Contents


```
to do thing:

pandoc --toc  --toc-depth=6 -V fontsize=10pt --pdf-engine pdflatex -V geometry:"left=1.5cm,right=1.5cm,top=2c
```

ECE 8 Notes, Lecture 11 Midterm Review

What are we capable of doing?

- Matlab usage and coding
  - Entering data in Matlab
    * from here we can make vectors, then Matricies, then plot
    * We enter data in variables "letters/characters"
    * `a = 7;`
  - vectors in Matlab
    * we can use real numbers
    * Instead of using variables with one element only, we can create vectors
      · `p =[ 5 3 9 0 10 ]` this is a row vector (bc of the spaces)
        · a space between two entries in Matlab is the same as a comma
        · It looks the way it is written
      · `q = [ 5 ; 3 ; 9 ; 0 ; 10 ];` this is a column vector (bc of the semicolons ;)
  - In addition to vectors we can create larger versions "Maticies"
    * Previously we defined a Matrix A like this:

      ```
      A(1,1) = 1;
      A(1,2) = 2;
      A(2,1) = -1;
      A(2,2) = 3;
      ```
  - plotting in Matlab
    * need to create vectors
    * Once a figure is created in Matlab, we can draw our data (ask matlab to draw) on a plot
    * Command:

      ```
      % If we have a vector p
      % or any variable/ vector defined, the simplest plot is:

      plot(p) % generates a plot of p
      ```
    * Best Practice is to set a figure to control

      ```
      figure(1) % figure 1 label 1
      plot(p)   % this kind of one vecotr plot will be an interpolation
                %     between data points. aka Matlab plots the points and
                %     connects them with lines
                %     y-axis = p
                %     x-axis = %% of points, in this case 1,2,3,4,5
      ```

```
%                 for a length 5 vecotor
```

Transition to Matlab examples

- See his file for lecuture 11 Review Review.m
- Code info:
  - Best practices use the following lines:

    ```
    close all;
    clear all;
    ```

  - eneter data by setting variables

    ```
    a = 7; % defines 'a' as a value 7
    % the simicolon supresses output from the Command Window
    % This variable can be seen in Workspace
    ```

  - Setting a vector
    * A vector can store more than one element
      - Consider making the vector `p = [ 5 3 9 0 10 ]`

      ```
      p(0) = 5; % This is not the first entry!

      % Matlab does not like zero, it does not represent
      %   the first entry (that's only true in other programming languages)

      % Correct way to set the first entry (this is how to do it one at a time)
      p(1) = 5; % first entry is equal to  5 and p(0) is not allowed

      p(2) = 3;
      p(3) = 9;
      p(4) = 0;
      p(5) = 10;

      % similarly all at once:
      p = [ 5 3 9 0 10 ];
      ```

      - Recall the comparison at discrete time `k`:

        ```
        p(1) = 5;  % position at k = 1 "intial position"
        p(2) = 3;  % position at k = 2
        p(3) = 9;
        p(4) = 0;
        p(5) = 10; % postition at k = 5 and so on
        % in the workspace you'll find p = [5,3,9,0,10]
        %       p is a 1 row 5 column vector
        %       row vector of dimension 5
        ```

      - Command to check what kind of vector we have

        ```
        % From the Command Window
        >> size(p)
        ans =
            1   5
        % aka 1 row 5 columns (or a row vecotor of Dimension 5)
        ```

    * Define a column vector, instead of a row
      - instead of `p = [ 5 3 9 0 10 ]` we want:
        - p = |5 | |3 | |9 | |0 | |10|

· Code:

```
% To create a column vector we need to add semicolons after each entry
% Let's choose a differnet variable
q = [ 5 ; 3 ; 9 ; 0 ; 10 ];  % this is now a column vector

% 5 rows and 1 column
% Since we already have the values defined above, we can also use:

q = [ p(1) ; p(2) ; p(3) ; p(4) ; p(5)] ;

% looks like: - q = |5 |
%                   |3 |
%                   |9 |
%                   |0 |
%                   |10|
```

· Command to check what kind of vector we have

```
% From the Command Window
>> size(q)
ans =
    5   1
% 5 rows and 1 column
```

· When did we use column vecotors?

· Answer:

* Cosider using the command for the transpose operation

· review matix operations and transpose

· Command for Transpose:

```
% using the vector p = [ 5 3 9 0 10 ]

% p transpose (turns a row vector into a column vector)

p'; % the apostrophe is the notation for transpose in Matlab
```

· Using this notation in the command window will let us see it better:

```
>> p'
ans = 5
      3
      9
      0
      10

% BUT notice this is the same as q, our column vecotor! (since they have the same entries)

% Therefore p' = 1 and q' = p
```

· What is more general than vectors? Answer: Matricies

· Code:

```
% Recall: A(row position , column position)

A(1,1) = 1;     % 1st row 1st column is 1
A(1,2) = 2;     % 1st row 3nd column is 2
A(2,1) = -1;    % 2nd row 1st column is -1
```

```
A(2,2) = 3;     % 2nd row 2nd column is 3

% looks like a 2x2 matrix
%        A = |  1      2 |
%            | -1      3 |
```

· Command to check what kind of Matrix we have

```
% From the Command Window
>> size(A)
ans =
    2   2
% 2 rows and 2 columns
```

· Check a specific column in the command window:

  · This is how you access your data in a matrix

```
>> A(1,2)

ans = 2
```

– Plotting

  * Code:

```
figure(1)
plot( p, '*' ) % recall '*' is just giveing the data points a star marker
```

  * Code: plot 2 lines on the same figure

```
figure(1)       % Create a figure to hold the plots

plot(p)         % Plot the vecotor p: y-axis, and [ 1 2 3 4 5 ] x-axis

hold on         % hold for another plot on the same figure

plot( p, '*' )  % 2nd plot, same as the previous but with markers
```

  * Adding labels:

```
% Plotting p as a function of its entries
% p vs # of entries
% p on y-axis and # of entries on x-axis
figure(1)       % Create a figure to hold the plots

plot(p)         % Plot the vecotor p: y-axis, and [ 1 2 3 4 5 ] x-axis

hold on         % hold for another plot on the same figure

plot( p, '*' )  % 2nd plot, same as the previous but with markers

% add Labels:
xlabel('number of data points')     % x-axis
ylabel('p')                         % y-axis
title('plot of p')                  % tiles
```

  * Create: a vector that indexes our data

    · To do this means we want control over our x-axis

    · this is hinting at our time axes discrete time `k` or continous time `t`

    · So, Suppose we want to plot the row vector `p` as a function of discrete time `k`

· aks the plot is `p` on the y axis with `k` on the x-axis

· See professors diagram of p vs k in plotting notes

· Code to create this:

```
% Goal is: plot(k,p)
% Plottin p as a function of discrete time where
% p(1) corresponds to k = 0
% p(2) corresponds to k = 1
% p(3) corresponds to k = 2
% p(4) corresponds to k = 3
% p(5) corresponds to k = 4

% Create a row vector k

k = [0 1 2 3 4];

figure(2) % creates Figure 2

plot(k,p) % plots p vs k

hold on

plot( k, p, '*')

% add Labels:
xlabel('k')     % x-axis
ylabel('p')                         % y-axis
title('plot of p vs discrete time')                 % title
```

* Plotting continous time `t` with Delta = 0.1 secounds

· Code:

```
% Goal is: plot(t,p)
% Plottin p as a function of discrete time where
% p(1) corresponds to k = 0 (or equivalently, t = 0)
% p(2) corresponds to k = 1 (or equivalently, t = Delta)
% p(3) corresponds to k = 2 (or equivalently, t = 2*Delta)
% p(4) corresponds to k = 3 (or equivalently, t = 3*Delta)
% p(5) corresponds to k = 4 (or equivalently, t = 4*Delta)

% Create a row vector t
Delta = 0.1;
t = [0 Delta 2*Delta 3*Delta 4*Delta];

% In terms of k

k = [0 1 2 3 4]; % from here we see we can get t by multiplying by Delta
k = [ 0*Delta 1*Delta 2*Delta 3*Delta 4*Delta ] % Therefore

t = Delta*k

% We can see that


figure(3) % creates Figure 3
```

```
plot(t,p) % plots p vs t

hold on

plot( t, p, '*')

% add Labels:
xlabel('t')     % x-axis
ylabel('p')                          % y-axis
title('plot of p vs Continous time')                  % title
```

* Subplots
    - Plot figure 2 and figure 3 from above that has 2 plots in the form
        - 1 row 2 columns
    - Code:

```
% subplot(rows, column, desired position)
figure(4)

subplot(1,2,1) % Plots row 1 of 2 columms position 1 coloumn 1
plot(k,p)

subplot(1,2,2) % Plots row 1 of 2 columns position 2 column 2
plot(t,p)
```

Dynamical Models of Robotics Systems

- Loops:
    - for loops
        * In discrete time systems and Dynamical Models
    - while loops
- Conditionals
    - if/else
    - See old notes for more information
- Dynamic Equations:
    - A differential equations can be used to model the change of physical properties of a robotic system
    - for example:
        * using physics, the change of position of a brick on the ground in terms of velocity is governed by :

$$\frac{dp_x}{dt} = v_x$$

    - A discrete time model of this continous-time model is given by the Forward Euler model:

$$p_{x,k+1} = p_{x,k} + \Delta v_{x,k}$$

        * Delta is the step size in time
    - Given the follwoing:
        * inital position $p_{x,0}$
        * stepsize Δ
        * velocity input (all of them over time): $v_{x,0}, v_{x,1}, v_{x,2}, \ldots$
    - We can calculate the trajectory of the sytem given by:
        * Recall: a trajectory is all the points (sequence of points) that make up the position vector

$$p_{x,0}, p_{x,1}, p_{x,2}, \ldots$$

- * Where the first term is the inital condition and,

$$p_{x,1} = p_{x,0} + \Delta v_{x,0}$$

$$p_{x,2} = p_{x,1} + \Delta v_{x,1}$$

$$p_{x,2} = p_{x,2} + \Delta v_{x,2}$$

  * and so on

- In Matlab:
  - Calculate N steps of the position p_x of the model in the notesˆabove
  - N and Delta is either given or asked for
  - Here we are given N = 10,000, Delta = 0.001
  - Code

```
% Step 1: Define all the givens
N = 10000;      % given number of iterations/computations to preform
px0 = 1;        % given inital condition
Delta = 0.001;  % given step size

% calculate px1,px2,...,pxN
% we can do this one at a time or use a loop (for-loop)

% We define an empty vecotr px and use a loop to fill the entries using
%   the model p_x,k+1 = p_x,k + Delta * v_x,k

% Step 2: Initialize the position vector with px0 as the 1st entry

px(1) = px0;

% Step 3: Initialize/ Define the veloctity vector

% for this example velocity is constant for all steps
vx = 1;  % This DOES NOT WORK

% CORRECT METHOD: make a vector
vx = ones(1,N); % Row vector with N columns/entries all equal to 1

% Step 4: Initialize the discrete time k row vector

k(1) = 0 % initial time is zero

% Step 5: for-loop to create filled position vector
for i = 1:N
    % model p_x,k+1 = p_x,k + Delta * v_x,k
    px(i+1) = px(i) + Delta * vx(i);

    % Define time (discrete time k)
    k(i+1) = k(i) + 1;
end

% Step 6: continuous time
t = delta * k;

% Step 7: Plot
```

```
figure(1)

plot(k,px)

figure(2)

plo(t,px)
```

- Side note: Recall on Tuesday 10/25/22
  - if we're asked to compute for 10 seconds we must find N
  - N is not equal to 10 seconds
  - Given delta = 0.1
    - N = 10 seconds/ delta
    - N = 10 seconds/ 0.1
    - this gives us the number of 0.1 inside of 10 seconds i.e. the number of N

Feedback Control

- A way to achive our desired key variables (usually labeled with subscript d)
- See the FEEDBACK CONTROL ARCHITECTURE DIAGRAM
- See Lecture 10 for proportional control
- Know: what it is but now how to calculate it

End of class notes

- Evey quiz has extra credit
- Potentailly we will have an extra credit quiz coming up