ORF307_HW3

February 7, 2022

ORF307 Homework 3

Due: Friday, Feburary 18, 2022 9:00 pm ET

- Please export your code with output as pdf.
- If there is any additional answers, please combine them as **ONE** pdf file before submitting to the Gradescope.

Q1 Robust approximate solution of linear equations

We wish to solve the square set of n linear equations $Ax = b$ for the n-vector x. If A is invertible the solution is $x = A^{-1}b$

In this exercise we address an issue that comes up frequently: We don't know A exactly. One simple method is to just choose a typical value of A and use it. Another method, which we explore here, takes into account the variation in the matrix A. We find a set of K versions of A, and denote them as $A^{(1)}, ..., A^{(K)}$. (These could be found by measuring the matrix A at different times, for example.) Then, we choose x so as to minimize

$$\|A^{(1)}x - b\|_2^2 + ... + \|A^{(K)}x - b\|_2^2,$$

the sum of the squares of residuals obtained with the K versions of A. This choice of x, which we denote x^{rob}, is called a robust (approximate) solution. Give a formula for x^{rob}, in terms of $A^{(1)}, ..., A^{(K)}$ and b. (You can assume that a matrix you construct has linearly independent columns.) Verify that for $K = 1$ your formula reduces to $x^{\text{rob}} = (A^{(1)})^{-1}b$

Q2 Least norm polynomial interpolation.

(a) Interpolation of polynomial values and derivatives. The 5-vector c represents the coefficients of a quartic polynomial $p(x) = c_1 + c_2x + c_3x^2 + c_4x^3 + c_5x^4$. Express the conditions

$$p(0) = 0, \quad p'(0) = 0, \quad p(1) = 1, \quad p'(1) = 0,$$

as a set of linear equations of the form $Ac = b$. Is the system of equations underdetermined, over-determined, or square?

(b) Find the polynomial of degree 4 that satisfies the interpolation conditions from part (a), and minimizes the sum of the squares of its coefficients. Plot it, to verify that it satisfies the interpolation conditions.

Q3 Linear Quadratic Control

We consider a time-varying linear dynamical system with state n-vector x_t and input m-vector u_t, with dynamics equations

$$x_{t+1} = A_t x_t + B_t u_t, \quad t = 1, 2, \dots .$$

The system has an output, the p-vector y_t, given by

$$y_t = C_t x_t, \quad t = 1, 2, \dots .$$

Usually, $m \leq n$ and $p \leq n$, i.e., there are fewer inputs and outputs than states. In control applications, the input u_t represents quantities that we can choose or manipulate, like control surface deflections or engine thrust on an airplane. The state x_t, input u_t, and output y_t typically represent deviations from some standard or desired operating condition, for example, the deviation of aircraft speed and altitude from the desired values. For this reason it is desirable to have x_t, y_t, and u_t small. Linear quadratic control refers to the problem of choosing the input and state sequences, over a time period $t = 1, \dots, T$, so as to minimize a sum of squares objective, subject to the dynamics equations

$$x_{t+1} = A_t x_t + B_t u_t, \quad t = 1, 2, \dots$$

the output equations

$$y_t = C_t x_t, \quad t = 1, 2, \dots ,$$

and additional linear equality constraints. In 'linear quadratic', 'linear' refers to the linear dynamics, and 'quadratic' refers to the objective function, which is a sum of squares. Most control problems include an initial state constraint, which has the form $x_1 = x^{\text{init}}$, where x^{init} is a given initial state. Some control problems also include a final state constraint $x_T = x^{\text{des}}$, where x^{des} is a given ('desired') final (also called terminal or target) state.

The objective function has the form $J = J_{\text{output}} + \rho J_{\text{input}}$, where $J_{\text{output}} = \|y_1\|^2 + \cdots + \|y_T\|^2 = \|C_1 x_1\|^2 + \cdots + \|C_T x_T\|^2$, $J_{\text{input}} = \|u_1\|^2 + \cdots + \|u_{T-1}\|^2$. The positive parameter ρ weights the input objective J_{input} relative to the output objective J_{output}.

Thus, the linear quadratic control problem is

$$\begin{array}{ll} \text{minimize} & J_{\text{output}} + \rho J_{\text{input}} \\ \text{subject to} & x_{t+1} = A_t x_t + B_t u_t, \quad t = 1, 2, \dots \\ & x_1 = x^{\text{init}}, x_T = x^{\text{des}} \end{array}$$

We can solve this linear quadratic control problem by setting it up as a big linearly constrained least squares problem. We define the vector z of all these variables, stacked: $z = (x_1, \dots, x_T, u_1, \dots, u_{T-1})$. The dimension of z is $Tn + (T-1)m$. The control objective can be expressed as $\|\tilde{A}z - \tilde{b}\|^2$, where $\tilde{b} = 0$ and $\tilde{A}$ is the block diagonal matrix

$$\tilde{A} = \left[\begin{array}{ccc|ccc} C_1 & & & & & \\ & C_2 & & & & \\ & & \ddots & & & \\ & & & C_T & & \\ \hline & & & \sqrt{\rho} I & & \\ & & & & \ddots & \\ & & & & & \sqrt{\rho} I \end{array}\right]$$

In this matrix, (block) entries not shown are zero, and the identity matrices in the lower right corner have dimension m. The lines in the matrix delineate the portions related to the states and the inputs. The dynamics constraints, and the initial and final state constraints, can be expressed as $\tilde{C}z = \tilde{d}$, with

$$\tilde{C} = \left[\begin{array}{ccccc|ccccc} A_1 & -I & & & & B_1 & & & & \\ & A_2 & -I & & & & B_2 & & & \\ & & \ddots & \ddots & & & & \ddots & & \\ & & & A_{T-1} & -I & & & & & B_{T-1} \\ \hline I & & & & & & & & & \\ & & & & I & & & & & \end{array}\right]$$

where (block) entries not shown are zero. The vertical line separates the portions of the matrix associated with the states and the inputs, and the horizontal lines separate the dynamics equations and the initial and final state constraints. Finally, vector

$$d = (0, 0, \ldots, 0, x^{\text{init}}, x^{\text{des}})$$

has a length of $n(T-1)+2$.

The solution $z^\star$ of the constrained least squares problem

$$\begin{array}{ll} \text{minimize} & \|\tilde{A}z - \tilde{b}\|^2 \\ \text{subject to} & \tilde{C}z = \tilde{d} \end{array}$$

gives us the optimal input trajectory and the associated optimal state (and output) trajectory. The solution $z^\star$ is a linear function of $\tilde{b}$ and $\tilde{d}$; since here $\tilde{b} = 0$, it is a linear function of x^{init} and x^{des}.

State feedback control of the longitudinal motions of a Boeing 747 aircraft. In this exercise we consider the control of the longitudinal motions of a Boeing 747 aircraft in steady level flight, at an altitude of 40000 ft, and speed 774 ft/s, which is 528 MPH or 460 knots, around Mach 0.8 at that altitude. Longitudinal means that we consider climb rate and speed, but not turning or rolling motions. For modest deviations from these steady state or trim conditions, the dynamics is given by the linear dynamical system $x_{t+1} = Ax_t + Bu_t$, with

$$A = \begin{bmatrix} 0.99 & 0.03 & -0.02 & -0.32 \\ 0.01 & 0.47 & 4.70 & 0.00 \\ 0.02 & -0.06 & 0.40 & 0.00 \\ 0.01 & -0.04 & 0.72 & 0.99 \end{bmatrix}, \quad B = \begin{bmatrix} 0.01 & 0.99 \\ -3.44 & 1.66 \\ -0.83 & 0.44 \\ -0.47 & 0.25 \end{bmatrix}.$$

[12]:
```
import numpy as np
A = np.array([[.99, .03, -.02, -.32],
              [.01, .47, 4.7, 0],
              [.02, -.06, .4, 0],
              [.01, -.04, .72, .99]])
B = np.array([[.01, .99],
              [-3.44, 1.66],
              [-.83, .44],
              [-.47, .25]])
```

with time unit one second. The state 4-vector x_t consists of deviations from the trim conditions of the following quantities.

- $(x_t)_1$ is the velocity along the airplane body axis, in ft/s, with forward motion positive. - $(x_t)_2$ is the velocity perpendicular to the body axis, in ft/s, with positive down. - $(x_t)_3$ is the angle of the body axis above horizontal, in units of 0.01 radian. - $(x_t)_4$ is the derivative of the angle of the body axis, called the pitch rate, in units of 0.01 radian/s.

The input 2-vector ut (which we can control) consists of deviations from the trim conditions of the following quantities. - $(u_t)_1$ is the elevator (control surface) angle, in units of 0.01 radian. - $(u_t)_2$ is the engine thrust, in units of 10000 lbs. You do not need to know these details;

we mention them only so you know what the entries of x_t and u_t mean.

(a) Open loop trajectory. Simulate the motion of the Boeing 747 with initial condition $x_1 = e_4$, in open-loop (i.e., with $u_t = 0$). Plot the state variables over the time interval $t = 1, ..., 120$ (two minutes). The oscillation you will see in the open-loop simulation is well known to pilots, and called the phugoid mode.

(b) Linear quadratic control. Solve the linear quadratic control problem with $C = I$, $\rho = 100$, and $T = 100$, with initial state $x_1 = e_4$, and desired terminal state $x^{des} = 0$. Plot the state and input variables over $t = 1, \dots, 120$. (For $t = 100, \dots, 120$, the state and input variables are zero.). You may use the following functions.

[9]:
```
def KKT_matrix_rhs(A, b, C, d):
    """
    returns the KKT matrix and the right hand side vector that describes the
    linear system to solve the constrained least squares problem
    """
    m, n = A.shape
    p, n = C.shape

    G = A.T @ A  # Gram matrix
    KKT = np.block([[2*G, C.T],
                    [C, np.zeros((p, p))]])
    rhs = np.hstack([2*A.T @ b, d])
    return KKT, rhs
```

[16]:
```
def construct_tilde_matrices(A, B, C, x_init, x_des, T, rho):
    """
    Construct A_tilde, b_tilde, c_tilde, d_tilde.
    """
    n, m = B.shape
    p = C.shape[0]

    A_tilde = np.block([[np.kron(np.eye(T), C), np.zeros((p * T, m * (T-1)))],
                        [np.zeros((m * (T - 1), n * T)), np.sqrt(rho) * np.
 ↪eye(m * (T-1))]])

    # construct btilde
    b_tilde = np.zeros(p * T + m * (T - 1))
```

```
    # construct Ctilde bit by bit
    C_tilde11 = np.hstack([np.kron(np.eye(T-1),A), np.zeros((n*(T-1),n))]) \
        - np.hstack([np.zeros((n*(T-1),n)), np.eye(n*(T-1))])
    C_tilde12 = np.kron(np.eye(T-1), B)
    C_tilde21 = np.block([[np.eye(n), np.zeros((n,n*(T-1)))],
                          [np.zeros((n,n*(T-1))), np.eye(n)]])
    C_tilde22 = np.zeros((2 * n, m * (T-1)))

    C_tilde = np.block([[C_tilde11, C_tilde12],
                        [C_tilde21, C_tilde22]])

    # construct dtilde
    d_tilde = np.hstack([np.zeros(n * (T - 1)), x_init, x_des])

    return A_tilde, b_tilde, C_tilde, d_tilde
```

[17]:
```
# Here is some skeleton code to get you started.
# Note that it will not run since we do not have some
# of the values: C, n, T, rho for example
# you will need to fill in some of the blanks
x_init = np.array([0, 0, 0, 1])
x_des = np.zeros(n)
A_tilde, b_tilde, C_tilde, d_tilde = construct_tilde_matrices(A, B, C, x_init,
                                                              x_des, T, rho)

# once we have the tilde matrices, we will need to solve the
# constrained least squares problem using KKT and rhs


# plot your results
```