

1P2a MATLAB

1.1 Introduction

MATLAB stands for Matrix Laboratories. It is a tool that provides a graphical interface for numerical and symbolic computation along with a number of data analysis, simulation and acquisition functions.

Installing MATLAB on your personal computer:

1. Register an account on: <http://www.mathworks.com> using your Oxford email address

* Important: Under “How will you use The Mathworks Software”, select “Academic use”. Otherwise it is not going to work.

2. Download MATLAB from “My Account” and Install. Use your Mathworks account to activate.

1.2 Start using MATLAB

MATLAB has four windows: current folder, workspace, command windows and command history.

In this practical the symbol `>>` is used when we expect you to enter the script that follows.

A good way to learn MATLAB commands and functions is to use help, try the following:

```
>> help exp
```

```
exp     Exponential.
```

```
      exp(X) is the exponential of the elements of X, e to the X.
```

```
      For complex Z=X+i*Y, exp(Z) = exp(X)*(COS(Y)+i*SIN(Y)).
```

Try to use help to learn the following functions: and, or, log, for, linspace

1.3 Variables

Variables in MATLAB are classified as:

- Scalars (with a single numerical value)

```
>> x=6
```

```
x =
```

1P2 Introduction to MATLAB and LabVIEW

or

```
>> materials=100;
```

* Note the difference with and without the semicolon (;)

- Vectors (with more than one value organised in one dimension)

```
>> r = [14,23,45,67] % row vector
```

```
>> r = [14;23;45;67] % column vector
```

```
>> abc = [1:10]
```

```
>> abc = [1:10]'
```

```
>> x = [0:10:100] % create a vector from 0 to 100 at increment of 10
```

How about? >> length(r)

- Matrix

```
>> m = [56,24,33;45,95,60;71,82,93]
```

* Use help elmat to learn matrix functions

```
>> size(m) % returns the number of row and array.
```

1.4 Useful Tips

- Give variables a meaningful names.
- To leave a comment insert a '%' before the comment... annotate your work!
- F9 Executes whatever is highlighted.
- F5 Saves and runs the whole script.
- The "up arrow" recalls the previous commands.

1.5 Indexing

```
>> abc(4) % returns the 4th number of array
```

```
>> abc(1:4) % returns 1st to 4th numbers
```

```
>> m(1,3) % 1st row, 3rd column
```

```
>> m(:,3) % all elements in 3rd column
```

1P2 Introduction to MATLAB and LabVIEW

How about? `>> m(:, :)`

`>> i=find(m>50)` % return the indices of the numbers that fulfill the equation

Now the indices are returned, use `>> m(i)` can find the numbers

1.6 Mathematical functions

Learn elementary maths functions `>> help elfun`

`>> x = 45*pi/180`

`>> t = sin(x)*cos(x)`

Now consider vector equation,

$$T_i = \sin(x_i) \cos(x_i), i = 1, \dots, \pi$$

`>> x=linspace(0,pi,5)` %create a vector x with 5 values uniformly spaced from 0 to pi

`>> T=sin(x).*cos(x)`

* Note that the dot `.` is used to multiply an element of a vector by a corresponding element of a vector with the same length. Otherwise, it will be calculated through linear algebra.

$$\begin{bmatrix} B_{11} & B_{12} & \dots & B_{1n} \\ B_{21} & B_{22} & \dots & B_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & \dots & \dots & B_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} B_{11}x_1 + B_{12}x_2 + \dots + B_{1n}x_n \\ B_{21}x_1 + B_{22}x_2 + \dots + B_{2n}x_n \\ \vdots \\ B_{m1}x_1 + B_{m2}x_2 + \dots + B_{mn}x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

`>> B=[11,23,45;45,34,56;98,77,12]`

`>> x=[1;2;3]`

`>> y=B*x`

* The number of columns in B must match with the number of rows in x.

2 MATLAB programming

2.1 Script M files

An M files contains a list of MATLAB commands. You can create a script file under +NEW in the top control panel.

1P2 Introduction to MATLAB and LabVIEW

The following programme shows you how to plot a graph in MATLAB using a script file.

```
clear          % It is a good practice to have clear in the first line.
x = linspace (0, 2*pi, 360);
y = sin (x);
plot (x*180/pi,y);
title('Sine Wave');
xlabel('degree'); ylabel('sin(x)');
grid;
axis ([0 360 -1 1]);
```

* use % to make a comment, the text will be regarded as a comment and displayed in green.

2.2 More graph commands

plot (y) % plot y against the element number;

plot (x1,y1) % plot y1 against x1 and y2 against x2;

plot(x,y,'r+') % use red plus sign

plot (x1,y1,'r+',x2,y2,'go')

* Symbols: . (points), o (circles), x (X marks), + (plus), * (star), - (solid line, default), : (dotted line),

-. (Dash dot line), -- (dashed line)

y (yellow), m (magenta), c (cyan), red, green, blue (default), white, black

Hold on % Hold the plot on the screen, so that the graphs after are plotted over this graph

Hold off % Release the plot on the screen

Exercise 1 plot both sin(x) and cos(x) on the same graph, sin(x) in red '+' sign and cos(x) in green dash line.

2.3 Flow control

- For loop: repeat the statement in the for loop for a number of times;

```
clear
for i = 1:10
    x (i) = i;
    y (i) = i^2;
end
plot (x,y);
```

1P2 Introduction to MATLAB and LabVIEW

- While loop: repeat the statement in the while loop while the condition is true;

```
clear
i=1;
while (i<=10)
    x(i)=i;
    y(i) = i^2;
    i=i+1;
end
plot (x,y);
```

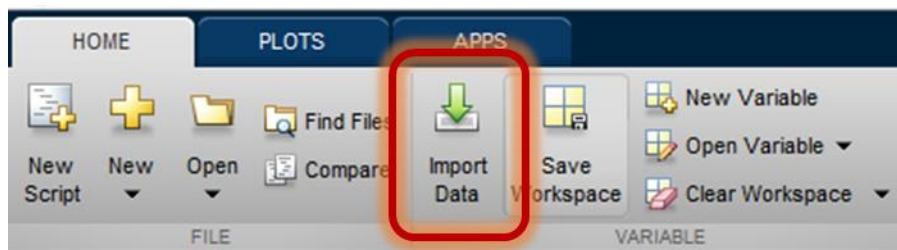
- If: if the condition is true, the statement is executed.

```
clear
x = input('Enter a value for x?') % The program will ask for a number
if (x<100)
    disp('OK');
else
    disp('x is invalid')
end
```

3 Handling Larger Data & Images

3.1 Importing data

Matlab can import data using the interactive menu bar:



By dragging items into the workspace, or using the command line; e.g. :

```
I = imread(example_image.jpg');
```

Different data types require slightly different import procedures. Try for yourself; search the help (or online) for how to import a text file, or to open only one slice from a TIFF stack.

3.2 Manipulating image data

Image data typically contains three channels, red, green and blue (often written RGB).

Greyscale images (intensity only) may also have three channels but often just need one that describes the grey-level.

Try for yourself; go online and download an example image. Check the image dimensions after you have imported the data. Try out:

- Crop the data to a smaller sub-region,
- Mirror the image from left to right,
- Swap two of the colour channels to change the image appearance,
- Create a synthetic image with two different inputs used for different colour channels.

3.3 Writing images to use outside MatLab

After working on image data these can be written back out as the same, or a different, file format.

- Try writing your new image as a TIFF or a PNG.

4 Curve Fitting

Matlab has built in functions for data-fitting in one or more dimensions. Below is an example for a simple 1D case.

- Create a vector of linearly spaced values from 1 to 4π . You can choose how many values but fewer than 1000 is recommended.
- Calculate the sine of these values, **BUT** with a deliberately added phase and a chosen amplitude and a constant.
- Add random noise to your new data. Choose between Gaussian noise, Poisson noise or some other.
- Next open the interactive curve fitting toolbox by typing “`cftool`” into the command line.
- Try to fit your data to a model and compare the results with your input parameters.
- What happens if you reduce the number of initial linearly spaced datapoints?
- What happens if you change the strength of the noise added.
- What happens if you add a constant to the data before adding the noise? What about adding it after?

5 Exercises

- 1) Create a vector with even numbers from 31 to 75;
- 2) $X = [2 \ 5 \ 1 \ 6]$
 - a. Add 16 to each element
 - b. Add 3 to the odd index element (Hint: use “1:2:end” to index all odd element)
 - c. Compute the square root of each element
- 3) Create two column vectors $x = [3 \ 2 \ 6 \ 8]'$ and $y = [4 \ 1 \ 3 \ 5]'$
 - a. Add the sum of elements in x to y (Hint: use the sum function)
 - b. Raise each element of x to the power specified by the corresponding element in y
 - c. Divide each element of y by the corresponding element in x
- 4) Calculate the following values. (Hint: use help to find out the meaning of round, floor and ceil)
 - a. $2 + \text{round}(6 / 9 + 3 * 2) / 2$
 - b. $2 + \text{floor}(6 / 9 + 3 * 2) / 2$
 - c. $2 + \text{ceil}(6 / 9 + 3 * 2) / 2$
- 5) Given the array $A = [2 \ 4 \ 1; 6 \ 7 \ 2; 3 \ 5 \ 9]$,
 - a. assign the first row of A to a vector called ax
 - b. assign the last 2 rows of A to an array called y (Hint: last row = function ‘end’)
- 6) Still the same array A , explain the following commands:
 - a. A'
 - b. $A(:, [1 \ 3])$
 - c. $A([2 \ 3], [3 \ 1])$
 - d. $A(:)$
 - e. $\text{flipud}(A)$
 - f. $\text{fliplr}(A)$

1P2 Introduction to MATLAB and LabVIEW

g. `[A A(:,end)]`

h. `A(1:3,:)`

i. `[A;A(1:2, :)]`

j. `sum(A)`

k. `sum(A')`

l. `sum(A,2)`

m. `[[A;sum(A)] , [sum(A,2) ; sum(A(:))]]`

n. `length (A)`

o. `size (A)`

7) Given the vector $x = [1 \ 8 \ 3 \ 9 \ 0 \ 1]$, create a short set of commands that will

a. Add up the values of the elements

b. Computes the running sum (for element j , the running sum is the sum of the elements from 1 to j , inclusive. Check with **cumsum**.)

c. computes the sine of the given x -values (should be a vector)

6 Homework

Complete the following, your answers should include your annotated code, any results and a brief discussion of your approach.

To solve the first problem, you are advised to read some textbooks about solar cells in the library or using online sources such as <http://www.pvcdrom.com> or Wikipedia. Further training courses and information can be found here <http://www.eng.ox.ac.uk/~labejp/Seminar/>.

- 1) To do this problem you will need to go back to CANVAS (<https://canvas.ox.ac.uk/courses/18064>) where you can download the current-voltage response of a Gallium Arsenide solar cell under 100 mW/cm^2 simulated sunlight. Write a MATLAB script that can calculate the following parameters of the solar cell:
 - a. Short circuit current density: J_{sc}
 - b. Open circuit voltage: V_{oc}
 - c. Fill factor: FF
 - d. Power conversion efficiency

- 2) Write a script to find the maximum number in an array. You are **NOT** allowed to use 'max' function. (Hint: use flow control).

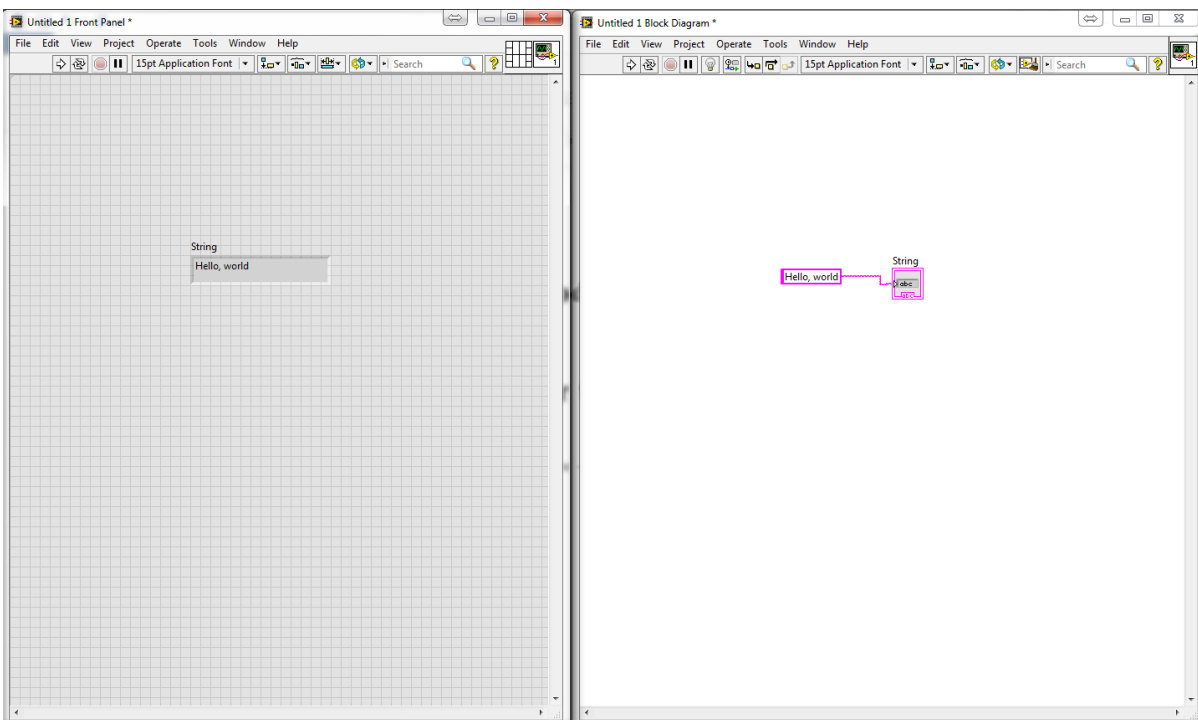
1P2b

LabVIEW

1 Introduction

First open NI labview and create a new project then a blank VI (virtual instrument).

Labview has two windows: 1. **Front panel** (User interface for input and output), 2. Graphic code (called **block diagram**).



2 Hello, world! program

1) In the **front panel** create a “**string indicator**” by dragging and dropping the function found in the control panel. (**Right click>>string & path**) (You can also find these functions using the ‘**search**’ box)

1P2 Introduction to MATLAB and LabVIEW

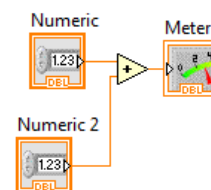
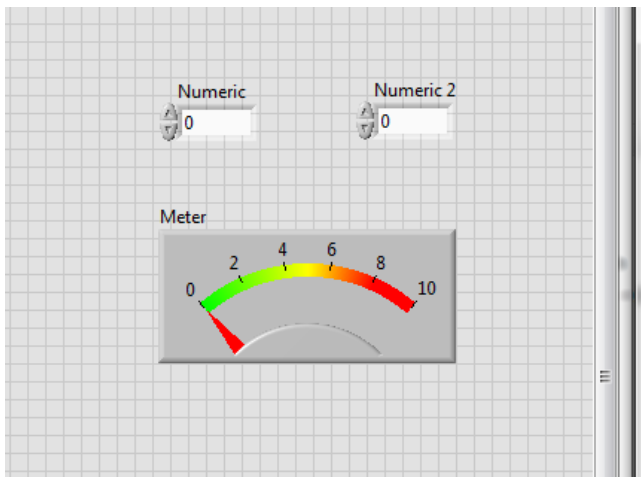
2) Right click in the **block diagram** and create a “**string constant**”.

3) Wire them together. Type ‘Hello, World!’ in the ‘**string constant**’ on the block diagram, and press run and observe the front panel. Your program should look like the picture above.


For more information about components placed on the front panel or block diagram, press ‘help’ in the menu bar then click ‘show context help’. This provides more information when the cursor hovers over a component.

Labview has different data types: string (text), numeric (numbers), boolean (true or false), array (an array of number) etc. For numerical types of data, you can perform some basic mathematical functions.

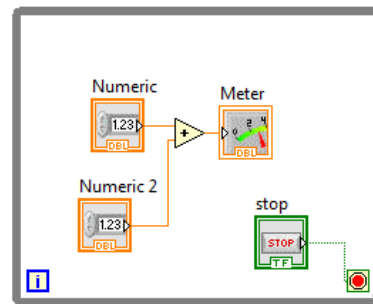
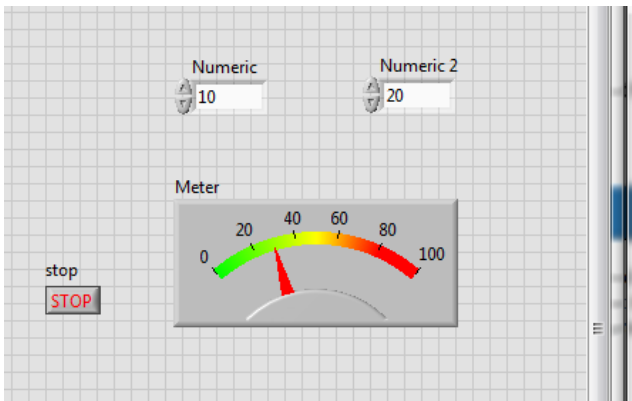
Exercise 1). Create a programme as shown in the figure below. The meter (in **numeric>>meter**) shows the sum of the value you enter in the two ‘**numeric control (numeric>>numeric control)**’ boxes. You can change the scale of the meter by **right clicking on the meter -> properties -> scale**.



3 While loop

A program inside a while loop stays running until a condition is met. The condition is set by wiring a true or false control to the . The 'i' at the left bottom corner counts how many iterations the loop has been executed.

Exercise 2). Create a while loop (**Structure>>while loop**) to the above program so that it will keep running until a 'STOP' (**Boolean>>STOP**) button is pressed.

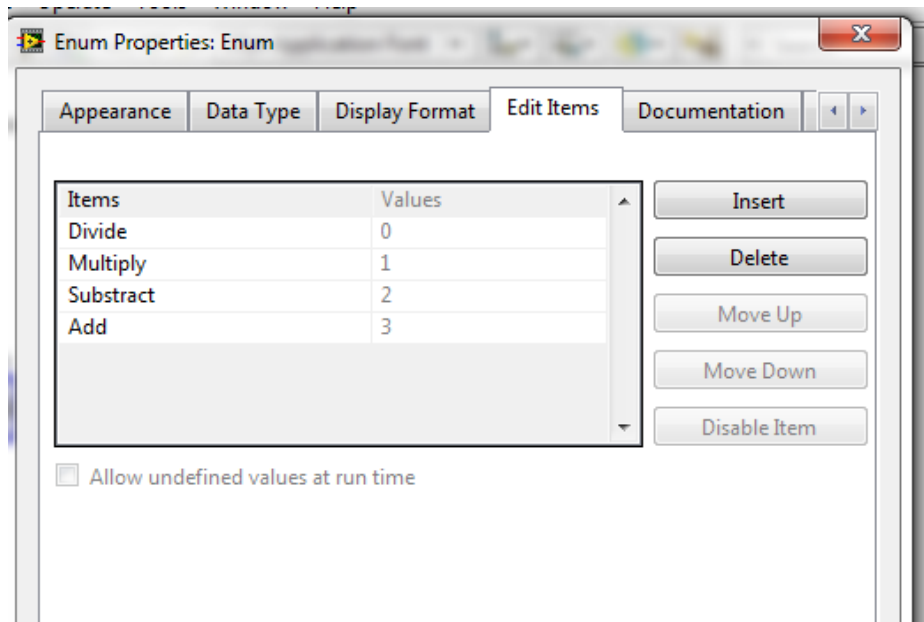


4 Case structure

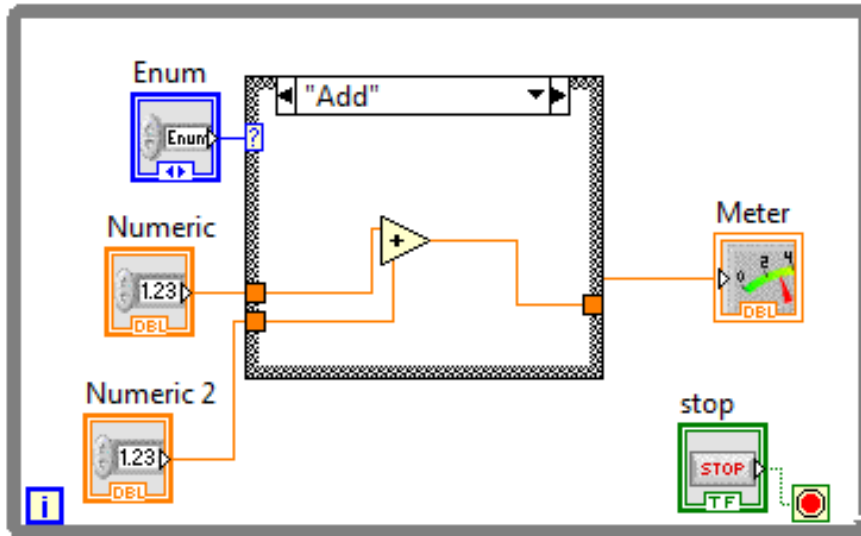
A **case structure** can have more than one subdiagram, i.e. more than one 'case'. A selector (can be numeric, string or boolean) is wired to the terminal to determine which case is to be executed.

Exercise 3). Use the **case structure** to allow user to select add, subtract, multiply and divide functions.

- 1) Delete all the wires and the **'add' operator** you have in the above programme.
- 2) Right click in the **Front Panel**, **'ring & enum>>enum'**. In the **properties** of the enum box, go to **'Edit items'** and insert 4 items -add, subtract, multiply, divide.



- 3) Create a **'case structure (structure>>case structure)'** in your **Block Diagram** and wire the **'enum'** to the **'?'**.
- 4) Right click on the border of the **case structure** and click **'Add Case for Every Value'**. You will see that your case structure now has 4 different cases and you can use the arrow to select one.
- 5) Drag and drop the corresponding operators to each case and wire them with the **numeric controls** and the **meter**. Your program should look like the diagram below.
- 6) Run the program and see what happens when you change the case in the enum.



5 Data acquisition

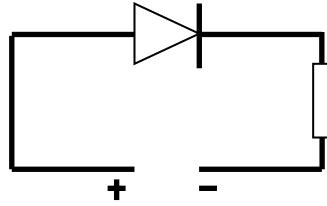
We use a National Instruments 6008 data acquisition device (DAQ) in this class to perform data acquisition. It has 8 analog inputs (AI), 2 analog outputs (AO, up to 5 V) and 12 digital I/O.

Exercise 3) Testing your DAQ. First, connect the DAQ to your computer, a green LED next to the USB port will flash.

1) Fire up **NI MAX (Measurement and Automation Explorer)**, go to 'Device and Interfaces', you should see a **NI-6008 device** in the list. Click 'Self-Test', it should say 'The self-test completed successfully'.

2) Connect the LED and resistor as shown in the below circuit diagram (The longer leg is the anode – positive end). Connect the analogue output (AO0) on the DAQ to positive and ground (GND) to negative. We have provided you with a 10 ohm (Brown), a 1 kohm (Blue) and a 1 Mohm (Blue with a yellow line) resistors. In this case, you will need the 1 kohm resistor to limit the current passing through the LED. Otherwise, your LED might blow up.

1P2 Introduction to MATLAB and LabVIEW



3) To light up the LED, go to 'Test Panels' >> 'Analog Output' and select your channel. Set **5V** as your output voltage and press 'Update', your LED should light up. Change the output voltage to **0V** to turn the LED off.

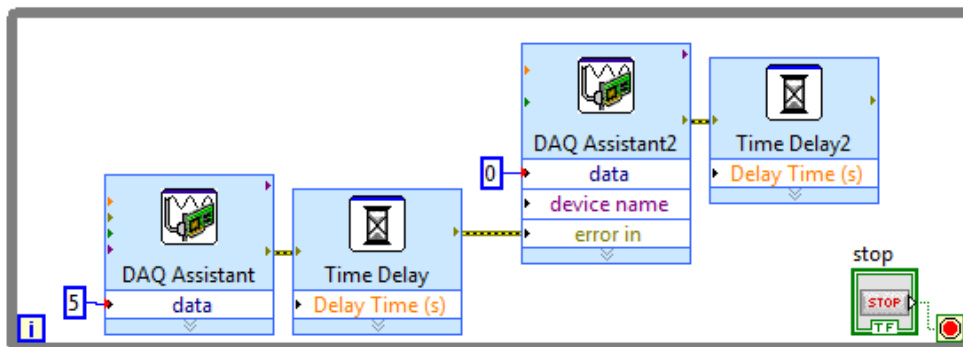
Exercise 4) Blink a LED for a number of times.

1) To control the DAQ in Labview, right click in the **Block diagram** >> **Measurement I/O** >> **NI DAQmx** >> **DAQ Assist**. Select **Generate Signal** >> **Analog Output** >> **Voltage**. Choose your channel (**AO0**) and finish. In the configuration page, keep everything as default and then press **OK**.

2) A **DAQ Assistant** will appear in the Block Diagram. Wire a **numeric constant** to 'Data' to control the voltage output of the DAQ. Set this to 5V.

3) Create another '**DAQ Assistant**' and set to 0V to turn the LED off.

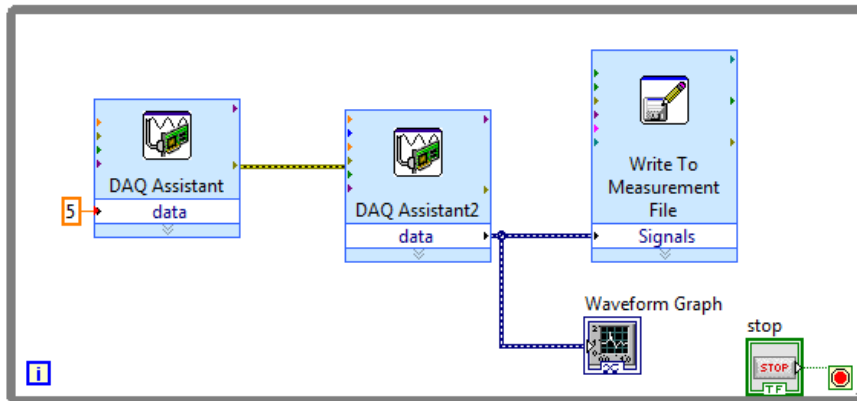
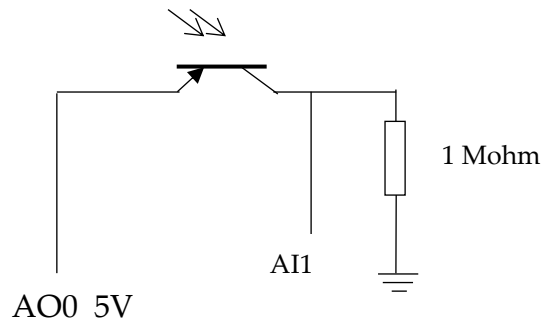
4) Add a time delay after each DAQ Assistant by using **Timing** >> **Time Delay** (1 second should be enough). You can control the flow of the diagram by wiring 'Error Out' to 'Error In'. Place all of them in a **while loop**. You should have a diagram that looks like this:



1P2 Introduction to MATLAB and LabVIEW

X Value **'One column only'**. You can do some other settings as you wish. Now wire them together as the figure below.

4) Shine a light onto the photodiode to observe the change. Open the LVM file you have saved using Notepad.



Exercise 6) Measure diode characteristic

We apply a voltage across a diode and measure its current to obtain a diode I-V characteristic curve.

1P2 Introduction to MATLAB and LabVIEW

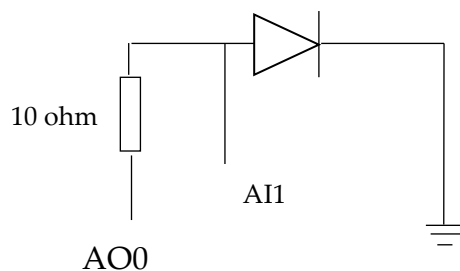
The circuit diagram is shown below. In this exercise, a voltage sweeps from 0 to 5V and is generated from the analog output by using a **for loop**. The voltage across the diode can be measured from the analog input, V_{ai} . The current can be calculated using $I=(V_{ao}-V_{ai})/R$.

Now, create a labview file as shown below. First, you will need to create two **numeric controls**. One is to set the maximum voltage and one is to set the number of steps for the voltage scan. You can change the name of the **numeric controls** to 'Max V' and 'No of steps'. Wire 'no of steps' to the 'N' to control how many times it will loop. Now the voltage output in each step will be 'Max V'/'No of step' * i , i indicates which iteration you are at and i starts from 0. Now wire this number to the DAQ Assistant for analog output. Drop a DAQAssistant for analog input to measure the voltage across the diode.

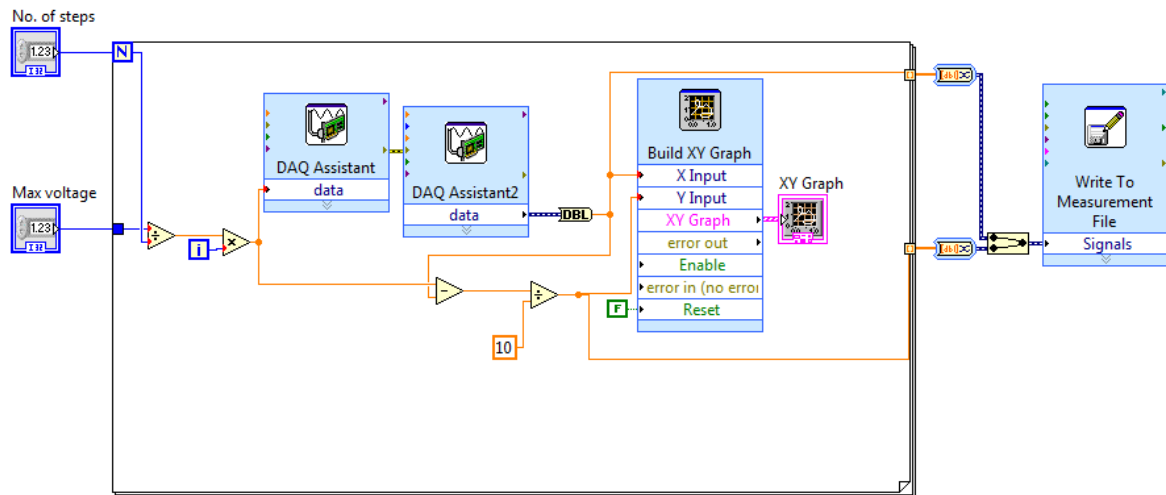
To create a XY graph, go to '**Graph>>Ex XY graph**'. Remember that the data output from the DAQ Assistant will need to convert to double precision (**DBL**). Go to '**numeric>>conversion>>To Double precision float**'.

Once the voltage is measured, you can calculate the current in the circuit.

Wire the current to **Y Input** and voltage to **X Input** in the '**Build XY graph**'. Make sure the **Reset** on the **Build XY graph** box is false otherwise the graph will reset itself in each loop. You can save the measurement data using the '**Write to measurement file**'. This needs to be outside the **for loop**. When you wire more than 1 signal to the '**Write to Measurement File**', the two current and voltage signals will be bundled automatically into one.



1P2 Introduction to MATLAB and LabVIEW



You can run your VI. The Max V generated by the DAQ is 5 V. Use at least 15 steps to obtain a smooth diode characteristic curve.