ECE-420: Discrete-Time Control Systems
Project Part A: Simulating a Plant

Due: *At the beginning of class Friday September 19th (e-mail me a memo)*

In this part of the project you will simulate a few discrete-time systems in both Matlab and Simulink to see how the embedded systems toolbox works. You will need to download the project files from the class website for this. This project was written for Matlab R2012b with a 64 bit operating system. You may need help getting this project to work since the embedded system toolbox requires an additional C compiler that works with Matlab. You should start project early so you can get the system all working.

Mathematical Background: Consider a simple discrete-time transfer function with input $R(z)$ and output $Y(z)$,

$$G(z) = \frac{Y(z)}{R(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{B(z)}{A(z)}$$

Cross multiplying we get

$$Y(z) + a_1 z^{-1} Y(z) + a_2 z^{-2} Y(z) = b_0 R(z) + b_1 z^{-1} R(z) + b_2 z^{-2} R(z)$$

In the time-domain this becomes

$$y(n) = -a_1 y(n-1) - a_2 y(n-2) + b_0 r(n) + b_1 r(n-1) + b_2 r(n-2)$$

In Matlab, the A and B arrays are then $A = \begin{bmatrix} 1 & a_1 & a_2 \end{bmatrix}$ $B = \begin{bmatrix} b_0 & b_1 & b_2 \end{bmatrix}$

In Matlab, there will be an equal number of elements in both arrays and they are row arrays (because of the way we constructed them). Let's assume that N denotes the maximum number of coefficients in the transfer function. Then to implement this algorithm to find the current output $y(n)$ we need the current input $r(n)$ and the previous N-1 inputs and outputs. This fact is important to understand what is going on in the tapped delays in the Simulink model.

We can then determine the current output using the following Matlab loop

```
y(n) = B(1)*r(n)
for k=2:N
    y(n) = y(n) - A(k)*y(n-k+1) + B(k)*r(n-k+1)
end;
```

Note that Matlab starts its indices with a 1 instead of 0, but you get the idea.

1) Open the files **openloop_DE.slx** and **openloop_driver.m.** These files generate both a Matlab and a Simulink simulation of the unit step response for the discrete-time transfer function

$$G_p(z) = \frac{1}{z-0.8} = \frac{z^{-1}}{1-0.8z^{-1}}$$

In this case we have $A = \begin{bmatrix} 1 & -0.8 \end{bmatrix}$, $B = \begin{bmatrix} 0 & 1 \end{bmatrix}$, $N = 2$, and $y(n) = 0.8y(n-1) + r(n-1)$

Run **openloop_driver.m**, you should get the graph shown in Figure 1, however, it may take awhile. This figure shows the results from the Matlab and Simulink simulations are identical, which is a good way to check your answers.

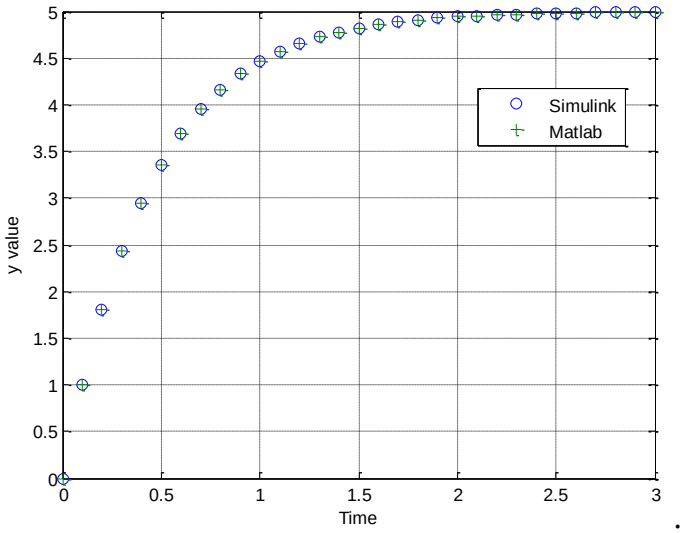

Figure 1. Open loop response of the system.

2) Now open up **openloop_DE.slx** (double click on it). It should look like Figure 2. We will be going through a number of the elements in this model.

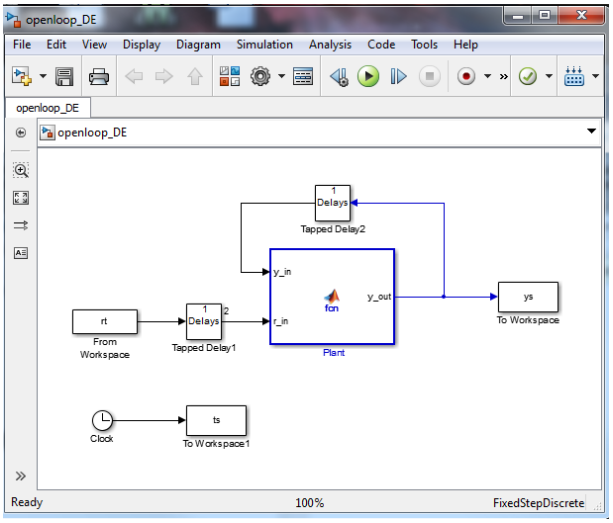

Figure 2. openloop_DE.slx

Before we get too far, there is a sort of logic to some of the parameter names used in this model. First of all, **Ap** and **Bp** refer to the **A** and **B** vectors of coefficients for the **plant** transfer function. This is to (later) differentiate them from **Ac** and **Bc**, which will be vectors of coefficients for the **controller** transfer function. Similarly, the parameter **Nbp** indicates the number of elements in the **Bp** array (and by default, the **Ap** array). Later we will use **Nbc** to indicate the number of elements in the **Bc** and **Ac** arrays. The variable **r_in** is the input to the plant, the variable **y_out** is the output of the plant, and the variable **y_in** is the output fed back to the input (with delays).

Starting at the left of the model (Figure 2), the input goes into a Delay Block. If you click on the delay block you will get the parameter block shown in Figure 3. This figure also shows what many of the parameters mean.

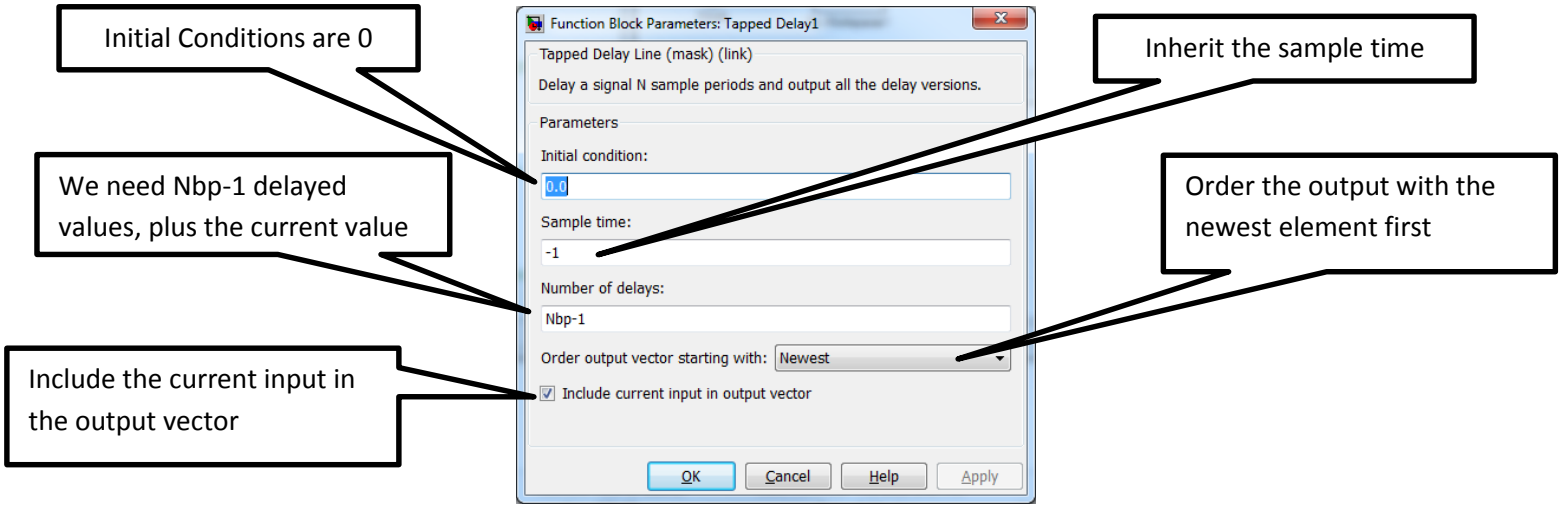

Figure 3. Parameter block for first delay, with explanations.

For Nbp = 4, the output of this delay block looks like the vector depicted in Figure 4. Note that Matlab uses a ***column vector of values*** for the output of a Delay block. This is important to remember!

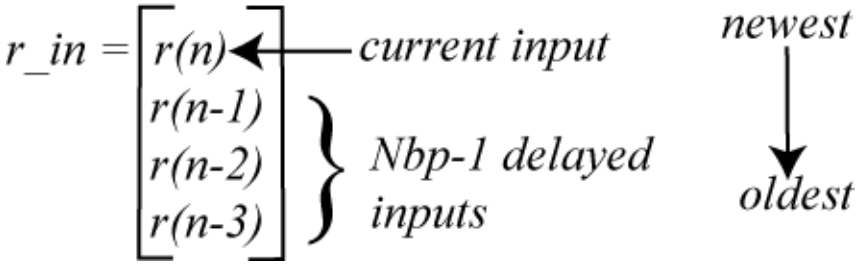

Figure 4. Output of the first Delay block, for Nbp = 4. The data is sorted from newest to oldest, the data includes the current input, and there are Nbp-1=3 delays. Note that Matlab uses a column vector of values for the output of a Delay block.

If we look at the Delay block used in feeding the output back into the system (at the top of Figure 2), it looks like that shown in Figure 5. This is very similar to the Delay block shown in Figure 3. The only difference is the current input is not included in the output. The output of this block for Nbp = 4 is shown in Figure 6.

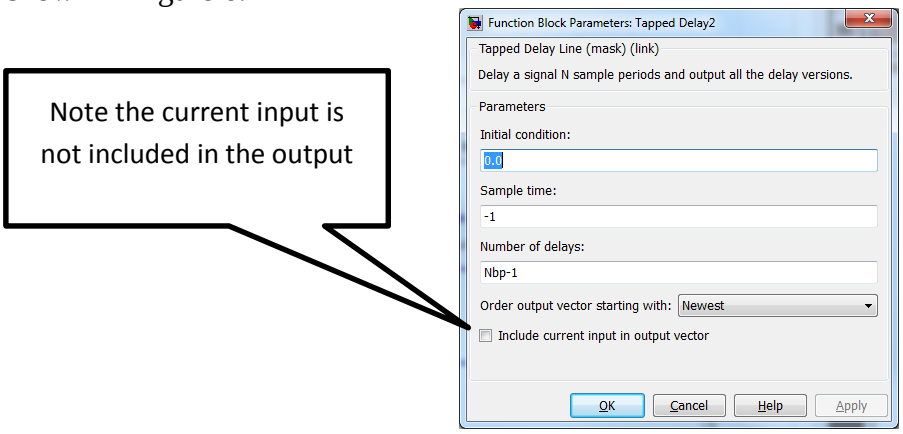

Figure 5. Delay block used for feeding the output back into the system.

$$y\_in = \begin{bmatrix} y(n-1) \\ y(n-2) \\ y(n-3) \end{bmatrix} \Big\} \; Nbp\text{-}1 \; delayed \; inputs \qquad \begin{matrix} newest \\ \downarrow \\ oldest \end{matrix}$$

Figure 6. Output of the delay block feeding the output back into the system for Nbp = 4. Note that Matlab uses a column vector of values for the output of a Delay block.

Now click on the plant (the large block in the center of Figure 2) and you will get the innocent looking piece of code for implementing a discrete-time transfer function shown in Figure 7. There are five things "passed" to this function: y_in, r_in, Bp, Ap, and Nbp.

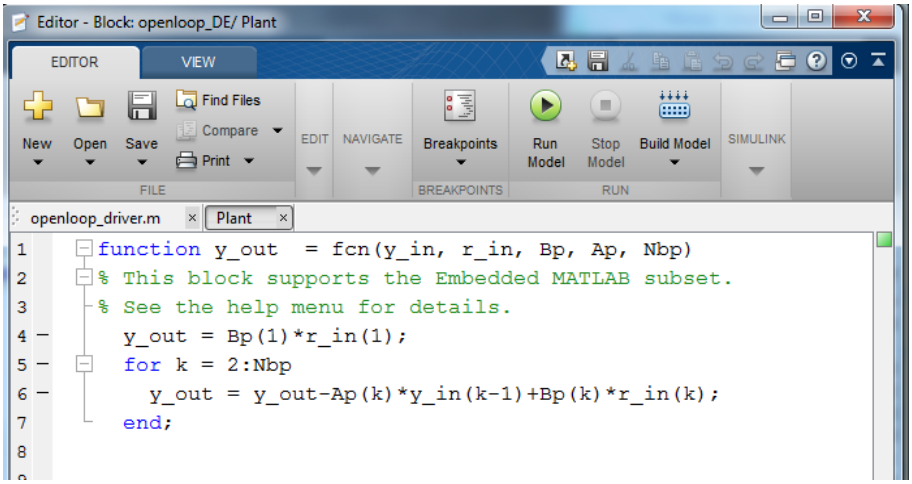

Figure 7. Code inside the plant block for implementing a discrete-time transfer function.

Click on the Simulink and then the Edit Data icon, as shown in Figure 8, to access the variables.

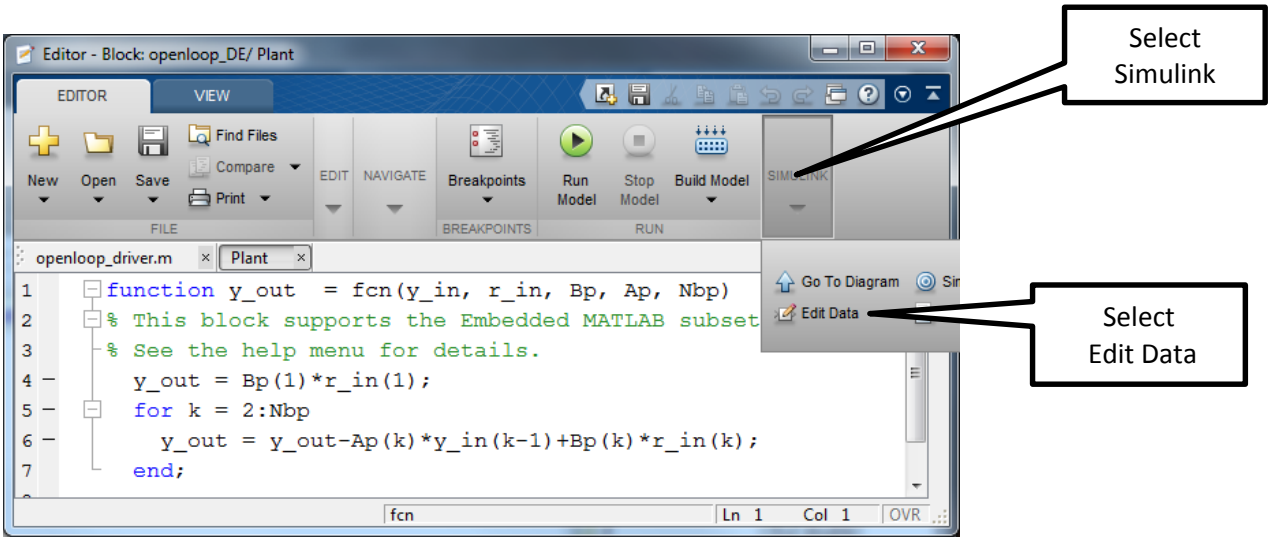

Figure 8. Accessing the "passed" items to this block via the Edit Data icon.

Once you click on the Edit Data icon, you will get a window like that shown in Figure 9.

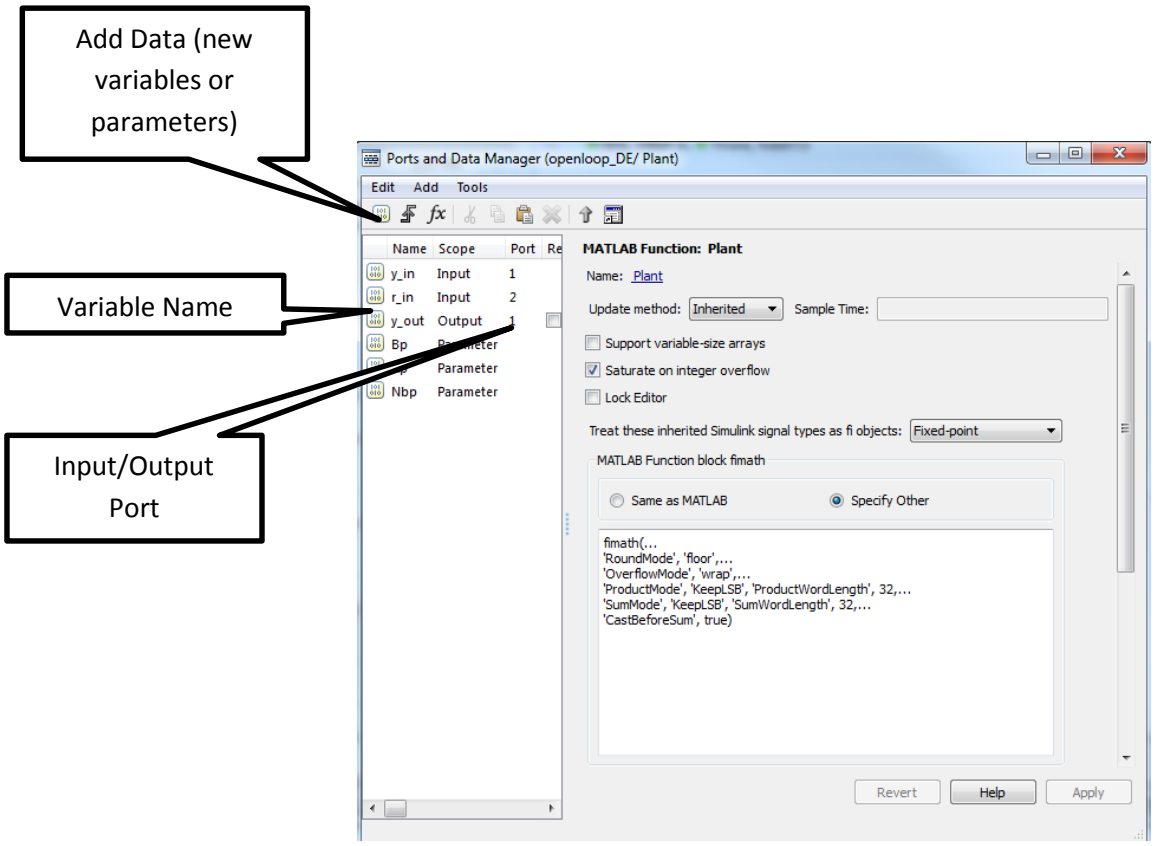

Figure 9. Modifying variables and parameters

If you select one of the variables (such as **y_in**), you will get a figure like that in Figure 10. Click in the left-half plane to get back to the previous window.

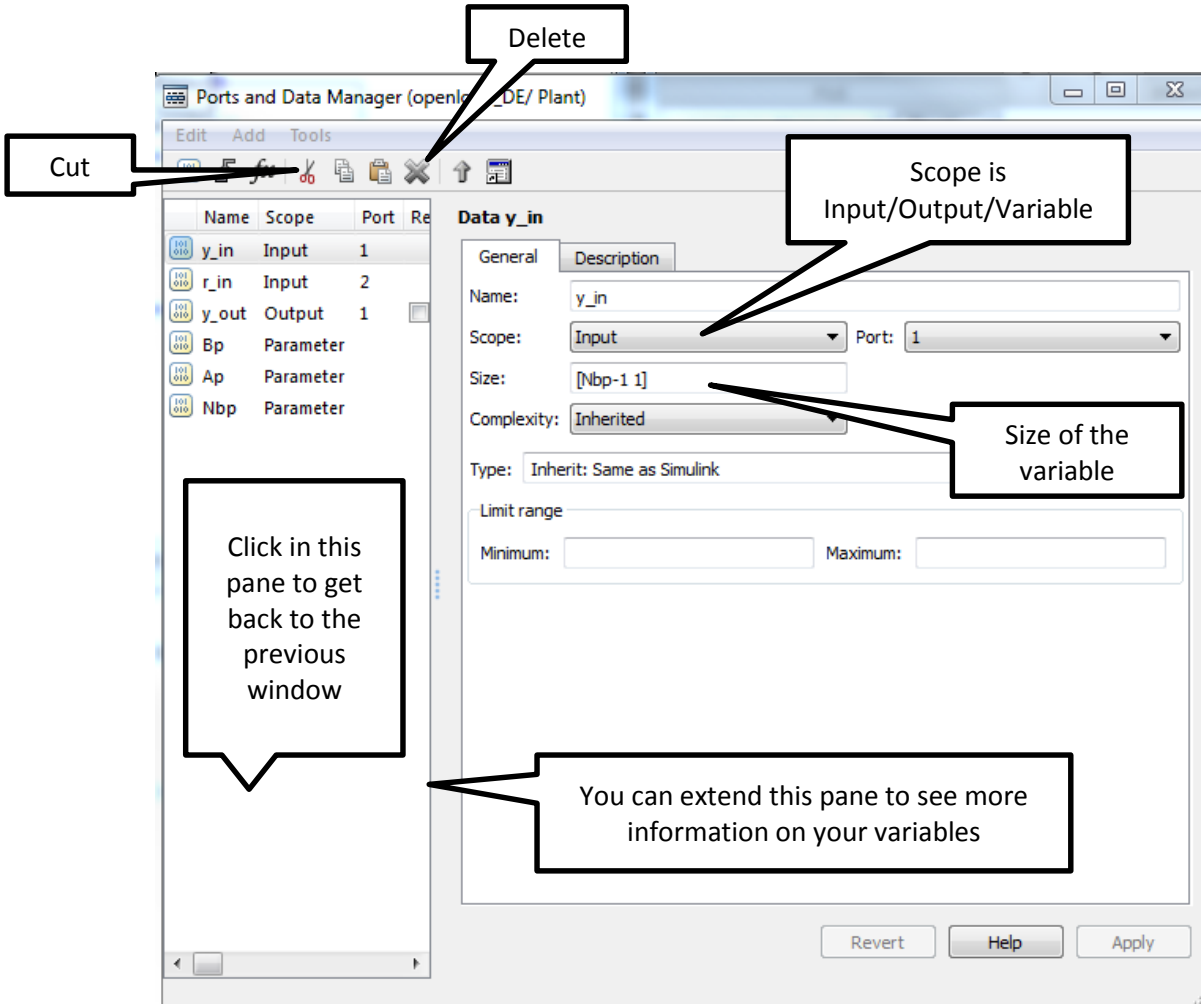

Figure 10. Information on simulation variables.

Referring to Figures 9 and 10,

- If the **scope** is a **parameter**, then you should not expect it to change as the simulation is running. Thus, **Nbp**, **Bp**, and **Ap** are parameters. If the scope is an input or output, then you should expect it to change as the simulation is running, such as **y_in**, **r_in**, and **y_out**.

- The **port** indicates the order in which the variable will appear in the left (input) or right (output) side of the block. For this simulation, the **port** of **y_in** is **one** and the **port** of **r_in** is **two**, and this is reflected in how they are placed on the left of the **plant** block (See Figure 2).

- Most likely you will be changing the **Size** of the variables more than anything else. It is a good idea (for the problems you are likely to get) to try and not hard-code numbers and use parameters

like Nbp, Nbp-1, etc. Note that for this example, **Bp** and **Ap** are ***row vectors*** with size [1 Nbp], while **y_in** and **r_in** are ***column vectors***, with sizes [Nbp-1 1] and [Nbp 1], respectively.

- **Add Data** allows you to add new variables and parameters.

3) Modify the code as needed to determine the step response of the discrete-time plant

$$G_p(z) = \frac{1-0.2z^{-1}}{1-0.8z^{-1}}$$

Assume a sampling interval of 0.1 seconds and run the simulation for 3 seconds. You should get a plot like that shown in Figure 10. Include your plot in your memo

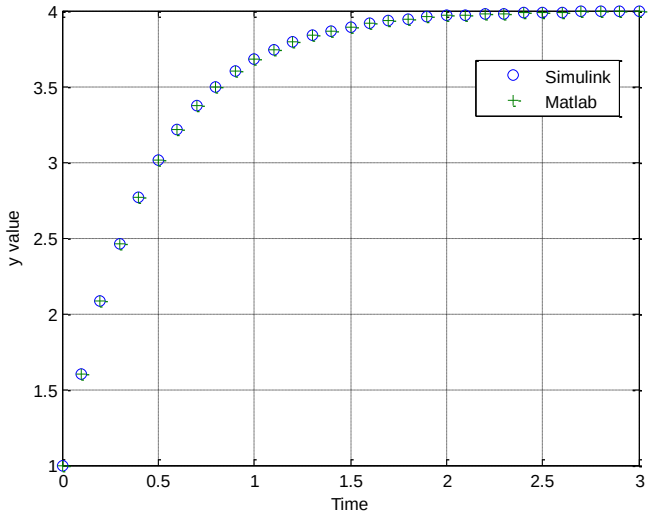

Figure 10. Step response for $G_p(z) = \frac{1-0.2z^{-1}}{1-0.8z^{-1}}$

4) Modify the code as needed to determine the step response of the discrete-time plant

$$G_p(z) = \frac{0.2z^{-1}+0.1z^{-2}}{1-0.1z^{-1}+0.6z^{-2}}$$

Assume a sampling interval of 0.1 seconds and run the simulation for 2 seconds. You should get a plot like that shown in Figure 11. Include your plot in your memo.

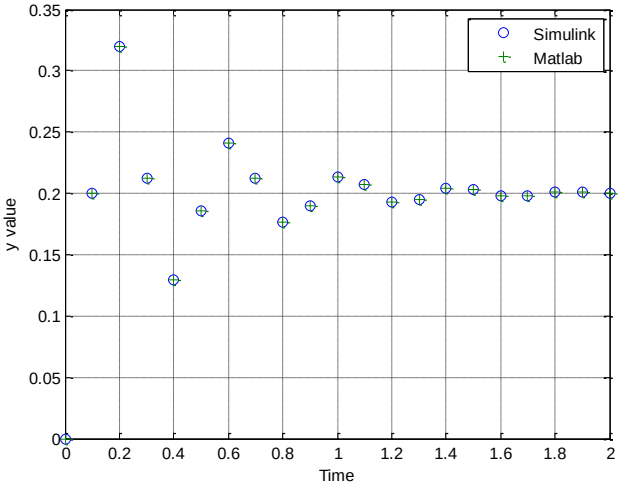

Figure 11. Step response for $G_p(z) = \dfrac{0.2z^{-1} + 0.1z^{-2}}{1 - 0.1z^{-1} + 0.6z^{-2}}$

To turn in: write a short memo including your graphs (with captions and figure numbers), and any suggestions you may have for improving this part of the project. e-mail me your memo.

Note; You should plan to refer to this project handout in future projects using the embedded system toolbox.