Data Handling in the Ph77 *Electronics Track*

This document describes how one might use Mathematica to analyze data in the Electronics Track in Ph77, starting with the Test Equipment handout. If you have any additional (or better) Mathematica commands to suggest, please let us know. Note that ChatGPT4o (or something similar) can be *very* useful for generating Mathematica code. Also, you are welcome to analyze your data using other software tools, such as Origin, Python, etc.
(To make text like this in a Mathematica notebook: select cell, then Format>Style>Text)

```
In[ ]:= SetDirectory[NotebookDirectory[]];
        (* sets directory to same as Mathematica notebook *)
        (* Must save the .nb file first *)
```

Reading a . csv file from a Keysight EDUX1052A oscilloscope

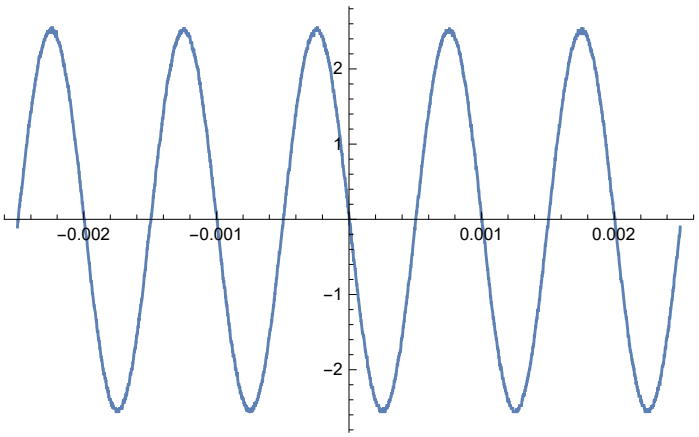
```
In[ ]:= aaa = Import["scope_273.csv", "Data", "HeaderLines" → 2];
        (* Look at the .csv file directly (it contains text) to see the data format *)
        (* The above skips 2 header lines in .csv file *)
        (* We use ; here to suppress printing the entire matrix *)
```

```
In[ ]:= Dimensions[aaa]
        (* to see size of list *)(* the Keysight 'scope usually saves about 2000 points *)
Out[ ]= {2000, 2}
```

```
In[ ]:= aaa[[1 ;; 10, All]]  (* print to check contents *)
Out[ ]= {{-0.0025, -0.105402}, {-0.0024975, -0.025}, {-0.002495, -0.025}, {-0.0024925, 0.015201},
         {-0.00249, 0.095603}, {-0.0024875, 0.135804}, {-0.002485, 0.176005},
         {-0.0024825, 0.176005}, {-0.00248, 0.256407}, {-0.0024775, 0.256407}}
```

```
In[ ]:= dataplot = ListLinePlot[aaa, PlotRange → All]
        (* Note can drag corner of plot to make larger or smaller *)
```

Out[ ]=



Interacting with the plot

Use Graphics > Drawing Tools to access, or type Ctrl - D (or Ctrl - T on Mac)
Select plot, then select the “dashed-+” tool to read positions on plot
This is useful, for example, for measuring the spacing between the peaks
In theory, these should be uniformly spaced. But there are nonlinearities in the laser frequency sweep.

Nonlinear Model Fit

This fits the above data to a sinusoidal function

In[ ]:=
```
fit = NonlinearModelFit[aaa, a - b * Sin[2 * Pi * fff * t], {{a, 0}, {b, 2.4}, {fff, 900}}, t]
 (* Estimate initial values estimated from the graph *)
```

Out[ ]=
```
FittedModel[ -0.0178437 - 2.52961 Sin[6283.37 t] ]
```

In[ ]:=
```
fit["ParameterTable"]
```

General: Exp[−7199.38] is too small to represent as a normalized machine number; precision may be lost.

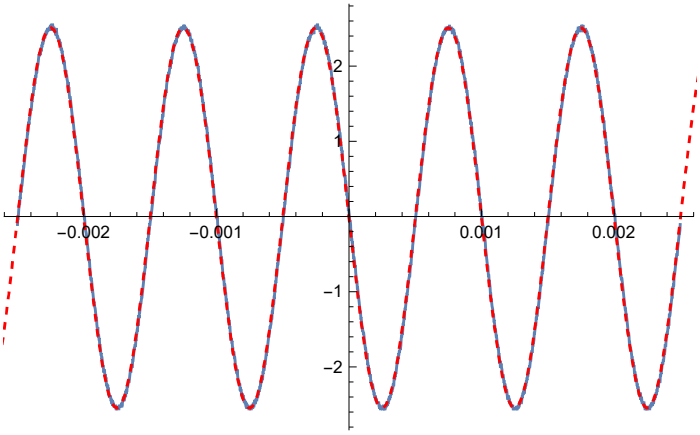
General: Exp[−11608.] is too small to represent as a normalized machine number; precision may be lost.

Out[ ]=

	Estimate	Standard Error	t-Statistic	P-Value
a	−0.0178437	0.00108905	−16.3845	1.04316×10^{-56}
b	2.52961	0.00154251	1639.93	0.
fff	1000.03	0.0670307	14 919.	0.

In[ ]:=
```
fitplot = Plot[fit[x], {x, -0.03, 0.03}, PlotStyle → {Red, Dashed}];
Show[dataplot, fitplot] (* overlay data and fit to verify that the fit is working *)
```

Out[ ]=



Plotting data points with a theory curve

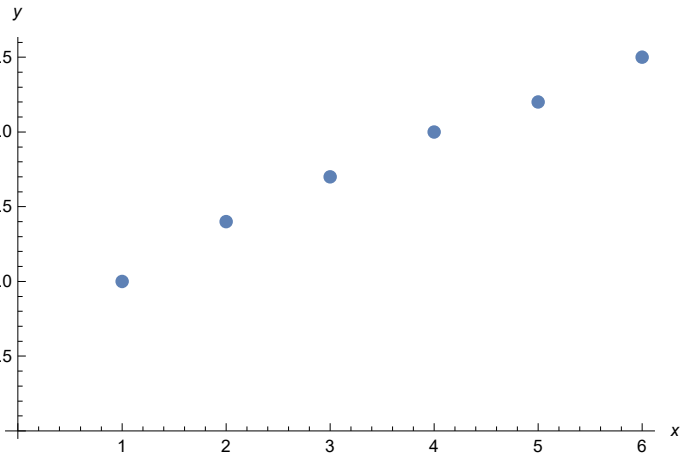
Same basic idea as above, except entering a small number of data points by hand

In[ ]:=
```
ccc = {{1, 1}, {2, 1.4}, {3, 1.7}, {4, 2}, {5, 2.2}, {6, 2.5}}
```

Out[ ]=
```
{{1, 1}, {2, 1.4}, {3, 1.7}, {4, 2}, {5, 2.2}, {6, 2.5}}
```

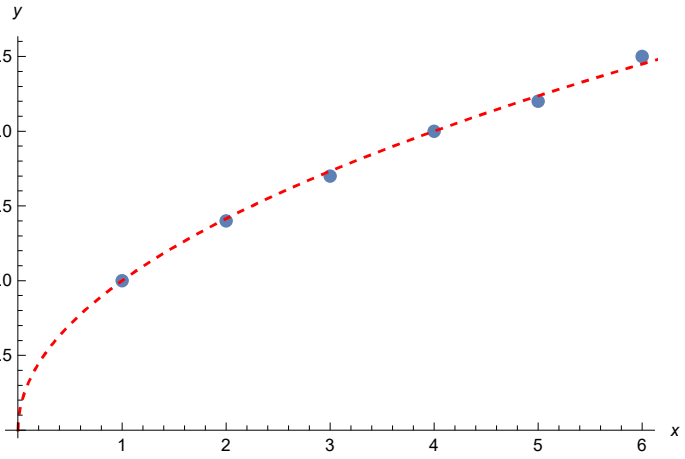
In[ ]:=
```
dataplot2 = ListPlot[ccc, AxesLabel → {x, y}, PlotStyle → PointSize[Large]]
```

Out[ ]=



In[ ]:=
```
fitplot2 = Plot[Sqrt[x], {x, 0, 8}, PlotStyle → {Red, Dashed}];
Show[dataplot2, fitplot2]
```

Out[ ]=



Plotting low-pass filter data

Enter data by hand:

In[ ]:=
```
data0 = {{freq1, y1}, {freq2, y2}}
```

Out[ ]=
```
{{freq1, y1}, {freq2, y2}}
```

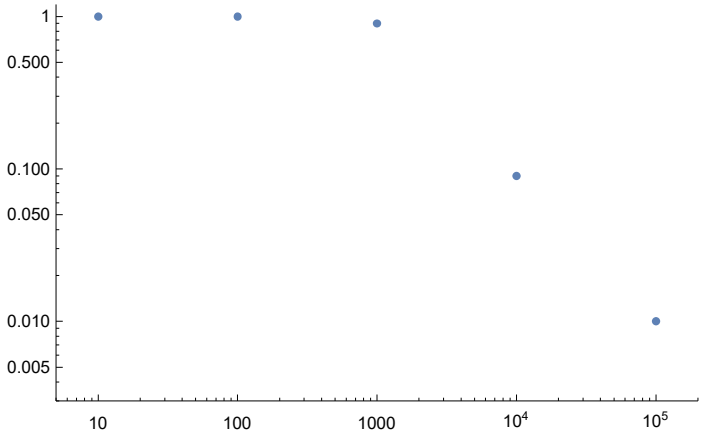
For Example:

In[ ]:=
```
data1 = {{10, 1}, {100, 1}, {1000, 0.9}, {10 000, 0.09}, {100 000, 0.01}}
```

Out[ ]=
```
{{10, 1}, {100, 1}, {1000, 0.9}, {10 000, 0.09}, {100 000, 0.01}}
```

In[ ]:=
```
dataplot = ListLogLogPlot[data1, PlotRange → {{5, 200 000}, {0.003, 1.2}}]
```

Out[ ]=



In[ ]:=
```
theoryplot6db = LogLogPlot[1 / Sqrt[1 + (x / 1000) ^ 2], {x, 5, 5 000 000}, PlotRange → All];
theoryplot12db = LogLogPlot[1 / (1 + (x / 1000) ^ 2),
   {x, 5, 5 000 000}, PlotRange → All, PlotStyle → {Dashed, Red}];
```

In[ ]:=
```
Show[dataplot, theoryplot6db, theoryplot12db]
```

Out[ ]=