Course reader: *Getting data into and out of MATLAB*

- Copy/paste is an easy and fast way to get data into and out of MATLAB. However, this method is neither scalable nor automatizable.

- MATLAB-formatted files have a .mat extension and are imported using either of the following two options:

```
load filename.mat
data = load('filename.mat');
```

  The first method puts the contents of `filename.mat` directly into the MATLAB workspace while the second puts the contents into the structure `data`.

- Text files with all numeric data and equal columns in all rows can also be loaded using

```
data = dlmread('filename.txt');
```

- Microsoft Excel (.xlsx) data can be imported using `xlsread('filename.xls')`. However, the imported formatting can be different from the Excel file, so best to use the third output, which is a cell-matrix of the entire Excel sheet:

```
[~,~,celldat] = xlsread('filename.xlsx');
```

- The most flexible, but also most coding-intensive, way to import data is by using `fprintf`. You need to know about the format of the file to use this approach successfully. Using this code relies on control statements, which you will learn about in the next section. Below is some simple fprintf-based import code to give you a sense of what it looks like. You'll learn more about this in the video at the end of the *Control statements* section.

```
fid = fopen('filename.dat','r'); % pointer to file
data=[]; row=1; % initialize
while ~feof(fid) % until the file ends
    aline = fgetl(fid); % read in a new line
    data(row,:) = sscanf(aline,'%g'); % convert to numbers
    row=row+1;% update row counter
end
fclose(fid); % close file
```

- If you have files from specific software or hardware, don't assume that MATLAB "out of the box" will have import scripts. Best to contact the manufacturer to ask about how best to import their data into MATLAB.

- The easiest way to save data from MATLAB to disk is with the `save` command:

```
save % save the entire workspace to matlab.mat
save file.mat a b % save variables a,b
save('filename','a','b') % .mat is assumed
```

- You can save text-only data from a vector or matrix with `dlmwrite`:

```
dlmwrite('filename.txt',datvar,' ') % delimeter is a space
```

- `xlswrite` writes Excel-format data

- The most flexible way to write out data is `fprintf`. Again, you'll learn more about this method in the next section. But it's always interesting to see the kind of code that you will be able to use in the near future!

```
fid = fopen('data.dat','w'); % pointer to write to a file
for i=1:length(datavar) % loop over datapoints
    fprintf(fid,'%g\n',datavar(i)); % write the datapoint and newline (\n)
end
fclose(fid); % close file
```

- As with importing data, many hardware/software manufacturers provide custom MATLAB code to export data into their native format.

Exercises

1. Create a 2×3 matrix of random numbers in variable `mat`. Evaluate the following two lines of code, and compare the output files in a text editor. What is the difference and why did it happened?

```
dlmwrite('test1.txt', mat ,' ')
dlmwrite('test2.txt','mat',' ')
```

2. The function `csvread` imports comma separated values (CSV) files. Using a text editor or spreadsheet application, create a csv file containing all numbers. Use `help csvread` to import the data. Next, use `help csvwrite` to export the same data into a different file. Confirm that the contents of the two files are identical.

3. Which—if any—of the following lines of code contain an error? You'll need to create some variables in order to run this code in MATLAB.

```
save(namevar,'a','b')
save(namevar;'a';'b')
save(namevar,'a';'b')
```

4. These next lines definitely contain errors. Find and fix them in MATLAB.

```
save(dataoutname '.mat','a','b')
save([ dataoutname '.mat' ],varmat)
dlmwrite('file.txt')
xlswrite('file.xls','data4me')
data = load("myData.mat");
data = load({ 'myData' num2str(3) '.mat' });
```

5. Parentheses are sometimes aesthetic and sometimes not. The three lines of code below produce different results simply because the parentheses are in difference positions.

```
 0:.1:(2>1)
(0:.1:2)>1
(0:.1):2>1
```

Answers

1. The first line will write out the *contents* of the variable `mat`. For the second line, MATLAB thinks that you want to write out the *string* `mat`, and the file will contain `m a t`

2. .

```
datavar = csvread('datafile.csv');
csvwrite('datafile2.csv',datavar)
```

Note: `datavar` not `'datavar'` !

3. The first line is the only one that will run. For some functions, you can use either commas or semimcolons to separate input arguments, but not with the `save` function (note: this is MATLAB version-dependent).

4. Fixed.

```
save([ dataoutname '.mat' ],'a','b')
save([ dataoutname '.mat' ],'varmat')
dlmwrite('file.txt',datavar)
xlswrite('file.xls',data4me)
data = load('myData.mat');
data = load([ 'myData' num2str(3) '.mat' ]);
```

5. If you are confused about this one, test each subpart of the code in MATLAB before running the entire line.

- `0:.1:(2>1)` → `(2>1)` is interpreted as a logical (2¿1 is true), which is then converted into the number 1.
- `(0:.1:2)>1` → MATLAB builds an array from 0 to 2 in steps of .1, then tests whether each element is greater than 1, and then it prints out a list of Boolean false/true (0/1).
- `(0:.1):2>1` → MATLAB starts by making a list from 0 to .1 in steps of 1. That's just zero. Then it counts from 0 to 2 in steps of 1, and finally it tests whether each element in that list is greater than 1 and prints the result as Boolean 0/1.