InvertedPendulum_ControlDigital

September 30, 2018

```
In [1]: import sympy
        from IPython.display import Latex, display
        from sympy import Poly
        from sympy.abc import s, z

        from control.matlab import *

In [2]: %pylab %matplotlib inline

UsageError: unrecognized arguments: inline
```

1 Inverted Pendulum: Digital Controller Design

In this digital control version of the inverted pendulum problem, we will use the state-space method to design the digital controller. If you refer to the Inverted Pendulum: System Modeling page, the linearized state-space equations were derived as:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{bmatrix} u$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u$$

where: (M) mass of the cart 0.5 kg (m) mass of the pendulum 0.2 kg (b) coefficient of friction for cart 0.1 N/m/sec (l) length to pendulum center of mass 0.3 m (I) mass moment of inertia of the pendulum 0.006 kg.m^2 (F) force applied to the cart (x) cart position coordinate (theta) pendulum angle from vertical (down) For this problem the outputs are the cart's displacement (x in meters) and the pendulum angle (ϕ in radians) where ϕ represents the deviation of the pedulum's position from equilibrium, that is, $\theta = \pi + \phi$. The design criteria for this system for a 0.2-m step in desired cart position x are as follows: * Settling time for x and *theta* of less than 5 seconds * Rise time for x of less than 0.5 seconds * Pendulum angle θ never more than 20 degrees (0.35 radians) from the vertical * Steady-state error of less than 2% for x and θ

1.1 Discrete state-space

Our first step in designing a digital controller is to convert the above continuous state-space equations to a discrete form. We will accomplish this employing the MATLAB function |c2d|. This function requires that we specify three arguments: a continuous system model, the sampling time (|Ts| in sec/sample), and the |'method'|. You should already be familiar with how to construct a state-space system from **A**, **B**, **C**, and **D** matrices. In choosing a sample time, note that it is desired that the sampling frequency be fast compared to the dynamics of the system. One measure of a system's "speed" is its closed-loop bandwidth. A good rule of thumb is that the sampling time be smaller than 1/30th of the closed-loop bandwidth frequency which can be determined from the closed-loop Bode plot. Assuming that the closed-loop bandwidth frequencies are around 1 rad/sec for both the cart and the pendulum, let the sampling time be 1/100 sec/sample. The discretization method we will use is the **zero-order hold** (|'zoh'|). For further details, refer to the Introduction: Digital Controller Design page. Now we are ready to use |c2d| function. Enter the following commands into an m-file. Running this m-file in the MATLAB command window gives you the following four matrices representing the discrete time state-space model.

```
In [3]: A = numpy.array(
            [
                [0, 1, 0, 0],
                [0, -(I + m * l ^ 2) * b / p, (m ^ 2 * g * l ^ 2) / p, 0],
                [0, 0, 0, 1],
                [0, -(m * l * b) / p, m * g * l * (M + m) / p, 0],
            ]
        )
        B = numpy.array([[0], [(I + m * l ^ 2) / p], [0], [m * l / p]])
        C = numpy.array([[1, 0, 0, 0], [0, 0, 1, 0]])
        D = numpy.array([[0], [0]])
        # M = 0.5;
        # m = 0.2;
        # b = 0.1;
        # I = 0.006;
        # g = 9.8;
        # l = 0.3;
        # p = I*(M+m)+M*m*l^2; %denominator for the A and B matrices
        # A = [0      1              0           0;
        #      0 -(I+m*l^2)*b/p  (m^2*g*l^2)/p   0;
        #      0      0              0           1;
        #      0 -(m*l*b)/p       m*g*l*(M+m)/p  0];
        # B = [     0;
        #      (I+m*l^2)/p;
        #           0;
        #         m*l/p];
        # C = [1 0 0 0;
        #      0 0 1 0];
        # D = [0;
        #      0];
        # states = {'x' 'x_dot' 'phi' 'phi_dot'};
        # inputs = {'u'};
```

```
# outputs = {'x'; 'phi'};
# sys_ss = ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',outputs);
# Ts = 1/100;
# sys_d = c2d(sys_ss,Ts,'zoh')
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-3-19094c55f3e7> in <module>
----> 1 A = numpy.array(
      2     [
      3         [0, 1, 0, 0],
      4         [0, -(I + m * l ^ 2) * b / p, (m ^ 2 * g * l ^ 2) / p, 0],
      5         [0, 0, 0, 1],


NameError: name 'numpy' is not defined
```

Now we have obtained the discrete state-space model of the form:

$$\begin{bmatrix} x(k+1) \\ \dot{x}(k+1) \\ \phi(k+1) \\ *\dot{\phi}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0.01 & 0.0001 & 0 \\ 0 & 0.9982 & 0.0267 & 0.0001 \\ 0 & 0 & 1.0016 & 0.01 \\ 0 & -0.0045 & 0.3119 & 1.0016 \end{bmatrix} \begin{bmatrix} x(k) \\ \dot{x}(k) \\ *\phi(k) \\ *\dot{\phi}(k) \end{bmatrix} + \begin{bmatrix} 0.0001 \\ 0.0182 \\ 0.0002 \\ 0.0454 \end{bmatrix} u(k)$$

$$\mathbf{y}(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \dot{x}(k) \\ *\phi(k) \\ *\dot{\phi}(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u(k)$$

1.2 Controllability and observability

The next step is to check the controllability and the observability of the system. For the system to be completely state controllable, the controllability matrix

$$C = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}$$

must have the **rank of n**. The rank of the matrix is the number of independent rows (or columns). In the same token, for the system to be completely state observable, the observability matrix

$$O = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}$$