HW3 EX1

Thursday, February 22, 2024 5:12 PM

- $X = SU + Mx_0$ From the notes

$$\underset{N\times n_x}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}}_{X} = \underset{N\times n_u}{\begin{bmatrix} B & 0 & 0 & 0 \\ AB & B & & \vdots \\ \vdots & & & \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}}_{S} \underset{N\times n_u}{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{U} + \underset{N\times n_x}{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} x_0 \\ \vdots \\ \\ \\ \\ \end{bmatrix}}_{n_x}$$

$N\times n_x$: 1 ; $N\times n_u$: $N\times n_u$; $N\times n_u$: 1 ; $N\times n_x$: n_x ; n_x : 1

- Y in vector form

$Y = DU + CX'$ We use X' instead of X because Y needs x_0 in the vector

$$\underset{N\times n_x}{\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix}}_{Y} = \underset{N\times n_u}{\begin{bmatrix} D & 0 & \cdots & 0 \\ 0 & D & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & D \end{bmatrix}}_{D} \underset{N\times n_u}{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{U} + \underset{N\times n_x}{\begin{bmatrix} C & 0 & \cdots & 0 \\ 0 & C & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & C \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}}_{X'}_{N\times n_x}$$

$N\times n_x$: 1 ; $N\times n_u$: $N\times n_u$; $N\times n_u$: 1 ; $N\times n_x$: $N\times n_x$; $N\times n_x$: 1

- where $X' = S'U + M'x_0$

$$\underset{N\times n_x}{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}}_{X'} = \underset{N\times n_x}{\begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B & & & \vdots \\ AB & B & \ddots & \vdots \\ \vdots & & & 0 \\ A^{N-2}B & A^{N-3}B & \cdots & B \end{bmatrix}}_{S'} \underset{N\times n_u}{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{U} + \underset{N\times n_x}{\begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^{N-1} \end{bmatrix}}_{M'} \underbrace{\begin{bmatrix} x_0 \\ \vdots \\ \\ \\ \\ \end{bmatrix}}_{n_x}$$

$N\times n_x$: $N\times n_x$; $N\times n_u$: 1 ; $N\times n_x$: n_x ; n_x : 1

- Substituting X' in $Y = DU + C(S'U + M'x_0)$

$$= DU + CS'U + CM'x_0$$

$$= (D + CS')U + CM'x_0$$

- Let constraints in vector form be

$$x_{min} = \underset{N\times n_x}{\begin{bmatrix} x_{min} \\ \vdots \\ x_{min} \end{bmatrix}} \quad X_{max} = \begin{bmatrix} x_{max} \\ \vdots \\ x_{max} \end{bmatrix} \quad U_{min} = \begin{bmatrix} u_{min} \\ \vdots \\ u_{min} \end{bmatrix} \quad U_{max} = \begin{bmatrix} u_{max} \\ \vdots \\ u_{max} \end{bmatrix} \quad Y_{min} = \begin{bmatrix} y_{min} \\ \vdots \\ y_{min} \end{bmatrix} \quad Y_{max} = \begin{bmatrix} y_{max} \\ \vdots \\ y_{max} \end{bmatrix}$$

- With constraints on x, u

$$X_{min} \le X = SU + Mx_0 \le X_{max}$$

$$U_{min} \le U \le U_{max}$$

$$Y_{min} \le Y = (D + CS')U + CM'x_0 \le Y_{max}$$

- In vectorized form:

$$\begin{bmatrix} S \\ -S \\ I \\ -I \\ D+CS' \\ -(D+CS') \end{bmatrix} U \le \begin{bmatrix} X_{max} - Mx_0 \\ -X_{min} + Mx_0 \\ U_{max} \\ -U_{min} \\ Y_{max} - M'x_0 \\ -Y_{min} + M'x_0 \end{bmatrix}$$

- This can be written as: $GU \le W + Tx_0$, where

$$G = \begin{bmatrix} S \\ -S \\ I \\ -I \\ D+CS' \\ -(D+CS') \end{bmatrix}, \quad W = \begin{bmatrix} X_{max} \\ -X_{min} \\ U_{max} \\ -U_{min} \\ Y_{max} \\ -Y_{min} \end{bmatrix}, \quad T = \begin{bmatrix} -M \\ M \\ 0 \\ 0 \\ -CM' \\ CM' \end{bmatrix}$$

- For the constrained minimization of $J_N = U^T HU + 2q^T U$, all the matrices in the cost function remain the same.
- In the constraint definition: $GU \le W + Tx_0$, the matrices G, W and T need to be extended as shown above to accomodate the constraint $y_{min} \le y_k \le y_{max}$ for $k = 0 \cdots N-1$

Table of Contents

AE740 HW1 akshatdy

```
clc;
clear;
close all;
clear variables
format shortG;
```

2. Spacecraft attitude dynamics

2.a Linearized dynamics

```
% differentiate to get jacobian
syms phi theta psi omega1 omega2 omega3 M1 M2 M3;
x = [phi; theta; psi; omega1; omega2; omega3; ];
u = [M1; M2; M3];
f = scdynamics(0, x, u);
A = jacobian(f, x);
B = jacobian(f, u);
% point of linearization
x0 = [0; 0; 0; 0; 0; 0; ];
nx = length(x0);
u0 = [0; 0; 0];
nu = length(u0);
Ac = double(subs(A, [x; u], [x0; u0]));
Bc = double(subs(B, [x; u], [x0; u0]));
% display
% disp('Ac = ');
% disp(Ac);
% disp('Bc = ');
% disp(Bc);
% convert to discrete time
Ts = 2;
Cc = zeros(size(Ac, 1), size(Ac, 2));
Dc = zeros(size(Cc, 1), size(Bc, 2));
scd = c2d(ss(Ac, Bc, Cc, Dc), Ts, 'zoh');
Ad = scd.A;
disp('Ad = ');
disp(Ad);
Bd = scd.B;
```

```
disp('Bd = ');
disp(Bd);
Cd = scd.C;
Dd = scd.D;
```

2.b Simulate using MPT Toolbox

```
disp("2.b Simulate using MPT Toolbox");
% consts
x0 = [-0.4; -0.8; 1.2; -0.02; -0.02; 0.02];
Q = diag([1, 1, 1, 0.01, 0.01, 0.01]);
R = diag([0.01, 0.01, 0.01]);
% get P with DARE
[Kinf, Pinf, CLeig] = dlqr(Ad, Bd, Q, R);

% MPT stuff, do once
tbxmanager restorepath
mpt_init

% setup model
modelMPT = LTISystem('A', Ad, 'B', Bd, 'C', Cd, 'D', Dd, 'Ts', Ts);
modelMPT.u.min = -0.1 * ones(nu, 1);
modelMPT.u.max = 0.1 * ones(nu, 1);
modelMPT.u.penalty = QuadFunction(R);
modelMPT.x.penalty = QuadFunction(Q);
modelMPT.x.with('terminalPenalty');
modelMPT.x.terminalPenalty = QuadFunction(Pinf);
ctrl2 = MPCController(modelMPT, 2);
ctrl20 = MPCController(modelMPT, 20);
```

```
2.b Simulate using MPT Toolbox
Toolbox "mpt:3.2.1:all" added to the Matlab path.
Toolbox "mptdoc:3.0.4:all" added to the Matlab path.
Toolbox "cddmex:1.0.1:glnxa64" added to the Matlab path.
Toolbox "fourier:1.0:glnxa64" added to the Matlab path.
Toolbox "glpkmex:1.0:glnxa64" added to the Matlab path.
Toolbox "hysdel:2.0.6:glnxa64" added to the Matlab path.
Toolbox "lcp:1.0.3:glnxa64" added to the Matlab path.
Toolbox "yalmip:R20230609:all" added to the Matlab path.
Toolbox "sedumi:1.3:glnxa64" added to the Matlab path.
Toolbox "espresso:1.0:glnxa64" added to the Matlab path.
MPT searches for solvers on the path ...

 LINPROG ..................................... linprog.m
 QUADPROG .................................... quadprog.m
 GLPK ........................................... glpkcc
 CDD ............................................ cddmex
 LCP ............................................... lcp
 SEDUMI ...................................... sedumi.m
 PLCP ....................................... mpt_plcp.m
 MPQP ....................................... mpt_mpqp.m
 MPLP ....................................... mpt_mplp.m
 ENUMPLCP ................................ mpt_enum_plcp.m
```

```
 ENUMPQP ..................................... mpt_enum_pqp.m
 RLENUMPQP ................................... mpt_enum_pqp.m

MPT toolbox @version@ initialized...
Copyright (C) 2003-2024 by M. Kvasnica, C.N. Jones, and M. Herceg
For news, visit the MPT web page at http://control.ee.ethz.ch/~mpt/
            LP solver: LCP
            QP solver: LCP
          MILP solver: GLPK
 parametric LP solver: PLCP
 parametric QP solver: PLCP

These default options can be changed. See "help mptopt" for more details.
```

2.c Simulate using Hybrid MPC Toolbox

```
disp("2.c Simulate using Hybrid MPC Toolbox");
```

```
2.c Simulate using Hybrid MPC Toolbox
```

Both the toolboxes give the same results, so we can use either one for simulation, and it assures us that the simulation is correct.

```
% Hybrid toolbox stuff
addpath( ...
  genpath('/home/akshatd/mine/umich/sem2/740/toolboxes/hybtbx-linux/'), ...
'-end')
clear cost constraints horizon2 horizon20;

% setup model
cost.Q = Q;
cost.R = R;
cost.P = Pinf;
constraints.umin = -0.1 * ones(nu, 1);
constraints.umax = 0.1 * ones(nu, 1);
NHoriz = 2;
horizon2.Nu = NHoriz;
horizon2.N = NHoriz;
horizon2.Ncu = NHoriz - 1;
horizon2.Ncy = NHoriz - 1;
ctrl2h = lincon(scd, 'reg', cost, horizon2, constraints, 'qpact', 0);

NHoriz = 20;
horizon20.Nu = NHoriz;
horizon20.N = NHoriz;
horizon20.Ncu = NHoriz - 1;
horizon20.Ncy = NHoriz - 1;
ctrl20h = lincon(scd, 'reg', cost, horizon20, constraints, 'qpact', 0);
% do sim for both toolboxes
Tsim = 200;
times = 0:Ts:Tsim;
h = Ts / 10;

data.X2 = zeros(nx, Tsim / Ts);
```

```
data.X20 = zeros(nx, Tsim / Ts);
data.X2h = zeros(nx, Tsim / Ts);
data.X20h = zeros(nx, Tsim / Ts);
data.U2 = zeros(nu, Tsim / Ts);
data.U20 = zeros(nu, Tsim / Ts);
data.U2h = zeros(nu, Tsim / Ts);
data.U20h = zeros(nu, Tsim / Ts);
data.execTime2 = zeros(1, Tsim / Ts);
data.execTime20 = zeros(1, Tsim / Ts);
data.execTime2h = zeros(1, Tsim / Ts);
data.execTime20h = zeros(1, Tsim / Ts);
x2 = x0;
x20 = x0;
x2h = x0;
x20h = x0;
dataIdx = 1;

for t = times
  tic;
  u2 = ctrl2.evaluate(x2);
  data.execTime2(dataIdx) = toc;
  data.U2(:, dataIdx) = u2;
  [~, Xode2] = ode45(@ (t, x) scdynamics(t, x, u2), [t + h:h:t + Ts], x2);
  x2 = Xode2(end, :)';
  data.X2(:, dataIdx) = x2;

  tic;
  u20 = ctrl20.evaluate(x20);
  data.execTime20(dataIdx) = toc;
  data.U20(:, dataIdx) = u20;
  [~, Xode20] = ode45(@ (t, x) scdynamics(t, x, u20), [t + h:h:t + Ts], x20);
  x20 = Xode20(end, :)';
  data.X20(:, dataIdx) = x20;

  tic;
  u2h = eval(ctrl2h, x2h);
  data.execTime2h(dataIdx) = toc;
  data.U2h(:, dataIdx) = u2h;
  [~, Xode2h] = ode45(@ (t, x) scdynamics(t, x, u2h), [t + h:h:t + Ts], x2h);
  x2h = Xode2h(end, :)';
  data.X2h(:, dataIdx) = x2h;

  tic;
  u20h = eval(ctrl20h, x20h);
  data.execTime20h(dataIdx) = toc;
  data.U20h(:, dataIdx) = u20h;
  [~, Xode20h] = ode45(@ (t, x) scdynamics(t, x, u20h), [t + h:h:t + Ts],
x20h);
  x20h = Xode20h(end, :)';
  data.X20h(:, dataIdx) = x20h;

  dataIdx = dataIdx + 1;
end
```

```
% some other Hybrid toolbox stuff?
rmpath(genpath('/home/akshatd/mine/umich/sem2/740/toolboxes/hybtbx-linux/'))

% plot MPT

figure();
sgtitle("MPT: x_1, x_2, x_3");

for i = 1:3
  subplot(3, 1, i);
  plot(times, data.X2(i, :), 'DisplayName', 'N=2');
  grid on;
  hold on;
  plot(times, data.X20(i, :), 'DisplayName', 'N=20');
  xlabel('t [s]');
  ylabel("x_" + num2str(i));
  legend('Location', 'best');
end

snapnow;

figure();
sgtitle("MPT: x_4, x_5, x_6");

for i = 4:6
  subplot(3, 1, i - 3);
  plot(times, data.X2(i, :), 'DisplayName', 'N=2');
  grid on;
  hold on;
  plot(times, data.X20(i, :), 'DisplayName', 'N=20');
  xlabel('t [s]');
  ylabel("x_" + num2str(i));
  legend('Location', 'best');
end

snapnow;

figure();
sgtitle("MPT: u_1, u_2, u_3");

for i = 1:3
  subplot(3, 1, i);
  plot(times, data.U2(i, :), 'DisplayName', 'N=2');
  grid on;
  hold on;
  plot(times, data.U20(i, :), 'DisplayName', 'N=20');
  xlabel('t [s]');
  ylabel("u_" + num2str(i));
  ylim([-0.12, 0.12]);
  legend('Location', 'best');
end

snapnow;
```

```
figure();
sgtitle('MPT: Computation Time');
plot(times, data.execTime2, 'DisplayName', 'N=2');
grid on;
hold on;
plot(times, data.execTime20, 'DisplayName', 'N=20');
xlabel('t [s]');
ylabel('Time [s]');
legend('Location', 'best');
snapnow;

% plot Hybrid

figure();
sgtitle("Hybrid Toolbox: x_1, x_2, x_3");

for i = 1:3
  subplot(3, 1, i);
  plot(times, data.X2h(i, :), 'DisplayName', 'N=2');
  grid on;
  hold on;
  plot(times, data.X20h(i, :), 'DisplayName', 'N=20');
  xlabel('t [s]');
  ylabel("x_" + num2str(i));
  legend('Location', 'best');
end

snapnow;

figure();
sgtitle("Hybrid Toolbox: x_4, x_5, x_6");

for i = 4:6
  subplot(3, 1, i - 3);
  plot(times, data.X2h(i, :), 'DisplayName', 'N=2');
  grid on;
  hold on;
  plot(times, data.X20h(i, :), 'DisplayName', 'N=20');
  xlabel('t [s]');
  ylabel("x_" + num2str(i));
  legend('Location', 'best');
end

snapnow;

figure();
sgtitle("Hybrid Toolbox: u_1, u_2, u_3");

for i = 1:3
  subplot(3, 1, i);
  plot(times, data.U2h(i, :), 'DisplayName', 'N=2');
  grid on;
  hold on;
  plot(times, data.U20h(i, :), 'DisplayName', 'N=20');
```

```
  xlabel('t [s]');
  ylabel("u_" + num2str(i));
  ylim([-0.12, 0.12]);
  legend('Location', 'best');
end

snapnow;

figure();
sgtitle('Hybrid Toolbox: Computation Time');
plot(times, data.execTime2h, 'DisplayName', 'N=2');
grid on;
hold on;
plot(times, data.execTime20h, 'DisplayName', 'N=20');
xlabel('t [s]');
ylabel('Time [s]');
legend('Location', 'best');
snapnow;
```

```
Hybrid Toolbox v.1.4.8 [February 03, 2024] - (C) 2003-2024 by Alberto
Bemporad <alberto.bemporad@imtlucca.it>
Hybrid Toolbox v.1.4.8 [February 03, 2024] - (C) 2003-2024 by Alberto
Bemporad <alberto.bemporad@imtlucca.it>
```

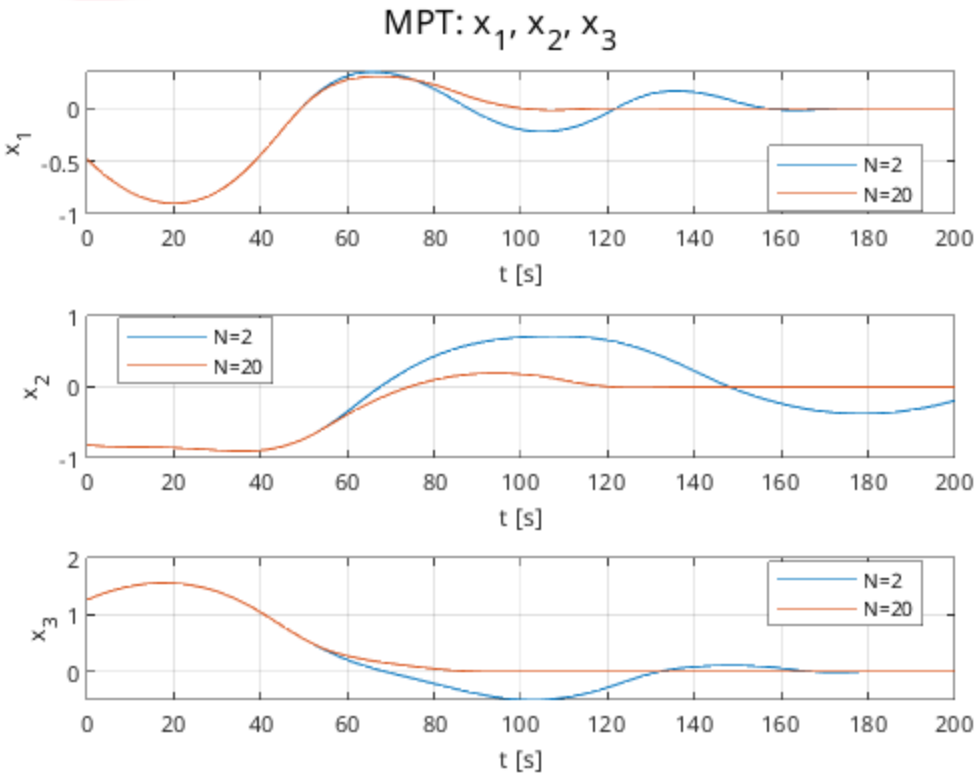
2.a scdynamics function

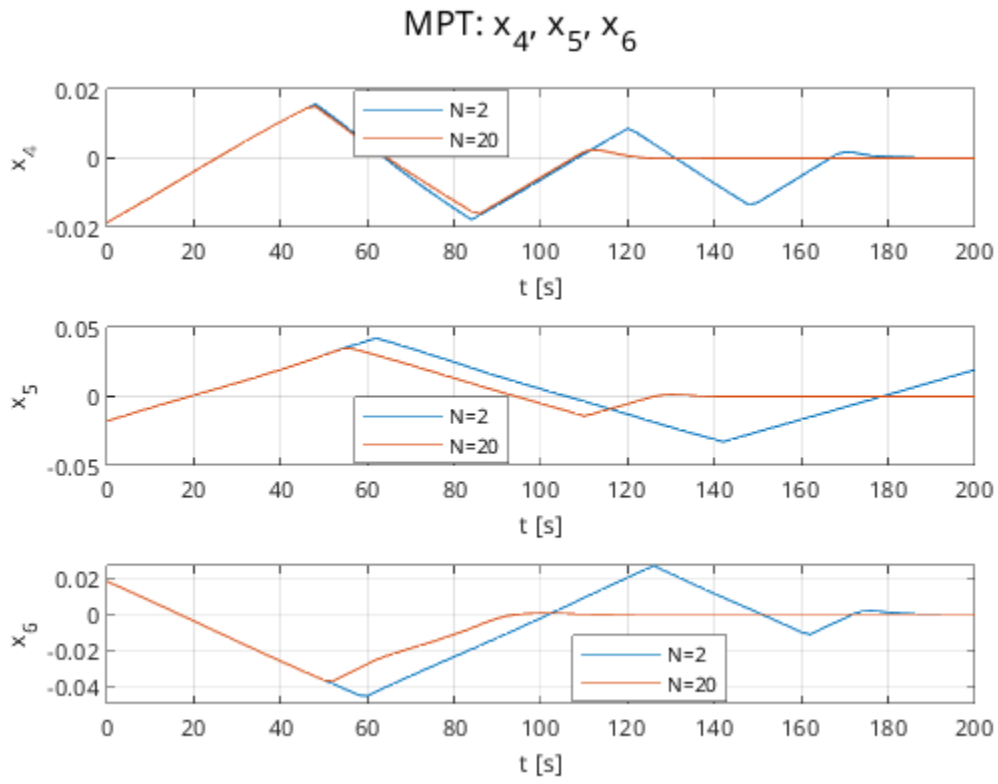
```
function [xdot] = scdynamics(t, x, u)
  J1 = 120;
  J2 = 100;
  J3 = 80;
  phi = x(1);
  theta = x(2);
  % psi = x(3);
  omega1 = x(4);
  omega2 = x(5);
  omega3 = x(6);
  M1 = u(1);
  M2 = u(2);
  M3 = u(3);
  angledot = 1 / cos(theta) * ...
    [
   cos(theta) sin(phi) * sin(theta) cos(phi) * sin(theta);
   0 cos(phi) * cos(theta) -sin(phi) * cos(theta);
   0 sin(phi) cos(phi);
   ] * ...
    [omega1; omega2; omega3];
  velocitydot = [
                 ((J2 - J3) * omega2 * omega3) / J1 + M1 / J1;
                 ((J3 - J1) * omega3 * omega1) / J2 + M2 / J2;
                 ((J1 - J2) * omega1 * omega2) / J3 + M3 / J3;
                 ];
  xdot = [
          angledot;
```

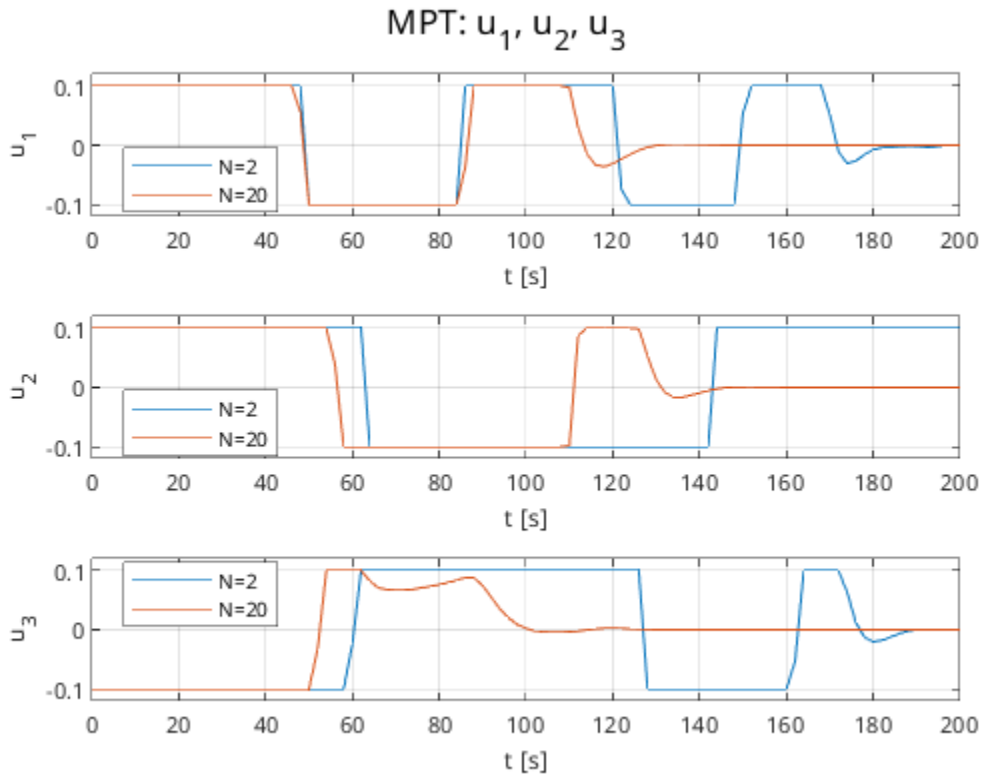
```
            velocitydot;
            ];
end
```

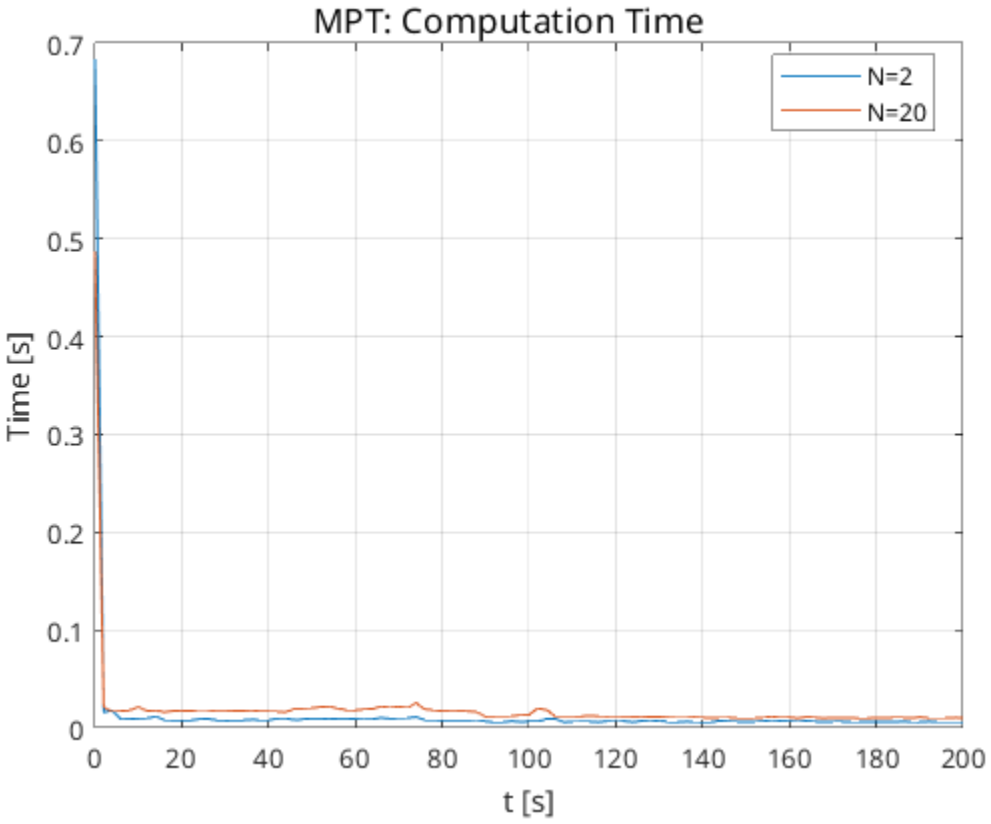
2-a

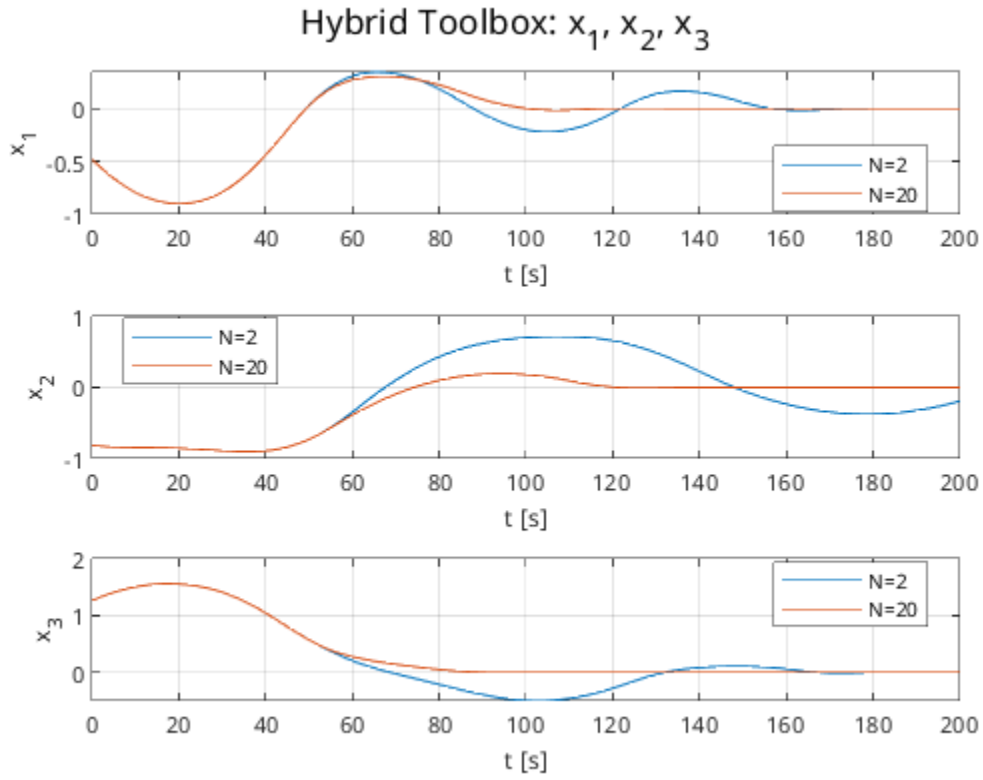
```
Ad =
     1     0     0     2     0     0
     0     1     0     0     2     0
     0     0     1     0     0     2
     0     0     0     1     0     0
     0     0     0     0     1     0
     0     0     0     0     0     1

Bd =
    0.016667            0            0
           0         0.02            0
           0            0        0.025
    0.016667            0            0
           0         0.02            0
           0            0        0.025
```

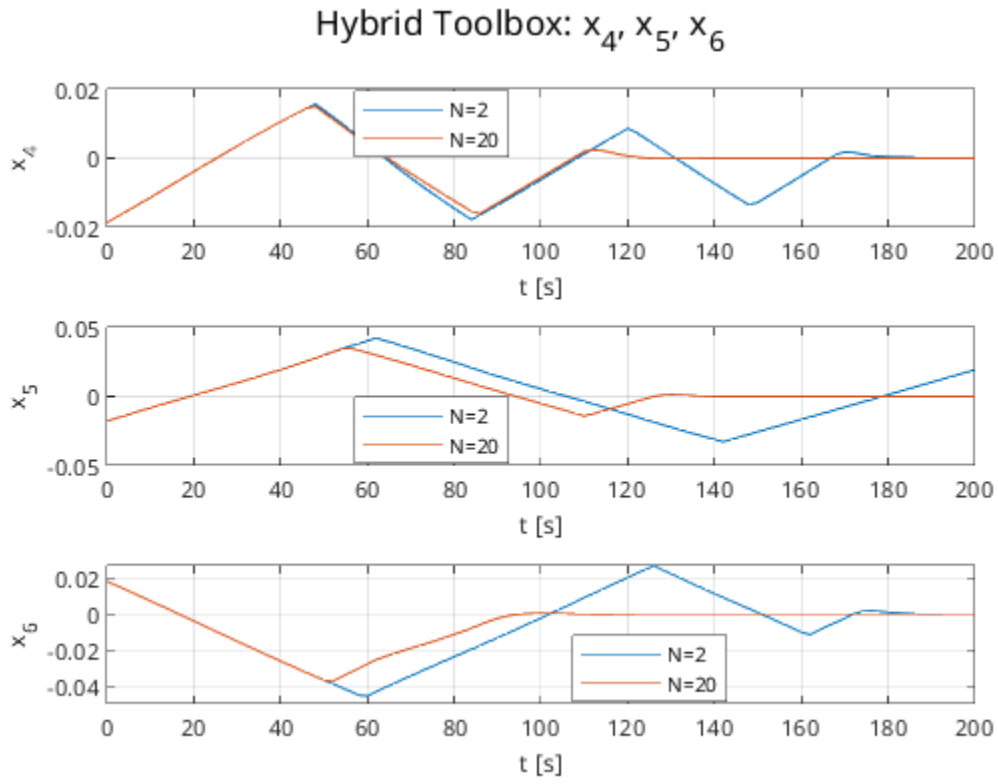


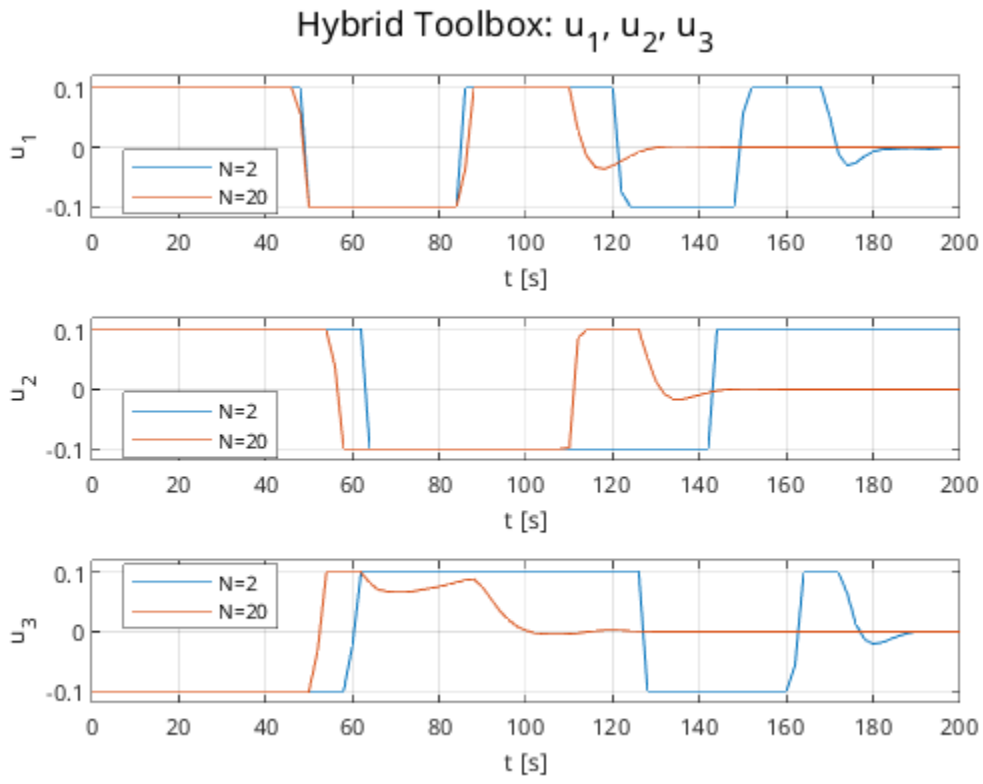

MPT: x4, x5, x6
x4
x5
x6
N=2
N=20
t [s]

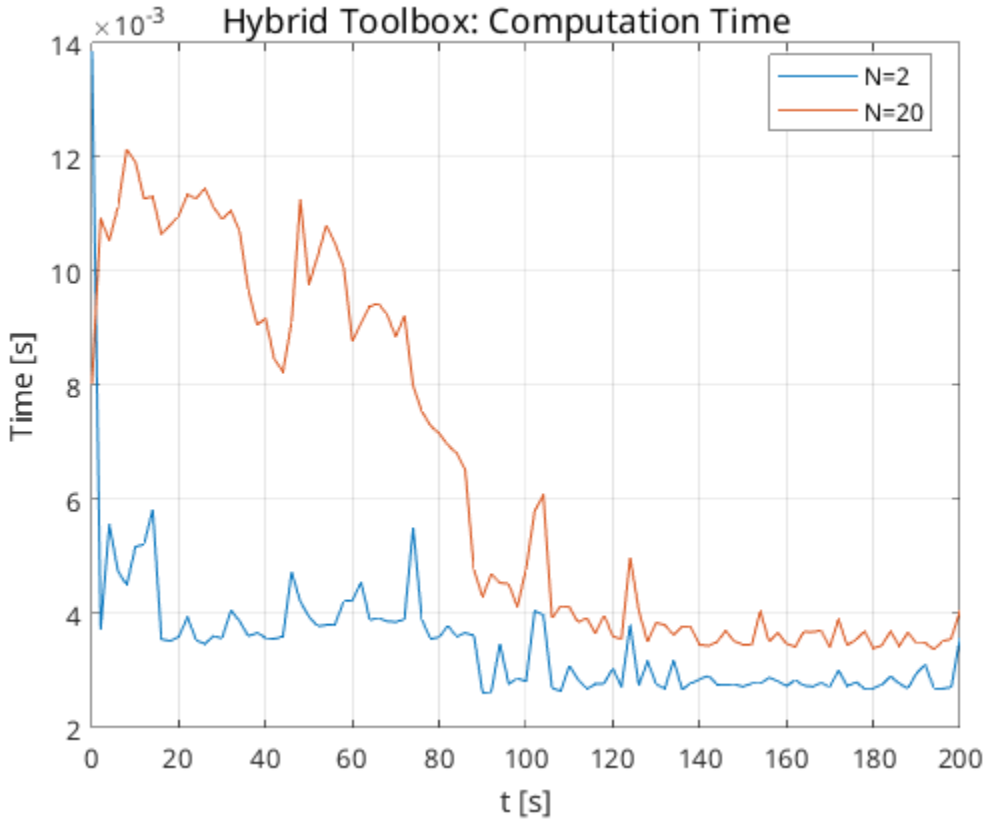

MPT: u1, u2, u3
u1
u2
u3
N=2
N=20
t [s]


MPT: Computation Time
Time [s]
t [s]
N=2
N=20
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20
40
60
80
100
120
140
160
180
200


Hybrid Toolbox: x1, x2, x3
x1
x2
x3
t [s]
N=2
N=20


Hybrid Toolbox: x4, x5, x6
x4
x5
x6
N=2
N=20
t [s]


Hybrid Toolbox: u1, u2, u3
u1
u2
u3
N=2
N=20
t [s]



Published with MATLAB® R2023b

Table of Contents


```
clc;
clear;
close all;
clear variables
format shortG;
```

3 Control of a car

3.a Discrete time model

```
disp('3.a Discrete time model');
% consts
m = 1891;
vx = 20;
Cf = -18e3;
Cr = -28e3;
lf = 1.5;
lr = 1.55;
Iz = 3200;
nx = 2;
nu = 1;

Ac = [2 * (Cf + Cr) / (m * vx) 2 * (Cf * lf - Cr * lr) / (m * vx) - vx;
      2 * (lf * Cf - lr * Cr) / (Iz * vx) 2 * (Cf * lf^2 + Cr * lr^2) / (Iz *
vx)];
Bc = [-2 * Cf / m;
      -2 * lf * Cf / Iz];
Cc = [1 / vx, lf / vx;
      1 / vx, -lr / vx;
      ];
Dc = [
      -1;
      0;
      ];
Ts = 20e-3;
card = c2d(ss(Ac, Bc, Cc, Dc), Ts, 'zoh');
Ad = card.A;
disp('Ad = ');
disp(Ad);
Bd = card.B;
disp('Bd = ');
```

```
disp(Bd);
Cd = card.C;
Dd = card.D;

% check if Ad is schur
if all(abs(eig(Ad)) < 1)
  disp('Ad is schur');
else
  disp('Ad is not schur');
end

if rank(ctrb(Ad, Bd)) == size(Ad, 1)
  disp('(Ad, Bd) has full rank, it is controllable');
else
  disp('(Ad, Bd) does not have full rank, it is not controllable');
end
```

```
3.a Discrete time model
Ad =
      0.95066     -0.36086
    0.0096662      0.93302

Bd =
      0.30929
      0.32805

Ad is schur
(Ad, Bd) has full rank, it is controllable
```

3.b simulate using MPT

```
disp('3.b simulate using MPT');
Ax = [Ad, Bd, 0 * Ad(:, 1);
      0 * Ad(1, :), 1, 0;
      0 * Ad(1, :), 0, 1;
      ];
Bx = [Bd;
      1;
      0;
      ];
% Cx is af and ar with stuff for u and r
Cx = [Cd, Dd, 0 * Cd(:, 1); ];
Dx = zeros(size(Cx, 1), size(Bx, 2));
Ex = [0, 1, 0, -1];
Qy = 3;
R = 0.1;
Q = Ex' * Qy * Ex;
P = zeros(size(Ax, 1));

N = 10; % Horizon

% MPT stuff, do once
tbxmanager restorepath
```

```
mpt_init
% create the model
modelMPT = LTISystem('A', Ax, 'B', Bx, 'C', Cx, 'D', Dx, 'Ts', Ts);
modelMPT.x.min = [-3, -1, -0.15, -inf];
modelMPT.x.max = -modelMPT.x.min;
modelMPT.u.min = -0.06;
modelMPT.u.max = -modelMPT.u.min;
modelMPT.y.min = [-0.1, -0.07];
modelMPT.y.max = -modelMPT.y.min;
modelMPT.u.penalty = QuadFunction(R);
modelMPT.x.penalty = QuadFunction(Q);
modelMPT.x.with('terminalPenalty');
modelMPT.x.terminalPenalty = QuadFunction(P);
ctrl = MPCController(modelMPT, N);
```

```
3.b simulate using MPT
Toolbox "mpt:3.2.1:all" added to the Matlab path.
Toolbox "mptdoc:3.0.4:all" added to the Matlab path.
Toolbox "cddmex:1.0.1:glnxa64" added to the Matlab path.
Toolbox "fourier:1.0:glnxa64" added to the Matlab path.
Toolbox "glpkmex:1.0:glnxa64" added to the Matlab path.
Toolbox "hysdel:2.0.6:glnxa64" added to the Matlab path.
Toolbox "lcp:1.0.3:glnxa64" added to the Matlab path.
Toolbox "yalmip:R20230609:all" added to the Matlab path.
Toolbox "sedumi:1.3:glnxa64" added to the Matlab path.
Toolbox "espresso:1.0:glnxa64" added to the Matlab path.
MPT searches for solvers on the path ...

 LINPROG ..................................... linprog.m
 QUADPROG .................................... quadprog.m
 GLPK ........................................... glpkcc
 CDD ............................................ cddmex
 LCP ............................................... lcp
 SEDUMI ...................................... sedumi.m
 PLCP ....................................... mpt_plcp.m
 MPQP ....................................... mpt_mpqp.m
 MPLP ....................................... mpt_mplp.m
 ENUMPLCP ................................ mpt_enum_plcp.m
 ENUMPQP ..................................... mpt_enum_pqp.m
 RLENUMPQP ................................... mpt_enum_pqp.m

MPT toolbox @version@ initialized...
Copyright (C) 2003-2024 by M. Kvasnica, C.N. Jones, and M. Herceg
For news, visit the MPT web page at http://control.ee.ethz.ch/~mpt/
            LP solver: LCP
            QP solver: LCP
          MILP solver: GLPK
 parametric LP solver: PLCP
 parametric QP solver: PLCP

These default options can be changed. See "help mptopt" for more details.
```

3.c Simulate using Hybrid toolbox

```
disp('3.c Simulate using Hybrid toolbox');
```

3.c Simulate using Hybrid toolbox

Both the toolboxes give the same results, so we can use either one for simulation, and it assures us that the simulation is correct. We also see that the constraints are satisfied.

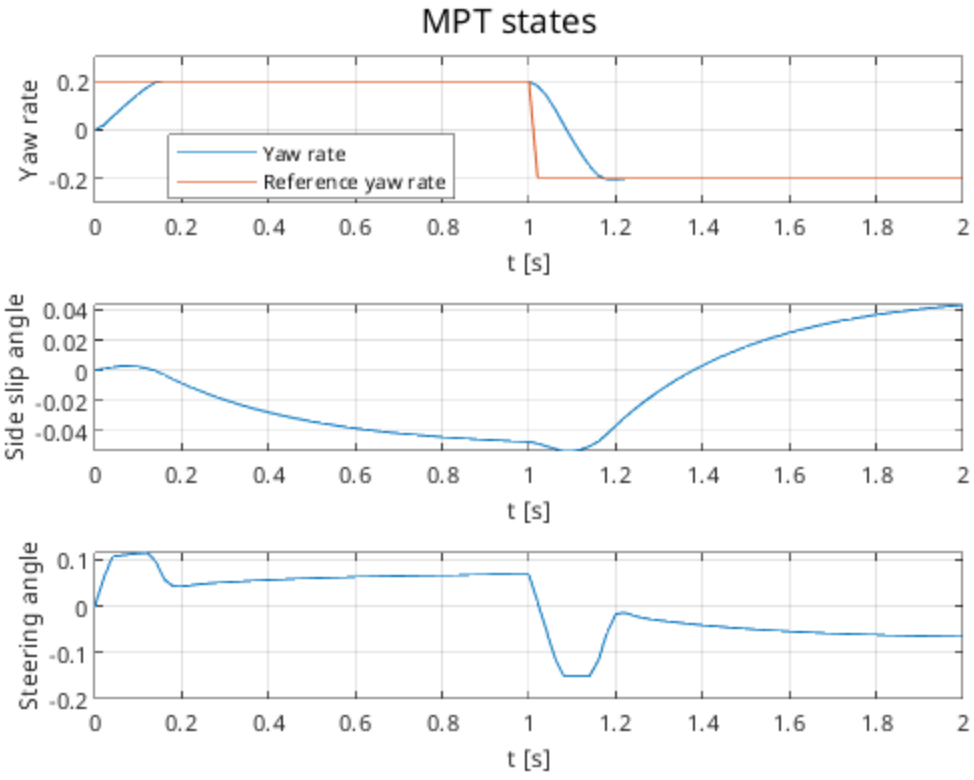
```
% Hybrid toolbox stuff
addpath( ...
  genpath('/home/akshatd/mine/umich/sem2/740/toolboxes/hybtbx-linux/'), ...
'-end')
clear cost constraints horizon;
% have to make a new C matrix to pass back x
Cxh = [Cx;
       eye(4);
       ];
Dxh = zeros(size(Cxh, 1), size(Bx, 2));

cost.Q = Q;
cost.R = R;
cost.P = P;
cost.rho = Inf;
% constraints.xmin = [-3, -1, -0.15, -inf];
% constraints.xmax = -constraints.xmin;
constraints.umin = -0.06;
constraints.umax = -constraints.umin;
% add the constraints for x here
constraints.ymin = [-0.1, -0.07, -3, -1, -0.15, -inf];
constraints.ymax = -constraints.ymin;

horizon.Nu = N;
horizon.N = N;
horizon.Ncu = N - 1;
horizon.Ncy = N - 1;
cardx = ss(Ax, Bx, Cxh, Dxh, Ts);
ctrlh = lincon(cardx, 'reg', cost, horizon, constraints, 'qpact', 0);

x0 = [0; 0; 0; 0.2];
y = Cx * x0;
yh = Cxh * x0;
Tsim = 2;
times = 0:Ts:Tsim;

data.X = zeros(4, Tsim / Ts + 1);
data.X(:, 1) = x0;
data.Xh = zeros(4, Tsim / Ts + 1);
data.Xh(:, 1) = x0;
data.Y = zeros(2, Tsim / Ts + 1);
data.Y(:, 1) = y;
data.Yh = zeros(6, Tsim / Ts + 1);
data.Yh(:, 1) = yh;
```

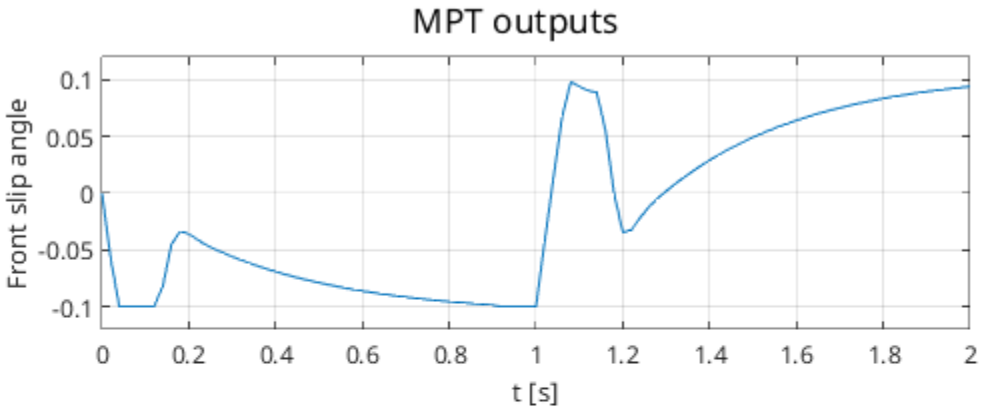
```
x = x0;
xh = x0;
dataIdx = 2;

for t = times(1: end - 1)

if t < 1
  x(4) = 0.2;
  xh(4) = 0.2;
else
  x(4) = -0.2;
  xh(4) = -0.2;
end

[u, feasible, openloop] = ctrl.evaluate(x);
% data.U(:, dataIdx) = u;
x = Ax * x + Bx * u;
data.X(:, dataIdx) = x;
data.Y(:, dataIdx) = Cx * x;
% calc side slip angle
data.X(1, dataIdx) = x(1) / vx;

uh = eval(ctrlh, xh);
xh = Ax * xh + Bx * uh;
data.Xh(:, dataIdx) = xh;
data.Yh(:, dataIdx) = Cxh * xh;
% calc side slip angle
data.Xh(1, dataIdx) = xh(1) / vx;

dataIdx = dataIdx + 1;
end

% some other Hybrid toolbox stuff?
rmpath(genpath('/home/akshatd/mine/umich/sem2/740/toolboxes/hybtbx-linux/'))

% plot MPT
figure();
sgtitle("MPT states");

subplot(3, 1, 1);
plot(times, data.X(2, :), "DisplayName", "Yaw rate");
grid on;
hold on;
plot(times, data.X(4, :), "DisplayName", "Reference yaw rate");
xlabel('t [s]');
ylabel("Yaw rate");
ylim([-0.3, 0.3]);
legend("location", "best");

subplot(3, 1, 2);
plot(times, data.X(1, :));
grid on;
xlabel('t [s]');
ylabel("Side slip angle");
```

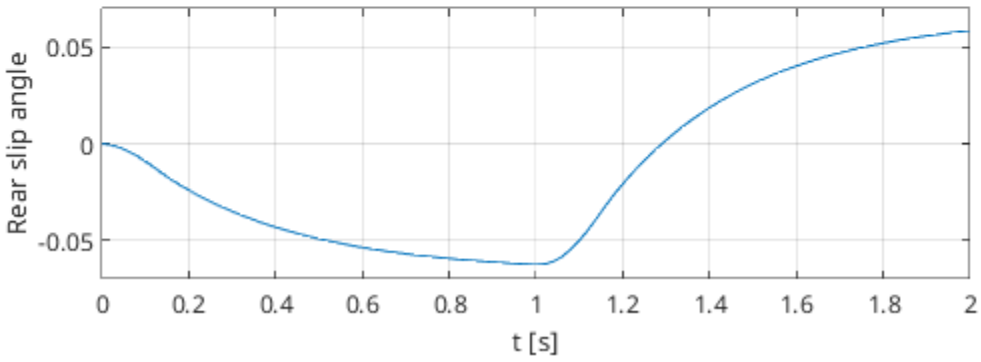
```
subplot(3, 1, 3);
plot(times, data.X(3, :));
grid on;
xlabel('t [s]');
ylabel("Steering angle");

snapnow;

figure();
sgtitle("MPT outputs");

subplot(2, 1, 1);
plot(times, data.Y(1, :));
grid on;
xlabel('t [s]');
ylabel("Front slip angle");
ylim([-0.12, 0.12]);

subplot(2, 1, 2);
plot(times, data.Y(2, :));
grid on;
xlabel('t [s]');
ylabel("Rear slip angle");
ylim([-0.07, 0.07]);

snapnow;

% plot Hybrid toolbox

figure();
sgtitle("Hybrid MPC states");

subplot(3, 1, 1);
plot(times, data.Xh(2, :), "DisplayName", "Yaw rate");
grid on;
hold on;
plot(times, data.Xh(4, :), "DisplayName", "Reference yaw rate");
xlabel('t [s]');
ylabel("Yaw rate");
ylim([-0.3, 0.3]);
legend("location", "best");

subplot(3, 1, 2);
plot(times, data.Xh(1, :));
grid on;
xlabel('t [s]');
ylabel("Side slip angle");

subplot(3, 1, 3);
plot(times, data.Xh(3, :));
grid on;
xlabel('t [s]');
ylabel("Steering angle");
```

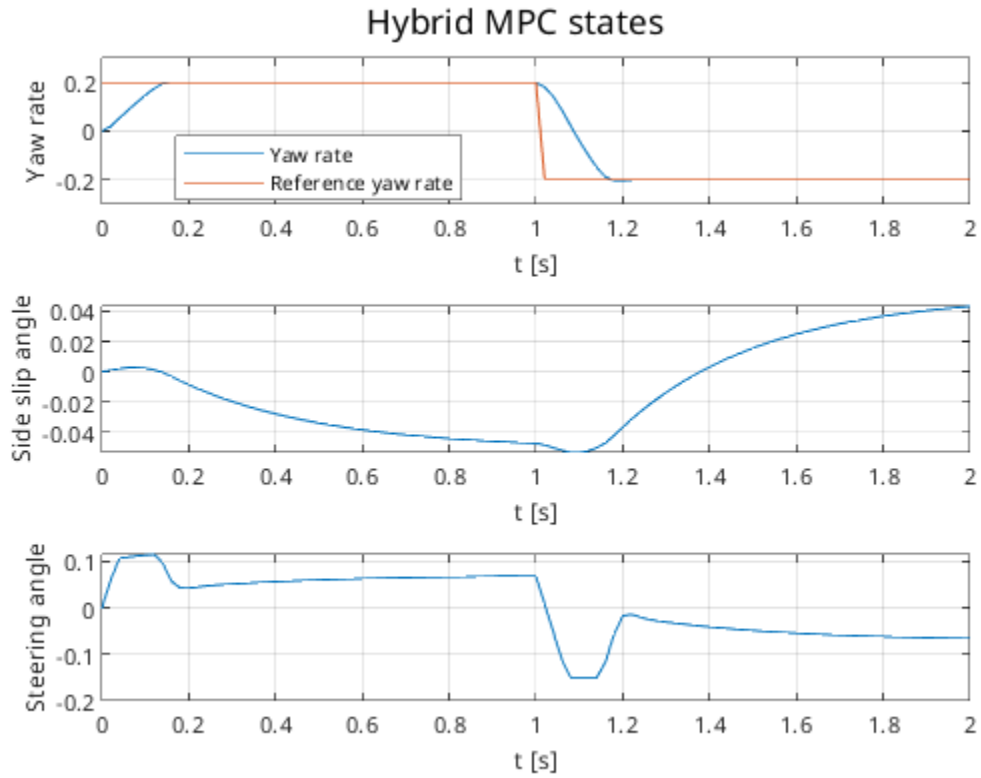
```
snapnow;

figure();
sgtitle("Hybrid MPC outputs");

subplot(2, 1, 1);
plot(times, data.Yh(1, :));
grid on;
xlabel('t [s]');
ylabel("Front slip angle");
ylim([-0.12, 0.12]);

subplot(2, 1, 2);
plot(times, data.Yh(2, :));
grid on;
xlabel('t [s]');
ylabel("Rear slip angle");
ylim([-0.07, 0.07]);

snapnow;
```

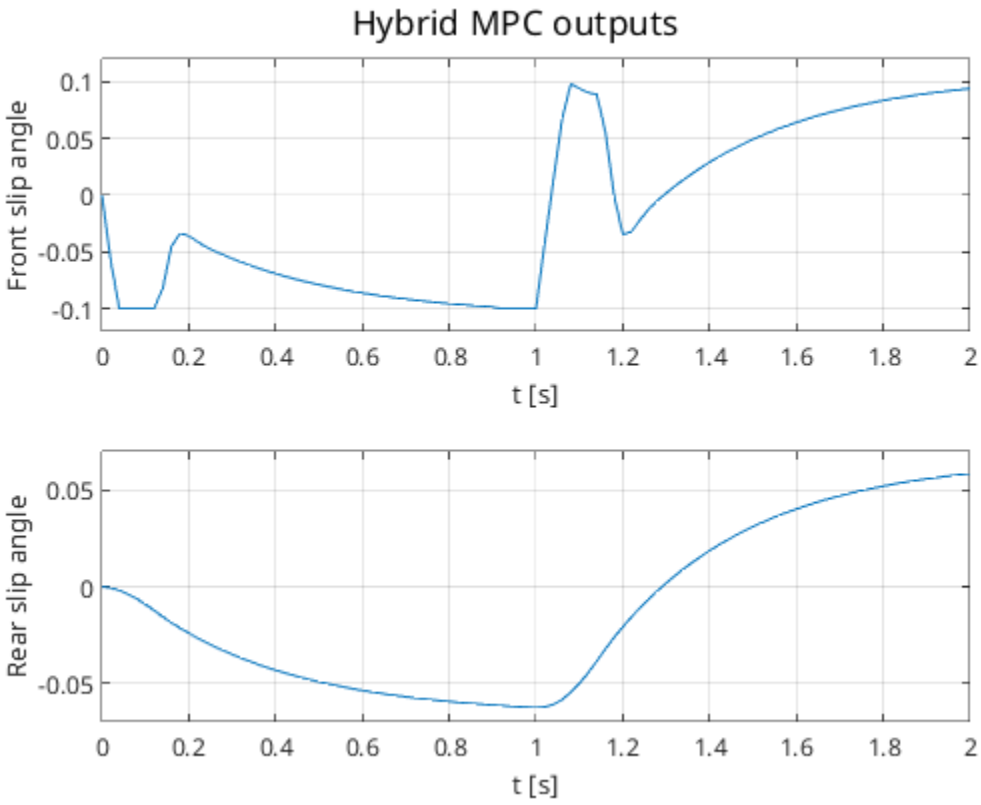
Hybrid Toolbox v.1.4.8 [February 03, 2024] - (C) 2003-2024 by Alberto Bemporad <alberto.bemporad@imtlucca.it>




MPT outputs
Front slip angle
0.1
0.05
0
-0.05
-0.1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
t [s]


Rear slip angle
0.05
0
-0.05
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
t [s]


Hybrid MPC states
Yaw rate
0.2
0
-0.2
Yaw rate
Reference yaw rate
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
t [s]
Side slip angle
0.04
0.02
0
-0.02
-0.04
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
t [s]
Steering angle
0.1
0
-0.1
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
t [s]



Published with MATLAB® R2023b

HW3 EX4.a

Saturday, February 24, 2024 4:51 PM

State space equations: $\dot{X} = A_C X$

$$: \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

Numerical A_C in code output

Table of Contents


```
clc;
clear;
close all;
clear variables
format shortG;
```

4 Spacecraft Motion

```
% consts
n = 0.00114;
```

4.a Open loop dynamics

```
disp('4.a Open loop dynamics');
Ac = [0, 0, 0, 1, 0, 0;
      0, 0, 0, 0, 1 0;
      0, 0, 0, 0, 0, 1;
      3 * n^2, 0, 0, 0, 2 * n, 0;
      0, 0, 0, -2 * n, 0, 0;
      0, 0, -n^2, 0, 0, 0;
      ];
disp('Ac = ');
disp(Ac);

eigAc = eig(Ac);
% check if all eigenvalues are in left half plane
notLeftHalfPlane = eigAc(real(eigAc) >= 0);

if size(notLeftHalfPlane) ~= 0
  fprintf('model is open loop unstable, not all eigenvalues in left half
plane:');
  display(notLeftHalfPlane);
else
  fprintf('model is open loop stable\n');
end
```

```
4.a Open loop dynamics
Ac =
            0            0            0            1            0            0
            0            0            0            0            1            0
```

```
            0            0            0            0            0            1
   3.8988e-06            0            0            0      0.00228            0
            0            0            0     -0.00228            0            0
            0            0  -1.2996e-06            0            0            0

model is open loop unstable, not all eigenvalues in left half plane:
notLeftHalfPlane =

            0 +           0i
            0 +           0i
            0 +    0.00114i
            0 -    0.00114i
            0 +    0.00114i
            0 -    0.00114i
```

4.b Observability

```
disp('4.b Observability')
Cc = [1, 0, 0, 0, 0, 0;
      0, 1, 0, 0, 0, 0;
      0, 0, 1, 0, 0, 0;
      ];
disp('C = ');
disp(Cc);

if rank(obsv(Ac, Cc)) == size(Ac, 1)
  disp('(C, A) has full rank, it is observable');
else
  disp('(C, A) does not have full rank, it is not observable');
end
```

```
4.b Observability
C =
     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0

(C, A) has full rank, it is observable
```

4.c Discrete time model

```
disp('4.c Discrete time model');

Ts = 30;
% Bc = zeros(size(Ac, 1), 1);
% Dc = zeros(size(Cc, 1), 1);

[Ad, ~, Cd, ~] = c2dm(Ac, [], Cc, [], Ts, 'zoh');
disp('Ad = ');
disp(Ad);
disp('Cd = ');
disp(Cd);
```

```
4.c Discrete time model
Ad =
       1.0018            0            0       29.994       1.0259            0
  -3.9999e-05            1            0      -1.0259       29.977            0
            0            0      0.99942            0            0       29.994
   0.00011694            0            0      0.99942     0.068387            0
  -3.9998e-06            0            0    -0.068387      0.99766            0
            0            0   -3.898e-05            0            0      0.99942

Cd =
     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
```

4.d Moving Horizon Observer

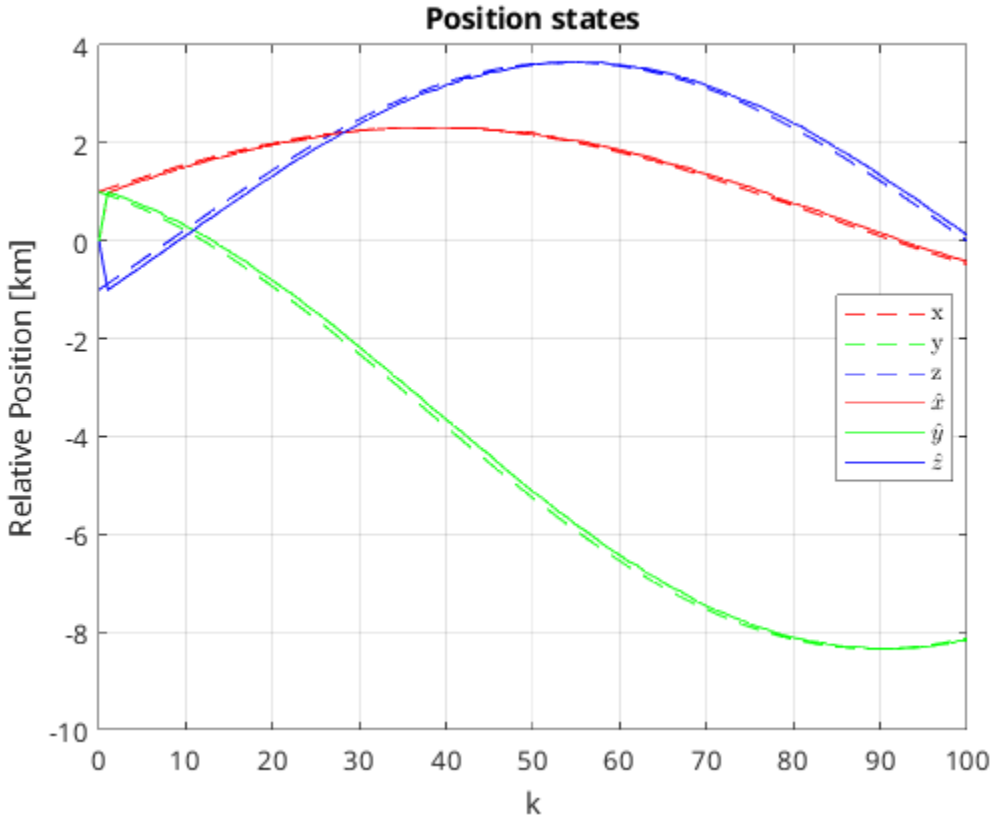
```
disp('4.d Moving Horizon Observer');
```

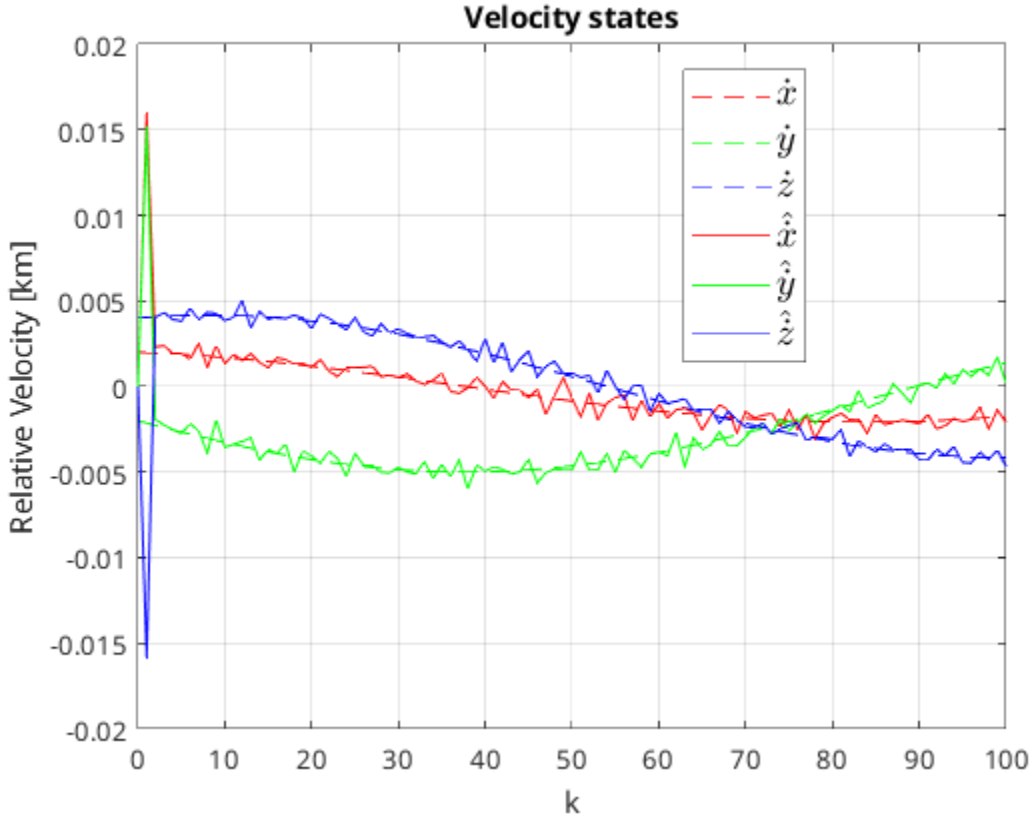
```
4.d Moving Horizon Observer
```

We can see that the moving horizon observer manages to track the position of the spacecraft quite well. The velocity states are also tracked well, but the estimate is a little noisy. There is also a small error in the beginning, because we had to use the initial states specified in the problem instead of using the a-priori state estimate as shown in the notes.

```
P = diag([1, 1, 1, 0.001, 0.001, 0.001]);
R = diag([0.01, 0.01, 0.01]);
Q = diag([0.001, 0.001, 0.001, 0.1, 0.1, 0.1]);

X0 = [1; 1; -1; 0.002; -0.002; 0.004];
X0_hat = [0; 0; 0; 0; 0; 0; ];

X = X0;
X_hat = X0_hat;
Y = Cd * X + 0.01 * randn(3, 1);

data.X = zeros(6, 101);
data.X(:, 1) = X;
data.X_hat = zeros(6, 101);
data.X_hat(:, 1) = X_hat;

for k = 1:100
  % observe
  Y = Cd * X + 0.01 * randn(3, 1);

  % estimate
  X_hat_bar = Ad * X_hat;
  P_bar = Q + Ad * P * Ad';
  L = P_bar * Cd' * inv(Cd * P_bar * Cd' + R);
  X_hat = X_hat_bar + L * (Y - Cd * X_hat_bar);
  P = P_bar - P_bar * Cd' * inv(Cd * P_bar * Cd' + R) * Cd * P_bar;

  % propagate
  X = Ad * X;
```

```
  data.X(:, k + 1) = X;
  data.X_hat(:, k + 1) = X_hat;
end

% plot
figure;
plot(0:100, data.X(1, :), '--r', "DisplayName", "x");
grid on;
hold on;
plot(0:100, data.X(2, :), '--g', "DisplayName", "y");
plot(0:100, data.X(3, :), '--b', "DisplayName", "z");
plot(0:100, data.X_hat(1, :), 'r', "DisplayName", "$\hat{x}$");
plot(0:100, data.X_hat(2, :), 'g', "DisplayName", "$\hat{y}$");
plot(0:100, data.X_hat(3, :), 'b', "DisplayName", "$\hat{z}$");
xlabel('k');
ylabel('Relative Position [km]');
legend("Location", "best", "Interpreter", "latex");
title('Position states');

figure;
plot(0:100, data.X(4, :), '--r', "DisplayName", "$\dot{x}$");
grid on;
hold on;
plot(0:100, data.X(5, :), '--g', "DisplayName", "$\dot{y}$");
plot(0:100, data.X(6, :), '--b', "DisplayName", "$\dot{z}$");
plot(0:100, data.X_hat(4, :), 'r', "DisplayName", "$\hat{\dot{x}}$");
plot(0:100, data.X_hat(5, :), 'g', "DisplayName", "$\hat{\dot{y}}$");
plot(0:100, data.X_hat(6, :), 'b', "DisplayName", "$\hat{\dot{z}}$");
xlabel('k');
ylabel('Relative Velocity [km]');
legend("Location", "best", "Interpreter", "latex", "FontSize", 14);
title('Velocity states');
```


Position states
Relative Position [km]
k
x
y
z
$\hat{x}$
$\hat{y}$
$\hat{z}$


Velocity states
Relative Velocity [km]
k
$\dot{x}$
$\dot{y}$
$\dot{z}$
$\hat{\dot{x}}$
$\hat{\dot{y}}$
$\hat{\dot{z}}$

Published with MATLAB® R2023b

HW3 EX5.a

Saturday, February 24, 2024 5:11 PM

Equations in State Space form: $\dot{X} = A_C X + B_C U$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 2n \\ 0 & 0 & -2n & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1/m & 0 \\ 0 & 1/m \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix}$$

Numerical values of A_C, B_C in code output

HW3 EX5.b

Saturday, February 24, 2024 12:04 AM

- State transition formula : $x_k = A^k x_0 + \sum_{i=0}^{k-1} A^{k-1-i} B u_i$

- We want : $X_N = P x_0 + R U$ where $U = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \} N \times n_u$
- Dimensions $P: n_x \left[\quad \right]_{n_x}$ $R: n_x \left[\quad \right]_{N \times n_u}$

- Just by inspection, $P = A^N$
- Expanding $\sum_{i=0}^{k-1} A^{k-1-i} B u_i$, for $k=N$ $\quad k-1-(k-1)$ $\quad k-x-x+1$

$$\Rightarrow A^{N-1} B u_0 + A^{N-2} B u_1 \ldots\ldots AB u_{N-2} + B u_{N-1}$$

$$\Rightarrow \begin{bmatrix} A^{N-1}B & A^{N-2}B & \ldots\ldots & AB & B \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{bmatrix}$$

Therefore $R = \begin{bmatrix} A^{N-1}B & A^{N-2}B & \ldots\ldots & AB & B \end{bmatrix}$

- With $A = A_d$, $B = B_d$ found after discretizing the model

HW3 EX5.c

Saturday, February 24, 2024 12:24 AM

Cost function $\tilde{J}_N = q\, x_N^T x_N + \sum_{k=0}^{N-1} 1^T \xi_i$, $1^T = [1 \;\; 1]$

From part 5.b, $X_N = P X_0 + RU$, $X_N^T = X_0^T P^T + U^T R^T$

Substituting

$$\tilde{J}_N = q\left(X_0^T P^T + U^T R^T\right)\left(PX_0 + RU\right) + \sum_{k=0}^{N-1} [1 \;\; 1]\, \xi_i$$

$$= q\left(X_0^T P^T P X_0 + X_0^T P^T RU + U^T R^T P X_0 + U^T R^T RU\right) + [1 \;\; 1]\xi_0 + [1 \;\; 1]\xi_1 + \cdots [1 \;\; 1]\xi_{N-1}$$

Since $\tilde{J}_N$ is scalar, q is scalar $\Rightarrow X_0^T P^T RU$ and $U^T R^T P X_0$ are scalar

for a scalar s, $s^T = s$ so, $(X_0^T P^T RU)^T = X_0^T P^T RU$

$$\Rightarrow U^T R^T P X_0 = X_0^T P^T RU$$

$$\therefore \tilde{J}_N = q\left(X_0^T P^T P X_0 + X_0^T P^T RU + X_0^T P^T RU + U^T R^T RU\right) + [1 \;\; 1]\xi_0 + [1 \;\; 1]\xi_1 + \cdots + [1 \;\; 1]\xi_{N-1}$$

$$= q\, X_0^T P^T P X_0 + 2q\, X_0^T P^T RU + q U^T R^T RU + [1 \;\; 1]\xi_0 + [1 \;\; 1]\xi_1 + \cdots + [1 \;\; 1]_{N-1}$$

(const not a function of z)

$$= q\, U^T R^T RU + 2q\, X_0^T P^T RU + [1 \;\; 1]\xi_0 + [1 \;\; 1]\xi_1 + \cdots [1 \;\; 1]_{N-1}$$

$$\tilde{J}_N = q\, U^T R^T RU + 2q \underbrace{X_0^T P^T R}_{1 \times (N)(n_u)} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}_{(N)(n_u)} + \underbrace{[[1 \;\; 1] \cdots [1 \;\; 1]]}_{N\ [1\ 1]} \begin{bmatrix} \xi_0 \\ \xi_1 \\ \vdots \\ \xi_{N-1} \end{bmatrix}_{N(n_u)} \Big\}(N)(N_u) \text{ where } n_u = 2$$

$= \#\text{cols in } 1^T$

$= \#\text{cols in } [1 \;\; 1]$

$$= q\, U^T R^T RU + 2q\, X_0^T P^T R \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} + 1_{1 \times (N)(n_u)} \begin{bmatrix} \xi_0 \\ \xi_1 \\ \vdots \\ \xi_{N-1} \end{bmatrix}$$

$$= q\, U^T R^T RU + \begin{bmatrix} 2q\, X_0^T P^T R & 1_{1 \times (N)(n_u)} \end{bmatrix} z$$

$RU = R \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$ Let $R' = \begin{bmatrix} R & \begin{matrix} 0 \cdots 0 \\ \vdots \;\; \vdots \\ 0 \cdots 0 \end{matrix} \end{bmatrix}_{(N)(N_u)}$, then $R'z = [R \;\; 0] \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ \xi_0 \\ \xi_1 \\ \vdots \\ \xi_{N-1} \end{bmatrix} = RU$

$$\tilde{J}_N = q\, z^T R'^T R' z + [2q X_0^T P^T R \;\; [1 \;\; 1] \cdots [1 \;\; 1]]z$$

$$= \frac{1}{2} z^T \underbrace{(2q R'^T R')}_{\Gamma_1} z + \underbrace{[2q X_0^T P^T R \;\; 1_{1 \times (N)(n_u)}]}_{\Gamma_2} z$$

$$\therefore \Gamma_1 = 2q R'^T R'$$

$$= 2q \begin{bmatrix} R^T \\ 0_{(N)(n_u) \times n_x} \end{bmatrix} \begin{bmatrix} R & 0_{n_x \times (N)(n_u)} \end{bmatrix} = 2q \begin{bmatrix} R^T R & 0_{(N)(n_u) \times (N)(n_u)} \\ 0_{(N)(n_u) \times (N)(n_u)} & 0_{(N)(n_u) \times (N)(n_u)} \end{bmatrix} //$$

$$\Gamma_2 = [2q X_0^T P^T R \;\; 1_{1 \times (N)(n_u)}] //$$

HW3 EX5.d

Saturday, February 24, 2024 1:33 PM

$$J_N = q\, x_N^T x_N + \sum_{k=0}^{N-1} \|u_k\|_1$$

and $$\tilde{J}_N = q\, x_N^T x_N + \sum_{k=0}^{N-1} 1^T \xi_k$$

How are $\sum_{k=0}^{N-1} \|u_k\|_1$ and $\sum_{k=1}^{N-1} 1^T \xi_k$ related?

$$\|u_k\|_1 = |u_{kx}| + |u_{ky}| \quad , \quad 1^T \xi_k = \xi_{kx} + \xi_{ky} \qquad -①$$

Given that $\xi_{N-1} \geq u_{N-1}$ and $\xi_{N-1} \geq -u_{N-1}$, $\Rightarrow \xi_{N-1} \geq |u_{N-1}|$, element wise

$$\Rightarrow \left.\begin{aligned} \xi_{N-1,x} &\geq |u_{N-1,x}| \\ \xi_{N-1,y} &\geq |u_{N-1,y}| \end{aligned}\right\} -②$$

Using ① and ②, $\xi_{kx} + \xi_{ky} \geq |u_{kx}| + |u_{ky}|$, Assuming Both ξ_{kx} and ξ_{ky} are +ve because they are constraints

$$\Rightarrow 1^T \xi_k \geq \|u_k\|_1 \text{ or } \|u_k\|_1 \leq 1^T \xi_k$$

When we minimize with respect to $1^T \xi_k$ the smallest $1^T \xi_k$ we can get can only be $= \|u_k\|_1$

$$\text{So, } \min_{\xi_0 \cdots \xi_{N-1}} \tilde{J} = \min_{\xi_0 \cdots \xi_{N-1}} q\, x_N^T x_N + \sum_{k=0}^{N-1} 1^T \xi_k$$

$$= q\, x_N^T x_N + \sum_{k=1}^{N-1} \|u_k\|_1$$

$$= J$$

$$\therefore \min_z \tilde{J} = \min_{u_0 \cdots u_{N-1}\, \xi_0 \cdots \xi_{N-1}} \tilde{J}$$

$$= \min_{u_0 \cdots u_{N-1}} \left(\min_{\xi_0 \cdots \xi_{N-1}} \tilde{J} \right)$$

$$= \min_{u_0 \cdots u_{N-1}} J \quad //$$

$\therefore$ minimizing $\tilde{J}$ wrt z minimizes J to produce the optimal control sequence $\{u_0^* \cdots u_{N-1}^*\}$

Table of Contents

5 Spacecraft Dynamics

```
clc;
clear;
close all;
clear variables
format shortG;

% consts
n = 1.107e-3;
m = 100;
```

5.a State space form

```
disp("5.a");
Ac = [
      0, 0, 1, 0;
      0, 0, 0, 1;
      3 * n^2, 0, 0, 2 * n;
      0, 0, -2 * n, 0;
      ];
Bc = [
      0, 0;
      0, 0;
      1 / m, 0;
      0, 1 / m;
      ];

disp("Ac = ");
disp(Ac);
disp("Bc = ");
disp(Bc)
eigAc = eig(Ac);
% check if all eigenvalues are in left half plane
notLeftHalfPlane = eigAc(real(eigAc) >= 0);

if size(notLeftHalfPlane) ~= 0
  fprintf('model is open loop unstable, not all eigenvalues in left half
plane:');
  display(notLeftHalfPlane);
else
```

```
  fprintf('model is open loop stable\n');
end

if rank(ctrb(Ac, Bc)) == size(Ac, 1)
  disp('(Ac, Bc) has full rank, it is controllable');
else
  disp('(Ac, Bc) has full rank, it is not controllable');
end
```

```
5.a
Ac =
            0            0            1            0
            0            0            0            1
   3.6763e-06            0            0     0.002214
            0            0    -0.002214            0

Bc =
            0            0
            0            0
         0.01            0
            0         0.01

model is open loop unstable, not all eigenvalues in left half plane:
notLeftHalfPlane =

            0 +          0i
   1.3553e-20 +   0.001107i
   1.3553e-20 -   0.001107i

(Ac, Bc) has full rank, it is controllable
```

5.b Discrete time

```
disp("5.b");
Ts = 20;
[Ad, Bd, ~, ~] = c2dm(Ac, Bc, [], [], Ts, 'zoh');
disp("Ad = ");
disp(Ad);
disp("Bd = ");
disp(Bd);
```

```
5.b
Ad =
       1.0007            0       19.998      0.44278
  -1.0852e-05            1     -0.44278       19.993
   7.3521e-05            0      0.99975     0.044276
  -1.6278e-06            0    -0.044276      0.99902

Bd =
       1.9999     0.029519
    -0.029519       1.9997
      0.19998    0.0044278
```

```
   -0.0044278      0.19993
```

5.e Minimize fuel consumption cost using inputs

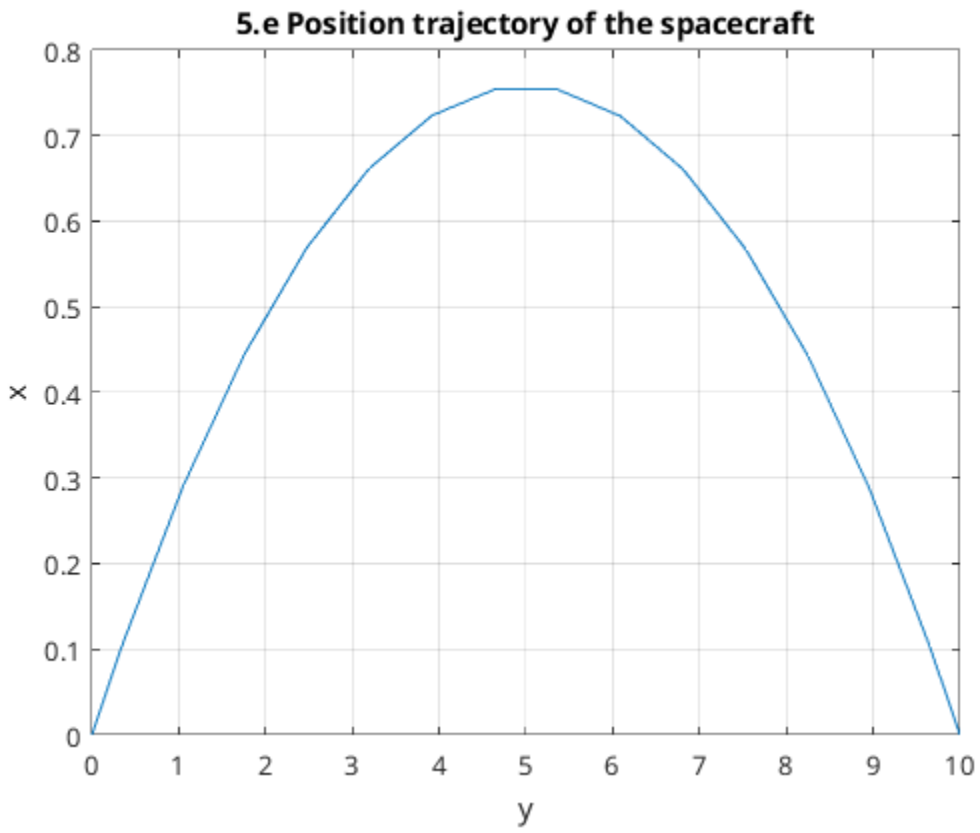
```
disp("5.e");
N = 15;
q = 10^4;
X0 = [0; 10; 0; 0];
nx = size(Ad, 1);
nu = size(Bd, 2);

% using var names from the question

% XN = P X0 + R U
% P = A^N
P = Ad^N;
% R = [A^(N-1) B, A^(N-2) B, ..., B]
% R = nx x (N)(nu)
R = zeros(nx, N * nu);

for i = 1:N
  R(:, (i - 1) * nu + 1:i * nu) = Ad^(N - i) * Bd;
end

% now Jtilde = (1/2)Z' GAMMA1 Z + GAMMA2  Z
% GAMMA1 = 2 q Rprime' Rprime
% = 2 q [R' R 0; 0 0]
GAMMA1 = 2 * q * ...
  [
 R' * R, zeros(N * nu, N * nu);
 zeros(N * nu, N * nu), zeros(N * nu, N * nu)
 ];

% GAMMA2 = 2 q X0' P' R ones(1, N*nu)
GAMMA2 = [2 * q * X0' * P' * R, ones(1, N * nu)];

% from the matlab docs, we minimize (1/2) x' H x + f' x in x = quadprog(H,f)
% where x = Z, H = GAMMA1, f = GAMMA2

% subject to Ax <= b, so
% cai >= U -> U <= cai -> U - cai <= 0
% cai >= -U -> -U <= cai -> -U - cai <= 0
% U's are in the first half of Z, cai's are in the second half of Z
% we need to set up the matrix A so that it minuses them off
A = [eye(N * nu), -eye(N * nu);
     -eye(N * nu), -eye(N * nu)];
b = zeros(2 * N * nu, 1);

Zopt = quadprog(GAMMA1, GAMMA2, A, b);
% get Uopt from Zopt in the format of nu * N
```

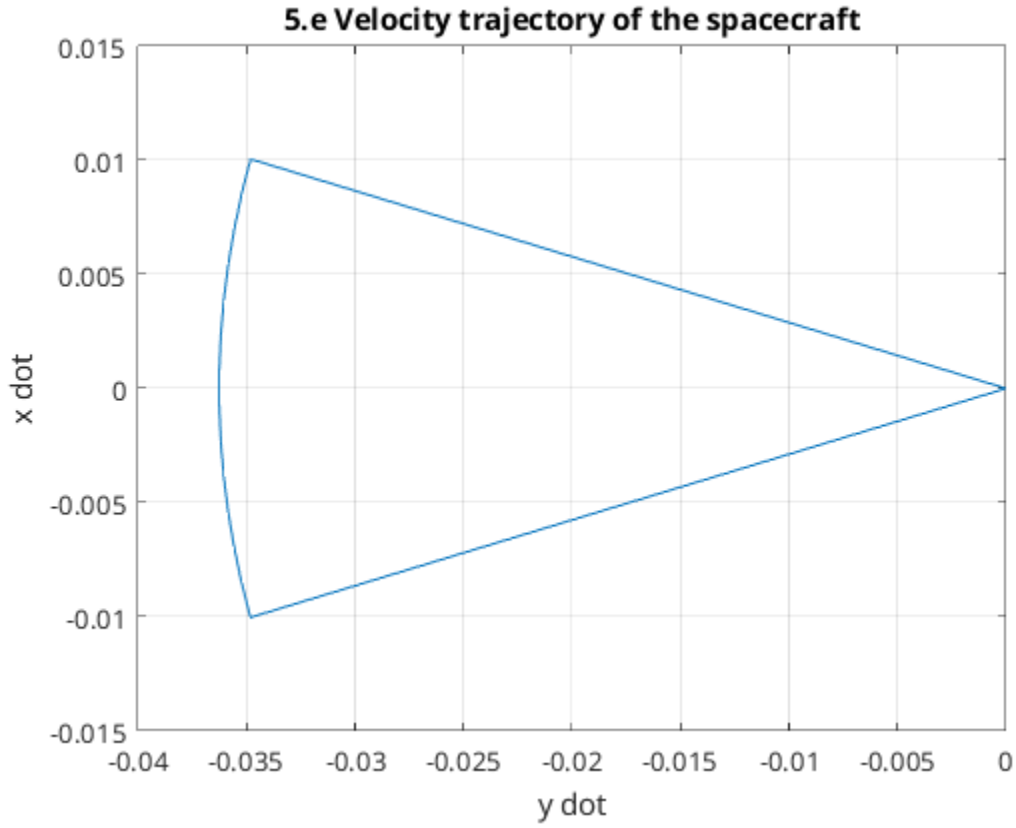
```
Uopt = reshape(Zopt(1:N * nu), nu, N);

data.X = zeros(nx, N + 1);
data.X(:, 1) = X0;

for k = 2:N
  data.X(:, k) = Ad * data.X(:, k - 1) + Bd * Uopt(:, k - 1);
end

% plot trajectory
figure();
plot(data.X(2, :), data.X(1, :));
grid on;
xlabel('y');
ylabel('x');
title('5.e Position trajectory of the spacecraft');

figure();
plot(data.X(4, :), data.X(3, :));
grid on;
xlabel('y dot');
ylabel('x dot');
title('5.e Velocity trajectory of the spacecraft');

% plot inputs
figure();
stairs(Ts * (1:N), Uopt(1, :), "DisplayName", "u_x");
grid on;
hold on;
stairs(Ts * (1:N), Uopt(2, :), "DisplayName", "u_y");
xlabel("Time (s)");
ylabel("Input");
xlim([0, Ts * N + 10]);
title("5.e Input trajectory of the spacecraft");
legend("Location", "best");
```

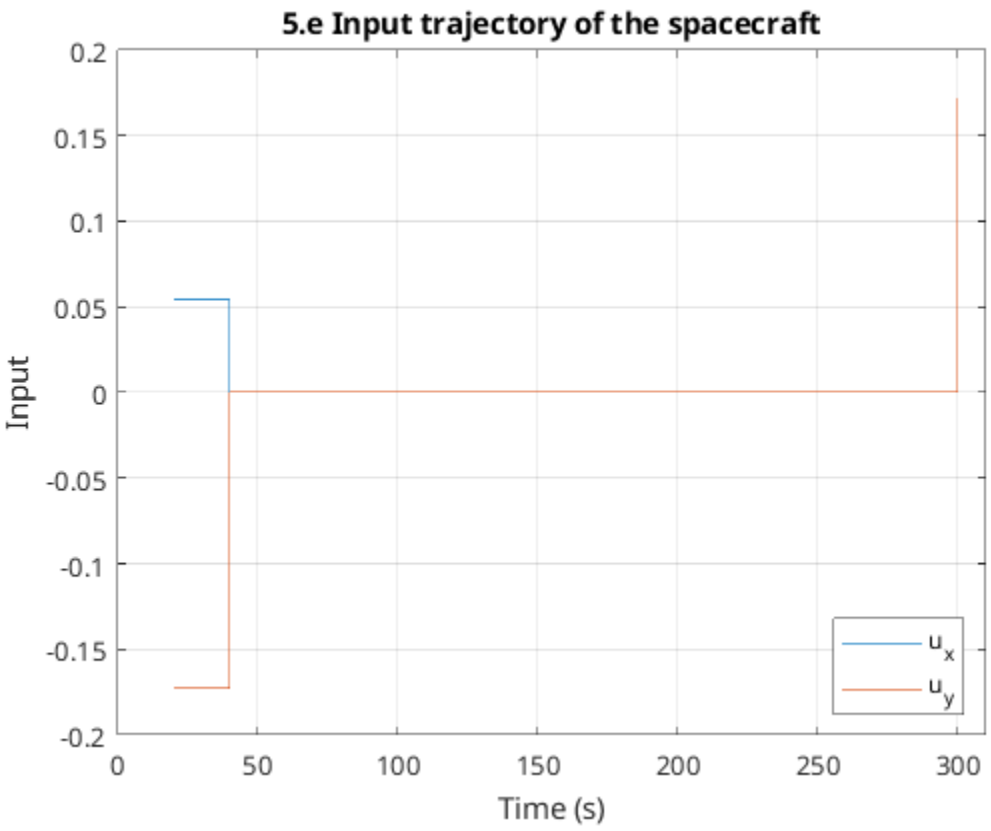
5.e

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in feasible directions, to within the value of the optimality tolerance, and constraints are satisfied to within the value of the constraint tolerance.


5.e Position trajectory of the spacecraft
x
y
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
5
6
7
8
9
10


5.e Velocity trajectory of the spacecraft
x dot
y dot
0.015
0.01
0.005
0
-0.005
-0.01
-0.015
-0.04
-0.035
-0.03
-0.025
-0.02
-0.015
-0.01
-0.005
0



5.f Fuel Cost

```
disp("5.f");
J = sum(abs(Uopt), "all");
disp("5.f Fuel cost = ");
disp(J);
```

```
5.f
5.f Fuel cost =
      0.45131
```

5.g Additional Constraints

```
disp("5.g");
% now we do quadprog(H,f,A,b,Aeq,beq,lb,ub), ignore Aeq, beq, lb
% we need to set up ub so that infnorm(U) <= ub
% which is just |U| <= ub and leave cai to be inf
ub = [0.05 * ones(N * nu, 1);
      inf * ones(N * nu, 1)];
lb = [-0.05 * ones(N * nu, 1);
      -inf * ones(N * nu, 1)];
Zopt = quadprog(GAMMA1, GAMMA2, A, b, [], [], lb, ub);
Uopt = reshape(Zopt(1:N * nu), nu, N);
```

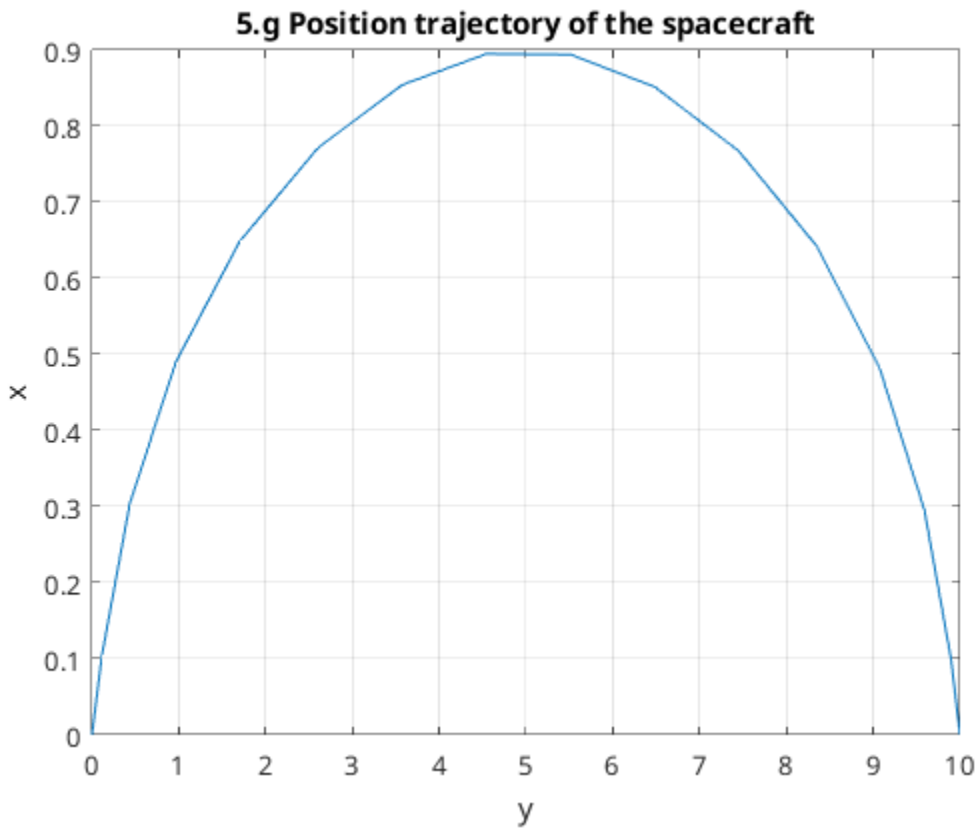
```
data.X = zeros(nx, N + 1);
data.X(:, 1) = X0;

for k = 2:N
  data.X(:, k) = Ad * data.X(:, k - 1) + Bd * Uopt(:, k - 1);
end

% plot trajectory
figure();
plot(data.X(2, :), data.X(1, :));
grid on;
xlabel('y');
ylabel('x');
title('5.g Position trajectory of the spacecraft');

figure();
plot(data.X(4, :), data.X(3, :));
grid on;
xlabel('y dot');
ylabel('x dot');
title('5.g Velocity trajectory of the spacecraft');

% plot inputs
figure();
stairs(Ts * (1:N), Uopt(1, :), "DisplayName", "u_x");
grid on;
hold on;
stairs(Ts * (1:N), Uopt(2, :), "DisplayName", "u_y");
xlabel("Time (s)");
ylabel("Input");
xlim([0, Ts * N + 10]);
title("5.g Input trajectory of the spacecraft");
legend("Location", "best");

% Fuel Cost
J = sum(abs(Uopt), "all");
disp("5.g Fuel cost = ");
disp(J);
```

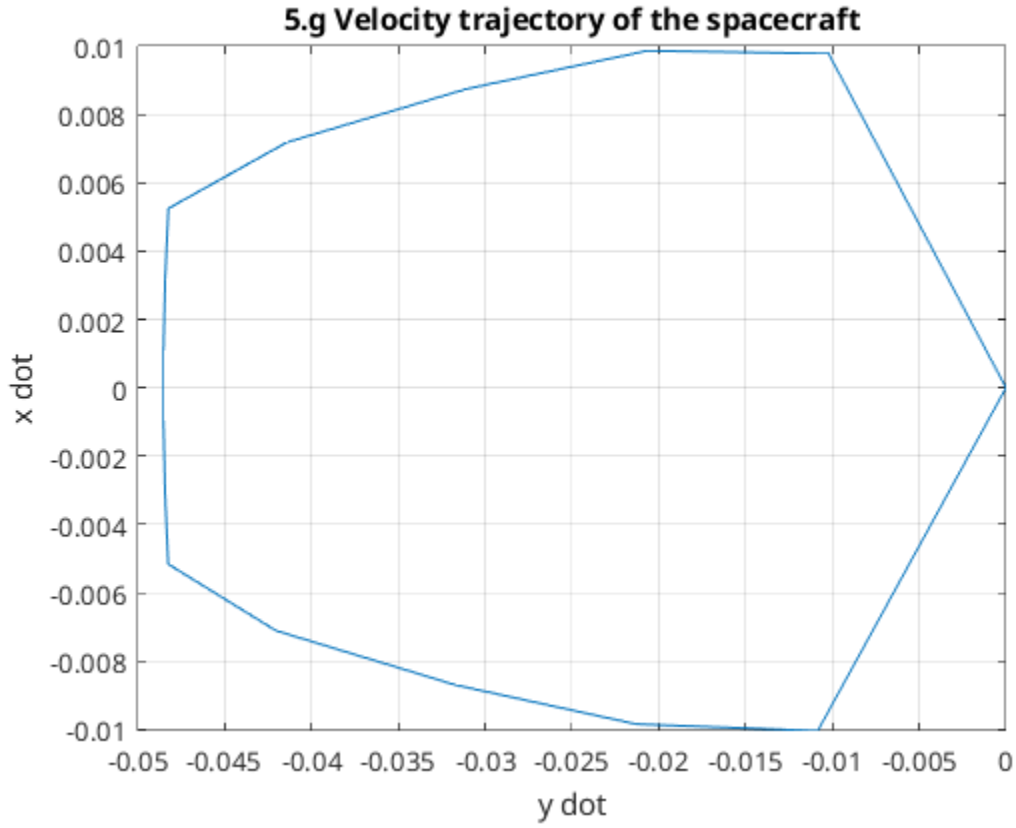
5.g

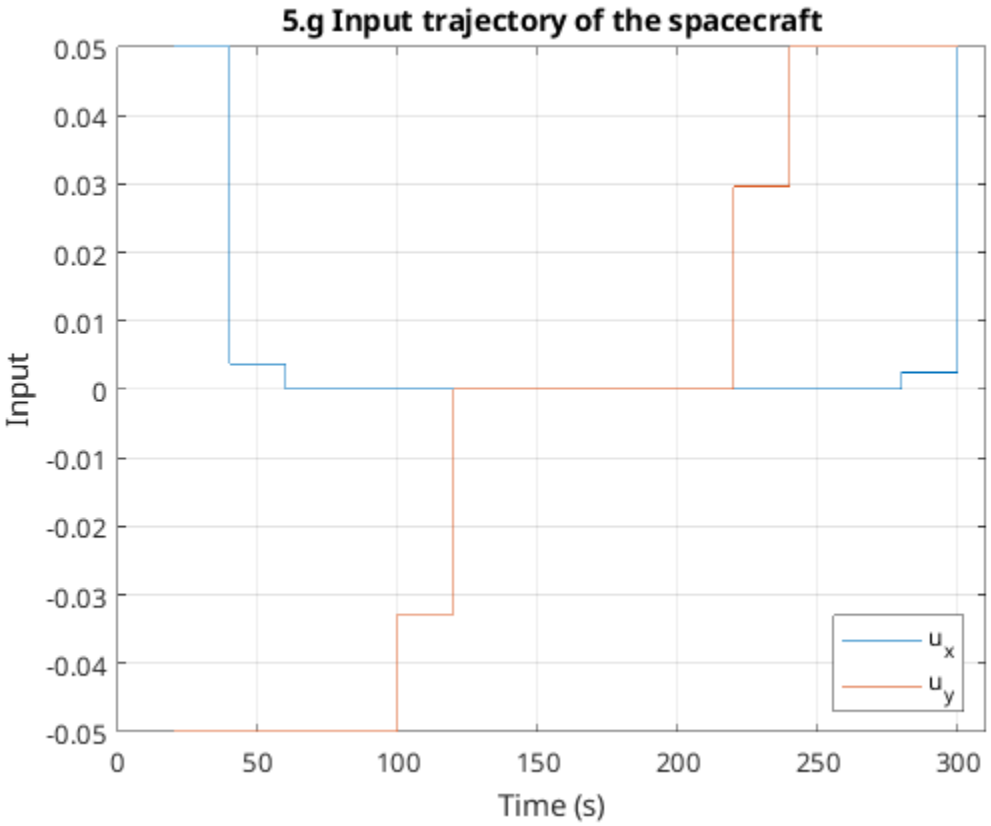
Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

5.g Fuel cost =
0.5687


5.g Position trajectory of the spacecraft
x
y
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
1
2
3
4
5
6
7
8
9
10


5.g Velocity trajectory of the spacecraft
x dot
y dot
0.01
0.008
0.006
0.004
0.002
0
-0.002
-0.004
-0.006
-0.008
-0.01
-0.05
-0.045
-0.04
-0.035
-0.03
-0.025
-0.02
-0.015
-0.01
-0.005
0



The fuel cost is higher with the additional constraints, as expected. The additional constraints limit the inputs, so the spacecraft can't move as freely. The spacecraft has to use additional fuel over more time to achieve the same target state, thus the higher fuel cost. The position and velocity trajectories are also different, as expected, as the spacecraft can't move as freely.

Published with MATLAB® R2023b