Guide to Good Lab Report Writing

1 General Formatting

- Different sections of the lab report should be easy to distinguish. (E.g. sections in this document).
  - Section titles should *not* be separated from content (i.e. do not put the title on the bottom of preceding page).
- General misspellings are not desirable, but are not going to hurt you significantly. What *will* hurt you are incorrect statements that demonstrate a poor conceptual understanding, incorrect references (wrong figure number, for example), or typos in equations or important numbers.
- Whatever style you choose to employ, *be consistent* throughout your report.
- *Proofread* before you turn in your report!

2 Results

- All results go in the Analysis section. This includes all calculations, plots, and answers to questions.
- Make sure to include each question *before* you answer them in your report. Paraphrasing is okay, as long as it is clear what it is you are answering.
- Important results should be clearly marked, e.g. using italics or bold text.
- Show the steps in your work and derivation! Someone reading your report should be able to follow your derivation without having to pull out a piece of paper and writing utensil. Just like coming across a typo in a textbook, theres nothing more frustrating than trying to reason your way through something thats just wrong, so make sure your work checks out.
- Whenever possible, explain your reasoning! It allows the reader to better interpret your results / conclusions and demonstrates your understanding of the material.
- In general, whenever you are asked to perform a task, your lab report should include *all* of your steps in completing the task. This includes calculation steps, important plots you generated, and code output. If applicable, explain your reasoning for the steps you are taking.

3 Plots

- All plots are expected to have a title, labeled axes, *and* a caption.
  - The caption may seem redundant with the plot title, but it gives an opportunity to provide additional information, as well as giving the plot a figure number that you can reference in your report.
  - Make sure said labels, titles, and captions are readable (do not shrink plot too much).
  - Use vector graphics (pdf, eps, svg) if possible.
  - Include units in axes labels when applicable.
- If you have more than one graph on the same plot, include a legend and make sure that you can easily distinguish which is which.

  - Ways to do this: 1) use distinct colors (and color printer), 2) use different line styles (solid, dashed, etc.), or 3) use text arrows (on figure, use Insert → Text Arrow)
  - Most possibly you'll print your report without color, so don't color code your line. Use `.-` or `--`.

- Make sure all relevant information is on the plot.
  - If you are tracking an input, it's probably a good idea to plot the input on the same plot.
  - If you are trying to meet certain criteria, it might be a good idea to show those on the plot as well: 1) overshoot line, 2) zoomed in graph for steady-state error, or 3) a text box on the plot with numerical values of criteria.
- The plot and the caption alone should be able to answer the following questions:
  - What is this a graph of?
  - What input was used?
  - If you were varying a parameter, which value is shown here?
  - What is important about the graph? (Why do we care?)

4 Simulink Models

- Can be placed in either the procedure or analysis section.
- Make sure all model parameters are specified:
  - If using variables, make sure they are defined in your lab writeup or in code.
  - Input/reference should be described in lab report or preferably in the caption.
- If the model is used in code, make sure to include the model name (.mdl) in the caption.
- Caption should describe what the model is and what it is being used for.

5 Code

- Can be placed in report or in an appendix.
  - If in appendix, makes sure to mention this and properly reference the code in your report.
- Should be placed inside a text box and captioned with a figure number. Preferably in a font such as `Courier` (looks like MATLAB font, so easier to distinguish).
- Code commenting is very helpful and greatly appreciated (use "`%`" in MATLAB).
- If your code prints output, include your final output values in your report!
- If you are typing your report in LaTeX, the mcode package[1] may be helpful.

[1] `http://www.mathworks.com/matlabcentral/fileexchange/8015-m-code-latex-package`

A Examples

The following examples are for different types of questions that will appear in your labs. They are mostly unconnected, so any references in one example will probably not actually be found in the other examples. Also, these were written in the way that we write labs. You definitely do not need to follow this formatting exactly, as long as you follow the guidelines above.

A.1 Equation Example

Question Apply the Laplace transform to the mass-spring-damper system with force input and derive the transfer function $X(s)/F(x)$.

Answer The equation of motion is given by

$$M\ddot{x} + b\dot{x} + kx = F$$

Assuming $x(0) = \dot{x}(0) = 0$, the Laplace transform is given by

$$Ms^2X(s) + bsX(s) + kX(s) = F(s)$$

Factoring out $X(s)$, we have

$$X(s)(Ms^2 + bs + k) = F(s)$$

Regrouping the terms,

$$\boxed{\frac{X(s)}{F(s)} = \frac{1}{Ms^2 + bs + k}}$$

A.2 Block Diagram Example

Question Create a Simulink block diagram of the plant in a negative feedback loop with a gain K as the controller.

Answer We created the Simulink model, shown in Figure 1, of the plant in a proportional feedback loop with proportional constant K.

Note Explanation of signals and other tidbits included in your figure caption *should* be repeated in the writeup.

A.3 Writing and Code Output Example

Linear interpolation improves the accuracy of our rise time calculation. Rise time is the time it takes the system to go from 10% to 90% of the steady-state value. Because our simulation is run in fixed time intervals, it is unlikely that $.1 * y_{ss}$ and $.9 * y_{ss}$ will fall exactly on one of our data points. We assume a

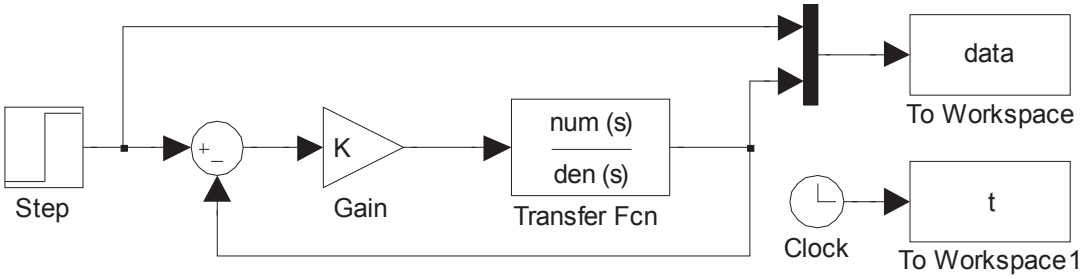

Figure 1: Plant in a proportional feedback loop with sep input of amplitude 1 (`PlantStep1.mdl`). The gain K and the polynomials `num(s)` and `den(s)` are defined in code (see Appendix)

linear interpolation between consecutive data points around our desired output value y_{des} and estimate the time t_{des} according to the linear interpolation assumption, as shown in Figure 2

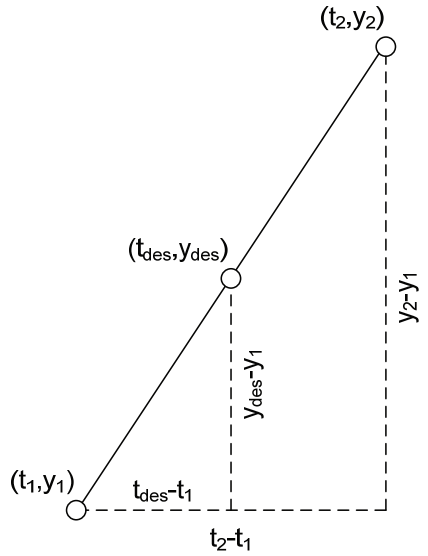

Figure 2: Diagram of variables used in linear interpolation with the desired time between data points (t_1, y_1) and (t_2, y_2)

The slope of the linear interpolation function is $(t_2 - t_1)/(y_2 - y_1)$. Thus the desired time satisfies the following equation:

$$t_{des} = t_1 + (y_{des} - y_1)\frac{t_2 - t_1}{y_2 - y_1}$$

Our code, `ee128prelab2.mm` is shown on the next page. We first iterated over integer values of K and found that the lower bound to meet the given criteria is between 17 and 17.1. Our final iteration, between 17 and 17.1, with a step of size 0.0001, is shown in the code. When run, we get the following results

```
>> ee128prelab2
K_crit = 17.050
Rise Time = 0.3999931
Overshoot = 3.32719%
```

Code example

Listing 1: MATLAB code to calculate rise time, overshoot, and error of step response of SysModel.mdl (plotpid.m)

```
%%% plotpid.m
%%% Justin Hsia – Fall 2006
%%% plots PID output and reference for specified values of k_p, k_i, k_d, and Tf
%%% – uses SysModel.m

% define PID constants and final time
k_p = 1.0;
k_d = 0.1;
k_i = 0.01;
Tf = 1.1;

% run simulation
sim('SysModel',Tf);
plot(t,y,'b',t,r,'k');

% calculate rise time using linear interpolation
thresh1 = 0.1*pi;
thresh2 = 0.9*pi;
index = find(y>thresh1,1);
t1 = t(index-1) + (t(index) - t(index-1))/(y(index) - y(index-1))*(thresh1 - y(index-1));
index = find(y>thresh2,1);
t2 = t(index-1) + (t(index) - t(index-1))/(y(index) - y(index-1))*(thresh2 - y(index-1));

% calculate overshoot and error
index = find(t>1,1); y1 = y(index-1) + (y(index) - y(index-1))/(t(index) - ...
    t(index-1))*(1 - t(index-1));
error = pi - y1;
os = max(y) - pi;

% output values to MATLAB console
sprintf('Kp = %f\nKd = %f\nKi = %f\n\nRise time = %f\n%% Overshoot = %f\n%% Error = ...
    %f',k_p,k_d,k_i,t2-t1,100*os/pi,100*error/pi)
```

Note Again, code can be placed in the middle of your report, or at the end in an Appendix. Code boxes should have a Figure number, too.

A.4 Plot Example

We purposely over-labeled the graph below (Figure 3). It demonstrates 1) using colors, 2) using text arrows, and 3) using different line styles. It also explicitly shows that the response meets the overshoot criteria while listing the calculated values of t_r and M_p (t_r is more difficult to show visually)

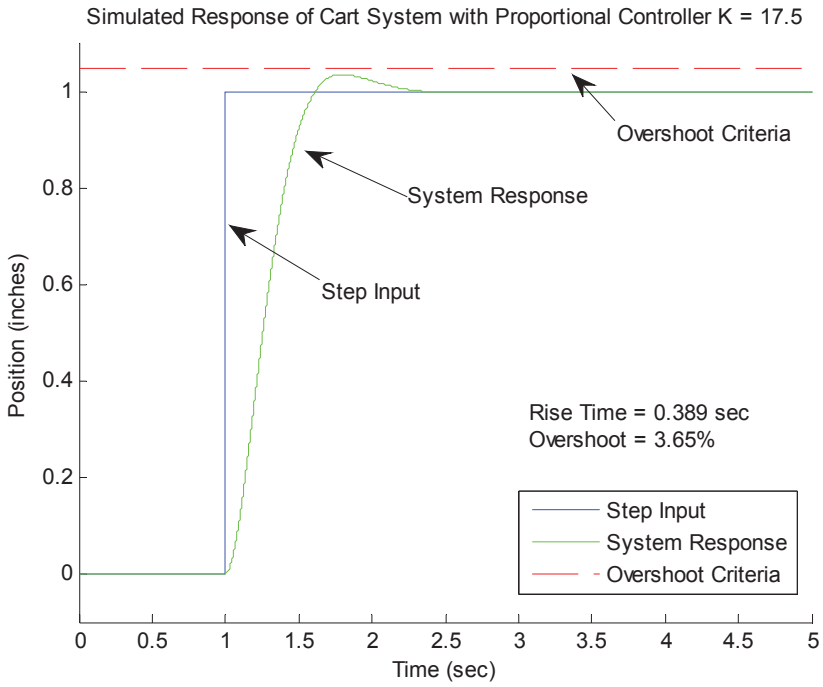

Figure 3: Purposely over-labeled plot of step response for chosen value $K = 17.5$ that meets the given desired criteria.

For those interested, the commands used to generate the plot are listed below. The text arrows are created directly on the plot. The listed rise time and overshoot were typed into a Text Box with linestyle set to "none". If you have questions about these MATLAB commands or functions, consult the MATLAB documentation or ask your classmates on piazza.

Listing 2: Useful MATLAB commands for creating plots

```
% allow multiple plots on same figure
hold on
% plot input (data col 1) in blue (default color) and dotted line
plot(t,data(:,1),':')
% plot response (data col 2) in green and solid line (default)
plot(t,data(:,2),'g')
% create overshoot line based on time variable (t)
oline = 1.05*ones(length(t),1);
% plot overshoot line in red and dashed line
plot(t,oline,'r--')
% adjust figure view for better visibility
axis([0 5 -0.1 1.1])
```

```
% label graph (legend, axis labels, title)
legend('Step Input','System Response','Overshoot Criteria')
xlabel('Time (sec)')
ylabel('Position (inches)')
title('Simulated Response of Cart System with Proportional Controller K = 17.5')
% Create text box
tbox = annotation(?textbox?,[0.6 0.35 0.28 0.1]);
set(tbox,?String?,sprintf(?Rise time = %1.3f sec\nOvershoot = %1.3f%%?,tr,os), ...
    ?LineStyle?,?none?)
```