. The main features of this control scheme will be outlined now.

The MSC algorithm attempts to find the optimum control action that must be taken at each step. The optimum control action is established in the following way:

1. Measure the output of the process.
2. Set-up reference trajectory. This trajectory must be defined based on the following parameters:
   (a) The present output.
   (b) The closed loop rise time.
   (c) Set-point of output.
   (d) Prediction horizon.

   This trajectory will 'steer' the system to the set point.
3. Find the optimum control action that must be taken, by minimising the cost function. The algorithm which predicts the optimum control can incorporate the constraints on the input/output parameters.
4. Execute the optimum control action.
5. Wait for beginning of next clock cycle.
6. Re-do algorithm. (i.e. Goto step 1)

The above procedure gives an overview of the Multi-Step Control algorithm.

Figure 1.1 shows how the optimum prediction is solved.

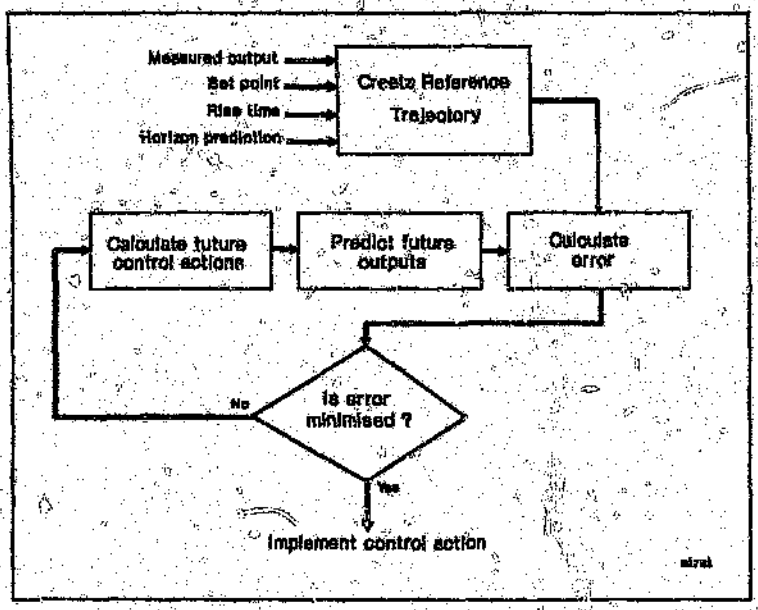
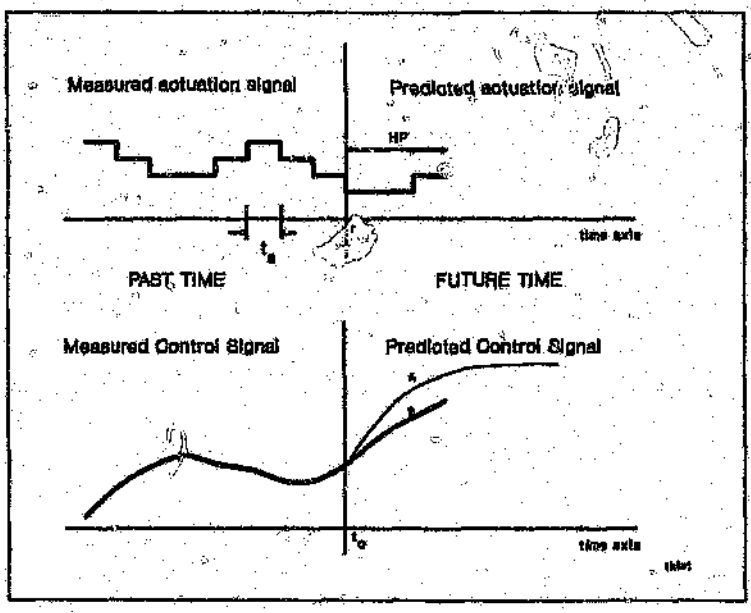


Figure 1.1: Schematic diagram of how to find optimum control action.

The time history of Multi-Step Control is given in Figure 1.2.



HP = Prediction Horizon
t_o = Present time
t_s = Sampling period
A = Reference trajectory
B = Predicted output

Figure 1.2: Time history of Multi-Step Predictive Control.

1.4 Computer Control Architecture

The architecture used to implement the Multi-Step Predictive Control System is shown in Figure 1.3. For an indepth discussion of different computer control architectures, the reader should refer to texts such as Computer Controlled Systems [2].

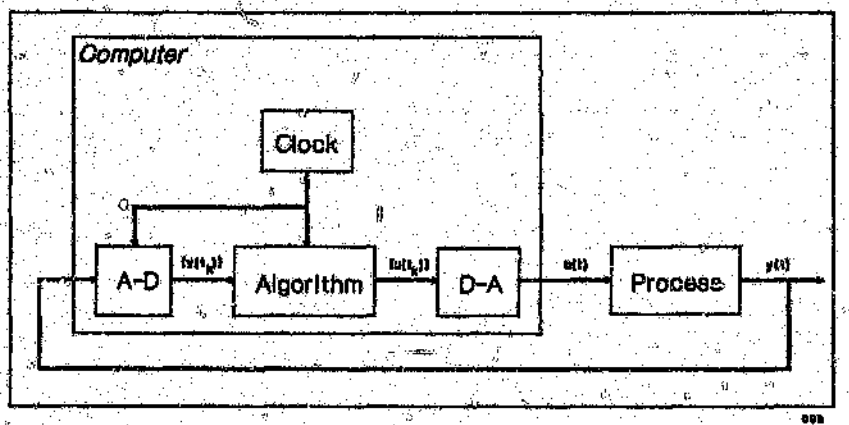


Figure 1.3: Schematic diagram of a computer controlled system.

Where: u(t) = Actuation signal to plant.
y(t) = Plant output.
$y(t_k)$ = Sampled signal y(t).
$u(t_k)$ = Algorithm output signal.
t_s = Sampling period.
k = Sample number. Hence $t_k = k * t_s$
A-D = Analog to digital converter.
D-A = Digital to Analog converter.

System operation:
The process to be controlled is interfaced to a digital computer via A-D and D-A converters.

The controlled variable (y(t)) of the process is measured using an electrical transducer. The output of the transducer, which is a continuous time signal, is fed into a digital computer via analog to digital converters.

The digital computer is controlled via a clock. At the beginning of each clock cycle the following functions are performed:

- *Read new controlled variable* of the process
  This is done at time t_k. The controlled variable is $y(t_k)$.

- Once the new controlled variable is imported into the computer, it is necessary to *calculate the actuation signal required for this clock cycle.* This operation is performed by the algorithm which is pre-programmed into the computer. This algorithm can be programmed to perform any desired control strategy, for example: PID control with a facility to tune the PID parameters (in real time) to best suit the process. When implementing the Multi-Step Control system, the algorithm is programmed with the MSC algorithm.

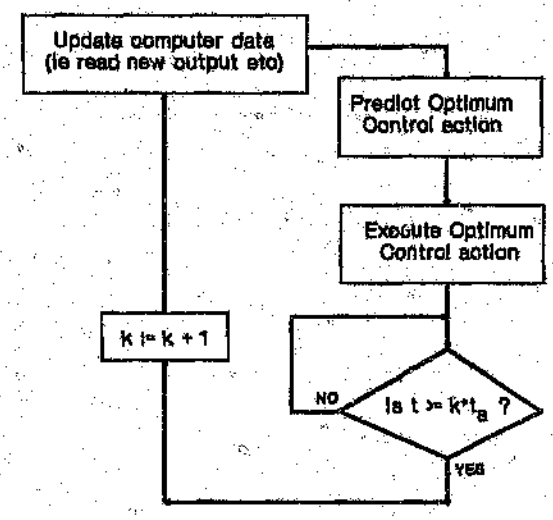
  Once the algorithm has completed its computation, which is at time $t_k + \delta$ seconds, the new actuation signal, $u(t_k + \delta)$ is ready to be output to the actuators.

  Note: The time lag of δ seconds, required for computing the new control action, is usually negligible (i.e. $\delta \ll T$).
  However, this time lag must never be greater than the sampling period. If it does exceed the sampling period, then the system will fail.

- The new control variable is *output to the digital to analog converter.* This D-A converter is used to drive the actuators of the process. The control variable ($u(t_k)$) is held constant until the next cycle, which requires a new actuation signal.

This cycle is repeated at the start of each clock cycle, which lasts for a duration of T seconds. Therefore, it is important to ensure that all the above operations are executed before the clock cycle ends.

The flow chart describing the cycle is shown in Figure 1.4.



t = Continuous time variable
t_s = Sampling period
k = Sampling index

Figure 1.4: Outline of Algorithm

The above completely describes the architecture used in Multi-Step Control.

Constraints on system
The following criteria must be adhered to in order to achieve a working system:

- **Sampling period**
  Nyquist sampling rate: The sampling frequency (f_s) must be selected in such a way that $f_s > 2f_h$ where f_h is the highest frequency of the signal being measured.

- **Computation time of Algorithm**
  As already mentioned, the time delay in calculating the control action to be taken must be less than the sampling period. Usually this time delay is very small, and really depends on the speed of the computer used. In other words, if the delay is too long, use a faster computer.

The required *computing power is dictated by the process dynamics*. The dynamics of the process dictate what sampling period should be used, which in turn dictates the computing power required. This is an important issue to bear in mind when designing a computer controlled system.

1.5 Research aim

The purpose of this research project is to derive a predictive control algorithm which can be used to control multivariable systems. The algorithm must be as flexible as possible, and must allow for constraints, which is the major incentive for using these control techniques.

1.6 Research procedure

An overview of the predictive control algorithm is given in section 1.3, which is followed by a description of the architecture used to implement this control algorithm.

The next step is to derive the algorithm:
Initially only single variable systems are explored, with the intention of setting up a basis for the principles on which the multivariable predictive algorithm operates. This algorithm is then extended to cater for multivariable systems, which is the main theme of this project.

The algorithm is evaluated using simulations, implemented as MATLAB programs. These programs simulate the Multi-Step Control system using a plant model.

Other issues, such as the stability and robustness are also touched on, however this is not the main theme of the project and is not considered in great detail.

1.7 Overview of the project report

This project report is structured as follows:

- **Chapter 2: Multi-Step Control (SISO systems)**
  The predictive control scheme is first presented for SISO systems. This presentation gives the reader a deeper understanding of the algorithm used. The analysis in this chapter forms a foundation for the predictive control algorithm.

- **Chapter 3: Multi-Step Control (MIMO systems)**
  The work in Chapter 2, forms a foundation for the predictive algorithm, which is now extended to handle MIMO systems. This chapter presents the transition from the SISO to the MIMO algorithm.

- **Chapter 4: Evaluation of Multi-Step Control Algorithm**
  This chapter evaluates the functionality and efficiency of the algorithm. Issues such as memory and computational effort are investigated.

- **Chapter 5: Stability and Robustness issues**
  This chapter investigates the method of designing stable and robust control systems. The aim of this chapter is to present a philosophy for designing a predictive control system.

  Where possible, examples are used to illustrate the design philosophy.

Chapter 2

Multi-Step Control (SISO system)

The algorithm used to control SISO systems is presented in this chapter. This SISO algorithm is used as a basis for controlling MIMO systems, and is extended to the MIMO version in a later chapter.

The algorithm is presented in the following way:

- The model representation describing the predictor model is presented in section 2.1.
- The algorithm describing the reference trajectory is presented in section 2.2.
- The Predictor Equation is presented in section 2.3.
- The Cost function is presented in section 2.4.
- The algorithm used to find the set of optimum inputs is presented in section 2.5.
- This chapter concludes with a summary, which integrates all the building blocks into the MSC algorithm. This is presented in section 2.6.

2.1 Model Representation

Previous predictive control techniques, such as the IDCOM [1] algorithm ([4],[9],[6]), use a truncated FIR [2] model as a predictor model. Although these models are adequate in many situations, there are many drawbacks in using this model representation:

- This model representation often requires many coefficients to adequately describe the dynamics of the process.
- This model representation is not able to model unstable systems, or systems that have pure integrators in their transfer functions. These limitations are not all that important, because the majority of the systems that need control do not have these dynamics.
  There are exceptions:
  1. There are many chemical rectors that are designed to be unstable (eg. exothermic reactions), so that the yield of the process can be increased.
  2. An example of a process that has the dynamics that exhibit that of an integrator: Tank being filled with a liquid.

It is for these re[illegible] that the finite impulse response model representation is not used in the MSC [illegible]gorithm.

The FIR model can be described equally well using the infinite impulse response (IIR) model [4]. An important point is that the IIR representation requires a far lower order model to describe the equivalent model of the FIR model representation. In addition to this, the IIR model representation is not restricted when considering systems that have complications such as a pure integrator in the transfer function.

The transfer function of an IIR model in discrete time is written:

$$A(q^{-1})y(t) = B(q^{-1})u(t) \tag{2.1}$$

Note: The variable q represents the shift operator, which is simply a shorthand notation used to represent time shifts.
Example: $x(t+n) = q^n x(t)$ and $x(t-n) = q^{-n}x(t)$

The equations describing $A(q^{-1})$ and $B(q^{-1})$ are:

$$A(q^{-1}) = A_1 + A_2q^{-1} + A_3q^{-2} + ... + A_{n+1}q^{-n} \tag{2.2}$$

$$B(q^{-1}) = B_1 + B_2q^{-1} + B_3q^{-3} + ... + B_{n+1}q^{-n} \tag{2.3}$$

In general $B_1 = 0$ (zero order hold is being used) and $A_1 = 1$.

The above notation is used in the MSC algorithm to describe the model of the process.

[1] IDentification and COMmand
[2] Finite Impulse Response

Example:
The following example shows how the dynamics of a continuous time transfer function is expressed in discrete time. Consider the following continuous time transfer function, which is written in the s domain:

$$P(s) = \frac{10(s+1)}{(s+3)(s^2+s+1)} \tag{2.4}$$

If the sampling period of the computer controlled system is set to 1 second, then the plant model in discrete time is (refer to `model.m`):

$$(1-0.8357q^{-1}+0.4070q^{-2}-0.0183q^{-3})y(t) = (0+2.0778q^{-1}+0.0742q^{-2}-0.3086q^{-3})u(t) \tag{2.5}$$

Illuminating the shift operator from equation 2.5 results in equation 2.6.

$$[1 \quad -0.8357 \quad 0.4070 \quad -0.0183]\begin{bmatrix} y(t) \\ y(t-1) \\ y(t-2) \\ y(t-3) \end{bmatrix} = [0 \quad 2.0778 \quad 0.0742 \quad -0.3086]\begin{bmatrix} u(t) \\ u(t-1) \\ u(t-2) \\ u(t-3) \end{bmatrix} \tag{2.6}$$

Equation 2.6 describes the same dynamics as the continuous time transfer function (Equation 2.4).

Consider the step response of the system, shown in Figure 2.1.

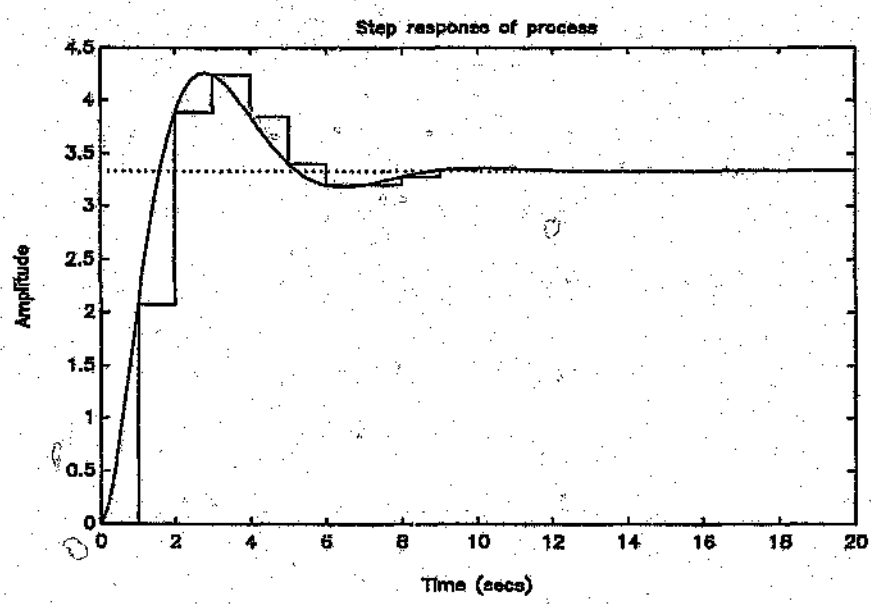


Figure 2.1: Step response of continuous and discrete time systems.

Figure 2.1, which is generated by the program `model.m`, shows that the discrete time system gives the same dynamic response as the continuous time system.

If a disturbance is incorporated into the transfer function model, then the transfer function can be rewritten as follows:

$$A(q^{-1})y(t) = B(q^{-1})u(t) + A(q^{-1})d(t) \tag{2.7}$$

Where the variable d(t) describes the disturbance input.

2.1.1 SISO system (without integral action)

This model representation is discussed in section 2.1.

The formulae for the model is:

$$y(t) = \frac{B(q^{-1})}{A(q^{-1})}u(t) + d(t) \tag{2.8}$$

The values of $A(q^{-1})$ and $B(q^{-1})$ are:

$$A(q^{-1}) = A_1 + A_2q^{-1} + A_3q^{-2} + ... + A_{n+1}q^{-n} \tag{2.9}$$

$$B(q^{-1}) = B_1 + B_2q^{-1} + B_3q^{-3} + ... + B_{n+1}q^{-n} \tag{2.10}$$

2.1.2 SISO system (with integral action)

When a control strategy is applied to process control problems, it is always advantageous to have integral action incorporated into the algorithm. The reason for this is that disturbance offsets (which occur often in process control) can be 'nulled out' using integral action.

The transfer function of a SISO system with an added disturbance is written:

$$y(t) = \frac{B(q^{-1})}{A(q^{-1})}u(t) + d(t) \tag{2.11}$$

If both sides of Equation 2.11 are operated on by $(1 - q^{-1})$, then Equation 2.12 is obtained:

$$(1 - q^{-1})y(t) = \frac{(1 - q^{-1})B(q^{-1})}{A(q^{-1})}u(t) + (1 - q^{-1})d(t) \tag{2.12}$$

If the disturbance is constant (which is assumed), then

$$(1 - q^{-1})d(t) = d(t) - d(t-1) = 0 \tag{2.13}$$

Using the above assumption (Equation 2.13), the Equation 2.12 simplifies to:

$$(1 - q^{-1})y(t) = \frac{(1 - q^{-1})B(q^{-1})}{A(q^{-1})}u(t) \tag{2.14}$$

Using the following notation,

$$\Delta y(t) = (1 - q^{-1})y(t) \tag{2.15}$$

$$\Delta u(t) = (1 - q^{-1})u(t) \tag{2.16}$$

allows the transfer function to be written as:

$$\Delta y(t) = \frac{B(q^{-1})}{A(q^{-1})}\Delta u(t) \tag{2.17}$$

This transfer function gives a relationship between the incremental change in the input ($\Delta u(t)$) and the rate of change in the output ($\Delta y(t)$).

$$\Delta u(t) \longrightarrow \boxed{B(q^{-1})/A(q^{-1})} \longrightarrow \Delta y(t)$$

This transfer function notation leads to integral action in the controller, as the rate of change of the output is being controlled and not the specific output value.

2.2 Setting-up the Reference Trajectory

The reference trajectory is intended to provide the system with a 'steering' mechanism. The reference trajectory specifies the way the system must respond. Typical specifications are:

- Overshoot.
- Rise-time of the response.
- Steady state value.

Figure 2.2 gives a graphical explanation of the reference trajectory.

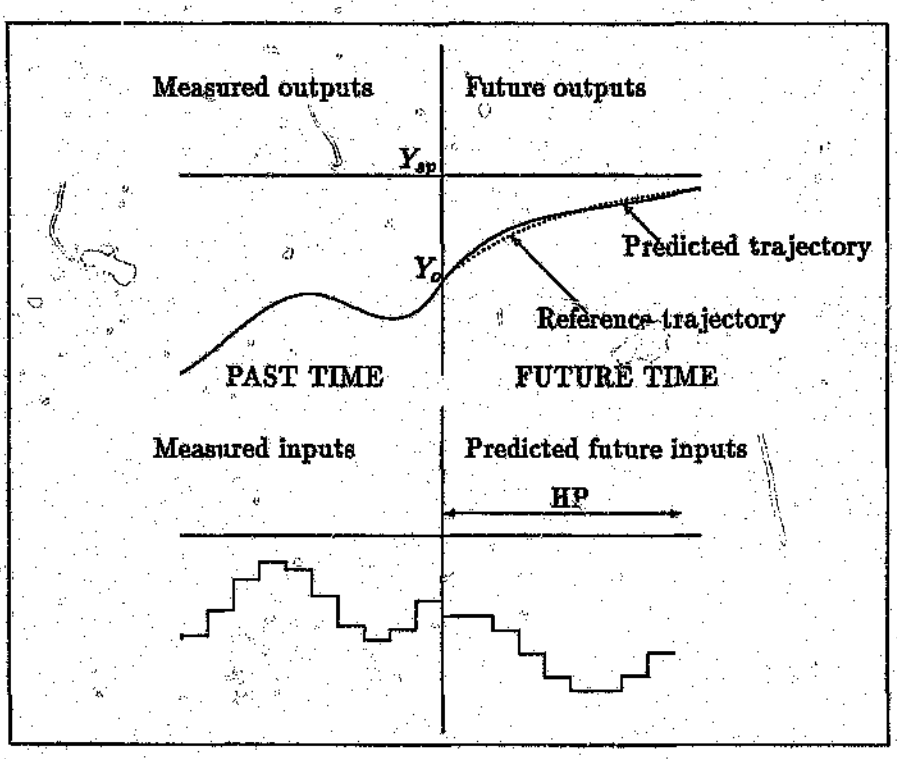


Figure 2.2: Explanation of reference trajectory.

Explanation:
The output of the process at the present time ($t = t_o$), is given by the constant Y_o. The desired set-point of the output is Y_{sp}. The reference trajectory guides the system, according to some criteria, to this set-point. This criteria may state that there must be no overshoot and that there must be a fixed rise time associated with the transient. In this particular example, there is no overshoot specified.

Setting-up the trajectory
In many applications a simple first order response is sufficient.

Specifications: First order response
Number of terms in trajectory = HP
Rise time = τ seconds

Algorithm:
The algorithm is based on a recursive equation.
The starting point is given by:

$$Y^*(t_o) = Y_o \tag{2.18}$$

The recursive equation is:

$$Y^*(t_o + i) = \alpha Y^*(t_o + i - 1) + (1 - \alpha)Y_{sp} \qquad (i = 1..\text{HP}) \tag{2.19}$$

The variable α is related to the rise time:

$$\alpha = e^{-T/\tau} \tag{2.20}$$

Where: T = Sampling period.
τ = Rise time of closed loop response.

The above algorithm is coded in the program `tref.m`, which is used as a building block in the main MSC algorithm.

$$(Y_o, Y_{sp}, \mathrm{HP}) \longrightarrow \boxed{\texttt{tref.m}} \longrightarrow Y^* \longrightarrow \boxed{(1-q^{-1})} \longrightarrow \Delta Y^*$$

If integral action is required, then the reference trajectory must specify the required change in the output. Hence it is necessary to find the change in the Y^* trajectory at each step. This is done by operating on the predicted reference trajectory (Y^*) by $1-q^{-1}$.

2.3 The Predictor Equation

This equation provides a mechanism for predicting future plant outputs for time $t = t_o+1...t_o+\mathrm{HP}$

This equation is a very useful building block in the MSC algorithm, because it provides a 'black box' equation, which predicts the trajectory of the output for future time. This equation is therefore a critical component of the optimisation algorithm, which is discussed later in this chapter.

The implementation of this equation relies on the following information:

- Past inputs to the plant.
- Past outputs of the plant.
- Predictor model of the plant.
- Predicted future inputs to the plant.

Appendix A derives this equation for the SISO and the MIMO case.
The derivation of this equation relies on the transfer function model, which has already been presented.

This sub-section describes the notation of the predictor equation, when applied to SISO systems. The reader should refer to appendix A for an indepth derivation of this equation.

Notation of Predictor Equation (SISO system, no integral action)

The predictor equation for the SISO case, without integral action is defined as:

$$\hat{\mathbf{Y}} = \mathbf{Q}\mathbf{Y}_{past} + \mathbf{W}\mathbf{U}_{pred} + \mathbf{R}\mathbf{U}_{past} \qquad (2.21)$$

Where: $\mathbf{Q}$ = Matrix of dimensions: HP × n

$\mathbf{W}$ = Matrix of dimensions: HP × HP

$\mathbf{R}$ = Matrix of dimensions: HP × (n-1)

$\mathbf{Y}_{past}$ = Vector of past inputs to plant.

$$= \begin{bmatrix} y(t) \\ y(t-1) \\ \vdots \\ y(t-n+1) \end{bmatrix}$$

$\mathbf{U}_{pred}$ = Vector of predicted future inputs to plant.

$$= \begin{bmatrix} u(t+\mathrm{HP}-1) \\ u(t+\mathrm{HP}-2) \\ \vdots \\ u(t) \end{bmatrix}$$

$\mathbf{U}_{past}$ =Vector of past inputs to plant.

$$= \begin{bmatrix} u(t-1) \\ u(t-2) \\ \vdots \\ u(t-n+1) \end{bmatrix}$$

$\hat{\mathbf{Y}}$ =Vector of predicted future outputs.

$$= \begin{bmatrix} \hat{y}(t+\mathrm{HP}) \\ \hat{y}(t+\mathrm{HP}-1) \\ \vdots \\ \hat{y}(t+1) \end{bmatrix}$$

The matrices Q,W and R are derived based on knowledge of the predictor model and the sampling rate.

Notation of Predictor Equation (SISO system, with integral action)

The predictor equation for the SISO case, with integral action is defined as:

$$\Delta\hat{\mathbf{Y}} = \mathbf{Q}\Delta\mathbf{Y}_{\text{past}} + \mathbf{W}\Delta\mathbf{U}_{\text{pred}} + \mathbf{R}\Delta\mathbf{U}_{\text{past}} \tag{2.22}$$

Where: $\mathbf{Q}$ = Matrix of dimensions: HP × n
$\mathbf{W}$ = Matrix of dimensions: HP × HP
$\mathbf{R}$ = Matrix of dimensions: HP × n
$\Delta\mathbf{Y}_{\text{past}}$ = Vector of past changes in outputs of plant.

$$= \begin{bmatrix} \Delta y(t) \\ \Delta y(t-1) \\ \vdots \\ \Delta y(t-n+1) \end{bmatrix}$$

$\Delta\mathbf{U}_{\text{pred}}$ = Vector of predicted future incremental inputs to plant.

$$= \begin{bmatrix} \Delta u(t+\text{HP}-1) \\ \Delta u(t+\text{HP}-2) \\ \vdots \\ \Delta u(t) \end{bmatrix}$$

$\Delta\mathbf{U}_{\text{past}}$ = Vector of past incremental inputs to plant.

$$= \begin{bmatrix} \Delta u(t-1) \\ \Delta u(t-2) \\ \vdots \\ \Delta u(t-n+1) \end{bmatrix}$$

$\Delta\hat{\mathbf{Y}}$ = Vector of predicted changes in future outputs.

$$= \begin{bmatrix} \Delta\hat{y}(t+\text{HP}) \\ \Delta\hat{y}(t+\text{HP}-1) \\ \vdots \\ \Delta\hat{y}(t+1) \end{bmatrix}$$

The matrices Q,W and R are derived based on knowledge of the predictor model and the sampling rate.

2.4 The Cost Function

The cost function is an equation which provides a measure of how different the reference and predicted trajectory are. This function provides a quantitative describe of this mismatch, which is important when finding the set of optimum inputs. The set of optimum inputs is found when this mismatch is lowest. In other words, when the cost function is at its minimum.

The cost function consists of two components:

- The first component provides a measure of the tracking error.

- The second component provides a measure of the deviation of the actuation signal. This component cannot be omitted, because good tracking is sometimes obtained at the expense of large deviations of the actuation signal.

The cost function is defined as follows:

$$\text{Cost} = \sum_{k=1}^{HP} \left[\underbrace{\lambda(k)^2(\Delta Y^*(t_o+k) - \Delta\hat{Y}(t_o+k))^2}_{\text{1st component}} + \underbrace{\beta(k)^2(\Delta u(t_o+k))^2}_{\text{2nd component}} \right] \tag{2.23}$$

Where: $\lambda(k)$ = Weighting variable for tracking error at time $t = t_o + k$
$\beta(k)$ = Weighting variable for change in actuation signal at time $t = t_o + k$
ΔY^* = Change in reference trajectory.
$\Delta\hat{Y}$ = Predicted change in output trajectory.
Δu = Future incremental inputs.

2.4.1 Weighting variables

The cost function has two sets of weighting variables, λ and β. These weighting variables describe which portion of the curve is more important. For example: If the steady state portion of the reference trajectory is more important than the transient portion, then the $\lambda(k)$ variables must be higher over the steady state portion than over the transient portion. This will result in errors occurring over the steady state portion to have a higher 'cost' than over the transient portion.

The selection of these weighting variables is critical when considering issues such as robustness and stability. Refer to the chapter on Robustness and Stability for a discussion of this issue.

2.4.2 Cost function in quadratic form

This sub-section shows how the cost function is expressed in quadratic form. This transformation is necessary because a quadratic optimisation routine is used to find the set of optimum inputs.
The predictor equation is written:

$$\Delta\hat{\mathbf{Y}} = \mathbf{Q}\Delta\mathbf{Y}_{\text{past}} + \mathbf{W}\Delta\mathbf{U}_{\text{pred}} + \mathbf{R}\Delta\mathbf{U}_{\text{past}} \tag{2.24}$$

Let $\Delta\hat{\mathbf{Y}}_1$ represent the predicted change in the output, assuming that the input signal to the process is unchanged (ie elements in $\Delta\mathbf{U}_{\text{pred}}$ set to zero).

$$\Delta\hat{\mathbf{Y}}_1 = \mathbf{Q}\Delta\mathbf{Y}_{\text{past}} + \mathbf{R}\Delta\mathbf{U}_{\text{past}} \tag{2.25}$$

Substituting Equation 2.25 into 2.24 results in the following:

$$\Delta\hat{\mathbf{Y}} = \Delta\hat{\mathbf{Y}}_1 + \mathbf{W}\Delta\mathbf{U}_{\text{pred}} \tag{2.26}$$

Now define the error variable, $\mathbf{Y}_e$, which represents the error between $\Delta\hat{\mathbf{Y}}_1$ and $\Delta\mathbf{Y}^*$ as follows:

$$\mathbf{Y}_e = \Delta\mathbf{Y}^* - \Delta\hat{\mathbf{Y}}_1 \tag{2.27}$$

The Cost Function (Equation 2.23) is expressed in matrix form as follows:

$$\begin{aligned} \text{Cost} &= [\gamma \mathbf{Y}_e - \gamma W(\Delta U_{\text{pred}})]^T[\gamma \mathbf{Y}_e - \gamma W(\Delta U_{\text{pred}})] \\ &\quad + [\beta(\Delta U_{\text{pred}})]^T[\beta(\Delta U_{\text{pred}})] \qquad (2.28) \\ &= [\gamma \mathbf{Y}_e - \gamma W(\Delta U_{\text{pred}})]^T[\gamma \mathbf{Y}_e - \gamma W(\Delta U_{\text{pred}})] \\ &\quad + (\Delta U_{\text{pred}})\beta^T\beta(\Delta U_{\text{pred}}) \qquad (2.29) \\ &= (\Delta U_{\text{pred}})^T[[\gamma W]^T[\gamma W] + \beta^T\beta](\Delta U_{\text{pred}}) \\ &\quad - 2(\gamma W)^T(\gamma \mathbf{Y}_e)(\Delta U_{\text{pred}}) + (\gamma \mathbf{Y}_e)^T(\gamma \mathbf{Y}_e) \qquad (2.30) \end{aligned}$$

Where the γ and β matrices are diagonal matrices with diagonal elements consisting of the weighting elements $\lambda(k)$ and $\beta(k)$.

In order to improve the notation, the following constants are defined:
Let:

$$P = (\gamma W)^T(\gamma W) + \beta^T\beta \qquad (2.31)$$

and

$$C = -2(\gamma W)^T(\gamma \mathbf{Y}_e) \qquad (2.32)$$

This quadratic function (Equation 2.30) is reduced to:

$$\text{Cost} = (\Delta U_{\text{pred}})^T P(\Delta U_{\text{pred}}) + C(\Delta U_{\text{pred}}) + (\gamma \mathbf{Y}_e)^T(\gamma \mathbf{Y}_e) \qquad (2.33)$$

Equation 2.33, which is expressed in quadratic form, is identical to the previously defined equation (2.23). However, this equation can be simplified even further. The last term in this equation is not a function of $\Delta \mathbf{U}_{\text{pred}}$ and is therefore constant as far as the optimisation routine is concerned. If this term is removed from the cost function, the minimisation of this function will still result in the same optimum input.

The new cost function is therefore:

$$\text{Cost} = (\Delta U_{\text{pred}})^T P(\Delta U_{\text{pred}}) + C(\Delta U_{\text{pred}}) \qquad (2.34)$$

The constants are:

$$P = (\gamma W)^T(\gamma W) + \beta^T\beta$$

$$C = -2(\gamma W)^T(\gamma \mathbf{Y}_e)$$

$$\gamma = \text{diag}[\lambda(1), \lambda(2), ..., \lambda(\text{HP})]$$

$$\beta = \text{diag}[\beta(1), \beta(2), ..., \beta(\text{HP})]$$

$$\mathbf{Y}_e = \Delta \mathbf{Y}^* - \Delta \hat{\mathbf{Y}}_1$$

2.5 Finding the set of optimum inputs

The cost function was derived in section 2.4. This function gives a quantitative measure of the error between the predicted change in the output trajectory and the change in the reference trajectory. The set of optimum inputs are predicted when this error is minimised. This section explains how these inputs are found.

The set of optimum inputs are found by minimising the cost function, subject to the constraint imposed on the inputs.

Constraints
In theory it is possible to specify the constraint on the inputs and outputs of the control system. However, this algorithm only considers the input constraints. If the predictive control system is designed correctly, then the output constraints are not critical to the operation of the system, as the output should theoretically track the reference trajectory.

It is important to realise that the computational effort of finding the solution to the minimisation problem is directly proportional to the number of constraints imposed on the systems. Ideally all the predicted inputs should have constraints imposed on them, however this might be an over-design. It is therefore proposed that only the first r inputs have constraints.

Where:

$$r \leq HP \tag{2.35}$$

The cost function is a function of ΔU_{pred}, which are the predicted incremental inputs to the process. Therefore, if constraints are to be incorporated into the minimisation of the cost function, then it is necessary to specify the constraints as a function of the incremental inputs. This is done as follows:

Constraint on input u(t+x)
Consider the sum of the incremental inputs, which in general satisfies the equation:

$$\sum_{j=1}^{x} \Delta u(t+j) = u(t+x) - u(t-1) \tag{2.36}$$

The constraint can be specified as:

$$U_{min} - u(t-1) \leq \sum_{j=1}^{x} \Delta u(t+j) \leq U_{max} - u(t-1) \tag{2.37}$$

Where: U_{min} = Minimum input constraint.
U_{max} = Maximum input constraint.
$u(t-1)$ = Present input.
x = Variable indicating which input is being considered.

Constraints expressed as a matrix
The equation 2.37 is separated into the minimum constraint, and the maximum

constraint, as follows:

$$\sum_{j=1}^{x} \Delta u(t+j) \leq U_{max} - u(t-1) \tag{2.38}$$

$$-\sum_{j=1}^{x} \Delta u(t+j) \leq -(U_{min} - u(t-1)) \tag{2.39}$$

The equations 2.38 and 2.39 represent the constraints imposed on the input u(t+x). This summation can be expressed as a vector multiplication.

$$\sum_{j=1}^{x} \Delta u(t+j) = \mathbf{a}_x(\Delta \mathbf{U}_{\text{pred}}) \tag{2.40}$$

The variable $\mathbf{a}_x$ is a row vector, consisting of ones and zeros, which describes which $\Delta u(t+j)$ must be added.

Each input is considered in turn. A matrix is set-up, which describes all the constraints on the inputs.

$$\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_r \\ -\mathbf{a}_0 \\ -\mathbf{a}_1 \\ \vdots \\ -\mathbf{a}_r \end{bmatrix} \Delta \mathbf{U}_{\text{pred}} \leq \begin{bmatrix} U_{\max} - u(t-1) \\ U_{\max} - u(t-1) \\ \vdots \\ U_{\max} - u(t-1) \\ -(U_{\min} - u(t-1)) \\ -(U_{\min} - u(t-1)) \\ \vdots \\ -(U_{\min} - u(t-1)) \end{bmatrix} \tag{2.41}$$

This matrix equation (2.41) is written as:

$$\mathbf{a}_{\text{con}} \Delta \mathbf{U}_{\text{pred}} \leq \mathbf{b}_{\text{con}} \tag{2.42}$$

Solution of minimisation problem

The cost function is a quadratic equation, which is a function of the set of incremental inputs ($\Delta \mathbf{U}_{\text{pred}}$):

$$\text{Cost} = (\Delta U_{\text{pred}})^T P(\Delta U_{\text{pred}}) + C(\Delta U_{\text{pred}}) \tag{2.43}$$

The solution of this quadratic problem is solved using a quadratic program. A 'black box' approach is used to solve this problem. MATLAB has a quadratic program solver qp.m, which is ideal for solving this problem. Refer to the MATLAB appendix for details of program.

This quadratic program solver is capable of solving a quadratic program subject to constraints. The solution of this problem, consists of solving the following quadratic problem:

$$\min\{ \ (\Delta U_{\text{pred}})^T P(\Delta U_{\text{pred}}) + C(\Delta U_{\text{pred}}) \ \mid \ \mathbf{a}_{\text{con}}(\Delta \mathbf{U}_{\text{pred}}) \leq \mathbf{b}_{\text{con}} \ \} \tag{2.44}$$

The quadratic programming routine which MATLAB offers has the feature of allowing the program to start at a predefined starting point. This feature is very useful in this particular application, as the starting point can be defined based on the previous steps set of predicted inputs. This decreases the computation time.

2.6 Summary

This section shows how the black box algorithms derived in the previous sections are integrated into the MSC algorithm.

2.6.1 MSC algorithm for SISO system

1. **Define parameters**

   Define: HP = Prediction Horizon
   τ = Rise Time
   Y_{sp} = Set point
   T = Sampling period

   $b = e^{-T/\tau}$

2. **Set-up Predictor Equation**

   (model, HP) $\longrightarrow$ `calc.m` $\longrightarrow$ Q,W,R matrices

   The model of the process is specified by the $B(q^{-1})$ and $A(q^{-1})$ vectors.

3. **Define weighting variables**
   $\gamma = \text{diag}[\lambda(1), \lambda(2), ..., \lambda(\text{HP})]$
   $\beta = \text{diag}[\beta(1), \beta(2), ..., \beta(\text{HP})]$

   $\boldsymbol{P} = (\gamma W)^T(\gamma W) + \beta^T\beta$

4. **Measure output**
   The output of the process is measured and the history vectors are updated.
   Present output = y(t) $\Rightarrow \Delta y(t) = (1 - q^{-1})y(t) = y(t) - y(t-1)$

5. **Set-up reference trajectory**

   (Y_{sp},b,y(t),HP) $\longrightarrow$ `tref.m` $\longrightarrow$ $\mathbf{Y^*}$ $\longrightarrow$ operate on $\mathbf{Y^*}$ $\longrightarrow$ $(\mathbf{\Delta Y^*})$

6. **Predict set of optimum inputs**
   $\mathbf{\Delta \hat{Y}_1 = Q\Delta Y_{past} + R\Delta U_{past}}$
   $\boldsymbol{C} = -2(\gamma W)^T\gamma(\mathbf{\Delta Y^* - \Delta \hat{Y}_1})$
   The constraints on the input are defined by: $\mathbf{a_{con}\,\Delta U_{pred} \leq b_{con}}$

   S = Starting point of quadratic program which is based on previous predicted set of inputs.

   Predict the set of optimum inputs, using a quadratic program:

   (P, C, $\mathbf{a_{con}}$, $\mathbf{b_{con}}$, S) $\longrightarrow$ Quadratic program $\longrightarrow$ $\mathbf{\Delta U_o}$

7. **Execute optimum input**
   optimum input = $\mathbf{\Delta U_o}(\text{HP}) + u(t-1)$

8. **Wait for end of cycle**
   Hold input constant, while waiting for cycle of end.

9. Goto beginning of cycle
Goto number 4.

Chapter 3

Extension to MIMO systems

Chapter 2 introduced the MSC algorithm for the SISO case. This chapter extends the analysis to handle MIMO systems.

3.1 Model Representation

The transfer function (s domain) of a MIMO system is usually written:

$$P(s) = \frac{\begin{bmatrix} num_{11}(s) & num_{12}(s) & \dots & num_{1l}(s) \\ num_{21}(s) & num_{22}(s) & \dots & num_{2l}(s) \\ \vdots & \vdots & & \vdots \\ num_{m1}(s) & num_{m2}(s) & \dots & num_{ml}(s) \end{bmatrix}}{den(s)} \tag{3.1}$$

Where: m = Number of plant outputs.
l = Number of plant inputs.

It is possible to transform each of the sub-transfer functions into its discrete transfer function form. Example:

$$\frac{num11(s)}{den(s)} \Longrightarrow \frac{B_{11}(q^{-1})}{A(q^{-1})} \tag{3.2}$$

If all the sub-transfer functions share the same common denominator, then the sub-transfer functions in the discrete time domain will all have the same $A(q^{-1})$.
The transfer function (in discrete time) of a system is written:

$$\mathbf{y(t)} = \frac{\mathbf{B}(q^{-1})}{\mathbf{A}(q^{-1})}\mathbf{u(t)} \tag{3.3}$$

Where:

$$\mathbf{A}(q^{-1}) = \mathbf{A}_1 + \mathbf{A}_2 q^{-1} + \mathbf{A}_3 q^{-2} + \dots + \mathbf{A}_{n+1} q^{-n} \tag{3.4}$$

$$\mathbf{B}(q^{-1}) = \mathbf{B}_1 + \mathbf{B}_2 q^{-1} + \mathbf{B}_3 q^{-2} + \dots + \mathbf{B}_{n+1} q^{-n} \tag{3.5}$$

When considering MIMO systems, the same equations is used.

Rules for extension from SISO to MIMO

The following extensions to Equations 3.3,3.4 and 3.5 are made when MIMO systems are considered:

1. All elements in matrices are now treated as matrix elements.
   Example: Each element in the $\mathbf{A}_1(q^{-1})$ and $\mathbf{B}_1(q^{-1})$ vectors are treated as matrices.

2. Each element of B is defined:

$$\mathbf{B}_x = \begin{bmatrix} B_{11_x} & B_{12_x} & \cdots & B_{1l_x} \\ B_{21_x} & B_{22_x} & \cdots & B_{2l_x} \\ \vdots & \vdots & & \vdots \\ B_{m1_x} & B_{m2_x} & \cdots & B_{ml_x} \end{bmatrix}, \qquad x = 1...(n+1) \qquad (3.6)$$

3. Each element of A is defined:

$$\mathbf{A}_x = A_x \mathrm{I}(m), \qquad x = 1...(n+1) \qquad (3.7)$$

   I(m) = identity matrix of size m.

4. The variable y(t+k) is defined (k is a dummy variable):

$$\mathbf{y}(t+k) = \begin{bmatrix} y_1(t+k) \\ y_2(t+k) \\ \vdots \\ y_m(t+k) \end{bmatrix} \qquad (3.8)$$

5. The variable u(t+k) is defined (k is a dummy variable):

$$\mathbf{u}(t+k) = \begin{bmatrix} u_1(t+k) \\ u_2(t+k) \\ \vdots \\ u_l(t+k) \end{bmatrix} \qquad (3.9)$$

The following example explains the notation:

Example:
Consider the following MIMO system.

$$P(s) = \begin{bmatrix} \frac{-2.1}{16.7s+1} & \frac{3.1}{21s+1} \\ \frac{-0.52}{10.9s+1} & \frac{1.52}{14.4s+1} \end{bmatrix}$$

$$= \frac{\begin{bmatrix} -6921.24s^3 - 1445.35s^2 - 97.23s - 2.1 & 8125.82s^3 + 1796.36s^2 + 130.20s + 3.1 \\ -2626.04s^3 - 464.66s^2 - 27.09s - 0.52 & 5810.40s^3 + 1157.68s^2 + 73.87s + 1.52 \end{bmatrix}}{55045.87s^4 + 14790.10s^3 + 1461.47s^2 + 83.00s + 1}$$

$$= \frac{\begin{bmatrix} num_{11}(s) & num_{12}(s) \\ num_{21}(s) & num_{21}(s) \end{bmatrix}}{den(s)}$$

Conversion to discrete time
Sampling period: T = 5 seconds

$$\frac{num_{11}(s)}{den(s)} \rightarrow \frac{B_{11_1} + B_{11_2}q^{-1} + B_{11_3}q^{-2} + B_{11_4}q^{-3} + B_{11_5}q^{-4}}{A_1 + A_2q^{-1} + A_3q^{-2} + A_4q^{-3} + A_5q^{-4}}$$

$$\rightarrow \frac{0 - 0.5433q^{-1} + 1.556q^{-2} - 0.8160q^{-3} + 0.1913q^{-4}}{1 - 2.8681q^{-1} + 3.0783q^{-2} - 1.4652q^{-3} + 0.2609q^{-4}}$$

$$\frac{num_{12}(s)}{den(s)} \rightarrow \frac{B_{12_1} + B_{12_2}q^{-1} + B_{12_3}q^{-2} + B_{12_4}q^{-3} + B_{12_5}q^{-4}}{A_1 + A_2q^{-1} + A_3q^{-2} + A_4q^{-3} + A_5q^{-4}}$$

$$\rightarrow \frac{0 + 0.6568q^{-1} - 1.3662q^{-2} + 0.9452q^{-3} - 0.2175q^{-4}}{1 - 2.8681q^{-1} + 3.0783q^{-2} - 1.4652q^{-3} + 0.2609q^{-4}}$$

$$\frac{num_{21}(s)}{den(s)} \rightarrow \frac{B_{21_1} + B_{21_2}q^{-1} + B_{21_3}q^{-2} + B_{21_4}q^{-3} + B_{21_5}q^{-4}}{A_1 + A_2q^{-1} + A_3q^{-2} + A_4q^{-3} + A_5q^{-4}}$$

$$\rightarrow \frac{0 - 0.1913q^{-1} + 0.4278q^{-2} - 0.3185q^{-3} + 0.0790q^{-4}}{1 - 2.8681q^{-1} + 3.0783q^{-2} - 1.4652q^{-3} + 0.2609q^{-4}}$$

$$\frac{num_{22}(s)}{den(s)} \rightarrow \frac{B_{22_1} + B_{22_2}q^{-1} + B_{22_3}q^{-2} + B_{22_4}q^{-3} + B_{22_5}q^{-4}}{A_1 + A_2q^{-1} + A_3q^{-2} + A_4q^{-3} + A_5q^{-4}}$$

$$\rightarrow \frac{0 + 0.4459q^{-1} - 0.9638q^{-2} + 0.6916q^{-3} - 0.1647q^{-4}}{1 - 2.8681q^{-1} + 3.0783q^{-2} - 1.4652q^{-3} + 0.2609q^{-4}}$$

Matrix variables:
According to the rules, the variables in the MIMO transfer function are defined as follows:

$$\mathbf{B}_1 = \begin{bmatrix} B_{11_1} & B_{12_1} \\ B_{21_1} & B_{22_1} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad \mathbf{A}_1 = \begin{bmatrix} A_1 & 0 \\ 0 & A_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{B}_2 = \begin{bmatrix} B_{11_2} & B_{12_2} \\ B_{21_2} & B_{22_2} \end{bmatrix} = \begin{bmatrix} -0.5422 & 0.6568 \\ -0.1913 & 0.4459 \end{bmatrix} \quad \mathbf{A}_2 = \begin{bmatrix} A_2 & 0 \\ 0 & A_2 \end{bmatrix} = \begin{bmatrix} -2.8681 & 0 \\ 0 & -2.8681 \end{bmatrix}$$

$$\mathbf{B}_3 = \begin{bmatrix} B_{11_3} & B_{12_3} \\ B_{21_3} & B_{22_3} \end{bmatrix} = \begin{bmatrix} 1.556 & -1.3662 \\ 0.4278 & -0.9638 \end{bmatrix} \quad \mathbf{A}_3 = \begin{bmatrix} A_3 & 0 \\ 0 & A_3 \end{bmatrix} = \begin{bmatrix} 3.0783 & 0 \\ 0 & 3.0783 \end{bmatrix}$$

$$\mathbf{B}_4 = \begin{bmatrix} B_{11_4} & B_{12_4} \\ B_{21_4} & B_{22_4} \end{bmatrix} = \begin{bmatrix} -0.8160 & 0.9452 \\ -0.3185 & 0.6916 \end{bmatrix} \quad \mathbf{A}_4 = \begin{bmatrix} A_4 & 0 \\ 0 & A_4 \end{bmatrix} = \begin{bmatrix} -1.4652 & 0 \\ 0 & -1.4652 \end{bmatrix}$$

$$\mathbf{B}_5 = \begin{bmatrix} B_{11_5} & B_{12_5} \\ B_{21_5} & B_{22_5} \end{bmatrix} = \begin{bmatrix} 0.1913 & -0.2175 \\ 0.0790 & -0.1647 \end{bmatrix} \quad \mathbf{A}_5 = \begin{bmatrix} A_5 & 0 \\ 0 & A_5 \end{bmatrix} = \begin{bmatrix} 0.2609 & 0 \\ 0 & 0.2609 \end{bmatrix}$$

The input and output vectors are defined as follows:

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} \qquad \mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

$$\mathbf{y}(t-1) = \begin{bmatrix} y_1(t-1) \\ y_2(t-1) \end{bmatrix} \qquad \mathbf{u}(t-1) = \begin{bmatrix} u_1(t-1) \\ u_2(t-1) \end{bmatrix}$$

$$\mathbf{y}(t-2) = \begin{bmatrix} y_1(t-2) \\ y_2(t-2) \end{bmatrix} \qquad \mathbf{u}(t-2) = \begin{bmatrix} u_1(t-2) \\ u_2(t-2) \end{bmatrix}$$

$$\mathbf{y}(t-3) = \begin{bmatrix} y_1(t-3) \\ y_2(t-3) \end{bmatrix} \qquad \mathbf{u}(t-3) = \begin{bmatrix} u_1(t-3) \\ u_2(t-3) \end{bmatrix}$$

$$\mathbf{y}(t-4) = \begin{bmatrix} y_1(t-4) \\ y_2(t-4) \end{bmatrix} \qquad \mathbf{u}(t-4) = \begin{bmatrix} u_1(t-4) \\ u_2(t-4) \end{bmatrix}$$

3.2 MSC algorithm for MIMO systems

1. Define parameters

Define: HP = Prediction Horizon

τ = Rise Time
$$= \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_m \end{bmatrix}$$

$\mathbf{Y}_{sp}$ = Set point
$$= \begin{bmatrix} Y_{sp_1} \\ Y_{sp_2} \\ \vdots \\ Y_{sp_m} \end{bmatrix}$$

T = Sampling period

$\mathbf{b(x)} = e^{-\mathbf{T}/\tau(x)}, \qquad \mathbf{x} = 1..,\mathbf{m}$

2. Set-up Predictor Equation

(model, HP) $\longrightarrow$ `calcm.m` $\longrightarrow$ Q,W,R matrices

The model of the process is specified by the $\mathbf{B}(q^{-1})$ and $\mathbf{A}(q^{-1})$ vectors.

3. Define weighting variables

$\lambda(x) = diag[\lambda_1(x), \lambda_2(x), ..., \lambda_m(x)] \qquad x = 1...HP$
$\gamma = \text{diag}[\lambda(1), \lambda(2), ..., \lambda(\text{HP})]$

$\beta(x) = diag[\beta_1(x), \beta_2(x), ..., \beta_l(x)] \qquad x = 1...HP$
$\beta = \text{diag}[\beta(1), \beta(2), ..., \beta(\text{HP})]$

$P = (\gamma W)^T(\gamma W) + \beta^T\beta$

4. Measure output

The output of the process is measured and the history vector updated.
Present output = $\mathbf{y(t)} \Rightarrow \Delta\mathbf{y}(t) = \mathbf{y}(t) - \mathbf{y}(t-1)$

5. Set-up reference trajectory

($\mathbf{Y}_{sp}$,b,y(t),HP) $\longrightarrow$ `tref.m` $\longrightarrow \mathbf{Y}^* \longrightarrow$ operate on $\mathbf{Y}^*$ $\longrightarrow (\Delta\mathbf{Y}^*)$

6. Predict set of optimum inputs

$\Delta\hat{\mathbf{Y}}_1 = \mathbf{Q}\Delta\mathbf{Y}_{past} + \mathbf{R}\Delta\mathbf{U}_{past}$
$C = -2(\gamma W)^T\gamma(\Delta\mathbf{Y}^* - \Delta\hat{\mathbf{Y}}_1)$
The constraints on the input are defined by: $\mathbf{a}_{con}\,\Delta\mathbf{U}_{pred} \leq \mathbf{b}_{con}$

S = Starting point of quadratic program which is based on previous predicted set of inputs.

Predict the set of optimum inputs, using a quadratic program:

$$(P, C, a_{con}, b_{con}, S) \longrightarrow \boxed{\text{Quadratic program}} \longrightarrow \Delta U_o$$

7. **Execute optimum inputs**
   optimum inputs = $\Delta U_o(HP) + u(t-1)$

8. **Wait for end of cycle**
   Hold inputs constant, while waiting for cycle of end.

9. **Goto beginning of cycle**
   Goto number 4

Chapter 4

Evaluation of Multi-Step Predictive Control Algorithm

This chapter is intended to provide evidence of the functionality of the predictive control algorithm. The efficiency of the algorithm is also investigated.

4.1 Functionality

The functionality of the algorithm is demonstrated by way of simulation. This simulation is merely intended to demonstrate that the algorithm which is derived in previous chapters does perform as intended. A simulation m-file is written (`simulate.m`), which synthesises the control of a system using the MSC algorithm. This simulation attempts to provide a realistic evaluation of the control technique. Realistic considerations are:

- The plant model and predictor model differ.
- Disturbances effect the performances of the control system.

For these reasons, separate plant and predictor models are used in the simulation. The plant model used has an uncontrollable input, which describes the disturbances on the system. The control system is expected to 'null-out' the effect of these disturbances.

4.1.1 Simulation m-file

The pre-amble of the simulation allows the following parameters to be set :

- HP: Horizon Prediction.
- T: Sampling period.
- Ysp: Column vector, describing the intended set-point of each output.
- tau: Column vector, describing the rise time of each trajectory.
- Umax: Column vector describing the maximum boundary of the constraint on the inputs.
- Umin: Column vector describing the minimum boundary of the constraint on the inputs.

- no_const: Number of future inputs constrained (r)
- d: Specification of disturbance input for each simulation step.
- The beta and alpha weighting variables can also be set.

The simulation m-file provides the user with an account of the systems performance at each step. The reference trajectory (ΔY^*), predicted output trajectory ($\Delta \hat{Y}$) for each output, as well as the predicted inputs for each input channel are displayed after each simulated step.

4.1.2 Simulation

The model of the plant selected is typical of a process control system. It has transport delays, significantly slow dynamics, disturbances and parameter variations. Different plant and predictor models are used in order to create an environment that mimics a system that has parameter variations.

Equation 4.1 shows the MIMO system which is controlled using the MSC algorithm.

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-8Ts}}{(14.9s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-Ts}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad (4.1)$$

Where: $y_1(t)$ = Output 1 of system.
$y_2(t)$ = Output 2 of system.
$u_1(t)$ = Input 1 of system (controllable).
$u_2(t)$ = Input 2 of system (controllable).
$d(t)$ = Disturance input (uncontrollable).
T = Sampling period (5 seconds).

Equation 4.2 shows the model that is used to predict the behaviour of the process. Note: The plant and predictor models are different.

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(17s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad (4.2)$$

Pre-amble settings

HP	$= 15$	T	$= 5$ seconds
Ysp	$= \begin{bmatrix} 0 \\ 1 \end{bmatrix}$	tau	$= \begin{bmatrix} 50 \\ 50 \end{bmatrix}$ seconds
Umax	$= \begin{bmatrix} 5 \\ 5 \end{bmatrix}$	Umin	$= \begin{bmatrix} -1 \\ -1 \end{bmatrix}$
no_const	$= 5$	no_steps	$= 200$

d = zero from 1...20, one from 20...100, zero from 100...200

β $\quad \beta_1(x) = 0.1 + 0.01x$
$\quad \beta_2(x) = 0.1 + 0.01x \qquad x = 1..HP$

λ $\quad \lambda_1(x) = 1$
$\quad \lambda_2(x) = 1 \qquad x = 1..HP$

Simulation results
These results confirm the functionality of the algorithm.

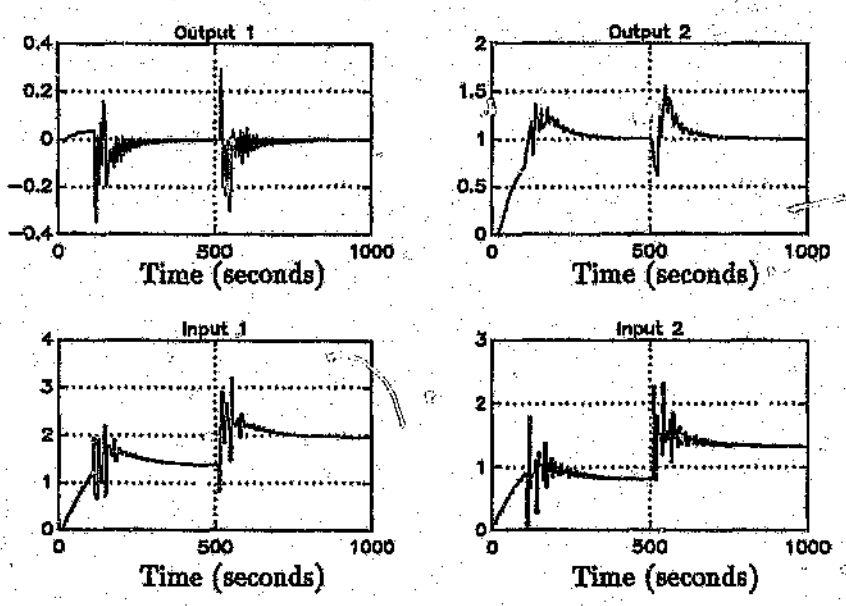


Figure 4.1: Test for functionality

Observations
The following observations are made from the results, shown in Figure 4.1.

1. The disturbance rejection properties of the system are quite clear from the plot of the outputs.
2. The disturbances to the system cause offset errors, which are cancelled out by the integral action of the MSC algorithm.
3. The output transients appear to track the reference trajectory. The rise-time was set to 50 seconds, which correspond with the simulation results.
4. The inputs to the system are both within the constraints defined. The ability for the system to limit the inputs would be made clearer if the β variables were reduces. This would increase the closed loop bandwidth, resulting in a more aggressive control action.
5. The set point of the first input is intentionally set to zero. In this way the dynamic coupling between the outputs are observed. One of the functions of a MIMO control system is to minimise the dynamic coupling between the outputs, so that the outputs of the system can be controlled independently. The dynamic coupling of the MSC system appears to be satisfactory, although this could be improved by improving the predictor model and specifying a larger closer loop bandwidth.

4.2 Efficiency

Evidence of the functionality of the algorithm is given in section 4.1. This section assumes that the algorithm is functional, and aims the investigation more on trying to find out how efficient the algorithm is.

Two factors are used to gauge the efficiency of the algorithm, namely the *computational effort* and the *memory* required when implementing the algorithm.
These two factors are related to the following parameters:

- r (Number of inputs constrained).
- HP (Prediction horizon).
- U_{max}, U_{min} (Constraints on inputs).

4.2.1 Efficiency as a function of r

The variable r describes the number of future predicted inputs that are constrained.

Example: If a prediction horizon of 10 steps are used (HP=10)
and only the first 5 steps are to have constraints, then
r = 5.

The computational effort is related to the variable r. The larger r, the more complex the optimisation problem is, which implies that more computations are required to find the optimum solution.

A series of simulations are performed to find the relationship between the computational effort and the variable r. The computational effort is measured as the number of flops [1] that are performed by the simulation.

Procedure: Simulations, similar to the functionality test are executed.
The variable r is changed after each simulation.
The total number of flops for each simulation is measured.
These results are then tabulated and analysed.

Refer to Appendix B for exact details of tests 2 to 10. Table 4.1 gives details of the simulation results.

Test#	r	Flops	Memory required	Relative to r=1 flops	memory
2	1	1171183	44648	1	1
3	2	1483813	45352	1.2935	1.0158
4	3	15313253	46056	1.3349	1.0315
5	4	18213188	46760	1.5877	1.0473
6	5	19945250	47464	1.7387	1.0631
7	6	21570767	48168	1.8804	1.0788
8	7	24527650	48872	2.1382	1.0946
9	8	27950286	49576	2.4366	1.1104
10	9	32077628	50280	2.7964	1.1261

Table 4.1: Results of simulations (r)

The relative computational effort gives a relative measure of the computational effort compared to the effort of the system with r=1. In other words these results give a qualitative analysis of the effort. For example, setting r to 8, gives a computational effort 2,1382 times more that for a system with r=1.

Figure 4.2 graph gives a graphical representation of the results.

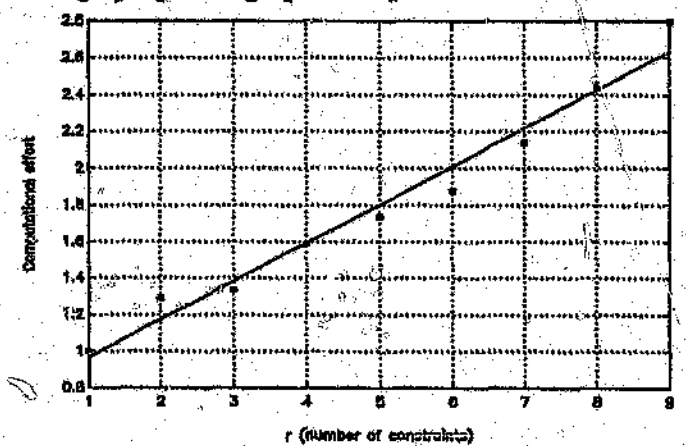


Figure 4.2: Plot of relative computational effort as a function of r

[1] Floating point operations

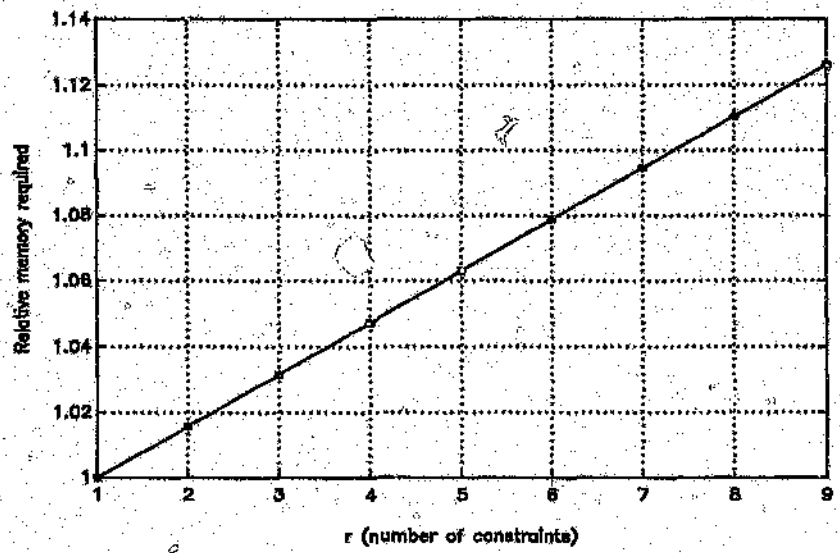


Figure 4.3: Plot of relative memory required as a function of r

Observation of results

- Refer to test8 and test10. The response of test8 is considered acceptable. Test9 and 10 obviously give better results, but at the expense of having a higher computational effort.
- It appears as though the computational effort and the memory required, is directory proportional to the r parameter.
- The memory required is not very dependent on the r variable, as can be seen from Figure 4.3.

4.2.2 Efficiency as a function of HP

The computational effort is related to HP, in that, the more inputs that are predicted, the more complex the optimisation problem becomes. A more complex optimisation problem requires more computations.

A series of simulations are performed to find the relationship between the computational effort and the variable HP. The computational effort is measured as the number of flops that are performed by the simulation.

Procedure: Simulations, similar to the functionality test are executed.
The variable HP is changed after each simulation.
The total number of flops for each simulation is measured.
These results are then tabulated and analysed.

Refer to Appendix B for exact details of tests 11 to 14. Table 4.2 gives details of the simulation results.

Test#	HP	Flops	Memory required	Relative to HP=10 flops	memory
11	10	19323703	47464	1	1
12	15	48570843	69144	2.5135	1.4568
13	20	96589305	97224	4.9985	2.0484
14	25	173195261	131704	8.9628	2.7748

Table 4.2: Results of simulations (HP)

The relative computational effort gives a relative measure of the computational effort compared to the effort of the system with HP=10.

Figure 4.4 gives a graphical representation of the results.

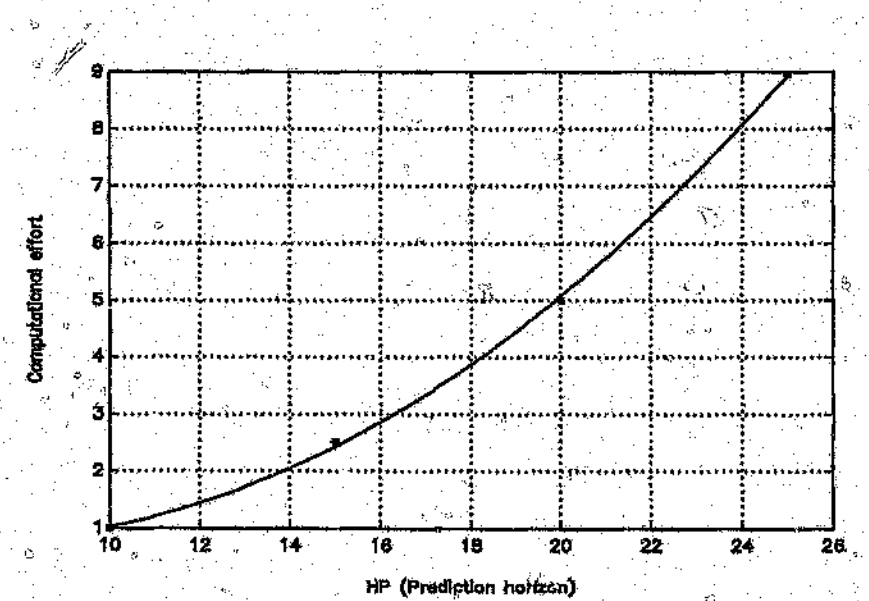


Figure 4.4: Plot of relative computational effort as a function of HP

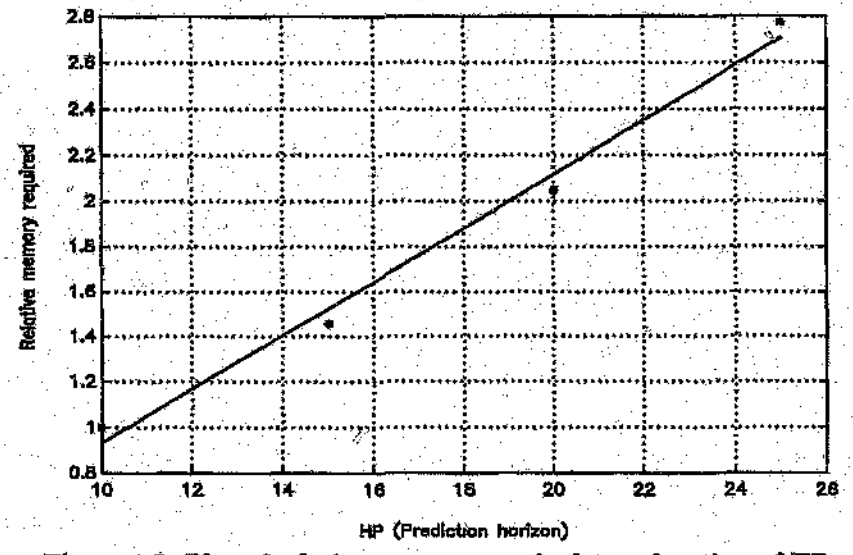


Figure 4.5: Plot of relative memory required as a function of HP

Observation of results

- Refer to test11 to test14. The response of test11 is considered acceptable. Test12,13 and 14 obviously give better results, but at the expense of having a higher computational effort.
- Refer to Figure 4.4. It appears as though the computational effort is directory proportional squared to the HP parameter.
- Refer to Figure 4.5. The memory required is proportional to the HP parameter.

4.2.3 Efficiency as a function of U_{min} and U_{max}

The computational effort is the only efficiency parameter that is dependent on the U_{min} and U_{max} settings.

If the U_{min} or U_{max} is set near the operating point of the process, then the computational effort required increases significantly. The optimisation routine attempts to find the set of optimum inputs which do not violate the input constraints. If the operating point is close to a constraint setting, then the solution of the optimisation problem is more complex. The solution to this problem therefore requires a greater computational effort.

To demonstrate this hypothesis, two simulations are performed. One simulation has the constraints far from the operating point, while the other has the constraint near the operating point. The computational effort of these two simulations are then compared.

Refer to Appendix B for exact details of these simulations.
Refer to test1 and test12.

Test#	U_{min}	U_{max}	Flops	Memory required
1	$[-1\ -1]^T$	$[5\ 5]^T$	43247677	69144
12	$[0\ 0]^T$	$[2.2\ 2]^T$	48570843	69144

Table 4.3: Results of simulations (constraints)

Table 4.3 shows the results of the simulations. The simulation with the constraint close to the operating point (test12), is computationally more demanding than the other simulation. This confirms the hypothesis.

Chapter 5

Stability and Robustness issues

An important design criteria of any process control system is that of stability and robustness.

The **stability** property of a control system is concerned with the stability of the closed loop response of the system. The **robustness** property of a control system is concerned with the ability of the system to function in the face of parameter variations of the plant model, as well as external disturbance imposed on the system.

The stability and robustness of predictive control techniques is extremely difficult to estimate due to the presence of the constraints, which cause the system to become non-linear. There are however many subtle ways to establish a stable and robust control system. This is the main theme of this chapter.

The investigation of these properties are supplemented with simulated examples which clarify the explanations.

5.1 Designing a robust and stable control system

The main building blocks for a control system based on classical control techniques is shown in Figure 5.1.

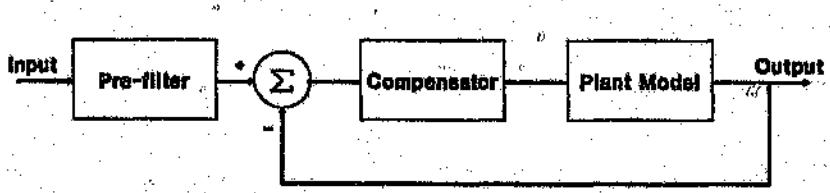


Figure 5.1: Block diagram of classical control system

The robustness and stability properties of the control system shown in Figure 5.1 are achieved by insuring:

1. The poles of the system are in the left hand plane.
2. The loop transmission is high over the bandwidth of the system.

The system shown in Figure 5.1 is a two degree of freedom system. The compensator is designed so that the system is stable and robust. The pre-filter is introduced, whose task is to *limit the bandwidth of the closed loop response* and also *specify the rise time* of the system.

5.2 The Reference Trajectory as a pre-filter

The rise time of the predictive control system is incorporated into the specification of the reference trajectory.

The output of the MSC system, hopefully, tracks the reference trajectory. If this is so, the system has the same rise time as the reference trajectory, in which case the closed loop bandwidth can be calculated from knowledge of the rise-time.

Note: There is a relationship between the rise-time and bandwidth of the system.

From the above discussion, it is clear that the reference trajectory and the pre-filter have much in common.

5.3 Limitations of Reference Model Representation

5.3.1 Frequency Band limitations

The model used to represent the dynamics of the system (predictor model) is merely an approximation of the true dynamic behaviour of the process. The reason for this approximation is that the true dynamic behaviour of physical system is very complex and changes with time. Fortunately, all dynamic systems are inherently low pass and have a finite bandwidth. This characteristic of physical systems means that the low frequency dynamics of the system dominate the behaviour of the system.

Using these principles a simplified dynamic model of the process is estimated by considering only the low frequency dynamic behaviour of the system. This simplified dynamic model of the system is used to make predictions on the behaviour of the system. Obviously the predictions are only valid if the predicted dynamics lie within the bandwidth of the predictor model. In more simpler terms, the closed loop bandwidth must be less than the bandwidth of the predictor model. If this is not the case, then the predictions made using the model are worthless and can lead to the instability of the MSC system.

5.3.2 Issues of selecting T, HP λ and β variables

The selection of the sampling period is dependent on the dynamics of the process and on the Nyquist Sampling Criteria.

The T and HP variables combined, describes the 'predictor window' (T*HP) of each control step. This predictor window should always be greater than the settling time of the process. This insures that the reference trajectory describes an accurate trajectory of the intended system response.

The λ variables describe the weightings on the respective portions of the predictor window. These weighting variables must be selected carefully so that the predicted response tracks the reference trajectory as best as possible. Although these variables add extra dimensions to the design capabilities of the system, they are not critical to the operation of the system. [1]

The β variables are very important when considering the robustness of the control system. Selecting 'high' β variables implies that the input signals to the system must not change quickly. This is a subtle way of controlling the bandwidth of the system.

An important constraint on the predicted change in the input ΔU_{pred} is that they must tend to zero, indicating that the predicted outputs have reached a steady state. One way of insuring this, is to select the β variables in such a way that they gradually increase.
For example: $\beta(x) = 0.1 + 0.01x$ Where x = 1...HP

By selecting the reference trajectory, λ and β variables carefully it is possible to control the robustness and stability of the system.

5.3.3 Changing β and its effects on the bandwidth

The principles for obtaining a stable and robust predictive control system are presented in section 5.3.2. This section gives evidence of these principles.
Refer to Appendix B, Tests15,16 and 17.

Test#	HP	$\beta(x)$ x=1...HP	Result
15	15	0.001+0.001x	Input signals to process oscillate. Poor steady state tracking.
16	15	0.01+0.001x	Input signals to process oscillate. Poor steady state tracking.
17	15	0.1+0.01x	Good robustness and tracking.

Table 5.1: Results of β tests

Table 5.1 shows the results of simulations which have different β values. The simulations which use 'small' β values, give poor steady state tracking and there is significant ringing in both the input and output variables. This result is to be expected because the bandwidth of the system is too large, which causes erroneous predictions (plant and predictor models are different).

The last test use 'higher' β values, and the results improve significantly. The β values are selected so that they increase as x increase. This improves the robustness of the system. The results (refer to Test17) shows that the system has good disturbance rejection, steady state tracking and robustness properties.

The following simulation illustrates the robustness of the MSC control technique. This simulation uses a predictor model which has parameter variations in all the model parameters. These parameter variations cause erroneous predictions, but due

[1] All simulations have λ=1

to the robustness property, the system is still operates.

Refer to Appendix B, Test18.

The simulation results show that the system remained stable and that there was no steady state error. This confirms the robustness of the MSC algorithm, which can be attributed to the careful selection of the β variables.

Chapter 6

Conclusion

6.1 General

The aim of this project was to establish a predictive control algorithm for controlling multivariable systems. The algorithm was first developed for the SISO case in chapter 2, and was then extended to handle MIMO systems in chapter 3. The reason for this approach is that the SISO algorithm created a useful framework, which can be easily extended to MIMO systems. An important feature of this algorithm is that it has inherent integral action. This feature is essential for the 'nulling' out of constant external disturbances imposed on the system. However, the most important feature of this control algorithm is that the designer can specify constraints on the input variables to the process.

In chapter 4, simulations where carried out with the intention of assessing the functionality and efficiency of the algorithm. The functionality investigation verified that the algorithm is functioning correctly and that the overall approach to solving the problem was correct. The efficiency investigation led to a qualitative analysis of the memory and computational effort requirements of the algorithm.

Chapter 5 gave a qualitative approach for designing a stable and robust predictive control system. The need for careful attention to selecting algorithm parameters was emphasised, and simulations where given to illustrate the concepts. The robustness properties of the predictive control system where shown with the aid of a simulated example. This example used a predictor model which would have given serious steady state errors, but good results were still achieved under MSC feedback control.

Used in the correct manner, Multi-Step Control can yield a very powerful and effective multivariable control system.

6.2 Assessment

Classical control techniques are useful when designing control systems that have linear models. These control concepts experience difficulties when non-linear systems are being considered. All systems have non-linear properties. The approach to designing control systems based on classical control concepts therefore forces the designer to select the operating points that are in the most linear region.

If the operating points of a control system are placed closer to the saturation limits and a stable, robust control technique is used, then the yield of the process can often be increased. This can be accomplished using predictive control techniques. Therefore, there is a an economic incentive for using predictive control techniques.

The work done in this project has shown that it is possible to design stable, robust multivariable control system, that allow for constraints on the process input variables.

This project has not investigated the performance of MSC control in a non-linear environment. A typical investigation would, for instance, require a simulation of the MSC system using a non-linear plant model.

6.3 Suggestions for future work

Some principles for achieving a stable and robust control system have been presented. However, only a qualitative method exists for establishing stability and robustness. There is a need for a more quantitative method that would take into consideration issues such as plant model mismatch.

A mathematical function, similar to the loop transmission (classical control), is required for predictive control systems. This function would enable the designer to accurately determine the stability and robustness properties of the system.

The predictive control algorithm developed in this project relies on a predictor model of the process. At present this predictor model is based on a linear system model representation. Future work might investigate converting the predictor model from a linear to a non-linear model.

Appendix A

Predictor Equation

This appendix gives a detailed analysis of how the predictor equation is derived. The predictor equation is derived assuming a known plant model is available.

Initially, only SISO systems are considered, however, this analysis is easily extended to cater for MIMO systems.
The analysis is as follows:

- **Section A.1:** SISO systems.
- **Section A.2:** MIMO systems.

The predictor equation:

This equation provides a mechanism for predicting the plant output
for $t = t_0 + 1 \ldots t_0 + \mathrm{HP}$

The implementation of this equation relies on the following information:

- Past input to plant.
- Past output of plant.
- Estimated model of the plant.
- Estimated future plant inputs.

A.1 Predictor Equation: SISO system

The transfer function (in discrete time) of a SISO system is written:

$$y(t) = \frac{B(q^{-1})}{A(q^{-1})}u(t) \tag{A.1}$$

Where:

$$A(q^{-1}) = A_1 + A_2q^{-1} + A_3q^{-2} + ... + A_{n+1}q^{-n} \tag{A.2}$$

$$B(q^{-1}) = B_1 + B_2q^{-1} + B_3q^{-2} + ... + B_{n+1}q^{-n} \tag{A.3}$$

n = order of transfer function.

NB: $A_1 = 1$ and $B_1 = 0$, for all systems.

In order to demonstrate the mechanism of deriving the predictor equation, an arbitrary system will be used. After this demonstration, the predictor equation will be derived for the general case.

Example: The predictor equation is derived for a SISO system of order 4 and prediction horizon of 2 steps.
i.e. n = 4 and HP = 2

The transfer function (A.1) can be expanded as follows:

$$[1 + A_2q^{-1} + A_3q^{-2} + A_4q^{-3} + A_5q^{-4}]y(t) = [B_2q^{-1} + B_3q^{-2} + B_4q^{-3} + B_5q^{-4}]u(t) \tag{A.4}$$

This may be re-arranged into the following equation:

$$y(t) = [B_2q^{-1} + B_3q^{-2} + B_4q^{-3} + B_5q^{-4}]u(t) - [A_2q^{-1} + A_3q^{-2} + A_4q^{-3} + A_5q^{-4}]y(t) \tag{A.5}$$

The shift operator may be illuminated from Equation A.5 by writing the equation in terms of the past inputs and outputs:

$$y(t) = [B_2\ B_3\ B_4\ B_5]\begin{bmatrix} u(t-1) \\ u(t-2) \\ u(t-3) \\ u(t-4) \end{bmatrix} + [-A_2\ -A_3\ -A_4\ -A_5]\begin{bmatrix} y(t-1) \\ y(t-2) \\ y(t-3) \\ y(t-4) \end{bmatrix} \tag{A.6}$$

Using the shift operator, the next predicted output may be written as:

$$\hat{y}(t+1) = qy(t) \tag{A.7}$$

Substituting Equation A.6 into Equation A.7 and illuminating the shift operator, the equation for $\hat{y}(t+1)$ may be written:

$$\hat{y}(t+1) = [B_2\ B_3\ B_4\ B_5]\begin{bmatrix} u(t) \\ u(t-1) \\ u(t-2) \\ u(t-3) \end{bmatrix} + [-A_2\ -A_3\ -A_4\ -A_5]\begin{bmatrix} y(t) \\ y(t-1) \\ y(t-2) \\ y(t-3) \end{bmatrix} \tag{A.8}$$

Using the shift operator, the next predicted output is:

$$\hat{y}(t+2) = q\hat{y}(t+1) \tag{A.9}$$

After substituting the equation for $\hat{y}(t+1)$ (Equation A.8), and again illuminating the shift operator, the equation for $\hat{y}(t+2)$ is:

$$\hat{y}(t+2) = [B_2\ B_3\ B_4 B_5]\begin{bmatrix} u(t+1) \\ u(t) \\ u(t-1) \\ u(t-2) \end{bmatrix} + [-A_2\ -A_3\ -A_4 - A_5]\begin{bmatrix} \hat{y}(t+1) \\ y(t) \\ y(t-1) \\ y(t-2) \end{bmatrix} \tag{A.10}$$

Now, the above equation $\hat{y}(t+2)$ (Equation A.10) is a function of $\hat{y}(t+1)$. In order to illuminate this, $\hat{y}(t+1)$ (Equation A.8) must be substituted into this equation.

$$\begin{aligned}
\hat{y}(t+2) &= [B_2\ B_3\ B_4\ B_5]\begin{bmatrix} u(t+1) \\ u(t) \\ u(t-1) \\ u(t-2) \end{bmatrix} + (-A_2)\hat{y}(t+1) + [-A_3\ -A_4\ -A_5]\begin{bmatrix} y(t) \\ y(t-1) \\ y(t-2) \end{bmatrix} \\
&= [B_2\ B_3\ B_4\ B_5]\begin{bmatrix} u(t+1) \\ u(t) \\ u(t-1) \\ u(t-2)) \end{bmatrix} + (-A_2)[B_2\ B_3\ B_4\ B_5]\begin{bmatrix} u(t) \\ u(t-1) \\ u(t-2) \\ u(t-3) \end{bmatrix} \\
&\quad + (-A_2)[-A_2\ -A_3\ -A_4\ -A_5]\begin{bmatrix} y(t) \\ y(t-1) \\ y(t-2) \\ y(t-3) \end{bmatrix} + [-A_3\ -A_4\ -A_5]\begin{bmatrix} y(t) \\ y(t-1) \\ y(t-2) \end{bmatrix} \\
&= [(B_2)\ (B_3 - A_2B_2)\ (B_4 - A_2B_3)\ (B_5 - A_2B_4)\ (-A_2B_5)]\begin{bmatrix} u(t+1) \\ u(t) \\ u(t-1) \\ u(t-2) \\ u(t-3) \end{bmatrix} \\
&\quad + [(-A_3 + A_2^2)\ (-A_4 + A_2A_3)\ (-A_5 + A_2A_4)\ (A_2A_5)]\begin{bmatrix} y(t) \\ y(t-1) \\ y(t-2) \\ y(t-3) \end{bmatrix}
\end{aligned} \tag{A.11}$$

The resulting equations for $\hat{y}(t+1)$ and $\hat{y}(t+2)$ are used to set up the predictor equation. The predictor equation is simply a matrix equation, which describes the equations $\hat{y}(t+1)$ ... $\hat{y}(t+\mathrm{HP})$.

The following notation will be used to describe the predictor equation:

$$\hat{Y} = QY_{\text{past}} + WU_{\text{pred}} + RU_{\text{past}} \tag{A.12}$$

Using the notation defined in Equation A.12, the equations $\hat{y}(t+1)$ and $\hat{y}(t+2)$ can be used to find the matrices:

$$Q = \begin{bmatrix} -A_2 & -A_3 & -A_4 & -A_5 \\ (-A_3 + A_2^2) & (-A_4 + A_2A_3) & (-A_5 + A_2A_4) & (A_2A_5) \end{bmatrix} \tag{A.13}$$

$$W = \begin{bmatrix} 0 & B_2 \\ B_2 & (B_3 - A_2B_2) \end{bmatrix} \tag{A.14}$$

$$R = \begin{bmatrix} B_3 & B_4 & B_5 \\ (B_4 - A_2B_3) & (B_5 - A_2B_4) & (-A_2B_5) \end{bmatrix} \tag{A.15}$$

And the matrices for Y_{past}, U_{pred} and U_{past} are defined as follows:

$$Y_{\text{past}} = \begin{bmatrix} y(t) \\ y(t-1) \\ y(t-2) \\ y(t-3) \end{bmatrix}, U_{\text{pred}} = \begin{bmatrix} u(t+1) \\ u(t) \end{bmatrix}, U_{\text{past}} = \begin{bmatrix} u(t-1) \\ u(t-2) \\ u(t-3) \end{bmatrix} \tag{A.16}$$

The previous analysis merely illustrates how the predictor equation is formulated.

The predictor equation is now derived for the general case:

For any system of order n, it can be shown that the equation for y(t) has the following structure:

$$y(t) = [B_2\ B_3\ \dots\ B_{n+1}]\begin{bmatrix} u(t-1) \\ u(t-2) \\ \vdots \\ u(t-n) \end{bmatrix} + [-A_2\ -A_3\ \dots\ -A_{n+1}]\begin{bmatrix} y(t-1) \\ y(t-2) \\ \vdots \\ y(t-n) \end{bmatrix} \tag{A.17}$$

Using Equation A.17, $\hat{y}(t+1)$ is written:

$$\hat{y}(t+1) = [B_2\ B_3\ \dots\ B_{n+1}]\begin{bmatrix} u(t) \\ u(t-1) \\ \vdots \\ u(t-n+1)) \end{bmatrix} + [-A_2\ -A_3\ \dots\ -A_{n+1}]\begin{bmatrix} y(t) \\ y(t-1) \\ \vdots \\ y(t-n+1)) \end{bmatrix} \tag{A.18}$$

$\hat{y}(t+k)$ is defined (in general) as follows:

$$\hat{y}(t+k) = [X_{1,k}\ X_{2,k}\ \dots\ X_{(n+k),k}]\begin{bmatrix} u(t+k-1) \\ u(t+k-2) \\ \vdots \\ u(t-n+1)) \end{bmatrix} + [Y_{1,k}\ Y_{2,k}\ \dots\ Y_{n,k}]\begin{bmatrix} y(t) \\ y(t-1) \\ \vdots \\ y(t-n+1)) \end{bmatrix} \tag{A.19}$$

Where the variables X and Y are 'dummy' variables which describe the elements in the vector.
If the equation describing the k'th prediction is known, then the (k+1)'th prediction is derived from the following equation:

$$\begin{aligned} \hat{y}(t+k+1) &= q\hat{y}(t+k) \\ &= [X_{1,k}\ X_{2,k}\ \dots\ X_{(n+k),k}]\begin{bmatrix} u(t+k) \\ u(t+k-1) \\ \vdots \\ u(t-n+2) \end{bmatrix} + [Y_{1,k}\ Y_{2,k}\ \dots Y_{n,k}]\begin{bmatrix} \hat{y}(t+1) \\ y(t) \\ \vdots \\ y(t-n+2) \end{bmatrix} \\ &= [X_{1,k}\ X_{2,k}\ \dots\ X_{(n+k),k}]\begin{bmatrix} u(t+k) \\ u(t+k-1) \\ \vdots \\ u(t-n+2) \end{bmatrix} + Y_{1,k}\hat{y}(t+1) + [Y_{2,k}\ \dots\ Y_{n,k}]\begin{bmatrix} y(t) \\ \vdots \\ y(t-n+2) \end{bmatrix} \\ &= [X_{1,k}\ X_{2,k}\ \dots\ X_{(n+k),k}]\begin{bmatrix} u(t+k) \\ u(t+k-1) \\ \vdots \\ u(t-n+2) \end{bmatrix} + [Y_{2,k}\ \dots\ Y_{n,k}]\begin{bmatrix} y(t) \\ \vdots \\ y(t-n+2) \end{bmatrix} \end{aligned}$$

$$+ Y_{1,k}[B_2\ B_3\ \ldots\ B_{n+1}]\begin{bmatrix} u(t) \\ u(t-1) \\ \vdots \\ u(t-n+1) \end{bmatrix} + Y_{1,k}[-A_2\ -A_3\ \ldots\ -A_{n+1}]\begin{bmatrix} y(t) \\ y(t-1) \\ \vdots \\ y(t-n+1) \end{bmatrix} \tag{A.20}$$

Equation A.20 is used to set-up the predictor equation matrices. The initial starting point is defined based on the equation of $\hat{y}(t+1)$.

Key result:

Initial settings for X and Y:

$$\left.\begin{aligned} X_{i,1} &= B_{(i+1)} \\ Y_{i,1} &= -A_{(i+1)} \end{aligned}\right] \quad Where\ i = 1...n \tag{A.21}$$

Recursive formulae:

$$\begin{aligned} \hat{y}(t+k+1) \;=\; & [X_{1,k}\ X_{2,k}\ \ldots\ X_{(n+k),k}\ 0]\begin{bmatrix} u(t+k) \\ u(t+k-1) \\ \vdots \\ u(t-n+2) \\ u(t-n+1) \end{bmatrix} \\ & + Y_{1,k}[B_2\ B_3\ \ldots\ B_{n+1}]\begin{bmatrix} u(t) \\ 1) \\ \\ u(t-n+1) \end{bmatrix} \\ & + [Y_{2,k}\ \ldots\ Y_{n,k}\ 0]\begin{bmatrix} \hat{y}(t) \\ \vdots \\ y(t-n+2) \\ y(t-n+1) \end{bmatrix} \\ & + Y_{1,k}[-A_2\ -A_3\ \ldots\ -A_{n+1}]\begin{bmatrix} y(t) \\ y(t-1) \\ \vdots \\ y(t-n+1) \end{bmatrix} \end{aligned} \tag{A.22}$$

MATLAB coding

Equations A.21 and A.22 are incorporated into a MATLAB code as follows:
Setup the equation describing $\hat{y}(t+1)$, using the recursive formulae to find successive equations. The above formulae operates on the principle that each prediction relies on the previous prediction. In other words if a formulae for $\hat{y}(t+1)$ is known, then using the recursive formulae, an equation for $\hat{y}(t+2)$, and hence all successive equations are computed.

Calculate the equations, using the index range k = 1...(HP-1)
Once this is done, it is a simple procedure to *extract the matrices Q, W and R.* [1]
Refer to the MATLAB program calc.m

[1] The MATLAB code does this in one step.

In order to make the notation used a little clearer, an example is done. This example attempts to explain the notation used and also how the predictor equation is formulated using the Equations A.21 and A.22.

Example: HP=5 and n=4
The following matrix equation shows how the dummy variables X and Y are used to describe equations $\hat{y}(t+1)...\hat{y}(t+HP)$.

$$\begin{bmatrix} \hat{y}(t+1) \\ \hat{y}(t+2) \\ \hat{y}(t+3) \\ \hat{y}(t+4) \\ \hat{y}(t+5) \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & X_{1,1} \\ 0 & 0 & 0 & X_{1,2} & X_{2,2} \\ 0 & 0 & X_{1,3} & X_{2,3} & X_{3,3} \\ 0 & X_{1,4} & X_{2,4} & X_{3,4} & X_{4,4} \\ X_{1,5} & X_{2,5} & X_{3,5} & X_{4,5} & X_{5,5} \end{bmatrix}}_{\text{W matrix}} \underbrace{\begin{bmatrix} X_{2,1} & X_{3,1} & X_{4,1} \\ X_{3,2} & X_{4,2} & X_{5,2} \\ X_{4,3} & X_{5,3} & X_{6,3} \\ X_{5,4} & X_{6,4} & X_{7,4} \\ X_{6,5} & X_{7,5} & X_{8,5} \end{bmatrix}}_{\text{R matrix}} \begin{bmatrix} u(t+4) \\ u(t+3) \\ u(t+2) \\ u(t+1) \\ u(t) \\ u(t-1) \\ u(t-2) \\ u(t-3) \end{bmatrix}$$

$$+ \underbrace{\begin{bmatrix} Y_{1,1} & Y_{2,1} & Y_{3,1} & Y_{4,1} \\ Y_{1,2} & Y_{2,2} & Y_{3,2} & Y_{4,2} \\ Y_{1,3} & Y_{2,3} & Y_{3,3} & Y_{4,3} \\ Y_{1,4} & Y_{2,4} & Y_{3,4} & Y_{4,4} \\ Y_{1,5} & Y_{2,5} & Y_{3,5} & Y_{4,5} \end{bmatrix}}_{\text{Q matrix}} \begin{bmatrix} y(t) \\ y(t-1) \\ y(t-2) \\ y(t-3) \end{bmatrix} \tag{A.23}$$

The initial starting point is specified by the equation describing $\hat{y}(t+1)$

$$\hat{y}(t+1) = [B_2 \; B_3 \; B_4 \; B_5] \begin{bmatrix} u(t) \\ u(t-1) \\ u(t-2) \\ u(t-3) \end{bmatrix} + [-A_2 \; -A_3 \; -A_4 \; -A_5] \begin{bmatrix} y(t) \\ y(t-1) \\ y(t-2) \\ y(t-3) \end{bmatrix} \tag{A.24}$$

The values of $X_{1,1}...X_{4,1}$ and $Y_{1,1}...Y_{4,1}$ may be obtained from Equation A.24. Each of the predicted equations ($\hat{y}(t+k)$ where k = 1...HP), are derived using the recursive equation (Equation A.22).

Once these equations are derived, the matrices W,R and Q are extracted from the set of equations as shown in Equation A.23.

Testing of code:

In order to assess the functionality of the MATLAB code (`calc.m`), a test program is written in which the following is done:

- Using the predictor equation, the response of a system to a step change in the input is predicted.
- The linear system is simulated using the MATLAB `lsim.m` program.
- Compare the results.

Test:

Arbitrary plant model:

$$P(s) = \frac{600(s+5)}{(s+10)(s^2+2(0.3)s+10^2)} \tag{A.25}$$

Initial input : 0.5
Initial output: 1.5
Step change of input signal to plant: 0.5 ... 1.0 at time t = 0;

Figure A.1 shows the results of such a test (refer to `stest.m`). The results indicate that the predictor equations predictions are identical to the results obtained by simulating the system. In other words the predictor equation for the SISO system is correct.

Results: * = Predictor equation.
line = Simulated system.

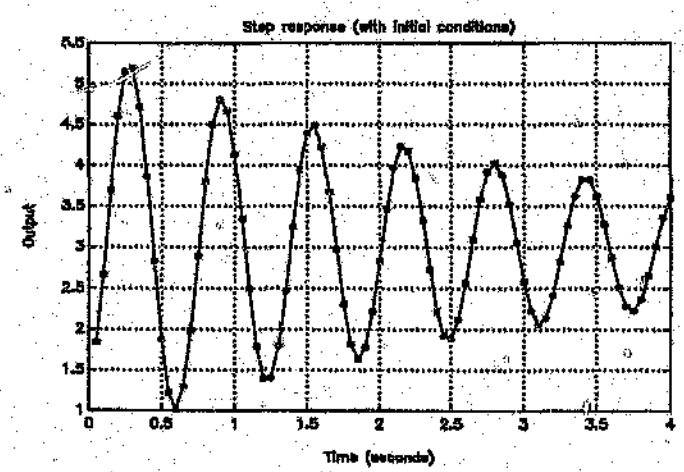


Figure A.1: Test of predictor equation (SISO system)

A.2 Predictor Equation: MIMO system

The predictor equation for the SISO case is derived in section A.1. As already mentioned, the conversion from SISO to MIMO is easily done. This section gives a detailed explanation of the conversion.

Conversion from SISO to MIMO
The basic structure used in the SISO case is extended to cater for MIMO systems by converting the elements in the formulae from scalar variables to matrix variables. This section outlines the rules for handling the MIMO conversion, concluding the analysis with an example of the predictor equation used in MIMO systems.

MIMO system representation
The transfer function (s domain) of a MIMO system is usually written:

$$P(s) = \frac{\begin{bmatrix} num_{11}(s) & num_{12}(s) & \dots & num_{1l}(s) \\ num_{21}(s) & num_{22}(s) & \dots & num_{2l}(s) \\ \vdots & \vdots & & \vdots \\ num_{m1}(s) & num_{m2}(s) & \cdots & num_{ml}(s) \end{bmatrix}}{den(s)} \tag{A.26}$$

Where: m = Number of plant outputs.
l = Number of plant inputs.

It is possible to transform each of the sub-transfer functions into its discrete transfer function form:
Example:

$$\frac{num11(s)}{den(s)} \Longrightarrow \frac{B_{11}(q^{-1})}{A(q^{-1})} \tag{A.27}$$

If sub-transfer functions share the same common denominator, then the sub-transfer functions in the discrete time domain has the same $A(q^{-1})$.

The transfer function (in discrete time) of a system is written:

$$\mathbf{y}(t) = \frac{\mathbf{B}(q^{-1})}{\mathbf{A}(q^{-1})}\mathbf{u}(t) \tag{A.28}$$

Where:

$$\mathbf{A}(q^{-1}) = \mathbf{A}_1 + \mathbf{A}_2 q^{-1} + \mathbf{A}_3 q^{-2} + ... + \mathbf{A}_{n+1} q^{-n} \tag{A.29}$$

$$\mathbf{B}(q^{-1}) = \mathbf{B}_1 + \mathbf{B}_2 q^{-1} + \mathbf{B}_3 q^{-2} + ... + \mathbf{B}_{n+1} q^{-n} \tag{A.30}$$

When considering MIMO systems, the same equations are used.

Rules for extension from SISO to MIMO

The following extension to Equations A.28,A.29 and A.30 are made when considering MIMO systems:

1. Each element in the equation (Eg. $\mathbf{A}_1$,$\mathbf{B}_1$) is treated as a matrix.
2. Each element of B is defined:

$$\mathbf{B}_x = \begin{bmatrix} B_{11_x} & B_{12_x} & \cdots & B_{1l_x} \\ B_{21_x} & B_{22_x} & \cdots & B_{2l_x} \\ \vdots & \vdots & & \vdots \\ B_{m1_x} & B_{m2_x} & \cdots & B_{ml_x} \end{bmatrix}, \qquad x = 1...(n+1) \tag{A.31}$$

3. Each element of A is defined:

$$\mathbf{A}_x = A_x \mathbf{I}(m), \qquad x = 1...(n+1) \tag{A.32}$$

   I(m) = identity matrix of size m.

4. The variable y(t+k) is defined (k is a dummy variable):

$$\mathbf{y}(t+k) = \begin{bmatrix} y_1(t+k) \\ y_2(t+k) \\ \vdots \\ y_m(t+k) \end{bmatrix} \tag{A.33}$$

5. The variable u(t+k) is defined (k is a dummy variable):

$$\mathbf{u}(t+k) = \begin{bmatrix} u_1(t+k) \\ u_2(t+k) \\ \vdots \\ u_l(t+k) \end{bmatrix} \tag{A.34}$$

The following example explains the notation:

Example:
Consider the following MIMO system.

$$P(s) = \begin{bmatrix} \frac{-2.1}{16.7s+1} & \frac{3.1}{21s+1} \\ \frac{-0.52}{10.9s+1} & \frac{1.52}{14.4s+1} \end{bmatrix}$$

$$= \frac{\begin{bmatrix} -6921.24s^3 - 1445.35s^2 - 97.23s - 2.1 & 8125.82s^3 + 1796.36s^2 + 130.20s + 3.1 \\ -2626.04s^3 - 464.66s^2 - 27.09s - 0.52 & 5810.40s^3 + 1157.68s^2 + 73.87s + 1.52 \end{bmatrix}}{55045.87s^4 + 14790.10s^3 + 1461.47s^2 + 63.00s + 1}$$

$$= \frac{\begin{bmatrix} num_{11}(s) & num_{12}(s) \\ num_{21}(s) & num_{21}(s) \end{bmatrix}}{den(s)}$$

Conversion to discrete time
Sampling period: T = 5 seconds

$$\frac{num_{11}(s)}{den(s)} \rightarrow \frac{B_{11_1} + B_{11_2}q^{-1} + B_{11_3}q^{-2} + B_{11_4}q^{-3} + B_{11_5}q^{-4}}{A_1 + A_2q^{-1} + A_3q^{-2} + A_4q^{-3} + A_5q^{-4}}$$

$$\rightarrow \frac{0 - 0.5433q^{-1} + 1.556q^{-2} - 0.8160q^{-3} + 0.1913q^{-4}}{1 - 2.8681q^{-1} + 3.0783q^{-2} - 1.4652q^{-3} + 0.2609q^{-4}}$$

$$\frac{num_{12}(s)}{den(s)} \rightarrow \frac{B_{12_1} + B_{12_2}q^{-1} + B_{12_3}q^{-2} + B_{12_4}q^{-3} + B_{12_5}q^{-4}}{A_1 + A_2q^{-1} + A_3q^{-2} + A_4q^{-3} + A_5q^{-4}}$$

$$\rightarrow \frac{0 + 0.6568q^{-1} - 1.3662q^{-2} + 0.9452q^{-3} - 0.2175q^{-4}}{1 - 2.8681q^{-1} + 3.0783q^{-2} - 1.4652q^{-3} + 0.2609q^{-4}}$$

$$\frac{num_{21}(s)}{den(s)} \rightarrow \frac{B_{21_1} + B_{21_2}q^{-1} + B_{21_3}q^{-2} + B_{21_4}q^{-3} + B_{21_5}q^{-4}}{A_1 + A_2q^{-1} + A_3q^{-2} + A_4q^{-3} + A_5q^{-4}}$$

$$\rightarrow \frac{0 - 0.1913q^{-1} + 0.4278q^{-2} - 0.3185q^{-3} + 0.0790q^{-4}}{1 - 2.8681q^{-1} + 3.0783q^{-2} - 1.4652q^{-3} + 0.2609q^{-4}}$$

$$\frac{num_{22}(s)}{den(s)} \rightarrow \frac{B_{22_1} + B_{22_2}q^{-1} + B_{22_3}q^{-2} + B_{22_4}q^{-3} + B_{22_5}q^{-4}}{A_1 + A_2q^{-1} + A_3q^{-2} + A_4q^{-3} + A_5q^{-4}}$$

$$\rightarrow \frac{0 + 0.4459q^{-1} - 0.9638q^{-2} + 0.6916q^{-3} - 0.1647q^{-4}}{1 - 2.8681q^{-1} + 3.0783q^{-2} - 1.4652q^{-3} + 0.2609q^{-4}}$$

Matrix variables:
According to the rules, the variables in the MIMO transfer function are defined as follows:

$$\mathbf{B_1} = \begin{bmatrix} B_{11_1} & B_{12_1} \\ B_{21_1} & B_{22_1} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad \mathbf{A_1} = \begin{bmatrix} A_1 & 0 \\ 0 & A_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{B_2} = \begin{bmatrix} B_{11_2} & B_{12_2} \\ B_{21_2} & B_{22_2} \end{bmatrix} = \begin{bmatrix} -0.5422 & 0.6568 \\ -0.1913 & 0.4459 \end{bmatrix} \quad \mathbf{A_2} = \begin{bmatrix} A_2 & 0 \\ 0 & A_2 \end{bmatrix} = \begin{bmatrix} -2.8681 & 0 \\ 0 & -2.8681 \end{bmatrix}$$

$$\mathbf{B_3} = \begin{bmatrix} B_{11_3} & B_{12_3} \\ B_{21_3} & B_{22_3} \end{bmatrix} = \begin{bmatrix} 1.556 & -1.3662 \\ 0.4278 & -0.9638 \end{bmatrix} \quad \mathbf{A_3} = \begin{bmatrix} A_3 & 0 \\ 0 & A_3 \end{bmatrix} = \begin{bmatrix} 3.0783 & 0 \\ 0 & 3.0783 \end{bmatrix}$$

$$\mathbf{B_4} = \begin{bmatrix} B_{11_4} & B_{12_4} \\ B_{21_4} & B_{22_4} \end{bmatrix} = \begin{bmatrix} -0.8160 & 0.9452 \\ -0.3185 & 0.6916 \end{bmatrix} \quad \mathbf{A_4} = \begin{bmatrix} A_4 & 0 \\ 0 & A_4 \end{bmatrix} = \begin{bmatrix} -1.4652 & 0 \\ 0 & -1.4652 \end{bmatrix}$$

$$\mathbf{B_5} = \begin{bmatrix} B_{11_5} & B_{12_5} \\ B_{21_5} & B_{22_5} \end{bmatrix} = \begin{bmatrix} 0.1913 & -0.2175 \\ 0.0790 & -0.1647 \end{bmatrix} \quad \mathbf{A_5} = \begin{bmatrix} A_5 & 0 \\ 0 & A_5 \end{bmatrix} = \begin{bmatrix} 0.2609 & 0 \\ 0 & 0.2609 \end{bmatrix}$$

The input and output vectors are defined as follows:

$$\mathbf{y(t)} = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} \qquad \mathbf{u(t)} = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

$$\mathbf{y(t-1)} = \begin{bmatrix} y_1(t-1) \\ y_2(t-1) \end{bmatrix} \qquad \mathbf{u(t-1)} = \begin{bmatrix} u_1(t-1) \\ u_2(t-1) \end{bmatrix}$$

$$\mathbf{y(t-2)} = \begin{bmatrix} y_1(t-2) \\ y_2(t-2) \end{bmatrix} \qquad \mathbf{u(t-2)} = \begin{bmatrix} u_1(t-2) \\ u_2(t-2) \end{bmatrix}$$

$$\mathbf{y(t-3)} = \begin{bmatrix} y_1(t-3) \\ y_2(t-3) \end{bmatrix} \qquad \mathbf{u(t-3)} = \begin{bmatrix} u_1(t-3) \\ u_2(t-3) \end{bmatrix}$$

$$\mathbf{y(t-4)} = \begin{bmatrix} y_1(t-4) \\ y_2(t-4) \end{bmatrix} \qquad \mathbf{u(t-4)} = \begin{bmatrix} u_1(t-4) \\ u_2(t-4) \end{bmatrix}$$

The above y and u matrices are now used to derive the predictor equation. The procedure of deriving the equation is identical to that of the complementary SISO case. The only difference is that in the SISO case, the elements inside the matrices were scalar variables. In the MIMO case these elements are now matrices.

Matlab Coding

The MATLAB program, `calcm.m`, which derives the predictor equation for the MIMO case can be found in the MATLAB appendix.

Unfortunately, MATLAB does not allow elements in a matrix to be defined as matrix elements. So the explanation given which converts SISO to MIMO is implemented exactly as explained. The way around this problem is to split the matrix up into sections, where each section can be treated as a matrix. This issue is not elaborated on any further, as the key issue is to use matrices as the base variable. If this code were to be re-written using a language such as Turbo C++, then this architecture would be easily obtained.

Testing of code:

In order to assess the functionality of the MATLAB code (`calcm.m`), a test program is written which performs the following:

- Using the predictor equation, predict the response of the system to a step change in the plants inputs.
- Simulate the linear system using MATLAB's lsim.m program.
- Compare the results.

Test:

Arbitrary plant model:

$$P(s) = \begin{bmatrix} \frac{-2.1}{16.7s+1} & \frac{3.1}{21s+1} \\ \frac{-0.52}{10.9s+1} & \frac{1.52}{14.4s+1} \end{bmatrix} \tag{A.35}$$

(A.36)

Initial inputs: input1 = 1
input2 = 2

Step change of input signal to plant at time t = 0.

New inputs: input1 = 0.5
input2 = 1

The following graph are the results of such a test (refer to `mtest.m`). The results indicate that the predictor equation is identical to the simulated system. In other words the predictor equation for the MIMO algorithm is correct.

Results: * = Predictor equation.
line = Simulated system.

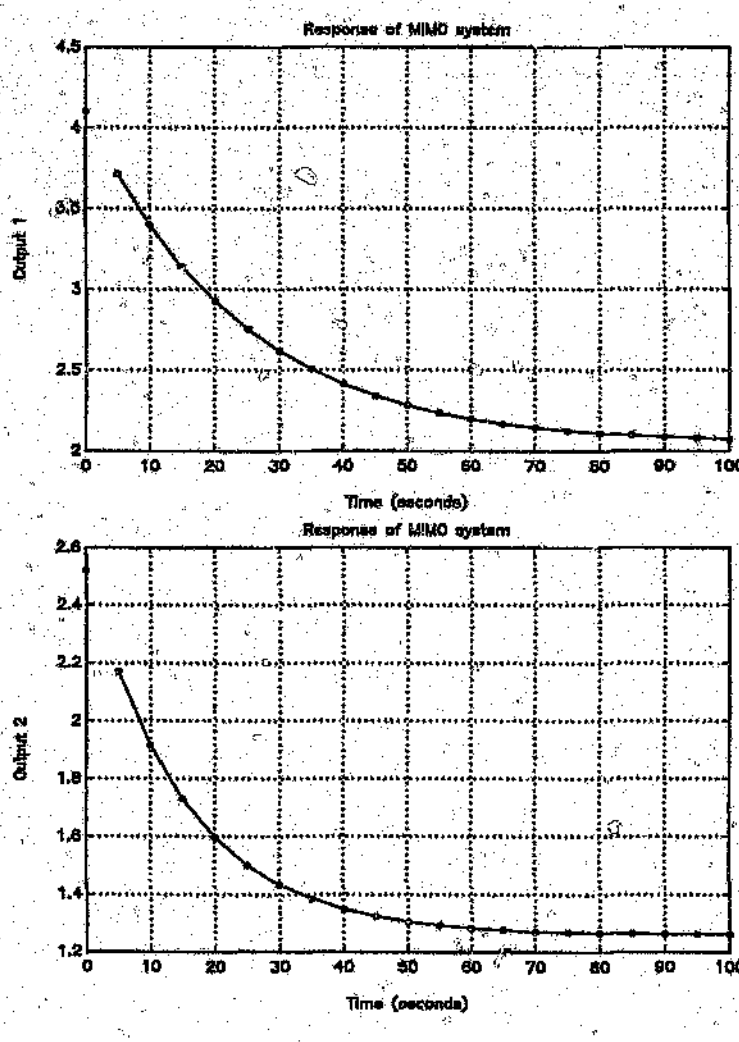


Figure A.2: Test of predictor equation (MIMO system)

Appendix B

Simulation results

This appendix gives the results of simulations carried out using the program `simulate.m`.

B.1 Test1

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-6Ts}}{(14.9s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-Ts}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \qquad \text{(B.1)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(17s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \qquad \text{(B.2)}$$

Pre-amble settings

HP	= 15	T	= 5 seconds
Ysp	= $[0\ 1]^T$	tau	= $[50\ 50]^T$ seconds
Umax	= $[5\ 5]^T$	Umin	= $[-1\ -1]^T$
no_const	= 5	no_steps	= 200

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1 + 0.01x$ x=1...HP
$\beta_2(x) = 0.1 + 0.01x$

λ $\lambda_1(x) = 1$ x=1...HP
$\lambda_2(x) = 1$

Simulation results

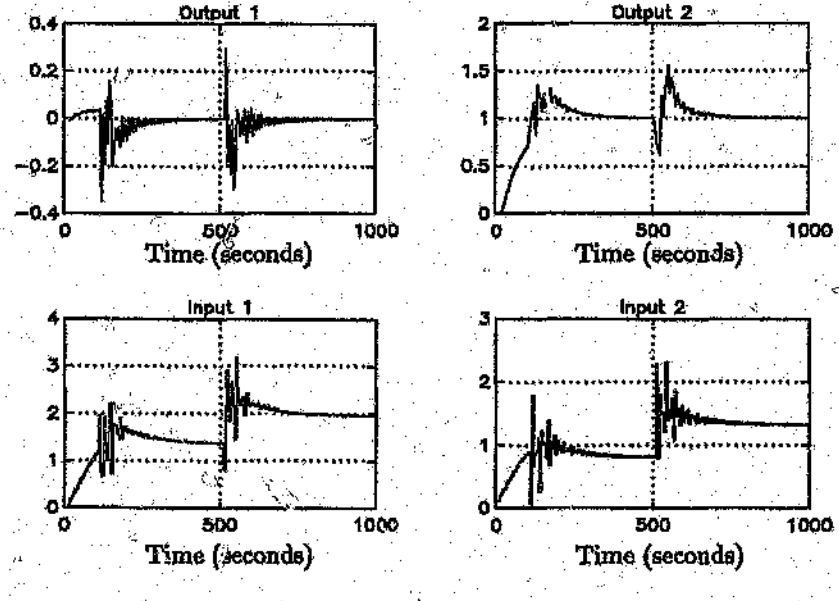


Figure B Simulation results of test1

B.2 Test2

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-8Ts}}{(14.9s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-Ts}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \qquad \text{(B.3)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(17s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(t) \end{bmatrix} \qquad \text{(B.4)}$$

Pre-amble settings

HP	$= 10$	T	$= 5$ seconds
ump	$= [0\ 1]^T$	tau	$= [50\ 50]^T$ seconds
Umax	$= [2.2\ 2]^T$	Umin	$= [0\ 0]^T$
no_const	$= 1$	no_steps	$= 200$

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1$, $\beta_2(x) = 0.1$ x=1...HP

λ $\lambda_1(x) = 1$, $\lambda_2(x) = 1$ x=1...HP

Simulation results

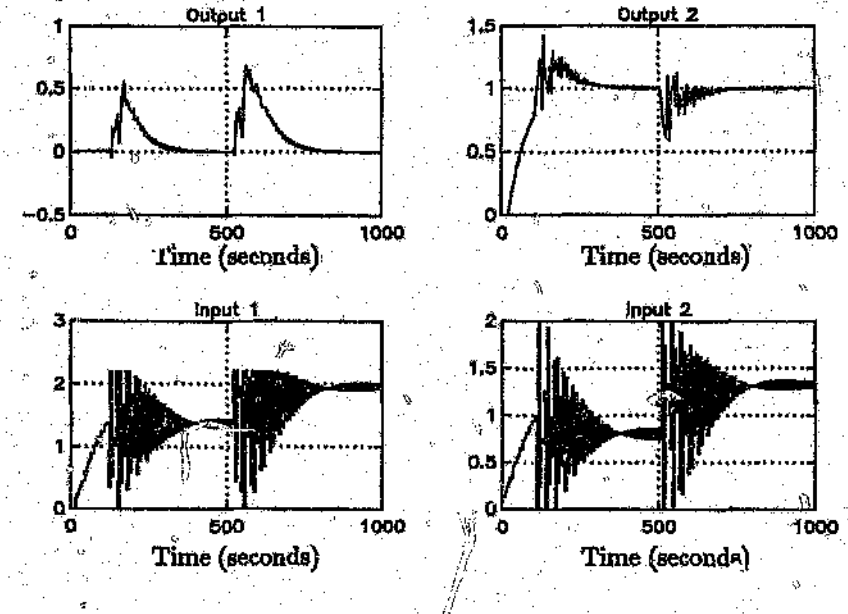


Figure B.2: Simulation results of test2

B.3 Test3

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3s}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-8s}}{(14.9s+1)} \\ \frac{-0.52e^{-7s}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3s}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-2s}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.5)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(17s+1)} & \frac{3.1e^{-3s}}{(21s+1)} \\ \frac{-0.52e^{-7s}}{(10.9s+1)} & \frac{1.52e^{-3s}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.6)}$$

Pre-amble settings

HP	= 10	T	= 5 seconds
Ysp	= $[6\ 1]^T$	tau	= $[50\ 50]^T$ seconds
Umax	= $[2.2\ 2]^T$	Umin	= $[0\ 0]^T$
no_const	= 2	no_steps	= 200

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1$, $\beta_2(x) = 0.1$ x=1...HP

λ $\lambda_1(x) = 1$, $\lambda_2(x) = 1$ x=1...HP

Simulation results

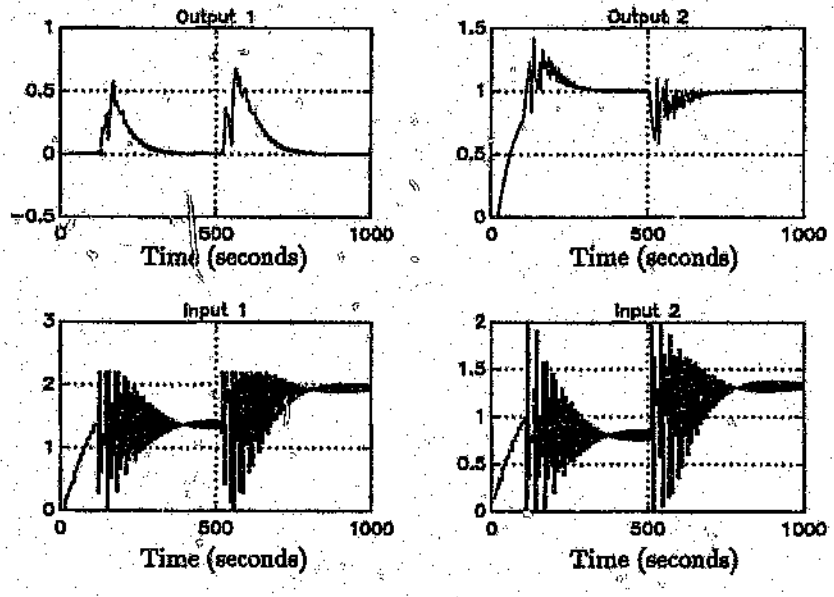


Figure B.3: Simulation results of test3

B.4 Test4

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3T_s}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-3T_s}}{(14.9s+1)} \\ \frac{-0.52e^{-7T_s}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3T_s}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-T_s}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.7)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(17s+1)} & \frac{3.1e^{-3T_s}}{(21s+1)} \\ \frac{-0.52e^{-7T_s}}{(10.9s+1)} & \frac{1.52e^{-3T_s}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.8)}$$

Pre-amble settings

HP	$= 10$	T	$= 5$ seconds
Ysp	$= [0\ 1]^T$	tau	$= [50\ 50]^T$ seconds
Umax	$= [2.2\ 2]^T$	Umin	$= [0\ 0]^T$
no_const	$= 3$	no_steps	$= 200$

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1$, $\beta_2(x) = 0.1$ x=1...HP

λ $\lambda_1(x) = 1$, $\lambda_2(x) = 1$ x=1...HP

Simulation results

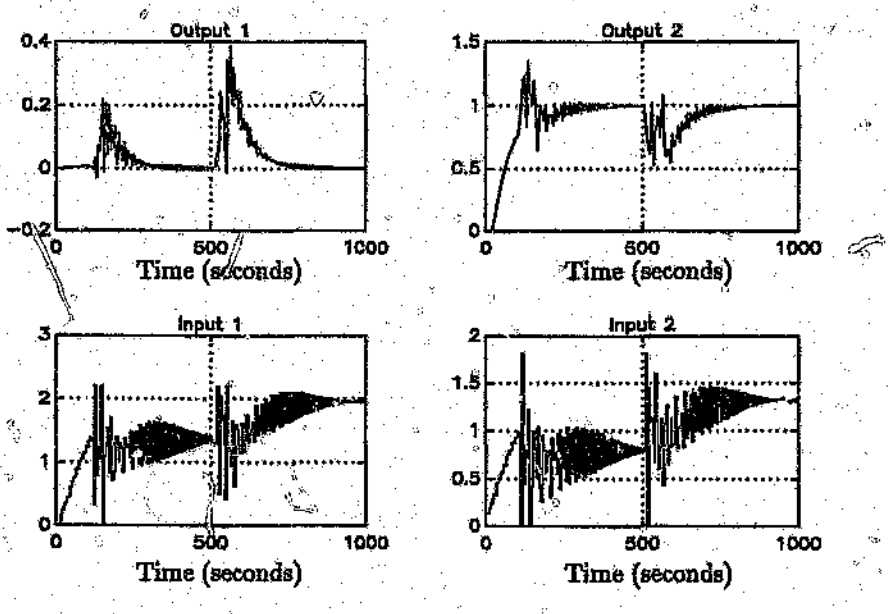


Figure B.4: Simulation results of test4

B.5 Test5

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3s}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-5s}}{(14.9s+1)} \\ \frac{-0.52e^{-7s}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3s}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-7s}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.9)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(17s+1)} & \frac{3.1e^{-3s}}{(21s+1)} \\ \frac{-0.52e^{-7s}}{(10.9s+1)} & \frac{1.52e^{-3s}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.10)}$$

Pre-amble settings

HP	$= 10$	T	$= 5$ seconds
Ysp	$= [0\ 1]^T$	tau	$= [50\ 50]^T$ seconds
Umax	$= [2.2\ 2]^T$	Umin	$= [0\ 0]^T$
no_const	$= 4$	no_steps	$= 200$

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1$, $\beta_2(x) = 0.1$ x=1...HP

λ $\lambda_1(x) = 1$, $\lambda_2(x) = 1$ x=1...HP

Simulation results

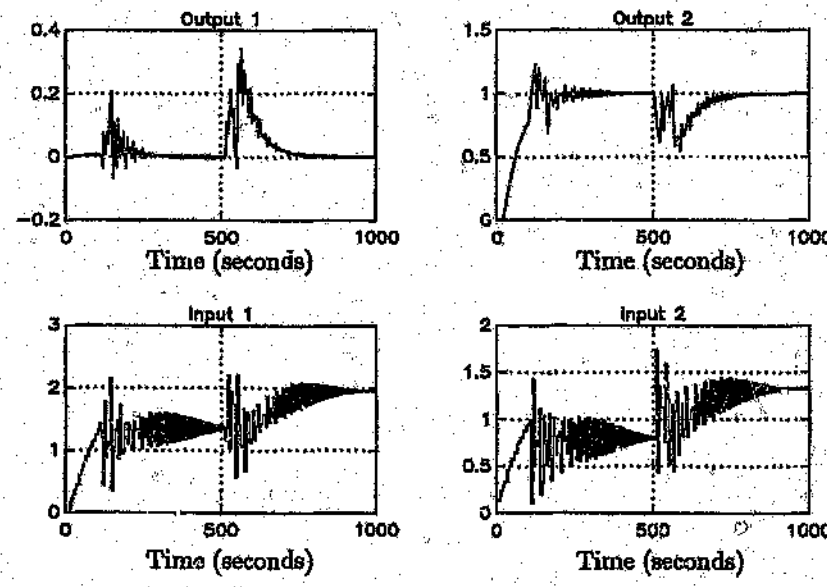


Figure B.5: Simulation results of test5

B.6 Test6

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3T_s}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-8T_s}}{(14.9s+1)} \\ \frac{-0.52e^{-7T_s}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3T_s}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-T_s}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.11)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(17s+1)} & \frac{3.1e^{-3T_s}}{(21s+1)} \\ \frac{-0.52e^{-7T_s}}{(10.9s+1)} & \frac{1.52e^{-3T_s}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.12)}$$

Pre-amble settings

HP	$= 10$	T	$= 5$ seconds
Ysp	$= [0\ 1]^T$	tau	$= [50\ 50]^T$ seconds
Umax	$= [2.2\ 2]^T$	Umin	$= [0\ 0]^T$
no_const	$= 5$	no_steps	$= 200$

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1$, $\beta_2(x) = 0.1$ x=1...HP

λ $\lambda_1(x) = 1$, $\lambda_2(x) = 1$ x=1...HP

Simulation results

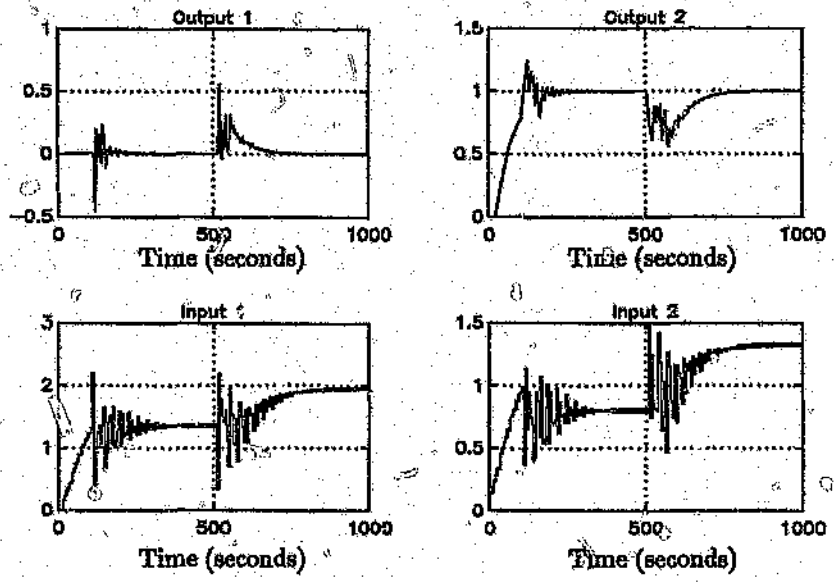


Figure B.6: Simulation results of test6

B.7 Test7

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-8Ts}}{(14.9s+1)} \\ \frac{-0.62e^{-7Ts}}{(10.9s+1)(0.1s+1)} & \frac{1.32e^{-3Ts}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-Ts}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.13)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(7s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)} \\ \frac{-0.62e^{-7Ts}}{(10.9s+1)} & \frac{1.32e^{-3Ts}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.14)}$$

Pre-amble settings

HP	$= 10$	T	$= 5$ seconds
Ysp	$= [0\ 1]^T$	tau	$= [50\ 50]^T$ seconds
Umax	$= [2.2\ 2]^T$	Umin	$= [0\ 0]^T$
no_const	$= 6$	no_steps	$= 200$

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1$, $\beta_2(x) = 0.1$ x=1...HP

λ $\lambda_1(x) = 1$, $\lambda_2(x) = 1$ x=1...HP

Simulation results

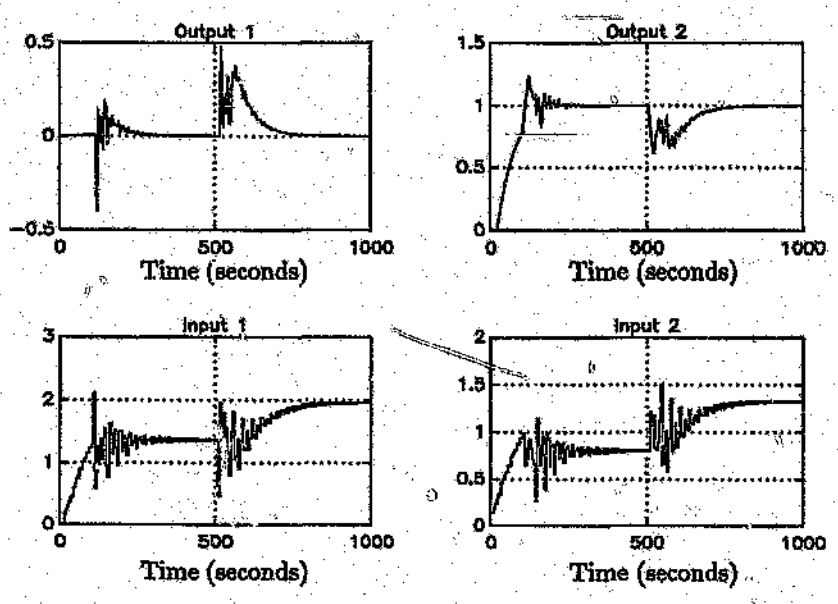


Figure B.7: Simulation results of test7

B.8 Test8

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{2.1e^{-3s}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-5s}}{(14.9s+1)} \\ \frac{-0.52e^{-7s}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3s}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-s}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.15)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(17s+1)} & \frac{3.1e^{-3s}}{(21s+1)} \\ \frac{-0.52e^{-7s}}{(10.9s+1)} & \frac{1.52e^{-3s}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.16)}$$

Pre-amble settings

HP	= 10	T	= 5 seconds
Ysp	= $[0\ 1]^T$	tau	= $[50\ 50]^T$ seconds
Umax	= $[2.2\ 2]^T$	Umin	= $[0\ 0]^T$
no_const	= 7	no_steps	= 200

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1$, $\beta_2(x) = 0.1$ x=1...HP

λ $\lambda_1(x) = 1$, $\lambda_2(x) = 1$ x=1...HP

Simulation results

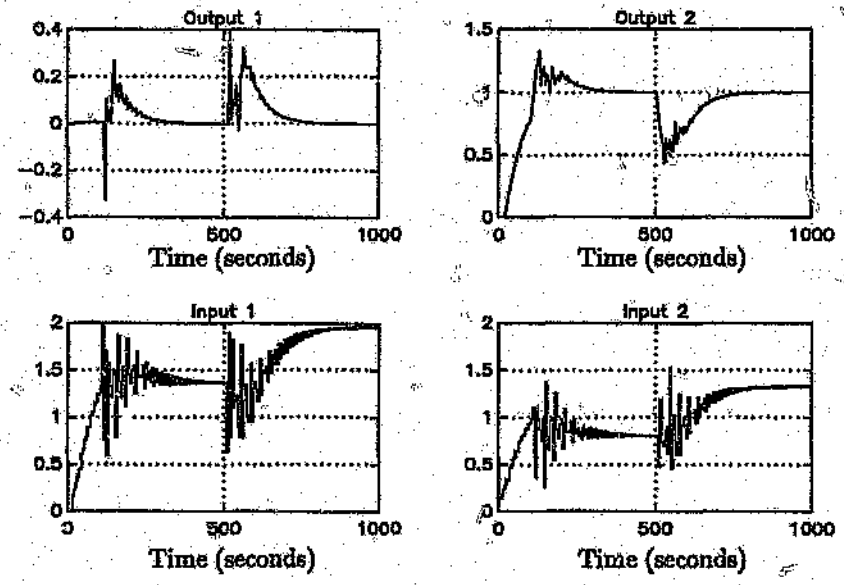


Figure B.8: Simulation results of test8

B.9 Test9

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3T_s}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-8T_s}}{(14.9s+1)} \\ \frac{-0.52e^{-7T_s}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3T_s}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-T_s}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.17)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(17s+1)} & \frac{3.1e^{-3T_s}}{(21s+1)} \\ \frac{-0.52e^{-7T_s}}{(10.9s+1)} & \frac{1.52e^{-3T_s}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.18)}$$

Pre-amble settings

HP	$= 10$	T	$= 5$ seconds
Ysp	$= [0\ 1]^T$	tau	$= [50\ 50]^T$ seconds
Umax	$= [2.2\ 2]^T$	Umin	$= [0\ 0]^T$
no_const	$= 8$	no_steps	$= 200$

d = zero from 1.. 20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1$, $\beta_2(x) = 0.1$ x=1...HP

λ $\lambda_1(x) = 1$, $\lambda_2(x) = 1$ x=1...HP

Simulation results

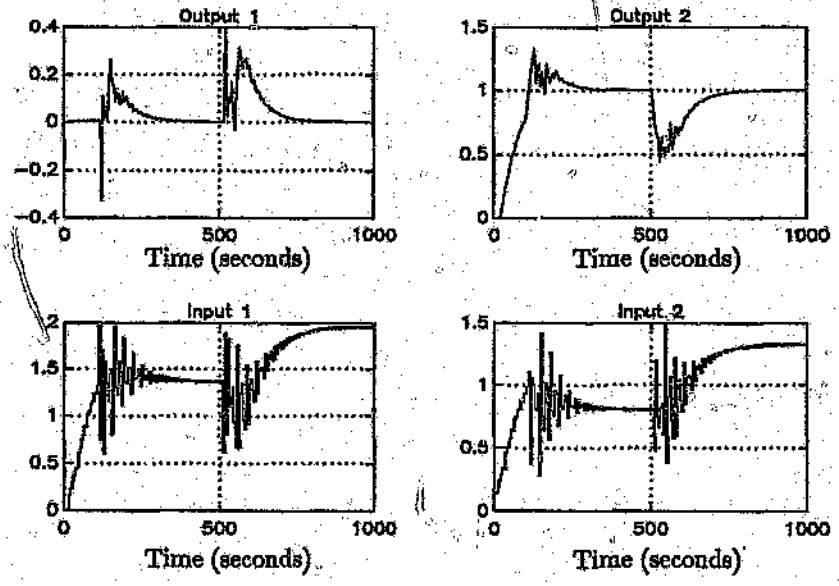


Figure B.9: Simulation results of test9

B.10 Test10

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-8Ts}}{(14.9s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-Ts}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.19)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(17s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.20)}$$

Pre-amble settings

HP	$= 10$	T	$= 5$ seconds
Ysp	$= [0\ 1]^T$	tau	$= [50\ 50]^T$ seconds
Umax	$= [2.2\ 2]^T$	Umin	$= [0\ 0]^T$
no_const	$= 9$	no_steps	$= 200$

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1$, $\beta_2(x) = 0.1$ x=1...HP

λ $\lambda_1(x) = 1$, $\lambda_2(x) = 1$ x=1...HP

Simulation results

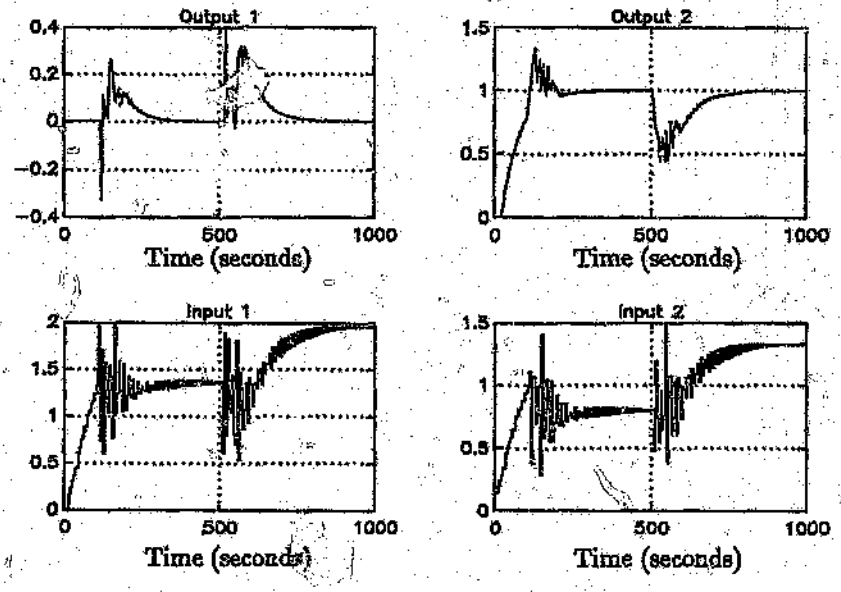


Figure B.10: Simulation results of test10

B.11 Test11

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3s}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-6s}}{(14.9s+1)} \\ \frac{-0.52e^{-7s}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3s}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-7s}}{(18.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.21)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(17s+1)} & \frac{3.1e^{-3s}}{(21s+1)} \\ \frac{-0.52e^{-7s}}{(10.9s+1)} & \frac{1.52e^{-3s}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.22)}$$

Pre-amble settings

HP	= 10	T	= 5 seconds
Ysp	= $[0\ 1]^T$	tau	= $[50\ 50]^T$ seconds
Umax	= $[2.2\ 2]^T$	Umin	= $[0\ 0]^T$
no_const	= 5	no_steps	= 200

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1 + 0.01x$
$\beta_2(x) = 0.1 + 0.01x$ x=1...HP

λ $\lambda_1(x) = 1$
$\lambda_2(x) = 1$ x=1...HP

Simulation results

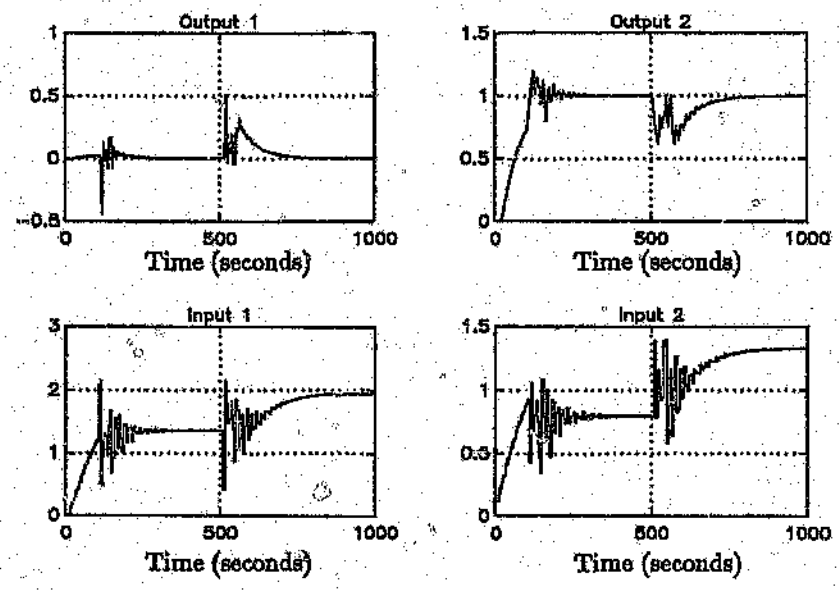


Figure B.11: Simulation results of test11

B.12 Test12

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-3Ts}}{(14.9s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-Ts}}{(18.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.23)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-Ts}}{(17s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.24)}$$

Pre-amble settings

HP	= 15	T	= 5 seconds
Ysp	= $[0\ 1]^T$	tau	= $[50\ 50]^T$ seconds
Umax	= $[2.2\ 2]^T$	Umin	= $[0\ 0]^T$
no_const	= 5	no_steps	= 200

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1 + 0.01x$, $\beta_2(x) = 0.1 + 0.01x$ x=1...HP

λ $\lambda_1(x) = 1$, $\lambda_2(x) = 1$ x=1...HP

Simulation results

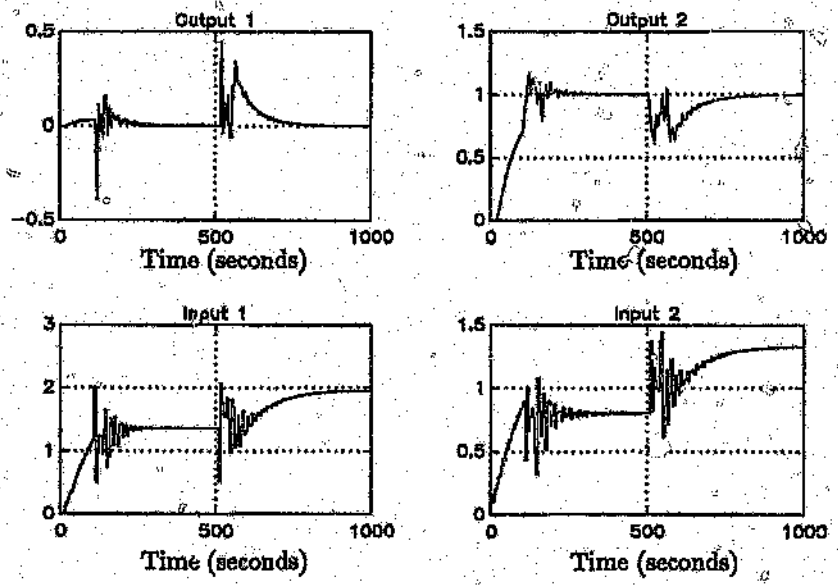


Figure B.12: Simulation results of test12

B.13 Test13

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3s}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-6s}}{(14.9s+1)} \\ \frac{-0.52e^{-7s}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3s}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-2s}}{(18.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.25)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(17s+1)} & \frac{3.1e^{-3s}}{(21s+1)} \\ \frac{-0.52e^{-7s}}{(10.9s+1)} & \frac{1.52e^{-3s}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.26)}$$

Pre-amble settings

HP	$= 20$	T	$= 5$ seconds
Ysp	$= [0\ 1]^T$	tau	$= [50\ 50]^T$ seconds
Umax	$= [2.2\ 2]^T$	Umin	$= [0\ 0]^T$
no_const	$= 5$	no_steps	$= 200$

d = zero from 1...20, one from 20...100, zero from 100...200

β $\quad \beta_1(x) = 0.1 + 0.01x$, $\beta_2(x) = 0.1 + 0.01x$ $\quad$ x=1...HP

λ $\quad \lambda_1(x) = 1$, $\lambda_2(x) = 1$ $\quad$ x=1...HP

Simulation results

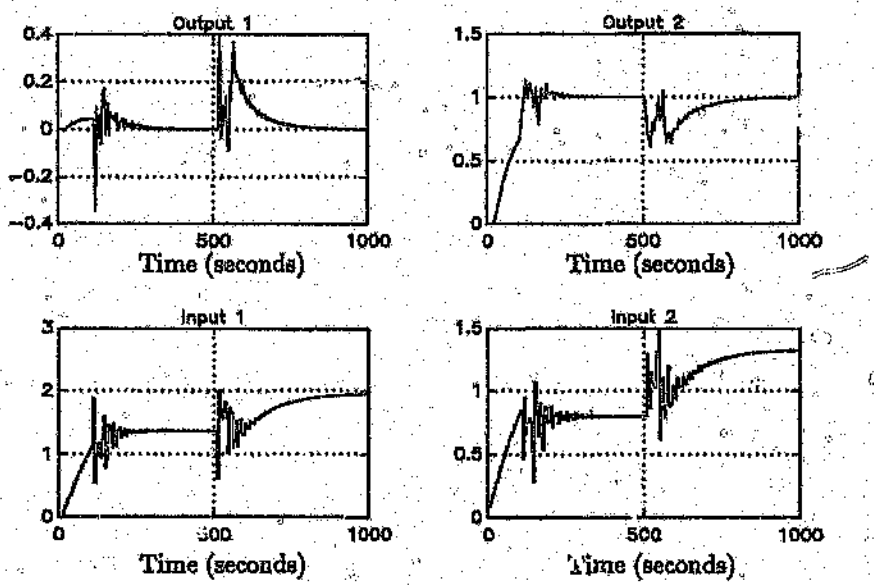


Figure B.13: Simulation results of test13

B.14 Test14

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-8Ts}}{(14.9s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-7s}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.27)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(17s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.28)}$$

Pre-amble settings

HP	$= 25$	T	$= 5$ seconds
Ysp	$= [0\ 1]^T$	tau	$= [50\ 50]^T$ seconds
Umax	$= [2.2\ 2]^T$	Umin	$= [0\ 0]^T$
no_const	$= 5$	no_steps	$= 200$

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1 + 0.01x$, $\beta_2(x) = 0.1 + 0.01x$ x=1...HP

λ $\lambda_1(x) = 1$, $\lambda_2(x) = 1$ x=1...HP

Simulation results

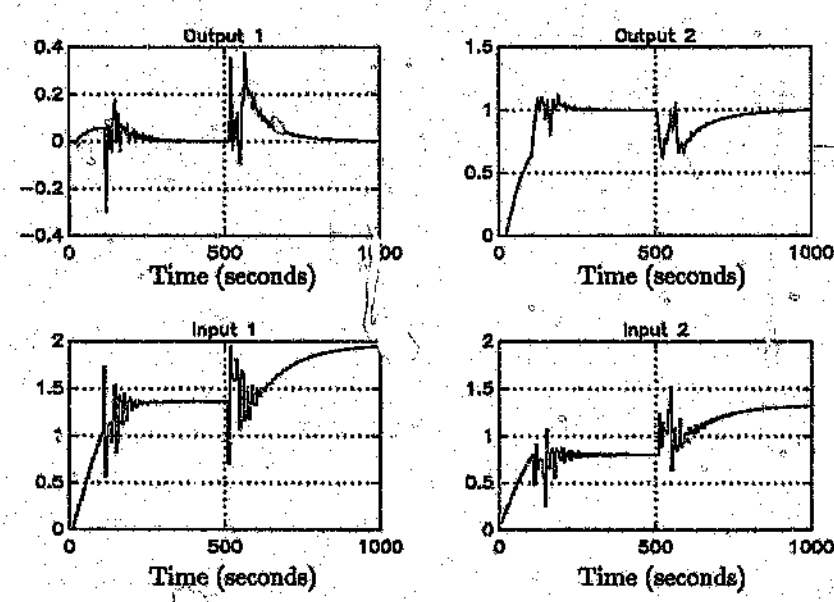


Figure B.14: Simulation results of test14

B.15 Test15

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3s}}{(21s+1)(0.1s+1)} & \frac{3.6e^{-8s}}{(14.9s+1)} \\ \frac{-0.52e^{-7s}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3s}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-7s}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.29)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(17s+1)} & \frac{3.1e^{-3s}}{(21s+1)} \\ \frac{-0.52e^{-7s}}{(10.9s+1)} & \frac{1.52e^{-3s}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.30)}$$

Pre-amble settings

HP	$= 15$	T	$= 5$ seconds
Ysp	$= [0\ 1]^T$	tau	$= [50\ 50]^T$ seconds
Umax	$= [5\ 5]^T$	Umin	$= [-1\ -1]^T$
no_const	$= 5$	no_steps	$= 200$

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.001 + 0.001x$
$\beta_2(x) = 0.001 + 0.001x$ x=1...HP

λ $\lambda_1(x) = 1$
$\lambda_2(x) = 1$ x=1...HP

Simulation results

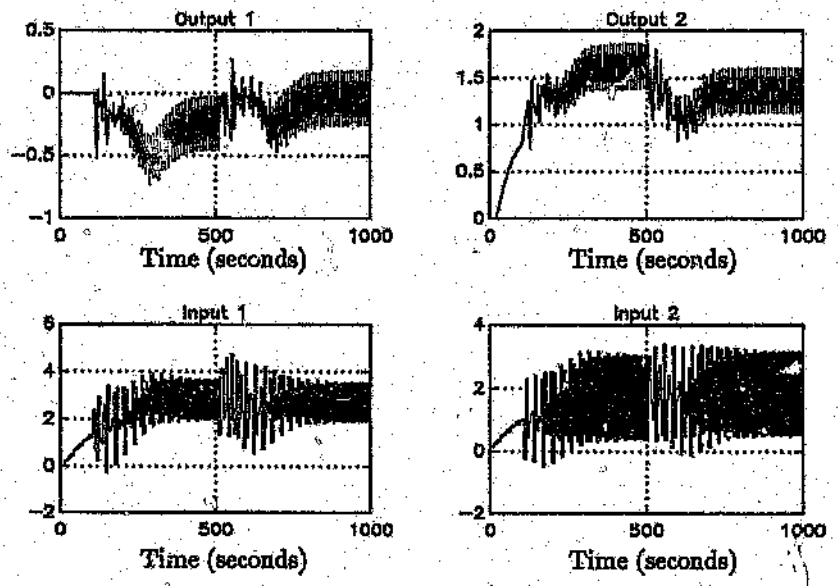


Figure B.15: Simulation results of test15

B.16 Test16

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)(0.1s+1)} & \frac{3.6e^{-8Ts}}{(14.9s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-Ts}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.31)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-Ts}}{(17s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.32)}$$

Pre-amble settings

HP	$= 15$	T	$= 5$ seconds
Ysp	$= [0\ 1]^T$	tau	$= [50\ 50]^T$ seconds
Umax	$= [5\ 5]^T$	Umin	$= [-1\ -1]^T$
no_const	$= 5$	no_steps	$= 200$

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.01 + 0.001x$
$\beta_2(x) = 0.01 + 0.001x$ x=1...HP

λ $\lambda_1(x) = 1$
$\lambda_2(x) = 1$ x=1...HP

Simulation results

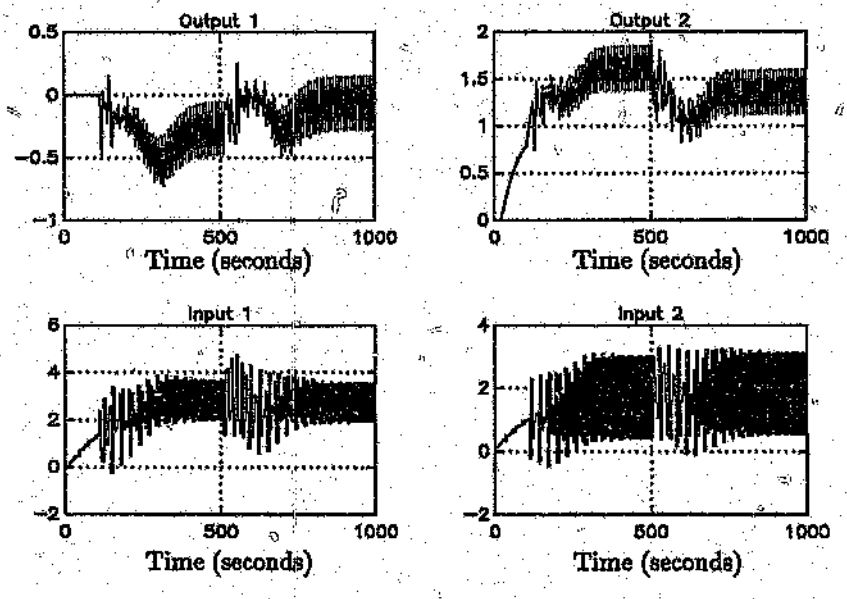


Figure B.16: Simulation results of test16

B.17 Test17

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-2s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)(0.1s+1)} & \frac{5.8e^{-8Ts}}{(14.9s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)(0.1s+1)} & \frac{4.9e^{-Ts}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.33)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-Ts}}{(17s+1)} & \frac{3.1e^{-3Ts}}{(21s+1)} \\ \frac{-0.52e^{-7Ts}}{(10.9s+1)} & \frac{1.52e^{-3Ts}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.34)}$$

Pre-amble settings

HP	$= 15$	T	$= 5$ seconds
Ysp	$= [0\ 1]^T$	tau	$= [50\ 50]^T$ seconds
Umax	$= [5\ 5]^T$	Umin	$= [-1\ -1]^T$
no_const	$= 5$	no_steps	$= 200$

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1 + 0.01x$
$\beta_2(x) = 0.1 + 0.01x$ x=1...HP

λ $\lambda_1(x) = 1$
$\lambda_2(x) = 1$ x=1...HP

Simulation results

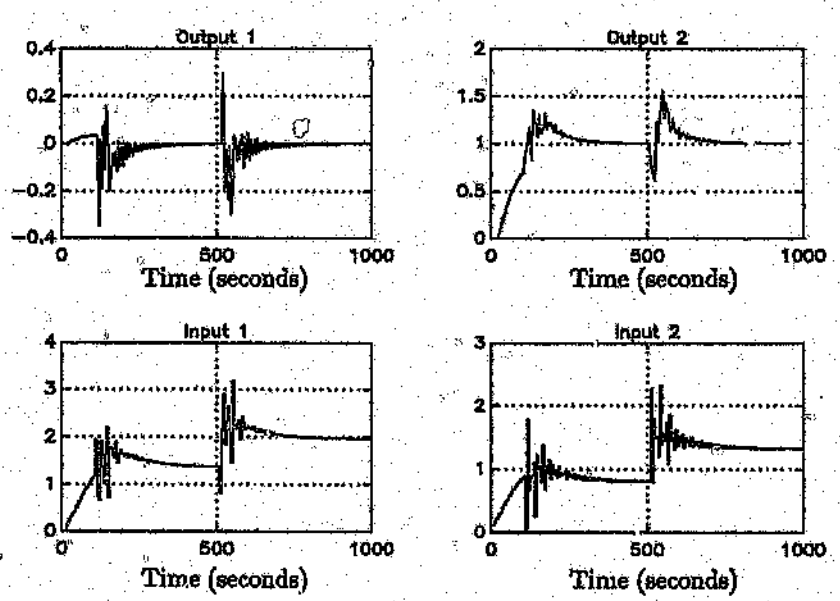


Figure B.17: Simulation results of test17

B.18 Test18

Plant Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.1e^{-7s}}{(16.7s+1)(0.1s+1)} & \frac{3.1e^{-3T_s}}{(21s+1)(0.1s+1)} & \frac{3.8e^{-8T_s}}{(14.9s+1)} \\ \frac{-0.52e^{-7T_s}}{(10.9s+1)(0.1s+1)} & \frac{1.52e^{-3T_s}}{(14.4s+1)(0.1s+1)} & \frac{4.0e^{-7s}}{(13.2s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ d(t) \end{bmatrix} \quad \text{(B.35)}$$

Predictor Model

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{-2.11e^{-7s}}{(17s+1)} & \frac{3.11e^{-3T_s}}{(21.1s+1)} \\ \frac{-0.521e^{-7T_s}}{(10.9s+1)} & \frac{1.525e^{-3T_s}}{(14.4s+1)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{(B.36)}$$

Pre-amble settings

HP	$= 15$	T	$= 5$ seconds
Ysp	$= [0\ 1]^T$	tau	$= [50\ 50]^T$ seconds
Umax	$= [2.2\ 2]^T$	Umin	$= [0\ 0]^T$
no_const	$= 5$	no_steps	$= 200$

d = zero from 1...20, one from 20...100, zero from 100...200

β $\beta_1(x) = 0.1 + 0.01x$, $\beta_2(x) = 0.1 + 0.01x$ x=1...HP

λ $\lambda_1(x) = 1$, $\lambda_2(x) = 1$ x=1...HP

Simulation results

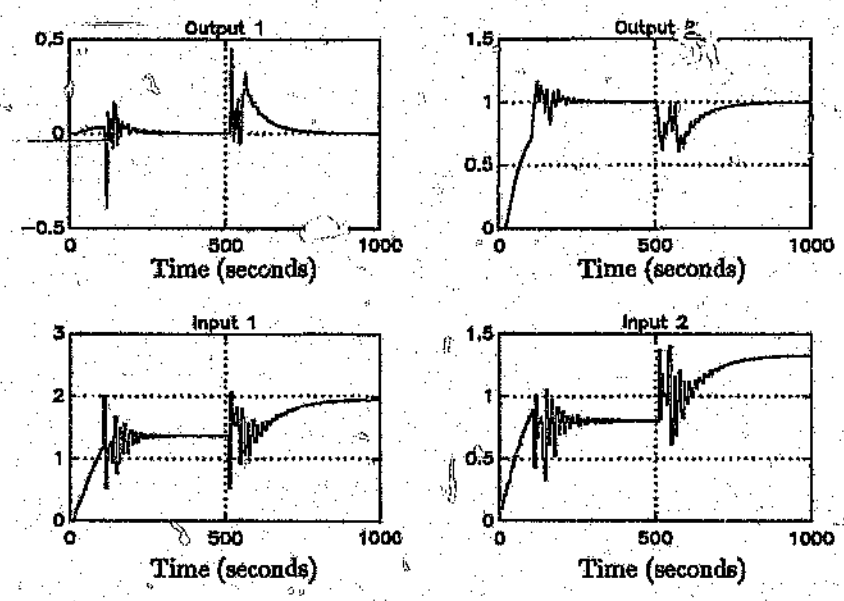


Figure B.18: Simulation results of test18

Appendix C

MATLAB programs

C.1 C2DP.M

```
function [B, A] = c2dp(num, den, T);

% function [B, A] = c2dp(num, den, T);
%
% converts a continuous time transfer function to discrete time
% assuming a zero order hold on the inputs
%                    -1                  -2          -3
%             B(q  )       0 + B(2)*q    + B(3)*q    + ....
%      P(q) = -------- = ------------------------------------
%                -1                 -2          -3
%             A(q  )       1 + A(2)*q    + A(3)*q    + ....

[a,b,c,d] = tf2ss(num, den);
[a,b]     = c2d(a,b,T);
[B, A]    = ss2tf(a,b,c,d,1);
end;
```

C.2 TREF.M

```
%% function [y] = TRef(Ysp, Y0, N, beta);
%%
%% Calculates the reference trajectory


function [y] = TRef(Ysp, Y0, N, beta);

y(1,1) = Y0;
for i = 1:N-1
   y(i+1,1) = beta*y(i,1) + (1-beta)*Ysp;
end;
```

C.3 CALC.M

```
%%% function [Q, W, R] = calc(B,A,HP);
%%%
%%% Calculates the Q, W and R matrix
%%%
%%% B  = [B1 B2 B3 B4 B5 B6 .... B(n+1)]
%%% A  = [A1 A2 A3 A4 A5 A6 .... A(n+1)]
%%% n  = order of system
%%% HP = horizon prediction
%%%
%%% predicted_output =   Q * [past_outputs]'
%%%                    + W * [predicted_inputs]'
%%%                    + R * [past_inputs]'
%%%
%%% where  :  predicted_outputs = [y(t+1)    y(t+2)    .... y(t+HP) ]'
%%%           predicted_inputs  = [x(t+HP-1) x(t+HP-2) .... x(t)    ]'
%%%           past_inputs       = [x(t-1)    x(t-2)    .... x(t-n+1)]'
%%%           past_outputs      = [y(t)      y(t-1)    .... y(t-n+1)]'

function [Q, W, R] = calc(B,A,HP);

n = length(A)-1;

clear Q W R

% set-up initial starting point
  Q = zeros(HP, n);
  W = zeros(HP, HP);
  R = zeros(HP, n-1);


Q(1,:) = -A(2:n+1);
W(1,:) = [zeros(1,HP-1) B(2)];
R(1,:) = B(3:n+1);

dummy1 = [W(1,:) R(1,:)];
dummy2 = Q(1,:);

dummy11 = dummy1; dummy22 = dummy2;

for i = 2:HP
    M = dummy22(1);
    dummy11 = [dummy11(2:length(dummy11)) 0] + M*dummy1;
    dummy22 = [dummy22(2:length(dummy22)) 0] + M*dummy2;

    W(i,:) = dummy11(1:HP);
    R(i,:) = dummy11(HP+1:length(Zi));
    Q(i,:) = dummy22;
end;
```

C.4 STEST.M

```
% This M-file will test the functionality of the predictor equation
% when considering a SISO system
%
% Procedure : A SISO system will be setup.
%             The output will be generated using both the predictor equation
%             and the simulation of the system.

clear;

HP = 80;                        % Horizon prediction
T  = 0.05;                      % sampling period (seconds)

% set-up plant model
num = 600*[1 5];
den = [1 10.6 106 1000];

[B, A] = c2dp(num, den, T);
n = length(A)-1;                % order of transfer function

[Q, W, R] = calc(B,A,HP);

% Predict the output
  %%% Using the z-domain model
    t = T*(1:1:HP);
    %% set-up the system with input = 0.5
     [y,X0] = lsim(num,den,0.5+zeros(HP,1),100*t);
     [A1,B1,C1,D1] = tf2ss(num,den);
    %% calculate the step response of the system with an initial condition
    %% of input = 0.5 and output = 1.50
    %% Note : The extra time was included because the lsim command always
    %%        includes the initial starting point in the output.
     y = lsim(A1,B1,C1,D1,ones(HP+1,1),[t T*(HP+1)],X0(HP,:));
     y = y(2:length(y));

  %%% Using the predictor equation
     y_past =  1.50 + zeros(n,1);
     u_past =  0.50 + zeros(n-1,1);
     u_future = ones(HP,1);
     Y = Q*y_past+W*u_future+R*u_past;

% Display results
     plot(T*(1:HP),Y,'*',t,y), grid,
     title('Step response (with initial conditions)'),
     ylabel('Output'), xlabel('Time (seconds)');
```

C.5 CALCM.M

```
%%% function [Q, W, R] = calcm(B,A,HP,l,m);
%%%
%%% Calculates the Q, W and R matrix for a MIMO system
%%%
%%%       | B11 |
%%%       | B12 |
%%%       |  .  |      where Bxy = row vector of coefficients of length n+1
%%%       |  .  |
%%%       |  .  |      HP = Horizon Prediction
%%%       | B1l |      l  = Number of inputs
%%%       | B21 |      m  = Number of outputs
%%%       | B22 |      n  = order of system
%%%       |  .  |
%%% B =   |  .  |
%%%       |  .  |
%%%       | B2l |
%%%       |  .  |
%%%       |  .  |
%%%       |  .  |
%%%       | Bm1 |
%%%       | Bm2 |
%%%       |  .  |
%%%       |  .  |
%%%       | Bml |


function [Q, W, R] = calcm(B,A,HP,l,m);

n = length(A)-1;                % n = order of transfer function

%% Set-Up Matrices
    Q = zeros(m*HP, m*n);
    W = zeros(m*HP, l*HP);
    R = zeros(m*HP, l*(n-1));

% set-up first row
  for i = 2:n+1
    Q(1:m , m*(i-2)+1:m*(i-1) ) = -A(i)*eye(m);
  end;
  W(1:m,:) = [zeros(m,l*(HP-1))'; get_b(B,2,l,m)']';
  for i = 3:n+1
    R(1:m , l*(i-3)+1:l*(i-2) ) = get_b(B,i,l,m);
  end;

% set-up Z10 and Z20;
  Z10 = [W(1:m,:) R(1:m,:)];
  Z20 = Q(1:m,:);

  Z1 = Z10; Z2 = Z20;

for i = 2:HP
  M  =  Z2(1:m , 1:m);
  Z1 = [Z1(1:m , 1+1:l*(HP+n-1)) zeros(m,l)] + mult(M,Z10,l);
  Z2 = [Z2(1:m,  m+1:m*n)        zeros(m,m)] + mult(M,Z20,m);
```

```
    W( m*(i-1)+1:m*i , :) = Z1(1:m , 1:1*HP);
    R( m*(i-1)+1:m*i , :) = Z1(1:m , 1*HP+1:1*(HP+n-1));
    Q( m*(i-1)+1:m*i , :) = Z2;
end;
```

C.6 MTEST.M

```
%% This MATLAB program will test the functionality of the MIMO predictor
%% equation program CALCH.M
%%
%% define s-domain model :
%%
%%          |   -2.1        3.1       |
%%          | ---------    ---------   |
%%          |  16.7S+1      21S+1      |
%%          |                          |
%% P(S) =   |                          |
%%          |  -0.52        1.52       |
%%          | ---------    ---------   |
%%          |  10.9S+1      14.4S+1    |
%%
%%          |                          |
%%          |    P11(s)     P12(s)     |
%%      =   |                          |
%%          |    P21(s)     P22(s)     |
%%          |                          |
%%          ----------------------------
%%                     den(s)
%%
%%
%% The discrete time model is as follows (ZOH on the input) :
%%
%%       | B11   B12 |
%%   P = |           |
%%       | B21   B22 |
%%      ---------------
%%             A

clear;

HP = 20;                    % prediction horizon
T  = 5;                     % sampling period
no_iterations = 40;

% Set-up model of MIMO system in the s-domain.
  l = 2;                    % number of inputs
  m = 2;                    % number of outputs
  num11     =  -2.1; den11 = [16.7 1];
  num12     =   3.1; den12 = [21.0 1];
  num21     =  -0.52; den21 = [10.9 1];
  num22     =   1.52; den22 = [14.4 1];

  den = conv( den11, den12);
  den = conv(   den, den21);
  den = conv(   den, den22);

% Convert the model to the notation as shown above
  P11 = deconv( conv( num11, den ), den11);
  P12 = deconv( conv( num12, den ), den12);
  P21 = deconv( conv( num21, den ), den21);
  P22 = deconv( conv( num22, den ), den22);
```

```
%%%% Find step response using s-domain model
  t = T*(0:1:HP);
  [A1,B1,C1,D1] = tf2ss([P11;P21],den);
  [A2,B2,C2,D2] = tf2ss([P12;P22],den);
  %%% set-up system with initial conditions
  %%% input1 = 1 and input2 = 2 (steady state conditions)
    % find step response wrt input 1
      [y1,X1] = lsim(A1,B1,C1,D1,1+zeros(HP+1,1),100*t);
    % find step response wrt input 2
      [y2,X2] = lsim(A2,B2,C2,D2,2+zeros(HP+1,1),100*t);
  %%% find response of the system with these initial conditions
  %%% input1 = 0.5 and input2 = 1
    % response due to input1
      y1 = lsim(A1,B1,C1,D1,0.5*ones(HP+1,1),t,X1(HP+1,:));
    % response due to input2
      y2 = lsim(A2,B2,C2,D2,ones(HP+1,1),t,X2(HP+1,:));
    y = y1+y2;

%%%% Find step response using the discrete time predictor equation
  % Convert the model from the s-plane to the discrete-time with ZOH @ input
      [B11,A] = c2dp(P11, den, T);
      [B12,A] = c2dp(P12, den, T);
      [B21,A] = c2dp(P21, den, T);
      [B22,A] = c2dp(P22, den, T);
      B = [B11; B12; B21; B22];
      n = length(A)-1;
  % Calculate the predictor equation matrices for the MIMO system
      [Q,W,R] = calcm(B,A,HP,1,m);
  % set-up system
      u_past(1:2:2*(n-1),1)  = 1.00 + zeros(n-1,1);   % past inputs for input 1
      u_past(2:2:2*(n-1),1)  = 2.00 + zeros(n-1,1);   % past inputs for input 2
      y_past(1:2:2*n,1)      = 4.10 + zeros(n,1);     % past outputs for output 1
      y_past(2:2:2*n,1)      = 2.52 + zeros(n,1);     % past outputs for output 2

  % find step response
    u_future(1:2:2*HP,1) = 0.50 + zeros(HP,1);
    u_future(2:2:2*HP,1) = 1.00 + zeros(HP,1);
    Y = Q*y_past + W*u_future + R*u_past;

plot( T*(1:HP),Y(1:2:length(Y)), t,y(:,1),'o' );
title('Response of MIMO system');
ylabel('Output 1'), xlabel('Time (seconds)');
grid, pause;

plot( T*(1:HP),Y(2:2:length(Y)), t,y(:,2),'o' );
title('Response of MIMO system');
ylabel('Output 2'), xlabel('Time (seconds)');
grid;
```

C.7 MODEL.M

```
%% This m-file demonstrates how the reference model of a process can be
%% defined using a difference equation
%
% Process Model in s-domain
%
%              10(s+1)
%  P(s) = ------------------
%          (s+3)(s^2+s+1)
%
% This program will find the coefficient needed to define the A and B vectors

clear, clc, clg;

% set-up plant model
num = 10*[1 1];
den = [1 4 4 3];
T   = 1;                          % sampling period (seconds);

[B, A] = c2dp(num, den, T);

disp('The B vector is : '),B
disp('The A vector is : '),A

y1 = dstep(B,A,20);
x = T*linspace(0,19,20)+T/2; x = x';
stairs(x,y1);
hold on;
step(num,den,T*linspace(0,20));
hold off

title('Step response of process')
```

C.8 QP.M

```
function [X,lambda,how]=qp(H,f,A,B,vlb,vub,X,neqcstr,verbosity,negdef)
% QP    Quadratic programming.
% X=QP(H,f,A,b) solves the quadratic programming problem:
%
%            min 0.5*x'Hx + f'x   subject to:  Ax <= b
%             x
%
%       [x,LAMBDA]=QP(H,f,A,b) returns the set of Lagrangian multipliers,
%       LAMBDA, at the solution.
%
%       X=QP(H,f,A,b,VLB,VUB) defines a set of lower and upper
%       bounds on the design variables, X, so that the solution
%       is always in the range VLB < X < VUB.
%
%       X=QP(H,f,A,b,VLB,VUB,N) indicates that the first N constraints defined
%       by A and b are equality constraints.
%
%       QP produces warning messages when the solution is either unbounded
%       or infeasible.  Warning messages can be turned off with the calling
%       syntax: X=QP(H,f,A,b,VLB

% Copyright (c) 1990 by the Math[illegible] Inc.
% Andy Grace 7-9-90.


% Handle missing arguments
if nargin <10, negdef = 0;
if nargin<9, verbosity = [];
if nargin<8, neqcstr=[];
if nargin < 7, X=[];
if nargin<6, vub=[];
if nargin<5, vlb=[];
end, end, end, end, end, end
[ncstr,nvars]=size(A);
nvars = length(f); % In case A is empty
if ~length(verbosity), verbosity = 0; end
if ~length(neqcstr), neqcstr = 0; end
if ~length(X), X=zeros(nvars,1); end

f=f(:);
B=B(:);

if  norm(H,'inf')==0 | ~length(H), H=0; QP=0; else, QP=~negdef; end
how = 'ok';
% Handle constraints as linear constraints
lenvlb=length(vlb);
if lenvlb > 0
A=[A;-eye(lenvlb,nvars)];
B=[B;-vlb(:)];
end
lenvub=length(vub);
if lenvub>0
A=[A;eye(lenvub,nvars)];
B=[B;vub(:)];
end
```

```
ncstr=ncstr+lenvlb+lenvub;
lambda=zeros(ncstr,1);
aix=lambda;
ACTCNT=0;
ACTSET=[];
ACTIND=0;
CIND=1;
eqix = 1:neqcstr;
%----------EQUALITY CONSTRAINTS----------
if neqcstr>0
aix(eqix)=ones(neqcstr,1);
ACTSET=A(eqix,:);
ACTIND=eqix;
ACTCNT=neqcstr;
CIND=neqcstr+1;
if max(abs(A(eqix,:)*X-B(eqix)))>1e-10
X=ACTSET\B(eqix);
end
Z=null(ACTSET);
err = 0;
if neqcstr > nvars
err = max(abs(A(eqix,:)*X-B(eqix)));
if (err > 1e-8)
how='infeasible';
if verbosity > -1
disp('Warning: The equality constraints are overly stringent;')
disp('         there is no feasible solution.')
end
end
end
if ~length(Z) | err > 1e-8
actlambda=abs(-ACTSET'\(H*X+f));
lambda(eqix)=actlambda;
return
end
% Check whether in Phase 1 of feasibility point finding.
if (verbosity == -2)
cstr = A*X-B;
    mc=max(cstr(neqcstr+1:ncstr));
if (mc > 0)
X(nvars) = mc + 1;
end
end
else
Z=1;
end

% Find Initial Feasible Solution
   cstr=A*X-B;
   mc=max(cstr(neqcstr+1:ncstr));
   if mc>eps
A2=[[A;zeros(1,nvars)],[zeros(neqcstr,1);-on (ncstr+1-neqcstr,1)]];
[XS,lambdas]=qp([],[zeros(nvars,1);1],A2,[B;1e-5],[],[],[X;mc+1],neqcstr,-2);
X=XS(1:nvars);
cstr=A*X-B;
        if XS(nvars+1)>eps
if XS(nvars+1)>1e-8
```

```
how='infeasible';
if verbosity > -1
disp('Warning: The constraints are overly stringent;')
disp('         there is no feasible solution')
end
else
how = 'overly constrained';
end
lambda=lambdas(1:ncstr);
  return
end
    end
gf=H*X+f;
if (QP)
   SD=-Z*((Z'*H*Z)\(Z'*gf));
% Check for -ve definite problems:
%  if SD'*gf>0, QP = 0; SD=-SD; end
  else
   SD=-Z*Z'*gf;
end
oldind = 0;
errcstr = norm(A)*1e-6;

%--------------Main Routine-----------------
 while ACTCNT<min(ncstr,nvars)+1
% Find distance we can move in search direction SD before a
% constraint is violated.
GSD=A*SD;

indf = find(GSD>0 & ~aix);
if ~length(indf)
STEPMIN=1e16;
else
[STEPMIN,ind2] =  min(abs(cstr(indf)./GSD(indf)));
ind=indf(ind2);
end
%------------------QP-------------
if (QP)
% If STEPMIN is 1 then this is the exact distance to the solution.
if STEPMIN>=1
X=X+SD;
  if ACTCNT>0
if ACTCNT>=nvars-1, ACTSET(CIND,:)=[];ACTIND(CIND)=[]; end
actlambda=-ACTSET'\(H*X+f);
[minl, lind] = min(actlambda(1+neqcstr:ACTCNT));
if (~length(minl)) minl = 1; end
if (minl>0)
lambda(ACTIND)=abs(actlambda);
return
end
% Remove constraint(s)
ACTSET(lind,:) = [];
aix(ACTIND(lind)) = 0;
ACTIND(lind) = [];
ACTCNT = ACTCNT-2;
ind = 0;
else
```

```
return
end
else
      X=X+STEPMIN*SD;
end
else
% Unbounded Solution
if STEPMIN>1e15;
if X(nvars)>eps
STEPMIN=abs((X(nvars)+1e-5)/(SD(nvars)+eps));
X=X+STEPMIN*SD;
end
how='unbounded';
if verbosity > -1
disp('Warning: The solution is unbounded and at infinity;')
disp('         the constraints are not restrictive enough.')
end
return
else
      X=X+STEPMIN*SD;
end
end %if (qp)

% Update X and calculate constraints
cstr=A*X-B;
% Check no constraint is violated
if max(cstr)>errcstr
if verbosity > -1
disp('Warning: The problem is badly conditioned;')
disp('         the solution is not reliable')
end
how='failed';
X=X-STEPMIN*SD;
return
end

% Calculate new search direction and update active set
gf=H*X+f;
if ind
aix(ind)=1;
ACTSET(CIND,:)=A(ind,:);
ACTIND(CIND)=ind;
end
if oldind
aix(oldind) = 0;
end
if ACTCNT<nvars-1
Z=null(ACTSET);
ACTCNT=ACTCNT+1;
CIND=ACTCNT+1;
oldind = 0;
else
actlambda=-ACTSET'\gf;
actlambda(eqix)=abs(actlambda(eqix));
[minl,CIND]=min(actlambda);
if minl<0
```

```
Z=null([ACTSET(1:CIND-1,:);ACTSET(CIND+1:nvars,:)]);
oldind = ACTIND(CIND);
else
lambda(ACTIND)=actlambda;
return
end
end %if ACTCNT<nvars
if (QP)
Zgf = Z'*gf;
if (norm(Zgf) < 1e-15)
SD = zeros(nvars,1);
elseif ~length(Zgf)
% Only happens in -ve semi-definite problems
disp('Warning: QP problem is -ve semi-definite.')
SD = zeros(nvars,1);
else
SD=-Z*((Z'*H*Z)\(Zgf));
end
% Check for -ve definite problems
    % if SD'*gf>0, QP = 0; SD=-SD; end
else
SD=-Z*Z'*gf;
if norm(SD)<1e-10
actlambda = abs(-ACTSET'\gf);
lambda(ACTIND)=actlambda;
return
end
end
end % while ACTCNT<min(nvars,ncstr)
```

C.9 SIMULATE.M

```
%%%% Multi-variable plant model simulation
%
%  Specifications :
%       Has got integral action
%       Constraints on inputs
%       Allowances for different predictor and plant model
%
% define continuous time model
%
%         | P11   P12 |
% P(S) =  |           |
%         | P21   P22 |
%       -----------------
%            den(s)

clear;

% setup specifications
  HP   = 20;                % prediction horizon
  T    = 5;                 % sampling period
  Ysp  = [0;1];             % set-point
  tau  = [60;60];           % time constant of ref trajectory
  Umax = [2.2;2];           % input constraints
  Umin = [0;0];             % input constraints
  no_const = 5;             % number of inputs constraints
  no_steps = 200;           % number of steps to run simulation for

%%% define disturbance input
  d = [zeros(20,1); 0.1+zeros(80,1); zeros(no_steps,1)];

%%% Define model of PREDICTOR
   % define dimensions of predictor model
     l_pred = 2;                    % number of inputs
     m_pred = 2;                    % number of outputs
   % Set-up model of MIMO system in the s-domain.
     num11 =  -2.15;  den11 = [17   1]; delay11 = 0;
     num12 =   3.2;   den12 = [21.1 1]; delay12 = 0;
     num21 =  -0.6;   den21 = [11   1]; delay21 = 0;
     num22 =   1.52;  den22 = [15   1]; delay22 = 0;
     max_delay_pred = max( [delay11 delay12 delay21 delay22] );
     den = conv( den11, den12);
     den = conv(   den, den21);
     den = conv(   den, den22);
   % Convert the model to the notation as shown above
     P11 = deconv( conv( num11, den ), den11);
     P12 = deconv( conv( num12, den ), den12);
     P21 = deconv( conv( num21, den ), den21);
     P22 = deconv( conv( num22, den ), den22);
   % Convert the model from the s-plane to the discrete-time with ZOH on input
     [B11,A] = c2dp(P11, den, T);
     [B12,A] = c2dp(P12, den, T);
     [B21,A] = c2dp(P21, den, T);
     [B22,A] = c2dp(P22, den, T);
     B11 = [ zeros(1,delay11) B11 zeros(1,max_delay_pred-delay11) ];
     B12 = [ zeros(1,delay12) B12 zeros(1,max_delay_pred-delay12) ];
```

```
    B21 = [ zeros(1,delay21) B21 zeros(1,max_delay_pred-delay21) ];
    B22 = [ zeros(1,delay22) B22 zeros(1,max_delay_pred-delay22) ];
    A_pred   = [ A zeros(1,max_delay_pred)];
    B_pred   = [B11; B12; B21; B22];
  % order of transder function
    n_pred = length(A_pred)-1;
  % set-up system (predictor model)
    u_past_pred(1:l_pred:l_pred*(n_pred-1) ,1) = 0+zeros(n_pred-1,1);
    u_past_pred(2:l_pred:l_pred*(n_pred-1) ,1) = 0+zeros(n_pred-1,1);
    y_past_pred(1:m_pred:m_pred*n_pred ,1)     = 0+zeros(n_pred,1);
    y_past_pred(2:m_pred:m_pred*n_pred ,1)     = 0+zeros(n_pred,1);
    present_inputs = [0;0];


%%% Define model of PLANT
  % define dimensions of plant model
    l_plant = 3;                    % number of inputs
    m_plant = 2;                    % number of outputs
  % Set-up model of MIMO system in the s-domain.
    num11 =  -2.1;  den11 = conv([16.7 1],[5 1]);  delay11 = 0;
    num12 =   3.1;  den12 = conv([21.0 1],[5 1]);  delay12 = 0;
    num13 =   3.8;  den13 = [14.9 1];              delay13 = 0;
    num21 =  -0.52; den21 = conv([10.9 1],[5 1]);  delay21 = 0;
    num22 =  1.52;  den22 = conv([14.4 1],[5 1]);  delay22 = 0;
    num23 =    4.9; den23 = [13.2 1];              delay23 = 0;
    max_delay = max( [delay11 delay12 delay13 delay21 delay22 delay23] );
    den = conv( den11, den12);
    den = conv(   den, den13);
    den = conv(   den, den21);
    den = conv(   den, den22);
    den = conv(   den, den23);

  % Convert the model to the notation as shown above
    P11 = deconv( conv( num11, den ), den11);
    P12 = deconv( conv( num12, den ), den12);
    P13 = deconv( conv( num13, den ), den13);
    P21 = deconv( conv( num21, den ), den21);
    P22 = deconv( conv( num22, den ), den22);
    P23 = deconv( conv( num23, den ), den23);
  % Convert the model from the s-plane to the discrete-time with ZOH on input
    [B11,A] = c2dp(P11, den, T);
    [B12,A] = c2dp(P12, den, T);
    [B13,A] = c2dp(P13, den, T);
    [B21,A] = c2dp(P21, den, T);
    [B22,A] = c2dp(P22, den, T);
    [B23,A] = c2dp(P23, den, T);
    B11 = [ zeros(1,delay11) B11 zeros(1,max_delay-delay11) ];
    B12 = [ zeros(1,delay12) B12 zeros(1,max_delay-delay12) ];
    B13 = [ zeros(1,delay13) B13 zeros(1,max_delay-delay13) ];
    B21 = [ zeros(1,delay21) B21 zeros(1,max_delay-delay21) ];
    B22 = [ zeros(1,delay22) B22 zeros(1,max_delay-delay22) ];
    B23 = [ zeros(1,delay23) B23 zeros(1,max_delay-delay23) ];
    A_plant   = [ A zeros(1,max_delay)];
    B_plant   = [B11; B12; B13; B21; B22; B23];
  % order of transfer function
    n_plant = length(A_plant)-1;
  % set-up system (plant model)
```

```
    y_past_plant(1:m_plant:m_plant*n_plant,1) = 0+zeros(n_plant,1);
    y_past_plant(2:m_plant:m_plant*n_plant,1) = 0+zeros(n_plant,1);
    u_past_plant(1:l_plant:l_plant*n_plant,1) = 0+zeros(n_plant,1);
    u_past_plant(2:l_plant:l_plant*n_plant,1) = 0+zeros(n_plant,1);
  % get suffled A and B matrices
    AA_plant = []; BB_plant = [];
    for i = 1:length(A_plant)
       BB_plant(1:m_plant, l_plant*(i-1)+1:l_plant*i) = get_b(B_plant,i,l_plant,m_plant);
       AA_plant(1:m_plant, m_plant*(i-1)+1:m_plant*i) = A_plant(i)*eye(m_plant);
    end;

output_history = []; input_history = [];

% Calculate the predictor equation matrices for the MIMO system
  [Q,W,R] = calcm(B_pred, A_pred, HP, l_pred, m_pred);

% set-up weighting matrices
  beta1 = []; alpha1 = [];
  for i = 1:HP
    beta1 = [beta1; 0.5+0.1*i];
    if i<HP/4
      alpha1 = [alpha1; 1];
    elseif i>3/4*HP
      alpha1 = [alpha1; 1.5];
    else
      alpha1 = [alpha1; 1.5];
    end;
  end;
  for i = 1:m_pred
    alpha(i:m_pred:m_pred*HP) = alpha1;
  end;
  for i = 1:l_pred
     beta(i:l_pred:l_pred*HP) = beta1;
  end;
  beta  = diag(beta); alpha = diag(alpha);


delta_u = zeros(l_pred*HP,1); X0=delta_u;

b = exp(-T./tau);
C = 2*( (alpha*W)'*(alpha*W) + beta'*beta );

%%% constraint
aa = zeros(l_pred*no_const,l_pred*HP);
for i = 1:no_const
  aa(i*l_pred-1, 1:l_pred:l_pred*HP) = [zeros(1,HP-i) ones(1,i)];
  aa(i*l_pred,   2:l_pred:l_pred*HP) = [zeros(1,HP-i) ones(1,i)];
end;
aa = [aa;-aa];
for i = 1:l_pred
  bb1(i:l_pred:l_pred*no_const,1) = Umax(i)+zeros(no_const,1);
  bb2(i:l_pred:l_pred*no_const,1) = Umin(i)+zeros(no_const,1);
end;
T_lag = [];

pack;
```

```
for counter = 1:no_steps

%%%% Predictive Control Section

%%% 1.  Measure the output
  % update the history inputs to the plant
    u1           = present_inputs(1);
    u2           = present_inputs(2);
    dummy        = [ [u1;u2;d(counter,1)]; u_past_plant];
    u_past_plant = dummy(1:l_plant*n_plant,1);
  % calclate output of plant
    y_out_plant  = BB_plant*[zeros(l_plant,1);u_past_plant]
                         - AA_plant*[zeros(m_plant,1);y_past_plant];

  % calculate the change in output and update the predictor model outputs
    dummy        = [ (y_out_plant-y_past_plant(1:m_plant,:)) ; y_past_pred];
    y_past_pred = dummy(1:n_pred*m_pred,1);

  % update history of outputs of plant
    dummy        = [y_out_plant; y_past_plant];
    y_past_plant = dummy(1:m_plant*n_plant,1);


%%% 2.  Set-Up Reference Trajectory
  X = [];
  for j = 1:m_pred
    dummy = [y_out_plant(j); tref(Ysp(j), y_out_plant(j,1), HP, b(j))];
    for i = 1:HP
       X(i,j) = dummy(i+1) - dummy(i);
    end;
  end;
  Y_Ref = [];
  for i = 1:m_pred
     Y_Ref(i:m_pred:m_pred*HP,1) = X(:,i);
  end;

%%% 3. Find optimum input
  Y1 = Q*y_past_pred + R*u_past_pred;

  p = -2*(alpha*W)'*alpha*(Y_Ref-Y1);

  for i = 1:l_pred
     bb(i:l_pred:l_pred*no_const,1) =
                    bb1(i:l_pred:l_pred*no_const,1) - present_inputs(i);
     bb(l_pred*no_const+i:l_pred:2*l_pred*no_const,1) =
                  -( bb2(i:l_pred:l_pred*no_const,1) - present_inputs(i) );
  end;

  t0 = clock;
  delta_u = qp(C,p,aa,bb,[],[],X0);
  T_lag = [T_lag; etime(clock, t0)];


%%% 4. Update the history vectors

  present_inputs = present_inputs+delta_u(1+l_pred*(HP-1):l_pred*HP,1);
  input_history  = [input_history; present_inputs'];
```

```
  u_past_pred = [delta_u(1+l_pred*(HP-1):l_pred*HP,1);
                 u_past_pred(1:l_pred*(n_pred-2),1) ];

  output_history = [output_history; y_out_plant'];

  % calculate trajectory of inputs
  input_traj = present_inputs';
  for i = 2:HP
    dummy = delta_u(1+l_pred*(HP-i):l_pred*(HP-i+1),1)
                              + input_traj(length(input_traj(:,1)),:);
    input_traj = [input_traj; dummy'];
  end;

  % show predicted results
    y_pred = Yi + W*delta_u;
    clg; subplot(221);
    [xx,yy] = steps(1:HP, y_pred(1:m_pred:m_pred*HP));
    plot(1:HP, Y_Ref(1:m_pred:m_pred*HP), xx,yy ); ylabel('Rate out #1');
    title( sprintf('Iteration # %g',counter) );
    [xx,yy] = steps(1:HP, y_pred(2:m_pred:m_pred*HP));
    plot(1:HP, Y_Ref(2:m_pred:m_pred*HP), xx,yy ); ylabel('Rate out #2');
    [xx,yy] = steps(1:HP, input_traj(:,1));
    plot(xx,yy), ylabel('input #1');
    [xx,yy] = steps(1:HP, input_traj(:,2));
    plot(xx,yy), ylabel('input #2');

  % new initial start for optimization routine
    X0 = [ delta_u(1:l_pred, 1); delta_u( 1:l_pred*(HP-1), 1)];
end;

%%% show results
clg; subplot(221);
plot(T*(1:no_steps), output_history(:,1)), grid, title('Output 1');
plot(T*(1:no_steps), output_history(:,2)), grid, title('Output 2');
[xx,yy] = steps(T*(1:no_steps), input_history(:,1));
plot(xx,yy), grid, title('Input 1');
[xx,yy] = steps(T*(1:no_steps), input_history(:,2));
plot(xx,yy), grid, title('Input 2');
```

References

[1] Arulalan, G.R. and Deshpande, P.B., "New algorithm for multivariable control," tech. rep., Hydrocarbon Processing, June 1986.

[2] Åström, K.J. and Wittenmark, B., *Computer Controlled Systems.* Prentice-Hall, 1984.

[3] Clarke, D.W., Mohtadi, C., and Tuffs, P.S., "Generalized Predictive Control - Part I. The Basic Algorithm," tech. rep., Automatica Vol 23. No 2 pp 137-148, 1987.

[4] Clarke, D.W. and Zhang, L., "Long-range predictive control using weighting-sequence models," tech. rep., Proceedings of the IEE (Vol 134, Pt. D, No 3), May 1987.

[5] Garcia, C.E. and Morshedi, A.M., "Quadratic Programming Solution of Dynamic Matrix Control (QDMC)," tech. rep., Shell Development Company, Houston, Texas, 1986.

[6] Martin, G.D., "Long-Range Predictive Control," tech. rep., AIChE Journal (Vol. 27, No. 5), Sept 1981.

[7] Richalet, J., Rault, A., Testud, J.L., and Papon, J., "Model Predictive Heuristic Control : Applications to Industrial Processes," tech. rep., Automatica, Vol 14, pp 413-428, 1978.

[8] Ricker, N.L., "Use of Quadratic Programming for Constrained Internal Model Control," tech. rep., American Chemical Society, 1985.

[9] Rouhani, R. and Mehra, R.K., "Model Algorithmic Control (MAC); Basic Theoretical Properties," tech. rep., Automatica (Vol. 18, No. 4, pp 401-414), 1982.

[10] Vinante, C.D., Bermudez, C., and Tarre, F., "Predictive Compensation Implemented with a Microprocessor," tech. rep., IEEE Control Systems Magazine, 1986.