Ph3 Mathematica Homework: Week 3

Eric D. Black
California Institute of Technology
v1.0

1 Working with data files

In the last lab session you downloaded a data set from an oscilloscope. It's file name should be something like `getwfm.isf`, and if you open it in a text editor its first few lines should look something like this.

```
-5.0000e-03,0.42
-4.9990e-03,0.44
-4.9980e-03,0.46
-4.9970e-03,0.46
-4.9960e-03,0.44
-4.9950e-03,0.5
-4.9940e-03,0.52
-4.9930e-03,0.54
-4.9920e-03,0.54
-4.9910e-03,0.52
```

The first column contains the time values, starting at five milliseconds before the trigger and incrementing by one microsecond for each line. The second column contains values of the voltage. Each line is a separate data point with its x- and y-values separated by commas.

Whenever possible you should look at your raw data before trying to import it into any software program for the first time. There's no substitute for knowing what you're working with.

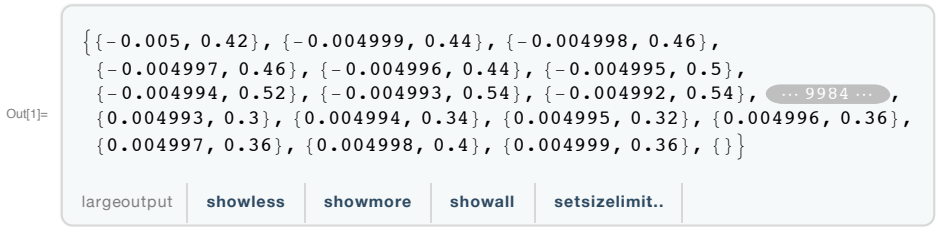
2 Importing data into Mathematica

Exercise 1: Import your data into Mathematica, and store it in the variable `data` using the command,

```
data = Import["~/Downloads/getwfm.isf"]
```

You may have to change the path name if your file is somewhere other than your `Downloads` folder, or you are using something other than a Mac.

The result should look something like this. Again, we need to inspect the data to make sure things are happening the way they are supposed to. Mathematica has recognized it as a list of pairs of values. It has correctly converted the "e" notation to numbers, as we expect, and it has even gotten the data in the right order.



This is a list of the form you expect, except for the last entry, an empty bracket. This is due to an extra (empty) line the 'scope adds to the data file. It is common to have extra lines at the beginning or end of a file when dealing with machine-generated files, and it provides an excellent example of why you should inspect your data and an excuse to exercise your list-manipulation skills. Usually these extra lines show up as headers at the beginning of a data file, but here it is an extra line at the end. You could remove it manually in a text editor before importing it, but it's easier to do it in Mathematica. Use the `Most` command to strip that last line out, and put the result into a new list.

```
data2 = Most[data]
```

(If you wanted to strip off the first line, to discard headers for example, you would use the `Rest` command. Its syntax is the same as `Most`.)

Exercise 2: If for some reason you can't (or don't want to) remember the `Most` and `Rest` commands, how would you strip the last line out using `Table`?

Exercise 3: Plot `data2` using the `ListPlot` command.

Exercise 4: For shorter lists, it is often useful to display the data as a conventional, two-column table. Generate a list of the first ten data points, and assign it to the variable `shortdata`. Execute the following command, and show its output.

```
TableForm[shortdata]
```

Exercise 5: Strip out the time (x) values of each data point, and generate a one-dimensional list of these values. Call this one-dimensional list `times`. The simplest way to do this is with the `Table` command.

```
times = Table[data2[[i, 1]], {i, 1, Length[data2]}]
```

Exercise 6: Plot this list using `ListPlot`. Note that, when it does not have any x values, `ListPlot` will use the index (number) of each data point as the x value.

Exercise 7: Lastly, see what the `Joined` option gets you by plotting your short data list the following two ways. (These ranges are specific to my data, yours may require different ones.)

```
ListPlot[shortdata,
 PlotRange -> { {-5.0*10^(-3), -4.99*10^(-3)}, {0, 1}},
 Frame -> True, FrameLabel -> {"number", "time"}]

ListPlot[shortdata,
 PlotRange -> { {-5.0*10^(-3), -4.99*10^(-3)}, {0, 1}},
 Frame -> True, FrameLabel -> {"number", "time"},
 Joined -> True]
```

3 When importing goes wrong

Sometimes Mathematica has trouble identifying the correct format in a file when it imports data. In this case you have to explicitly tell Mathematica what that format is by adding a second argument to the `Import` com-

mand. The data generated by the oscilloscopes is in a format called Comma-Separated Values, or `CSV` for short, and you could make that explicit to Mathematica by modifying your `Input` command to the following.

```
data = Import["~/Downloads/getwfm.isf", "CSV"]
```

In this case the `"CSV"` option is superfluous. Mathematica gets by just fine with or without it. Data in this form, however, does not fare so well.

```
X       Y
0.0     3.4039
0.5     3.9881
1.0     4.2004
1.5     5.0291
2.0     5.1880
2.5     5.3914
...
```

This data is in the form of a table, is separated by tabs instead of commas, and it has a header line.

Exercise 8: Go to `http://pmaweb.caltech.edu/~phy003/labs/Data1.txt`, download this data file, import it into Mathematica, and plot it. Try the import first without any data format specified. If your system recognizes it and correctly imports it as a list, great. If not, try importing it as a table (`"Table"`) or Tab-Separated Values (`"TSV"`). Deal with the headers as you need to. You can skip downloading it to your computer and just put the url in as the first argument in `Import`, if you want to get fancy. Mathematica can import things over the internet.

Exercise 9: Plot your data with error bars of ± 0.3 on each y-value.

4 Reference

4.1 Data formats

Mathematica supports hundreds of file formats for both importing and exporting. A few useful ones are,

1. `List` imports or exports a text file as a one-dimensional list, treating each line as an element.

2. `Table` imports or exports a text file as an N-dimensional list, where N is the number of columns in the text file.

3. `CSV` - Comma-Separated Values

4. `TSV` - Tab-Separated Values

5. `XLS` - Excel format.

6. `Binary` - Use for binary files. Data is separated into bytes.

7. `MAT` - Matlab format.

You can also import and export images, sounds, web sites, etc. Basically, if it can be stored on a computer, Mathematica can work with it. For more information see

`http://reference.wolfram.com/language/guide/ImportingAndExporting.html`

4.2 Miscellaneous list commands

1. `Table` - Generate a list of values. This is your basic, go-to command. Works like this.

```
In[8]:= squares = Table[i^2, {i, 0, 9}]
Out[8]= {0, 1, 4, 9, 16, 25, 36, 49, 64, 81}
```

2. Get individual elements of a list using double square brackets.

```
In[9]:= squares[[3]]
Out[9]= 4
```

The `Part` command does the same thing. You can both read from and write to individual locations in the list.

3. `Length` - Number of elements in a list.
4. `Dimensions` - Dimensions of a list, *e.g.* a 3×5 matrix.
5. `Most` - Get all but the last element of a list.
6. `Rest` - Get all but the first element of a list.
7. `First` - Get the first element of a list.
8. `Last` - Get the last element of a list.
9. `Take` - Get specific elements of a list. See the documentation for syntax.
10. `Drop` - Drop specific elements of a list.
11. `Delete` - Remove elements at specific locations. Very similar to `Drop`.
12. `Append` - Add an element to the end of a list.
13. `Prepend` - Add an element to the beginning of a list.
14. `Insert` - Add an element to a list at a specific location.
15. `Join` - Join two lists together.
16. `Sort` - Sort a list. Also see `SortBy`.
17. `Transpose` - Transpose a list like you would a matrix. It can be used to combine lists to make an N-dimensional list. This is actually a really useful one, and it works like this.

```
In[4]:= Transpose[{{x1, x2, x3}, {y1, y2, y3}, {z1, z2, z3}}]
Out[4]= {{x1, y1, z1}, {x2, y2, z2}, {x3, y3, z3}}
```

18. `Partition` - Divide a one-dimensional list into columns. Works like this.

    There are lots of other ways to use `Partition`. See the documentation for a more complete explanation.

```
In[5]:= Partition[{x1, y1, z1, x2, y2, z2, x3, y3, z3}, 3]
Out[5]= {{x1, y1, z1}, {x2, y2, z2}, {x3, y3, z3}}
```

19. `Flatten` - Remove all inner brackets and make one, big, one-dimensional list. Sort of the reverse of `Partition`.

20. `Reverse` - Reverse the order of a list.

21. `TableForm` - Display (or write) a list so that it looks like a table.

22. `MatrixForm` - Display (or write) a list as a matrix.