AEROSP 740: Assignment 2

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 2.

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

1. Let
$$J(x) = x^{\mathrm{T}} Q x + c^{\mathrm{T}} x,$$
where $x, c \in \mathbb{R}^{n\times 1}$, $n \geq 1$, and $Q = Q^{\mathrm{T}} \in \mathbb{R}^{n\times n}$. Prove that the gradient of J satisfies
$$\nabla_x J = 2Qx + c.$$
   (a) Consider the second term first, $c^{\mathrm{T}} x = c_1 x_1 + \cdots c_n x_n$ and compute its partial derivatives, $\frac{\partial}{\partial x_i} c^{\mathrm{T}} x$.
   (b) Consider the first term next,
$$x^{\mathrm{T}} Q x = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i Q_{ij} x_j,$$
   extract all the terms that contain x_i for a particular i in this sum and apply $\frac{\partial}{\partial x_i}$ to these terms. Recall that $Q_{ij} = Q_{ji}$ since Q is symmetric. You can try $n = 2$ first to see the pattern.
   (c) What is the Hessian (matrix of second partial derivatives) of J, $\nabla^2 J(x)$?

2. Consider an unconstrained LQ-MPC problem for a system with a discrete-time model of the form,
$$x_{k+1} = Ax_k + Bu_k,$$
where
$$A = \begin{bmatrix} \frac{4}{3} & -\frac{2}{3} \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$
Let
$$C = \begin{bmatrix} -\frac{2}{3} & 1 \end{bmatrix},$$

and define the weighting matrices as

$$Q = C^{\mathrm{T}}C, \quad R = 0.001, \; P = 0$$

(a) Is A a Schur matrix? Is (A, B) stabilizable? Is (A, B) controllable? Is (C, A) detectable? Is (C, A) observable?

(b) For LTI discrete-time systems, the transfer function is given by $T(z) = C(zI - A)^{-1}B + D$, where the variable z plays a role similar to Laplace variable s in continuous-time. Use symbolic computations,

begin code

```
z = sym('z')
simplify(C*inv(z*eye(2,2) - A)*B + 0)
```

end code

to compute the transfer function. Where are poles (zeros of denominator) and zeros (zeros of numerator) of the open-loop transfer function? Are they inside or outside the unit disk of the complex plane? Are poles the same as eigenvalues of A?

(c) Generate the step response of the open-loop system. Since our model is discrete-time, use `dstep(A,B,C,0)` command rather than `step` command. Note that the step response shows an initial undershoot, which is a symptom of a nonminimum phase (unstable) zero.

3. (a) Implement a Matlab function that performs unconstrained LQ-MPC computations shown in class. Use a template

begin code

```
[S, M, Qbar, Rbar, K0N] = uncMPC(N, A, B, Q, R, P)
```

end code

Your function should implement all the computations "manually" and not call any MPC toolboxes or QP solvers. Implement computations leading to

$$K_{0,N} = -\left[\begin{array}{cccc} \mathbb{I}_{n_u \times n_u} & 0 & \cdots & 0 \end{array}\right] (S^{\mathrm{T}}\bar{Q}S + \bar{R})^{-1} S^{\mathrm{T}}\bar{Q}M,$$

i.e., **not** the Riccati recursion version.

(b) For the prediction horizon N between 1 and 20, use your function in (3a) for Problem 2 and compute the feedback gain, $K_{0,N}$. Generate a plot of the spectral radius of the closed-loop dynamics matrix $(A + BK_{0,N})$ versus the horizon N. For what values of the horizon N, is the closed-loop stable?

(c) Generate a plot of closed-loop eigenvalues under MPC feedback law, i.e., of $A + BK_{0,N}$, in the complex plane for $N = 1, \cdots, 20$. Also draw a unit circle on the complex plane to delineate the stability boundary.

(d) Generate now the LQR gain K_∞ using `dlqr.m` command. Use sign convention from the class (multiply the gain computed by dlqr.m by -1). Give and compare values of $K_{0,20}$ and K_∞. Are they close? Is LQR gain stabilizing?

(e) Generate a plot of each of the two components of the gain $K_{0,N}$ versus the horizon, N. Indicate the values of K_∞ on the same plot by dashed horizontal lines. Does $K_{0,N}$ converge to K_∞ as $N \to \infty$ based on your plot?

4. (a) Implement a Matlab function that performs a Riccati recursion version of unconstrained LQ-MPC computations shown in class. Use a template

```
begin code
[S, M, Qbar, Rbar, KON, PON] = uncMPCric(N, A, B, Q, R, P)
end code
```

Hint: You may wish to check that $K_{0,N}$ that you compute matches the one in (3a).

(b) Using your function in (4a), for the Problem 2, compute $P_{0,2}$, $P_{0,11}$ and $P_{0,20}$. Compute also P_∞ with the help of `dlqr.m`. Comment on the differences.

(c) Introduce now the terminal penalty as $P = P_\infty$. Using your code in (4a), compute the gains $K_{0,N}$ for $N = 2, 11, 20$. Compare these gains with K_∞. Are any of these gains stabilizing? Compute also the corresponding values of $P_{0,2}$, $P_{0,11}$ and $P_{0,20}$.

(d) Suppose that a control gain K is stabilizing, i.e., $A + BK$ is Schur. Show that the infinite horizon performance, i.e., the value of the cost function,

$$J = \sum_{k=0}^{\infty} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k$$

can be computed as $J = x_0^{\mathrm{T}} P_K x_0$, where P_K is a solution of a Lyapunov equation of the form,

$$(A + BK)^{\mathrm{T}} P_K (A + BK) - P_K + (Q + K^{\mathrm{T}} R K) = 0$$

(e) Pick an initial condition, $x_0 = [1, -0.5]^{\mathrm{T}}$. Using your result in (4d)) and Matlab function `dlyap.m`, compute the infinite horizon performance under the control $u_k = K_\infty x_k$ and under the control $u_k = K_{0,11} x_k$ for Problem 2.

5. The spacecraft attitude dynamics are represented by nonlinear ODEs

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \frac{1}{\cos(\theta)} \begin{bmatrix} \cos(\theta) & \sin(\phi)\sin(\theta) & \cos(\phi)\sin(\theta) \\ 0 & \cos(\phi)\cos(\theta) & -\sin(\phi)\cos(\theta) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix},$$

$$\begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{bmatrix} = \begin{bmatrix} ((J_2 - J_3)\omega_2\omega_3)/J_1 + M_1/J_1 \\ ((J_3 - J_1)\omega_3\omega_1)/J_2 + M_2/J_2 \\ ((J_1 - J_2)\omega_1\omega_2)/J_3 + M_3/J_3 \end{bmatrix},$$

where ϕ, θ, ψ are spacecraft roll, pitch and yaw angles, respectively; ω_1, ω_2, ω_3 are components of spacecraft angular velocity vector in the body fixed frame; $J_1 = 120$, $J_2 = 100$, $J_3 = 80$ are spacecraft principal moments of inertia; and $u = [M_1,\ M_2,\ M_3]^{\mathrm{T}}$ is the vector of control moments.

In this exercise, we will use `mpctools` package to implement an MPC controller for spacecraft motion. Review mpctools lecture slides and examine examples in canvas. We will modify them for the case of our spacecraft. The objective is to control spacecraft orientation to specified Euler angles defined by a command vector, $r \in \mathbb{R}^3$. In this example, we will add the commands as the extra states with the trivial dynamics, $\dot{r} = 0$.

(a) Implement continuous-time nonlinear spacecraft model with principal moments of inertia $J_1 = 120$, $J_2 = 100$ and $J_3 = 80$, state vector $x = [\phi, \theta, \psi, \omega_1, \omega_2, \omega_3, r_1, r_2, r_3]^{\mathrm{T}}$ and control vector $u = [M_1, M_2, M_3]^{\mathrm{T}}$ consisting of three control moments in a function

begin code
```
function dxdt = ode(x, u)
```
end code

(b) Assume

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix}, \; R = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix},$$

and implement the stage cost function

begin code
```
function l = stagecost(x, u)
  r = x(7:9);
  Q = diag([1, 1, 1, 0.01, 0.01, 0.01]);
  R = diag([1,1,1])*0.01;
  e = [x(1:3)-r(:);x(4:6)];
  l = e'*Q*e + u'*R*u;
end
```
end code

(c) What is the continuous-time linearized model at $x = 0$, $u = 0$? The model should have the form,

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

and you should give A_c and B_c matrices as an answer. Use symbolic linearization, and keep J_1, J_2, J_3 as symbols.

(d) Convert the continuous-time linearized model (with numeric values for J_1, J_2, J_3) plugged-in to discrete-time assuming the sampling period of $T_s = 2$ sec. Use `c2d.m`. What are eigenvalues of discrete-time dynamics matrix, A_d? Is A_d a Schur matrix? Does the pair (A_d, B_d) satisfies the rank controllability conditions?

(e) Implement the terminal cost function using the template

begin code
```
function Vf = termcost(x)
  r = x(7:9);
  Pinf = ...;
  e = [x(1:3)-r(:);x(4:6)];
  Vf = e'*Pinf*e;
end
```
end code

Hard code `Pinf` as the solution to Discrete Algebraic Riccati Equation (DARE) corresponding to the spacecraft linearized model at the origin and Q and R matrices given in the function `stagecost`. Give `Pinf` as the answer.

(f) Define the sampling time as 2 sec and configure simulations for 200 sec. Set the prediction horizon to 30 steps. Configure `lb.x` and `lb.u` to reflect constraints $|\omega_i| \leq 0.03$, $i = 1, 2, 3$, and $|u_i| \leq 0.1$, $i = 1, 2, 3$. Use both linear MPC and nonlinear MPC solvers to

simulate the closed-loop response for the initial Euler angles and angular velocities given by $-0.4, -0.8, 1.2, -0.02, -0.02, 0.02$, respectively. Set $r = [0, 0, 0]^{\mathrm{T}}$ for half of the simulation time and $r = [0.5, 0, -0.5]$ for the other half of the simulation time. This can be accomplished by setting components $7, 8, 9$ of the vector x to r i.e.,

begin code

```
x(7:9,k) = r;
solver.fixvar('x', 1, x(:,k));
solver.solve();
```

end code

Show traces of the time histories of the Euler angles, angular velocities and control moments for both linear and nonlinear MPC controllers.