ASEN 5044 Midterm 1 Problem 1c-e script

Table of Contents

By: Ian Faber, 10/04/2024

Housekeeping

```
clc; clear; close all
```

Problem 1c

Define model parameters

```
l = 1.85; % m
m = 2; % kg
M = 4; % kg
g = 9.81; % m/s^2
dT = 0.05; % sec

% Define linearized LTI system
Abar = [
            0    1    0                  0
            0    0    (m*g)/M            0
            0    0    0                  1
            0    0    (g*(m+M))/(l*M)    0
       ];

Bbar = [
            0
            1/M
            0
            1/(l*M)
       ];

Cbar = [1 0 -l 0];

Dbar = 0;

Ahat = [
            Abar Bbar
            zeros(1,5)
```

```
      ];

% Disretize the CT system according to the given timestep
matExp = expm(Ahat*dT);

F = matExp(1:size(Abar,1),1:size(Abar,1))
G = matExp(1:size(Bbar,1),size(Abar,1)+1:size(matExp,2))
H = Cbar
M = Dbar

% Compute eigenvalues of F to determine stability
[v, lambda] = eig(F)

% Compute observability matrix to determine observability
O = [
        H
        H*F
        H*F^2
        H*F^3
    ]

obsRank = rank(O)


F =

    1.0000    0.0500    0.0061    0.0001
         0    1.0000    0.2461    0.0061
         0         0    1.0100    0.0502
         0         0    0.3990    1.0100


G =

    0.0003
    0.0125
    0.0002
    0.0068


H =

    1.0000         0   -1.8500         0


M =

     0


v =

    1.0000   -1.0000    0.1754   -0.1754
         0    0.0000    0.4947    0.4947
```

```
         0         0    0.2845   -0.2845
         0         0    0.8022    0.8022


lambda =

    1.0000         0         0         0
         0    1.0000         0         0
         0         0    1.1514         0
         0         0         0    0.8685


O =

    1.0000         0   -1.8500         0
    1.0000    0.0500   -1.8623   -0.0927
    1.0000    0.1000   -1.8994   -0.1866
    1.0000    0.1500   -1.9620   -0.2831


obsRank =

     4
```

Problem 1d

Load midterm data

```
load("midterm1problem1data.mat")

Y = yNLhist;
t = thist';

% Build matrices for the output system of equations
Khat = G*Kc;
E = [];
R = [];
Uhat = [];
for k = 1:length(Y)
    kMath = k - 1; % Create index for math that starts at 0 for
implementation

    % Build E
    E = [E; H*(F^kMath)];

    % Build R
    block = []; % Reset helper variable for building R
    for kk = k:-1:1 % Build up the block from left to right
        kkMath = kk - 1; % Create index for math that starts at 0 for
implementation
        if kk == 1
            block = [block, M];
```

```
        else
            block = [block, H*(F^(kkMath-1))*G];
        end
    end
    if kk < length(Y) % Fill out the rest of this block of R with 0's
        block = [block, zeros(size(block,1), size(Y,1)*1 - size(block, 2))];
% We only have 1 input, *1 is a placeholder
    end
    R = [R; block];

    % Build Uhat
    Uhat = [Uhat; -Kc*(F-Khat)^kMath];

end

L = E + R*Uhat;

dx0 = ((L'*L)^-1)*L'*Y
```

```
dx0 =

    1.5986
    0.2982
   -0.0010
   -0.0042
```

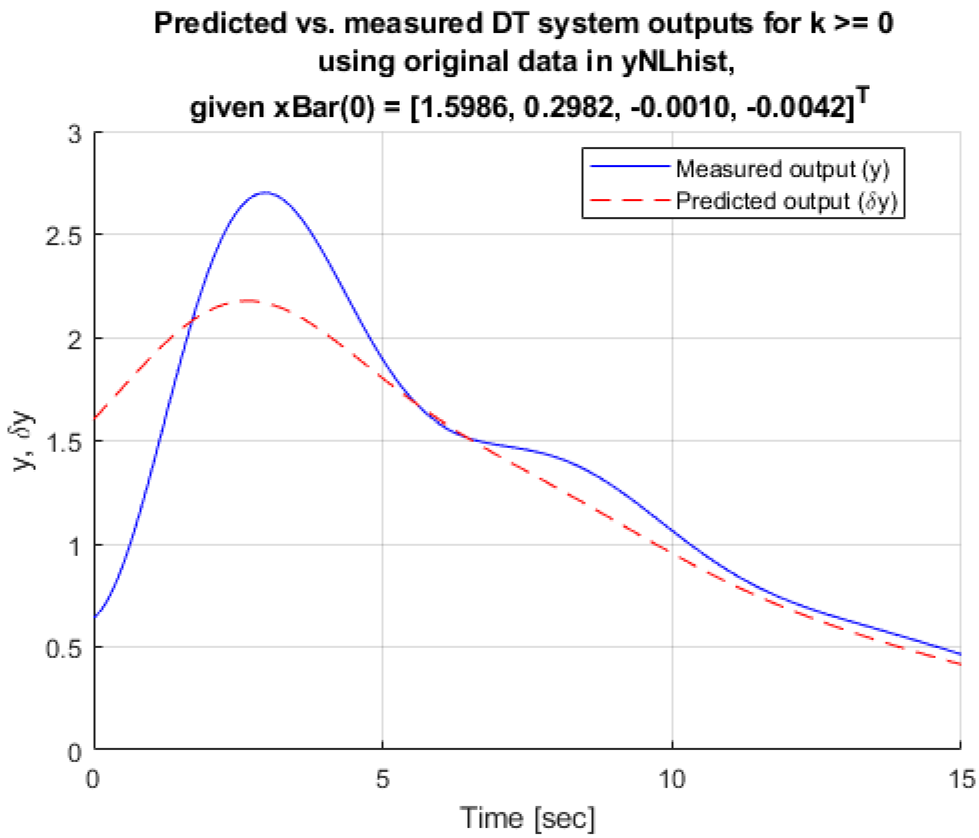
Problem 1e

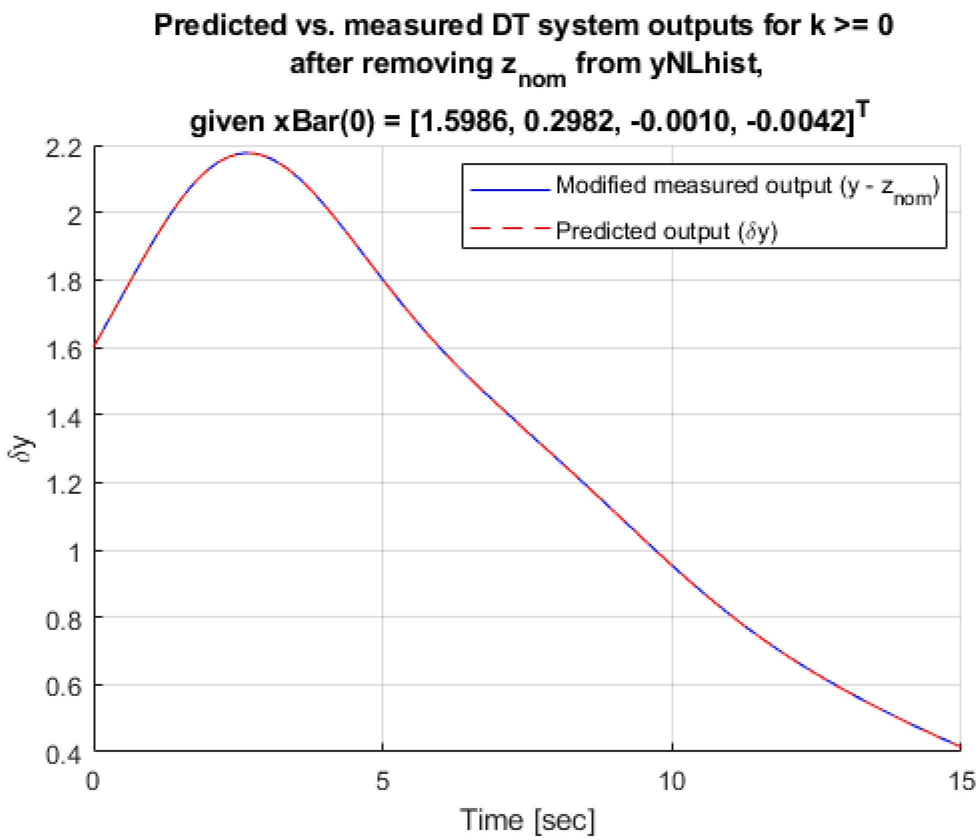
Reconstruct states and predict output based on x0

```
dx = [];
dyPred = [];
zNom = [];
dxCurr = dx0; % Initialize at x0
for k = 1:length(Y)
    u = -Kc*dxCurr;
    dxNext = F*dxCurr + G*u;
    dy = H*dxCurr + M*u;

    dx = [dx, dxCurr];
    dyPred = [dyPred, dy];
    zNom = [zNom, Y(k) - H*dxCurr];

    dxCurr = dxNext; % Reinitialize for next loop
end

% Plot!
dyPred = dyPred';
zNom = zNom';
    % Predicted vs. measured outputs - zNom left in y
titleText = sprintf("Predicted vs. measured DT system outputs for k >= 0
\n using original data in yNLhist, \n given xBar(0) = [%.4f, %.4f, %.4f,
```

```
%.4f]^T", dx0);
figure(1)
hold on; grid on;
title(titleText)
measY = plot(t, yNLhist, 'b-');
predY = plot(t, dyPred, 'r--');
xlabel("Time [sec]")
ylabel("y, \deltay")
legend([measY, predY], ["Measured output (y)", "Predicted output
(\deltay)"], 'location', 'best')
    % Predicted vs. measured outputs - zNom removed from y
titleText = sprintf("Predicted vs. measured DT system outputs for k >= 0 \n
after removing z_{nom} from yNLhist, \n given xBar(0) = [%.4f, %.4f, %.4f,
%.4f]^T", dx0);
figure(2)
hold on; grid on;
title(titleText)
measY = plot(t, yNLhist-zNom, 'b-');
predY = plot(t, dyPred, 'r--');
xlabel("Time [sec]")
ylabel("\deltay")
legend([measY, predY], ["Modified measured output (y - z_{nom})", "Predicted
output (\deltay)"], 'location', 'best')

return % Not interested in system states for this problem, but we can plot
them anyways

    % Recovered system states
titleText = sprintf("Recovered DT system states for k >= 0 \n given xBar(0)
= [%.4f, %.4f, %.4f, %.4f]^T", dx0);
figure(3)
ax = zeros(4,1);
sgtitle(titleText)
ax(1) = subplot(4,1,1);
    hold on; grid on;
    title("\deltaz vs. time")
    plot(t, dx(1,:))
    xlabel("Time [sec]")
    ylabel("\deltaz [m]")
ax(2) = subplot(4,1,2);
    hold on; grid on;
    title("\deltazDot vs. time")
    plot(t, dx(2,:))
    xlabel("Time [sec]")
    ylabel("\deltazDot [m/s]")
ax(3) = subplot(4,1,3);
    hold on; grid on;
    title("\delta\theta vs. time")
    plot(t, dx(3,:))
    xlabel("Time [sec]")
    ylabel("\delta\theta [rad]")
ax(4) = subplot(4,1,4);
    hold on; grid on;
    title("\delta\thetaDot vs. time")
```

```
plot(t, dx(4,:))
xlabel("Time [sec]")
ylabel("\delta\thetaDot [rad/s]")
```





Published with MATLAB® R2023b