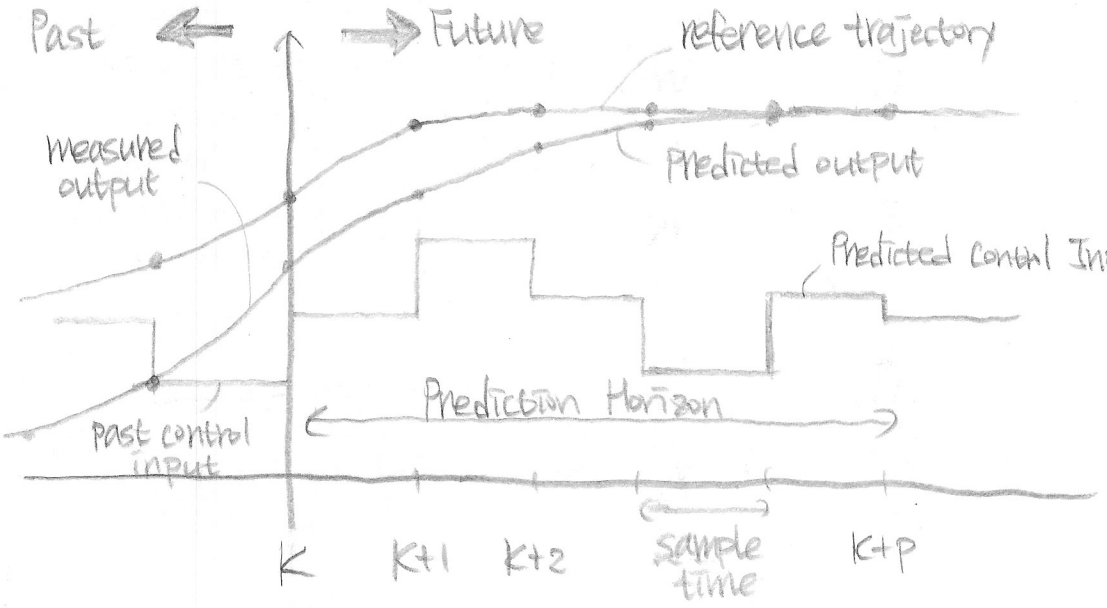
Model Predictive Control (MPC)

12/18/21 ~ 2/21/22

June Kwon

- MPC is an optimal control strategy on a moving horizon window based on the model of the system.
- MPC solves an online optimization problem at each time step



Key Issues

1. Model
2. Measurements
3. Realization of control

- Procedure for MPC

1. Within a given horizon window (prediction horizon)
2. Optimize & produce the optimal control input trajectory (control horizon) based on the current plant information (state)
3. Apply only the "1st" step of the optimal control input trajectory to the system.
4. System gets updated
5. Based on the updated states, move (advance) the horizon window, and re-optimize the system on a new horizon window and produce the optimal control input trajectory (and apply 1st step) 6. Repeat

1. Moving Horizon Window: the time dependent window from an arbitrary time t_i to $t_i + T_p$. Here, T_p, the length of window remains constant

   If $t_i = 0$, $T_p = 5$,

   1st Window: $[0, 5]$
   2nd Window: $[1, 6]$
   3rd Window: $[2, 7]$
   ⋮ ⋮

2. Prediction Horizon: dictates how far we wish the future to be predicted for. (= T_p : This parameter equals the length of the moving horizon window.) Thus, T_p

3. Receding Horizon Control: Although the optimal trajectory of future control signal is completely described within the moving horizon window, the actual control input to the plant only takes the first sample of the control signal, while neglecting the rest of trajectory

4. In the planning process, we need the information at time t_i in order to predict the future. This information is denoted as $x(t_i)$ which is a vector containing many relavent factors, and is either directly measured or estimated

5. A given model that will describe the dynamics of system is paramount in predictive control. A good dynamic model will give a consistent and accurate prediction of future.

6. In order to make the best decision, a criterion is needed to reflect the objective. The objective is related to an error function based on difference between the desired and the actual responses.

This objective function is often called the cost function, J, and the optimal control action is found by minimizing this cost function within the optimization window.

- Ok, let's begin. Given continuous model... we discretize it...

$$\begin{bmatrix} \text{continuous} \\ \dot{x}_m = A_c x_m + B_c u \\ y = C_c x_m \end{bmatrix} \Rightarrow \begin{bmatrix} \text{discrete} \\ x_m(k+1) = A_d x_m(k) + B_d u(k) \\ y(k) = C_d x_m(k) \end{bmatrix}$$

(* Generally, we have $y = C_c x_m + D_c u$. However, due to the principle of receding horizon control, where a current information of the plant is required for prediction and control, it is assumed that the input $u(k)$ cannot affect the output $y(k)$ at the same time. Thus, $D_c = 0$. in the plant model)

- Next, taking a difference operation on both sides of the discretized model,

$$\underbrace{x_m(k+1) - x_m(k)}_{\Delta x_m(k+1)} = A_d\underbrace{(x_m(k) - x_m(k-1))}_{\Delta x_m(k)} + B_d\underbrace{(u(k) - u(k-1))}_{\Delta u(k)}$$

Thus...

$$\boxed{\Delta x_m(k+1) = A_d \Delta x_m(k) + B_d \Delta u(k)}$$

y: Process Output
u: manipulated variable
x_m: state variable vector $(n_1 \times 1)$

$$n_1 \begin{bmatrix} x \\ x \\ x \end{bmatrix}$$
$$1$$

For output...

$$\begin{aligned} y(k+1) - y(k) &= C_d \Delta x_m(k+1) \\ &= C_d(A_d \Delta x_m(k) + B_d \Delta u(k)) \\ &= C_d A_d \Delta x_m(k) + C_d B_d \Delta u(k) \end{aligned}$$

Thus....

$$\boxed{y(k+1) = y(k) + C_d A_d \Delta x_m(k) + C_d B_d \Delta u(k)}$$

Next, set $x(k) = \begin{bmatrix} \Delta x_m(k)^T \\ y(k) \end{bmatrix}$

Then,

$$\underbrace{\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}}_{x(k+1)} = \underbrace{\begin{bmatrix} A_d & 0^T \\ C_d A_d & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}_{x(k)} + \underbrace{\begin{bmatrix} B_d \\ C_d B_d \end{bmatrix}}_{B} \Delta u(k)$$

$$y(k) = \underbrace{[0 \quad 1]}_{C}\underbrace{}_{n_1} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}$$

where $0 = \underbrace{[0\ 0 \ldots 0]}_{n_1}$

Thus, the "discrete-time augmented state-space" model is then ↑("Augmented State-space Models with Embedded Integrator.")

$$\begin{bmatrix} x(k+1) = Ax(k) + B\Delta M(k) \\ y(k) = Cx(k) \end{bmatrix}$$ ← The basic model to start MPC

where $A = \begin{bmatrix} A_d & 0 \\ C_d A_d & 1 \end{bmatrix}$, $B = \begin{bmatrix} B_d \\ C_d B_d \end{bmatrix}$, $C = \begin{bmatrix} 0 & 1 \end{bmatrix}$

, and $x(k) = \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}$

* Notice that $x(k)$ is not about state variable x. It is about variation in x ($\Delta x_m(k)$) with $y(k)$ (change)

Next, Prediction of State & Output

- N_p : Prediction Horizon (Number of Future Outputs that I want to predict)
  : length of optimization window
- N_c : Control Horizon (Number of Future Control Inputs that I want to predict.)
  (*Always $N_p \geq N_c$)

- Future control trajectory

$\Delta M(k_i)$
$\Delta M(k_i+1)$
$\Delta M(k_i+2)$
$\vdots$
$\Delta M(k_i+N_c-1)$ *

- Future State Variable

$x(k_i+1|k_i)$ ← where $x(k_i+m|k_i)$ is the predicted state variable at k_i+m with given current plants information $x(k_i)$
$x(k_i+2|k_i)$
$x(k_i+3|k_i)$
$\vdots$
$x(k_i+N_p|k_i)$ *

- Future State Variable, $x(k+1)$

$$x(k_i+1|k_i) = Ax(k_i) + B\Delta M(k_i)$$

$$x(k_i+2|k_i) = Ax(k_i+1) + B\Delta M(k_i+1)$$
$$= A^2x(k_i) + AB\Delta M(k_i) + B\Delta M(k_i+1)$$

$$x(k_i+3|k_i) = Ax(k_i+2) + B\Delta M(k_i+2)$$

$$\vdots$$

$$\boxed{x(k_i+N_p|k_i) = A^{N_p}x(k_i) + A^{N_p-1}B\Delta M(k_i) + A^{N_p-2}B\Delta M(k_i+1) + \cdots + A^{N_p-N_c}B\Delta M(k_i+N_c-1)}$$

- Future Output, $Y(k) = Cx(k)$

$$Y(k_i+1|k_i) = CAx(k_i) + CB\Delta M(k_i)$$

$$Y(k_i+2|k_i) = CAx(k_i+1) + CB\Delta M(k_i+1)$$
$$= CA^2x(k_i) + CAB\Delta M(k_i) + CB\Delta M(k_i+1)$$

$$Y(k_i+3|k_i) = CAx(k_i+2) + CB\Delta M(k_i+2)$$

$$\vdots$$

$$\boxed{Y(k_i+N_p|k_i) = CA^{N_p}x(k_i) + CA^{N_p-1}B\Delta M(k_i) + CA^{N_p-2}B\Delta M(k_i+1) + \cdots + CA^{N_p-N_c}B\Delta M(k_i+N_c-1)}$$

- Thus,

$$Y = \begin{bmatrix} y(k_i+1|k_i) \\ y(k_i+2|k_i) \\ y(k_i+3|k_i) \\ \vdots \\ y(k_i+N_p|k_i) \end{bmatrix} \Big\} N_p \qquad \Delta U = \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i+1) \\ \Delta u(k_i+2) \\ \vdots \\ \Delta u(k_i+N_c-1) \end{bmatrix} \Big\} N_c$$

- Thus, ~~together~~...

$$\boxed{Y = Fx(k_i) + \Phi \Delta U}$$

where

$$F = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_p} \end{bmatrix}$$

$$\Phi = \begin{bmatrix} CB & 0 & 0 & \cdots & 0 \\ CAB & CB & 0 & \cdots & 0 \\ CA^2B & CAB & CB & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \cdots & CA^{N_p-N_c}B \end{bmatrix}$$

↑ called "Toeplitz Matrix"

Optimization

→ Computing the right control input to achieve the control objective: Future output becomes the same as reference (desired trajectory, set-point). That is, the error between reference and future output is minimized

→ Reference

$$R_s = \underbrace{[1 \;\; 1 \;\; 1 \;\; \dots \;\; 1]^T}_{\in \mathbb{R}^{N_p}} \overbrace{r(k_i)}$$

set-point signal at sample time k_i

↳ Notice the prediction horizon size

→ Objective Function (Cost Function)

$$J = \underbrace{(R_s - Y)}_{\text{Error}}^T (R_s - Y) + \underbrace{\Delta U^T \bar{R} \Delta U}_{\text{Input Cost}}$$

and we would like to minimize this cost function (error, input cost)

Where

$$\bar{R} = \begin{bmatrix} r_w & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & r_w \end{bmatrix} \begin{matrix} \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ N_c \end{matrix}$$

(N_c columns)

$$= r_w I_{N_c \times N_c}$$

$$(r_w \geq 0)$$

↳ Tuning parameter for the desired closed-loop performance

$\bar{R}$ is a weighting matrix,

- If r_w is big, $\Delta U^T \bar{R} \Delta U$ will dominate in the cost function, thus in minimizing J, the small control input will be spent.
- If r_w is small, $\Delta U^T \bar{R} \Delta U$ will not dominate in the cost function, thus in minJ, the large control input will be spent.

→ Thus, given the objective function,

$$J = (R_s - Y)^T (R_s - Y) + \Delta U^T \bar{R} \Delta U$$

⤵ Substitute $Y = Fx(k_i) + \Phi \Delta U$

$$J = (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - 2\Delta U^T \Phi^T (R_s - Fx(k_i)) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U$$

⤵ Partial differentiate J with respect to ΔU and set equal to 0

$$\frac{\partial J}{\partial \Delta U} = -2\Phi^T (R_s - Fx(k_i)) + 2(\Phi^T \Phi + \bar{R}) \Delta U = 0$$

⤵ Rearrange to find $\Delta U^{optimal}$ that minimizes J (ΔU that makes $\frac{\partial J}{\partial \Delta U} = 0$)

$$\left[\Delta U = \underbrace{(\Phi^T \Phi + \bar{R})^{-1}}_{\text{Notice weighting matrix } \bar{R} \text{ is in denominator of } \Delta U} \Phi^T (R_s - Fx(k_i)) \right]$$

Condition # (Also, called Hessian Matrix): Thus, the choice of right N_p and N_c is imperative. I cannot choose any N_p or N_c b/c it could cause inverse not exist.

Thus, if

- r is big → Small Control Input ΔU → (Reduced Control Performance & Increased Energy Efficiency)
- r is small → Large Control Input ΔU → (Increased Control Performance & Reduced Energy Efficiency)

* Note that R_s is a data vector that contains the set-point information, expressed as

$$R_s = \underbrace{[1\ 1\ 1\ \dots\ 1]^T}_{N_p} r(k_i) = \bar{R}_s\, r(k_i)$$

where

$$\bar{R}_s = \underbrace{[1\ 1\ 1\ \dots\ 1]^T}_{N_p}$$

→ Thus, the optimal solution of the control signal linked to the set-point signal $r(k_i)$ and the state variable $x(k_i)$ is..

$$\Delta U = \left(\Phi^T \Phi + \bar{R}\right)^{-1} \Phi^T \left(\bar{R}_s\, r(k_i) - F\, x(k_i)\right)$$

• EX1.1

→ Given a discrete-time model..

$$x_m(k+1) = A_m x_m(k) + B_m u(k)$$
$$y(k) = C_m x_m(k)$$

*where

$$A_m = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad B_m = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$
$$C_m = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

→ Find 1) augmented model (A,B,C)

2) eigenvalues of the system matrix, A, of the augmented model

→ 1) Augmented Model for the plant is...

$$x(k+1) = A x(k) + B \Delta u(k)$$
$$y(k) = C x(k)$$

where

$$A = \begin{matrix} n_1 \\ m_1 \end{matrix} \begin{bmatrix} \overbrace{A_m}^{n_1} & \overbrace{0_m^T}^{m_1} \\ C_m A_m & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}; \quad B = \begin{matrix} n_1 \\ m_1 \end{matrix} \begin{bmatrix} \overbrace{B_m}^{n\_in} \\ C_m B_m \end{bmatrix} = \begin{bmatrix} 0.5 \\ 1 \\ 0.5 \end{bmatrix}$$

$$C = \underbrace{\begin{bmatrix} 0_m & 1 \end{bmatrix}}_{n_1 \quad m_1} \} m_1 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

→ 2) characteristic equation of matrix A is given by,

$$\rho(\lambda) = \det(\lambda I - A) = \det\left(\begin{bmatrix} \lambda I - A_m & 0_m^T \\ -C_m A_m & \lambda - 1 \end{bmatrix} \right)$$

Embedded Integrator

$$= \underbrace{(\lambda - 1)} \det(\lambda I - A_m) = (\lambda - 1)^3 \rightarrow \text{3 eigenvalues at } \lambda = 1$$

"Among three eigenvalues at $\lambda = 1$, two are from the original integrator plant, and one is from augmentation of the plant model"

MPC - E1.1 : Augmented State-space Model

```
% June Kwon
% 1/15/2022

clc
clear
close all
%#ok<*ASGLU>
%#ok<*NASGU>
```

1. Augmented State-space Model of Given Dynamics

```
% Original Dynamics
Am = [1 1 ; 0 1], Bm = [0.5 ; 1], Cm = [1 0], Dm = zeros(1,1)
```

```
Am = 2×2
     1     1
     0     1
Bm = 2×1
    0.5000
    1.0000
Cm = 1×2
     1     0
Dm = 0
```

```
% Augemented State-space Model
[m1,n1] = size(Cm);
[n1,n_in] = size(Bm);
A = eye(n1+m1,n1+m1);
A(1:n1,1:n1) = Am;
A(n1+1:n1+m1,1:n1) = Cm*Am
```

```
A = 3×3
     1     1     0
     0     1     0
     1     1     1
```

```
B = zeros(n1+m1,n_in);
B(1:n1,:) = Bm;
B(n1+1:n1+m1,:) = Cm*Bm
```

```
B = 3×1
    0.5000
    1.0000
    0.5000
```

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

```
C = 1×3
     0     0     1
```

2. Eigenvalues of Augmented Model

```
E = eig(A)
```

```
E = 3×1
     1
     1
     1
```

The eigenvalues of the augmented model are located at $\lambda = 1, 1, 1$. As discussed in the paper, the two eigenvalues ($\lambda = 1, 1$) are from the original integrator plant, and the one ($\lambda = 1$) is from the augmentation of the plant model.

• Augmented State-Space Model (T1.1)

→ 1. Discrete System Model

$$A_c = \begin{bmatrix} 0 & 1 & 0 \\ 3 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}; \quad B_c = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}; \quad C_c = [0\ 1\ 0]; \quad D_c = 0;$$

```
Delta_t = 1;
[Ad, Bd, Cd, Dd] = c2dm(Ac, Bc, Cc, Dc, Delta_t)  % discretize
```

→ 2. Determine the dimension of the system

```
[m1, n1] = size(Cd);
[n1, n_in] = size(Bd);
```

$C_c = \underbrace{[0\ 1\ 0]}_{n_1}$ → m_1 rows

$B_c = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}$ → n_1 rows, n_{in} columns

n_1 : # of states
m_1 : # of output
n_{in} : # of input

→ 3. Augmented State-Space Model

```
Ae = eye(n1+m1, n1+m1);
Ae(1:n1, 1:n1) = Ad;
Ae(n1+1:n1+m1, 1:n1) = Cd*Ad;

Be = zeros(n1+m1, n_in);
Be(1:n1, :) = Bd
Be(n1+1:n1+m1, :) = Cd*Bd;

Ce = zeros(m1, n1+m1);
Ce(:, n1+1:n1+m1) = eye(m1, m1);
```

MPC - T1.1 : Augmented Design Model

```
% June Kwon
% 12/18/2021

clc
clear
close all
%#ok<*ASGLU>
```

1. Continuous to Discrete Time Domain

```
Ac = [0 1 0; 3 0 1; 0 1 0], ...
Bc = [1 1 3]', ...
Cc = [0 1 0], ...
Dc = zeros(1,1), ...
h = 1
```

```
Ac = 3×3
     0     1     0
     3     0     1
     0     1     0
Bc = 3×1
     1
     1
     3
Cc = 1×3
     0     1     0
Dc = 0
h = 1
```

```
[Ad,Bd,Cd,Dd] = c2dm(Ac,Bc,Cc,Dc,h)
```

```
Ad = 3×3
    3.0716    1.8134    0.6905
    5.4403    3.7622    1.8134
    2.0716    1.8134    1.6905
Bd = 3×1
    2.9107
    5.9567
    4.9107
Cd = 1×3
     0     1     0
Dd = 0
```

2. Augmented Discrete Time Model

```
[m1,n1] = size(Cd)
```

```
m1 = 1
n1 = 3
```

```
[n1,n_in] = size(Bd)
```

```
n1 = 3
n_in = 1
```

3. Augmented State-space Model

```
Ae = eye(n1+m1 , n1+m1)
```

```
Ae = 4×4
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

```
Ae(1:n1 , 1:n1) = Ad
```

```
Ae = 4×4
    3.0716    1.8134    0.6905         0
    5.4403    3.7622    1.8134         0
    2.0716    1.8134    1.6905         0
         0         0         0    1.0000
```

```
Ae(n1+1 : n1+m1 , 1 : n1) = Cd * Ad
```

```
Ae = 4×4
    3.0716    1.8134    0.6905         0
    5.4403    3.7622    1.8134         0
    2.0716    1.8134    1.6905         0
    5.4403    3.7622    1.8134    1.0000
```

```
Be = zeros(n1+m1 , n_in)
```

```
Be = 4×1
     0
     0
     0
     0
```

```
Be(1:n1 , :) = Bd
```

```
Be = 4×1
    2.9107
    5.9567
    4.9107
         0
```

```
Be(n1+1 : n1+m1 , :) = Cd * Bd
```

```
Be = 4×1
    2.9107
    5.9567
    4.9107
    5.9567
```

```
Ce = zeros(m1 , n1+m1)
```

```
Ce = 1×4
     0     0     0     0
```

```
Ce(: , n1+1 : n1+m1) = eye(m1,m1)
```

```
Ce = 1×4
     0     0     0     1
```

4. Final Augmented State-space Model

```
AugG = ss(Ae,Be,Ce,[])
```

```
AugG =

  A =
           x1      x2      x3      x4
   x1   3.072   1.813  0.6905       0
   x2    5.44   3.762   1.813       0
   x3   2.072   1.813   1.691       0
   x4    5.44   3.762   1.813       1

  B =
         u1
   x1  2.911
   x2  5.957
   x3  4.911
   x4  5.957

  C =
       x1  x2  x3  x4
   y1   0   0   0   1

  D =
       u1
   y1   0

Continuous-time state-space model.
```

• EX1.2

→ Given,

$$x_m(k+1) = a x_m(k) + b u(k)$$
$$y(k) = x_m(k)$$

where

$a = 0.8$ $\quad N_p = 10$

$b = 0.1$ $\quad N_c = 4$

→ Find

1. Augmented State-space Model
2. Find F and Φ and $\Phi^T\Phi$, $\Phi^T F$, $\Phi^T R_s$

(→ Assume

ⓐ $k_i = 10$, $r(k_i) = 1$ and $x(k_i) = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$)

3. Find the optimal solution ΔU for $r_w = 0$ and $r_w = 10$ and compare results.

• 1) Augmented State-space Model is given as..

From

$$x_m(k+1) = 0.8\, x_m(k) + 0.1\, u(k)$$
$$y(k) = x_m(k)$$

↕ we know..

$$x_m(k+1) = A_d x_m(k) + B_d u(k)$$
$$y(k) = C_d x_m(k)$$

$$\begin{bmatrix} A_d = a = 0.8 \\ B_d = b = 0.1 \\ C_d = 1 \end{bmatrix}$$

- Thus, augmented state-space model is then..

Set $x(k) = \begin{bmatrix} \Delta x_m(k)^T \\ y(k) \end{bmatrix}$

Then,

$$\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} A_d & 0^T \\ C_d A_d & 1 \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} B_d \\ C_d B_d \end{bmatrix} \Delta u(t)$$

$$y(k) = \begin{bmatrix} 0 & 1 \end{bmatrix} x(k)$$

Plugging in, we have..

$$x(k+1) = \begin{bmatrix} 0.8 & 0 \\ 0.8 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} \Delta u(t) = A\,x(k) + B\,\Delta u(t)$$

$$y(k) = \begin{bmatrix} 0 & 1 \end{bmatrix} x(k) = C\,x(k)$$

Ans to 1.)

- 2) Find F and Φ, and $\Phi^T\Phi$, $\Phi^T F$, $\Phi^T \bar{R}_s$

Previously... F and Φ are given by..

$$F = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{NP} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 0.8 & 0 \\ 0.8 & 1 \end{bmatrix}^1 \\ \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 0.8 & 0 \\ 0.8 & 1 \end{bmatrix}^2 \\ \vdots \\ \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 0.8 & 0 \\ 0.8 & 1 \end{bmatrix}^{10} \end{bmatrix} = \begin{bmatrix} 0.8 & 1 \\ 1.44 & 1 \\ 1.952 & 1 \\ 2.3616 & 1 \\ 2.6893 & 1 \\ 2.9514 & 1 \\ 3.1611 & 1 \\ 3.3289 & 1 \\ 3.4631 & 1 \\ 3.5705 & 1 \end{bmatrix} \checkmark$$

- Similarly for $\mathcal{E}$...

$$\mathcal{E} = \begin{bmatrix} CB & 0 & 0 & 0 \\ CAB & CB & \cdots & \vdots \\ CA^2B & CAB & CB & \ddots \\ CA^{Np-1}B & CA^{Np-2}B & \cdots & CA^{Np-Nc}B \end{bmatrix} = \begin{bmatrix} CB & 0 & 0 & 0 \\ CAB & CB & \cdots & \vdots \\ CA^2B & CAB & CB & \ddots \\ \vdots & \vdots & \ddots & \vdots \\ CA^9B & CA^8B & \cdots & CA^6B \end{bmatrix}$$

$$= \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0.18 & 0.1 & 0 & 0 \\ 0.244 & 0.18 & 0.1 & 0 \\ 0.2952 & 0.244 & 0.18 & 0.1 \\ 0.3362 & 0.2952 & 0.2440 & 0.18 \\ 0.3689 & 0.3362 & 0.2952 & 0.2440 \\ 0.3951 & 0.3689 & 0.3362 & 0.2952 \\ 0.4161 & 0.3951 & 0.3689 & 0.3362 \\ 0.4329 & 0.4161 & 0.3951 & 0.3689 \\ 0.4463 & 0.4329 & 0.4161 & 0.3951 \end{bmatrix} \checkmark$$

- Thus, accordingly...

$$\mathcal{E}^T\mathcal{E} = \begin{bmatrix} 1.1541 & 1.0407 & 0.9116 & 0.7726 \\ 1.0407 & 0.9549 & 0.8475 & 0.7259 \\ 0.9116 & 0.8475 & 0.7675 & 0.6674 \\ 0.7726 & 0.7259 & 0.6674 & 0.5943 \end{bmatrix} \checkmark$$

$$\Phi^T F = \begin{bmatrix} 9.2325 & 3.2147 \\ 8.3259 & 2.7684 \\ 7.2927 & 2.3355 \\ 6.1811 & 1.9194 \end{bmatrix}_{\checkmark} \qquad \Phi^T \bar{R}_s = \begin{bmatrix} 3.2147 \\ 2.7684 \\ 2.3355 \\ 1.9194 \end{bmatrix}_{\checkmark}$$

where $\bar{R}_s = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]^T$

We can note that the vector $\Phi^T \bar{R}_s$ is identical to the last column in the matrix $\Phi^T F$. → This is because the last column of F matrix is identical to $\bar{R}_s$.

- 3.) Find Optimal Solution ΔU for $r_w = 0$ and $r_w = 10$ and compare.

From assumption,

→ As $y = 0.2$ goes 1, we expect $\Delta x \to 0$

@ $k_i = 10$, $x(k_i) = \begin{bmatrix} \Delta x_m(k_i) \\ y(k_i) \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$, $r(k_i) = 1$

↳ reference signal

↳ We want y to follow $r(k_i)$!

$y = 0.2 \to 1$

For $r_w = 0$

$$\bar{R} = \begin{bmatrix} r_w & & \\ 0 & \ddots & \\ 0 & 0 & r_w \end{bmatrix}_{N_c} \Bigg|_{r_w = 0} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} = 0$$

(N_c)

$$\left(^{*} R_s = \bar{R}_s r(k_i)\right)$$

$$\Delta U = \left(\Phi^T \Phi + \bar{R}^{0}\right)^{-1} \Phi^T \left(R_s - F x(k_i)\right)$$

$$= \underbrace{\left(\Phi^T \Phi\right)^{-1}}_{\text{we need to make sure this is 'invertible'!}} \left(\Phi^T \bar{R}_s r(k_i) - \Phi^T F x(k_i)\right)$$

$r(k_i) \leftarrow 1$, $x(k_i) \leftarrow \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$

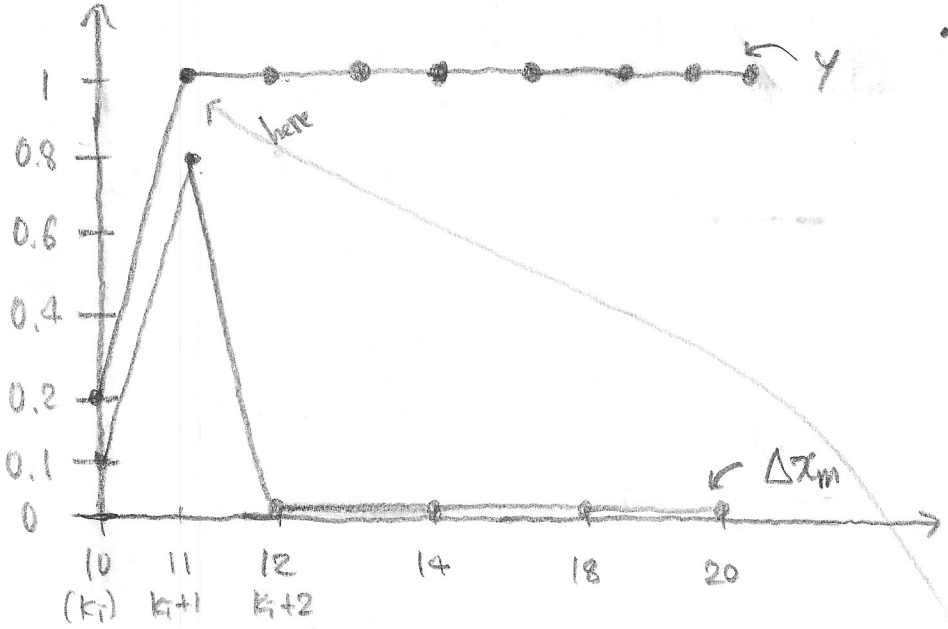
$\det(\Phi^T \Phi + \bar{R}) \neq 0$ Then, Invertible.

$$= \begin{bmatrix} 7.2 & -6.4 & 0 & 0 \end{bmatrix}^T$$

↳ 1st Control Input Changes $\Delta u(k_i)$

↳ 2nd $\Delta u(k_i + 1)$

- Let's plot system response with $r_w = 0$!



- As shown left, we can see the changes of state variables where predicted output y has reached the desired set-point $r(k_i)=1$ while Δx_m decays to 0.

$$Y = \begin{bmatrix} Y(k_i+1|k_i) \\ \vdots \\ Y(k_i+N_p|k_i) \end{bmatrix} = Fx(k_i) + \Phi\Delta U$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$$

- As shown above, the error between predicted Y and R_s is reduced without any consideration to the magnitude of control changes.

- Next,

for $r_w = 10$

$$\bar{R} = \begin{bmatrix} r_w & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & r_w \end{bmatrix}\Big|_{r_w=10} = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}_{N_c \times N_c} = I r_w$$

(I: Identity Matrix (4x4))

Thus,

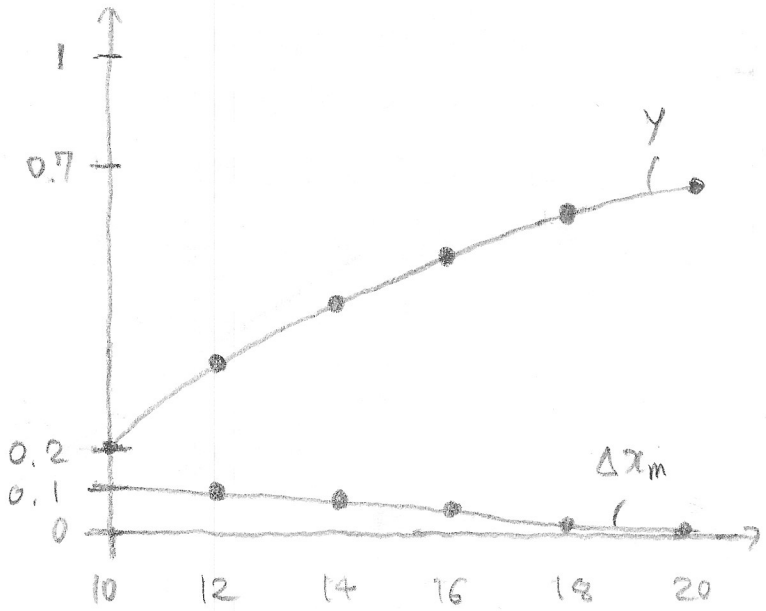
$$\Delta U = (\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i))$$

$$= (\Phi^T\Phi + r_w \times I_{N_c})^{-1}(\Phi^T \bar{R}_s r(k_i) - \Phi^T F x(k_i))$$

($r_w = 10$, $r(k_i) = 1$, $x(k_i) = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$)

$$= \begin{bmatrix} 0.1269 & 0.1034 & 0.0829 & 0.0650 \end{bmatrix}^T$$

- Again, plotting the system response with $r_w = 10$...



- With $r_w = 10$, the magnitude of first two control increments is significantly reduced, also the last two components are no longer zero.
- As shown left, output y did not reach the set-point value of 1, however, the Δx_m approaches zero.
- Big r_w → Small ΔU → Reduced Control Performance, Increased Energy Efficiency
- Small r_w → Big ΔU → Increased Control Performance, Reduced Energy Efficiency

- Thus, from comparison between $r_w = 0$ and $r_w = 10$,

  → If we want the control to move cautiously, then it takes longer for the control signal to reach its steady-state, (i.e. the values in ΔU decreases more slowly) because the optimal control energy is distributed over the longer period of future time

MPC - E1.2 : Discrete-time MPC for Beginners

```
% June Kwon
% 1/25/2022

clc
clear
close all
%#ok<*ASGLU>
%#ok<*NASGU>
```

1. Augmented State-space Model of Given Dynamics

```
% Original Dynamics
Ad = 0.8;
Bd = 0.1;
Cd = 1;
Dd = zeros(1,1);

% Augemented State-space Model
[m1,n1] = size(Cd);
[n1,n_in] = size(Bd);
A_e = eye(n1+m1,n1+m1);
A_e(1:n1,1:n1) = Ad;
A_e(n1+1:n1+m1,1:n1) = Cd*Ad
```

```
A_e = 2×2
    0.8000         0
    0.8000    1.0000
```

```
B_e = zeros(n1+m1,n_in);
B_e(1:n1,:) = Bd;
B_e(n1+1:n1+m1,:) = Cd*Bd
```

```
B_e = 2×1
    0.1000
    0.1000
```

```
C_e = zeros(m1,n1+m1);
C_e(:,n1+1:n1+m1) = eye(m1,m1)
```

```
C_e = 1×2
     0     1
```

```
% Prediction & Control Horizon Window
Np = 10
```

```
Np = 10
```

```
Nc = 4
```

```
Nc = 4
```

2. Computation of F and Φ, and $\Phi^T\Phi$, $\Phi^T F$, and $\Phi^T \overline{R_s}$

```
% Computation of F
n = n1 + m1;
h(1,:) = C_e;
F(1,:) = C_e * A_e;
for kk = 2:Np
    h(kk,:) = h(kk-1,:) * A_e;
    F(kk,:) = F(kk-1,:) * A_e;
end
F
```

```
F = 10×2
    0.8000    1.0000
    1.4400    1.0000
    1.9520    1.0000
    2.3616    1.0000
    2.6893    1.0000
    2.9514    1.0000
    3.1611    1.0000
    3.3289    1.0000
    3.4631    1.0000
    3.5705    1.0000
```

```
% Computation of Phi (P)
v = h*B_e;
P = zeros(Np,Nc);   % declare the dimension of Phi
P(:,1) = v;         % first column of Phi
for i = 2:Nc
    P(:,i) = [zeros(i-1,1) ; v(1:Np-i+1,1)]; % Toeplitz matrix
end
P
```

```
P = 10×4
    0.1000         0         0         0
    0.1800    0.1000         0         0
    0.2440    0.1800    0.1000         0
    0.2952    0.2440    0.1800    0.1000
    0.3362    0.2952    0.2440    0.1800
    0.3689    0.3362    0.2952    0.2440
    0.3951    0.3689    0.3362    0.2952
    0.4161    0.3951    0.3689    0.3362
    0.4329    0.4161    0.3951    0.3689
    0.4463    0.4329    0.4161    0.3951
```

```
% Phi' * Phi, Phi' * F, Phi' * Rbar_s
PP =  P' * P
```

```
PP = 4×4
    1.1541    1.0407    0.9116    0.7726
    1.0407    0.9549    0.8475    0.7259
    0.9116    0.8475    0.7675    0.6674
    0.7726    0.7259    0.6674    0.5943
```

```
PF = P' * F
```

```
PF = 4×2
    9.2325    3.2147
    8.3259    2.7684
    7.2927    2.3355
    6.1811    1.9194
```

```
Rbs = ones(Np,1)
```

```
Rbs = 10×1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
```

```
PR = P' * Rbs
```

```
PR = 4×1
    3.2147
    2.7684
    2.3355
    1.9194
```

3 - 1. Computation of Optimal Control Input Chage, ΔU , for $r_w = 0$

```
% Weighting Matrix
rw = 0
```

```
rw = 0
```

```
Rb = rw * eye(Nc,Nc)
```

```
Rb = 4×4
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

```
% Reference Signal
rki = 1
```

```
rki = 1
```

```
% Initial Condition (Starting Point)
xki = [0.1 ; 0.2]
```

```
xki = 2×1
    0.1000
    0.2000
```

```
% Optimal Control Input Change
DU = (PP + Rb) \ P' * (Rbs * rki - F * xki)
```

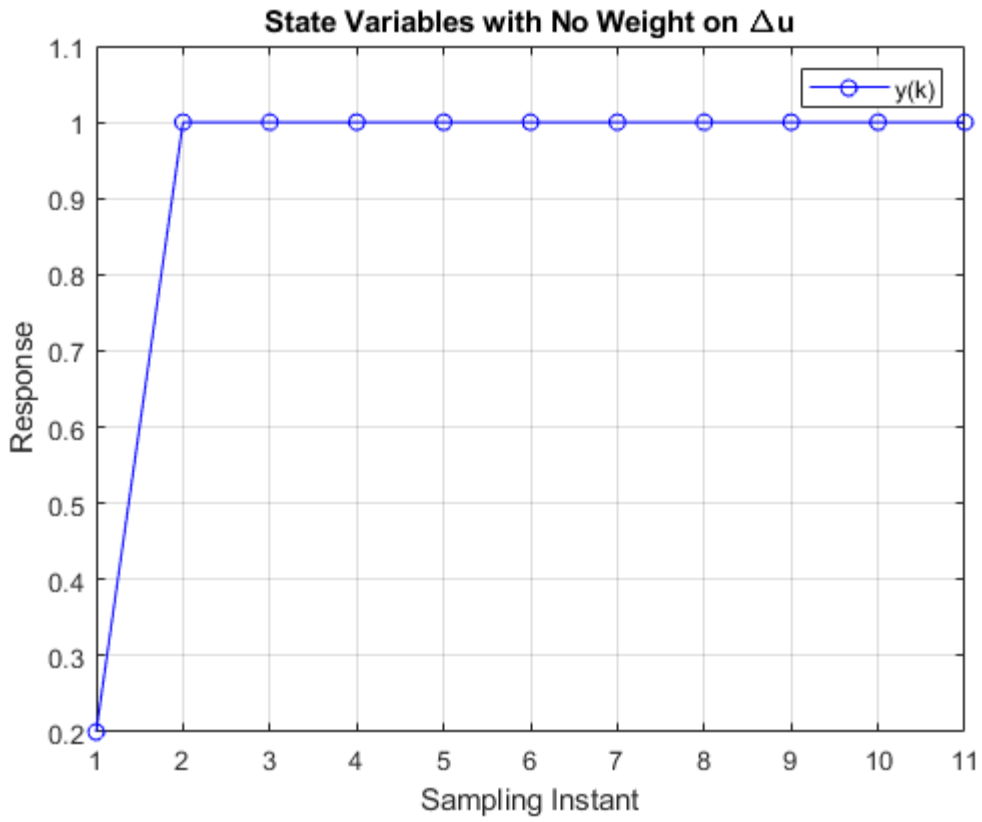
```
DU = 4×1
    7.2000
   -6.4000
    0.0000
   -0.0000
```

```
% Predicted Output from ki+1 to ki+Np
```

```
Y = F * xki + P * DU
```

```
Y = 10×1
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
```

```
% Change in State Variables (Not Finished)
DX1 = A_e * xki + B_e * DU(1);
DX2 = A_e * DX1 + B_e * DU(2);
DX3 = A_e * DX2 + B_e * DU(3);
DX4 = A_e * DX3 + B_e * DU(4);

% Plot
plot([xki(2) ; Y],'bo-')
grid on; legend('y(k)');
xlabel('Sampling Instant'); ylabel('Response');
title('State Variables with No Weight on \Deltau');
```



3 - 2. Computation of Optimal Control Input Chage, ΔU, for $r_w = 10$

```
% Weighting Matrix
```

```
rw = 10
```

```
rw = 10
```

```
Rb = rw * eye(Nc,Nc)
```

```
Rb = 4×4
    10     0     0     0
     0    10     0     0
     0     0    10     0
     0     0     0    10
```

```
% Reference Signal
rki = 1
```

```
rki = 1
```

```
% Initial Condition (Starting Point)
xki = [0.1 ; 0.2]
```

```
xki = 2×1
    0.1000
    0.2000
```

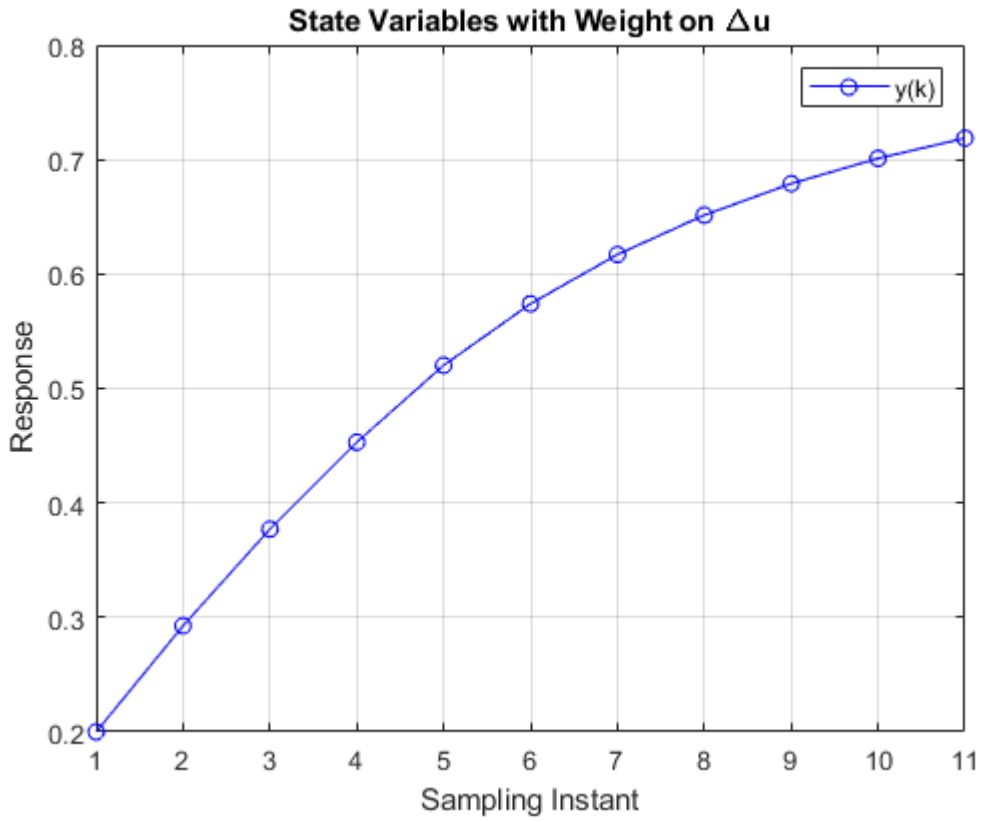
```
% Optimal Control Input Change
DU = (PP + Rb) \ P' * (Rbs * rki - F * xki)
```

```
DU = 4×1
    0.1269
    0.1034
    0.0829
    0.0650
```

```
% Predicted Output from ki+1 to ki+Np
Y = F * xki + P * DU
```

```
Y = 10×1
    0.2927
    0.3772
    0.4531
    0.5203
    0.5740
    0.6170
    0.6515
    0.6790
    0.7010
    0.7186
```

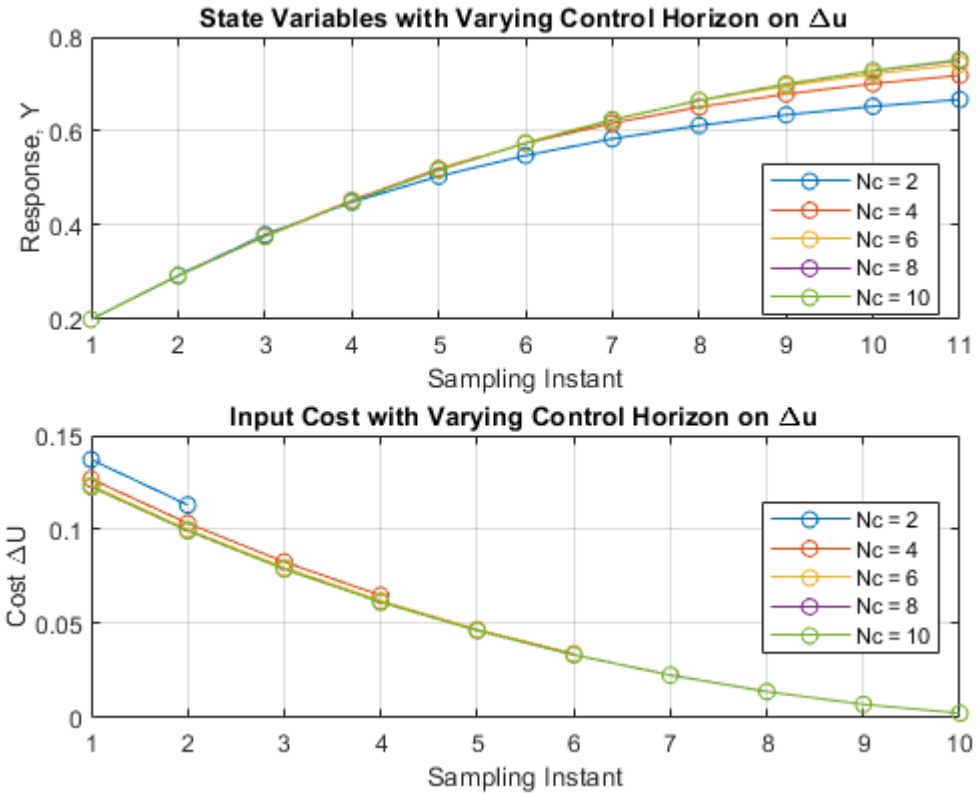
```
% Change in State Variables (Not Finished)
DX1 = A_e * xki + B_e * DU(1);
DX2 = A_e * DX1 + B_e * DU(2);
DX3 = A_e * DX2 + B_e * DU(3);
DX4 = A_e * DX3 + B_e * DU(4);

% Plot
plot([xki(2) ; Y],'bo-')
grid on; legend('y(k)');
xlabel('Sampling Instant'); ylabel('Response');
title('State Variables with Weight on \Deltau');
```



4 - 1. Effect of Varying Control Horizon (N_c)

```
figure;
for i = 1:5
    Nc = [2 4 6 8 10];
    Np = 10;

    [P,F,PP,PF,PR,Ae,Be,Ce] = mpcgain(Ad,Bd,Cd,Nc(i),Np);

    % Weighting Matrix
    rw = 10;
    Rb = rw * eye(Nc(i),Nc(i));
    % Reference Signal
    rki = 1;
    Rbs = ones(Np,1);
    % Initial Condition (Starting Point)
    xki = [0.1 ; 0.2];
    % Optimal Control Input Change
    DU = (PP + Rb) \ P' * (Rbs * rki - F * xki);
    % Predicted Output from ki+1 to ki+Np
    Y = F * xki + P * DU;

    % Plot
    hold on;
    subplot(2,1,1); plot([xki(2) ; Y],'o-')
end
grid on; legend('Nc = 2','Nc = 4','Nc = 6','Nc = 8','Nc = 10','location',"best");
```

```
xlabel('Sampling Instant'); ylabel('Response, Y'); hold off;
title('State Variables with Varying Control Horizon on \Deltau');

for i = 1:5
    Nc = [2 4 6 8 10];
    Np = 10;

    [P,F,PP,PF,PR,Ae,Be,Ce] = mpcgain(Ad,Bd,Cd,Nc(i),Np);

    % Weighting Matrix
    rw = 10;
    Rb = rw * eye(Nc(i),Nc(i));
    % Reference Signal
    rki = 1;
    Rbs = ones(Np,1);
    % Initial Condition (Starting Point)
    xki = [0.1 ; 0.2];
    % Optimal Control Input Change
    DU = (PP + Rb) \ P' * (Rbs * rki - F * xki);
    % Predicted Output from ki+1 to ki+Np
    Y = F * xki + P * DU;

    % Plot
    hold on;
    subplot(2,1,2); plot(DU,'o-')
end
grid on; legend('Nc = 2','Nc = 4','Nc = 6','Nc = 8','Nc = 10','location',"best");
xlabel('Sampling Instant'); ylabel('Cost \DeltaU'); hold off;
title('Input Cost with Varying Control Horizon on \Deltau');
```

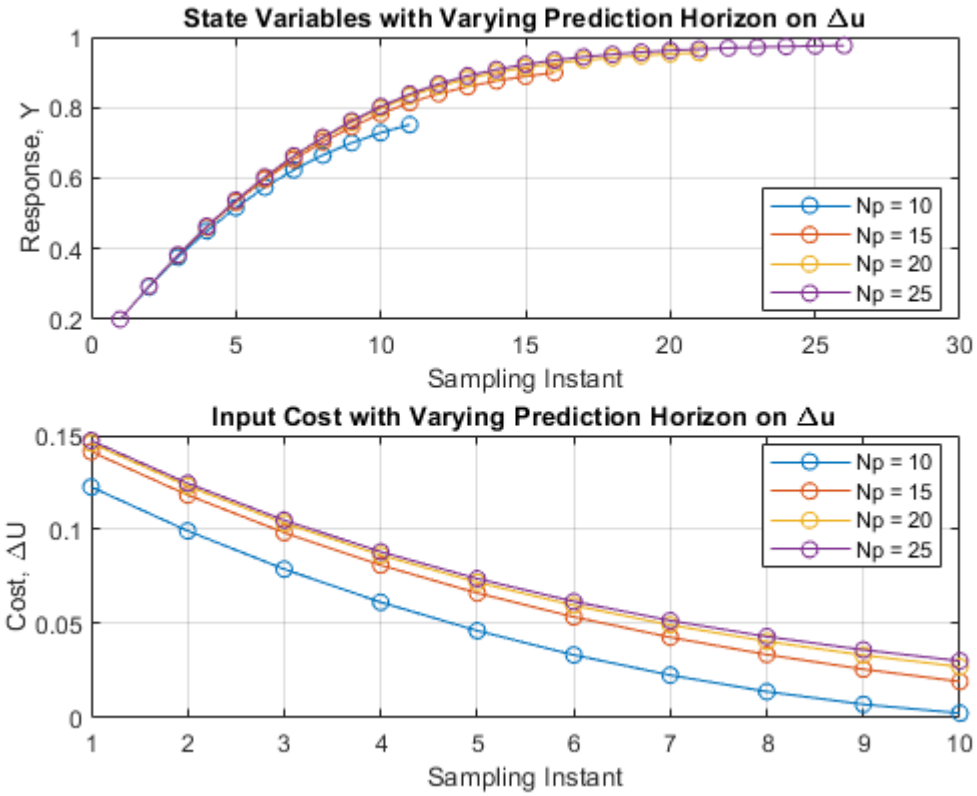


- **As shown above, as the control horizon (Nc) increases, the control performance is increased (becomes closer to the reference signal of 1), as well as the lower cost.**

4 - 2. Effect of Varying Prediction Horizon (N_P)

```
figure;
for i = 1:4
    Nc = 10;
    Np = [10 15 20 25];

    [P,F,PP,PF,PR,Ae,Be,Ce] = mpcgain(Ad,Bd,Cd,Nc,Np(i));

    % Weighting Matrix
    rw = 10;
    Rb = rw * eye(Nc,Nc);
    % Reference Signal
    rki = 1;
    Rbs = ones(Np(i),1);
    % Initial Condition (Starting Point)
    xki = [0.1 ; 0.2];
    % Optimal Control Input Change
    DU = (PP + Rb) \ P' * (Rbs * rki - F * xki);
    % Predicted Output from ki+1 to ki+Np
    Y = F * xki + P * DU;

    % Plot
```

```
    hold on;
    subplot(2,1,1); plot([xki(2) ; Y],'o-')
end
grid on; legend('Np = 10','Np = 15','Np = 20','Np = 25','location',"best");
xlabel('Sampling Instant'); ylabel('Response, Y'); hold off;
title('State Variables with Varying Prediction Horizon on \Deltau');

for i = 1:4
    Nc = 10;
    Np = [10 15 20 25];

    [P,F,PP,PF,PR,Ae,Be,Ce] = mpcgain(Ad,Bd,Cd,Nc,Np(i));

    % Weighting Matrix
    rw = 10;
    Rb = rw * eye(Nc,Nc);
    % Reference Signal
    rki = 1;
    Rbs = ones(Np(i),1);
    % Initial Condition (Starting Point)
    xki = [0.1 ; 0.2];
    % Optimal Control Input Change
    DU = (PP + Rb) \ P' * (Rbs * rki - F * xki);
    % Predicted Output from ki+1 to ki+Np
    Y = F * xki + P * DU;

    % Plot
    hold on;
    subplot(2,1,2); plot(DU,'o-')
end
grid on; legend('Np = 10','Np = 15','Np = 20','Np = 25','location',"best");
xlabel('Sampling Instant'); ylabel('Cost, \DeltaU'); hold off;
title('Input Cost with Varying Prediction Horizon on \Deltau');
```



- **As shown above, as prediction horizon increases, the control performance increases (becomes closer to the reference signal of 1), however, with the increased cost.**

• EX 1.3

→ Alternative way to find the minimum of the cost function via completing the squares. (The minimum of cost function becomes a by-product of approach)

→ From $J = (R_s - Fx(k_i))^T(R_s - Fx(k_i)) - 2\Delta U^T \Phi^T(R_s - Fx(k_i)) + \Delta U^T(\Phi^T\Phi + \bar{R})\Delta U$

by adding & subtracting $(R_s - Fx(k_i))^T \Phi(\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i))$,

→ That is,

$$J = (R_s - Fx(k_i))^T(R_s - Fx(k_i)) \underbrace{- 2\Delta U^T \Phi^T(R_s - Fx(k_i)) + \Delta U^T(\Phi^T\Phi + \bar{R})\Delta U}$$

$$\underbrace{+ (R_s - Fx(k_i))^T \Phi(\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i))}$$

$$- (R_s - Fx(k_i))^T \Phi(\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i)) = J$$

→ Where the quantities above ~ are completed squares...

$$\left[J_o = (\Delta U - (\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i))^T \times (\Phi^T\Phi + \bar{R})(\Delta U - (\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i)) \right] \text{"Completed Squares"}$$

→ Since the first term and the last term are independent of the variable ΔU (decision variable), and $(\Phi^T\Phi + \bar{R})$ is assumed to be positive definite, then a minimum of the cost function J is achieved if the quantity J_o equals zero.

→ Thus, set $J_o = 0$, and solving, we obtain "Optimal Control Solution"

$$\left[\Delta U = (\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i)) \right]$$ (same as what we had in previous pages...)

- By substituting this optimal solution into the cost function, we obtain the minimum of the cost as

$$J_{min} = (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - (R_s - Fx(k_i))^T \Phi (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (R_s - Fx(k_i))$$

MPC - T1.2 : Computation of MPC Gains

```
% June Kwon
% 1/25/2022

clc
clear
close all
%#ok<*ASGLU>
%#ok<*NASGU>
%#ok<*DEFNU>
```

1. MPC Gain Function

```
function [P,F,PP,PF,PR,Ae,Be,Ce] = mpcgain(Ap,Bp,Cp,Nc,Np)

    % Augemented State-space Model
    [m1,n1] = size(Cp);
    [n1,n_in] = size(Bp);
    Ae = eye(n1+m1,n1+m1);
    Ae(1:n1,1:n1) = Ap;
    Ae(n1+1:n1+m1,1:n1) = Cp * Ap;
    Be = zeros(n1+m1,n_in);
    Be(1:n1,:) = Bp;
    Be(n1+1:n1+m1,:) = Cp * Bp;
    Ce = zeros(m1,n1+m1);
    Ce(:,n1+1:n1+m1) = eye(m1,m1);

    % Computation of F
    n = n1 + m1;
    h(1,:) = Ce;
    F(1,:) = Ce * Ae;
    for kk = 2 : Np
        h(kk,:) = h(kk-1,:) * Ae;
        F(kk,:) = F(kk-1,:) * Ae;
    end

    % Computation of Phi (P)
    v = h * Be;
    P = zeros(Np,Nc);     % declare the dimension of Phi
    P(:,1) = v;           % first column of Phi
    for i = 2 : Nc
        P(:,i) = [zeros(i-1,1) ; v(1:Np-i+1,1)]; % Toeplitz matrix
    end

    BarRs = ones(Np,1);
    PP = P' * P;
    PF = P' * F;
    PR = P' * BarRs;

end
```

Receding Horizon Control

- Although the optimal parameter vector ΔU contains controls $\Delta u(k_i)$, $\Delta u(k_i+1)$, $\Delta u(k_i+2)$, ..., $\Delta u(k_i+N_c-1)$, with receding horizon control principle, we only implement the first sample of this sequence $\rightarrow \Delta u(k_i)$, while ignoring the rest of the sequence
- When the next sample period arrives, the more recent measurement is taken to form the state vector $x(k_i+1)$ for calculation of the new sequence of control signal. This procedure is repeated in real time to give the receding horizon control law.
- EX1.4 Perform Receding Horizon Control

→ Given: $x_m(k+1) = 0.8\,x_m(k) + 0.1\,u(k)$ * following EX1.2

where $r_w = 0$, $x(10) = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$, $u(9) = 0$

→ Previously, with $r_w = 0$ and same dynamics, we obtained

$$\Delta U = [\Delta u(k_i) \quad \Delta u(k_i+1) \quad \Delta u(k_i+2) \quad \Delta u(k_i+3)]^T$$
$$= [7.2 \quad -6.4 \quad 0 \quad 0]^T$$

Thus, we know first control input variation ($k_i = 10$)

$$\Delta u(k_i = 10) = 7.2$$

→ With $k_i = 10$, we know

$$\Delta u(10) = u(10) - u(9) = 7.2$$

Oh! we know $u(9) = 0$. Thus, $u(10)$ is...

$$u(10) = \Delta u(10) + u(9) = 7.2 + 0 = 7.2$$

[control signal to the plant at $k_i = 10$

→ Also, since same dynamics ... $y(k) = x_m(k)$ $\left(^*C = 1\right)$

Thus,

$$x_m(10) = y(10)$$

→ $\Delta x_m(10) = x_m(10) - x_m(9)$

And we know $y(10) = 0.2$ $\left(^*x(10) = \begin{bmatrix} \Delta x_m(10) \\ y(10) \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}\right)$

Thus,

$$x_m(10) = y(10) = 0.2$$

→ Thus! We can calculate the next simulated plant state variable

$$x_m(11) = 0.8\, x_m(10) + 0.1\, u(10)$$

$$= 0.8(0.2) + 0.1(7.2) = 0.88$$

→ At $k_i = 11$,

the new plant information is $\Delta x_m(11) = 0.88 - 0.2 = 0.68$

and $y(11) = x_m(11) = 0.88$. Thus,

$$x(11) = \begin{bmatrix} \Delta x_m(11) \\ y(11) \end{bmatrix} = \begin{bmatrix} 0.68 \\ 0.88 \end{bmatrix}$$

→ Thus, at $k_i = 11$, same optimization will be performed again in this new window.

Thus,

$$\Delta U = (\Phi^T \Phi)^{-1}(\Phi^T R_s - \Phi^T F x(11)) = \begin{bmatrix} -4.24 \\ -0.96 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i+1) \\ \Delta u(k_i+2) \\ \Delta u(k_i+3) \end{bmatrix}$$

→ This leads to the optimal control input at $k_i = 1$

$$u(11) = \Delta u(11) + u(10)$$

$$= -4.24 + 7.2 = \underline{2.96} \quad \star$$

This new control input is implemented in plant to obtain next state variable,

$$x_m(12) = 0.8\, x_m(11) + 0.1\, u(11)$$

$$= 0.8(0.88) + 0.1(2.96) = \underline{1} \quad \star$$

→ Thus, at $k_i = 12$, the new plant information is

$\Delta x_m(12) = x_m(12) - x_m(11) = 1 - 0.88 = 0.12$ and

$y(12) = x_m(12) = 1$

Then

$$x(12) = \begin{bmatrix} \Delta x_m(12) \\ y(12) \end{bmatrix} = \begin{bmatrix} 0.12 \\ 1 \end{bmatrix}$$

Initial Condition in new window $k_i = 12$

→ Then, again, same optimization in this new window ($k_T=12$) yields...

$$\Delta U = (\Phi^T\Phi)^{-1}(\Phi^T R_s - \Phi^T F\, x(12)) = \begin{bmatrix} -0.96 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \Delta u(k_T) \\ \Delta u(k_T+1) \\ \Delta u(k_T+2) \\ \Delta u(k_T+3) \end{bmatrix}$$

→ This leads to the control at $k_T=12$ as

$$u(12) = \Delta u(12) + u(11)$$

$$= -0.96 + 2.96 = 2$$

→ By implementing this control, we obtain the next plant output as

$$x_m(13) = 0.8\, x_m(12) + 0.1\, u(12)$$

$$= 0.8(1) + 0.1(2) = 1$$

→ Thus, new plant information at $k_T=13$ is...

$$\Delta x_m(13) = x_m(13) - x_m(12) = 1 - 1 = 0$$

$$y(13) = x_m(13) = 1$$

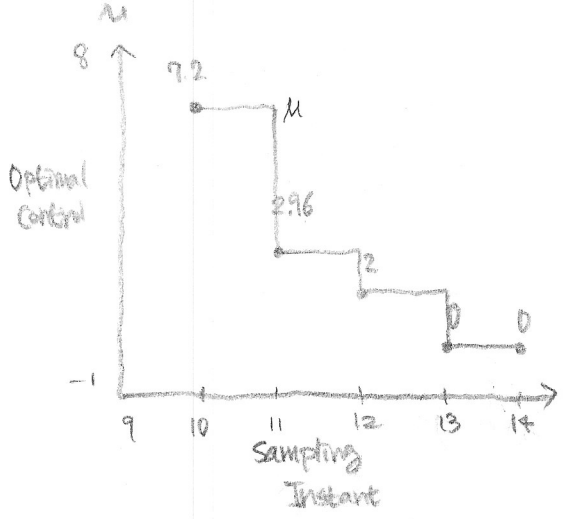
Then,

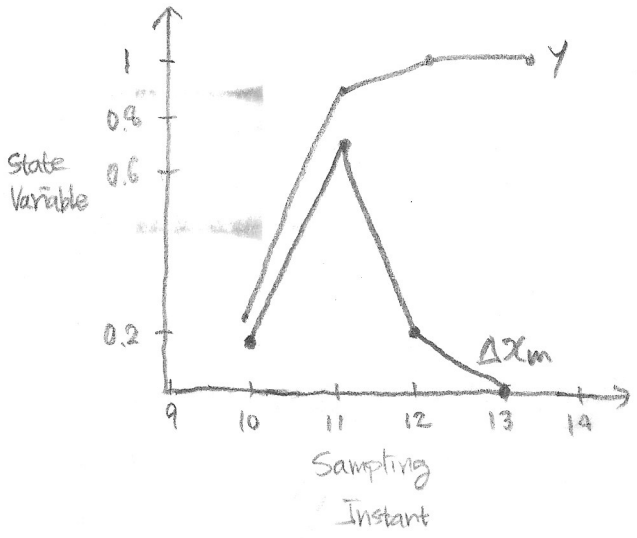
$$x(13) = \begin{bmatrix} \Delta x_m(13) \\ y(13) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Again, optimization in new widow $k_T=13$

$$\Delta U = (\Phi^T\Phi)^{-1}(\Phi^T R_s - \Phi^T F\, x(13)) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

→ Thus, plotting receding horizon control...





→ As the output response (y) reaches the disired set-point (1) signal, the parameters in ΔU approach zero.

For example,

$$\Delta U = \begin{bmatrix} 7.2 \\ -6.4 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} -4.24 \\ -0.96 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} -0.96 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

at $k_i = 10$ at $k_i = 11$ at $k_i = 12$ at $k_i = 13$

MPC - E1.4 : Receding Horizon Control

```
% June Kwon
% 2/19/2022

clc
clear
close all
%#ok<*ASGLU>
%#ok<*NASGU>
```

1. Augmented State-space Model of Given Dynamics

```
% Original Dynamics
Ad = 0.8;
Bd = 0.1;
Cd = 1;
Dd = zeros(1,1);

% Prediction Horizon & Control Horizon
Np = 10;
Nc = 4;

% Augmentization & Gain Computation
[P,F,PP,PF,PR,Ae,Be,Ce] = mpcgain(Ad,Bd,Cd,Nc,Np);

% Weighting Matrix
rw = 0;
Rb = rw * eye(Nc,Nc);

% Reference Signal
rki = 1;
Rbs = ones(Np,1);

% Initial Condition (at ki = 10)
xki = [0.1 ; 0.2];

% Pre-allocating
DU = zeros(4,4);                % Optimal Control Input Change
u  = zeros(1,4);                % Optimal Control Input
xm = zeros(1,4); xm(1) = xki(2); % State Variable
x  = zeros(2,4); x(:,1) = xki;   % Augmented State Variable
```

2. Computation of Optimal Control Input Change, ΔU, at $k_i = 10$

```
% Optimal Control Input Change at ki = 10
DU(:,1) = (PP + Rb) \ P' * (Rbs * rki - F * xki)
```

```
DU = 4×4
    7.2000         0         0         0
   -6.4000         0         0         0
    0.0000         0         0         0
```

```
   -0.0000         0         0         0
```

```
% Optimal Control Input at ki = 10
u(1) = DU(1,1) + 0
```

```
u = 1×4
    7.2000         0         0         0
```

```
% State Variable at ki = 10
xm(2) = Ad * xm(1) + Bd * u(1)
```

```
xm = 1×4
    0.2000    0.8800         0         0
```

```
% Augmented State Variable at ki = 10
% x = [delta_xm y]' where y(k) = xm(k)
x(1,2) = xm(2) - xm(1);
x(2,2) = xm(2)
```

```
x = 2×4
    0.1000    0.6800         0         0
    0.2000    0.8800         0         0
```

3. Computation of Optimal Control Input Change, ΔU, at $k_i = 11$

```
% Optimal Control Input Change at ki = 11
DU(:,2) = (PP + Rb) \ P' * (Rbs * rki - F * x(:,2))
```

```
DU = 4×4
    7.2000   -4.2400         0         0
   -6.4000   -0.9600         0         0
    0.0000   -0.0000         0         0
   -0.0000    0.0000         0         0
```

```
% Optimal Control Input at ki = 11
u(2) = DU(1,2) + u(1)
```

```
u = 1×4
    7.2000    2.9600         0         0
```

```
% State Variable at ki = 11
xm(3) = Ad * xm(2) + Bd * u(2)
```

```
xm = 1×4
    0.2000    0.8800    1.0000         0
```

```
% Augmented State Variable at ki = 11
% x = [delta_xm y]' where y(k) = xm(k)
x(1,3) = xm(3) - xm(2);
x(2,3) = xm(3)
```

```
x = 2×4
    0.1000    0.6800    0.1200         0
    0.2000    0.8800    1.0000         0
```

4. Computation of Optimal Control Input Change, ΔU, at $k_i = 12$

```
% Optimal Control Input Change at ki = 12
DU(:,3) = (PP + Rb) \ P' * (Rbs * rki - F * x(:,3))
```

```
DU = 4×4
    7.2000   -4.2400   -0.9600         0
   -6.4000   -0.9600   -0.0000         0
    0.0000   -0.0000   -0.0000         0
   -0.0000    0.0000    0.0000         0
```

```
% Optimal Control Input at ki = 12
u(3) = DU(1,3) + u(2)
```

```
u = 1×4
    7.2000    2.9600    2.0000         0
```

```
% State Variable at ki = 12
xm(4) = Ad * xm(3) + Bd * u(3)
```

```
xm = 1×4
    0.2000    0.8800    1.0000    1.0000
```

```
% Augmented State Variable at ki = 12
% x = [delta_xm y]' where y(k) = xm(k)
x(1,4) = xm(4) - xm(3);
x(2,4) = xm(4)
```

```
x = 2×4
    0.1000    0.6800    0.1200    0.0000
    0.2000    0.8800    1.0000    1.0000
```

5. Computation of Optimal Control Input Change, ΔU, at $k_i = 13$

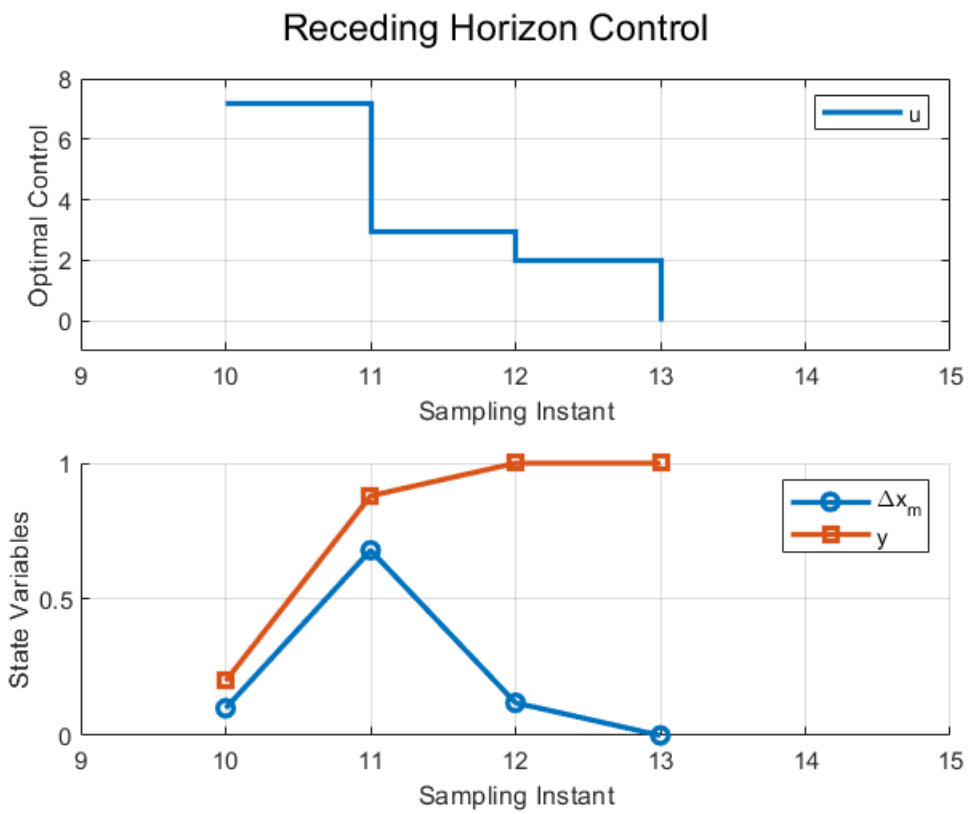
```
% Optimal Control Input Change at ki = 13
DU(:,4) = (PP + Rb) \ P' * (Rbs * rki - F * x(:,4))
```

```
DU = 4×4
    7.2000   -4.2400   -0.9600   -0.0000
   -6.4000   -0.9600   -0.0000   -0.0000
    0.0000   -0.0000   -0.0000    0.0000
   -0.0000    0.0000    0.0000   -0.0000
```

6. Plot

```
figure;
subplot(211)
stairs([10 11 12 13],u,'linewidth',2); legend('u');
xlim([9 15]); ylim([-1 8]);
xlabel('Sampling Instant'); ylabel('Optimal Control');
grid on;
subplot(212); hold on;
plot([10 11 12 13],x(1,:),'o-','linewidth',2);
plot([10 11 12 13],x(2,:),'s-','linewidth',2); legend('\Deltax_m','y');
xlim([9 15]); ylim([0 1]);
xlabel('Sampling Instant'); ylabel('State Variables');
grid on;
```

```
sgtitle('Receding Horizon Control');
```



Closed-loop Control System

→ We find that, at a given time k_i, the optimal parameter vector ΔU is solved using

$$\Delta U = (\bar{\Phi}^T\bar{\Phi} + \bar{R})^{-1}(\bar{\Phi}^T R_s - \bar{\Phi}^T F\, x(k_i))$$

$$= \underbrace{(\bar{\Phi}^T\bar{\Phi} + \bar{R})^{-1}\bar{\Phi}^T R_s}_{} - \underbrace{(\bar{\Phi}^T\bar{\Phi} + \bar{R})^{-1}\bar{\Phi}^T F}_{}\, x(k_i)$$

where.

$(\bar{\Phi}^T\bar{\Phi} + \bar{R})^{-1}\bar{\Phi}^T R_s$! corresponds to the set-point

$-(\bar{\Phi}^T\bar{\Phi} + \bar{R})^{-1}\bar{\Phi}^T F$! corresponds to the state feedback control within the framework of predictive control

→ Both depend on the system parameters, hence are constant matrices for a time invariant system

→ Because of the receding horizon control principle, we only take the first element of ΔU at time k_i as the incremental control. Thus,

$$\Delta u(k_i) = \overbrace{[1\ 0\ \dots\ 0]}^{N_c}(\bar{\Phi}^T\bar{\Phi} + \bar{R})^{-1}(\bar{\Phi}^T\bar{R}_s\, r(k_i) - \bar{\Phi}^T F\, x(k_i))$$

$$= K_y\, r(k_i) - K_{mpc}\, x(k_i)$$

where

K_y : 1st Element of $(\bar{\Phi}^T\bar{\Phi} + \bar{R})^{-1}\bar{\Phi}^T\bar{R}_s$

K_{mpc} : 1st Row of $(\bar{\Phi}^T\bar{\Phi}+\bar{R})^{-1}\bar{\Phi}^T\bar{F}$

→ Thus, $\Delta u(k_i) = k_y r(k_i) - K_{mpc}\, x(k_i)$, is in a standard form of linear time-invariant state feedback control

→ The state feedback control gain vector is K_{mpc}

→ Therefore, with augmented design model,

$$x(k+1) = Ax(k) + B\Delta u(k)$$

the closed-loop system is obtained by substituting $\Delta u(k_i) = k_y r(k_i) - K_{mpc}\, x(k_i)$ into the augmented system equation; changing index k_i to k, leading to the closed-loop equation

$$\begin{aligned} x(k+1) &= Ax(k) - BK_{mpc}\, x(k) + BK_y r(k) \\ &= \underbrace{(A - BK_{mpc})}_{*} x(k) + BK_y r(k) \end{aligned}$$

→ Thus, the closed-loop eigenvalues can be evaluated through the closed-loop characteristic equation

$$\det(\lambda I - (A - BK_{mpc})) = 0$$

(* Note $x(k) = \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}$)

→ Because of the special structures of the matrices C and A the last column of F is identical to $\bar{R}_s$, which is $[1\ 1 \dots 1]^T$, therefore K_y is identical to the last element of K_{mpc}.

→ Noting that the ~~state~~ variable vector $x(k_i) = [\Delta x_m(k)^T \ y(k)]^T$, and with the definition of K_y, we can write

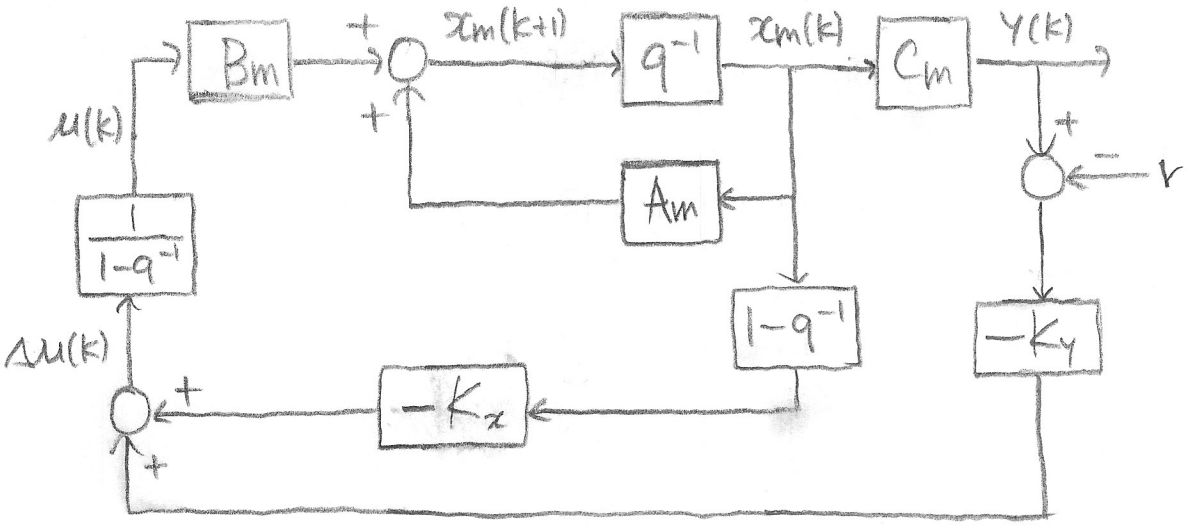
$$K_{mpc} = [K_x \ K_y]$$

where

K_x: corresponds to the feedback gain vector related to $\Delta x_m(k)$

K_y: corresponds to the feedback gain related to $y(k)$

→ Then, we obtain the closed-loop block diagram for predictive control system



where

q^{-1} : denotes the backward shift operator

→ The diagram shows the state feedback structure for the discrete model prediction control (DMPC) with integral action in which the module $\frac{1}{1-q^{-1}}$ denotes the discrete-time integrator ($\Delta M(k) \longrightarrow \boxed{\frac{1}{1-q^{-1}}} \longrightarrow M(k)$)

EX 1.5

→ Taking the previous example from EX 1.2, the eigenvalues of the closed-loop system with weight $r_w = 0$ and $r_w = 10$

→ For $r_w = 0$, we had...

$$\Delta U = (\bar{\Phi}^T \bar{\Phi} + \bar{R})^{-1} (\bar{\Phi}^T \bar{R}_s\, r(k_i) - \bar{\Phi}^T F\, x(k_i))$$

$$= \underbrace{\left[(\bar{\Phi}^T \bar{\Phi} + \bar{R})^{-1} \bar{\Phi}^T \bar{R}_s\right]}_{K_y} r(k_i) - \underbrace{\left[(\bar{\Phi}^T \bar{\Phi} + \bar{R})^{-1} \bar{\Phi}^T F\right]}_{K_{mpc}} x(k_i)$$

(1st Element; 1st Row)

$$= \begin{bmatrix} 7.2 \\ -6.4 \\ 0 \\ 0 \end{bmatrix} \leftarrow \Delta M(k_i)$$

→ Plugging the values we already have...

$$K_y = [1\ 0\ 0\ 0](\bar{\Phi}^T \bar{\Phi} + \bar{R})^{-1} (\bar{\Phi}^T \begin{bmatrix} 1 \\ \vdots \end{bmatrix}) = 10$$

$$K_{mpc} = [1\ 0\ 0\ 0](\bar{\Phi}^T \bar{\Phi} + \bar{R})^{-1} \bar{\Phi}^T F) = [8\ \ 10]$$

→ Hence, the eigenvalues of the closed-loop system are calculated,

$$\det\left(\lambda I - (A - B K_{mpc})\right) = 0$$

(*where $A = \begin{bmatrix} 0.8 & 0 \\ 0.8 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$)

↓

$$\lambda_1 = -6.409 \times 10^{-7}, \quad \lambda_2 = +6.409 \times 10^{-7}$$

→ Approximately on the origin of the complex plane...

→ For $r_w = 10$, we had, $\Delta U = [0.1269 \;\; 0.1034 \;\; 0.0829 \;\; 0.0650]^T$

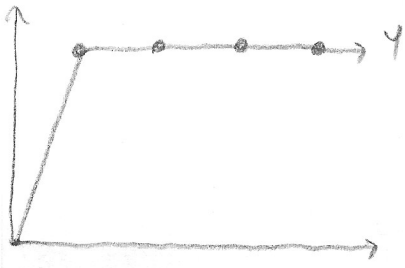
$$K_y = [1\ 0\ 0\ 0]\left(\Phi^T\Phi + \bar{R}\right)^{-1}\left(\Phi^T \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}\right) = 0.2453$$

$$K_{mpc} = [1\ 0\ 0\ 0]\left(\Phi^T\Phi + \bar{R}\right)^{-1}\left(\Phi^T F\right) = [0.6939 \;\; 0.2453]$$

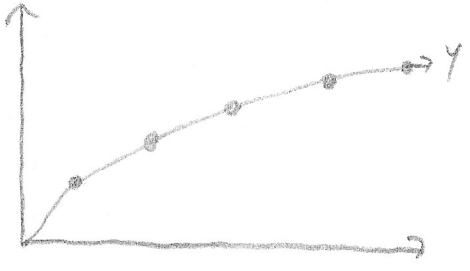
→ With this gain vector, eigenvalues of the closed-loop system are...

$$\lambda_{1,2} = 0.8530 \pm j0.0542$$

→ With $r_w = 10$, the dynamics of closed-loop system have a much slower response than the one in the case when $r_w = 0$



$r_w = 0$



$r_w = 10$

MPC - E1.5 : MPC Closed-loop Stability

```
% June Kwon
% 2/19/2022

clc
clear
close all
%#ok<*ASGLU>
%#ok<*NASGU>
```

1. Augmented State-space Model of Given Dynamics (from E1.2)

```
% Original Dynamics
Ad = 0.8;
Bd = 0.1;
Cd = 1;
Dd = zeros(1,1);

% Prediction Horizon & Control Horizon
Np = 10;
Nc = 4;

% Augmentization & Gain Computation
[P,F,PP,PF,PR,A,B,C] = mpcgain(Ad,Bd,Cd,Nc,Np)
```

```
P = 10×4
    0.1000         0         0         0
    0.1800    0.1000         0         0
    0.2440    0.1800    0.1000         0
    0.2952    0.2440    0.1800    0.1000
    0.3362    0.2952    0.2440    0.1800
    0.3689    0.3362    0.2952    0.2440
    0.3951    0.3689    0.3362    0.2952
    0.4161    0.3951    0.3689    0.3362
    0.4329    0.4161    0.3951    0.3689
    0.4463    0.4329    0.4161    0.3951
F = 10×2
    0.8000    1.0000
    1.4400    1.0000
    1.9520    1.0000
    2.3616    1.0000
    2.6893    1.0000
    2.9514    1.0000
    3.1611    1.0000
    3.3289    1.0000
    3.4631    1.0000
    3.5705    1.0000
PP = 4×4
    1.1541    1.0407    0.9116    0.7726
    1.0407    0.9549    0.8475    0.7259
    0.9116    0.8475    0.7675    0.6674
    0.7726    0.7259    0.6674    0.5943
PF = 4×2
    9.2325    3.2147
    8.3259    2.7684
    7.2927    2.3355
    6.1811    1.9194
```

```
PR = 4×1
    3.2147
    2.7684
    2.3355
    1.9194
A = 2×2
    0.8000         0
    0.8000    1.0000
B = 2×1
    0.1000
    0.1000
C = 1×2
     0     1
```

2-1. Eigenvalues of Closed-loop System with weight, $r_w = 0$

```
% Weighting Matrix
rw = 0;
Rb = rw * eye(Nc,Nc)
```

```
Rb = 4×4
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

```
% Reference Signal
rki = 1;
Rbs = ones(Np,1);

% Initial Condition (at ki = 10)
xki = [0.1 ; 0.2];

% Optimal Control Input Change at ki = 10
DU = (PP + Rb) \ P' * (Rbs * rki - F * xki)
```

```
DU = 4×1
    7.2000
   -6.4000
    0.0000
   -0.0000
```

```
% Ky Gain Vector
Ky_Vec = (PP + Rb) \ (P' * Rbs)
```

```
Ky_Vec = 4×1
   10.0000
   -8.0000
    0.0000
   -0.0000
```

```
% Ky Gain (1st Element of Ky Gain Vector)
Ky = [1 0 0 0] * Ky_Vec
```

```
Ky = 10.0000
```

```
% Kmpc Gain Matrix
Kmpc_Mat = (PP + Rb) \ (P' * F)
```

```
Kmpc_Mat = 4×2
    8.0000   10.0000
   -0.0000   -8.0000
    0.0000    0.0000
   -0.0000   -0.0000
```

```
% Kmpc Gain (1st Row of Kmpc Gain Matrix)
Kmpc = [1 0 0 0] * Kmpc_Mat
```

```
Kmpc = 1×2
    8.0000   10.0000
```

```
% Closed-Loop System
CL = A - B * Kmpc
```

```
CL = 2×2
   -0.0000   -1.0000
   -0.0000   -0.0000
```

```
% Eigenvalue of the close-loop system
eig(CL)
```

```
ans = 2×1
10^-6 ×
    0.2389
   -0.2389
```

Approximately on the origin of the complex plane.

2.2 Eigenvalues of Closed-loop System with weight, $r_w = 10$

```
% Weighting Matrix
rw = 10;
Rb = rw * eye(Nc,Nc)
```

```
Rb = 4×4
    10     0     0     0
     0    10     0     0
     0     0    10     0
     0     0     0    10
```

```
% Reference Signal
rki = 1;
Rbs = ones(Np,1);

% Initial Condition (at ki = 10)
xki = [0.1 ; 0.2];

% Optimal Control Input Change at ki = 10
DU = (PP + Rb) \ P' * (Rbs * rki - F * xki)
```

```
DU = 4×1
    0.1269
    0.1034
    0.0829
    0.0650
```

```
% Ky Gain Vector
```

```
Ky_Vec = (PP + Rb) \ (P' * Rbs)
```

```
Ky_Vec = 4×1
    0.2453
    0.2070
    0.1713
    0.1383
```

```
% Ky Gain (1st Element of Ky Gain Vector)
Ky = [1 0 0 0] * Ky_Vec
```

```
Ky = 0.2453
```

```
% Kmpc Gain Matrix
Kmpc_Mat = (PP + Rb) \ (P' * F)
```

```
Kmpc_Mat = 4×2
    0.6939    0.2453
    0.6220    0.2070
    0.5413    0.1713
    0.4561    0.1383
```

```
% Kmpc Gain (1st Row of Kmpc Gain Matrix)
Kmpc = [1 0 0 0] * Kmpc_Mat
```

```
Kmpc = 1×2
    0.6939    0.2453
```

```
% Closed-Loop System
CL = A - B * Kmpc
```

```
CL = 2×2
    0.7306   -0.0245
    0.7306    0.9755
```

```
% Eigenvalue of the close-loop system
eig(CL)
```

```
ans = 2×1 complex
   0.8530 + 0.0542i
   0.8530 - 0.0542i
```

EX1.6 Prediction Horizon Sensitivity

→ Given: $G(s) = \dfrac{w^2}{s^2 + 0.1ws + w^2}$

where $w = 10$ $\quad N_c = 3$ $\quad \bar{R} = 0.5I$

$\Delta t = 0.01$ $\quad N_p = 20$ or 200

→ Find: Examine the sensitivity issue in the selection of design parameters ($N_p = 20$ or 200; difference?)

→ First, continuous → discrete

$$G(s) = \frac{100}{s^2 + s + 100} \longrightarrow A_c = \begin{bmatrix} -1 & -100 \\ 1 & 0 \end{bmatrix} \quad B_c = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$C_c = [0 \quad 100] \quad D_c = [\,]$$

⇩ Discrete $\Delta t = 0.01$

$$A_d = \begin{bmatrix} 0.9851 & -0.9934 \\ 0.0099 & 0.9950 \end{bmatrix} \quad B_d = \begin{bmatrix} 0.0099 \\ 0.0000 \end{bmatrix}$$

$$C_d = [0 \quad 100] \quad D_d = [\,]$$

↓

$$x_m(k+1) = A_d x_m(k) + B_d M(k)$$

$$Y(k) = C_d x_m(k)$$

→ Next, augmentize

$$x(k+1) = A x(k) + B \Delta M(k)$$

$$Y(k) = C x(k)$$

where $x(k) = \begin{bmatrix} \Delta x_m(k) \\ Y(k) \end{bmatrix}$

where

$$A=\begin{bmatrix} 0.9851 & -0.9934 & 0 \\ 0.0099 & 0.9950 & 0 \\ 0.9934 & 99.5021 & 1 \end{bmatrix} \quad B=\begin{bmatrix} 0.0099 \\ 0.0000 \\ 0.0050 \end{bmatrix}$$

$$C=\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

→ Thus, at $k_T=10$, with initial condition of $x(10)=[0.1 \quad 0.2 \quad 0.3]^T$, the solution of ΔU for $N_p=20$ is,

$$\Delta U=\begin{bmatrix} -144.9984 \\ -65.4710 \\ 1.2037 \end{bmatrix}$$

→ By using receding horizon control principle, the state feedback control gain is... $K_{mpc}=[1\ 0\ 0](\Xi^T\Xi+\bar{R})^{-1}(\Xi^T F)$

$$K_{mpc}=[45.4168 \quad 705.6132 \quad 0.9513]$$

and closed-loop eigenvalues are... $eig(A-BK_{mpc})$

$$\lambda = 0.6974,\ 0.8959 \pm 0.1429j \qquad \text{(unstable)}$$

→ Interesting... now let's take a look at $N_p=200$ case, With $N_p=200$, ΔU, K_{mpc}, λ are...

$$\Delta U=\begin{bmatrix} -645.5885 \\ -0.4664 \\ 629.0276 \end{bmatrix} \quad K_{mpc}=[80.6 \quad 3190 \quad 0.79]$$

$$\lambda = 0.9749,\ 0.5207 \pm 0.2919j$$

→ In comparison with the previous case where the shorter prediction horizon ($N_p = 20$) was used, the parameter ΔU has changed significantly

→ This comparison study illustrated the existing sensitivity in the design with respect to the choice of prediction horizon

→ Looking closely, we see that Hessian Matrix

$$\Phi^T \Phi + \bar{R}$$

(*where $\bar{R} = \begin{bmatrix} r_w & 0 & \cdots \\ 0 & r_w & \cdots \\ 0 & \vdots & \ddots \end{bmatrix}$ Weighting Matrix)

is a function of prediction horizon.

For $N_p = 20$,

$$\Phi^T \Phi = \begin{bmatrix} 9.8796 & 8.9387 & 8.0099 \\ 8.9387 & 8.1020 & 7.2737 \\ 8.0099 & 7.2737 & 6.5425 \end{bmatrix}$$

with condition #, $k(\Phi^T \Phi + 0.5\,I) = \boxed{49.98}$

However, for $N_p = 200$,

$$\Phi^T \Phi = \begin{bmatrix} 236.0557 & 235.5010 & 234.5466 \\ 235.5010 & 235.3753 & 234.8473 \\ 234.5466 & 234.8473 & 234.7473 \end{bmatrix}$$

with condition #, $k(\Phi^T \Phi + 0.5\,I) = \boxed{1410}$

↳ (indicates matrix's sensitivity to the inverse calculations.)

→ The condition number of the Hessian Matrix has significantly increased as the prediction horizon, N_p, increased to 200.

→ This large condition number of Hessian Matrix for a long prediction horizon results in the numerical sensitivity that causes the significant difference between the short and long prediction horizon cases

→ With short prediction & control horizons, the closed-loop predictive control system is not necessarily stable

→ Traditionally, these are the tuning parameters for closed-loop stability & performance

* What is norm?

- Norm is length of the vector (Direction → Length)
- Norm is like a stronger absolute value of the vector...
- Why named as norm? because we can use it to normalize a vector... (ex: $\vec{a}/\|\vec{a}\|$ = unit vector of $\vec{a}$)

$$\|\vec{a}\| = (\vec{a}\cdot\vec{a})^{\frac{1}{2}}$$

$$\frac{\vec{a}}{\|\vec{a}\|} = \frac{\text{Direction}\cdot\text{Length}}{\text{Length}} = \text{(Pure) Direction}$$ (1 Unit)

- Determinant: Volume of the span of vectors in Matrix

* What is condition number?

- A condition number for a matrix and computational task measures how sensitive the answer is to changes in the input data and round off errors in the solution process
- The condition number for inversion of a matrix measures the sensitivity of the solution of a system of linear equations to errors in the data
  - It gives an indication of accuracy of results from matrix inversion and the linear equation solution

$$\left[K(A) = \|A\| \, \|A^{-1}\| \right]$$

$$\langle \quad \text{cond}(A) = \text{norm}(A, p) * \text{norm}(\text{inv}(A), p) \quad \rangle$$

  - A large condition number indicates that a small change in the coefficient matrix A can lead to larger changes in the output b in the linear equations $Ax = b$.
  - The extreme case is when A is so poorly conditioned that it is singular (an ∞ condition number), in which case it has no inverse and the linear equation has no unique solution

$$\left\langle \begin{array}{l} \text{COND near 1} \rightarrow \text{well-conditioned matrix} \\ \text{COND too large} \rightarrow \text{ill-conditioned matrix} \end{array} \right\rangle$$

singular matrix has COND = ∞

- Simple EX
  - Given $A = \begin{bmatrix} 4.1 & 2.8 \\ 9.7 & 6.6 \end{bmatrix} \rightarrow \text{cond}(A) = 1623 \;,\; \text{inv}(A) = \begin{bmatrix} -66 & 28 \\ 97 & -41 \end{bmatrix}$
  - Since cond is much larger than 1, the matrix is sensitive to the inverse calculations.
  - Make a small change in 2nd row of A...

$$A2 = \begin{bmatrix} 4.1 & 2.8 \\ 9.611 & 6.608 \end{bmatrix} \rightarrow \text{inv}(A2) = \begin{bmatrix} 412 & -200 \\ -690.8 & 292.8 \end{bmatrix}$$

small change

  - Comparing inv(A) and inv(A2), we can see that making a small change in A can completely change the result of the inverse calculations...

MPC - E1.6 : Prediction Horizon Sensitivity

```
% June Kwon
% 2/20/2022

clc
clear
close all
%#ok<*ASGLU>
%#ok<*NASGU>
```

1. Effect of Prediction Horizon, $N_p = 20$, on Control Input Change

```
% Given Parameters
w = 10;
h = 0.01;

% Np = 20
Np = 20;
Nc = 3;

% Given Dynamics
NUM = w^2;
DEN = [1 0.1*w w^2];
G = tf(NUM,DEN)
```

```
G =

       100
  -------------
  s^2 + s + 100

Continuous-time transfer function.
```

```
% State-space Realization
[Ac,Bc,Cc,Dc] = tf2ss(NUM,DEN)
```

```
Ac = 2×2
    -1  -100
     1     0
Bc = 2×1
     1
     0
Cc = 1×2
     0   100
Dc = 0
```

```
% Discretization
[Ad,Bd,Cd,Dd] = c2dm(Ac,Bc,Cc,Dc,h)
```

```
Ad = 2×2
    0.9851   -0.9934
    0.0099    0.9950
Bd = 2×1
    0.0099
    0.0000
```

```
Cd = 1×2
     0   100
Dd = 0
```

```
% Augmentation
[P,F,PP,PF,PR,Ae,Be,Ce] = mpcgain(Ad,Bd,Cd,Nc,Np)
```

```
P = 20×3
    0.0050         0         0
    0.0198    0.0050         0
    0.0442    0.0198    0.0050
    0.0779    0.0442    0.0198
    0.1204    0.0779    0.0442
    0.1712    0.1204    0.0779
    0.2298    0.1712    0.1204
    0.2955    0.2298    0.1712
    0.3674    0.2955    0.2298
    0.4450    0.3674    0.2955
      ⋮
F = 20×3
    0.9934   99.5021    1.0000
    2.9603  197.5219    1.0000
    5.8716  293.0999    1.0000
    9.6890  385.3099    1.0000
   14.3653  473.2688    1.0000
   19.8457  556.1444    1.0000
   26.0676  633.1634    1.0000
   32.9619  703.6183    1.0000
   40.4531  766.8739    1.0000
   48.4610  822.3731    1.0000
      ⋮
PP = 3×3
    9.8796    8.9387    8.0099
    8.9387    8.1020    7.2737
    8.0099    7.2737    6.5425
PF = 3×3
10^3 ×
    1.0342    9.9247    0.0111
    0.9349    8.8193    0.0098
    0.8371    7.7570    0.0085
PR = 3×1
   11.0929
    9.7596
    8.5108
Ae = 3×3
    0.9851   -0.9934         0
    0.0099    0.9950         0
    0.9934   99.5021    1.0000
Be = 3×1
    0.0099
    0.0000
    0.0050
Ce = 1×3
     0     0     1
```

```
% Weighting Matrix
rw = 0.5;
Rb = rw * eye(Nc,Nc)
```

```
Rb = 3×3
```

```
    0.5000         0         0
         0    0.5000         0
         0         0    0.5000
```

```
% Reference Signal
rki = 1;
Rbs = ones(Np,1)
```

```
Rbs = 20×1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     ⋮
```

```
% Initial Condition
xki = [0.1 ; 0.2 ; 0.3]
```

```
xki = 3×1
    0.1000
    0.2000
    0.3000
```

```
% Optimal Control Input Change for Np = 20
DU = (PP + Rb) \ P' * (Rbs * rki - F * xki)
```

```
DU = 3×1
 -144.9984
  -65.4710
    1.2037
```

```
% Ky Gain
KyVec = (PP + Rb) \ (P' * Rbs)
```

```
KyVec = 3×1
    0.9513
    0.3084
   -0.1920
```

```
Ky = [1 0 0] * KyVec
```

```
Ky = 0.9513
```

```
% Kmpc Gain
KmpcMat = (PP + Rb) \ (P' * F)
```

```
KmpcMat = 3×3
   45.4168  705.6132    0.9513
   36.8540  310.0074    0.3084
   29.1412  -21.2610   -0.1920
```

```
Kmpc = [1 0 0] * KmpcMat
```

```
Kmpc = 1×3
```

```
   45.4168  705.6132    0.9513
```

```
% Closed-loop System
CL = Ae - Be * Kmpc
```

```
CL = 3×3
    0.5339   -8.0026   -0.0095
    0.0077    0.9599   -0.0000
    0.7672   95.9887    0.9953
```

```
% Eigenvalues of Closed-loop System
eig(CL)
```

```
ans = 3×1 complex
   0.6974 + 0.0000i
   0.8959 + 0.1429i
   0.8959 - 0.1429i
```

```
% Condition Number of Hessian Matrix
cond(PP + Rb)
```

```
ans = 49.9856
```

2. Effect of Prediction Horizon, $N_p = 200$, on Control Input Change

```
% Given Parameters
w = 10;
h = 0.01;

% Np = 200
Np = 200;
Nc = 3;
% Given Dynamics
NUM = w^2;
DEN = [1 0.1*w w^2];
G = tf(NUM,DEN)
```

```
G =

       100
  -------------
  s^2 + s + 100

Continuous-time transfer function.
```

```
% State-space Realization
[Ac,Bc,Cc,Dc] = tf2ss(NUM,DEN)
```

```
Ac = 2×2
    -1  -100
     1     0
Bc = 2×1
     1
     0
Cc = 1×2
     0   100
Dc = 0
```

```
% Discretization
[Ad,Bd,Cd,Dd] = c2dm(Ac,Bc,Cc,Dc,h)
```

```
Ad = 2×2
    0.9851   -0.9934
    0.0099    0.9950
Bd = 2×1
    0.0099
    0.0000
Cd = 1×2
     0   100
Dd = 0
```

```
% Augmentation
[P,F,PP,PF,PR,Ae,Be,Ce] = mpcgain(Ad,Bd,Cd,Nc,Np)
```

```
P = 200×3
    0.0050         0         0
    0.0198    0.0050         0
    0.0442    0.0198    0.0050
    0.0779    0.0442    0.0198
    0.1204    0.0779    0.0442
    0.1712    0.1204    0.0779
    0.2298    0.1712    0.1204
    0.2955    0.2298    0.1712
    0.3674    0.2955    0.2298
    0.4450    0.3674    0.2955
      ⋮
F = 200×3
    0.9934   99.5021    1.0000
    2.9603  197.5219    1.0000
    5.8716  293.0999    1.0000
    9.6890  385.3099    1.0000
   14.3653  473.2688    1.0000
   19.8457  556.1444    1.0000
   26.0676  633.1634    1.0000
   32.9619  703.6183    1.0000
   40.4531  766.8739    1.0000
   48.4610  822.3731    1.0000
      ⋮
PP = 3×3
  236.0557  235.5010  234.5466
  235.5010  235.3753  234.8473
  234.5466  234.8473  234.7473
PF = 3×3
10^4 ×
    2.3604    1.5341    0.0196
    2.3527    1.1027    0.0195
    2.3410    0.6742    0.0195
PR = 3×1
  196.2662
  195.4413
  194.6489
Ae = 3×3
    0.9851   -0.9934         0
    0.0099    0.9950         0
    0.9934   99.5021    1.0000
Be = 3×1
    0.0099
    0.0000
    0.0050
```

```
Ce = 1×3
     0     0     1
```

```
% Weighting Matrix
rw = 0.5;
Rb = rw * eye(Nc,Nc)
```

```
Rb = 3×3
    0.5000         0         0
         0    0.5000         0
         0         0    0.5000
```

```
% Reference Signal
rki = 1;
Rbs = ones(Np,1)
```

```
Rbs = 200×1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     ⋮
```

```
% Initial Condition (Starting Point)
xki = [0.1 ; 0.2 ; 0.3]
```

```
xki = 3×1
    0.1000
    0.2000
    0.3000
```

```
% Optimal Control Input Change for Np = 200
DU = (PP + Rb) \ P' * (Rbs * rki - F * xki)
```

```
DU = 3×1
 -645.5885
   -0.4664
  629.0276
```

```
% Ky Gain
KyVec = (PP + Rb) \ (P' * Rbs)
```

```
KyVec = 3×1
    0.7945
    0.0338
    0.0015
```

```
Ky = [1 0 0] * KyVec
```

```
Ky = 0.7945
```

```
% Kmpc Gain
KmpcMat = (PP + Rb) \ (P' * F)
```

```
KmpcMat = 3×3
10^3 ×
    0.0806    3.1904    0.0008
    0.0331   -0.0141    0.0000
   -0.0139   -3.1382    0.0000
```

```
Kmpc = [1 0 0] * KmpcMat
```

```
Kmpc = 1×3
10^3 ×
    0.0806    3.1904    0.0008
```

```
% Closed-loop System
CL = Ae - Be * Kmpc
```

```
CL = 3×3
    0.1841  -32.6856   -0.0079
    0.0059    0.8362   -0.0000
    0.5919   83.6163    0.9960
```

```
% Eigenvalues of Closed-loop System
eig(CL)
```

```
ans = 3×1 complex
   0.5207 + 0.2919i
   0.5207 - 0.2919i
   0.9749 + 0.0000i
```

```
% Condition Number of Hessian Matrix
cond(PP + Rb)
```

```
ans = 1.4099e+03
```

MPC - T1.3 : Receding Horizon Control

```
% June Kwon
% 2/18/2022

clc
clear
close all
%#ok<*ASGLU>
%#ok<*NASGU>
```

1. Plant Dynamics & Horizons & MPC Gain

```
% Plant Dynamics
Ap = [1 1 ; 0 1];
Bp = [0.5 ; 1];
Cp = [1 0];
Dp = 0;

% Prediction & Control Horizons
Np = 20, Nc = 4
```

```
Np = 20
Nc = 4
```

```
% MPC Gain Computation & Augmentation
[P,F,PP,PF,PR,Ae,Be,Ce] = mpcgain(Ap,Bp,Cp,Nc,Np)
```

```
P = 20×4
    0.5000         0         0         0
    2.0000    0.5000         0         0
    4.5000    2.0000    0.5000         0
    8.0000    4.5000    2.0000    0.5000
   12.5000    8.0000    4.5000    2.0000
   18.0000   12.5000    8.0000    4.5000
   24.5000   18.0000   12.5000    8.0000
   32.0000   24.5000   18.0000   12.5000
   40.5000   32.0000   24.5000   18.0000
   50.0000   40.5000   32.0000   24.5000
      ⋮

F = 20×3
     1     1     1
     2     3     1
     3     6     1
     4    10     1
     5    15     1
     6    21     1
     7    28     1
     8    36     1
     9    45     1
    10    55     1
     ⋮

PP = 4×4
10^5 ×
    1.8067    1.5933    1.3944    1.2097
    1.5933    1.4067    1.2323    1.0704
```

```
    1.3944    1.2323    1.0809    0.9399
    1.2097    1.0704    0.9399    0.8184
PF = 4×3
10^5 ×
    0.2205    1.9169    0.0143
    0.1928    1.6898    0.0123
    0.1673    1.4780    0.0105
    0.1438    1.2816    0.0089
PR = 4×1
10^3 ×
    1.4350
    1.2350
    1.0545
    0.8925
Ae = 3×3
     1     1     0
     0     1     0
     1     1     1
Be = 3×1
    0.5000
    1.0000
    0.5000
Ce = 1×3
     0     0     1
```

2. Initial Condition & Reference Signal

```
% Initial Condtion
[n,n_in] = size(Be);      % n : number of states in augmentized state-space model
                          % n_in : number of input
xm = [0 0]';              % xm : initial condition of the original state-space model
y = 0;                    % y : initial condition of the origianl state-space model
Xf = [xm ; y];            % Xf = [delta_xm' y']'
y_initial = y;
X_initial = Xf;
n, n_in, xm, y, Xf
```

```
n = 3
n_in = 1
xm = 2×1
     0
     0
y = 0
Xf = 3×1
     0
     0
     0
```

```
% Reference Signal
N_SIM = 100;              % N_SIM : Number of Simulation (ki)
r = ones(N_SIM,1);        % r : Reference Signal = 1
u = 0;                    % u(k-1) = 0
u_initial = u;
N_SIM, r, u
```

```
N_SIM = 100
r = 100×1
     1
     1
```

```
     1
     1
     1
     1
     1
     1
     1
     1
     ⋮
u = 0
```

```
% Weighting Matrix
rw = 0.1;
Rb = rw * eye(Nc,Nc);
rw, Rb
```

```
rw = 0.1000
Rb = 4×4
    0.1000         0         0         0
         0    0.1000         0         0
         0         0    0.1000         0
         0         0         0    0.1000
```

3. Receding Horizon Control

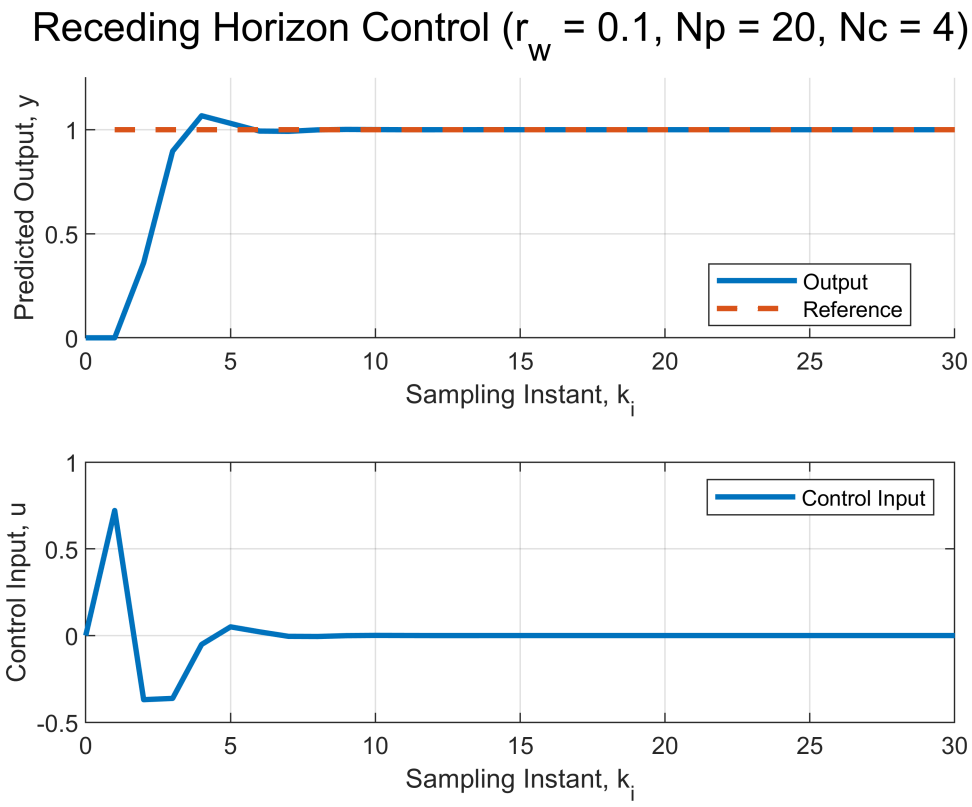
```
% Pre-allocation
u_store = zeros(N_SIM,1);
y_store = zeros(N_SIM,1);
X_store = zeros(3,N_SIM);

% Receding Horizon Control
for ki = 1 : N_SIM

    DU = (PP + Rb)\(PR*r(ki) - PF*Xf);
    du = DU(1,1);
    u = u + du;

    u_store(ki) = u;
    y_store(ki) = y;

    xm_old = xm;
    xm = Ap*xm + Bp*u;
    y = Cp*xm;
    Xf = [xm-xm_old ; y];
    X_store(:,ki) = Xf;

end
```

- From the receding horizon control, at sample k_i, the ΔU vector is calculated using the set-point signal, $r(k_i)$, and the state vetor, X_f.
- As k_i iterates forward, r and X_f will be updated accordingly.

- Then, $\Delta u(k_i)$ is taken as the first element of ΔU and $u(k_i) = u(k_i - 1) + \Delta u(k_i)$

4. Plotting ($r_w = 0.1$)

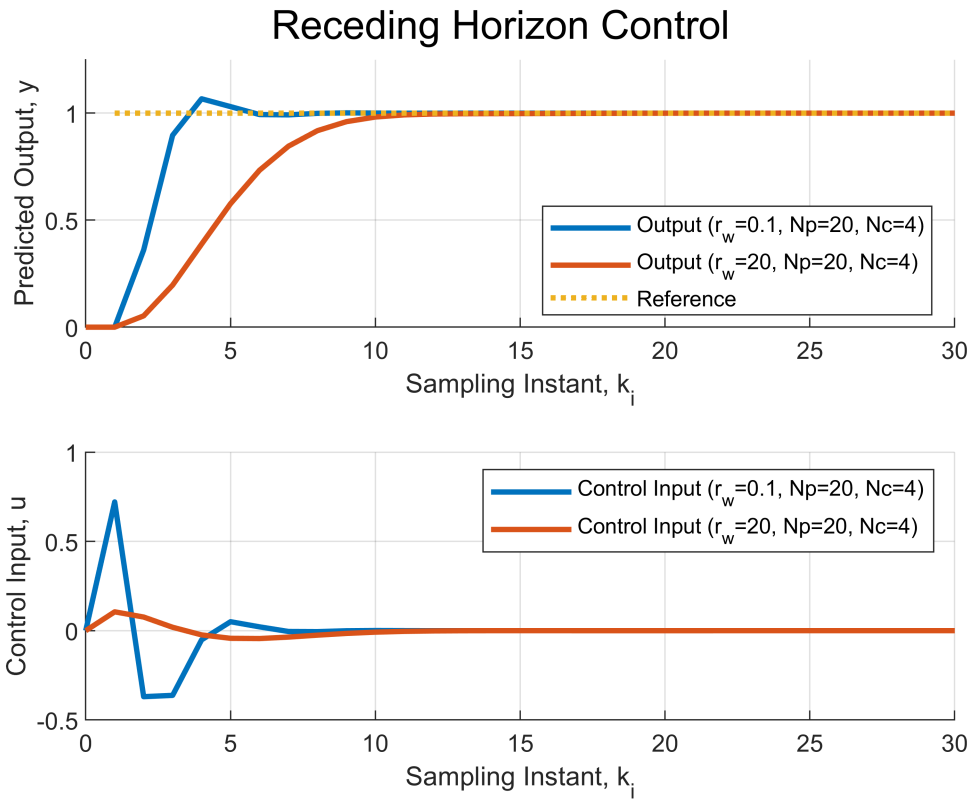
```
figure;
k = 0:N_SIM;
X_store = [X_initial X_store];

subplot(211); hold on;
plot(k,[y_initial ; y_store],'linewidth',2);
plot(k(2:end),r,'--','linewidth',2);
xlabel('Sampling Instant, k_i'); ylabel('Predicted Output, y')
legend('Output','Reference',"Location","best");
grid on; ylim([0 1.25]); xlim([0 30]);

subplot(212);
plot(k,[u_initial ; u_store],'linewidth',2);
xlabel('Sampling Instant, k_i'); ylabel('Control Input, u')
legend('Control Input');
grid on; ylim([-0.5 1]); xlim([0 30]);

sgtitle('Receding Horizon Control (r_w = 0.1, Np = 20, Nc = 4)')
```



- **With small weight ($r_w = 0.1$), we can observe the receding horizon control successfully enabled the closed-loop system to reach the reference signal. But, it should be noted that, with faster response, higher input cost was spent.**

5 - 1. What if we increase the weight? (from $r_w = 0.1$ to $r_w = 20$)

```
% Increase from Np = 20 to Np = 60
Np = 20; Nc = 4; rw = 20;
[u_store_5_1,y_store_5_1,X_store_5_1] = RHC(Ap,Bp,Cp,Dp,Np,Nc,rw,r,N_SIM);

% Plot
figure;
k = 0:N_SIM;
X_store_5_1 = [X_initial X_store_5_1];

subplot(211); hold on;
plot(k,[y_initial ; y_store],'linewidth',2);
plot(k,[y_initial ; y_store_5_1],'linewidth',2);
plot(k(2:end),r,':','linewidth',2);
xlabel('Sampling Instant, k_i'); ylabel('Predicted Output, y')
legend('Output (r_w=0.1, Np=20, Nc=4)','Output (r_w=20, Np=20, Nc=4)',...
    'Reference',"Location","best");
grid on; ylim([0 1.25]); xlim([0 30]);

subplot(212); hold on;
plot(k,[u_initial ; u_store],'linewidth',2);
plot(k,[u_initial ; u_store_5_1],'linewidth',2);
xlabel('Sampling Instant, k_i'); ylabel('Control Input, u');
legend('Control Input (r_w=0.1, Np=20, Nc=4)',...
    'Control Input (r_w=20, Np=20, Nc=4)');
grid on; ylim([-0.5 1]); xlim([0 30]);

sgtitle('Receding Horizon Control')
```



As shown aobve,

- **With increasing weight ($r_w = 20$) (red), we can observe that the output reaches the reference signal slower, however, with much less overshoot than $r_w = 0.1$ case (blue). Moreover, we can also observe that the low input cost was spent to achive the slow system response.**

Thus, increasing weight results in

1. **The system response becomes slower (less overshoot, oscilliation, and settling time).**
2. **Since the system response is slower, less input cost will be spent.**

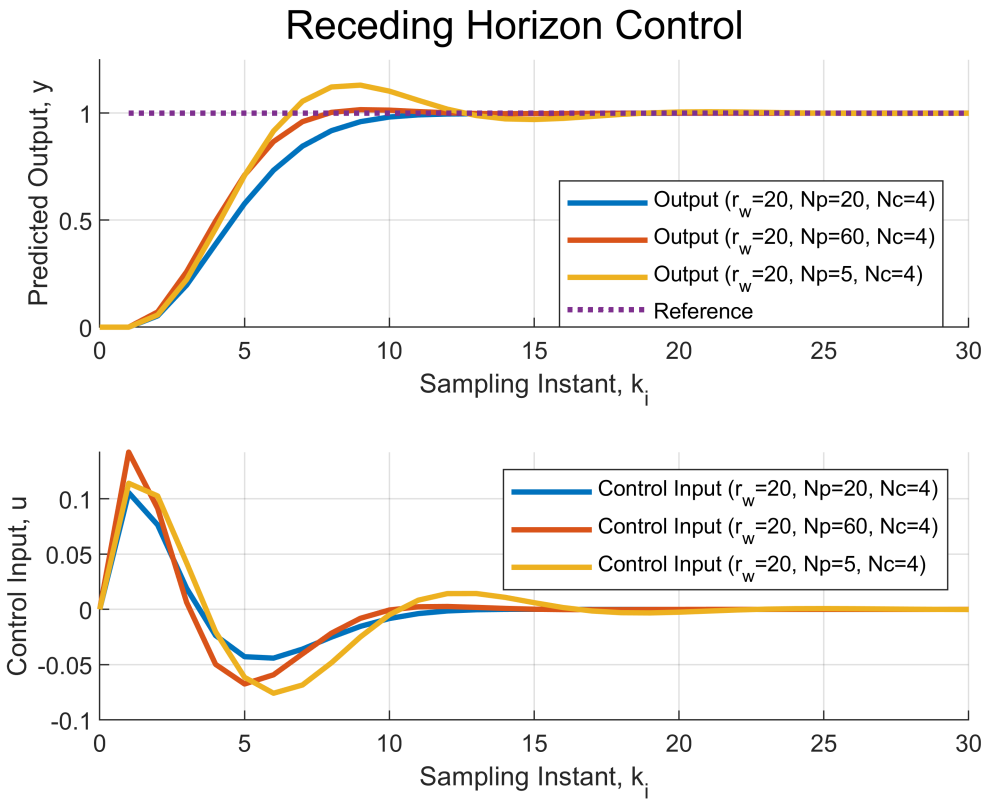
Descreasing the weight will result in

- **The system response becomes faster (reaches the reference faster), but with possibly more overshoot in the system (more osciiliation and longer settling time).**
- **Higher input cost will be spent to achieve the faster system response.**

5 - 2. What if we increase/decrease the prediction horizon? (from $N_p = 20$ to $N_p = 60$, $N_p = 5$)

```
% Increase from Np = 20 to Np = 60
Np = 60; Nc = 4; rw = 20;
[u_store_5_2_NpUp,y_store_5_2_NpUp,X_store_5_2_NpUp] = RHC(Ap,Bp,Cp,Dp,Np,Nc,rw,r,N_SIM);
```

```
% Decrease from Np = 20 to Np = 5
Np = 5; Nc = 4; rw = 20;
[u_store_5_2_NpDown,y_store_5_2_NpDown,X_store_5_2_NpDown] = RHC(Ap,Bp,Cp,Dp,Np,Nc,rw,r,N_SIM);

% Plot
figure;
k = 0:N_SIM;

subplot(211); hold on;
plot(k,[y_initial ; y_store_5_1],'linewidth',2);
plot(k,[y_initial ; y_store_5_2_NpUp],'linewidth',2);
plot(k,[y_initial ; y_store_5_2_NpDown],'linewidth',2);
plot(k(2:end),r,':','linewidth',2);
xlabel('Sampling Instant, k_i'); ylabel('Predicted Output, y')
legend('Output (r_w=20, Np=20, Nc=4)','Output (r_w=20, Np=60, Nc=4)',...
    'Output (r_w=20, Np=5, Nc=4)','Reference',"Location","best");
grid on; ylim([0 1.25]); xlim([0 30]);

subplot(212); hold on;
plot(k,[u_initial ; u_store_5_1],'linewidth',2);
plot(k,[u_initial ; u_store_5_2_NpUp],'linewidth',2);
plot(k,[u_initial ; u_store_5_2_NpDown],'linewidth',2);
xlabel('Sampling Instant, k_i'); ylabel('Control Input, u');
legend('Control Input (r_w=20, Np=20, Nc=4)','Control Input (r_w=20, Np=60, Nc=4)',...
    'Control Input (r_w=20, Np=5, Nc=4)');
grid on; xlim([0 30]);

sgtitle('Receding Horizon Control')
```



As shown above, when increasing the prediction horizon (red vs blue),

1. **The system response becomes faster (but with a small overshoot).**
2. **To make the system response faster, higher input cost is being spent.**

When decreasing the prediction horizon (yellow vs blue)

1. **The system response becomes faster. (However, large overshoot was created, leading to longer settling time to reach the steady-state.)**
2. **Input cost also slightly increased.**

Thus, it should be noted that decreasing the prediction horizon is not necessarily a good option to increase the performance.

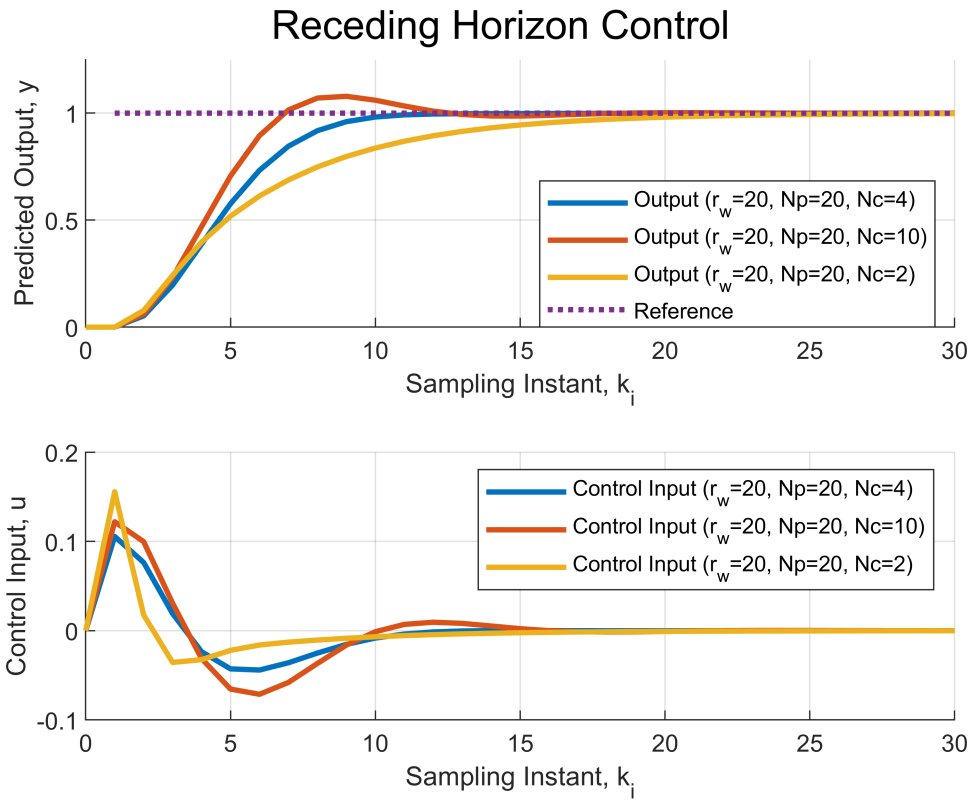
However, at the same time, increasing the preediction horizon also does not necessariliy promise the performance increase.

As shown above, the prediction horizon almost tripled (from Np = 20 to Np = 60) just to achieve a small bit of improvement in the system response.

Thus, there will be some tradeoff in increasing the prediction horizon.

5 - 3. What if we increase/decrease the control horizon? (from $N_c = 4$ to $N_c = 10$, $N_c = 2$)

```
% Increase from Nc = 4 to Nc = 10
Np = 20; Nc = 10; rw = 20;
[u_store_5_3_NcUp,y_store_5_3_NcUp,X_store_5_3_NcUp] = RHC(Ap,Bp,Cp,Dp,Np,Nc,rw,r,N_SIM);

% Decrease from Nc = 4 to Np = 2
Np = 20; Nc = 2; rw = 20;
[u_store_5_3_NcDown,y_store_5_3_NcDown,X_store_5_3_NcDown] = RHC(Ap,Bp,Cp,Dp,Np,Nc,rw,r,N_SIM);

% Plot
figure;
k = 0:N_SIM;

subplot(211); hold on;
plot(k,[y_initial ; y_store_5_1],'linewidth',2);
plot(k,[y_initial ; y_store_5_3_NcUp],'linewidth',2);
plot(k,[y_initial ; y_store_5_3_NcDown],'linewidth',2);
plot(k(2:end),r,':','linewidth',2);
xlabel('Sampling Instant, k_i'); ylabel('Predicted Output, y')
legend('Output (r_w=20, Np=20, Nc=4)','Output (r_w=20, Np=20, Nc=10)',...
    'Output (r_w=20, Np=20, Nc=2)','Reference',"Location","best");
grid on; ylim([0 1.25]); xlim([0 30]);

subplot(212); hold on;
plot(k,[u_initial ; u_store_5_1],'linewidth',2);
plot(k,[u_initial ; u_store_5_3_NcUp],'linewidth',2);
plot(k,[u_initial ; u_store_5_3_NcDown],'linewidth',2);
xlabel('Sampling Instant, k_i'); ylabel('Control Input, u');
legend('Control Input (r_w=20, Np=20, Nc=4)','Control Input (r_w=20, Np=20, Nc=10)',...
    'Control Input (r_w=20, Np=20, Nc=2)');
grid on; xlim([0 30]);

sgtitle('Receding Horizon Control')
```



As shown above, when increasing the control horizon (red vs blue),

1. **The system response becomes faster (but with a large overshoot and longer settiling time).**
2. **Higher input cost is spent to make the system response faster**

When decreasing the control horizon (yellow vs blue)

1. **The system response becomes slower. (with less overshoot)**
2. **However, this does not necessarily decrease the input cost.**

From simulation, it can be learned that how increasing control horizon affects the system response depends on the size of the prediction horizon.

(EX: For our given system, if Nc = Np/2, then we see faster response with overshoot. If Nc = Np/10, we see a slower response.)

Thus, it is important that rw, Nc, and Np are all correlated in determining the system response.

<u>To be able to have the desired control performance, it would be important to understand the right balance in rw, Np, and Nc.</u>

For the given system, I think rw = 20, Np = 20, and Nc = 4 yields relatively well-balanced results...

6. Simulation of ($r_w = 20$, $N_p = 20$, $N_c = 4$)

```
% Custom Reference Signal
```

```
N_SIM = 200;
r = ones(N_SIM,1);
r(30:60) = 0;
r(110:130) = 0.5;
r(131:160) = 0;

% Receding Horizon Control
rw = 20; Np = 20; Nc = 4;
[u_store_6,y_store_6,X_store_6] = RHC(Ap,Bp,Cp,Dp,Np,Nc,rw,r,N_SIM);

% Plot
figure;
k = 0:N_SIM;

subplot(211); hold on;
plot(k,[y_initial ; y_store_6],'linewidth',2);
plot(k(2:end),r,'--','linewidth',2);
xlabel('Sampling Instant, k_i'); ylabel('Predicted Output, y');
legend('Output','Reference',"Location","best");
grid on;

subplot(212);
plot(k,[u_initial ; u_store_6],'linewidth',2);
xlabel('Sampling Instant, k_i'); ylabel('Control Input, u');
legend('Control Input',"Location","best");
grid on; ylim([-0.15 0.15])

sgtitle('Receding Horizon Control (r_w = 20, Np = 20, Nc = 4)')
```