Dyson School of Design Engineering

Imperial College London

DE2 Electronics 2

Lab Experiment 3: System Characterization & Transfer Function

(webpage: http://www.ee.ic.ac.uk/pcheung/teaching/DE2_EE/)

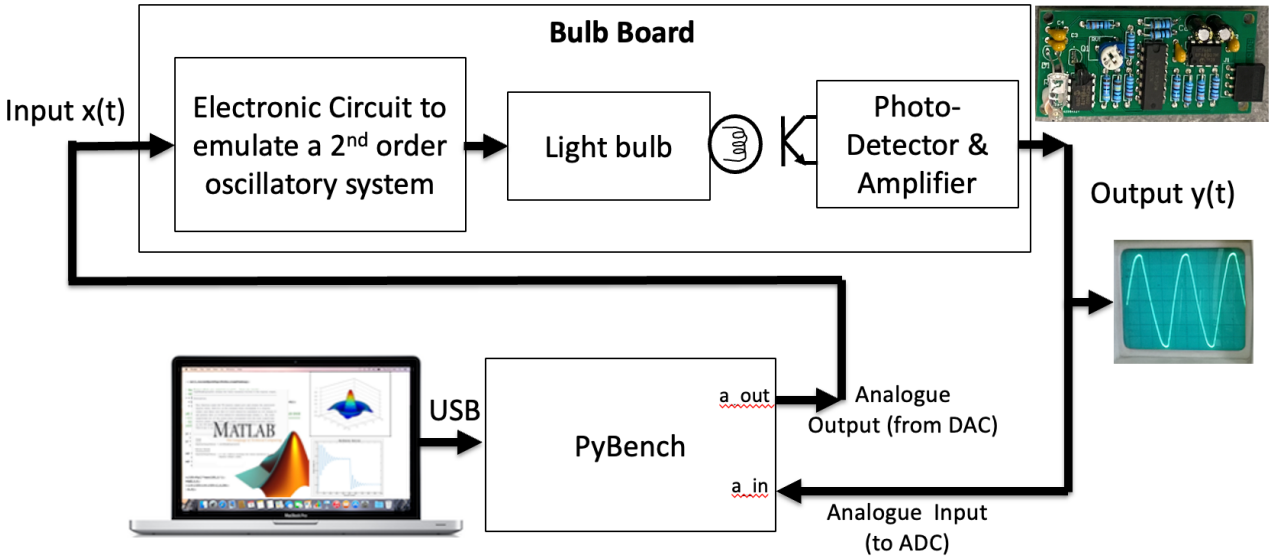
Objectives

By the end of this experiment, you should have achieved the following:

- Obtain the input-output relationship of a system at zero frequency (DC).
- Obtain the frequency response of a system at different frequencies.
- Obtain the step response of a system.
- Understand the meaning of transfer function of a system in the s-domain.
- Use Matlab to predict frequency response of a system from its transfer function through simulation only.
- Compare the measured and the theoretical behaviour of a system.
- Understand the transient behaviour of a system.

The Experimental System

In this experiment, you will connect the PyBench board to the **Bulb Board**, which is a special PCB to emulate (meaning 'mimic' or 'pretend to be') a **second-order system**. The following figure shows the experimental setup.



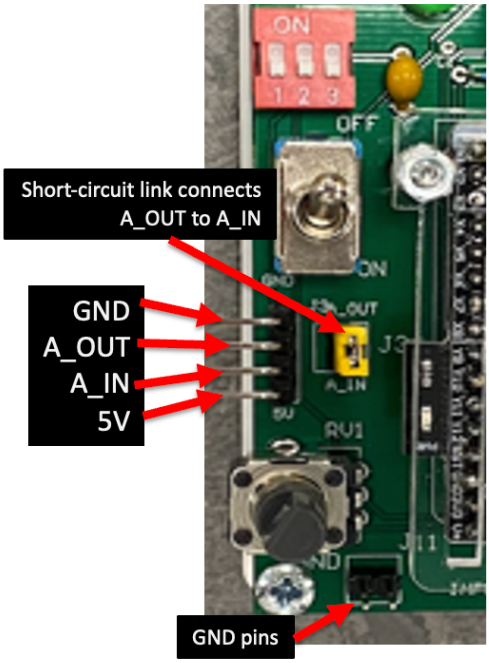
On the Bulb Board is an electronic circuit that behaves like a second-order system that is highly oscillatory. It is driven by an input signal x(t), which is produced at the DAC of PyBench board. The output of this circuit drives an incandescent light bulb. The intensity of the light emitted from the bulb is measured by a photo-transistor. The photo-transistor produces an electrical current that is approximately proportional to the light intensity. This current is amplified by an op-amp to produce the system output y(t).

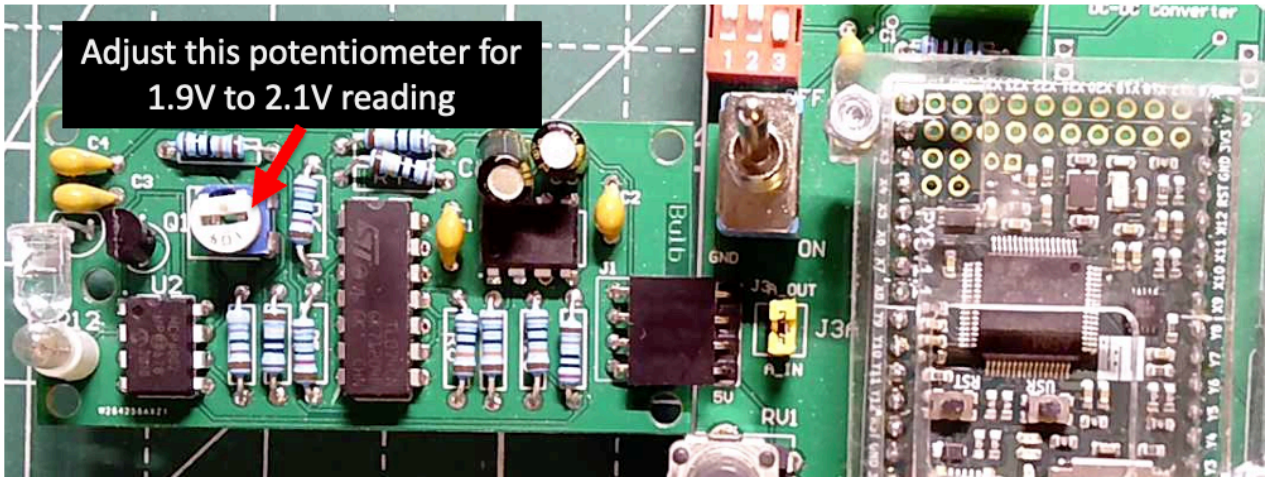
You do not need to know exactly how the Bulb Board works. However, for those who are interested, the full schematic diagram is available on the course webpage.

Check that Bulb Board works

Throughout this experiment, you will be driving and measuring signals to and from the Bulb Board using PyBench under the control of Matlab. Before you start this lab, do the following preliminary steps:

- **Take out the short-circuit header link** (yellow in picture) between A_OUT and A_IN as shown here. Put it back on the two GND pins (next to the potentiometer) so that you don't lose them.
- Plug the Bulb board into the 4-ways right angle header on the left side of PyBench as shown in the diagram below.
- Put PyBench into setting 101 (5). This outputs a voltage of 1.5V on the light bulb. Adjust the potentiometer so that the photodiode and amplifier shows a **measured voltage of 1.9V to 2.1V**. (Not 1.5V as instructed on the OLED display.)
- Put PyBench setting back to setting 111 (7) and press the RESET button. We are now in a position to control the bulb board via the ADC and DAC on Pybench using Matlab.





Task 1: DC Characteristic of the Bulb Board

Purpose: To obtain the input-output relationship of the Bulb Board for DC (zero frequency) voltage only.

- Open Matlab on your PC or MacBook, and enter the following Matlab code in the command window (comments not required):

```
clear all
ports = serialportlist;
pb = PyBench(ports(end));  % create a PyBench object
pb.dc(1.5);      % output 1.5V to Bulb Box to x_dc
pause(1);        % delay for 1 sec
pb.get_one()     % read y_dc
```

- This code snippet sends out a 1.5V DC on the DAC, which drives the Bulb Board. It then waits for 1 second for the bulb to settle to steady-state condition. It then takes one sample of the photo-transistor to read back the intensity of the bulb using the ADC. You should get around 2.0V as the light intensity reading. You should also see the light bulb brightly lid.
- Now try to output 0V and take one output measurement with **pb.get_one( )**. The reading should be close to zero and the bulb should be OFF.
- By changing the DC voltage sent to the bulb, find out: a) the minimum voltage (x_dc_L) applied to the bulb for it to light up; b) the voltage (x_dc_H) beyond which the bulb will not get brighter.
- Set the x_dc between these two limits, and measure the light intensity y_dc. Roughly sketch the **y_dc** vs **x_dc** on a graph for a few values of x_d.
- Doing measurement manually like this is tedious. It is much better to write a Matlab program **lab3task1.m** that performs the characterization automatically. Since we are only interested in the **steady-state response** (at DC or zero frequency) of the system to the input drive **x_dc**, you need to put in a delay of at least 0.5 sec after changing **x_dc**. Wait for the output **y_dc** to settle to its final value before measuring. Otherwise, you will be measuring the **dynamic** behaviour of the system.
- Enter the following Matlab code WITHOUT COMMENTS and carefully examine the results.
- This is the **DC characteristic** of the Bulb Board system. Make sure you understand the reason for the shape of this graph. Explain why this system is **non-linear**.
- What would you expect the **output** vs **input** curve would be like if the system were linear?
- Over what region of input voltage x that the system may be approximately linear?

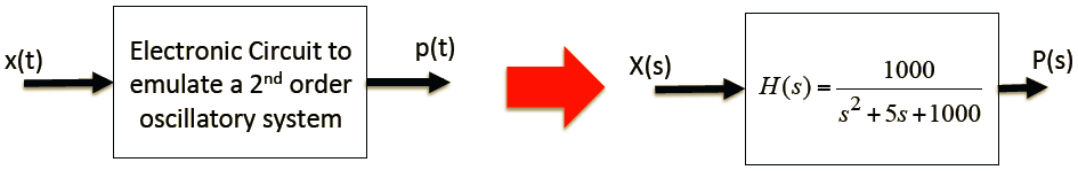
```
% Lab 3 - Task 1 input-output steady-state DC behaviour
ports = serialportlist;     % find all serial ports
pb = PyBench(ports(end));   % create a PyBench object

%  measure the steady-state DC output vs input
pb.samp_freq = 100;         % take sample at 100Hz
NSTEPS=25;                  % take 25 steps in input
input = zeros(NSTEPS,1);    % reserve 25 x 1 vector memory
output = zeros(NSTEPS,1);   % .. faster then appending one-by-one
tic                         % start timer
disp('Sweeping drive voltage for DC steady-state characteristics');
for i = [1:NSTEPS]
    v = (i-1)*2.5/NSTEPS;   % voltage to drive input
    input(i) = v;
    pb.dc(v);               % write to DC to drive bulb
    pause(0.5);             % wait for output to settle
    data = pb.get_block(10);    % make 10 measurements
    output(i) = mean(data);     %  ... to average
end
pb.dc(0.0);                 % turn bulb off
toc                         % stop timer
figure
plot(input,output)          % plot output vs input
xlabel('Input (V)');        % Label the axes
ylabel('Output (V)');
title('DC input-output transfer function');
fclose(instrfind());        % close the port
```

Task 2: Frequency Response of the Bulb Board system – Theoretical only

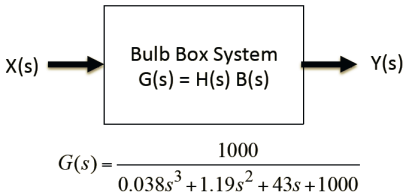
Purpose:	**Use the transfer function equations of the bulb board system to predict the frequency response $H(j\omega)$ of the bulb board using Matlab.**

The Bulb Board system can be modeled mathematically as a linear system (ignoring the non-linear DC characteristics for now) as two things connected in series. Firstly, the second order electronic circuit, which is known to have a transfer function as shown:



The light bulb, detector/amplifier can be modeled (ignoring the non-linearity) as a first order system as:

The overall system can therefore be modeled in the Laplace s-domain as G(s) = H(s) B(s):



We are interested in discovering the system's gain G for sinusoidal signals x(t) at different signal frequencies. This is called the **frequency response** of the system. That is, we want to find:

$$\mathrm{G(j}\omega) = \frac{\text{amplitude of y(t)}}{\text{amplitude of x(t)}} \quad \text{at different } \omega.$$

We can derive the theoretical frequency response G(jω) by evaluating G(s) with s = jω. The following Matlab program will perform the calculation and plot the theoretical frequency response:

```
1   %  Lab 3 Task 2 - Plot theoretical freq. response of Bulb Board
2   f = (0:0.1:20);
3   D = [0.038 1.19 43 1000];     % specify denominator coefficients
4   s = 1i*2*pi*f;                % s = jw (1i is sqrt(-1))
5   G = 1000./abs(polyval(D,s)); % polynomial evaluation
6   Gdb = 20*log10(G);            % Gain in dB
7   figure;
8   plot(f,Gdb);
9   xlabel('Frequency (Hz)');
10  ylabel('Gain (dB)');
11  title('Frequency Response - Theoretical');
```

Line 2: specifies the vector f, from 0Hz to 20Hz in steps of 0.1Hz. This range is selected because it is known that system's oscillatory behaviour is well below 20Hz.

Line 3: creates a vector specifying the coefficients of the denominator polynomial of G(s), in the order of highest power of s to lowest power.

Line 4: s is the vector specifying the frequency of interest. Note that to get the frequency response, we evaluate G(s) at $s = j\omega$, where $\omega = 2\pi f$. Matlab uses 1i to specify imaginary number instead of j.

Line 5: This calculates G at the different frequencies. "1000./" performs **element-by-element** divide. Look up Matlab help page for **polyval(.)** function.

The rest of the Matlab code is self-explanatory.

- Enter the Matlab script, run the script and comment on the results.
- Manually evaluate $G(s)|_{s=j\omega}$ at frequencies 0 Hz, 5 Hz and 20 Hz, and check that these agree with the Matlab prediction. (Remember that $\omega = 2\pi f$.)

Task 3: Measure the Frequency Response of the Bulb Board system using PyBench

Purpose:	**Measure the real frequency response of the Bulb Board using small amplitude sinewaves at different frequencies.**

In this exercise, you will measure the frequency response of the actual Bulb Board using PyBench and compare your measurements with the theoretical prediction.

Since the system is non-linear, we will only operate the system for input x(t) between 1.55V to 1.45V. Is this a good approximation to a linear system? Why? (Hint: examine the DC characteristics in Task 1.)

Enter and test the following Matlab code shown below **as lab3task3.m** to produce a sinusoidal signal at 5Hz and capture 300 samples of the output signal y(t) at a sampling frequency fs of 100Hz, and plot the signal. Finally, it calculates the gain of the system $|G(j\omega)|$ at 5Hz (i.e $\omega = 2\pi*5$) and convert this to dB (G_dB). Make sure that you understand how this works.

Explain the measured value of G and compare that to the theoretical model. Why do you think there is a difference between the measured and theoretic gain at 5Hz?

```
1   %  Lab 3 Task 3
2   % Measure system gain at frequency f_sig
3   %
4   clear all
5   ports = serialportlist;
6   pb = PyBench(ports(end));  % create a PyBench object
7   % Generate a sine wave at sig_freq Hz
8   max_x = 1.55;
9   min_x = 1.45;
10  f_sig = 5.0;
11  pb=pb.set_sig_freq(f_sig);
12  pb=pb.set_max_v(max_x);
13  pb=pb.set_min_v(min_x);
14  pb.sine();
15  pause(2)
16  % Capture output y(t)
17  pb=pb.set_samp_freq(100);  % sample at 100Hz
18  N = 300;     % no of samples to capture
19  y = pb.get_block(N);
20  % plot signal
21  plot(y);
22  xlabel('Sample no.');
23  ylabel('Output voltage');
24  title('Bulb Box output (V)');
25  % Compute Gain
26  x_pk2pk = max_x - min_x;
27  y_pk2pk = max(y) - min(y);
28  G = y_pk2pk/x_pk2pk
29  G_dB = 20*log10(y_pk2pk/x_pk2pk)
```

Repeat the measurement at f = 1Hz, 3Hz, 4Hz, 5Hz, 7Hz and 9Hz. (You can do this easily by changing line 10.)

Sketch the measured gain $G(j\omega)$ vs frequency to get an idea of the shape as compared to that found theoretically in Task 2. (You may also mark the measured points on the theoretical frequency response plot.)

Now create a your own Matlab program **lab3task3a.m** which performs automatic measurement of $G(j\omega)$ for frequencies between 1Hz and 20Hz in steps of 1Hz (i.e. 19 measurements). Plot the frequency response (i.e. Gain in dB vs Frequency). Compare the measured frequency response with the theoretical plot. Comment on the results.

(My solution to lab3task3a.m is included in the Appendix for those running out of time.)

Task 4: Measure the Step Response of the Bulb Board

Purpose:	**Measure the transient step response of the Bulb Board.**

Frequency response is useful only to measure the "*steady state*" condition of a system. That's why we used the **pause(2)** statement in the previous exercise to wait for the "transient" to die down. In this exercise, you will be examining the transient behaviour of the Bulb Board by driving it with a step function.

In this task, we will drive the bulb with a rising step, then a falling step, between 0.7V and 1.2V, and capture the transient behaviour of the system.

Enter the following Matlab code, and obtain the step response of the Bulb Board. Explain the results.

```
% Lab 3 Task 4 - Transient behaviour of Bulb Box
clear all
clf
ports = serialportlist;
pb = PyBench(ports(end));  % create a PyBench object
% set various parameters
fs = 50;
pb = pb.set_samp_freq(fs);
x_min = 0.7;
x_max = 1.2;
N = 150;   % no of data samples
% Capture step response
pb.dc(x_min);               % initial bulb value
pause(1);                   % wait 1 sec for it to settle
pb.dc(x_max)                % rising step to bulb
rise = pb.get_block(N);     % capture N samples
pb.dc(x_min)                % falling step to bulb
fall = pb.get_block(N);     % capture another N samples
data = [rise' fall'];       % combine rise with fall
% plotting
figure(1)
clf
plot(data)
xlabel('Sample number');
ylabel('Output (V)');
title('Step Response - Experimental');
fclose(instrfind());        % close the port
```

APPENDIX

Solution to Task 1

```
% Lab 3 Task 1 - input-output DC Characteristic
clear all
ports = serialportlist;     % find all serial port
pb = PyBench(ports(end));   % create a PyBench object with last port
pb.samp_freq = 100;
NSTEPS=30;                  % do 30 measurement points
tic                         % start clock
for i = [1:NSTEPS]
    v = (i-1)*2.5/NSTEPS;   % x_dc goes from 0 to 2.5V
    x_dc(i) = v;
    pb.dc(v);
    pause(0.5);
    data = pb.get_block(10);  % make 10 measurements
    y_dc(i) = mean(data);      % then average
end
pb.dc(0.0);
toc                         % report elapsed time
figure(1)
clf
plot(x_dc,y_dc)
xlabel('Input to bulb x\_dc (V)');
ylabel('Output brightness y\_dc(V)');
title('DC Characteristic');
```

Solution to Task 3a

```
% Lab 3  Task 3a - frequency response measurement
clear all
ports = serialportlist;
pb = PyBench(ports(end));  % create a PyBench object
pb=pb.set_samp_freq(100);
MIN_F = 1;
MAX_F = 20;
NSTEPS = 19;
fstep = (MAX_F - MIN_F)/NSTEPS;
x_max = 1.55;
x_min = 1.45;
pb=pb.set_max_v(x_max);
pb=pb.set_min_v(x_min);
tic
for i = [1:NSTEPS]
    f = (i-1)*fstep + MIN_F
    x(i) = f;
    pb = pb.set_sig_freq(f);
    pb.sine();
    pause(2.0)
    data = pb.get_block(200);
    y(i) = 20*log10((max(data)-min(data))/(x_max-x_min));
end
toc
plot(x,y,'o')
hold on
plot(x,y)
xlabel('Frequency (Hz)');
ylabel('Gain (dB)');
title('Measured Frequency Response');
fclose(instrfind());        % close the port
```