Phys 238 : Importing Data into *Mathematica*

Getting Started

Mathematica remembers definitions, even if you delete them from the notebook. This can sometimes be confusing, so it's often a good idea to simply clear everything and start anew. This is especially true if you use the Evaluation -> Evaluate Notebook menu item.

In[81]:=
```
Clear["Global`*"] (* Clear all variable and function definitions *)
```

Entering Data

Consider the data for static torque from the torsional oscillator experiment. This data consists of {mass, angle} pairs. There are multiple ways to enter the data.

The data is a list of data points. All lists are made of elements in curly braces { } separated by commas. Here, each data point is itself a list, consisting of two numbers (in curly braces) separated by a comma. To see what the data looks like, display it with TableForm[ ].

There are many ways to enter the data:

Typing the data directly:

In[82]:=
```
rawdata = {
   {0, 2.98},
   {50, 2.79},
   {100, 2.59},
   {150, 2.38},
   {200, 2.17},
   {250, 1.96},
   {300, 1.66},
   {350, 1.41},
   {400, 1.21},
   {0, 2.98},
   {-50, 3.20},
   {-100, 3.40},
   {-150, 3.60},
   {-200, 3.81},
   {-250, 4.06},
   {-300, 4.32},
   {-350, 4.56},
   {-400, 4.74},
   {0, 3.00}
  }
```

Out[82]=
```
{{0, 2.98}, {50, 2.79}, {100, 2.59}, {150, 2.38},
 {200, 2.17}, {250, 1.96}, {300, 1.66}, {350, 1.41}, {400, 1.21},
 {0, 2.98}, {-50, 3.2}, {-100, 3.4}, {-150, 3.6}, {-200, 3.81},
 {-250, 4.06}, {-300, 4.32}, {-350, 4.56}, {-400, 4.74}, {0, 3.}}
```

In[83]:=
```
TableForm[rawdata]
```

Out[83]//TableForm=

0	2.98
50	2.79
100	2.59
150	2.38
200	2.17
250	1.96
300	1.66
350	1.41
400	1.21
0	2.98
−50	3.2
−100	3.4
−150	3.6
−200	3.81
−250	4.06
−300	4.32
−350	4.56
−400	4.74
0	3.

Typing the data using the Classroom Assistant: $\begin{pmatrix} \square & \square \\ \square & \square \end{pmatrix}$

In[84]:=

From the Palettes -> Classroom Assistant palette, look for the matrix item: $\begin{pmatrix} \square & \square \\ \square & \square \end{pmatrix}$. (Depending on the version of *Mathematica*, it might be in the “Typesetting” menu, or in the “Advanced” menu. (Use Ctrl-Enter to add a new row.)

In[85]:=
```
rawdata = {{0, 2.98}, {50, 2.79}, {100, 2.59}, {150, 2.38}, {200, 2.17}, {250, 1.96}, {300, 1.66}, {350, 1.41}, {400, 1.21}, {0, 2.98}, {-50, 3.20}, {-100, 3.40}, {-150, 3.60}, {-200, 3.81}, {-250, 4.06}, {-300, 4.32}, {-350, 4.56}, {-400, 4.74}, {0, 3.00}}
```

Out[85]=

$$\begin{pmatrix} 0 & 2.98 \\ 50 & 2.79 \\ 100 & 2.59 \\ 150 & 2.38 \\ 200 & 2.17 \\ 250 & 1.96 \\ 300 & 1.66 \\ 350 & 1.41 \\ 400 & 1.21 \\ 0 & 2.98 \\ -50 & 3.20 \\ -100 & 3.40 \\ -150 & 3.60 \\ -200 & 3.81 \\ -250 & 4.06 \\ -300 & 4.32 \\ -350 & 4.56 \\ -400 & 4.74 \\ 0 & 3.00 \end{pmatrix}$$

Reading from a file:

To read the data from a text file in the same directory as your notebook, first tell *Mathematica* what directory to use. I like to keep the data file in the same directory as the notebook.

In[86]:=
```
SetDirectory[NotebookDirectory[]]
```

Out[86]=

```
/Users/doughera/notes/www/public_html/courses/phys238-2025/Mathematica
```

Typically, files might have header information (names of columns, dates, etc.) There might also be other comments. Sometimes the columns are separated by spaces, tabls, or commas. It is usually helpful to see what the raw data file looks like.

In[87]:=
```
FilePrint["TO-static-20250212.csv"]
```

```
"Mass (g)", "Raw Angle", "Voltage"
0, 2.98, -0.0456
50, 2.79, -0.4083
100, 2.59, -0.7733
150, 2.38, -1.172
200, 2.17, -1.556
250, 1.96, -1.958
300, 1.66, -2.483
350, 1.41, -2.726
400, 1.21, -2.566
0, 2.98, -0.0606
-50, 3.20, 0.3049
-100, 3.40, 0.6583
-150, 3.60, 1.028
-200, 3.81, 1.387
-250, 4.06, 1.812
-300, 4.32, 2.237
-350, 4.56, 2.502
-400, 4.74, 2.369
0, 3.00, -0.04502
```

This particular file has 3 columns, separated by commas. There is a single header line at the top giving the column titles.

Importing as a CSV (Comma Separated Values) file

If you know the import format, you can specify it explicitly:

In[88]:=
```
tableImport = Import["TO-static-20250212.csv", "CSV"]
```

Out[88]=
```
{{Mass (g), Raw Angle, Voltage}, {0, 2.98, -0.0456}, {50, 2.79, -0.4083},
 {100, 2.59, -0.7733}, {150, 2.38, -1.172}, {200, 2.17, -1.556},
 {250, 1.96, -1.958}, {300, 1.66, -2.483}, {350, 1.41, -2.726},
 {400, 1.21, -2.566}, {0, 2.98, -0.0606}, {-50, 3.2, 0.3049}, {-100, 3.4, 0.6583},
 {-150, 3.6, 1.028}, {-200, 3.81, 1.387}, {-250, 4.06, 1.812},
 {-300, 4.32, 2.237}, {-350, 4.56, 2.502}, {-400, 4.74, 2.369}, {0, 3., -0.04502}}
```

In[89]:=
```
TableForm[tableImport]
```

Out[89]//TableForm=

Mass (g)	Raw Angle	Voltage
0	2.98	-0.0456
50	2.79	-0.4083
100	2.59	-0.7733
150	2.38	-1.172
200	2.17	-1.556
250	1.96	-1.958
300	1.66	-2.483
350	1.41	-2.726
400	1.21	-2.566
0	2.98	-0.0606
-50	3.2	0.3049
-100	3.4	0.6583
-150	3.6	1.028
-200	3.81	1.387
-250	4.06	1.812
-300	4.32	2.237
-350	4.56	2.502
-400	4.74	2.369
0	3.	-0.04502

In[90]:=
```
Dimensions[tableImport] (* It has 20 rows and 3 columns *)
```

Out[90]=
```
{20, 3}
```

Importing as a Dataset

Another useful way is to Import the data into what Mathematica calls a “Dataset”, which is a structured object with it’s own special query language. Here’s how it works for the torsional oscillator data. This data file has a third column, “Voltage”.

In[91]:=
```
rawdata = Import["TO-static-20250212.csv", "Dataset"]
```

Out[91]=

Mass (g)	Raw Angle	Voltage
0	2.98	−0.0456
50	2.79	−0.4083
100	2.59	−0.7733
150	2.38	−1.172
200	2.17	−1.556
250	1.96	−1.958
300	1.66	−2.483
350	1.41	−2.726
400	1.21	−2.566
0	2.98	−0.0606
−50	3.2	0.3049
−100	3.4	0.6583
−150	3.6	1.028
−200	3.81	1.387
−250	4.06	1.812
−300	4.32	2.237
−350	4.56	2.502
−400	4.74	2.369
0	3.0	−0.04502

If you just want a single column (say the angles then this format is very convenient. The syntax asks for 'All' rows, and the "Raw Angle" column.

In[92]:=
```
angles = rawdata[All, "Raw Angle"]
```

Out[92]=

2.98	2.79	2.59	2.38	2.17	1.96
1.66	1.41	1.21	2.98	3.2	3.4
3.6	3.81	4.06	4.32	4.56	4.74
3.0					

This is still a data set, but it's a very simple one. Many functions, such as Mean[ ] and StandardDeviation[ ] can handle it just fine. You can convert it to a simple list with the Normal[ ] command.

In[93]:=
```
Mean[angles]
```
Out[93]=
```
2.99053
```

In[94]:=
```
Normal[angles]
```
Out[94]=
```
{2.98, 2.79, 2.59, 2.38, 2.17, 1.96, 1.66, 1.41,
 1.21, 2.98, 3.2, 3.4, 3.6, 3.81, 4.06, 4.32, 4.56, 4.74, 3.}
```

Dataset Complication:

If you want a two-dimensional array, say of {mass, angle} pairs, you can extract it simply with

In[95]:=
```
anglevsMass = rawdata[All, {"Mass (g)", "Raw Angle"}]
```
Out[95]=

Mass (g)	Raw Angle
0	2.98
50	2.79
100	2.59
150	2.38
200	2.17
250	1.96
300	1.66
350	1.41
400	1.21
0	2.98
−50	3.2
−100	3.4
−150	3.6
−200	3.81
−250	4.06
−300	4.32
−350	4.56
−400	4.74
0	3.0

But the resulting two-dimensional array is not a simple object. It is an Associative Array. More importantly for us, it is not in a form that LinearModelFit can handle. You will need to manually build up a suitable two-dimensional array. For this reason, the Dataset method is sometimes more trouble than

it's worth.

In[96]:=
```
Normal[anglevsMass]
```

Out[96]=
```
{⟨|Mass (g) → 0, Raw Angle → 2.98|⟩,
 ⟨|Mass (g) → 50, Raw Angle → 2.79|⟩, ⟨|Mass (g) → 100, Raw Angle → 2.59|⟩,
 ⟨|Mass (g) → 150, Raw Angle → 2.38|⟩, ⟨|Mass (g) → 200, Raw Angle → 2.17|⟩,
 ⟨|Mass (g) → 250, Raw Angle → 1.96|⟩, ⟨|Mass (g) → 300, Raw Angle → 1.66|⟩,
 ⟨|Mass (g) → 350, Raw Angle → 1.41|⟩, ⟨|Mass (g) → 400, Raw Angle → 1.21|⟩,
 ⟨|Mass (g) → 0, Raw Angle → 2.98|⟩, ⟨|Mass (g) → -50, Raw Angle → 3.2|⟩,
 ⟨|Mass (g) → -100, Raw Angle → 3.4|⟩, ⟨|Mass (g) → -150, Raw Angle → 3.6|⟩,
 ⟨|Mass (g) → -200, Raw Angle → 3.81|⟩, ⟨|Mass (g) → -250, Raw Angle → 4.06|⟩,
 ⟨|Mass (g) → -300, Raw Angle → 4.32|⟩, ⟨|Mass (g) → -350, Raw Angle → 4.56|⟩,
 ⟨|Mass (g) → -400, Raw Angle → 4.74|⟩, ⟨|Mass (g) → 0, Raw Angle → 3.|⟩}
```

In[97]:=
```
LinearModelFit[anglevsMass, {1, m}, m]
```

LinearModelFit : The first argument is not a vector, matrix, or a list containing a design matrix and response vector.

Out[97]=

LinearModelFit[

Mass (g)	Raw Angle
0	2.98
50	2.79
100	2.59
150	2.38
200	2.17
250	1.96
300	1.66
350	1.41
400	1.21
0	2.98
−50	3.2
−100	3.4
−150	3.6
−200	3.81
−250	4.06
−300	4.32
−350	4.56
−400	4.74
0	3.0

, {1, m}, m]

Selecting and Calculating With Data

No matter how you import data, you will likely need to make some adjustments or calculations. Two very useful commands are Select[ ] and Table[ ]. For simplicity, assume you import the data as a CSV file.

In[98]:=
```
rawdata = Import["TO-static-20250212.csv", "CSV"];
TableForm[rawdata]
```

Out[99]//TableForm=

Mass (g)	Raw Angle	Voltage
0	2.98	-0.0456
50	2.79	-0.4083
100	2.59	-0.7733
150	2.38	-1.172
200	2.17	-1.556
250	1.96	-1.958
300	1.66	-2.483
350	1.41	-2.726
400	1.21	-2.566
0	2.98	-0.0606
-50	3.2	0.3049
-100	3.4	0.6583
-150	3.6	1.028
-200	3.81	1.387
-250	4.06	1.812
-300	4.32	2.237
-350	4.56	2.502
-400	4.74	2.369
0	3.	-0.04502

Looking at parts of an array:

The [[ ]] notation is used to look at an element of an array.

In[100]:=
```
rawdata[[2]] (* The second row *)
```

Out[100]=
```
{0, 2.98, -0.0456}
```

To look at the 3rd item on the second row, use the [[row, column]] notation

In[101]:=
```
rawdata[[2, 3]]
```

Out[101]=
```
-0.0456
```

The special name All can also be used:

In[102]:=
```
rawdata[[All, 3]] (* All rows, column 3 *)
```

Out[102]=
```
{Voltage, -0.0456, -0.4083, -0.7733, -1.172, -1.556, -1.958, -2.483, -2.726, -2.566,
 -0.0606, 0.3049, 0.6583, 1.028, 1.387, 1.812, 2.237, 2.502, 2.369, -0.04502}
```

In[103]:=
```
rawdata[[All, {2, 3}]] (* All rows, columns 2 and 3. *)
```
Out[103]=
```
{{Raw Angle, Voltage}, {2.98, -0.0456}, {2.79, -0.4083},
 {2.59, -0.7733}, {2.38, -1.172}, {2.17, -1.556}, {1.96, -1.958},
 {1.66, -2.483}, {1.41, -2.726}, {1.21, -2.566}, {2.98, -0.0606},
 {3.2, 0.3049}, {3.4, 0.6583}, {3.6, 1.028}, {3.81, 1.387}, {4.06, 1.812},
 {4.32, 2.237}, {4.56, 2.502}, {4.74, 2.369}, {3., -0.04502}}
```

In[104]:=
```
rawdata[[2, All]] (* Row 2, all columns *)
```
Out[104]=
```
{0, 2.98, -0.0456}
```

The Select Command:

The basic syntax for the Select command is Select[data, criteria] (see the online help for full details The Select command loops through the data and returns all points for which the criteria are true. Along the way, it sets the symbol '#' equal to each element of the data. Since # is itself a list (with 3 elements) you can look at the first, second, and third elements of each point with #[[1]], #[[2]], and #[[3]]. (You must include the '&' at the end.) You can combine tests with '&&' to mean logical "AND". You would use '||' to mean logical OR. Thus to select lines where the first, second, and third elements are all numbers, you could do

In[105]:=
```
data = Select[rawdata, NumberQ[#[[1]] && NumberQ[#[[2]] && NumberQ[#[[3]] &]
```
Out[105]=
```
{{0, 2.98, -0.0456}, {50, 2.79, -0.4083}, {100, 2.59, -0.7733},
 {150, 2.38, -1.172}, {200, 2.17, -1.556}, {250, 1.96, -1.958}, {300, 1.66, -2.483},
 {350, 1.41, -2.726}, {400, 1.21, -2.566}, {0, 2.98, -0.0606}, {-50, 3.2, 0.3049},
 {-100, 3.4, 0.6583}, {-150, 3.6, 1.028}, {-200, 3.81, 1.387}, {-250, 4.06, 1.812},
 {-300, 4.32, 2.237}, {-350, 4.56, 2.502}, {-400, 4.74, 2.369}, {0, 3., -0.04502}}
```

To select only points where the mass is between -200 and +200 (including the endpoints) you could do

In[106]:=
```
somedata = Select[data, -200 ≤ #[[1]] ≤ 200 &]
```
Out[106]=
```
{{0, 2.98, -0.0456}, {50, 2.79, -0.4083}, {100, 2.59, -0.7733},
 {150, 2.38, -1.172}, {200, 2.17, -1.556}, {0, 2.98, -0.0606}, {-50, 3.2, 0.3049},
 {-100, 3.4, 0.6583}, {-150, 3.6, 1.028}, {-200, 3.81, 1.387}, {0, 3., -0.04502}}
```

Alternatively, if you know you simply want to drop the first row, you can use the Drop[ ] command.

In[107]:=
```
Drop[rawdata, 1]
```
Out[107]=
```
{{0, 2.98, -0.0456}, {50, 2.79, -0.4083}, {100, 2.59, -0.7733},
 {150, 2.38, -1.172}, {200, 2.17, -1.556}, {250, 1.96, -1.958}, {300, 1.66, -2.483},
 {350, 1.41, -2.726}, {400, 1.21, -2.566}, {0, 2.98, -0.0606}, {-50, 3.2, 0.3049},
 {-100, 3.4, 0.6583}, {-150, 3.6, 1.028}, {-200, 3.81, 1.387}, {-250, 4.06, 1.812},
 {-300, 4.32, 2.237}, {-350, 4.56, 2.502}, {-400, 4.74, 2.369}, {0, 3., -0.04502}}
```

The Table Command:

Suppose you want to create a new table with {angle, mass pairs, and you also want to convert the masses from gram to kilogram. The Table[ ] function is a handy way to do that.

First, note that the general syntax to generate a counter 'i' that runs from 1 to the length of the data is

In[108]:=
```
Table[i, {i, 1, Length[data]}]
```
Out[108]=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19}
```

To create our desired array, instead of just printing out the counter, we want to create a list (in curly brackets) of {mass, angle} pairs. Since angle is the second column, you use data[[i, 2]] to get angle. Here I also convert grams to kilograms

In[109]:=
```
massvsAngle = Table[ {data[[i, 1]] / 1000., data[[i, 2]]}, {i, 1, Length[data]}]
```
Out[109]=
```
{{0., 2.98}, {0.05, 2.79}, {0.1, 2.59}, {0.15, 2.38},
 {0.2, 2.17}, {0.25, 1.96}, {0.3, 1.66}, {0.35, 1.41}, {0.4, 1.21},
 {0., 2.98}, {-0.05, 3.2}, {-0.1, 3.4}, {-0.15, 3.6}, {-0.2, 3.81},
 {-0.25, 4.06}, {-0.3, 4.32}, {-0.35, 4.56}, {-0.4, 4.74}, {0., 3.}}
```

In[110]:=

```
TableForm[massvsAngle] (* Check what it looks like *)
```

Out[110]//TableForm=

0.	2.98
0.05	2.79
0.1	2.59
0.15	2.38
0.2	2.17
0.25	1.96
0.3	1.66
0.35	1.41
0.4	1.21
0.	2.98
−0.05	3.2
−0.1	3.4
−0.15	3.6
−0.2	3.81
−0.25	4.06
−0.3	4.32
−0.35	4.56
−0.4	4.74
0.	3.