

# Technical Report EE04025 - Notes on Linear Model Predictive Control

A. G. Wills\*

December 22, 2004

## Abstract

Linear model predictive control (MPC) assumes a linear system model, that the constraints sets are representable via linear inequalities and that the objective function is convex quadratic. Linear MPC is appealing because the associated optimisation problem - typically solved at each time interval - may be expressed as a convex quadratic program, which can be solved efficiently online. Topics *not* covered include methods for solving quadratic programs, soft constraints, terminal state constraints, closed-loop stability, closed-loop robustness and optimal reference trajectory calculation. A Matlab simulation example is included for illustration.

## 1 Introduction

It is generally accepted that MPC (model predictive control) had its genesis in the process industries with the works of (Richalet *et al.* 1976), (Richalet *et al.* 1978), (Cutler and Ramaker 1980) and (Cutler *et al.* 1983) who devised heuristics to combat the problem of process constraints. With this in mind, it is interesting that MPC is often said to be the only advanced control strategy to have a substantial impact on industrial control engineering (Maciejowski 2002).

Like many statements about origin the above is no exception to controversy. Indeed, the central tenet of MPC is well aligned with that of optimal control from the 1960's, some twenty years its prior. From an optimal control perspective, MPC may be interpreted as a numerical implementation of the classical Receding Horizon Control (RHC) strategy and (Mayne *et al.* 2000) make reference to cases in the literature where this is explicit.

From a different perspective, Clarke *et al.* (1987) developed Generalised Predictive Control (GPC) which has received much attention from both theoretical and applied groups alike. GPC was developed within the adaptive control setting and is associated with input-output model structures. Inequality constraints on the process inputs were not considered within GPC until some time later. In light of this, (Bitmead *et al.* 1990) showed that unconstrained MPC is adequately subsumed within Linear Quadratic Gaussian (LQG) control.

The intention of writing these notes is not to provide a comprehensive historical account of MPC, but rather to develop one particular form of linear state-space MPC from the ground up. Further

---

\*School of Electrical Engineering and Computer Science, University of Newcastle, University Drive, Callaghan, NSW, 2308, Australia. Email: onyx@ecemail.newcastle.edu.au. Phone:+61 2 49215204. Fax:+61 2 49216993.

discussion of the former can be found in many surveys including those by (Clarke *et al.* 1987), (Garcia *et al.* 1989), (Muske and Rawlings 1993), (Chen and Allgöwer 1998), (Mayne *et al.* 2000), (Qin and Badgwell 1997) and (Maciejowski 2002).

Linear MPC, as opposed to more general forms of MPC, is attractive because the plant is modelled using a linear state-space relation and plant constraints are modelled using linear equalities and inequalities. When combined with a convex quadratic cost function this implies that the desired control action can be obtained, at each sample interval, via the solution of a corresponding quadratic program. This in turn is attractive because quadratic programs can be solved efficiently online.

These notes are structured as follows. In Section 2 a “big picture” discussion of MPC is provided which includes block diagrams, timing diagrams and a simplified algorithm. With a recipe in place, the ingredients are collected, which begins with a discussion of plant models and prediction in Section 3. This is then used in Section 4 where control objectives are discussed and a control objective function is designed. The MPC optimisation problem is defined and constraints are discussed at the end of this section. The important practical aspect of integral action is presented in Section 5, which amounts to a special choice of disturbance model. A more detailed algorithm, that separates offline and online computation is given in Section 6 and Section 7 concludes these notes with an example.

## 2 Big Picture

In these notes, the development of MPC is based on the block diagram given in Figure 1 and the timing diagram given in Figure 2. The main idea is to measure the plant output  $y(t)$ , make some calculations (including a state estimate  $\hat{x}(t)$ ) and deliver a new control action to the plant input  $u(t)$  all the while trying to achieve some prespecified objective (usually in the form of tracking some reference signal and rejecting plant disturbances). Other important signals (not present in Figure 1) will be introduced as needed.

In terms of separating the physical plant from the “controller”, it is constructive to think of the observer and MPC blocks as one unit since these operations are typically performed on a microprocessor, usually quite separate to the plant. Such a microprocessor must be able to complete all necessary computations within the sampling period  $T_s$  (see Figure 2) and this is possibly the greatest inhibitor for using MPC in general.

In the timing diagram  $T_c$  denotes total computation time relative to  $t$  and  $T_a$  denotes the time relative to  $t$  that the calculated control action is applied. The control action is usually applied at the next sample interval, i.e.  $T_a = T_s$ , since this is a fixed and predictable time reference<sup>1</sup>.

Broadly speaking, MPC follows the steps in the following algorithm at each iteration.

### ALGORITHM 1

1. Measure the system outputs and inputs.
2. Estimate the current state.
3. Compute the next control move by solving optimisation problem and apply at the correct time.

---

<sup>1</sup>If  $T_c$  is very small compared with  $T_s$ , then it may be useful to apply the control action sooner than the next sample interval since improved performance is possible (Maciejowski 2002).

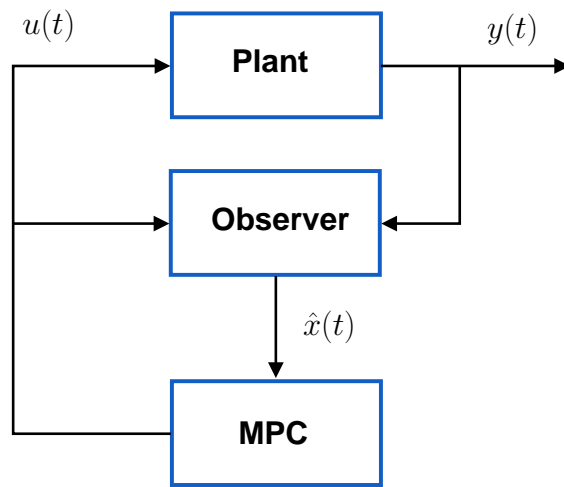


Figure 1: Block diagram of MPC.

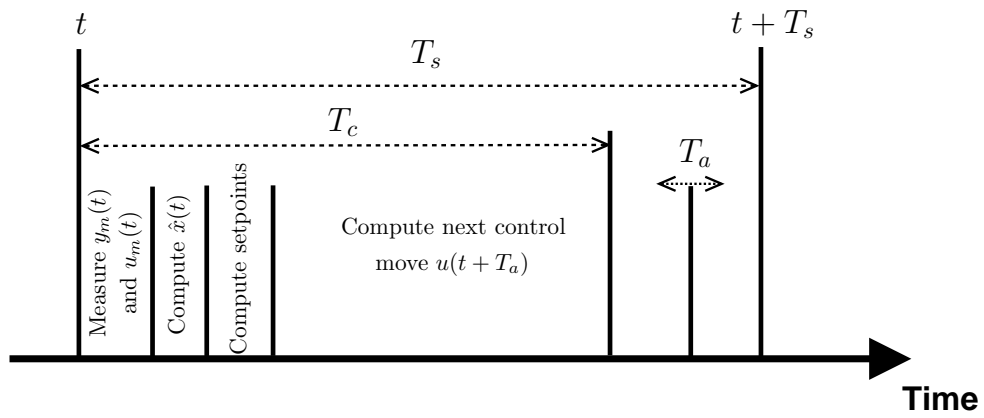


Figure 2: Timing diagram of MPC.

### 3 Plant Model and Prediction

As the name suggests, model predictive control uses a model to make predictions which are then used for control purposes. In this report a state-space model structure is preferred to input-output model structures because, in the author's opinion, the derivation is simplified, MIMO (multi-input-multi-output) plants are handled with relative ease and numerical conditioning is less of an issue.

The plant is modelled using the following state-space system.

$$\begin{aligned}x_{t+1} &= Ax_t + Bu_t + \omega_t, \\y_t &= Cx_t + Du_t + \nu_t.\end{aligned}\tag{1}$$

In the above, the state  $x_t \in \mathbb{R}^{n_x}$ , the input  $u_t \in \mathbb{R}^{n_u}$  and the output  $y_t \in \mathbb{R}^{n_y}$ . Further, the state noise  $\omega_t$  and measurement noise  $\nu_t$  are assumed to be Gaussian distributed with zero mean and respective covariances of  $W$  and  $V$  with cross-covariance  $Z$ , i.e.

$$\begin{bmatrix} \omega_t \\ \nu_t \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} W & Z \\ Z^T & V \end{bmatrix} \right)$$

This model is used to make predictions about plant behaviour over what is called the *prediction horizon*, denoted by  $N$ , using information (measurements of inputs and outputs) up to and including the current time  $t$ .

With the Gaussian assumptions on state and measurement noise in place, it is possible to make optimal (in the minimum variance sense) predictions of state and output using a Kalman Filter. Accordingly, let  $\hat{x}_{i|j}$  and  $\hat{y}_{i|j}$  represent estimates of the state and output at time  $i$  given information up to and including time  $j$  where  $j \leq i$ . Then

$$\hat{x}_{t+1|t} = A\hat{x}_{t|t-1} + Bu_t + K(y_t - \hat{y}_{t|t-1}),\tag{2}$$

$$\hat{y}_{t|t-1} = C\hat{x}_{t|t-1} + Du_t,\tag{3}$$

$$K = (APC^T + Z)(CPC^T + V)^{-1},\tag{4}$$

$$P = W + APA^T - (APC^T + Z)(CPC^T + V)^{-1}(Z^T + CPA^T).\tag{5}$$

The above equation for  $P$  is known as the discrete-time-algebraic-Riccati-equation (DARE) and robust solvers for such equations are available, see e.g. (Benner *et al.* 1997).

MPC usually requires estimates of the state and/or output over the entire prediction horizon from time  $t + 1$  until time  $t + N$ , and can only make these predictions based on information up to and including the current time  $t$ . Equations (2)–(5) can be used to obtain  $\hat{x}_{t+1|t}$ , and optimal estimates from time  $t + 2$  to  $t + N$  can be obtained as follows<sup>2</sup>.

$$\hat{x}_{t+i+1|t} = A\hat{x}_{t+i|t} + Bu_{t+i|t},\tag{6}$$

$$\hat{y}_{t+i|t} = C\hat{x}_{t+i|t} + Du_{t+i|t}.\tag{7}$$

In the above  $i = 1, \dots, N$  and the notation  $u_{t+i|t}$  is used to distinguish the actual input at time  $t + i$  from that used for prediction purposes, namely  $u_{t+i|t}$ .

Equation (6) can be expanded in terms of the initial state  $\hat{x}_{t+1|t}$  and future control actions  $u_{t+i|t}$

---

<sup>2</sup>For more detail regarding optimal prediction see e.g. (Goodwin and Sin 1984).

as follows.

$$\begin{aligned}
\hat{x}_{t+2|t} &= A\hat{x}_{t+1|t} + Bu_{t+1|t}, \\
\hat{x}_{t+3|t} &= A\hat{x}_{t+2|t} + Bu_{t+2|t}, \\
&= A(A\hat{x}_{t+1|t} + Bu_{t+1|t}) + Bu_{t+2|t}, \quad (\text{substituting for } \hat{x}_{t+2|t}) \\
&= A^2\hat{x}_{t+1|t} + ABu_{t+1|t} + Bu_{t+2|t}, \\
&\vdots \\
\hat{x}_{t+j|t} &= A^{j-1}\hat{x}_{t+1|t} + \sum_{k=1}^{j-1} A^{j-k-1}Bu_{t+k|t}.
\end{aligned}$$

Now in terms of predicting the output, Equation (7) can be expanded in terms of the above expression for  $\hat{x}_{t+j|t}$ , which results in a series of equations that provide optimal output predictions. The key point to note is that each output prediction is a function of the initial state  $\hat{x}_{t+1|t}$  and future inputs  $u_{t+i|t}$  only.

$$\begin{aligned}
\hat{y}_{t+1|t} &= C\hat{x}_{t+1|t} + Du_{t+1|t}, \\
\hat{y}_{t+2|t} &= C\hat{x}_{t+2|t} + Du_{t+2|t}, \\
&= C(A\hat{x}_{t+1|t} + Bu_{t+1|t}) + Du_{t+2|t}, \quad (\text{substituting for } \hat{x}_{t+2|t}) \\
&= CA\hat{x}_{t+1|t} + CBu_{t+1|t} + Du_{t+2|t}, \\
&\vdots \\
\hat{y}_{t+j|t} &= CA^{j-1}\hat{x}_{t+1|t} + C\left(\sum_{k=1}^{j-1} A^{j-k-1}Bu_{t+k|t}\right) + Du_{t+j|t}.
\end{aligned}$$

This series of output prediction equations can be stated in an equivalent but more convenient manner using matrix vector notation. Let

$$Y_t \triangleq \begin{bmatrix} y_{t+1|t} \\ \vdots \\ y_{t+N|t} \end{bmatrix}, \quad U_t \triangleq \begin{bmatrix} u_{t+1|t} \\ \vdots \\ u_{t+N|t} \end{bmatrix},$$

and

$$\Lambda = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{N-1} \end{bmatrix}, \quad \Phi = \begin{bmatrix} D & & & & \\ CB & D & & & \\ CAB & CB & D & & \\ \vdots & & & \ddots & \\ CA^{N-2}B & \dots & \dots & CB & D \end{bmatrix}.$$

Then

$$Y_t = \Lambda\hat{x}_{t+1|t} + \Phi U_t. \quad (8)$$

## 4 Control Objective

MPC is really no different from many other forms of feedback control in that the overarching goal is to reject disturbances whilst tracking a reference signal and at the same time ensuring that input energy (used for actuation) is sensible. In many practical situations the actuators have some form of physical limitations which limit their authority for control, but inside these limits it is still

possible to use high energy actuation, which may be undesirable. All of these control objectives can be represented mathematically using an objective function with a restricted domain, i.e. the objective function obeys certain constraints (limits).

In terms of tracking a reference signal, denoted by  $r_t$ , a frequently employed approach is to penalise deviations of the output from this reference by including a term like  $\|y_t - r_t\|$  in the objective function. This has the dual effect of rejecting plant disturbances since such events usually manifest as a change in the measured plant output. On the other hand, penalising actuator movements has the effect of reducing input energy and can be catered for by including a term like  $\|u_t - u_{t-1}\|$  in the control objective function. Modelling physical (and desired) limitations for plant inputs and outputs can be expressed using constraints, e.g. valve limits can be modelled as requiring that  $a \leq u_t \leq b$  where  $a$  and  $b$  are lower and upper limits respectively. The whole point of having an objective function is to determine the best (optimal) control action according to this performance criterion.

These control objectives are also applicable in the long term control of the plant. As such, it seems reasonable to make a control decision based on an appropriate forecast of its effect. As such, MPC includes a prediction horizon  $N$  in order to “see around corners”. However, there is little point making long term forecasts based on a poor model and this very important subject is often overlooked or underplayed.

With all this in mind, an appropriate control objective function may be chosen as follows. Note that the objective is a function of the initial state  $\hat{x}_{t+1|t}$  (i.e. the estimated state one sample in the future) and future control moves  $U_t$ .

$$J(\hat{x}_{t+1|t}, U_t) \triangleq \frac{1}{2} \sum_{k=1}^N \|\hat{y}_{t+k|t} - r_{t+k}\|_Q^2 + \|u_{t+k|t} - u_{t+k-1|t}\|_S^2. \quad (9)$$

In the above the matrices  $Q$  and  $S$  are assumed to be symmetric and positive definite. The quadratic form  $\|y - r\|_Q^2 \triangleq (y - r)^T Q (y - r)$  provides a mechanism to allow different weightings on different outputs. Similarly for  $\|u_{t+k|t} - u_{t+k-1|t}\|_S^2 \triangleq (u_{t+k|t} - u_{t+k-1|t})^T S (u_{t+k|t} - u_{t+k-1|t})$ , which allows different penalties for different input moves. Furthermore,  $u_{t|t} \triangleq u_t$  which is the input applied at the current time, i.e. the input calculated during the previous period.

In the case where the reference trajectory is known in advance then  $r_{t+k}$  may well vary during the prediction horizon. In the absence of such information then it seems reasonable to assume that  $r_{t+k} = r_t$ , i.e. the same reference point is held throughout the prediction horizon.

The summation terms in (9) can be expanded to offer a more convenient quadratic objective function in terms of  $\hat{x}_{t+1|t}$  and  $U_t$ . Let

$$R_t = \begin{bmatrix} r_{t+1} \\ \vdots \\ r_{t+N} \end{bmatrix}.$$

Then

$$\begin{aligned} \frac{1}{2} \sum_{k=1}^N \|\hat{y}_{t+k|t} - r_{t+k}\|_Q^2 &= \frac{1}{2} \|Y_t - R_t\|_Q^2, \\ &= \frac{1}{2} U_t^T \Phi^T \bar{Q} \Phi U_t + U_t^T [\Phi^T \bar{Q} \Lambda \hat{x}_{t+1|t} - \Phi^T \bar{Q} R_t] + C_1. \end{aligned}$$

In the above  $C_1$  is a constant term that does not depend on  $U_t$  or  $\hat{x}_{t+1|t}$  and  $\bar{Q}$  is given by

$$\bar{Q} \triangleq \begin{bmatrix} Q & & & \\ & Q & & \\ & & \ddots & \\ & & & Q \end{bmatrix}$$

Now

$$\frac{1}{2} \sum_{k=1}^N \|u_{t+k|t} - u_{t+k-1|t}\|_S^2 = \frac{1}{2} U_t^T \bar{S} U_t - u_{t+1|t}^T S u_t + C_2.$$

In the above  $C_2$  is a constant term that does not depend on  $U_t$  and  $\bar{S}$  is given by

$$\bar{S} \triangleq \begin{bmatrix} 2S & -S & & & \\ -S & 2S & -S & & \\ & & \ddots & \ddots & \ddots \\ & & & -S & 2S & -S \\ & & & & -S & S \end{bmatrix}$$

Combining the above the objective function can be expressed as

$$J(\hat{x}_{t+1|t}, U_t) = \frac{1}{2} U_t^T H U_t + U_t^T f + C_3. \quad (10)$$

Here  $C_3$  is the combination of previous constant terms and may be safely ignored. The terms  $H$  and  $f$  are given by

$$H \triangleq \Phi^T \bar{Q} \Phi + \bar{S}, \quad (11)$$

$$f \triangleq \Gamma \begin{bmatrix} \hat{x}_{t+1|t} \\ R_t \end{bmatrix} - \begin{bmatrix} S u_t \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad (12)$$

$$\Gamma \triangleq [\Phi^T \bar{Q} \Lambda \quad -\Phi^T \bar{Q}]. \quad (13)$$

It is important to note that  $H$  depends on certain matrices, namely  $A, B, C, D, Q, S$ , that infrequently change (if ever). This means that  $H$  may, and should, be computed off-line. Furthermore, the size of this matrix is  $Nn_u \times Nn_u$ , but only half this matrix need be stored since it is symmetric by construction. On the other hand, only part of  $f$  can be computed off-line since  $\hat{x}_{t+1|t}$ ,  $R_t$  and  $u_t$  change often. Nevertheless,  $f$  can be computed using a matrix vector multiplication.

With the objective function defined, attention can be given to constraints. Physical limitations of the plant infer limitations on control and neglecting these limitations can have undesirable effects including unstable closed-loop behaviour. Constraints need not be directly related to physical limitations either, for example safety considerations often impose limitations on plant outputs (temperature, heights in tanks, flow-rates, pressure etc.).

Constraints enter MPC in a natural manner as side conditions on the optimal control action calculation. For example, actuator bounds can be modelled using the linear inequality

$$\begin{bmatrix} I \\ -I \end{bmatrix} U_t \leq \begin{bmatrix} b_u \\ -b_\ell \end{bmatrix}.$$



Figure 3 shows a block diagram of the new plant model which includes an output disturbance model<sup>4</sup>. The corresponding state-space system is given by

$$\begin{bmatrix} x_{t+1} \\ d_{t+1} \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x_t \\ d_t \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_t + \omega_t, \quad (15)$$

$$y_t = [C \quad I] \begin{bmatrix} x_t \\ d_t \end{bmatrix} + Du_t + \nu_t. \quad (16)$$

With such a model in place, zero offset tracking is now possible and the remainder of the previous discussion on MPC holds. However, there are particular simplifications in this special case that affect computational load.

Once the estimate  $\hat{d}_{t+1|t}$  has been calculated (in accordance with Equations (2)–(6)), then according to (15) the optimal estimate for the remainder of the prediction horizon is also  $\hat{d}_{t+1|t}$ . This implies that the output estimates over the entire prediction horizon have a constant offset term given by  $\hat{d}_{t+1|t}$ , hence for prediction purposes the disturbance part of (15) may be safely ignored provided  $\hat{d}_{t+1|t}$  is added to every output estimate. To go one step further, this constant term can be subsumed within the reference trajectory by replacing  $r_{t+k}$  with  $r_{t+k} - \hat{d}_{t+1|t}$  for  $k = 1, \dots, N$ .

## 6 MPC Algorithm

MPC can be split into offline and online calculations by noticing that  $H$  and a large proportion of  $f$  depend on system and weighting matrices that rarely change (if ever). In the context of these notes, offline calculations proceed as follows.

### PROCEDURE 1

Given a plant model in terms of the state-space system matrices  $A, B, C, D$ , compute the following:

1. Determine noise covariance matrices  $W, V$  and  $Z$  according to knowledge of noise signals, or more likely according to acceptable performance.
2. Calculate  $K$  according to Equations (4) and (5).
3. Choose the prediction horizon  $N$  and state and input weighting matrices  $Q$  and  $S$  according to acceptable performance.
4. Construct  $H$  and  $\Gamma$  according to (11) and (13) respectively.
5. Choose constraints  $L_{\text{eq}}, L_{\text{in}}, b_{\text{eq}}$  and  $b_{\text{in}}$  according to physical limitations and desired operating ranges.

Once the above procedure is complete, MPC can be implemented according to the following algorithm.

---

<sup>4</sup>Another possibility would see the disturbance on the plant input, which is more appropriate to the types of observed disturbances in certain cases.

## ALGORITHM 2

At each time interval  $t$ , complete the following tasks,

1. Apply the previously calculated control move  $u_t$  to the system (as calculated in Step 5 of the previous iteration).
2. Measure the system output  $y_t$  and if necessary input  $u_t$ .
3. Estimate the current state  $\hat{x}_{t+1|t}$  using the measured outputs and inputs according to (2).
4. Compute  $f$  according to (12).
5. Compute the next control move  $u_{t+1}$  by selecting the first control move from  $U_t^*$ , which is obtained by solving 14.

## 7 Simulation Example

A single-input-single-output simulation example is provided for illustration purposes in this section. Matlab code is provided in place of a detailed exposition of the simulation. A Brief description follows.

The simulation is based on the following state-space plant

$$\begin{aligned}x_{t+1} &= \begin{bmatrix} 0.9 & 0 \\ 0.9 & 0.9 \end{bmatrix} x_t + \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} u_t, \\ y_t &= [1 \ 1] x_t + \nu_t.\end{aligned}$$

The noise signal  $\nu_t$  is described in the Matlab code, but is essentially a low-pass filtered integrated white noise signal.

Figure 4 shows the simulation results, which, after the initial transient due to non-zero initial states, tracks a reference signal which changes at time 200 samples and again at time 600 samples. At time 300 samples a small output disturbance is added.

```
%-----  
% Clear Stuff  
%-----  
clear all; close all;  
  
%-----  
% Construct model  
%-----  
%Specify system here in state space form with A B C D  
Ts = 1.0; %Sample interval  
A=[0.9 0;0.9 0.9]; B=[0.1;0.2]; C=[1 1]; D=0;  
  
%Specify system dimensions  
n_in = size(B,2); %# inputs  
n_out = size(C,1); %# outputs  
n_states = size(A,1); %# states  
  
%-----
```

```

% Specify horizons
%-----
M = 10; %Control Horizon
N = M; %Prediction Horizon

%-----
% Initial Matrix Filling
%-----
Lambda = zeros((N+1)*n_out,n_states);
Phi = zeros((N+1)*n_out);
AB = zeros((N+1)*n_out,n_in);
AB(1:n_out,:) = D;
for i = 2:N+1,
    AB((i-1)*n_out+1:(i)*n_out,:) = C*A^(i-2)*B;
end;
for i = 1:M+1,
    Phi(1+(i-1)*n_out:end,1+(i-1)*n_in:(i)*n_in) = AB(1:(N-i+2)*n_out,:);
end;
Phi = sparse(Phi);
Lambda(1:n_out,:) = C;
for i = 2:N+1,
    Lambda(1+(i-1)*n_out:i*n_out,:) = C*A^(i-1);
end;

%-----
% Quadratic cost weighting Q (error) and R (control) generation.
%-----
r = diag([1e4*ones(1,n_in)]); %Weights on input moves (i.e. ||u(t)-u(t-1)||)
q = diag([ones(1,n_out)]); %Weights on output deviation from setpoint
Q = sparse(kron(eye(N+1),q));
R = kron(diag([2*ones(1,N),1]),r);
R = R + kron(diag([-ones(1,N)],-1),r);
R = R + kron(diag([-ones(1,N)],1),r);

%-----
% Generate the Hessian H
%-----
H = full(Phi'*Q*Phi) + R;

%-----
% Create an observer
%-----
Aa = blkdiag(A,eye(n_out)); Ba = [B;zeros(n_out,n_in)];
Ca = [C eye(n_out)]; Da = D;
Ga = diag([ones(1,n_states),ones(1,n_out)]);
Qa = 1*eye(n_states+n_out);
Ra = 1*eye(n_out);
L = dlqe(Aa,Ga,Ca,Qa,Ra);
L = Aa*L;

%-----
% Create plant constraints
%-----
lb = -0.1*ones(n_in,1); ub = 0.1*ones(n_in,1);

```

```

Lb = kron(ones((M+1),1),lb); Ub = kron(ones((M+1),1),ub);

%-----
% Set initial conditions
%-----
sim_length = 1000; %Simulation length in sec
T = sim_length/Ts; %number of samples in simulation
t = 0:Ts:sim_length-Ts; %time steps

%-----
% Set system initial conditions.....
%-----
x = zeros(n_states,T);
x(:,1) = -2*ones(n_states,1); %Plant state
xh = zeros(n_states+n_out,T); %Estimated state
u = zeros(n_in,T); %Input record
u_old = zeros(n_in,1);
y = zeros(n_out,T); %Output record
Aobs = Aa-L*Ca; %Generate observer state gain matrix
Bobs = Ba-L*Da; %Generate observer input gain matrix
fd = full([Phi'*Q*Lambda, Phi'*kron(ones(N+1,1),q)]);
opt=optimset('display','off');

%-----
% Generate reference signal
%-----
ref = ones(1,T); ref(201:600) = -ref(201:600);

%-----
% Generate noise signal
%-----
noise = 1e-2*cumsum(randn(T,n_out)')' + 1e-2*randn(T,n_out);
[bfilt,afilt] = butter(4,0.9);
noise = filter(bfilt,afilt,noise)';

%-----
% MPC Evolution
%-----
for i = 1:T,
    %SIMULATE REAL PLANT-----
    x(:,i+1) = A*x(:,i) + B*u(:,i);
    y(:,i) = C*x(:,i) + D*u(:,i) + noise(:,i);
    %Add disturbance
    if i>300, y(:,i)=y(:,i)+0.5*ones(n_out,1); end

    %CONTROLLER STARTS HERE-----
    %Observe state
    xh(:,i+1) = Aobs*xh(:,i) + Bobs*u(:,i) + L*y(:,i);
    %Solve dynamic optimisation problem
    f = fd*[xh(1:n_states,i+1);xh(n_states+1:end,i+1)-ref(:,i)];
    f(1:n_in) = f(1:n_in)-r*u_old;
    U = quadprog(H,f,[],[],[],[],Lb,Ub,[],opt);

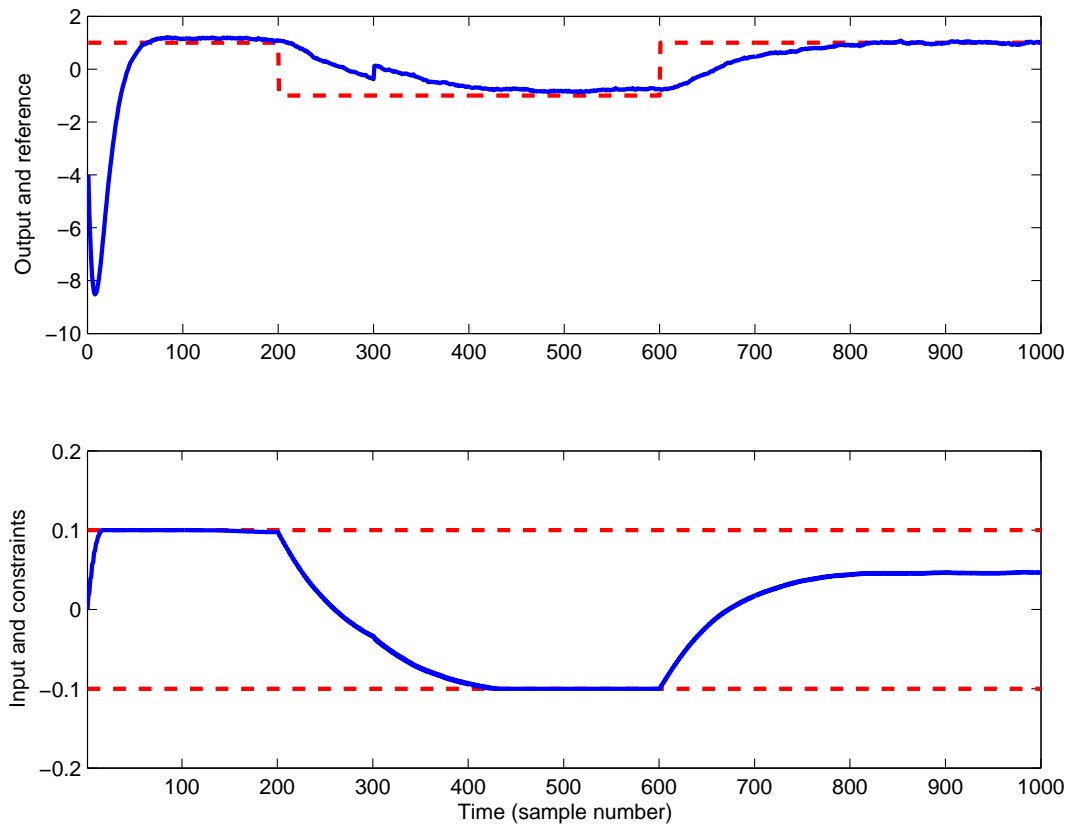
```

```

    u(:,i+1) = U(1:n_in); u_old = u(:,i);
end;

%-----
% Plots
%-----
subplot(2,1,1);
plot(ref','r--','Linewidth',2); hold on
plot(y','b-','Linewidth',2); hold off;
ylabel('Output and reference');
subplot(2,1,2);
plot([1 T],[ub ub],'r--','Linewidth',2); hold on;
plot([1 T],[lb lb],'r--','Linewidth',2);
plot(u(:,1:T)','b-','Linewidth',2); hold off;
xlabel('Time (sample number)');
ylabel('Input and constraints');
axis([1 T 2*lb 2*ub]);

```



**Figure 4:** Simulation results. Top plot: reference in red dashed and output in blue solid. Bottom plot: actuator bounds in red dashed and input in blue solid.

## References

- Benner, P., V. Mehrmann, V. Sima, S. Van Huffel and A. Varga (1997). SLICOT - A Subroutine Library in Systems and Control Theory. Technical report. NICONET Report 97-3.
- Bitmead, R. R., M. Gevers and V. Wertz (1990). *Adaptive Optimal Control - The Thinking Man's GPC*. Prentice Hall, Inc.. Englewood Cliffs, New Jersey.
- Chen, H. and F. Allgöwer (1998). A quasi-infinite nonlinear model predictive control scheme with guaranteed stability. *Automatica* **34**(10), 1205–1217.
- Clarke, D. W., C. Mohtadi and P. S. Tuffs (1987). Generalized predictive control, parts 1 and 2. *Automatica* **23**(2), 137–148.
- Cutler, C. R. and B. L. Ramaker (1980). Dynamic matrix control - a computer control algorithm. In: *Proceedings Joint American Control Conference*. San Francisco, CA.
- Cutler, D. W., A. Morshedi and J. Haydel (1983). An industrial perspective on advanced control. *AIChE Annual Meeting*.
- Garcia, C. E., D. M. Prett and M. Morari (1989). Model predictive control: Theory and practice - A survey. *Automatica* **25**(3), 335–348.
- Goodwin, G. C. and K. S. Sin (1984). *Adaptive filtering prediction and control*. Prentice-Hall Inc.. Englewood Cliffs, New Jersey.

- Kwakernaak, H. and R. Sivan (1972). *Linear Optimal Control Systems*. John Wiley & Sons, Inc.. New York.
- Maciejowski, J. M. (2002). *Predictive Control with Constraints*. Pearson Education Limited. Harlow, Essex.
- Mayne, D. Q., J. B. Rawlings, C. V. Rao and P. O. M. Scokaert (2000). Constrained model predictive control: Stability and optimality. *Automatica* **36**, 789–814.
- Muske, K. R. and J. B. Rawlings (1993). Model predictive control with linear models. *AIChE Journal* **39**(2), 262–287.
- Qin, S. Joe and T. A. Badgwell (1997). An overview of industrial model predictive control technology. *AIChE Symposium Series, 5th International Symposium on Chemical Process Control* **93**, 232–256.
- Richalet, J., A. Rault, J. L. Testud and J. Papon (1976). Algorithmic control of industrial processes. In: *Proceedings of Fourth IFAC symposium on identification and system parameter estimation*. pp. 1119–1167.
- Richalet, J., A. Rault, J. L. Testud and J. Papon (1978). Model predictive heuristic control: Applications to industrial processes. *Automatica* **14**, 413–428.