7

Continuous-time MPC with Constraints

7.1 Introduction

This chapter discusses continuous-time model predictive control with constraints. Section 7.2 formulates the constraints for the continuous-time predictive control system, including the cases of set-point following and disturbance rejection of constant signals. Section 7.3 presents the numerical solution of the constrained control problem with an example, where Hildreth's quadratic programming procedure is employed. Because of the nature of the continuous-time formulation such as fast sampling, there might be computational delays when the quadratic programming procedure is used in the solution of the real-time optimization problem. Section 7.4 discusses the real-time implementation of continuous-time model predictive control in the presence of constraints.

7.2 Formulation of the Constraints

7.2.1 Frequently Used Constraints

There are three types of constraints frequently encountered in continuous-time applications. The first two types deal with constraints imposed on the manipulated variables, and the third type of constraint deals with output or state variable constraints.

Constraints on the Manipulated Variable Rate of Change

In the context of continuous-time control, the rate of change on the control signal is given by the limit:

$$\lim_{\Delta t \to 0} \frac{\Delta u(t_i) - \Delta u(t_i - \Delta t)}{\Delta t} = \dot{u}(t) \mid_{t=t_i} . \tag{7.1}$$

We often use an approximation of the derivative for a given small Δt to calculate the upper and lower limits of the derivative. Assume that for a single-input system, the upper limit is du^{max} and the lower limit is du^{min}. Then, constraints are specified as

$$du^{min} \leq \dot{u}(t) \leq du^{max}. \tag{7.2}$$

Note that we use $\leq$, not $<$, which includes the case of $=$. This is because the optimal solution involves the active constraints, and they are the equality ones. If we have more than one input, say m inputs, then we can specify the constraints for each input independently. In the multi-input case, suppose that the constraints are given for the upper limit as

$$\left[du_1^{max} \; du_2^{max} \; \ldots \; du_m^{max} \right],$$

and lower limit as

$$\left[du_1^{min} \; du_2^{min} \; \ldots \; du_m^{min} \right].$$

We can specify each variable with rate of change, as

$$\begin{aligned} du_1^{min} \leq \; & \dot{u}_1(t) \; \leq du_1^{max} \\ du_2^{min} \leq \; & \dot{u}_2(t) \; \leq du_2^{max} \\ & \vdots \\ du_m^{min} \leq \; & \dot{u}_m(t) \leq du_m^{max}. \end{aligned} \tag{7.3}$$

The derivative constraints can be used to impose directions of movement on the manipulated variables. For instance, if $u_1(t)$ can only increase, and not decrease, then we select $0 \leq \dot{u}_1(t) \leq du_1^{max}$.

Constraints on the Amplitude of the Manipulated Variable

Suppose that the constraints are given for the upper limit of the control signal as

$$\left[u_1^{max} \; u_2^{max} \; \ldots \; u_m^{max} \right],$$

and lower limit as

$$\left[u_1^{min} \; u_2^{min} \; \ldots \; u_m^{min} \right].$$

Then, we specify the amplitude of each control signal to satisfy the constraints:

$$\begin{aligned} u_1^{min} \leq \; & u_1(t) \; \leq u_1^{max} \\ u_2^{min} \leq \; & u_2(t) \; \leq u_2^{max} \\ & \vdots \\ u_m^{min} \leq \; & u_m(t) \leq u_m^{max}. \end{aligned} \tag{7.4}$$

Output Constraints

We can also specify the operating range for the plant output. For instance, supposing that the output $y(t)$ has an upper limit y^{max} and a lower limit y^{min}, then the output constraints are specified as

$$y^{min} \leq y(t) \leq y^{max}. \tag{7.5}$$

Output constraints are often used in the regulation case where disturbance rejection is the primary focus for control. However, because output constraints could cause a predictive control system to become unstable, we need to be cautious when implementing output constraints. So they are often implemented as 'soft' constraints and a slack variable $s_v > 0$ is added to the constraints, forming

$$y^{min} - s_v \leq y(t) \leq y^{max} + s_v. \tag{7.6}$$

This means that they will not become active if the slack variable s_v is chosen large enough.

There are several reasons why we use a slack variable to form 'soft' constraints for outputs. One is that the output constraints often cause the problem of conflict constraints in the situation where the input constraints become activated. Also, when the predictive control system tries to alter the behaviour of a plant output, severe nonlinearity appears in the control law, which may result in closed-loop system oscillation, and system instability. We have observed this in the discrete-time case when output constraints are enforced (see Section 2.5.4 and Section 3.8.1). Similarly, this can occur in continuous-time control. When this happens, we relax the output constraints by selecting a larger slack variable s_v to resolve the problem so that the active constraints do not become active. Another reason is that the constraints are imposed based on the model and in the case where there is a model-plant mismatch, the prediction of the output may not be accurate. The constraints are not meaningful unless the model used for prediction is reasonably accurate.

Similarly, we can impose constraints on the state variables if they are measured or estimated. They also need to be in the form of 'soft' constraints for the same reasons as the output case.

7.2.2 Constraints as Part of the Optimal Solution

Having formulated the constraints as part of design requirements, the next step is to translate them into linear inequalities, and relate them to the original model predictive control problem. The key here is to parameterize the constrained variables using the same orthonormal basis functions as the ones used in the design of predictive control. Subsequently, we represent the constraints in terms of the parameter vector η. Since the predictive control problem is formulated and solved in the framework of receding horizon control, the

constraints are taken into consideration frame-by-frame for each moving horizon window. This allows us to vary the constraints at the beginning of each frame and also gives us the means to solve the constrained control problem numerically.

Constraints on the Derivative of Control

If we want to impose the constraints on the derivative of the control signal $\dot{u}(t)$ at time t_i, the constraints are expressed as

$$du^{min} \leq \dot{u}(t) \leq du^{max},$$

where du^{min} and du^{max} are the minimum and maximum limits of the derivative at time $t = t_i$. From the time instance t_i, the predictive control scheme looks into the future, and we need to express the future of the derivative of the control signal in terms of the Laguerre coefficient vector η. Differing from the discrete-time case, the future control trajectory in the continuous-time case is represented by continuous-time Laguerre functions. Thus, in order to obtain a finite set of linear inequalities for the constraints, the future time within the optimization window is discretized to obtain the time intervals $0, \tau_1, \tau_2, \ldots$ for which we wish to impose the constraints. The parameters τ_1, τ_2, ..., may not be related to the sampling interval Δt used for the implementation of the continuous-time predictive control, however, the first sample $\tau_0 = 0$ is always imposed to ensure that the constraints are satisfied when the receding horizon principle is applied. The implementation of the constraints on the first sample is performed as

$$-L(0)^T\eta \leq -du^{min} \tag{7.7}$$

$$L(0)^T\eta \leq du^{max}. \tag{7.8}$$

At an arbitrary future time τ_i, the derivative of the control signal is expressed as

$$du^{min} \leq \dot{u}(\tau_i) \leq du^{max}. \tag{7.9}$$

Note that

$$\dot{u}(\tau_i) = L(\tau_i)^T\eta,$$

where $L(\tau_i) = \left[l_1(\tau_i)\ l_2(\tau_i)\ \ldots\ l_N(\tau_i) \right]^T$ is the vector of the Laguerre functions and η is the parameter vector. Thus, the inequalities for the constraints at the time τ_i are

$$-L(\tau_i)^T\eta \leq -du^{min} \tag{7.10}$$

$$L(\tau_i)^T\eta \leq du^{max}. \tag{7.11}$$

The inequalities are readily extended to a multi-input system, as illustrated by the following example.

Example 7.1. Consider a continuous-time system with two inputs, u_1 and u_2. The constraints for the derivatives of the control signals are specified as

$$du_1^{min} \le \dot{u}_1(t) \le du_1^{max}; \quad du_2^{min} \le \dot{u}_2(t) \le du_2^{max}, \tag{7.12}$$

where $du_1^{min} = 0.5$, $du_1^{max} = 0.8$, $du_2^{min} = -0.3$, $du_1^{max} = 1$. Within the optimization window, the constraints are imposed on the first sample $\tau_0 = 0$ and $\tau_1 = 0.2$. Assuming $p_1 = 0.6$ and $p_2 = 1$; $N_1 = N_2 = 2$, formulate the inequalities for imposing the constraints in the solution of continuous-time predictive control.

Solution. In the design of predictive control, the two derivatives of the control are expressed as

$$\dot{u}_1(\tau) = L_1(\tau)^T \eta_1; \quad \dot{u}_2(\tau) = L_2(\tau)^T \eta_2. \tag{7.13}$$

Letting $\eta = \begin{bmatrix} \eta_1^T & \eta_2^T \end{bmatrix}^T$ with η_1 and η_2 having two coefficients each ($N_1 = N_2 = 2$), the inequalities corresponding to the constraints are

$$\begin{bmatrix} L_1(0)^T & o_2^T \\ o_1^T & L_2(0)^T \end{bmatrix} \eta \le \begin{bmatrix} du_1^{max} \\ du_2^{max} \end{bmatrix} \tag{7.14}$$

$$\begin{bmatrix} L_1(\tau_1)^T & o_2^T \\ o_1^T & L_2(\tau_1)^T \end{bmatrix} \eta \le \begin{bmatrix} du_1^{max} \\ du_2^{max} \end{bmatrix} \tag{7.15}$$

$$-\begin{bmatrix} L_1(0)^T & o_2^T \\ o_1^T & L_2(0)^T \end{bmatrix} \eta \le \begin{bmatrix} -du_1^{min} \\ -du_2^{min} \end{bmatrix} \tag{7.16}$$

$$-\begin{bmatrix} L_1(\tau_1)^T & o_2^T \\ o_1^T & L_2(\tau_1)^T \end{bmatrix} \eta \le \begin{bmatrix} -du_1^{min} \\ -du_2^{min} \end{bmatrix}, \tag{7.17}$$

where o_1 and o_2 are the zero vectors with their dimensions equal to those in $L_1(0)$ and $L_2(0)$, respectively. Numerically, these eight linear inequalities are shown as

$$\begin{bmatrix} 1.0954 & 1.0954 & 0 & 0 \\ 0 & 0 & 1.4142 & 1.4142 \\ 0.9716 & 0.7384 & 0 & 0 \\ 0 & 0 & 1.1579 & 0.6947 \\ -1.0954 & -1.0954 & 0 & 0 \\ 0 & 0 & -1.4142 & -1.4142 \\ -0.9716 & -0.7384 & 0 & 0 \\ 0 & 0 & -1.1579 & -0.6947 \end{bmatrix} \eta \le \begin{bmatrix} 0.8 \\ 1 \\ 0.8 \\ 1 \\ -0.5 \\ 0.3 \\ -0.5 \\ 0.3 \end{bmatrix}. \tag{7.18}$$

MATLAB tutorial: how to formulate the derivative constraints

Tutorial 7.1. *The purpose of this tutorial is to show how to formulate the derivative constraints using MATLAB.*

Step by Step

1. *Create a new file called Mder.m.*
2. *We only impose constraints on the first sample $\tau_0 = 0$ and a sample at τ in the future time. The program can be easily modified to include a larger number of constraints within the optimization window. The parameters p and N are the parameters for the Laguerre functions; n_{in} is the number of inputs and τ is the future sample of the constraints to be imposed. The first part of the program will formulate the constraints for the initial sample, so $L(0)$ will be generated and used. Because it is a multi-input system, attention is paid to the dimension of the matrix.*
3. *Enter the following program into the file:*

```
function [M_dv,Lzerot]=Mder(p,N,n_in,tau)
N_pa=sum(N);
k0=1;
[Al,L0]=lagc(p(k0),N(k0));
L_t=zeros(n_in,N_pa);
L_t(1,1:N(1))=L0';
cc=N(1);
for k0=2:n_in;
[Al,L0]=lagc(p(k0),N(k0));
L_t(k0,cc+1:cc+N(k0))=L0';
cc=cc+N(k0);
end
```

4. *The second part of the program is to formulate the constraints for the second sample at time τ.*
5. *Continue entering the following program into the file:*

```
Lzerot=L_t;
k0=1;
[Al,L0]=lagc(p(k0),N(k0));
L1=expm(Al*tau)*L0;
L_t1=zeros(n_in,N_pa);
L_t1(1,1:N(1))=L1';
cc=N(1);
for k0=2:n_in;
[Al,L0]=lagc(p(k0),N(k0));
L1=expm(Al*tau)*L0;
L_t1(k0,cc+1:cc+N(k0))=L1';
cc=cc+N(k0);
end
M_dv=[L_t1];
```

6. *Test the function with the following parameters*

```
p=[0.2 1];
```

```
N=[2 3];
n_in=2;
tau=0.3;
```

7. The results are

```
M_dv=   0.5956    0.5241         0         0         0
          0         0    0.7079    0.5805    0.4645

Lzerot= 0.6325    0.6325         0         0         0
          0         0    0.7746    0.7746    0.7746
```

Constraints on the Amplitude of Control

As for the control signal, assuming that Δt is the sampling interval for implementation, the first sample of the control signal at the optimization window, is calculated as

$$u(t_i) = u(t_i - \Delta t) + L(0)^T \eta \Delta t, \tag{7.19}$$

where $L(0)^T\eta$ is the derivative of control at the beginning of the optimization window. This leads to the constraints for the control signal at the first sample time of the window as

$$u^{min} - u(t_i - \Delta t) \leq L(0)^T \eta \Delta t \leq u^{max} - u(t_i - \Delta t), \tag{7.20}$$

which is in agreement with how the control is calculated using the velocity form. At the arbitrary time τ_i, we have

$$u(\tau_i) = u(t_i) + \int_0^{\tau_i} \dot{u}(\gamma) d\gamma \tag{7.21}$$

$$= u(t_i) + \int_0^{\tau_i} L(\gamma)^T \eta d\gamma \tag{7.22}$$

$$= u(t_i) + (L(\tau_i)^T - L(0)^T) A_p^{-T} \eta. \tag{7.23}$$

With the information of $u(t_i - \Delta t)$, the future control signal at time τ_i ($\neq 0$) is expressed as

$$u(\tau_i) = u(t_i - \Delta t) + \overbrace{(L(0)^T \Delta t + L(\tau_i)^T A_p^{-T} - L(0)^T A_p^{-T})}^{C_u} \eta. \tag{7.24}$$

Therefore, the inequality constraints are formulated as

$$u^{min} - u(t_i - \Delta t) \leq C_u \eta \leq u^{max} - u(t_i - \Delta t), \tag{7.25}$$

where u^{min} and u^{max} are the lower and upper limits of control signal u.

Note that the choice of the set of future time instants plays an important role in the numerical solution of the constrained control problem. In the

continuous-time case, the choice of τ_1, τ_2, ...,τ_m needs to be considered carefully. If they are selected too close to each other, then the constraints will be approximately equal to each other, which leads to redundancy among them. An appropriate choice will reduce the number of constraints.

Tutorial 7.2. *This tutorial shows how to produce the data matrix C_u for a multi-input system.*

Step by Step

1. *Create a new file called Mucon.m.*
2. *We only impose constraints on the first sample $\tau_0 = 0$ and a sample at τ in the future time. The program can be easily modified to include a larger number of constraints within the optimization window. The parameters p and N are the parameters for the Laguerre functions; n_{in} is the number of inputs and τ is the future time of the constraints to be imposed. The first part of the program will formulate the constraints for the initial sample, so $L(0)\Delta t$ will be generated and used. Because it is a multi-input system, attention is paid to the dimension of the matrix.*
3. *Enter the following program into the file:*

```
function [Mu,Mu1]=Mucon(p,N,n_in,delta_t,tau)
%function for generating matrix M for
%the constraints on the control signal
%constraints are imposed on the zero time and tau time
%delta-t is the sampling interval
%Mu is for constraints to be imposed on the zero sample
%Mu1 is for constraints to be imposed on tau time
N_pa=sum(N);
k0=1;
[Al,L0]=lagc(p(k0),N(k0));
L_t=zeros(n_in,N_pa);
L_t(1,1:N(1))=L0';
cc=N(1);
for k0=2:n_in;
[Al,L0]=lagc(p(k0),N(k0));
L_t(k0,cc+1:cc+N(k0))=L0';
cc=cc+N(k0);
end
% constraints on second sample
k0=1;
[Al,L0]=lagc(p(k0),N(k0));
L1=expm(Al*tau)*L0;
L_t1=zeros(n_in,N_pa);
L_t1(1,1:N(1))=(L1'-L0')*inv(Al')+L0'*delta_t;
cc=N(1);
```

```
for k0=2:n_in;
[Al,L0]=lagc(p(k0),N(k0));
L1=expm(Al*tau)*L0;
L_t1(k0,cc+1:cc+N(k0))=(L1'-L0')*inv(Al')+L0'*delta_t;
cc=cc+N(k0);
end
Mu=[L_t*delta_t];
Mu1=[L_t1];
```

4. Test the function using the following parameters:

```
p=[1  2];
N=[3  2];
n_in=2;
tau=1;
delta_t=0.1;
```

5. The results are

```
Mu= 0.1414    0.1414    0.1414         0         0
         0         0         0    0.2000    0.2000

Mu1=1.0354    0.2880   -0.0051         0         0
         0         0         0    1.0647   -0.1233
```

7.3 Numerical Solutions for the Constrained Control Problem

Now, the predictive control problem with hard constraints imposed in the design becomes the problem of finding the optimal solution of the quadratic cost function:

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(t_i) + x(t_i)^T \int_0^{T_p} \mathrm{e}^{A^T \tau} Q \mathrm{e}^{A\tau} d\tau x(t_i), \tag{7.26}$$

where Ω and Ψ are given as

$$\Omega = \int_0^{T_p} \phi(\tau) Q \phi(\tau)^T d\tau + R_L \tag{7.27}$$

$$\Psi = \int_0^{T_p} \phi(\tau) Q \mathrm{e}^{A\tau} d\tau \tag{7.28}$$

$$\phi(\tau)^T = \int_0^{\tau} e^{A(\tau-\gamma)} B L(\gamma)^T d\gamma, \tag{7.29}$$

subject to the linear inequality constraints that are formed from the previous analysis. When set-point following is required in the predictive control, if it

is a piece-wise constant signal, the last block element corresponding to the output y will be the error signal between the actual and desired signals of the output.

Finding the numerical solution to the continuous-time predictive control with constraints is concerned with the problems of constrained minimization where the constraint functions are linear and the objective function is a positive definite quadratic function. Similar to the discrete-time case, because the constraint functions are expressed in the form of linear inequalities, in general, the solutions involve quadratic programming procedures as outlined in Chapter 2.

Tutorial 7.3. *In this tutorial, we will produce a MATLAB function that performs constrained control for a continuous-time system. The program is written for a MIMO system, and we only impose constraints on the first sample of the control signals (both derivative and amplitude). sp is the set-point signal, which has the same number of rows as the output and number of columns greater than or equal to the simulation time N_{sim}. The plant model is described by (A_p, B_p, C_p), and the augmented model is described by (A, B, C). Lzerot is the data matrix for reconstructing the derivative of the control from Laguerre functions. M is the data matrix for the inequality constraints ($M\eta \leq \gamma$) where η is the Laguerre parameter vector. h is the sampling interval. The initial conditions of the plant are specified by the data vectors xm, u, y.*

Step by Step

1. *Create a new file called cssimucon.m. We will set the initial conditions and the data vector γ. Enter the following program into the file:*

```
function [u1,y1,udot1,t]=
cssimucon(xm,u,y,sp,Ap,Bp,Cp,A,B,C,N_sim,Omega,Psi,
K_ob,Lzerot,h,M,u_max,u_min,Deltau_max,Deltau_min);
up=u;
gamma=[(u_max-up);(-u_min+up);Deltau_max;-Deltau_min];
[m1,n1]=size(Cp);
[n1,n_in]=size(Bp);
X_hat=zeros(n1+m1,1);
```

2. *In order to incorporate the set-point signal in the simulation, we define a vector Xsp, which has the same dimension as the observed state with all zero elements except the last m1 rows as the set-point signal. The state feedback variable Xf is defined and the constrained control problem is solved using the MATLAB function QPhild.m (see Section 2.4.4 in Chapter 2). With the optimized udot ($\dot{u}$), the control and observer state are updated. Continue entering the following program into the file:*

```
for kk=1:N_sim;
Xsp=[zeros(n1,1);sp(:,kk)];
```

```
Xf=X_hat-Xsp;
eta=QPhild(Omega,Psi*Xf,M,gamma);
udot=Lzerot*eta;
u=u+udot*h;
udot1(1:n_in,kk)=udot;
u1(1:n_in,kk)=u;
y1(1:m1,kk)=y;
X_hat=X_hat+(A*X_hat+K_ob*(y-C*X_hat))*h+B*udot*h;
```

3. *We update the plant state and output in the simulation. (If the predictive control were implemented on a real plant, then the control signal would be sent to the actuator and the output y would be the signal sensed in the plant operation.) Continue entering the following program into the file:*

```
xm=xm+(Ap*xm+Bp*u)*h;
y=Cp*xm;
up=u;
gamma=[(u_max-up);(-u_min+up);Deltau_max;-Deltau_min];
end
t=0:h:(N_sim-1)*h;
```

4. *Test your program using the data from Tutorial 7.4.*

Tutorial 7.4. *In this tutorial, we will present a case study of continuous-time model predictive control with constraints based on an industrial process.*

A sugar mill model was presented in Goodwin et al. (2000) for a case study of control system design. In the study, a single stage of a milling train was described by the following continuous-time transfer function model:

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} g_{11}(s) & g_{12}(s) \\ g_{21}(s) & g_{22}(s) \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}, \tag{7.30}$$

where the output y_1 is the mill torque, and y_2 is the buffer chute height; the input u_1 is the flap position and input u_2 is the turbine speed set-point. The four transfer functions are given as

$$g_{11}(s) = \frac{-5}{25s+1};\ g_{12}(s) = \frac{s^2 - 0.005s - 0.005}{s(s+1)};$$

$$g_{21}(s) = \frac{1}{25s+1}; g_{22}(s) = \frac{-0.0023}{s}.$$

This process has significant multivariable interaction, nonminimum-phase behaviour. Also, the plant contains integrators.

The design specifications for the model predictive control are $Q = C^T C$, $R = I$, $T_p = 100$, $N_1 = N_2 = 6$, $p_1 = p_2 = 0.6$; and the observer design specifications are $Q_1 = I$, and $R_1 = 0.2I$. A sampling interval $\Delta t = 0.03$ (sec) is suggested for this continuous-time system simulation.

The constraints are imposed on the derivatives and amplitudes of both u_1 and u_2, and we only implement them on the first sample of the signals. The simulation scenario assumes that with zero initial conditions, the output y_1 performs a positive unit set-point change, and output y_2 performs a negative unit set-point change, while the constraints are specified as, $-1 \leq u_1 \leq 0$; $-3 \leq u_2 \leq 3$; $-0.4 \leq \dot{u}_1$; $-0.4 \leq \dot{u}_2 \leq 4$.

Step by Step

1. *When the turbine speed set-point u_2 changes, the effect on the mill torque is almost instantaneous, and this is reflected on the model by having $g_{12}(s)$ as a proper transfer function (both numerator and denominator are second-order s-polynomials). The structure of this proper transfer function causes a problem in the design of predictive control, as we have assumed that there is no direct connection between the input and output to allow the computation of receding horizon control law. However, by adding a filter with a very small time constant to $g_{12}(s)$, we obtain the modified transfer function as:*

$$g_{12}(s) = \frac{s^2 - 0.005s - 0.005}{s(s+1)(0.1s+1)}.$$

   The modified $g_{12}(s)$ is a strictly proper transfer function, and there is no direct link between u_2 and y_1.
2. *Create a new MATLAB program called sugarmill.m. We will first define the plant and create the augmented state-space model. Enter the following program into the file:*

```
num11=-1;
den11=[25 1];
num12=[1 -0.005 -0.005];
den12=conv([1 0],[1 1]);
den12=conv(den12,[0.1 1]);
num21=1;
den21=[25 1];
num22=-0.0023;
den22=[1 0];
Gs=tf({num11 num12; num21 num22},
{den11 den12; den21 den22});
Gs1=ss(Gs,'min');
[Ap,Bp,Cp,Dp]=ssdata(Gs1);
[m1,n1]=size(Cp);
[n1,n_in]=size(Bp);
A=zeros(n1+m1,n1+m1);
A(1:n1,1:n1)=Ap;
A(n1+1:n1+m1,1:n1)=Cp;
B=zeros(n1+m1,n_in);
```

```
B(1:n1,:)=Bp;
C=zeros(m1,n1+m1);
C(:,n1+1:n1+m1)=eye(m1,m1);
```

3. *We enter the design parameters and compute Ω and Ψ. Continue entering the following program into the file:*

```
n=n1+m1;
Q=C'*C;
R=1*eye(2,2);
p1=0.6;
p2=0.6;
N1=6;
N2=6;
Tp1=100;
Tp2=100;
p=[p1 p2];
N=[N1 N2];
Tp=[Tp1 Tp2];
[Omega,Psi]=cmpc(A,B,p,N,Tp,Q,R);
```

4. *We design the observer. Continue entering the following program into the file:*

```
Q1=eye(n,n);
R1=0.2*eye(m1,m1);
K_ob=lqr(A',C',Q1,R1)';
```

5. *The initial conditions and the set-point signals are specified. Pay attention to the data structure of the set-point signal. Continue entering the following program into the file:*

```
xm=zeros(n1,1);
u=zeros(n_in,1);
y=zeros(m1,1);
h=0.03;
N_sim=8*1400;
sp1=ones(1,N_sim);
sp2=[zeros(1,N_sim/2) -ones(1,N_sim/2)];
sp=[sp1;sp2];  %set-point signal
```

6. *Define the constraints by calling 'Mder.m' and 'Mucon.m'. Since we only impose constraints on the first sample, the second sample of constraints is neglected (we choose $\tau = 0.1$ as an arbitrary choice). Continue entering the following program into the file:*

```
[Md,Lzerot]=Mder(p,N,n_in,0.1);
[Mu,Mu1]=Mucon(p,N,n_in,h,0.1);
M=[Mu;-Mu;Lzerot;-Lzerot];
```

```
u_max=[0;3];
u_min=[-1;-3];
Deltau_max=[0.4;0.4];
Deltau_min=[-0.4;-0.4];
```

7. *The constrained control simulation is performed using the function 'cssimucon.m'. Continue entering the following program into the file:*

```
[u1,y1,udot1,t]=cssimucon(xm,u,y,sp,Ap,Bp,Cp,A,B,C,N_sim,
Omega,Psi,K_ob,Lzerot,h,M,u_max,u_min,Deltau_max,
Deltau_min);
```

Figure 7.1 shows the constrained control results for this example. There are eight constraints in the case. So it is difficult to take a guess at the active constraints, and it is necessary to identify the active constraints in the optimization. We have done so by using Hildreth's quadratic programming algorithm. If we carefully examine the number of iterations required to identify the active constraints in the quadratic programming, then we find that the convergence rate of the Lagrange multipliers is very fast. In fact, it takes about two iterations to achieve the convergence, and this is due to the fact that the active constraints are linearly independent.

It is seen from Figure 7.1 that all the constraints are satisfied. By comparing with the unconstrained case, we see in Figure 7.1 that the response speed of the constrained control system is slightly slower. The Laguerre parameters p_1, p_2, N_1 and N_2 could be used as performance related tuning parameters. For instance, in this example, when we choose $p_1 = p_2 = 0.1$ and $N_1 = N_2 = 3$, the closed-loop control results are different from the previous case (see Figure 7.2). In particular, the overshoot and undershoot in y_1 are reduced, and also the constraints on the control signal u_2 become active in shorter time intervals.

7.4 Real-time Implementation of Continuous-time MPC

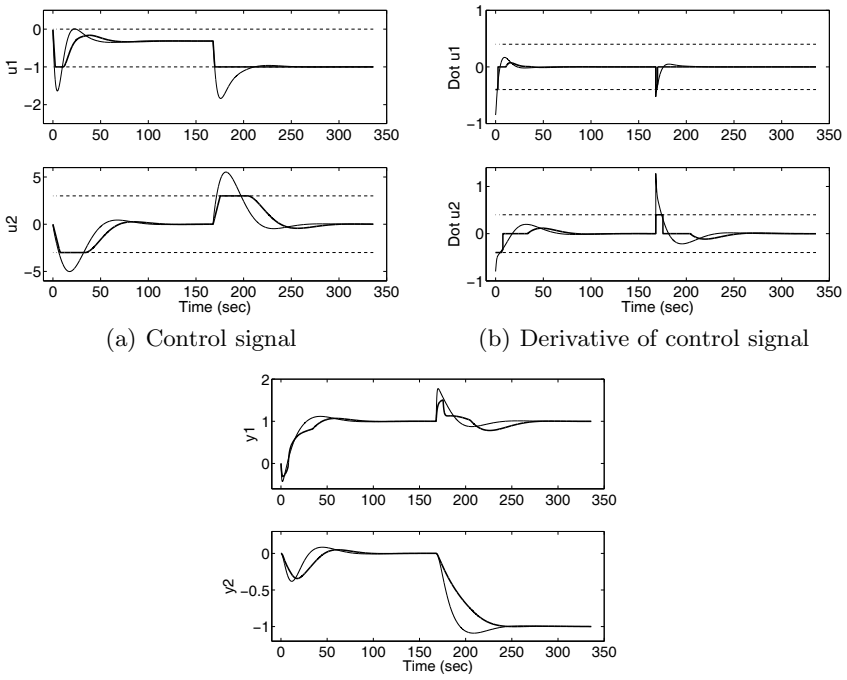
The essence of continuous-time model predictive control is to minimize the cost function:

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(t_i) + constant,$$

subject to the set of inequality constraints:

$$M\eta \leq \gamma.$$

This formulation in the continuous-time case is fundamentally identical to the one in the discrete-time case. One of the key advantages in using continuous-time predictive control instead of discrete-time predictive control is that the design model and the algorithm are robust in a fast sampling environment. The discrete-time models and predictive control algorithms could encounter



(a) Control signal

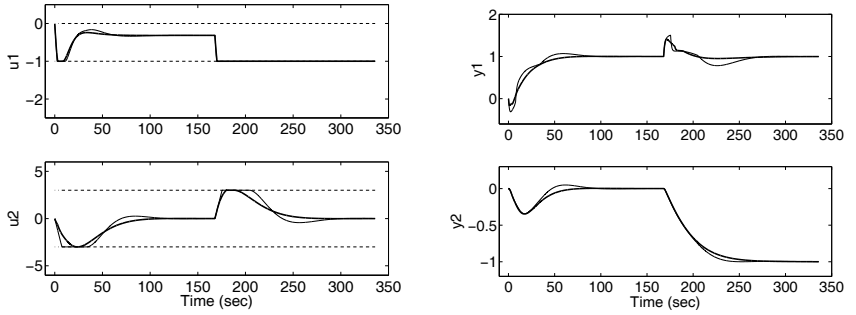
(b) Derivative of control signal

(c) Output

Fig. 7.1. Comparison of CMPC with and without constraints. Key: solid-line, without constraints; darker-solid-line, with constraints ($-1 \leq u_1 \leq 0$; $-3 \leq u_2 \leq 3$; $-0.4 \leq \dot{u}_1$; $-0.4 \leq \dot{u}_2 \leq 4$)

problems when the system is under a fast sampling condition. Another advantage is that because the design is performed in the continuous-time domain, the discretization is carried out in the implementation stage and the framework permits the control of an irregular sampled-data system.

Without imposing constraints, the computational requirement of minimizing the objective function J is negligible because the analytical expression is presented for the optimal solution with Ω and Ψ computed off-line. Therefore, implementing a continuous-time predictive control without constraints is not a challenging task. Also, if it is a single-input and single-output system, then the closed-form solution of the constrained control problem, introduced for discrete-time systems in Chapter 2, can be adapted to the applications in continuous-time. However, when introducing constraints for a multi-input and multi-output system, it is often necessary to use quadratic programming procedures to find the active constraints because the violated constraints may not be the active constraints. An optimal combination of the input signals needs to be found through the iterative search procedure (see Chapter 2). Therefore, there is a larger computational demand for finding the optimal



(a) Control signal (b) Output

Fig. 7.2. Comparison of CMPC with different performance parameters. Key: solid-line ($p_1 = p_2 = 0.6, N_1 = N_2 = 6$); darker-solid-line ($p_1 = p_2 = 0.1, N_1 = N_2 = 3$); ($-1 \leq u_1 \leq 0$; $-3 \leq u_2 \leq 3$; $-0.4 \leq \dot{u}_1$; $-0.4 \leq \dot{u}_2 \leq 4$)

solution of the constrained control problem for a MIMO system. Since the optimal solution cannot be found instantaneously, a computational delay will occur in the implementation of a continuous-time predictive control. If the computational rate is faster than the sampling rate in implementation, then the computational delay is not an issue. However, if the computational rate is slower than the sampling rate, then the computational delay must be addressed carefully.

Understanding the Problem of Computational Delay

As we understand, at time t_i the receding horizon control is performed within the optimization window $0 \leq \tau \leq T_p$. Closed-loop feedback is introduced through the initial condition of the state variable $x(t_i)$. Or in the general case, $x(t_i)$ is replaced by its estimate $\hat{x}(t_i)$ through an observer. The continuous-time observer is represented by the observer equation

$$\frac{d\hat{x}(t)}{dt} = A\hat{x}(t) + B\dot{u}(t) + K_{ob}(y(t) - C\hat{x}(t)).$$

With approximation for sufficiently small Δt, at time t_i

$$\frac{d\hat{x}(t)}{dt} \approx \frac{\hat{x}(t_i + \Delta t) - \hat{x}(t_i)}{\Delta t}.$$

Therefore, the estimated future state variable $\hat{x}(t_i + \Delta t)$ is calculated based on current values of $\hat{x}(t_i)$, $\dot{u}(t_i)$ and $y(t_i)$, using the formula,

$$\hat{x}(t_i + \Delta t) = (A\hat{x}(t_i) + B\dot{u}(t_i) + K_{ob}(y(t_i) - C\hat{x}(t_i)))\,\Delta t + \hat{x}(t_i). \quad (7.31)$$

With the estimated future state variable $\hat{x}(t_i + \Delta t)$ and the set-point information $r(t_i + \Delta t)$ as the new initial condition at $t_i + \Delta t$, the predictive control

algorithm optimizes J subject to the set of linear inequality constraints to find the new $\dot{u}(t_i + \Delta t)$. Therefore, the optimization is required to be completed within the allocated Δt time to avoid any computational delay. If the computational algorithm is sufficiently fast to produce $\dot{u}(t_i + \Delta t)$ on time when the real-time clock ticks at $t_i + \Delta t$, then there will not be a computational delay. However, if the computational speed is not matched with the fast sampling rate, then $\dot{u}(t_i + \Delta t)$ is produced with the time $\Delta t + \Delta c$, and Δc is the computational delay.

Strategy to Deal with Computational Delay

When computational delay occurs, the implementation clock time ticks at $t_i + \Delta t$ and the optimal coefficient vector η is not available. To distinguish the difference, let us call η_c the vector η computed using the information $\hat{x}(t_i + \Delta t)$ and $r(t_i + \Delta t)$, and η_p the vector η computed using the information $\hat{x}(t_i)$ and $r(t_i)$. A common strategy is to replace the optimal control derivative

$$\dot{u}(t_i + \Delta t) = L(0)^T \eta_c, \tag{7.32}$$

where η_c is still being computed, by the extended optimal control derivative from the previously computed η_p with the expression:

$$\dot{u}(t_i + \Delta t) = L(\Delta t)^T \eta_p. \tag{7.33}$$

This is not an optimal solution, but a sub-optimal solution. The difference is that $\dot{u}(t_i + \Delta t)$ used the estimated state variable $\hat{x}(t_i)$, instead of $\hat{x}(t_i + \Delta t)$. However, we argue that there are two scenarios related to this sub-optimal solution.

The first scenario is the case where there is no external set-point change or disturbance occurring at time t_i or during the time from t_i to $t_i + \Delta t$. Another important underlying assumption is that the model (A, B, C) is an accurate representation of the dynamic system. This statement actually means that the prediction generated by the model is sufficiently close to the actual output. If this happens, then

$$\hat{x}(t_i + \Delta t) = (A\hat{x}(t_i) + B\dot{u}(t_i))\,\Delta t + \hat{x}(t_i); \quad r(t_i + \Delta t) = r(t_i). \tag{7.34}$$

The predictive control has taken this prediction into account in its design, thus the optimal control at time $t_i + \Delta t$ is identical to $L(\Delta t)^T \eta_p$ and there is no compromise in the solution. We have demonstrated similar cases in the examples presented in the discrete-time case (see Chapters 3 and 4).

The second scenario is the case where there are external set-point changes or disturbances occurring at time t_i or during the time from t_i to $t_i + \Delta t$. Or the predicted response from the model is different from the actual system response. Then, the information presented in (7.34) is no longer accurate. There will be a difference between the solutions obtained from $L(0)^T \eta_c$ and

from $L(\Delta t)^T \eta_p$. The changes in the external signals could not be contained in the information $L(\Delta t)^T \eta_p$. Thus, the computational delay appears as a time delay in the measurement. We emphasize that if computational delay is anticipated, then the constraints need to be imposed on the time intervals where the extended control trajectory will be used. For instance, if we prepare to extend the solution at Δt, then the constraints will be imposed at least at 0 and Δt so to ensure that they are satisfied when computational delay occurs and η_p is used to replace the unavailable optimal solution η_c.

7.5 Summary

This chapter has discussed continuous-time predictive control with constraints. Similar to the discrete-time counterpart, the constraints in the continuous-time MPC are also placed on the current and future values of the control signal, and the derivative of the control signal. In addition, if desired, constraints on the future state variable and plant output are introduced. The constrained control problem in continuous time is formulated as a minimization of a quadratic cost function subject to linear inequality constraints, which is a quadratic programming problem. Similar to the discrete-time case, the coefficients of the Laguerre functions are the decision variables. In the solution, Hildreth's quadratic programming procedure is used to identify the active constraints, and subsequently the optimal decision variable η is found. When the set of active constraints can be correctly guessed, the decision variable η is obtained using the closed-form solution (see Chapter 2). This is particularly useful in the continuous-time case, as a fast sampling rate is often used to take advantage of the continuous-time setting.

One of the key differences between continuous-time and discrete-time MPC systems is that the orthonormal basis functions are in the continuous-time domain. Therefore, when introducing constraints on the future trajectories, a set of fixed nodes are chosen as the discretized time intervals for the constraints to be enforced upon. If the nodes were selected to be too close to each other, then constraints could be redundant. Therefore, a sensible choice needs to be made from application to application. In any case, the constraints are always enforced at the current time, and perhaps, one or two nodes are sufficient for the future time.

When the sampling rate in the implementation is faster than the computational rate dictated by the quadratic programming procedure, a computational delay occurs. A natural way to deal with the computational delay is to extend the optimal control trajectory obtained from previous computation to the current sample time, through the Laguerre functions and the previous optimal Laguerre coefficients. Like any other predictor-based approach, this strategy will work well if the modelling error between the plant and the model is small. An application of this approach to an inverted pendulum was pre-

sented in Gawthrop and Wang (2006). The approach was termed intermittent model predictive control in Gawthrop and Wang (2007).

Problems

7.1. A DC motor with speed as output variable and voltage as input variable has a first-order transfer function model given by

$$G(s) = \frac{1}{s+1}$$

Design a continuous-time predictive controller with constraints for this DC motor. Assuming zero initial conditions, the output response is required to follow a positive and a negative step input change where the constraints are specified as

$$-0.5 \leq \dot{u}(t) \leq 0.5; -1 \leq u(t) \leq 1.$$

The other performance parameters are $Q = C^T C$, $R = 1$, $N = 2$, $p = 0.5$, and the prediction horizon $T_p = 3$. Sampling interval $\Delta t = 0.01$.

1. Impose the constraints on the first sample of the signal and solve the constrained control problem analytically.
2. Simulate the constrained MPC system with zero initial conditions, and a positive unit step reference change at $t = 0$ and a negative unit step change at $t = 5$.
3. Show that without constraints the control signal can be written as components of proportional and integral controls,

$$u(t) = k_1 \int (r(\tau) - y(\tau)) d\tau - k_2 y(t),$$

where k_1 and k_2 are the gains of the predictive control system. Present schematically the configuration of the predictive control system, which is identical to a traditional proportional integral (PI) control system (see Astrom and Hagglund, 1995, Johnson and Moradi, 2005).

7.2. Assume that a second-order system has the continuous-time transfer function

$$G(s) = \frac{b_0}{s^2 + a_1 s + a_0}.$$

Verify that the augmented state-space model using $x_1(t) = \ddot{y}(t)$, $x_2(t) = \dot{y}(t)$ and $x_3(t) = y(t)$ has the following form:

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} -a_1 & -a_0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} b_0 \\ 0 \\ 0 \end{bmatrix} \dot{u}(t) \\ y(t) &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{7.35}$$

1. Show that a continuous-time MPC based on this model structure has the control signal in the form:

$$u(t) = k_1 \int (r(\tau) - y(\tau))d\tau - k_2 y(t) - k_3 \dot{y}(t),$$

which is equivalent to a proportional integral derivative (PID) controller.
2. Suppose that $a_1 = 1$, $a_0 = 0.1$ and $b_0 = 1$. Find the predictive controller parameters for $Q = C^T C$, $R = 0.1$, $p = 0.5$, $N = 2$, and $T_p = 20$.
3. Implement this predictive control system with constraints in a closed-form solution, assuming sampling interval $\Delta t = 0.01$, step reference input signal, zero initial conditions. The constraints are

$$-0.1 \leq \dot{u}(t) \leq 0.1; \quad -0.1 \leq u(t) \leq 0.1.$$

4. In the implementation of a PID controller (see Astrom and Hagglund, 1995), a derivative filter is often needed because of the presence of measurement noise. Similarly, you could use a derivative filter in the implementation here by using the relationship

$$k_3 s Y(s) \approx \frac{k_3 s}{\beta k_3 s + 1} Y(s),$$

where k_3 is the state feedback gain corresponding to $\ddot{y}(t)$, and β is 0.1 or less. Use of an observer is another option in the presence of measurement noise. Positioning observer poles at -2,-2.1 and -2.2, implement the predictive control system. Compare the implementations in the presence of measurement noise that is simulated using a sequence of zero-mean white noise with standard deviation of 0.1.

7.3. A distillation column is described by the transfer function model

$$G(s) = \begin{bmatrix} \frac{0.66e^{-2.6s}}{6.7s+1} & \frac{-0.51e^{-3s}}{8s+1} & \frac{-0.3e^{-s}}{10s+1} \\ \frac{1.5e^{-8s}}{4.1s+1} & \frac{-5e^{-2s}}{5.4s+1} & \frac{-1.3e^{-2s}}{9s+1} \\ \frac{-0.3e^{-11s}}{8s+1} & \frac{0.1e^{-10.8s}}{9s+1} & \frac{0.9e^{-2s}}{50s+1} \end{bmatrix}, \tag{7.36}$$

where typically the outputs are product concentration and product purity, and the inputs are the feed flow, temperature and vaporization rate. A case study of a distillation train was given in (Wang and Cluett, 2000) for the purpose of mathematical modelling.

1. Approximate the time delays in the transfer function model using the second-order Pade approximation where $e^{-ds} = \frac{(ds-4)^2}{(ds+4)^2}$, and find a minimal state-space realization (A_m, B_m, C_m) of the transfer function model using MATLAB functions 'tf' and 'ss' (see *e.g.*, Tutorial 3.3).

2. Design a continuous-time MPC system that will reject a constant input disturbance and follow a constant setpoint change without steady-state errors. The performance specifications are $Q = C^T C$, $R = 0.2I$ where I is the 3×3 identity matrix, $T_p = 50$. $N_1 = 3$, $N_2 = 3$ and $N_3 = 3$. $p_1 = 1/6$, $p_2 = 1/5$, $p_3 = 1/40$, which are close to the dominant poles of the diagonal elements in $G(s)$. The Laguerre scaling parameters should not be equal to the poles in the model as the inverse of matrix $A + pI$ is required in the solution of convolution integral equations (see section 6.3.3).
3. An observer is required in the implementation, where the observer gain K_{ob} is found by using the MATLAB lqr function with the pair A^T, C^T and the weight matrices are $Qob = I$ and $Rob = 0.1I$.
4. Simulate the nominal closed-loop performance without constraints, with zero initial conditions and a unit step reference input for y_2 at time 0 and a unit input disturbance added to $u_1(t)$ at time 60. The reference signals for y_1 and y_3 are zero. The sampling interval for implementation is 0.01.
5. It is relatively easy to build a SIMULINK simulation program when the plant has time delays. Writing the predictive control system as state feedback control $\dot{u}(t) = -K_{mpc}x(t)$, without constraints, simulate the predictive control system with the plant model that contains the time delays and compare the closed-loop responses with the nominal closed-loop responses.

7.4. Continue from Problem 7.3, where we will design and implement constrained MPC for the distillation column. The objective of the constraints is to maintain plant input and output within a desired operating region when performing set-point changes and rejecting disturbances originating from plant operations such as upstream feed flow changes. Assuming zero steady state for the system, introduce unit step change at $y_2(t)$ at time $t = 0$, then introduce unit step input disturbance at $u_1(t)$ at time $t = 30$. The operational constraints are specified as

$$-1.3 \le u_1 \le 0, \ -0.3 \le u_2 \le 0.1, \ -0.4 \le u_3 \le 0.1$$
$$-0.1 \le \dot{u}_1(t) \le 0.05, \ -0.1 \le \dot{u}_2(t) \le 0.1, \ -0.05 \le \dot{u}_3(t) \le 0.05.$$

Realize the constrained control using Hildreth's quadratic programming procedure. Impose the constraints on the first sample of the control signals in the first set of simulation experiments; then impose constraints on the first sample, $\tau_0 = 0$ and the second sample $\tau_1 = 1$.

7.5. Continue from Problem 7.4. The sampling interval for implementation of the constrained control is 0.01 in the simulation study. In the actual real-time implementation, the solution of the quadratic programming problem may require a longer computational time. Assume that the computational time on average is about 20 times the Δt used in the simulation. Revise the constrained predictive control scheme to cope with this computational delay using the technique introduced in Section 7.4. Compare the results obtained from using this predictive control to the results obtained in Problem 7.4.