Homework 6 (due at the start of class November 1st, 2022)

1 Problem: System Identification (Unknown Model Structure)

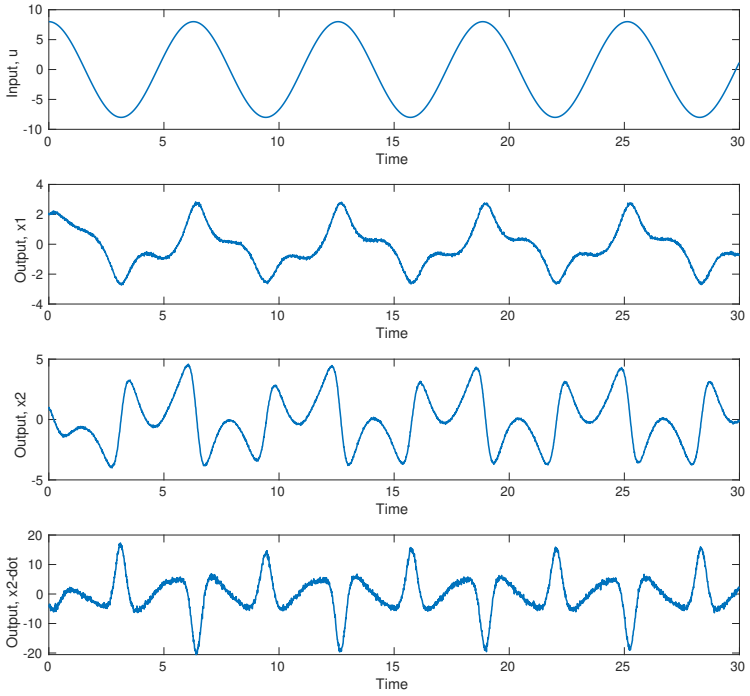
An experiment is performed in which a single degree of freedom dynamical system is driven by a scalar input u and the resulting (noisy) states and state-rates are recorded along with the time and input. The results of the experiment are stored in a MATLAB data structure within the file `h6_prob1.mat` with the following fields:

```
data =

  struct with fields:

    x1dot: [3000×1 double]
    x2dot: [3000×1 double]
       x1: [3000×1 double]
       x2: [3000×1 double]
        u: [3000×1 double]
        t: [3000×1 double]
       x0: [2×1 double]
```

plotted below:



You postulate that the system is second-order, time-invariant, and control-affine. Let x_1 denote the state and x_2 denote the state-rate. The (noise-free) model of the system you wish to find

is $f(x_1, x_2)$ where

$$\dot{x}_1 = x_2 \tag{1}$$
$$\dot{x}_2 = f(x_1, x_2) + u \tag{2}$$

1. Use stepwise regression (via the `stepwiselm` MATLAB command) to find a function $f(x_1, x_2)$ that best fits the data. Consider only terms that are up to cubic in order ($x_1 x_2$, x_2^3, etc.) by specifying `'poly33'` as an option with `stepwiselm`. Analyze the output of `stepwiselm` (a table is produced in the MATLAB Console) and state the form of the model. Ignore any terms that have a p-value reported in the table as greater than 0.001. Note that the output produced is specified in Wilkinson Notation.

2. Simulate the system using the model you determined and plot the results overtop of the experimental data for $x_2(t)$. Hint: You may use Euler's method to simulate the system rather than ode45 for ease of implementation with the provided control input history.

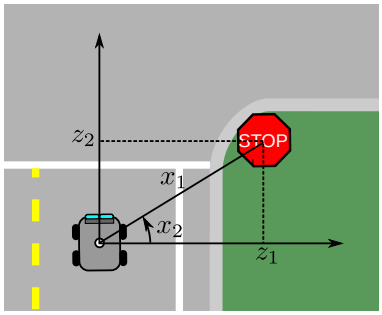
3. Bonus: State the name of this famous ODE

2 Problem: Random Vector Transformations

An autonomous car is equipped with a sensor that measures range x_1 and bearing x_2 (measured CCW from the horizontal) to nearby obstacles. The sensor is located at the origin and a range/bearing measurement maps to a planar relative z_1-z_2 according to:

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_1 \cos x_2 \\ x_1 \sin x_2 \end{bmatrix} \tag{3}$$

Suppose that at the particular instant shown in Fig. 2 the sensor reports an obstacle at a range of 150 m and at a bearing $30\pi/180$ rad. The range/bearing measurements are statistically independent random variables $X_1 \sim \mathcal{N}(\mu_{x_1}, \sigma_{x_1}^2)$ with $\mu_{x_1} = 150$ m and $\sigma_{x_1} = 1$ m, and $X_2 \sim \mathcal{N}(\mu_{x_2}, \sigma_{x_2}^2)$ where $\mu_{x_2} = 30\pi/180$ rad. and $\sigma_{x_2} = 5\pi/180$ rad. Let $\boldsymbol{x} = (x_1 \;\; x_2)^{\mathrm{T}}$ and $\boldsymbol{z} = (z_1 \;\; z_2)^{\mathrm{T}}$ be random vectors related by the system (3).



1. State the mean $\boldsymbol{\mu_x}$ and covariance $\boldsymbol{P}_x$ of the random vector $\boldsymbol{x}$.

2. Compute the first-order approximate mean $\boldsymbol{\mu_z}$ and covariance $\boldsymbol{P}_z$ of the random vector $\boldsymbol{z}$.

3. Simulate $N = 500$ random measurements from the distributions $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{P}_x)$ you deter-

mined above (use the `randSamplesWithMeanCov.m` function provided). Plot these samples with the x_1 values on the x-axis and the x_2 values on the y-axis. Use circular markers without a connecting line. Use a large legible font size and label axes.

4. Propagate each sample from Step 3 using the system (3). Plot the resulting samples with the z_1 values on the x-axis and the z_2 values on the y-axis. Use circular markers without a connecting line. Use a large legible font size and label axes.

5. Compute and mean the covariance of the samples in Step 4 using

$$\boldsymbol{\mu}_z = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{z}_i \tag{4}$$

and

$$\boldsymbol{P}_z = \frac{1}{N-1}\sum_{i=1}^{N} (\boldsymbol{z}_i - \boldsymbol{\mu}_z)(\boldsymbol{z}_i - \boldsymbol{\mu}_z)^{\mathrm{T}} \tag{5}$$

How does this compare to your result in Step 2? What if you increase the number of samples? Explain your result.

6. Does the distribution in the x_1, x_2 plane look Gaussian? Comment on how the distribution changes if you make the bearing measurement noise 50% larger or smaller and similarily with the range noise.

3 Problem: Scalar Kalman Filter

Consider the following scalar, linear, time-invariant, discrete-time system

$$x_k = Fx_{k-1} + Gu_{k-1} + w_{k-1} \tag{6}$$

$$y_k = Hx_k + v_k \tag{7}$$

with zero-mean Gaussian process noise that is with variance $E[w_k^2] = \sigma_w^2$ and zero-mean Gaussian measurement noise with variance $E[v_k^2] = \sigma_v^2$. Both sources of noise are uncorrelated in time. Let the initial state estimate mean and covariance be $\hat{x}_{0|0}$ and $\hat{\sigma}_{0|0}$. Determine expressions for each of the quantities listed below. Each expression should be in terms of the variables of the problem statement: $F, G, H, \sigma_w^2, \sigma_v^2, u_0$ and the initial guess $\hat{x}_{0|0}$ and $\sigma_{0|0}^2$. Since this is a scalar system your expression should simplify accordingly (i.e., no matrix transposes or inverses).

1. the predicted state at time $k = 1$ given the information at time $k = 0$ as well as the corresponding covariance

2. the Kalman gain

3. the (posterior) corrected state and covariance at time $k = 1$ given all information available at time $k = 1$

Also, answer the following:

4. What are the range of possible values for the scalar Kalman gain? How is it affected by the sensor measurement noise being large (noisy sensor) or being small (accurate sensor)?

5. How is the posterior mean and variance affected by the sensor measurement noise being large (noisy sensor) or being small (accurate sensor)?

4 Problem: Discrete-Time Kalman Filter Algorithm

Write a function of the form shown below (by modifying the template provided `discreteKalmanFilter.m`) to implement one iteration of a discrete-time Kalman filter. Test your function using the sample input/output provided below to ensure it is correct.

```
function [xk_k, Pk_k, K, xk_km1, Pk_km1] =
                        discreteKalmanFilter(xkm1_km1, ukm1, Pkm1_km1, yk, F, G, H, Q, R)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Description: discreteKalmanFilter.m
%   Perform one iteration of the discrete time Kalman filter
%
%  Consider the discrete time system given by:
%   x(k+1)  = F(k)x(k) + G(k)u(k) + w(k) (motion model)
%   y(k)    = H(k)x(k) + v(k)         (measurement model)
%   w(k)  ~ (0, Q)   (zero mean process noise)
%   v(k)  ~ (0, R) (zero mean measurement noise)
%
% Inputs:       xkm1_km1    : estimate of x at time-step (k-1) given data
%                             at (k-1)
%               ukm1        : control applied at time-step (k-1)
%               Pkm1_km1    : covariance of state x at time-step (k-1)
%                             given data at (k-1)
%               ykp1        : measurement at time-step (k)
%
%               F           : F state-transition matrix
%               G           : G control input matrix
%               Q           : process noise covariance matrix
%               H           : H deterministic measurement model
%               R           : measurement noise covariance matrix
%
% Outputs:      xk_k        : estimate of x at time-step (k) given data at
%                             (k) (i.e., posterior)
%               Pk_k        : covariance of state x at time-step (k) given
%                             data at (k)
%               K           : Kalman gain
%               xk_km1      : estimate of x at time-step (k) given data
%                             at (k-1) (i.e., motion update)
%               Pk_km1      : covariance of state x at time-step (k) given
%                             data at (k-1)
```

A test case input/output is provided below:

```
clear; close all; clc;
xkm1_km1 = [1; 2];
ukm1 = 1;
```

```
Pkm1_km1 = [10 5; 5 2];
F = [1 2;3 4];
G = [5; 6];
H = [7 8];
Q = eye(2,2);
R = 1;
yk = 0;
[xk_k, Pk_k, K, xk_km1, Pk_km1] =
discreteKalmanFilter(xkm1_km1, ukm1, Pkm1_km1, yk, F, G, H, Q, R)

xk_k =
   2.399844060107741
  -2.098950949815706
Pk_k =
   0.593386730932803  -0.514601644457045
  -0.514601644457038   0.461865608165603
K =
   0.036893960873263
   0.092713354125319
xk_km1 =
    10
    17
Pk_km1 =
    39    96
    96   243
```