AEROSP 548: Assignment 1

Note 1: Completed solutions should be uploaded into Gradescope in a single pdf file before 11:59 p.m. on the day due. **Put boxes around final answers (except for problems involving derivations/proofs, simulation figures and MATLAB codes). 1-2 Points could be deducted if not following the instruction**. Carefully explain and justify your solutions. Plots should be clearly labeled. Include print out of all the codes. Homework should be neat in appearance. You can develop your own code from scratch or follow coding hints given.

Note 2: When submitting the assignment, please follow the steps in the Gradescope and assign corresponding pages to each problem number. **5 points** deduction will be taken if not following this procedure.

Note 3: This assignment pdf together with other complementary documents are uploaded into Canvas: Files > Homeworks and Solutions > Homework 1.

Note 4: For other homework related policies, please consult the syllabus.

1. Consider an inverted pendulum with equations of motion given by

$$\ddot{\theta} = -\frac{g}{l} sin(\theta) + \frac{\tau}{ml^2},$$

where θ is the angle in rad, $g = 9.81$ m/sec^2, $m = 1$ kg and $l = 10$ m.

(a) (1 point) Combine the governing equations into state variable form,

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= h(x(t)) \end{aligned}$$

where $x_1 = \theta$, $x_2 = \dot{\theta}$, $u = \tau$ and $y = x_1$. Write out the resulting equations.

(b) (2 points) For the later purpose of calculating the Jacobian linearization, explicitly write out the function $f(x, u)$. Note that f maps 3 real numbers (x_1, x_2, u) into 2 real numbers. There is **no** explicit time dependence in this particular f.

Write a function called `pendModel` with function declaration line

begin code
```
function xdot = pendModel(x, u)
```
end code

The input arguments are

- x, 2-by-1 array, state x
- u, scalar, input u

The output argument is a 2-by-1 array which equals the derivative $\dot{x}$, as given by the governing equations.

(c) (2 points) Let $u = u_{eq}$ denote an equilibrium motor torque. Write a function `findPendEq`, with function declaration line

```
function xeq = findPendEq(ueq)
```

which computes the equilibrium values of the states given the value of the input, u_{eq}. A straightforward way to accomplish this is to use the function `fsolve.m` and pass it the function handle

`@ (x) pendModel(x,ueq).`

You will also need to pass `fsolve.m` an initial guess, which you can set to `[0;0]`. For $u_{eq} = 20$, what is x_{eq}?

(d) (3 points) Implement a function to compute the linearization of your governing equations (A and B matrices) at specified x_0, u_0 based on symbolic/analytic expressions. You can derive these expressions by hand and code them into the function or use Matlab symbolic toolbox (e.g., `jacobian.m` function). A possible layout for your function is

```
function [A,B] = symLin(x0, u0)
```

For $u_{eq} = 20$, and x_{eq} you have computed, what are A and B matrices?

(e) (3 points) Next implement a function to linearize your governing equations at specified x_0, u_0 using numerical linearization with the center difference formula. The layout for the function is

```
function [A,B] = cdLin(x0, u0)
```

which would call `pendModel` as needed in the computations[1]. For $u_{eq} = 20$, and x_{eq} you have computed, what are A and B matrices? There should be close agreement with the matrices you computed in 1d.

(f) (3 points) Now implement a simulation of your pendulum model using `ode45.m` over the time period of 10 sec. The input should be set to $u(t) = u_{eq} + 2\sin(t),\ 0 \le t \le 10$, where $u_{eq} = 20$. The initial condition at $t = 0$ is `[0;0]`. Plot the time history of the pendulum angle versus time. For instance, you could use the following code snippet:

```
odeFun = @(t,x) pendModel(x, ueq + uIn(t) );
tspan = [0:0.01:10]; x0 =[0;0];
[T,X] = ode45(odeFun, tspan, x0);

function u = uIn(t)
        u = 2*sin(t);
return

```

Note that we pass the function handle `odeFun` to `ode45.m`.

(g) (3 points) Now implement a simulation of your pendulum model using forward Euler approximation over the time period of 10 sec. Recall that forward Euler integration is based

[1]If you wish to make your function generic, you can make function handle as another argument (say f) in `cdLin` and pass `@ pendModel` as this argument.

on

$$x((k+1)h) = x(kh) + hf(x(kh), u(kh)),\ k = 0, 1, 2, \cdots, \tag{1}$$

where h is the integration time step. Consider two cases with different time step, $h = 0.01$ sec and $h = 0.1$ sec. Plot the time histories of the pendulum angle versus time for these two cases. Use the same input, final time and the initial condition as in the previous exercise (1f). What can you conclude regarding the influence of integration time step, h on accuracy of the simulation? To solve this problem you could implement a function with the similar layout to `ode45.m`

begin code
```
function [T, X] = eulerF(odeFun, tspan, x0);
```
end code

that implements (1).

(h) (3 points) Finally, implement the simulation of your linearized model,

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u},$$

where $\tilde{x} = x - x_{eq}$, $\tilde{u} = u - u_{eq}$. Use `ode45.m` to propagate your linearized equations. Plot the time history of the pendulum angle versus time. Here you need to plot the actual pendulum angle which is the sum of nominal and relative. Assume the same torque input, final time and the initial condition as in the previous exercise (1f), however, note that the linearized model input and state are relative to the nominal values at which linearization was performed. The following code snippet may be useful:

begin code
```
[Acd,Bcd]      = cdLin(xeq,ueq);
odeLinSim = @(t,x) Acd*x + Bcd*uIn(t);
tspan = [0:0.01:10]; x0 = [0;0];
[Tl,Xl]   = ode45(odeLinSim,tspan,x0(:)-xeq(:));
```
end code

What can you conclude about prediction accuracy using the linearized model?

2. A linearized (about 0.6 Mach and altitude of 3000 feet flight condition) longitudinal model of F-16 aircraft is given by

$$\dot{x}_c = A_c x + B_c u,$$
$$y = C_c x,$$

where the states are pitch angle (rad), pitch rate (rad/sec), angle of attack (rad), elevator deflection (rad), and flaperon deflection (rad). The input vector u consists of elevator and flaperon commands in rad. The model output vector consists of pitch angle and flight path angle in rad. The matrices A_c, B_c and C_c are given by

$$A_c = \begin{pmatrix} 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & -0.86939 & 43.2230 & -17.2510 & -1.5766 \\ 0.0000 & 0.99335 & -1.3411 & -0.16897 & -0.25183 \\ 0.0000 & 0.0000 & 0.0000 & -20.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & -20.0000 \end{pmatrix}, B_c = \begin{pmatrix} 0.0000 & 0.0000 \\ 0.0000 & 0.0000 \\ 0.0000 & 0.0000 \\ 20.0000 & 0.0000 \\ 0.0000 & 20.0000 \end{pmatrix},$$

$$C_c = \begin{pmatrix} 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 1.0000 & 0.0000 & -1.0000 & 0.0000 & 0.0000 \end{pmatrix}.$$

(a) (1 point) Is the model open-loop stable? Stability here is in the sense of asymptotic stability at the origin or A_c being a Hurwitz matrix.

(b) (2 points) Is the model controllable? If it is, what is the minimum time horizon over which any state in $\mathbb{R}^5$ can be reached?

(c) (1 point) Convert the model to discrete-time using sampling period of $T_s = 0.01$ sec. Use `c2d.m` function in Matlab with 'zoh' option. For the answer, give discrete-time model matrices, A_d and B_d.

(d) (2 points) Attempt to perform the same conversion as in but now using expressions given in class,

$$A_d = e^{AT_s}, \quad B_d = A^{-1}(e^{AT_s} - I)B.$$

If you are not successful, explain why. Note that you should use `expm()` for the matrix exponential and not `exp()`!

(e) (1 point) Is the discrete-time open-loop model stable? Stability here is in the sense of asymptotic stability at the origin or A_d being a Schur matrix.

(f) (2 points) Is the discrete-time open-loop model controllable? If it is, what is the minimum horizon over which any state in $\mathbb{R}^5$ can be reached from any other state?

(g) (2 points) Suppose that the controller has the form of a feedforward and feedback and is given by

$$u = F_c r + K_c x,$$

where the feedforward and feedback gain are defined by

$$F_c = \begin{pmatrix} -2.8900 & 0.7780 \\ 1.9800 & 3.3400 \end{pmatrix}, K_c = \begin{pmatrix} 2.1100 & 0.8906 & 4.9107 & -0.5343 & -0.1009 \\ -5.3200 & -0.8980 & -4.6618 & 0.4280 & 0.1099 \end{pmatrix},$$

and r is the vector of the commanded pitch angle and flight path angle commands. Is the closed-loop system stable? What is the DC/Static Gain matrix of the closed-loop system, i.e., $H \in \mathbb{R}^{2\times 2}$ such that $y = Hr$ in steady-state.

(h) (3 points) Implement simulations of the continuous-time closed loop system for 5 sec. Use `ode45.m`. Saturate the first input channel to ± 0.4363 rad (maximum elevator deflection) and the second input channel to ± 0.3491 rad (maximum flaperon deflection). Simulate two cases. Case 1: `r=[0.01745;-0.01745]` and Case 2 with larger commands: `r=[0.1745;-0.1745]`. Include simulation plots of pitch angle, flight path angle, elevator position and flaperon position time histories. Assume zero initial conditions. Explain the effect of the input saturation on the closed-loop system response.

(i) (2 points) Suppose now that you implement your feedback in discrete-time. The closed-loop system in discrete-time, assuming no saturation, is given by

$$x^+ = (A_d + B_d K_c)x + B_d F r,$$

and its stability will be determined by the eigenvalues of $A_d + B_d K_c$. Is the closed-loop stable?

(j) (2 points) Suppose now T_s is increased to 0.5 sec. Re-compute $A_d + B_d K_c$ and check its eigenvalues. What can you conclude about the effect of increasing sampling period on the closed-loop stability? Note that sampling introduces a delay. A communication delay, e.g., if our F-16 is a drone which is remotely piloted through a satellite network, will have a similar effect!

(k) (2 points) Implement the discrete-time simulation of the closed-loop system with saturation based on A_d, B_d you have computed in (2d). Include the same plots as in Problem (2h) for the command in Case 1.

3. Consider an estimation problem for a piece of orbital debris in a space situational awareness problem. The state vector is given by

$$X = \left[\begin{array}{cccccc} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{array}\right]^{\mathrm{T}},$$

and consists of three relative coordinates (in km units) and three relative velocities (in km/sec units), all relative to a nominal position on a circular orbit. Specifically, x is the radial coordinate (R-bar), y is in-track radial coordinate (V-bar) and z is the out of orbital plane coordinate. The debris relative motion dynamics are given by Clohessy-Wilthsire-Hill (CWH) equations,

$$\dot{X} = A_c X,$$

$$A_c = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{pmatrix},$$

where $n = 0.00114$ rad/sec is the mean motion on the nominal circular orbit. A spacecraft has two sensors which measure two signals

$$Y_1 = 0.5x + 0.5y, \quad Y_2 = z.$$

(a) (4 points) Assume that the measurements are taken every $\Delta T = 30$ sec. Write out the dynamics and output measurement equation in the discrete-time form,

$$X_{k+1} = A_d X_k, \quad Y_k = C_d X_k.$$

Give A_d and C_d as your answers.

(b) (2 points) Is the pair (C_d, A_d) observable?

(c) (4 points) To estimate the state we will design the Luenberger observer,

$$\hat{X}_{k+1} = A_d \hat{X}_k + L(\hat{Y}_k - Y_k), \quad \hat{Y}_k = C_d \hat{X}_k,$$

where L is the observer gain which needs to be selected so that all eigenvalues of $A_d + LC_d$ are strictly inside the unit disk of the complex plane. To design the observer gain, use Matlab `place.m` command and place the eigenvalues of $A_d + LC_d$ at

$$E = \begin{pmatrix} 0.9712 \\ 0.9377 \\ 0.9348 + 0.0686\,\mathrm{i} \\ 0.9348 - 0.0686\,\mathrm{i} \\ 0.9471 + 0.0753\,\mathrm{i} \\ 0.9471 - 0.0753\,\mathrm{i} \end{pmatrix}.$$

Give the value of the observer gain, L.
Hint: `place.m` command places eigenvalues of $A - BK$ for a controllable pair (A, B) and gives feedback gain K. Note that eigenvalues of $A_d + LC_d$ are the same as eigenvalues of its transpose, i.e., of $A_d^{\mathrm{T}} - C_d^{\mathrm{T}}(-L)^{\mathrm{T}}$ to which `place.m` command can be applied to compute $(-L)^{\mathrm{T}}$.

(d) (5 points) Implement discrete-time simulations of the nominal model and of the observer for 100 steps assuming initial conditions, $X_0 = [0.1, 1, 2, 0, 0.01, -0.01]^{\mathrm{T}}$ for the debris and $\hat{X}_0 = [0, 0, 0, 0, 0, 0]^{\mathrm{T}}$ for the observer estimate. You can use the following code template:

begin code

```
Traj = [];
for k = 0:1:100,
        Traj.time(k+1)    = k;
        Traj.Xd(k+1,:)    = Xd;
        Traj.Xdhat(k+1,:) = Xdhat;

        Yd    = Cd*Xd;    % true output measurement
        Ydhat = Cd*Xdhat; % estimated output measurement

        Xd = Ad*Xd; % update model
        Xdhat = Ad*Xdhat + Ld*(Ydhat - Yd); % update the observer
end;
```

end code

Plot the discrete-time histories of true and estimated y, $\dot{y}$, z, and $\dot{z}$.

(e) (5 points) Repeat the simulations in (3d) but assume that each measurement channel is contaminated with identically distributed and independent gaussian noise with the zero mean and standard deviation of 0.5 (500 m). Hint: Use `randn.m` command of Matlab. Plot true and estimated values of y, $\dot{y}$, z and $\dot{z}$.