MODEL PREDICTIVE CONTROL

A THESIS SUBMITTED IN PARTIAL FUFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND INSTRUMENTATION ENGINERING

BY

DEBADATTA PATRA

DEBASISH JENA

SUNIL KUMAR MOHANTY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA

2007

MODEL PREDICTIVE CONTROL

A THESIS SUBMITTED IN PARTIAL FUFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND INSTRUMENTATION ENGINERING

BY

DEBADATTA PATRA

DEBASISH JENA

SUNIL KUMAR MOHANTY

Under the Guidance of

Prof. TARUN KUMAR DAN

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA

2007

National Institute of technology
ROURKELA

CERTIFICATE

This is to certify that the thesis entitled , " MODEL PREDICTIVE CONTROL" , submitted by Mr. DEBADATTA PATRA in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in ' ELECTRONICS AND INSTRUMENTATION ' Engineering at the national institute of Technology , Rourkela (Deemed University) is an authentic work carried out by him under my supervision.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other university / institute for the award of any Degree or Diploma

Date

Prof. Tarun Kumar Dan
Dept. of electronics and communication Engg.
National Institute of Technology
Rourkela-769008

ACKNOWLEDGEMENT

Under the esteem guidance of our project guide professor TARUN KUMAR DAN a detail study of MODEL PREDICTIVE CONTROL and its applications to various models has been studied as well as simulation has been done. We are very thankful for his whole-hearted co-operation without which this project could not have been completed.

ABSTRACT

This project thesis provides a brief overview of Model Predictive Control (MPC).A brief history of industrial model predictive control technology has been presented first followed by a some concepts like the receding horizon, moves etc. which form the basis of the MPC. It follows the Optimization problem which ultimately leads to the description of the Dynamic Matrix Control (DMC).The MPC presented in this report is based on DMC. After this the application summary and the limitations of the existing technology has been discussed and the next generation MPC, with an emphasis on potential business and research opportunities has been reviewed. Finally in the last part we generate Matlab code to implement basic model predictive controller and introduce noise into the model. We have also taken up some case studies like Swimming pool water temperature control and helicopter flight control etc. by applying the MPC controller on these models.

Originally developed to meet the specialized control needs of power plants and petroleum refineries, MPC technology can now be found in a wide variety of application areas including chemicals, food processing, automotive, and aerospace applications Its reason for success is many, like it handles multivariable control problems naturally. But the most important reason for its success is its ability to handle constraints. Model predictive control (MPC) refers to a class of computer control algorithms that utilize an explicit process model to predict the future response of a plant. At each control interval an MPC algorithm attempts to optimize future plant behavior by computing a sequence of future manipulated variable adjustments. The first input in the optimal sequence is then sent into the plant, and the entire calculation is repeated at subsequent control intervals. The basic MPC controller can be designed with proper restrictions on the prediction horizon and model length. The prediction horizon has to be kept sufficiently larger than control horizon. But after applying to many other applications we find as the complexity increases then we need techniques other than DMC like generalized predictive control (GPC) which are better.

List of FIGURES

CONTENTS

CHAPTER 1

A BRIEF HISTORY OF INDUSTRIAL MODEL PREDICTIVE CONTROL

A Brief History of Industrial MPC

This section presents an abbreviated history of industrial MPC technology. Control algorithms are emphasized here because relatively little published information is available on the identification technology.

The development of modern control concepts can be traced to the work of Kalman in the early 1960's, who sought to determine when a linear control system can be said to be optimal [,]. Kalman studied a Linear Quadratic Regulator (LQR) designed to minimize an quadratic objective function. The process to be controlled can be described by a discrete-time, deterministic linear state-space model:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\,\mathbf{x}_k + \mathbf{B}\,\mathbf{u}_k \\ \mathbf{y}_k &= \mathbf{C}\,\mathbf{x}_k \end{aligned}$$

The vector represents process inputs, or manipulated variables; vector describes process output measurements. The vector represents process states. Figure 1 provides a schematic representation of a state space model. The state vector is defined such that knowing its value at time *k* and future inputs allows one to predict how the plant will evolve for all future time. Much of the power of Kalman's work relies on the fact that this general process model was used.

The objective function to be minimized penalizes squared input and state deviations from the origin and includes separate state and input weight matrices and to allow for tuning trade-offs:

$$J = \sum_{j=1}^{\infty} \left(\|\mathbf{x}_{k+j}\|_{\mathbf{Q}}^2 + \|\mathbf{u}_{k+j}\|_{\mathbf{R}}^2 \right)$$

where the norm terms in the objective function are defined as follows:

$$\|\mathbf{x}\|_{\mathbf{Q}}^2 = \mathbf{x}^T \mathbf{Q} \mathbf{x}$$

Implicit in the representation is the assumption that all variables are written in terms of deviations from a desired steady-state. The solution to the LQR problem was shown to be a proportional controller, with a gain matrix computed from the solution of a matrix Ricatti equation:

$$\mathbf{u}_k = -\mathbf{K}\,\mathbf{x}_k$$

The infinite prediction horizon of the LQR algorithm endowed the algorithm with powerful stabilizing properties; it was shown to be stabilizing for any reasonable linear plant (stablizable and detectable) as long as the objective function weight matrices **Q** and **R** are positive definite. A dual theory was developed to estimate plant states from noisy input and output measurements, using what is now known as a *Kalman Filter*. The combined LQR controller and Kalman filter is called a Linear Quadratic Gaussian (LQG) controller. Constraints on the process inputs, states and outputs were not considered in the development of LQG theory.

Although LQG theory provides an elegant and powerful solution to the problem of controlling an unconstrained linear plant, it had little impact on control technology development in the process industries. The most significant of the reasons cited for this failure include [,] :

- constraints
- process nonlinearities
- model uncertainty (robustness)
- unique performance criteria
- Cultural reasons (people, education, etc.)

It is well known that the economic operating point of a typical process unit often lies at the intersection of constraints []. A successful industrial controller must therefore maintain the system as close as possible to constraints without violating them. In addition, process units are typically complex, nonlinear, constrained multivariable systems whose dynamic behavior changes with time due to such effects as changes in operating conditions and catalyst aging. Process units are also quite individual so that development of process models from fundamental physics and chemistry is difficult to

justify economically. Indeed the application areas where LQG theory had a more immediate impact, such as the aerospace industry, are characterized by physical systems for which it is technically and economically feasible to develop accurate fundamental models. Process units may also have unique performance criteria that are difficult to express in the LQG framework, requiring time dependent output weights or additional logic to delineate different operating modes. However the most significant reasons that LQG theory failed to have a strong impact may have been related to the culture of the industrial process control community at the time, in which instrument technicians and control engineers either had no exposure to LQG concepts or regarded them as impractical.

This environment led to the development, *in industry*, of a more general model based control methodology in which the dynamic optimization problem is solved on-line at each control execution. Process inputs are computed so as to optimize future plant behavior over a time interval known as the *prediction horizon*. In the general case any desired objective function can be used. Plant dynamics are described by an explicit process *model* which can take, in principle, any required mathematical form. Process input and output constraints are included directly in the problem formulation so that future constraint violations are anticipated and prevented. The first input of the optimal input sequence is injected into the plant and the problem is solved again at the next time interval using updated process measurements. In addition to developing more flexible control technology, new process identification technology was developed to allow quick estimation of empirical dynamic models from test data, substantially reducing the cost of model development. This new methodology for industrial process modeling and control is what we now refer to as Model Predictive Control (MPC) technology.

In modern processing plants the MPC controller is part of a multi-level hierarchy of control functions. It is often difficult to translate the control requirements at this level into an appropriate conventional control structure. In the MPC methodology this combination of blocks is replaced by a single MPC controller.

Although the development and application of MPC technology was driven by industry, it should be noted that the idea of controlling a system by solving a sequence of open-loop dynamic optimization problems was not new. Propoi, for example, described a moving horizon controller in 1963 []. Lee and Markus [] anticipated current MPC practice in their 1967 optimal control text:

One technique for obtaining a feedback controller synthesis from knowledge of open-loop controllers is to measure the current control process state and then compute very rapidly for the open-loop control function. The first portion of this function is then used during a short time interval, after which a new measurement of the function is computed for this new measurement. The procedure is then repeated.

There is, however, a wide gap between theory and practice. The essential contribution of industry was to put these ideas into practice on operating units. Out of this experience came a fresh set of problems that has kept theoreticians busy ever since.

CHAPTER 2

THE RECEDING HORIZON

The 'receding horizon' idea

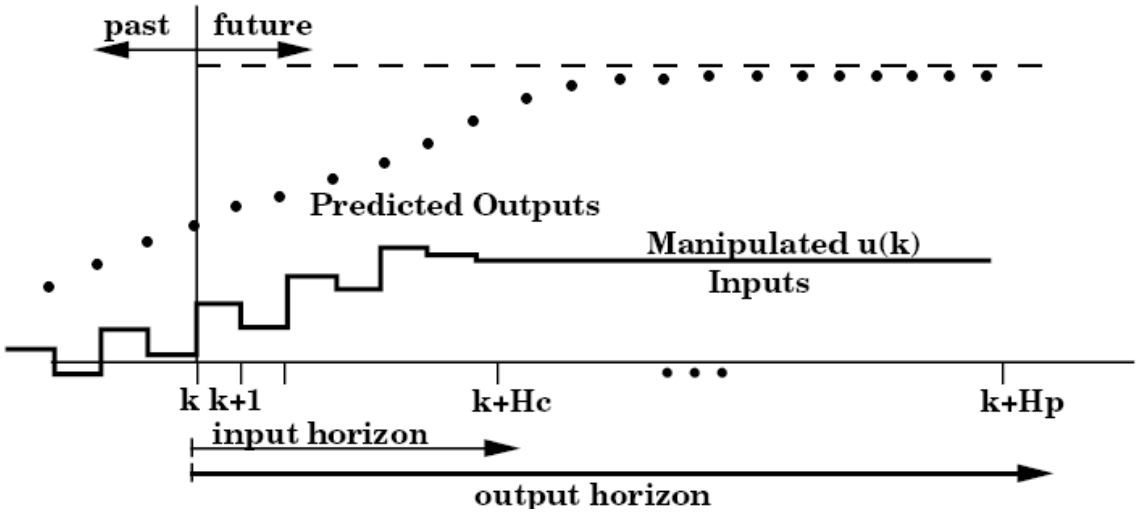

Fig2.1 The receding horizon concept showing Optimization Problem

The figure shows the basic idea of predictive control. In this presentation of the basics, we confine ourselves to discussing the control of a single-input, single-output (SISO) plant. We assume a discrete-time setting, and that the current time is labeled as time step k.at the current time the plant output is y(k), and that the figure shows the previous history of the output trajectory. Also shown is a set point trajectory, which is the trajectory that the output should follow, ideally. The value of the set-point trajectory at any time t is denoted by s(t).

Distinct from the set-point trajectory is the *reference trajectory* .This starts at the current output y(k), and defines an ideal trajectory along which the plant should return to the set-point trajectory, for instance after a disturbance occurs. The reference trajectory therefore defines an important aspect of the closed-loop behavior of the controlled plant. It is not necessary to insist that the plant should be driven back to the set-point trajectory as fast as possible, although that choice remains open. It is frequently assumed that the reference trajectory as fast as possible, although that choice remains open. It is frequently assumed that the reference trajectory approaches the set point exponentially, which we shall denote *Tref* , defining the speed of response. That is the current error is

$$Є\ (k) = s\ (k) - y(k)$$

Then the reference trajectory is chosen such that the error i steps later , if the output followed it exactly, would be

$$Є(k+i) = \exp(-iTs/Tref) * Є(k)$$

$$= \lambda i * Є(k)$$

where Ts is the sampling interval and $\lambda = \exp(-Ts/Tref)$.(note that $0<\lambda<1$). That is , the reference trajectory is defined to be

$$r(k+i|k) = s(k+i) - Є(k+i)$$

$$= s(k+i) - \exp(-Ti/Ts) * Є(k)$$

The notation r(k+i|k) indicates that the reference trajectory depends on the conditions at time k,in general. Alternative definitions of the reference trajectory are possible—For e.g. , a straight line from the current output which meets the set point trajectory after a specified time .

A predictive controller has an *internal model* which is used to predict the behaviour depends on the assumed input trajectory ŭ(k+i|k) (i=0,1,…,Hp-1) that is to applied over the prediction horizon, and the idea is to select that input which promises best predicted behaviour.We shall assume that internal model is linear ; this makes the calculation of the best input relatively straightforward. The notation ŭ rather than u here indicates that at time step k we only have a prediction of what the input at time k+i may be; the actual input at that time,u(k+i),will probably be different from ŭ(k+i|k).Note that we assume that we have the output measurement y(k) available when deciding, the value of the input u(k).This implies that our internal model must be strictly proper , namely that according to the model y(k) depends on the past inputs u(k-1),u(k-2), …, but not on the input u(k).

In the simplest case we can try to choose the input trajectory such as to bring output at the end of the prediction horizon, namely at time k_Hp , to the required value r(k + Hp). In this case we say, using the terminology of richalet,that we have a single *coincidence point* at time k+Hp. There are several input trajectories {ŭ(k|k),ŭ(k+1|k),…,ŭ(k+Hp-1|k)} which achieve this , and we could choose one of

them , for example the one which requires smallest input energy. But is usually adequate , and in a fact preferable, to impose some simple structure o the input trajectory, parameterized by a smaller number of variables. The figure shows the input assumed to vary over the first three steps of the prediction horizon, but to remain constant thereafter: ŭ(k|k)=ŭ(k+1|k)=…=ŭ(k+Hp-1|k).In this case there is only one equation to be satisfied ---ŷ (k+Hp|k)=r(k+Hp|k)--- there is a unique solution .

Once a future input trajectory has been chosen , only the *first element* of that trajectory is applied as the input signal to the plant . That is , we set u(k)=ŭ(k|k), where u(k) denotes the actual input signal applied. Then the whole cycle of output measurement is repeated, prediction, and input trajectory determination is repeated., one sampling interval later: a new output measurement y(k+1) is obtained ;a new reference trajectoryr(k+i|k+1)(i=2,3,…,) is defined ; predictions are made over the horizon k+1+I,with i=1,2,…Hp; a new trajectory ŭ(k+1+i|k+1),with i=0,1,…,Hp-1) is chosen; and finally the next input is applied to the plant : u(k+1)=ŭ(k+1|k+1).Since the horizon prediction remains of the same length as before, but slides along by one sampling interval at each step this way of controlling a plant is often called a *receding horizon strategy*

CHAPTER 3

OPTIMIZATION PROBLEM

OBJECTIVE FUNCTIONS

MOEDELS

FINITE STEP RESPONSE

FINITE IMPULSE RESPONSE

OPTIMIZATION PROBLEM

The term optimization implies a best value for some type of performance criterion. This performance criterion is Known as an objective function. Here, we first discuss possible objective functions, then possible process models that can be used for MPC.

OBJECTIVE FUNCTIONS

Here, there are several different choices for objectives functions. The first one that comes to mind is a standard *least-squares* or "quadratic "objective function. The objective function is a "sum of squares " of the predicted errors (differences between the set points and model-predicted outputs) and the control moves (changes in control action from step to step)

A quadratic objective function for a prediction horizon of 3 and a control horizon of 2 can be written

$$\Phi = (Rk+1 - \hat{y}k+1)^2 + ((Rk+2 - \hat{y}k+2)^2 + (Rk+3 - \hat{y}k+3)^2 + w\Delta Uk^2 + w\Delta Uk+1^2$$

Where $\hat{y}$ represents the model predicted output ,r is the set point, ΔU is the change in manipulated input from one sample to the next ,w is a weight for the changes in the manipulated input, and the subscripts indicate the sample time (k is the current sample time). For a prediction horizon of P and a control horizon of M,the least Squares objective function is written

$$\Phi = \sum (Rk+1 - \hat{y}k+1)^2 + w\sum \Delta Uk+1^2$$

Another possible objective function is to simply take a sum of the absolute values of the predicted errors and control moves.

For a prediction horizon of 3 and a control horizon of 2, the absolute value objective function is

$$\Phi = | (Rk+1 - \hat{y}k+1) | + | (Rk+2 - \hat{y}k+2) | + | (Rk+3 - \hat{y}k+3) | + w| \Delta Uk| + w| \Delta Uk+1|$$

Which has the following general form for a prediction horizon of P and a control horizon of M:

$$\Phi = \sum | (Rk+1 - \hat{y}k+1)| + w\sum |\Delta Uk+1|$$

The optimization problem solved stated as a minimization of the objective function, obtained by adjusting the M control moves, subject to modeling equations (equality constraints), and constraints on the inputs and outputs.

$$\text{Min} \quad \Phi$$

Least-squares formulations are by far the most common objective functions in MPC.Least squares yields analytical solutions for unconstrained problems and penalizes larger errors(relatively) more then smaller errors. The absolute value objective function has been used in a few algorithms because linear programming (LP) problem results.LPs are routinely solved in large-scale scheduling and allocation problems. For example, an oil company often uses an LP to decide how to distribute oil to various refineries and to decide how much and what product to produce at each plant .The LP approach is not useful for model predictive control, because the manipulated variable moves often " hop" from one extreme constraint to another.

MODELS

Many different types of models are possible for calculating the predicted values of the process outputs, which are used in evaluating at discrete steps , it makes sense to use discrete models for the output prediction . Here , we review step and impulse response models both of which are used in common MPC algorithms.

FINITE STEP RESPONSE

FSR models are obtained by making a unit step input change to a process operating at steady state. The model coefficients are simply the output values at each time step.

Here , si represents the step response coefficients for the ith sample time after the unit step input change. If a non-unit step change is made, the output is scaled accordingly.

The step response model is the vector of step response coefficients,

$$S=[\ s1\ s2\ s3\ s4\ s5\ \ldots\ sN]'$$

Where the model length N is long enough so that the coefficients values are relatively constant (i.e. the process is close to a new steady state).

FINITE IMPULSE RESPONSE

Another common form of model is a finite impulse (FIR). Here , a unit pulse is applied to the manipulated input, and the model coefficients are simply the values of the outputs the ith impulse response coefficients.

There is a direct relationship between step and impulse response models:

$$Hi=Si-Si-1$$

$$Si=\sum hj$$

The impulse response coefficients are simply the changes in the step response coefficient at each time step. Similarly , step response coefficient is the sum of the impulse response coefficients to that point. It should be noted that there are two major limitations to step and impulse response models. They can only be used to represent open-loop stable processes, and they require a large number of parameters (model coefficients) compared to state space and transfer function models.

CHAPTER 4

DYNAMIC MATRIX CONTROL

DMC

Engineers at Shell Oil developed their own independent MPC technology in the early 1970's, with an initial application in 1973. Cutler and Ramaker presented details of an unconstrained multivariable control algorithm which they named Dynamic Matrix Control (DMC) at the 1979 National AIChE meeting [] and at the 1980 Joint Automatic Control Conference []. In a companion paper at the 1980 meeting Prett and Gillette [] described an application of DMC technology to an FCCU reactor/regenerator in which the algorithm was modified to handle nonlinearities and constraints. Neither paper discussed their process identification technology. Key features of the DMC control algorithm include:

- linear step response model for the plant
- quadratic performance objective over a finite prediction horizon
- future plant output behavior specified by trying to follow the set point as closely as possible
- optimal inputs computed as the solution to a least-squares problem

The linear step response model used by the DMC algorithm relates changes in a process output to a weighted sum of past input changes, referred to as input moves. For the SISO case the step response model looks like:

$$y_{k+j} = \sum_{i=1}^{N-1} s_i \, \Delta u_{k+j-i} \; + \; s_N \, u_{k+j-N}$$

The move weights are the step response coefficients. Mathematically the step response can be defined as the integral of the impulse response; given one model form the other can be easily obtained. Multiple outputs were handled by superposition. By using the step response model one can write predicted future output changes as a linear combination of future input moves. The matrix that ties the two together is the so-called *Dynamic Matrix*. Using this representation allows the optimal move vector to be computed analytically as the solution to a least-squares problem. Feed forward control is readily included in this formulation by modifying the predicted future

outputs. In practice the required matrix inverse can be computed off-line to save computation. Only the first row of the final controller gain matrix needs to be stored because only the first move needs to be computed.

The objective of a DMC controller is to drive the output as close to the set point as possible in a least-squares sense with a penalty term on the MV moves. This is equivalent to increasing the size of the diagonal terms in the square solution matrix prior to inversion. This results in smaller computed input moves and a less aggressive output response. As with the IDCOM reference trajectory, this technique provides a degree of robustness to model error. Prett and Gillette formalized this concept mathematically by defining move suppression factors designed to penalize excessive input movement. Move suppression factors also provide an important numerical benefit in that they can be used to directly improve the conditioning of the numerical solution.

Cutler and Ramaker showed results from a furnace temperature control application to demonstrate improved control quality using the DMC algorithm. Feedforward response of the DMC algorithm to inlet temperature changes was superior to that of a conventional PID lead/lag compensator.

In their paper Prett and Gillette [] described an application of DMC technology to FCCU reactor/regenerator control. Four such applications were already completed and two additional applications were underway at the time the paper was written. The overall FCCU control system was implemented in a multi-level hierarchy, with a nonlinear steady-state FCCU model at the top. At the start of each optimization cycle, parameters in the nonlinear model were estimated so as to match model predictions with measured steady-state operating data. The calibrated nonlinear model was then perturbed numerically to generate partial derivatives of each process output with respect to each process input (the matrix of partial derivatives is known as the Jacobian matrix in numerical analysis). The partial derivatives were then used in a Linear Program (LP) to compute a new economic optimal operating point for the FCCU, subject to steady-state process constraints. The optimal process input and output targets were then passed to a DMC algorithm for implementation. As soon as

the DMC controller moved the unit to the new steady state the optimization cycle was repeated. This separation of the control system into constrained steady-state optimization and dynamic control is quite similar to the structure described by Richalet et al. and has since become standard in industrial control system design.

The DMC algorithm had the job of moving from the system from one optimal steady-state to another. Although the LP solution provided optimal targets for process inputs and outputs, dynamic disturbances could potentially cause the DMC algorithm to move inputs away from their optimal steady-state targets in order to keep outputs at their steady-state targets. Since moving one input away from its optimal target may be much more expensive than moving another, the control system should determine this trade-off in a rational way. The DMC algorithm was modified to account for such trade-offs by including an additional equation for each input in the process model. The new equation required that the sum of all moves for a particular input should equal the total adjustment required to bring that input to its optimal steady-state target. This allowed the inputs some freedom to move dynamically but required that the steady-state input solution be satisfied in a least-squares sense, with trade-offs determined by the appropriate objective function weights.

Prett and Gillette described additional modifications to the DMC algorithm to prevent violation of absolute input constraints. When a predicted future input came sufficiently close to an absolute constraint, an extra equation was added to the process model that would drive the input back into the feasible region. These were referred to as time variant constraints. Because the decision to add the equation had to be made on-line, the matrix inverse solution had to be recomputed at each control execution. Prett and Gillette developed a matrix tearing solution in which the original matrix inverse could be computed off-line, requiring only the matrix inverse corresponding to active time variant constraints to be computed on-line.

The initial IDCOM and DMC algorithms represent the *first generation* of MPC technology; they had an enormous impact on industrial process control and served to define the industrial MPC paradigm.

Summarizing the main steps involved in implementing DMC on a process are as follows:

1. Develop a discrete step response model with length N based on sample time Δt.
2. Specify the prediction(P) and control (M) horizons.$N \geq P \geq M$
3. Specify the weighting on the control action(w=0 if no weighting on the control action(w=0 if no weighting).
4. All calculations assume deviation variable form, so remember to convert to/from physical units.

The effect of all these tuning parameters is now discussed for SISO systems.

Model-length and sample-time selection are independent. The model length should be approximately the 'settling time' of the process, that is, the time required to reach a new steady state after a step input change. For most systems, the model length is roughly 50 coefficients. The sample time is usually on the order of one tenth the dominant time constant, so the model length is roughly the settling time of the process.

Prediction and control horizons differ in length. Usually, the prediction horizon is selected to be much longer than the control horizon. This is particularly true if the control weighting factor is selected to be zero. Usually, if the prediction horizon is much longer than the control horizon, the control system is less sensitive to model error. Often P=20 or so , while M=1-3.

Control weighting is often step to zero if the prediction horizon is much longer the control horizon. As the control horizon is increased, the control moves tend to become more aggressive so larger weight is needed to penalize the control moves.

CHAPTER5

IMPLEMENTATION OF MPC IN MATLAB

VAN DE VUSSE REACTOR

INTRODUCTION OF NOISE TO MPC

VAN DE VUSSE REACTOR

Consider the Van de Vusse reactor problem. The continuous state space model is given by

A= [-2..4048 0; 0.8333 -2.2381];

B= [7;-1.117]

C= [0 1]

D= [0]

Where the measured state (output) is the concentration of the second component and the manipulated input is the dilution rate.

MATLAB code

```
%initialization of parameters
P=10;%prediction horizon
M=1;%control horizon
N=50;%model length
w=0.0;%weight
ysp=1;%output set point from 0
timesp=1;%time of set point change
delt=0.1;%sampling time interval
tfinal=6;%final simulation time
noise=0;
%define time
tvec=0:delt:tfinal;
ksp=fix(timesp/delt);
kfinal=length(tvec);
```

```
%define set point vector
r=[zeros(1,ksp),ones(1,(kfinal-ksp))*ysp];
%/////////////////define plant as 'SISO' LTI object//////////////////////
c=input('enter plant in 1.statespace 2.transferfunction 3.polezero
4.frquencyresponse');
if c==1  a=input('enter A matrix');b=input('enter B matrix');
c=input('enter C matrix');d=input('enter D matrix');plant=ss(a,b,c,d);
elseif c==2 nump=input('enter numerator coefficients');denp=input('enter
denomenator coefficients');plant=tf(nump,denp);
elseif c==3  zero=input('enter zeroes');pole=input('enter poles');K=input('enter gain');
plant=zpk(zero,pole,K);
elseif c==4 resp=input('enter response');freq=input('enter
frequencies');plant=frd(resp,freq,'Units','Hz');
end
plant=tf(plant);
%plant=s/(s*s - 1.4*s  +0.45),it is continous
%define plant parameters here

% nump=[1];
% denp=[1,-1.4,0.45];
% plant=tf(nump,denp);

%discretize the plant
plant=c2d(plant,delt);
%/////////////////define model here/////////////////////////////////
%assumption plant = model
model=plant;
% [numm,denm,tm]=tfdata(plant);
numm = get(model,'num'); numm = numm{:}; % Get numerator polynomial
denm = get(model,'den'); denm = denm{:}; % Get denominator polynomial
numm
```

```
%define step response coefficient matrix
s=step(model,0:delt:N*delt);
%define free response i.e. Sp matrix for past control moves
for i=1:P
   for j=1:N-2
     if(i+j<=N-1)
      Sp(i,j)=s(i+j);
     else
        Sp(1,j)=0;
     end
   end
end
 %define forced response i.e. Sf matrix for future and control moves
 for i=1:P
    for j=1:M
       if i+1-j>0
       Sf(i,j)=s(i+1-j);
       else
          Sf(i,j)=0;
       end
    end
 end
Sf
 % obtain W matrix
 W=w*eye(M,M);
 %obtain Kmat where Kmat=(Sf'*Sf  +   W)^-1*Sf'
Kmat=inv(Sf'*Sf + W)*Sf';
%piant initial conditions
ndenm=length(denm)-1;
nnumm=length(numm)-1;
umpast=zeros(1,nnumm);
```

```
ympast=zeros(1,ndenm);
% uu=zeros(1,kfinal);
% yy=zeros(1,kfinal);
% xinit=zeros(1,size(

% nump=[zeros(1,ndenp-nnump-1),nump]; % Pad numerator with leading zeros
% numm=[zeros(1,ndenm-nnumm-1),numm];
uinit=0;
yinit=0;
%initialize input vector
u=ones(1,min(P,kfinal))*uinit;
u
dist(1)=0;
y(1)=yinit;
% x(:,1)=xinit;
dup=zeros(1,N-2);
for k=1:kfinal
    [m,p]=size(Kmat);
   for i=1:p
      if k-N+i>0
         uold(i)=u(k-N+i);
      else
         uold(i)=0;
      end
   end
   dvec=dist(k)*ones(1,p);
   rvec=r(k)*ones(p,1);
   y_free=Sp*dup' + s(N)*uold'+dvec';
   E=rvec-y_free;
   delup(k)=Kmat(1,:)*E;
   if k>1
```

```
    u(k)=u(k-1)+delup(k);
  else
    u(k)=delup(k)+uinit;
  end
  %plant equations
  umpast=[u(k),umpast(1,1:length(umpast)-1)];
  y(k+1)=-denm(2:ndenm+1)*ympast'+numm(2:nnumm+1)*umpast';
  ympast=[y(k+1),ympast(1:length(ympast)-1)];
  %model prediction
  if k-N+1>0
    ymod(k+1)=Sf(1,1)*delup(k)+Sp(1,:)*dup'+s(N)*u(k-N+1);
  else
    ymod(k+1)=Sf(1,1)*delup(k)+Sp(1,:)*dup';
  end
  %disturbance compensation
  dist(k+1)=y(k+1)-ymod(k+1);
  %additive disturbance compensation
  %put input change into vector of past control moves
  dup=[delup(k),dup(1,1:N-3)];
end
%stairs plotting for input(zero order hold) and setpoint
[tt,uu]=stairs(tvec,u);
[ttr,rr]=stairs(tvec,r);
figure(1)
subplot(2,1,1)
plot(ttr,rr,'--',tvec,y(1:length(tvec)))
ylabel('y');
xlabel('time');
title('plant output');
subplot(2,1,2)
plot(tt,uu)
```

```
ylabel('u');
xlabel('time');
```

OUTPUT IN MATLAB WINDOW

```
enter plant in     1.statespace    2.transferfunction     3.polezero    4.frquencyresponse1
enter A matrix[-2.4048 0;0.8333 -2.2381]
enter B matrix[7;-1.117]
enter C matrix[0 1]
enter D matrix[0]

numm =

         0   -0.0751    0.1001


Sf =

         0
   -0.0751
   -0.0940
   -0.0768
   -0.0376
    0.0137
    0.0704
    0.1281
    0.1840
    0.2362
```

u = 0 0 0 0 0 0 0 0 0 0

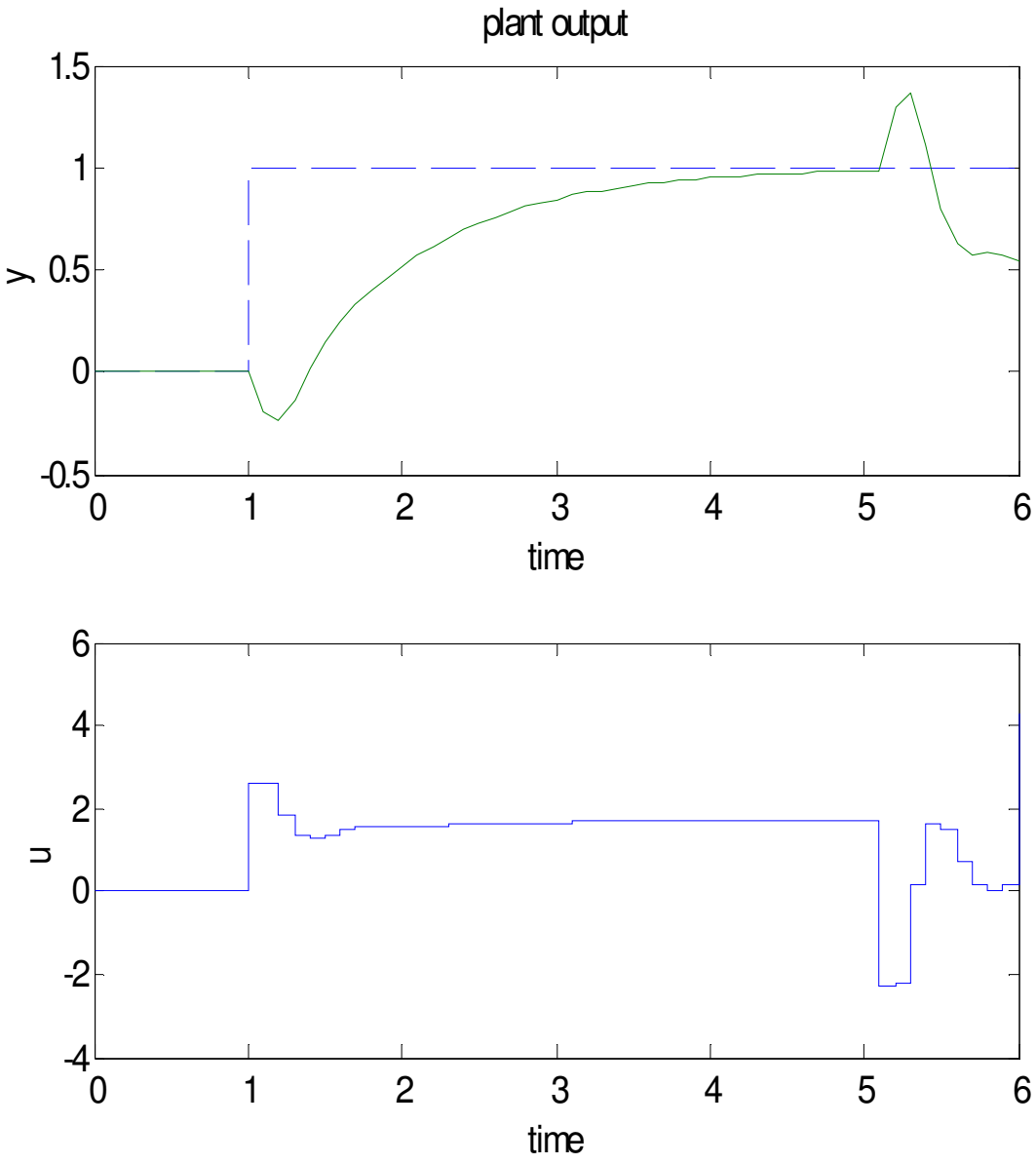

Fig 5.1 Output after applying MPC to the Van De Vusse Reactor

INFERENCE:

Effect of prediction horizon: If we have a fixed control horizon, then it is seen that choosing a smaller prediction horizon results set point being achieved in smaller time .However the shorter prediction horizon is more sensitive to model uncertainty.

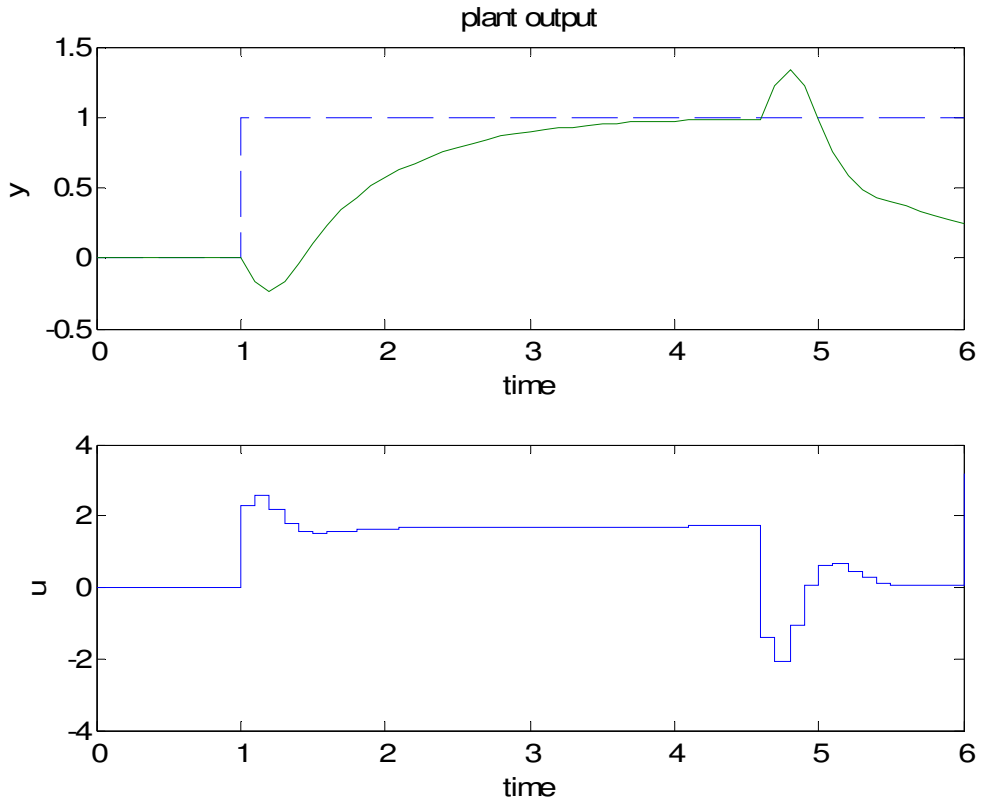

Fig 5.2 Output after applying MPC to the Van De Vusse Reactor with P=15

As seen in figure when P=15 requires much more control action compared to P=10. But still we find that prediction horizon does not have appreciable effect for this case .The performance for this case is roughly the same for P=10 and P=15.However there is a lower limit to the length of the prediction horizon below which it results in an unstable system. Here it is P=3. This is not due to any model error, since we have assumed a perfect model in these simulations. If the prediction horizon is too short, the initial step response coefficients dominate. Since these are negative while the later coefficients are positive (corresponding to a positive process gain), the predictive is

really in error. The effect is the same as using a PID controller with a controller gain that is the wrong sign.

Effect of model length: Choosing a smaller model length does not capture the complete dynamics of the process. This results in a model error and poor performance.

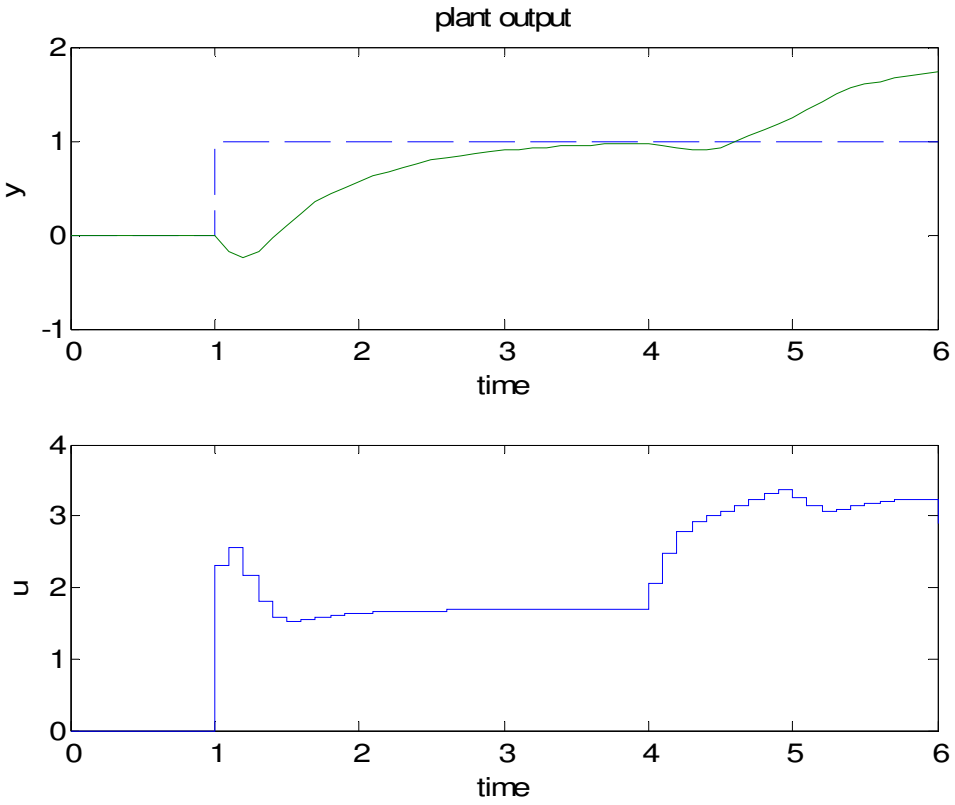

Fig 5.3 Output after applying MPC to the Van De Vusse Reactor with N=70

we find that N=50 gives better results than N=70.the performance degrades sharply as N increases.

INTRODUCTION OF NOISE IN MODEL PREDICTIVE CONTROL

This program provides offset tracking, but in addition simulates the effects of measurement noise, and of input and output disturbances as shown in figure

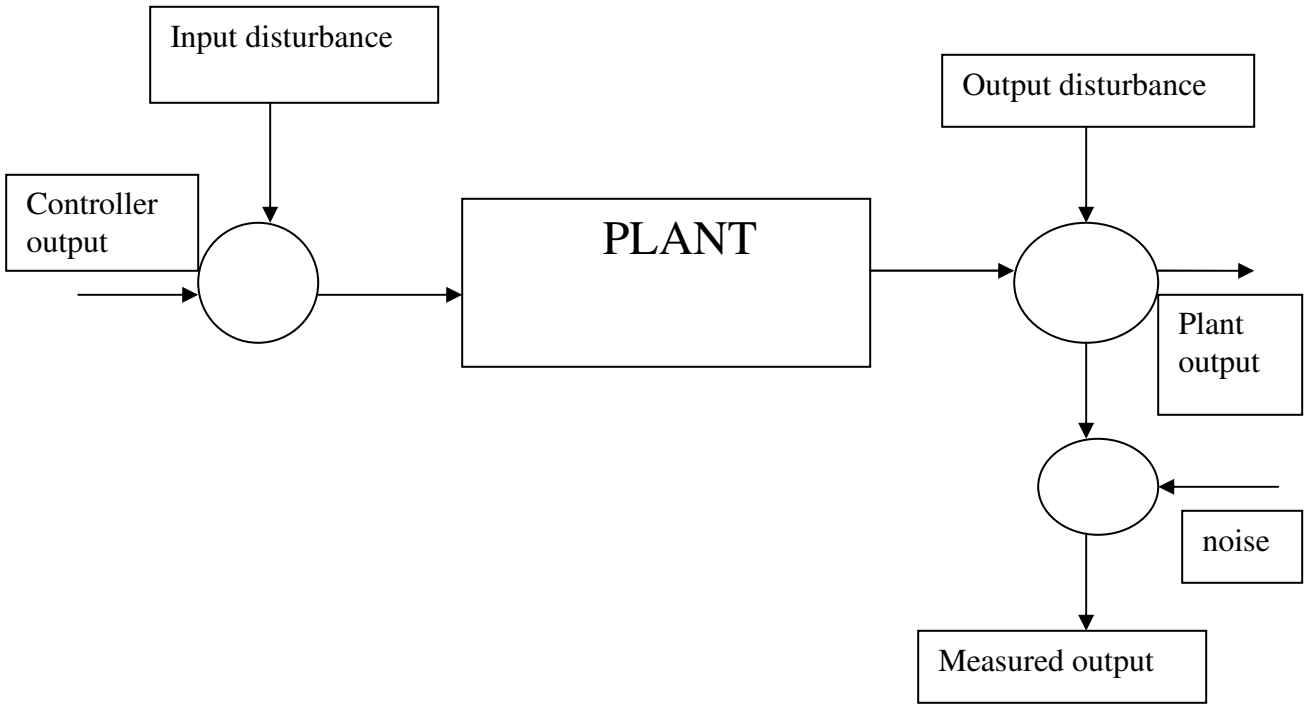

Fig 5.4 Input and output disturbances with measurement Noise

MATLAB code

```
%initialization of parameters
P=10;%prediction horizon
M=1;%control horizon
N=50;%model length
w=0.0;%weight
ysp=1;%output set point from 0
timesp=1;%time of set point change
delt=0.1;%sampling time interval
tfinal=4;%final simulation time
noise=0;
%define time
tvec=0:delt:tfinal;
ksp=fix(timesp/delt);
kfinal=length(tvec);
%define set point vector
r=[zeros(1,ksp),ones(1,(kfinal-ksp))*ysp];
%/////////////////define plant as 'SISO' LTI object//////////////////////
c=input('enter plant in 1.statespace 2.transferfunction 3.polezero
4.frquencyresponse');
if c==1  a=input('enter A matrix');b=input('enter B matrix');
c=input('enter C matrix');d=input('enter D matrix');plant=ss(a,b,c,d);
elseif c==2 nump=input('enter numerator coefficients');denp=input('enter
denomenator coefficients');plant=tf(nump,denp);
elseif c==3  zero=input('enter zeroes');pole=input('enter poles');K=input('enter gain');
plant=zpk(zero,pole,K);
elseif c==4 resp=input('enter response');freq=input('enter
frequencies');plant=frd(resp,freq,'Units','Hz');
end
```

```
plant=tf(plant);
%plant=s/(s*s - 1.4*s  +0.45),it is continous
%define plant parameters here

% nump=[1];
% denp=[1,-1.4,0.45];
% plant=tf(nump,denp);

%discritize the plant
plant=c2d(plant,delt);
%//////////////////define model here////////////////////////////////
%assumption plant = model
model=plant;
% [numm,denm,tm]=tfdata(plant);
numm = get(model,'num'); numm = numm{:}; % Get numerator polynomial
denm = get(model,'den'); denm = denm{:}; % Get denominator polynomial
numm
%define step response coefficient matrix
s=step(model,0:delt:N*delt);
%define free response i.e. Sp matrix for past control moves
for i=1:P
   for j=1:N-2
      if(i+j<=N-1)
       Sp(i,j)=s(i+j);
      else
         Sp(1,j)=0;
      end
   end
end
 %define forced response i.e. Sf matrix for future and control moves
 for i=1:P
```

```
    for j=1:M
        if i+1-j>0
        Sf(i,j)=s(i+1-j);
        else
            Sf(i,j)=0;
        end
    end
 end
Sf
 % obtain W matrix
 W=w*eye(M,M);
 %obtain Kmat where Kmat=(Sf'*Sf  +   W)^-1*Sf'
Kmat=inv(Sf'*Sf + W)*Sf';


% Noise and disturbances:
sd = 0.1; % Standard deviation of measurement noise
randn('state',0); % Resets state of random number generator. Change to
                % get different random sequences generated.
noise = sd*randn(kfinal,1); % Measurement noise, normal distribution,
                        % mean=0, standard deviation = sd.
udist = 0.1*ones(kfinal,1); % Input disturbance (default constant 0.1)
ydist = 0.1*[ones(floor(kfinal/2),1);-ones(ceil(kfinal/2),1)];
  % Output disturbance (default constant +/-0.1, changing sign halfway)


%piant initial conditions
ndenm=length(denm)-1;
nnumm=length(numm)-1;
umpast=zeros(1,nnumm);
ympast=zeros(1,ndenm);
```

```
% uu=zeros(1,kfinal);
% yy=zeros(1,kfinal);
% xinit=zeros(1,size(

% nump=[zeros(1,ndenp-nnump-1),nump]; % Pad numerator with leading zeros
% numm=[zeros(1,ndenm-nnumm-1),numm];
uinit=0;
yinit=0;
%initialize input vector
u=ones(1,min(P,kfinal))*uinit;
u
dist(1)=0;
y(1)=yinit;
% x(:,1)=xinit;
dup=zeros(1,N-2);


 for k=1:kfinal
    [m,p]=size(Kmat);
   for i=1:p
      if k-N+i>0
         uold(i)=u(k-N+i)+udist(k);
      else
         uold(i)=0+udist(k);
      end
   end
   dvec=ydist(k)*ones(1,p);
   rvec=r(k)*ones(p,1);
   dnoise=noise(k)*ones(1,p);
   y_freed=Sp*dup' + s(N)*uold'+dvec';
   y_free=y_freed+dnoise';
```

```
    E=rvec-y_free;
    delup(k)=Kmat(1,:)*E;
    if k>1
       u(k)=u(k-1)+delup(k);
    else
       u(k)=delup(k)+uinit;
    end
    %plant equations
    umpast=[u(k)+udist(k),umpast(1,1:length(umpast)-1)];
    y(k+1)=-denm(2:ndenm+1)*ympast'+numm(2:nnumm+1)*umpast';
    ympast=[y(k+1),ympast(1:length(ympast)-1)];
    %model prediction
    if k-N+1>0
       ymod(k+1)=Sf(1,1)*delup(k)+Sp(1,:)*dup'+s(N)*u(k-N+1);
    else
       ymod(k+1)=Sf(1,1)*delup(k)+Sp(1,:)*dup';
    end
    %disturbance compensation
    dist(k+1)=y(k+1)-ymod(k+1);
    %additive disturbance compensation
    %put input change into vector of past control moves
    dup=[delup(k),dup(1,1:N-3)];
end
%stairs plotting for input(zero order hold) and setpoint
[tt,uu]=stairs(tvec,u);
[ttr,rr]=stairs(tvec,r);
figure(1)
subplot(2,1,1)

% Plot output, solid line and set-point, dottedd line:
plot(tvec,y(1:length(tvec)),'-',tvec,y(1:length(tvec))+noise',':',...
```

```
        ttr,rr,'--');
grid;
title(...
'Plant output (solid), Measured output (dotted) and set-point (dashed)')
xlabel('Time')

subplot(212)
% plot input signal as staircase graph:
plot(tt,uu,'-');
hold on;
plot(tvec,u+udist',':')
grid;
title('Controller output (solid), Plant input (dotted)')
xlabel('Time')
```

output in Matlab window

```
enter plant in     1.statespace    2.transferfunction     3.polezero    4.frquencyresponse1
enter A matrix[-2.4048 0;0.8333 -2.2381]
enter B matrix[7;-1.117]
enter C matrix[0 1]
enter D matrix[0]

numm =

        0   -0.0751    0.1001
```

Sf =

0
-0.0751
-0.0940
-0.0768
-0.0376
0.0137
0.0704
0.1281
0.1840
0.2362

u =

0 0 0 0 0 0 0 0 0 0

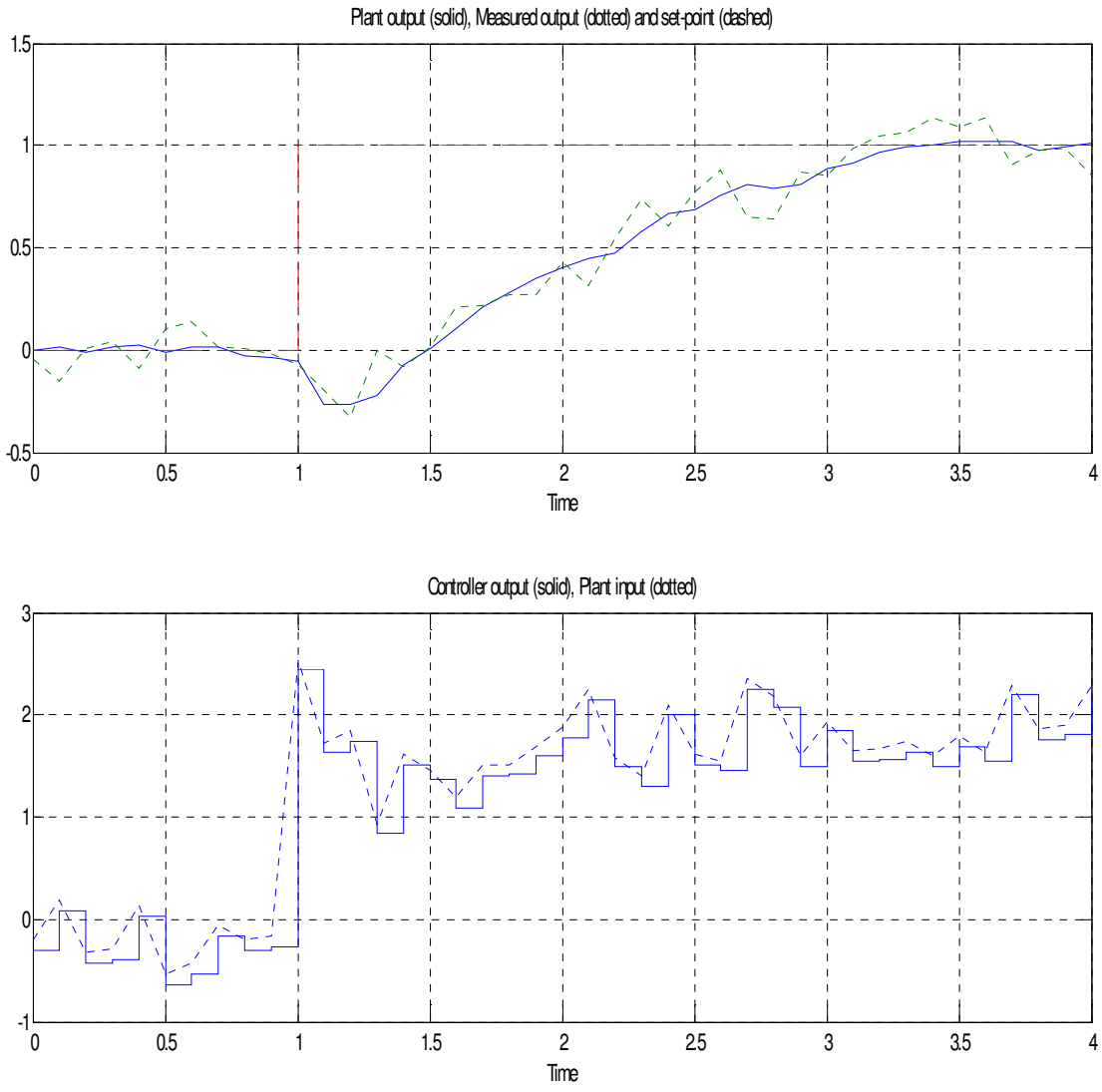

Fig 5.5 Output after adding Input and output disturbances with measurement Noise

CHAPTER 6

CASE STUDIES

CONTROL OF UNSTABLE HELICOPTER

SWIMMING POOL WATER TEMPERATURE CONTROL

CESSNA CITATION 500 AIRCRAFT CONTROL

CONTROL OF UNSTABLE HELICOPTER

The transfer function from the rotor angle to the forward speed of a helicopter in a particular flight condition is given by

$$9.8(S^2 - 0.5 + 6.3) / ((S + 0.6565)(S^2 - 0.2366S + 0.1493))$$

This has zeros at +0.25 + 2.5j ,+0.25 – 2.5j and poles at +0.118 + 0.37j,+0.118 – 0.37j. It is therefore both minimum phase and unstable –certainly a difficult plant to control .The figure below shows the response obtained using the program below when an exact model is assumed, and parameters Tref=6 and Ts=0.6 are used,with a single coincidence point P1=8,Hu=1.

MATLAB code:

```
% CONTROLLING HELICOPTER's FLIGHT CONDITION BY USING MODEL PREDICTIVE CONTROL
% The transfer function from the rotor angle to the forward speed of a helicopter
% in a particular flight condition is given by
% T(s)=9.8(s^2 -0.5s +6.3)/(s +0.6565)(s^2 -0.2366s +0.1493);

% Define time-constant of reference trajectory Tref:
Tref = 6;

% Define sampling interval Ts (default Tref/10):
if Tref == 0,
  Ts = 1;
else
```

```
  Ts = Tref/10;
end

% Define plant as SISO discrete-time 'lti' object 'plant'
%%%%%  CHANGE FROM  HERE TO DEFINE NEW PLANT %%%%%
nump=9.8*[1, -0.5, 6.3];                    % Helicopter example
denp=conv([1, 0.6565],[1, -0.2366, 0.1493]); % Continuous time
plant = tf(nump,denp);
plant = c2d(plant,Ts);  % Discretise plant
%%%%%  CHANGE UP TO HERE TO DEFINE NEW PLANT  %%%%%
plant = tf(plant);  % Coerce to transfer function form
nump = get(plant,'num'); nump = nump{:}; % Get numerator polynomial
denp = get(plant,'den'); denp = denp{:}; % Get denominator polynomial
nnump = length(nump)-1; % Degree of plant numerator
ndenp = length(denp)-1; % Degree of plant denominator
if nump(1)~=0, error('Plant must be strictly proper'), end;
if any(abs(roots(denp))>1), disp('Warning: Unstable plant'), end

% Define model as SISO discrete-time 'lti' object 'model'
% (default model=plant):
%%%%%  CHANGE FROM  HERE TO DEFINE NEW MODEL %%%%%
model = plant;
%%%%%  CHANGE UP TO HERE TO DEFINE NEW MODEL  %%%%%
model = tf(model);  % Coerce to transfer function form
numm = get(model,'num'); numm = numm{:}; % Get numerator polynomial
denm = get(model,'den'); denm = denm{:}; % Get denominator polynomial
nnumm = length(numm)-1; % Degree of model numerator
ndenm = length(denm)-1; % Degree of model denominator
if numm(1)~=0, error('Model must be strictly proper'), end;
if any(abs(roots(denm))>1), disp('Warning: Unstable model'), end
```

```
nump=[zeros(1,ndenp-nnump-1),nump]; % Pad numerator with leading zeros
numm=[zeros(1,ndenm-nnumm-1),numm]; % Pad numerator with leading zeros

% Define prediction horizon P (steps)(default corresponds to 0.8*Tref):
if Tref == 0,
  P = 5;
else
  P = round(0.8*Tref/Ts);
end

% Define control horizon (default 1):
M = 1;

% Compute model step response values over coincidence horizon:
stepresp = step(model,[0:Ts:max(P)*Ts]);
N=50;%model length
w=0.0;%weight
ysp=1;%output set point from 0
timesp=1;%time of set point change
delt=0.6;%sampling time interval
tfinal=25;%final simulation time
noise=0;
%define time
tvec=0:delt:tfinal;
ksp=fix(timesp/delt);
kfinal=length(tvec);
%define set point vector
r=[zeros(1,ksp),ones(1,(kfinal-ksp))*ysp];
%define step response coefficient matrix
s=step(model,0:delt:N*delt);
%define free response i.e. Sp matrix for past control moves
```

```
for i=1:P
   for j=1:N-2
     if(i+j<=N-1)
      Sp(i,j)=s(i+j);
     else
        Sp(1,j)=0;
     end
   end
end
 %define forced response i.e. Sf matrix for future and control moves
 for i=1:P
    for j=1:M
       if i+1-j>0
       Sf(i,j)=s(i+1-j);
       else
          Sf(i,j)=0;
       end
    end
 end
Sf
 % obtain W matrix
 W=w*eye(M,M);
 %obtain Kmat where Kmat=(Sf'*Sf  +   W)^-1*Sf'
Kmat=inv(Sf'*Sf + W)*Sf';
%piant initial conditions
ndenm=length(denm)-1;
nnumm=length(numm)-1;
umpast=zeros(1,nnumm);
ympast=zeros(1,ndenm);
% uu=zeros(1,kfinal);
% yy=zeros(1,kfinal);
```

```
% xinit=zeros(1,size(

% nump=[zeros(1,ndenp-nnump-1),nump]; % Pad numerator with leading zeros
% numm=[zeros(1,ndenm-nnumm-1),numm];
uinit=0;
yinit=0;
%initialize input vector
u=ones(1,min(P,kfinal))*uinit;
u
dist(1)=0;
y(1)=yinit;
% x(:,1)=xinit;
dup=zeros(1,N-2);
for k=1:kfinal
    [m,p]=size(Kmat);
   for i=1:p
      if k-N+i>0
         uold(i)=u(k-N+i);
      else
         uold(i)=0;
      end
   end
   dvec=dist(k)*ones(1,p);
   rvec=r(k)*ones(p,1);
   y_free=Sp*dup' + s(N)*uold'+dvec';
   E=rvec-y_free;
   delup(k)=Kmat(1,:)*E;
   if k>1
      u(k)=u(k-1)+delup(k);
   else
      u(k)=delup(k)+uinit;
```

```
    end
    %plant equations
    umpast=[u(k),umpast(1,1:length(umpast)-1)];
    y(k+1)=-denm(2:ndenm+1)*ympast'+numm(2:nnumm+1)*umpast';
    ympast=[y(k+1),ympast(1:length(ympast)-1)];
    %model prediction
    if k-N+1>0
       ymod(k+1)=Sf(1,1)*delup(k)+Sp(1,:)*dup'+s(N)*u(k-N+1);
    else
       ymod(k+1)=Sf(1,1)*delup(k)+Sp(1,:)*dup';
    end
    %disturbance compensation
    dist(k+1)=y(k+1)-ymod(k+1);
    %additive disturbance compensation
    %put input change into vector of past control moves
    dup=[delup(k),dup(1,1:N-3)];
end
disp('***** Results from program file :')
disp(['Tref = ',num2str(Tref),',  Ts = ',num2str(Ts),...
          '  P = ',int2str(P'),' (steps),  M = ',int2str(M)])
diffpm = get(plant-model,'num');
if diffpm{:}==0,
  disp('Model = Plant')
else
  disp('Plant-Model mismatch')
end

figure
subplot(211)
% Plot output, solid line and set-point, dashed line:
[tt,uu]=stairs(tvec,u);
```

```
[ttr,rr]=stairs(tvec,r);
figure(1)
subplot(2,1,1)
plot(ttr,rr,'--',tvec,y(1:length(tvec)),'-')
ylabel('y');
xlabel('time');
title('plant output');

% plot(tvec,yp(1:nsteps),'-',tvec,setpoint(1:nsteps),'--');
grid; title('Plant output (solid) and set-point (dashed)')
% xlabel('Time')

subplot(212)
% plot input signal as staircase graph:
% stairs(tvec,uu,'-');
plot(tt,uu)
ylabel('u');
xlabel('time');
grid; title('Input')
% xlabel('Time')
```

OUTPUT IN MATLAB WINDOW

Warning: Unstable plant
Warning: Unstable model

Sf =

0
6.4720
21.9203
55.8944
115.3825
204.9810
326.9132
480.9769

u =

0 0 0 0 0 0 0 0

***** Results from program file :
Tref = 6, Ts = 0.6 P = 8 (steps), M = 1
Model = Plant

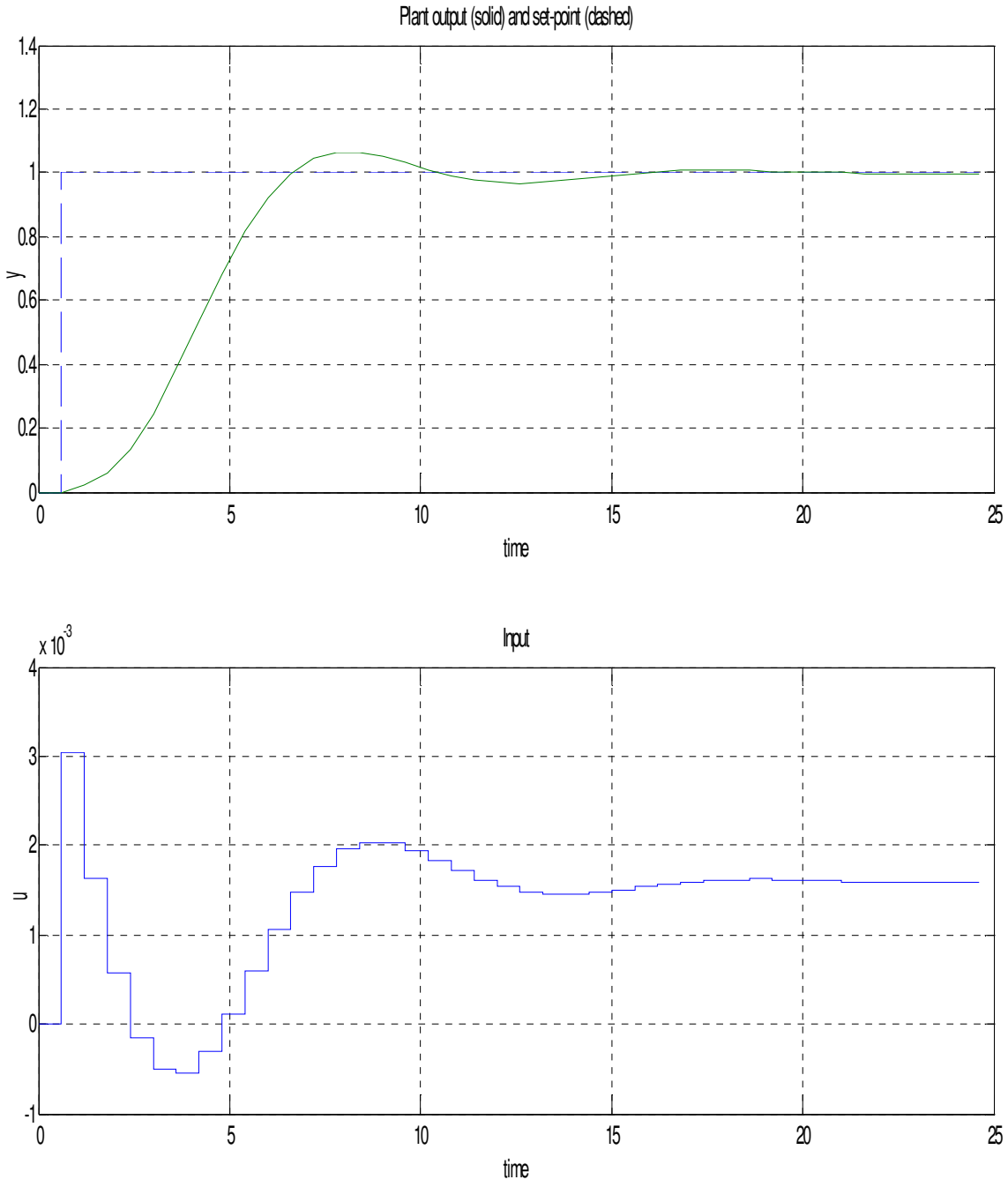

Fig 6.1 Output after applying MPC to control of unstable helicopter.

INFERENCE

This example shows that predictive control, although simple and intuitive in its formulation can produced some very sophisticated control action when required. In order to stabilize this unstable plant, classical theory tells us that Nyquist locus of the loop-gain frequency response must encircle the point -1 twice, and this can only be achieved by supplying phase lead at the appropriate frequencies. Doing this requires classical 'loop shaping' techniques. Predictive control achieves this implicitly, given only some reasonable design specifications.

On the other hand classical theory cannot be forgotten .This example tells us, for instance that the loop gain must increase to values greater than 1 at a frequency no lower than 0.4 rad/sec, approximately, and must decrease again to values smaller than 1 at a frequency no higher than 2.5 rad/sec approximately. Its range of possible behavior is rather restricted. This knowledge certainly helps to give the predictive controller a 'reasonable' specification to achieve. We find that improving the performance (for e.g. reducing the overshoot or speeding up the response) by adjusting the parameters of the predictive controller is not easy in this case.

SWIMMING POOL WATER TEMPERATURE CONTROL

The water temperature in a heated swimming pool ,θ, is related to the heater input power ,q, and the ambient air temperature ,θa, according to the equation

$$T(d\theta/dt) = kq + \theta a - \theta$$

Where T=1 hour and k=0.2 °C/kW (it is assumed that water is perfectly mixed, so that it has uniform temperature).Predictive control is to be applied to keep the water at a desired temperature, and a sampling interval Ts=0.25 hour is to be used. The control update is to be same as Ts.

Suppose the air temperature follows a sinusoidal diurnal variation with amplitude 10°C

$$\theta a(t) = 15 + 10\sin(2\pi t/24)$$

(Where t is measured in hours). Verify that in the steady state the mean water temperature reaches the set-point exactly, but that θ has a small oscillation of amplitude approximately 0.5°C.

MATLAB CODE:

```
% define parameters

%hour, sampling interval
Ts=0.25;
% input constraints, default
 ulim = [-inf,inf,1e6];
 % hour, time constant, default
```

```
Tplant = 1;
% degC/kW, heater gain, default
kplant = 0.2;
% artificial noise covariance, default
V =1e-3;
 % artificial process noise, default
 W =1;


 % Define parameters of internal model:

Tmodel = 1;   % hour, time constant
kmodel = 0.2; % degC/kW, heater gain

% Weights for MPC cost function:
Q=1; R=0;
% Horizons for MPC:
Hp=10; Hu=3;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Define plant:

aplantc = -1/Tplant;  bplantc = [kplant, 1]/Tplant; % continuous-time
cplant = 1;           dplant = [0, 0];
plantc = ss(aplantc,bplantc,cplant,dplant);  % LTI object

plantd = c2d(plantc,Ts);  % discrete-time equivalent
[aplantd,bplantd] = ssdata(plantd); % A and B matrices. (C and D stay unchanged)

plantinfo = [Ts,1,1,0,1,1,0]; % information for MOD format
```

```
                  % (1 state, SISO, 1 unmeasured disturbance)
plant = ss2mod(aplantd,bplantd,cplant,dplant,plantinfo); % plant in MOD format

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Define controller's internal model:

% Disturbance dynamics (diurnal variation):
aairc = [0, 1; -(2*pi/24)^2, 0]; % Air temp A matrix, continuous-time
% Complete continuous-time model, with state vector =
%   [water temp, air temp, air temp derivative]':
amodelc = [-1/Tmodel, 1/Tmodel, 0;
        zeros(2,1),  aairc    ];
bmodelc = [kmodel/Tmodel; 0; 0];
cmodel = [1, 0, 0];
dmodel = 0;
modelc = ss(amodelc,bmodelc,cmodel,dmodel);  % LTI object

modeld = c2d(modelc,Ts); % discrete-time equivalent
[amodeld,bmodeld] = ssdata(modeld); % A and B matrices. (C and D stay unchanged)

modinfo = [Ts,3,1,0,0,1,0];
model = ss2mod(amodeld,bmodeld,cmodel,dmodel,modinfo); % model in MOD format
% Now compute observer gain:
 % Kest = smpcest(model,W,V);
% Kest = smpccon(model,Q,R,Hu,Hp);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Now simulate pool with predictive control:
```

```
tend = 50; % end time for simulation
tvector = [0:0.25:tend]';  % time vector for plots etc
setpoint =  20; % deg C, setpoint for water temperature

airtemp = 15 + 10*sin(2*pi*tvector/24); % sine wave, period 24 hours

[wtemp,power] = scmpc(plant,model,Q,R,Hu,Hp,tend,setpoint,ulim,[],[],...
                    [],[],airtemp,[]);

 % Display results:
figure  % New figure
plotall(wtemp,power,tvector);
subplot(211), grid
hold on
plot(tvector,airtemp,'--')
xlabel('Time (hours)'), ylabel('Temperature (deg C)')
title('Water (solid) and Air (broken) Temperatures')
subplot(212), grid
xlabel('Time (hours)'), ylabel('Heater power (kW)')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

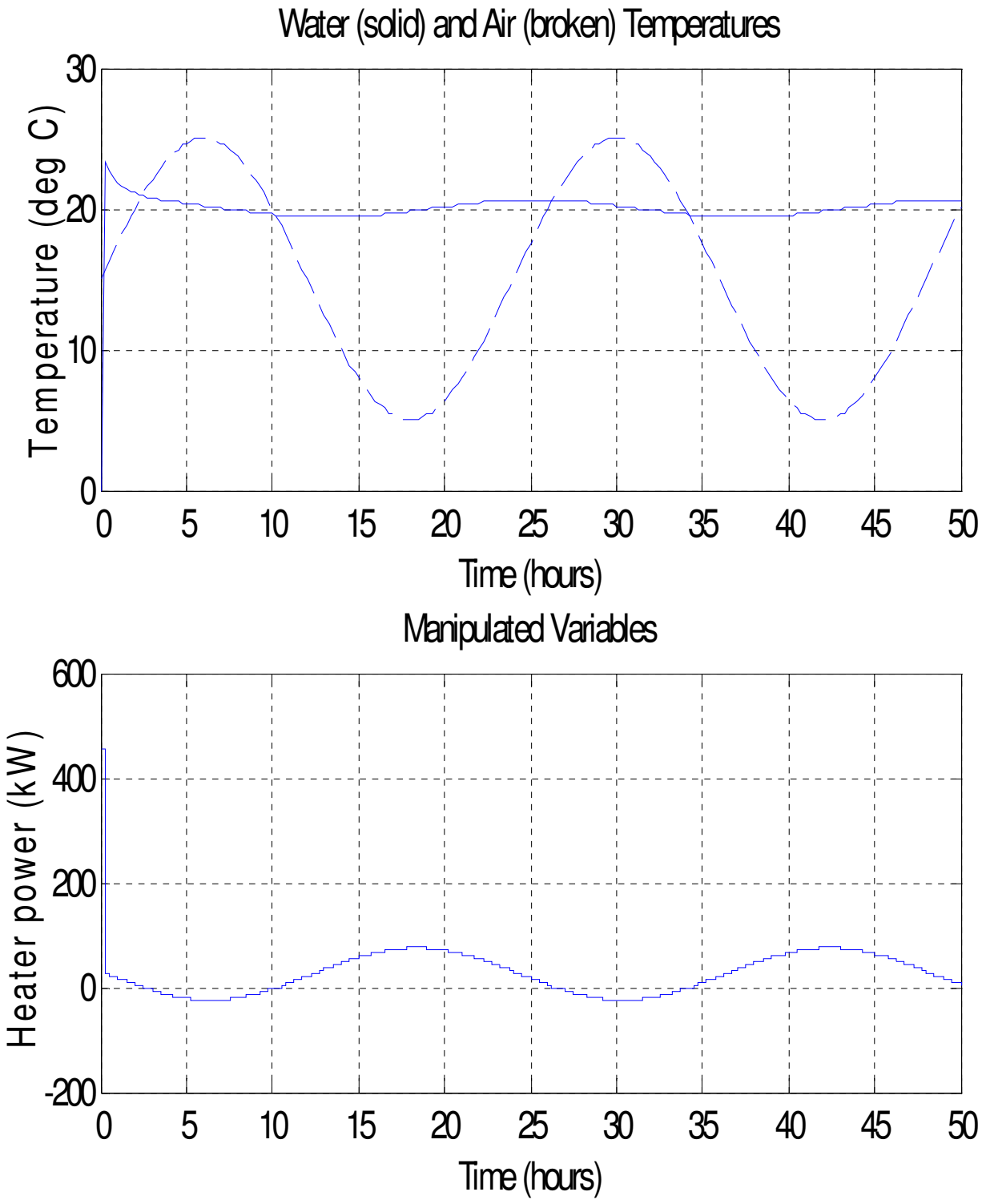

Fig 6.2 Output after applying MPC to control of water temperature of swimming pool

INFERENCE

Here to investigate the behavior of the predictive system we use *Model Predictive Control* toolbox function *scmpc.*

SCMPC-- It simulates closed loop problems by designing an MPC-type controller for constrained problems

USAGE: [y,u,ym]=scmpc(pmod,imod,ywt,uwt,M,P,tend, ...

r,ulim,ylim,Kest,z,v,w,wu)

Above program shows that a predictive controller with standard 'DMC' disturbance model does not compensate perfectly for a sinusoidal diurnal variation for a sinusoidal variation of the air temperature if the air temperature is not measured With the particular choice of parameters used there and a 10°C amplitude of the air temperature ,the water temperature oscillates with an amplitude of about 0.5°C if there are no constraints on the heater power. It is seen that if the air temperature is measured and used for feed forward control, then air temperature is perfectly compensated,providing that model is perfect .if some modeling error then residual oscillation remains. But if we want to remove it completely, then we have to model the sinusoidal disturbance and designing a suitable observer, even if the air temperature is not measured.

CHAPTER 11

CONCLUSION

Conclusions

MPC technology has progressed steadily in the twenty two years since the first IDCOM and DMC applications. Survey data reveal approximately 2200 applications to date, with a solid foundation in refining and petrochemicals, and significant penetration into a wide range of application areas from chemicals to food processing.

Current generation MPC technology offers significant new capabilities but the controllers still retain, for the most part, an IDCOM-like or a DMC-like personality. The SMC-Idcom and HIECON algorithms are IDCOM-like controllers which have evolved to use multiple objective functions and ranked constraints. The DMC, RMPCT and OPC algorithms are DMC-like controllers that use a single dynamic objective function to evaluate control and economic trade-offs using weighting factors. The PFC controller inherits some of the IDCOM personality but is significantly different in that it can accommodate nonlinear and unstable processes and uses basis functions to parameterize the input function.

An important observation is that industrial MPC controllers almost always use empirical dynamic models identified from test data. The impact of identification theory on process modeling is perhaps comparable to the impact of optimal control theory on model predictive control. It is probably safe to say that MPC practice is one of the largest application areas of system identification. The current success of MPC technology may be due to carefully designed plant tests.

Another observation is that process identification and control design are clearly separated in current MPC technology. Efforts towards integrating identification and control design may bring significant benefits to industrial practice. For example, uncertainty estimates from process identification could be used more directly in robust control design. Ill-conditioned process structures could be reflected in the identified models and also used in control design.

Choosing an MPC technology for a given application is a complex question involving issues not addressed in this paper. It is the opinion of the authors that for most applications, a knowledgeable control engineer could probably achieve acceptable control performance using any of the packages discussed here, although the time and

effort required may differ. If the process is nonlinear or unstable, or needs to track a complex set point trajectory with no offset, the PFC algorithm may offer significant advantages. If a vendor is to be selected to design and implement the control system, it would be wise to weigh heavily their experience with the particular process in question.

Research needs as perceived by industry are mostly control engineering issues, not algorithm issues. Industrial practitioners do not perceive closed loop stability, for example, to be a serious problem. Their problems are more like: Which variables should be used for control? When is a model good enough to stop the identification plant test? How do you determine the source of a problem when a controller is performing poorly? When can the added expense of an MPC controller be justified? How do you design a control system for an entire plant? How do you estimate the benefits of a control system? Answering these questions could provide control practitioners and theoreticians with plenty of work in the foreseeable future.

Several technical advances have not yet been incorporated into industrial MPC technology. These include using an infinite prediction horizon to guarantee nominal closed loop stability, and using linear estimation theory to improve output feedback. In addition, robust stability conditions have been developed for a modified QDMC algorithm. It would seem that the company which first implements these advances will have a significant marketing and technical advantage.

The future of MPC technology is bright, with all of the vendors surveyed here reporting significant applications in progress. Next-generation MPC technology is likely to include multiple objective functions, an infinite prediction horizon, nonlinear process models, better use of model uncertainty estimates, and better handling of ill-conditioning.

REFERENCE

BOOK

1. BEQUETTE B.W. "PROCESS CONTROL MODELLING DESIGN AND SIMULATION" prentice hall edition

2. MACIEJOWSKI J.M. "PREDICTIVE CONTROL WITH CONSTRAINTS", prentice hall edition.

INTERNET

S. Joe Qin and Thomas A. Badgwell "An Overview of Industrial Model Predictive Control Technology"

http://www.che.utexas.edu/~qin/cpcv/node2.html#

http://www.che.utexas.edu/~qin/cpcv/node4.html#

MIKAEL JOHANSON "Control theory and practice"

Mikael Johansson mikaelj@ee.kth.se

PDF file: William B. Dunbar "Notes on model predictive control"

dunbar@cds.caltech.edu