Lab Tutorial 1: Introduction to MATLAB

Before you start:

- If you don't have a network drive at

  \\shared.surrey.ac.uk\Shared_Labs\NetworkExamples

  make one by going to $\boxed{\text{Start}} \mapsto \boxed{\text{My Computer}}$. In the window that opens, go to $\boxed{\text{Tools}} \mapsto \boxed{\text{Map Network Drive}}$ on the top. A menu will open, put into the $\boxed{\text{Folder}}$ box the address above. Then click on $\boxed{\text{Finish}}$ and this should give a network drive to this address. Go to this drive and find the folder "COM2023". This has folders like "Data sets" which contains several data sets for use with MATLAB, and "Matlab" which contains files that can be run in MATLAB.

- In your personal files (H-drive), make a folder "COM2023" with a subfolder "Matlab" to do all your MATLAB work.

- If you don't finish the tutorial during the lab session, finish it at later time, but before the next lab session.

- Bookmark the COM2023 web page:

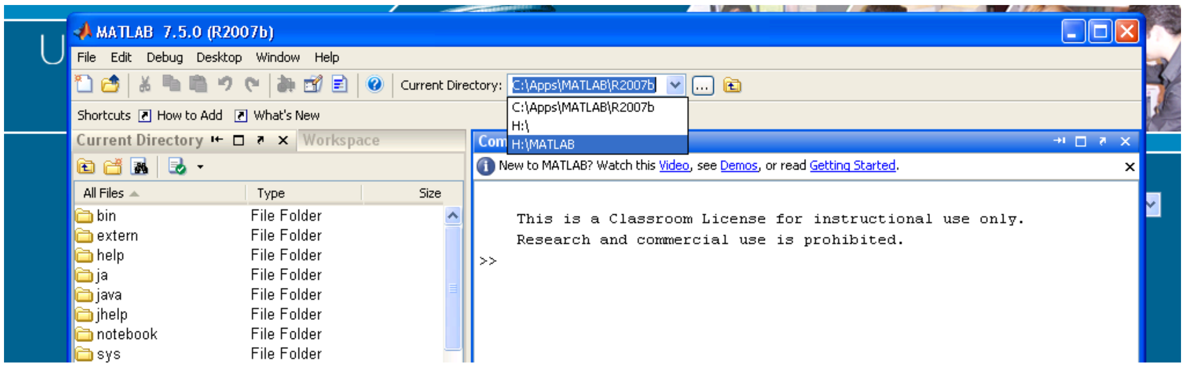
  `http://www.maths.surrey.ac.uk/modules/COM2023.html`

  in your web browser as you may need to check information here.

- It might be useful to check your "Mathematics for Computing I" notes with some parts of this tutorial to refresh your memory.

1.1 How to access MATLAB in the AP Labs

After logging in, click on
$\boxed{\text{Start}} \mapsto \boxed{\text{All programs}} \mapsto \boxed{\text{Departmental Software}} \mapsto \boxed{\text{FEPS}} \mapsto \boxed{\text{Matlab R2009b}}$
to start MATLAB. MATLAB will be able to read and save files to your personal folders on the H-drive if you change the Current Directory to your personal "Matlab" folder, see Figure 1.1.



Current Directory: C:\Apps\MATLAB\R2007b
C:\Apps\MATLAB\R2007b
H:\
H:\MATLAB

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
  This is a Classroom License for instructional use only.
  Research and commercial use is prohibited.
>>
```

Figure 1: Changing "Current Directory" to H-drive.

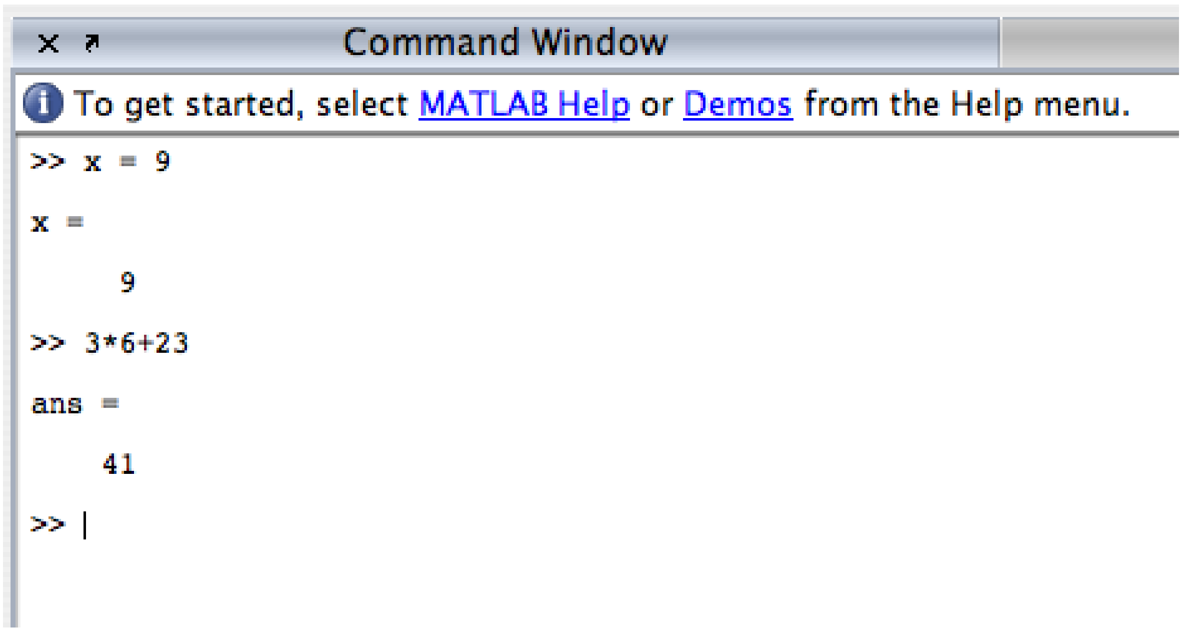
1.2 How to use this MATLAB guide

We will use `type-writer` font for all MATLAB commands. For instance, the command "sin" will appear as `sin`. Once you type a command in the command window and press the return key, it will be executed.

MATLAB will ignore anything after a percentage sign (%). In the rest of this guide, it will be explained what commands do by appending comments on the same line. You do not need to type these comments into MATLAB (they will be ignored if you did). Thus, if you see the text

```
x = 9          % assigns 9 to the variable x
3*6+23         % Matlab can be used as a calculator
```

then you should type only `x=9` followed by the return key and `3*6+23` followed by the return key. The output you will get from these commands is in Figure 2.


Command Window
To get started, select MATLAB Help or Demos from the Help menu.

```
>> x = 9

x =

     9

>> 3*6+23

ans =

    41

>> 
```

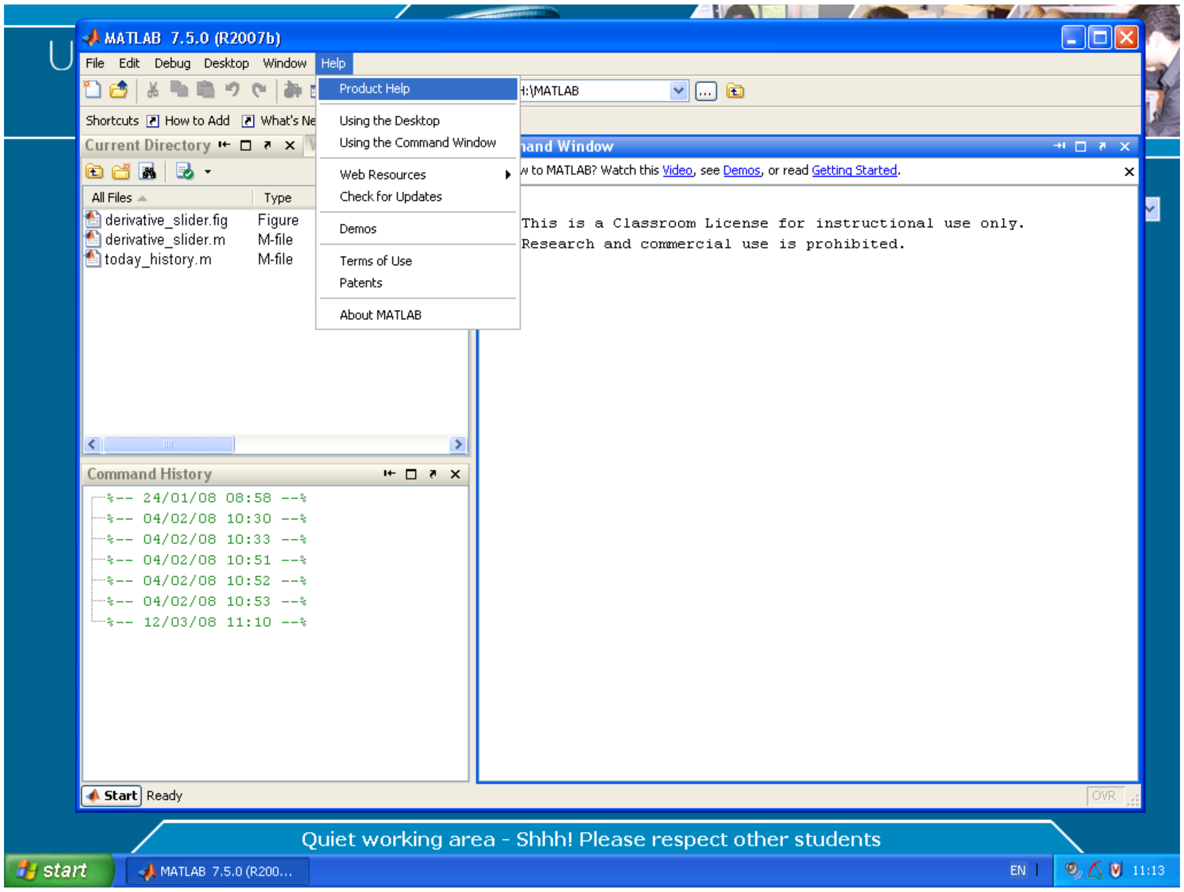
Figure 2: Output of MATLAB commands.

1.3 How to access documentation in MATLAB

MATLAB's Product Help is extremely detailed and useful. You can access help from the menubar as indicated in Figure 3. Afterwards, enter your search term as shown here in Figure 4. Alternatively, if you seek help on a specific command, say on the command `plot`, type

```
doc plot
```

in the command window.



```
%-- 24/01/08 08:58 --%
%-- 04/02/08 10:30 --%
%-- 04/02/08 10:33 --%
%-- 04/02/08 10:51 --%
%-- 04/02/08 10:52 --%
%-- 04/02/08 10:53 --%
%-- 12/03/08 11:10 --%
```

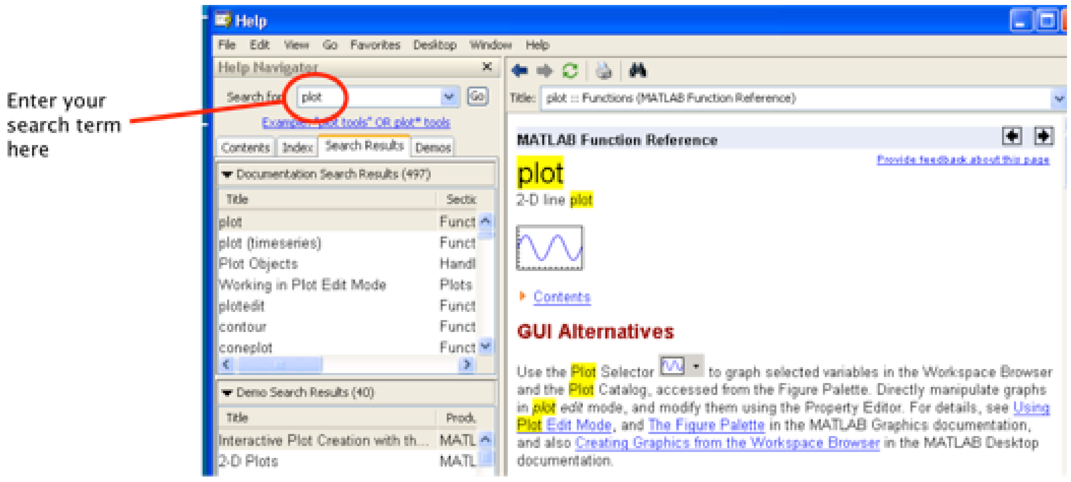
```
This is a Classroom License for instructional use only.
Research and commercial use is prohibited.
```

Figure 3: Accessing MATLAB's Product Help.

1.4 Numbers and equations

MATLAB can be used as a calculator:

```
3+4*5
3^2
pi                       % reserved for pi
format long              % displays more digits (14-15)
pi
format short             % displays only 5 digits
pi
sqrt(9)                  % computes square root of 9
x = sqrt(9)              % assigns output 3 to variable x
x^2                      % x^2=3^2 should give 9 ...
9^(1/2)                  % same as sqrt(9)
sin(pi)                  % note that we do not get exactly zero
log(4), exp(1)           % computes logarithms and exponentials
x = exp(1);              % semicolons at the end hide the output
x                        % prints the value of x
2*10^(-8), 2e-8          % one over 200million can be entered in various ways
```


Enter your search term here
Help
File Edit View Go Favorites Desktop Window Help
Help Navigator
Search for: plot
Go
Title: plot :: Functions (MATLAB Function Reference)
Contents Index Search Results Demos
Provide feedback about this page

Documentation Search Results (497)

Title	Sectic
plot	Funct
plot (timeseries)	Funct
Plot Objects	Handl
Working in Plot Edit Mode	Plots
plotedit	Funct
contour	Funct
coneplot	Funct

Demo Search Results (40)

Title	Prod.
Interactive Plot Creation with th...	MATL
2-D Plots	MATL

MATLAB Function Reference

plot

2-D line plot

Contents

GUI Alternatives

Use the Plot Selector to graph selected variables in the Workspace Browser and the Plot Catalog, accessed from the Figure Palette. Directly manipulate graphs in plot edit mode, and modify them using the Property Editor. For details, see Using Plot Edit Mode, and The Figure Palette in the MATLAB Graphics documentation, and also Creating Graphics from the Workspace Browser in the MATLAB Desktop documentation.

Figure 4: Searching in MATLAB's Product Help.

MATLAB can solve quadratic equations, and many other equations, for us: The command `solve('eqn','var')` solves the equation `eqn` with respect to the variable `var`. Note the use of apostrophes to enclose equations and variables - the apostrophes tell MATLAB that the expressions are symbolic and not numeric:

```
xx=solve('x^2-7*x+10','x')        % matlab can solve equations
xx(1)^2-7*xx(1)+10                % checking that the first solution
                                  % indeed solves the polynomial

solve('x^2=a^2','x')              % solving an equation with symbolic variables
solve('cos(x)=0','x')             % gives only one solution
f = solve('a*x^2+b*x+c','x')      % general solution of quadratic equations
pretty(f)                         % displays answer in a nicer format
y=solve('exp(x)-10','x')
```

Exercise 1

1. Find the first 15 significant digits of π^2.
2. Solve the equation $x^4 - x^3 - 8x^2 + 2x + 12 = 0$ and display the solutions to at least 10 digits.

1.5 M-files

You can use the built-in editor to write longer programs or to store the commands used in a lab tutorial. Open the editor by going to $\boxed{\text{File}} \mapsto \boxed{\text{New}} \mapsto \boxed{\text{Script}}$ (or use the most left button on the menu bar). Type the following lines in it:

```
x = sqrt(2)
x^3
y = sqrt(5)
y+x
```

Click on the $\boxed{\text{Run}}$ button in the menu bar of the editor to save the program and to run it in MATLAB: all MATLAB M-files need to be saved with the extension ".m" Once you saved the program, say as test_problem.m, you can run it from the MATLAB command window by typing `test_problem`. You cannot use hyphens in your file names or begin file names with numbers.

It is a good idea to start each lab tutorial with opening an m-file to save all correct commands for this tutorial with comments to explain waht they are doing. An example with the commands so far can be found in "practical1.m" in the "COM2023/Matlab" folder on the shared drive. Download it, so you can run it. Now continue with this file, copy all correct commands for the exercises in this tutorial in it, and save the file at the end of this session.

1.6 Matrices and vectors

Matrices and vectors are natural ways to store data sets. It is easy to define and use matrices and vectors in MATLAB. The 3×3 matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 10 & 11 & 12 \end{pmatrix}$$

can be entered as follows row by row (note the use of commas and semicolons to separate columns and rows):

```
A = [1,2,3; 4,5,6; 10,12,13]
```

Its columns and rows can be accessed using the colon operator which selects entire rows or columns:

```
A(:,1)                      % first column
A(3,:)                      % third row
A(1:2,:)                    % first and second row
A(3,2)                      % entry in 3rd row, 2nd column
```

Vectors are entered similarly:

```
b = [10; 11; 15]            % column vector
b = [10, 11, 15]            % row vector
```

Vectors and matrices can be transposed by appending an apostrophe:

```
b = [10, 11, 15]            % row vector
b = [10, 11, 15]'           % transpose of row vector = column vector
A                           % outputs A for comparison
A'                          % transpose of matrix A
```

Matrices, vectors and scalars can be multiplied using `*`:

```
A,b                           % outputs A and b
A*b                           % matrix-vector product
B = [1,0,1,2; 2,1,0,1; 0,0,3,1]  % B is a 3x4 matrix
size(B)                       % gives size of matrix/vector: #rows #columns
```

```
A*B                                % matrix-matrix product
2*A                                % scalar-matrix multiplication
A^2                                % same as A*A
b                                  % outputs b
c = [1; 2; 3]                      % second column vector
b'*c                               % scalar product of b and c
b.*c                               % computes a vector whose j-th component is b_j c_j
size(b)                            % size works also for vectors
```

Note the difference between `b'*c` and `b.*c`. The operation `*.` acts pointwise on the elements of the vectors or matrices, which must have the same size. Compare the results of the following commands, also look at the sizes of the resulting matrices or vectors.

```
A=[1,2;3,4;5,6]                    % a 3x2 matrix
B=[7,8,9;1,2,3]                    % a 2x3 matrix
A*B
B*A
A.*B                               % !error: size(A) differs from size(B)
A.*B'
B.*A'
b=[1,2,3], c=[4,5,6]
b'*c
b*c'
b.*c
b'.*c'
```

You can also apply operations to certain columns or rows

```
B(:,1) = 3*B(:,1)                  % multiplies the first column of B by 3
B(:,1) = B(:,1)+B(:,2)             % adds second column to first column
a = [1; 2; 3; 4]                   % column vector
a = [a ; 5]                        % adds an entry at the bottom
b = [ -1; -2]                      % another column vector
c = [b;a]                          % prepends a with b
```

To get a matrix with ones or zeros use `ones` or `zeros`

```
ones(3,2)                       % gives a 3x2 matrix with ones
zeros(2)                        % gives a 2x2 matrix with zeros
eye(2)                          % gives the 2x2 identity matrix
```

Exercise 2 Define $D = \begin{pmatrix} 2 & 4 \\ 6 & 8 \\ 1 & 3 \end{pmatrix}$. Find the matrix product of D with a 2×4 matrix with ones and the point wise product of D with a 3×2 matrix with ones. Why do we have to take differently sized matrices in those operations. Are these the only possible choices?

1.7 Plotting functions

Matlab is good at plotting graphs of functions. There are at least two ways of graphing functions. First, we can use `ezplot`. Note the use of apostrophes to enclose the function - the reason will become clear later on:

```
ezplot('sin(x)',[-pi,2*pi])   % plots sin(x) for -pi<x<2*pi
ezplot('log(x)',[0,10])       % plots the logarithm
```

The second option is to use the command `plot` which can be used to visualize two vectors **x** and **y** whose components x_k and y_k define points (x_k, y_k) in the plane. For this to make sense, **x** and **y** need to have the same dimension:

```
x = [0,pi/4,pi/2,3*pi/4,pi]
y = sin(x)                    % compute the sine of each entry of x
plot(x,y)                     % a rather ragged graph of the sine function
y= x.^2                       % compute x^2: note .^
plot(x,y)                     % plot x.^2
```

We can improve on this using the colon operator:

```
x = [0 : 0.1 : 2*pi]
```

The result is a vector whose first entry is zero, while consecutive entries increase from zero by steps of 0.1 until we reach the largest value below 2π. Applying `y=sin(x)` gives the corresponding values of the sine function which we can plot as before:

```
x = [0:0.1:2*pi];             % the semicolon hides the output of x
y = sin(x);                   % applies sin to each entry of vector x
plot(x,y,'r')                 % plots using a red line
hold on                       % consecutive plots will be added to the same figure
ezplot('sin(x)',[0,2*pi])     % same graph
hold off                      % the next graph will replace the previous graphs:
plot(x,y,'b')                 % plots graph in blue
```

Matlab offers many ways of changing the color and style of your plots and adding titles and axis labels. To get started, click on the arrow point in your plot, see Figure 5. Next, click on the Show Plot Tools button, see left image in Figure 6. An extra window will pop up. If you select the graph with your arrow pointer, the graph will look as shown in the right image of Figure 6. You can now edit colour and line thickness in the second window, see Figure 7.

Matlab can also plot implicitly defined curves. A circle of radius 2, for instance, is the set of points (x, y) for which $x^2 + y^2 = 2^2$. It can be plotted using the command

```
ezplot('x^2+y^2=4')           % plots solutions of x^2+y^2=4: a circle of radius 2
```

Alternatively, we could have used the command

```
ezplot('x^2+y^2-4')           % plots solutions of x^2+y^2=4: a circle of radius 2
```

as Matlab implicitly assumes that we set the expression equal to zero.

Finally, Matlab plots surfaces in three dimensions. Two examples are given below:

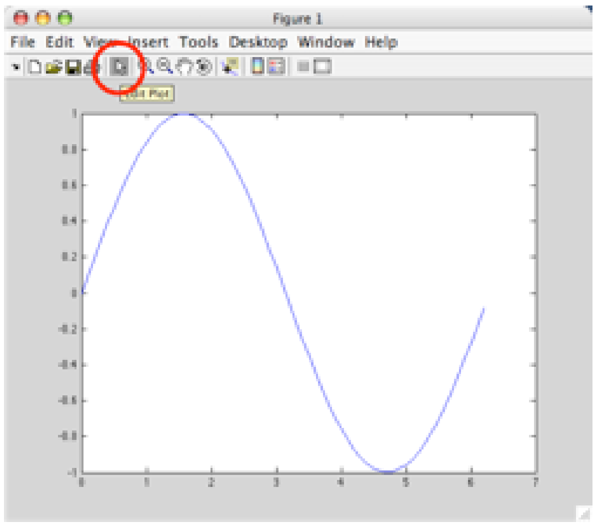


Figure 5: Using the arrow point in graphs.

```
ezsurf('x^2-y^2')                  % plots a hyperbola in three dimensions
ezsurf('x^2+y^2','circ')           % plots a parabola over a disk
```

Select the rotation handle as shown in Figure 8. Next, point your mouse into the figure, hold your mouse button down and move the mouse around: This allows you to rotate the surface and look at it from different perspectives.

Exercise 3 Plot two functions of your choice using both the ezplot and plot commands. Use ezsurf to plot a new surface and rotate it around.

1.8 Importing data

We can import data from text files and change and modify these as well. An example data file can be downloaded from the "Data set" folder in the shared drive. You have created a network drive to this folder in the beginning of this session. Now download the file "sample_data.txt" in your personal "Matlab" folder.

This file contains a header line with the entries year and apples, and two columns of numbers, separated by tabs. Now, in your MATLAB window, click on $\boxed{\text{File}} \mapsto \boxed{\text{Import Data}}$ and select the downloaded file. MATLAB will recognize the tab-separated data, so simply click on Next. In the next window, select "Create vectors from each column using column names" option and click on $\boxed{\text{Finish}}$. The two columns of data can be accessed by typing

```
year
apples
```

We can plot these data as before using the plot command:

```
plot(year,apples)
```

It is possible to plot only the individual points in the data file:

```
doc plot                      % find out how we can plot with red stars
plot(year,apples,'r*')
```

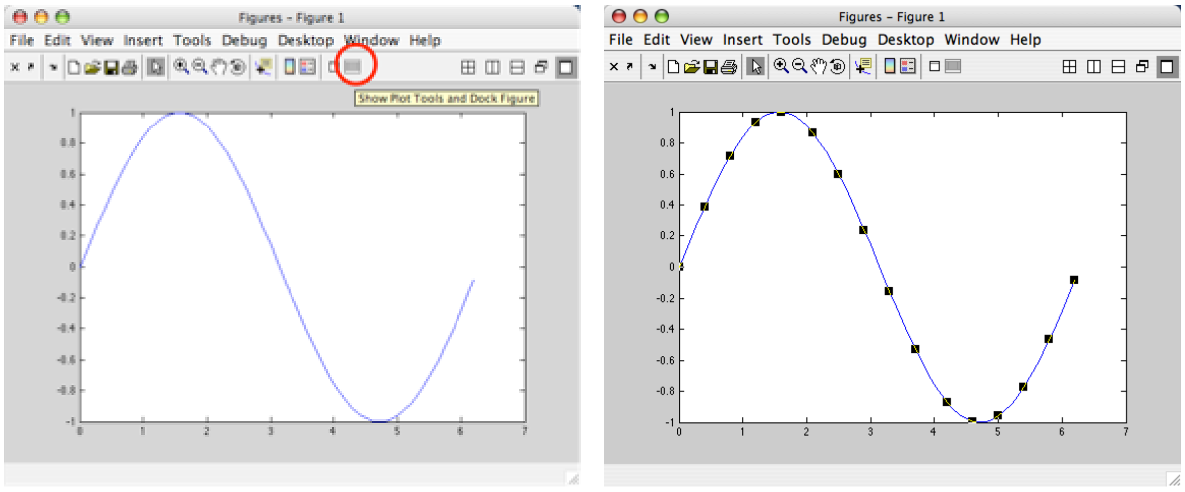
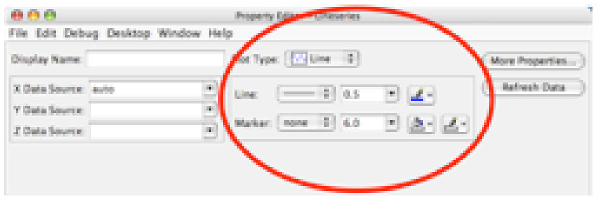

Figures – Figure 1
File Edit View Insert Tools Debug Desktop Window Help
Show Plot Tools and Dock Figure
0
1
2
3
4
5
6
7
-1
-0.8
-0.6
-0.4
-0.2
0.2
0.4
0.6
0.8

Figure 6: Using the Show Plot Tools button (left) and the arrow pointer (result is right)in graphs.


Property Editor – Lineseries
File Edit Debug Desktop Window Help
More Properties...
Refresh Data

Display Name:
X Data Source: auto
Y Data Source:
Z Data Source:
Plot Type: Line
Line: 0.5
Marker: none 6.0

Figure 7: Edit colour and line thickness in graphs.

1.9 Linear Algebra

MATLAB excels at linear algebra problems. If you re-use matrices and vectors, it is a good idea to clear them before using them for a different purpose:

```
B                            % displays the 3x4 matrix B
clear B                      % clears B and forgets that it is a 3x4 matrix
B                            % gives an error
```

MATLAB computes determinants and inverses:

```
format long                  % displays more digits
A = [1,2,3; 4,5,6; 10,12,13] % sets A
det(A)                       % determinant of A
C = A^(-1)                   % C is inverse of A
A*C, C*A                     % should give the identity
```

Using the inverse matrix, we can solve systems of linear equations:

```
b = [10; 11; 15]             % sets b
x = A^(-1)*b                 % solution x of Ax=b
```

The system $A\mathbf{x} = \mathbf{b}$ can also be solved by using the backslash ($\backslash$) operator (which is much faster for large systems). MATLAB uses Gaussian elimination to compute the solution $\mathbf{x} = A\backslash\mathbf{b}$:

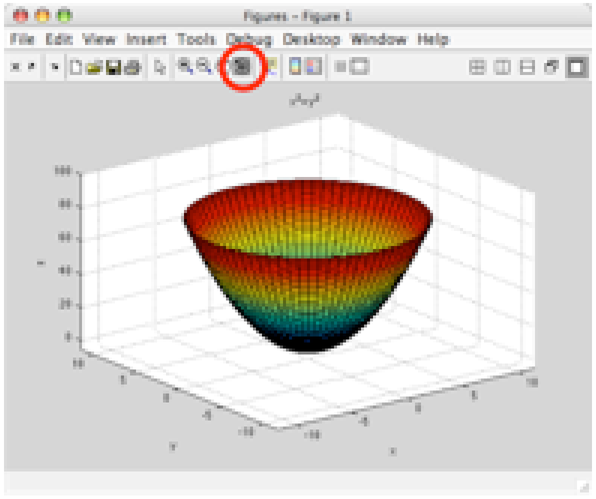


Figure 8: Selecting the rotation angle.

```
x = A\b                          % solution x of Ax=b via Gaussian elimination
A*x-b                            % check answer: Ax-b should be approximately zero
```

MATLAB can also compute eigenvalues and eigenvectors: Recall that λ is a eigenvalue of A if there is a nonzero vector $\mathbf{v}$ so that $A\mathbf{v} = \lambda\mathbf{v}$. In particular, $A\mathbf{v} - \lambda\mathbf{v} = \mathbf{0}$ for every eigenvalue λ with eigenvector $\mathbf{v}$. Eigenvalues can be computed using `eig`:

```
eig(A)                           % eigenvalues of A
```

Eigenvectors can be computed by assigning `[V,d]` to `eig(A)`:

```
[V,D] = eig(A)
```

The columns of the 3×3 matrix `V` contains the eigenvectors. The corresponding eigenvalues are the entries on the diagonal of the diagonal 3×3 matrix `D`. Thus, the column vector `V(:,1)` is the eigenvector belonging to the eigenvalue `D(1,1)` of A. Similarly, `V(:,2)` is the eigenvector belonging to the eigenvalue `D(2,2)`, and so on. In particular,

```
A*V(:,1) - D(1,1)*V(:,1)
```

should be approximately zero as this equation is $A\mathbf{v} - \lambda\mathbf{v} = \mathbf{0}$ for the first eigenvalue.

Exercise 4 Calculate the eigenvalues and eigenvectors of $A = \begin{pmatrix} 2 & 1 \\ 2 & 3 \end{pmatrix}$.

1.10 Sums and loops

MATLAB has powerful programming capabilities. We will not use them much but loops will be very useful to compute series and sums.

Suppose that we wish to compute the sum $\sum_{k=1}^{n} k^2$ for a given number n, say $n = 20$ (the sum is actually equal to $n(n+1)(2n+1)/6$ for each n). A way of computing this sum is by using a for loop which executes several lines of instructions a given number of times. In Matlab, type

```
x=0;                % initializes the sum x
for k=1:20          % loop will be traversed 20 times with k increasing from 1 to 20
  x=x+k^2;          % x is increased by k^2
                    % note that Matlab does not execute anything yet
end                 % after pressing return, the loop is executed
x
```

The `x` at the end of the previous line outputs the value of the sum. Alternatively, you can write the above commands in a single line:

```
x=0; for k=1:20, x=x+k^2; end; x
```

If we wish to compute a large number of values of a sum, we can store them in a vector. Consider the sum

$$a_n = \sum_{k=1}^{n} \frac{1}{k^2}$$

so that $a_n = a_{n-1} + \frac{1}{n^2}$. We can compute a_n in Matlab using

```
clear a                     % clear a to be on the safe side
a(1)=1;                     % value of  for
for n=2:100                 % computes  for  to 200
    a(n)=a(n-1)+1/n^2;
end
plot(a,'r*')
```

which computes and plots a_n for the first 200 n's. The plot indicates that the series converges (which it does: the limit is $\frac{\pi^2}{6}$).

Exercise 5

1. Compute the value of the series $\sum_{k=5}^{25} k^3$ (make sure to clear `a` to erase the computation done above).
2. Compute the values $a_n = \sum_{k=1}^{n} (-1)^k/k$ for n between 1 and 30. Plot these values.
3. Find the `sum` command in the help and recompute the above sums using this command by defining first a vector `a`. Hint: `k = 1:n; a = 1./k;` defines a vector $a(k) = 1/k$.

1.11 Symbolic computations

Matlab can also perform a range of symbolic calculations. To access those, variables need to be declared explicitly as symbolic:

```
syms x                      % defines x as a symbolic variable
```

which allows us to use `x` in the solve command without apostrophes:

```
solve(x^2-10)
```

The following example explores the relation between syms and apostrophes further:

```
syms x                        % defines x as symbolic variable
ezplot(sin(x))                % plots sin(x)
clear x                       % clears definition of x
ezplot(sin(x))                % error! x is not recognized as symbolic variable
ezplot('sin(x)')              % use 'sin(x)' or
syms x                        % define x again as symbolic variable
ezplot(sin(x))
```

Matlab can differentiate and integrate:

```
diff(sin(x),x)                % differentiates sin(x)
syms a b
diff(a*x^2-b,x)               % differentiates with respect to x
diff(a*x^2-b,a)               % differentiates with respect to a
diff(a*x^2-b,b)               % differentiates with respect to b
int(x^2,x,0,1)                % integrates x^2 over 0<x<1
int(a*x,a,0,1)                % integrates a*x over 0<a<1
int(sin(a*x),x,0,2*pi)
```

We can also use symbolic computations for matrices:

```
A = [1,2,3; 4,5,6; 6,7,8]
eig(A)
A = sym(A)                    % treats A as a symbolic matrix
eig(A)                        % compare the two expressions you got!
```

The command `simple` allows you to simplify expressions. Simplifying means finding the shortest expression:

```
f = simple(cos(x)^2+sin(x)^2)  % finds shortest equivalent expression
f = (1/x^3+6/x^2+12/x+8)^(1/3) % lengthy expression
f1 = simple(f)                 % much shorter
f2 = simple(f1)                % applying 'simple' again may simplify further
```

If you use `simple` without assigning the output to a variable, it shows you which commands it applied to arrive at the shortest equivalent expression:

```
simple(f)                     % shows which commands were tried out
```

1.12 Accuracy of numerical versus symbolic calculations in Matlab

Numerical computations are only accurate to a certain finite number of digits. The variable `eps` contains the smallest positive number so that 1+`eps` is not equal to 1. Thus, if we add a positive number smaller than `eps` to 1, the computer will think it is equal to 1. To find out what `eps` is on your computer, type

```
eps                           % eps should be of the order 1e-16
```

Symbolic calculations, on the other hand, are precise: There is no error involved as symbolic computations are done in Matlab as you would do them with pen and paper (maybe a tiny bit faster and more accurate).

There are situations where one has to be careful in interpreting the results of numerical computations in Matlab. Here is an example: The matrix $A = \begin{pmatrix} 1 & 1 \\ 0 & 10^{-18} \end{pmatrix}$ is invertible (check!) and therefore has rank 2 (the two rows are linearly independent). We now check this in Matlab

```
format long
A = [1, 1; 0, 1e-18]          % this matrix is invertible and has rank 2
rank(A)                       % oops: the lower right entry is too small!
A = sym(A)                    % treat A as a symbolic matrix
rank(A)                       % gives the correct answer
syms B                        % defines B as a symbolic variable
B = A^(-1)                    % computes the inverse symbolically
A*B                           % gives the identity (exactly!)
```

Why do we not compute everything using symbolic calculations? The answer is that symbolic calculations take much more time to carry out which renders them impractical for even moderately sized problems.

Lab Tutorial 2 - Simple data graphics and line plots

2.1 Simple data graphics

To illustrate the use of a pie chart and bar plot, we consider the following dataset about the average annual consumption of breakfast cereals in various countries:

Country	Consumption (pounds)
Ireland (IRL)	15.4
Great Britain (GB)	12.8
Australia (AUS)	12.3
USA	9.8
Canada (CAN)	8.7
Denmark (DMK)	4.6

The commands below construct a pie chart for this data. Execute these commands

```
>> Consumption=[15.4, 12.8, 12.3, 9.8, 8.7, 4.6];
>> Countries={'IRL','GB','AUS','USA', 'CAN','DMK'};
>> pie(Consumption,Countries)
>> title(['\fontsize{17}{Annual Breakfast Cereal Consumption}'])
```

Execute the following commands to create a bar plot:

```
>> bar(Consumption)
>> set(gca,'XTickLabel',Countries)
>> title(['\fontsize{17}{Annual Breakfast Cereal Consumption}'])
```

2.2 Plotting Data

The `count.dat` dataset depicts the hourly vehicle count at a 3 different intersections. Load the `count.dat` dataset. This creates a 24×3 matrix called *count* where the rows give 24 hourly vehicle counts and the columns denote the intersection. Take a look at the matrix, `count`.

```
>> load count.dat
>> count
```

Set up 3 different vectors called $I1$, $I2$ and $I3$ to store the vehicle counts for each intersection separately and a vector i which gives the index of the hours.

```
>> I1=count(:,1);
>> I2=count(:,2);
>> I3=count(:,3);
>> i=1:24;
```

Note that the ';' at the end of a command supresses the output. View plots of the data by typing the following command:

```
>> plot(i,I1,i,I2,i,I3)
```

By providing the plot command with 3 different pairs of data, it will put the 3 graphs on one plot. MATLAB will automatically make it easy to distinguish between the graphs by plotting them in different colours. We can specify which colours we want the graphs to be. Try the following:

```
>> plot(i,I1,‘b’,i,I2,‘y’,i,I3,‘g’)
```

We can also change the style of the graphs. Type in the following:

```
>> plot(i,I1,‘b.:’,i,I2,‘y*--’,i,I3,‘gx-.’)
```

The graph colours are the same but the styles have been changed. Use `help plot` to find out what other colours and styles are available. Experiment with a few.

You can add graphs to a plot one at a time with the following commands:

```
>> plot(i,I1,‘b.:’)
>> hold on
>> plot(i,I2,‘y*--’)
>> plot(i,I3,‘gx-.’)
>> hold off
```

Alternatively, we can view the graphs separately in subplots.

```
>> subplot(3,1,1); plot(i,I1,‘b.:’)
>> subplot(3,1,2); plot(i,I2,‘y*--’)
>> subplot(3,1,3); plot(i,I3,‘gx-.’)
```

It is hard to see how the bottom graph compares with the other 2 because the vertical scale on the bottom graph is different from the others. Type in

```
>> subplot(3,1,1); set(gca,’YLim’,[0,400]); subplot(3,1,2); set(gca,’YLim’,[0,400]);
```

Use `help set` for details of what this command does. Among other things, this can also be achieved within the plot window, as opposed to the Matlab workspace, by clicking on `File - Edit - Axes Properties`. or you can use the “Property Editor”, which can be accessed by clicking on `View - Property Editor`. An extra window will be attached at the bottom of the plot. If you click on an element of the plot (e.g., the line or the axis), the bottom window will display options for this element. Use this to change the colour of the lines and to change the markers.

2.3 Labelling Plots

Making sure that `hold` is off, we return to the single plot of 3 graphs and this time we give the plot a title:

```
>> plot(i,I1,‘b.:’,i,I2,‘y*--’,i,I3,‘gx-.’)
>> title(‘Hourly Vehicle Counts at 3 Intersections’)
```

You might think that the title appears quite small relative to the graph and want to change the size of it. Type in the following line:

```
>> title([’\fontsize{17}{Hourly Vehicle Counts at 3 Intersections}’])
```

Try a few other sizes to see which size you prefer. Add the following lines:

```
>> xlabel(‘Hour Index’)
>> ylabel(‘Vehicle Count’)
>> legend(‘Intersection 1’, ‘Intersection2’, ‘Intersection 3’,...
   ‘Location’, ‘NorthWest’)
```

These commands add x and y labels and a legend to the plot. The last two arguments of the legend command specify where you want the legend to appear on the plot. Use `help legend` to find other options for the `legend` command and experiment with a few.

2.4 Plotting Functions

The `plot(x,y)` command, where x and y have the same length, plots x against y and so when plotting functions, for example $y = \sin(x)$, only values of the function evaluated for given x values are plotted. The following commands plot the graph of $y = sin(x)$ for various x:

```
>> x1=(-1:0.5:1)*pi;
>> x2=(-1:0.1:1)*pi;
>> x3=(-1:0.05:1)*pi;
>> y1=sin(x1); y2=sin(x2); y3=sin(x3);
>> plot(x1,y1,x2,y2,x3,y3)
```

The higher the number of points to evaluate a function at, the more accurate the graphical representation.

Alternatively, one may use `ezplot` to plot functions without defining a vector `x`. Type in the following command:

```
>> ezplot('sin(x)',[-pi,pi])
```

`ezplot('f(x)',[a,b])` will plot the function $f(x)$ over the range $[a, b]$. Note that $f(x)$ is in apostrophes.

MATLAB can also create 3-dimensional plots with the `plot3` function:

```
>> t=0:0.01:10;
>> plot3(sin(t),cos(t),t)
```

By clicking on the 'Rotate 3D' button on the plot toolbar, one can then grab anywhere on the plot and move it around to obtain better views.

MATLAB can also plot surfaces:

```
>> [x,y]=meshgrid(-2:0.1:2,-2:0.1:2);
>> surf(x,y,x.^2-y.^2)
```

Note that '.' before an operator tells MATLAB that you want to perform that operation for the whole vector and so MATLAB will do each of the calculations in a piecewise fashion.

2.5 Exercises

1. Load the `gas` dataset. This creates 2 vectors: `price1` and `price2`. The vector `price1` lists January 1993 gasoline prices at 20 different locations in Massachusetts. And `Price2` lists the gasoline prices at the same locations one month later.

   (a) Show the graphs `price1` and `price2` against location index on the same plot. Make sure that `price2` has a different colour and style to `price1`.

   (b) Give the plot a title.

   (c) Add x and y labels to the plot.

   (d) Add a legend to identify the different price graphs.

   What do you conclude?

2. Let $f(x) = \cos(x)$, $g(x) = 3\cos(x)$ and $h(x) = \cos(3x)$.
   (a) Show the graphs of $f(x)$, $g(x)$ and $h(x)$ on subplots.
   (b) Make sure that the vertical and horizontal scales are the same.
   (c) Give each subplot an x-label and a y-label which describes the function.
   (d) Give the overall plot a title.
3. Consider the surface $z = x^3 + y$.
   (a) Plot the surface against x and y.
   (b) Give the plot x and y labels and a title.
   (c) Use the `help` command to switch off the gridlines.

Lab Tutorial 3 - Data numerics and graphics

3.1 Numerical Summaries

Type the following data set into a row vector called "redcell":

$\{243, 251, 275, 291, 347, 354, 380, 392, 206, 210, 226, 249, 255, 273, 289, 295, 309, 241, 258, 270, 293\}$.

These numbers are measurements from a red cell folate level measurement test, which is why the vector is called `redcell`.

```
>> redcell=[243, 251, 275, 291, 347, 354, 380, 392, 206, 210, 226,
    249, 255, 273, 289, 295, 309, 241, 258, 270, 293];
```

The commands `mean(x)`, `median(x)`, `min(x)`, `max(x)`, `var(x)`, and `std(x)` enable you to compute the mean, median, minimum value, maximum value, variance and the standard deviation (square root of variance) of a data set `x`. Find the deviation for the `redcell` data set by using the commands below.

```
>> mean(redcell)
>> median(redcell)
>> min(redcell)
>> max(redcell)
>> var(redcell)
>> std(redcell)
>> iqr(redcell)
```

Note that `var(redcell)` displays the variance of `redcell` in floating point notation: $2.6589e+03 = 2.6589 \times 10^3$. Write this in the standard notation by looking at the documentation for the command `format`.

The `quantile` and `iqr` commands allow you to calculate individual quartiles and the inter-quartile range (the difference between the 75th and 25th percentile) respectively.

```
>> quantile(redcell,0.25);
>> quantile(redcell,0.75);
>> iqr(redcell);
```

3.2 Graphical Summaries

Use the following commands to produce histograms for the redcell data set. Notice the effect of executing each line at a time.

```
>> hist(redcell)
>> hist(redcell,5)
>> hist(redcell,20)
>> title('Red Cell Results')
```

The number in the 2nd argument of `hist` determines how many groups will be included on the histogram; 10 is the default if it isn't specified. Now produce boxplots of the same data set using the following commands:

```
>> boxplot(redcell)
>> boxplot(redcell,'colors', 'r')
```

Use the `help` to find out more information about `boxplot`. Using appropriate subcommands produce a blue boxplot with a title, a y-axis label and green asterisk outliers.

Now produce empirical cumulative distribution functions for the redcell data set using:

```
>> ecdf(redcell)
```

3.3 Example

MATLAB has a built-in data set called '*fisheriris*' which is a dataset including 'sepal length', 'sepal width', 'petal length' and 'petal width' measurements for 150 irises; 50 readings from 3 different species of iris. The '*fisheriris*' dataset can be loaded into MATLAB with the following command:

```
>> load fisheriris
```

By executing this command you have told MATLAB to create 2 arrays: `meas` and `species`. `meas` has 150 rows and 4 columns whereas `species` has 150 rows and 1 column. `meas` contains the observed measurements whereas `species` contains a set of repeated strings to identify the species of the corresponding measurements.

Produce a box plot of sepal length according to each species with the following commands:

```
>> SW=meas(:,1);
>> boxplot(SW,species)
```

One can store just the setosa, versicolor and virginica information in vectors called *setosa*, *versicolor* and *virginica* using

```
>> SWL=meas(1:50,:);
>> VWL=meas(51:100,:);
>> ViWL=meas(101:150,:);
```

Note that the ' : ', means that the column number is unspecified in which case MATLAB will take the entire row. Alternatively,

```
>> indices1=strcmp('setosa',species);
>> indices2=strcmp('versicolor',species);
>> indices3=strcmp('virginica',species);
```

`strcmp(entry1,entry2)` compares *entry1* with *entry2* and returns a 1 where they are the same and a 0 otherwise. Thus `indices1` will be vector with 150 elements, being either "0" or "1". Now execute these commands:

```
>> setosa=meas(indices1,:);
>> versicolor=meas(indices2,:);
>> virginica=meas(indices3,:);
```

We have set up 3 matrices which have extracted the information from `meas` corresponding to each species.

```
>> iris=cat(3,setosa,versicolor,virginica);
```

takes the 3 matrices and concatenates them into a 51 by 4 by 3 array called `iris`.

The following commands set up an array called `iris1` which contains the same information as `iris` but also adds index names to each dimension:

```
>> iris1 = cell(51,5,3);
>> obsnames = strcat({'Obs'},num2str((1:50)','%-d'));
>> iris1(2:end,1,:) = repmat(obsnames,[1 1 3]);
>> varnames = {'SepalLength','SepalWidth','PetalLength','PetalWidth'};
>> iris1(1,2:end,:) = repmat(varnames,[1 1 3]);
>> iris1(2:end,2:end,1) = num2cell(setosa);
>> iris1(2:end,2:end,2) = num2cell(versicolor);
>> iris1(2:end,2:end,3) = num2cell(virginica);
>> iris1{1,1,1} = 'Setosa';
>> iris1{1,1,2} = 'Versicolor';
>> iris1{1,1,3} = 'Virginica';
```

Using the help command where necessary, follow and understand the above steps.

What do the following commands do? Note the use of `iris1` and `iris` and how the use of indices will differ.

```
>> iris1(:,:,1)
>> iris(:,:,1)
>> iris1(2,:,3)
>> iris(2,:,3)
>> iris(1,:,3)
>> iris(:,2,2)
>> iris1(:,3,2)
>> iris1(2,:,3)
```

3.4 Exercises

1. What is the mean and variance of the petal length of the versicolor species of iris in the data set '*fisheriris*'?
2. Illustrate the sepal width distribution of virginica species of iris given in the '*fisheriris*' dataset with a histogram, including titles and labels.
3. Create a diagram which shows 3 boxplots of the petal width corresponding to each species sampled in the '*fisheriris*' dataset, including titles and labels.

Lab Tutorial 4 - Probability and probability density function

4.1 Probability

We can use MATLAB to help calculate probabilities. Load the `hogg` data set and view the matrix `hogg` that it creates.

```
>> load hogg
>> hogg
```

The matrix `hogg` shows 6 sets of bacteria counts (rows) from 5 different shipments of milk (columns). Thus there are $6 \times 5 = 30$ counts in total. We also calculate the total number of bacteria found:

```
>> total = 6*5;
>> sum_sample = sum(hogg)
   sum_sample =
       143    80    70    55   107
>> totalbac=sum(sum_sample)
```

Define the following events:

- A is the event "one count has 7 bacteria";
- B is the event "one count has more than 20 bacteria".

To find the number of events in A and B and the probabilities of A and$B$, we use the following (check what is going on!):

```
>> hogg7 = (hogg == 7)         % creats a matrix with ones where hogg
                               % is 7 and 0 if not
>> tot_A = sum(sum(hogg7))     % add all the elements in the rows and colums
>> PA=tot_A/total              % probability of A
>> hogg20plus = (hogg > 20)
>> tot_B = sum(sum(hogg20plus))
>> PB=tot_B/total
```

You should have found that 5 have a bacteria count of 7 and 6 have a bacteria count of over 20. Hence, the probability of A and B are given by $P(A) = \frac{1}{6}$ and $P(B) = \frac{1}{5}$.

The event A^c is the event that a number other than 7 is counted and its probability is given by $P(A^c) = 1 - P(A) = \frac{5}{6}$.

```
>> PAc=(total-tot_A)/total
```

Note that A and B are disjoint since you cannot have a count of 7 that is more than 20, hence, $P(A \cap B) = 0$. However, we can have a count of 7 or more than 20 and since the 2 events are disjoint, $P(A \cup B) = P(A) + P(B) = 11/30$. To verify this, we use the following commands (again, check what is going on):

```
>> AnB = ((hogg7 == 1) & (hogg20plus == 1))
>> AuB = ((hogg7 == 1) | (hogg20plus == 1))
>> PAnB = sum(sum(AnB))/total
>> PAuB = sum(sum(AuB))/total
```

Now define two more events:

- C is the event "a bacteria came from Shipment 4";
- D is the event "a bacteria was found in Sample 6".

To find the probabilities of those events:

```
>> PC = sum_sample(4)/totalbac
```

The total number of bacteria found is 455, 55 of which came from Shipment 4, hence, the probability that a bacteria came from Shipment 4 is $P(C) = 0.1209$.

Similarly, $P(D) = 0.2088$ (recall that `hogg'` is the transpose of `hogg`)

```
>> sum_ship = sum(hogg')
>> PD=sum_ship(6)/totalbac
```

Next we find the probability that a bacteria came from Shipment 4 and was found in sample 6:

```
>> CnD= hogg(6,4);
>> PCnD=CnD/totalbac
```

On the other hand, the probability that a bacteria came from Shipment 4 given that it was found in Sample 6 is given by $P(C|D) = \frac{P(C \cap D)}{P(D)}$:

```
>> PCgD=PCnD/PD
```

4.2 Probability Density and Distribution Functions

In section 4.2 of the notes, we have seen that for continuous random variables, we have to calculate the area under the graph of the probability density function (pdf) in order to find probabilities for intervals and to get the distribution function. Consider the probability density function:

$$f(x) = \frac{3(2-x)(2+x)}{32} \qquad \text{for} -2 \le x \le 2.$$

In order to view a plot of this pdf, let x be the sequence of values from -2, increasing by 0.1 up to 2, then calculating the value of $f(x)$ for each value of x:

```
>> x=-2:0.1:2;
>> f=3*(2-x).*(2+x)./32;
>> plot(x,f)
```

Note that '.' before an operator lets MATLAB know that we are performing piecewise operations on the elements of x.

Note that an alternative to the method above is to create a function, save it as an $M-file$ and call the function for element of x. In the MATLAB workspace click `File - New - M-file` and type the following lines into the editor:

```
    function ff = alternateF(x)
    ff=3*(2-x)*(2+x)/32;
```

the command `function` tells MATLAB that we are writing a function, `ff` is what we want the output of the function to be, `alternateF` is what we are calling our function and x is the input for the function. The proceeding lines then determine how our output, ff, depends on the input x. In this environment, the function only works on one value of x at a time and so there is no need for '.' before each operator. Save the file as *alternateF.m*.

An equivalent way of defining a function is the following command

```
>> func= @(x) 3*(2-x).*(2+x)./32;
```

This defines a function `func`, similar to the function `alternateF` defined with the m-file. The method with the m file is more suitable if the function is more complicated than this one.

These methods are useful for when a function is used over and over again.

Now go back to the MATLAB workspace and type the following:

```
 >> for i=1:length(x)
    f2(i)=alternateF(x(i));
    end
>> f3 = func(x)
```

This gives vectors `f2` and `f3` which the same as the vector `f` in the previous commands. Looking the commands for `f2`, on the RHS, for each value of $x(i)$ we are calling the *alternateF* function, $x(i)$ is the input and so every x in the function editor takes the value of $x(i)$. Whatever is returned from the function call gets assigned to the LHS. Since we are increasing the index on the LHS by 1 each time, we are creating a vector which is storing the output of each function call.

Try plotting `f2` and `f3` and see if you get the same pdf. Recall that a pdf f should be non-negative, i.e. $f(x) \geq 0$ for all x. The area under the graph of a pdf should also equate to 1. MATLAB can calculate areas under a graph (definite integrals) and also socalled indefinite integrals. In both cases, before an integral is calculated with respect to x, x must first be declared as symbolic:

```
 >> syms x
 >> int(func(x),x,-2,2)         % area under the graph of func with x
                                % between -2 and 2
>> int(alternateF(x),x)         % indefinite integral
```

In order to find the probability that a random variable, X, with pdf,

$$f(x) = P(X = x) = \frac{3(2-x)(2+x)}{32}, \text{ for } -2 \leq x \leq 2 \text{ and } f(x) = 0 \text{ for } |x| > 2 \qquad (1)$$

lies within a certain interval, we can change the limit arguments in the commands above. For example, the probability that X is between -0.5 and 0.25 is given by

```
 >> int(func(x),x,-0.5,0.25)  % area under the graph of func with x
                              % between -0.5 and 0.25
```

We can modify the file *alternateF.m* to include values with $|x| > 2$ and truely represent equation (1). Open *alternateF.m* again and change to

```
function ff = alternateF(x)
if (-2 <= x) & (x <= 2) ff=3*(2-x)*(2+x)/32;
else ff= 0;
end
```

And save again.

Recall that the probability distribution function (pcf), $F(y)$, of a random variable X gives the probability that $X \leq y$, i.e. $F(y) = P(X \leq y)$. The pcf can be found by integrating the pdf, $f(x)$, over the interval $-\infty \leq X \leq y$. Now we are ready to calculate the pcf. For example at $x = 0$, $F(0)$, is given by

$$F(0) = P(-\infty \leq X \leq 0) = P(-2 \leq X \leq 0) = \int_{-2}^{0} f(x)dx = \frac{1}{2}$$

as expected (why do we expect this?)

```
>> int(func(x),x,-2,0)
```

Note that we use `func`. Unfortunately, MATLAB is not very good in symbolic calculations and it struggles with our definitions which use inequalitites in `alternateF`. This can be fixed by using the socalled `heaviside` function in the definition of `alternateF`, but we will not discuss the details here. Ask the lecturer if you are interested.

To see a plot of the distribution function we take x as the sequence before, and let `F` be the vector of integrals from -2 to each value of `x`:

```
>> x=-2:0.1:2;
>> F=zeros(1,length(x));
>> syms y
>> for i=1:length(x)
   F(i)=int(func(y),y,-2,x(i));
   end
>> plot(x,F)
```

Note that we have to define `x` again as we declared it to be a symbolic variable earlier and hence lost its meaning as a vector. To visualise that the probability distribution function is also defined for $|x| > 2$:

```
>> xx=[-3 x 3]
>> FF = [0 F 1]
>> plot(xx,FF)
```

4.3 Exercises

1. Complete the exercises of section 3.5 of the lecture notes (blue booklet).
2. Consider a random variable X with probability density function, $f(x) = \frac{3}{2} - 6\left(x - \frac{1}{2}\right)^2$ for $0 \leq x \leq 1$.
   (a) Plot the density function and check that it is non-negative.
   (b) Integrate the pdf and check that the area under the graph is 1.
   (c) Find the following probabilities:
      i. $P(X \leq 0.5)$;
      ii. $P(0.2 \leq X \leq 0.7)$;
      iii. $P(X > 0.8)$.
   (d) Plot the pcf.
   (e) Find an expression for the distribution function. Create and call a function which gives the pcf and plot it again.

Lab Tutorial 5: Random variables: expectation and variance

5.1 Samples and population

First we investigate the difference between a sample and a random variable and check the statement that the sample mean converges to the expectation of a corresponding random variable and the sample variance to the variance of random variable. The command `unidrnd(3,1,10)` gives a 1×10 matrix with entries being 1, 2, or 3 and each number has the same probability of being chosen. Thus there is probability $\frac{1}{3}$ that the value 1 is recorded, probability $\frac{1}{3}$ that the value 2 is recorded and also probability $\frac{1}{3}$ that the value 3 is recorded. The random process X is the recording of the values 1, 2 and 3. And the pmf is

x	1	2	3
$P(X=x)$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

The expectation and variance of the random variable X are given by

```
EX = (1+2+3)/3
VarX= ((1-EX)^2+(2-EX)^2+(3-EX)^2)/3
```

The command below creates a vector `x10` of 10 random numbers 1, 2 or 3:

```
x10 = unidrnd(3,1,10)
```

To determine the mean and variance of the sample vector `x10`:

```
m(1) = mean(x10)
v(1) = var(x10)
```

Next we take samples of size 20, 30, $\ldots 1000$ and record the mean and variance:

```
for i=2:100
x=unidrnd(3,1,10*i);
m(i)=mean(x); v(i)=var(x);
end;
```

Finally we plot the means and variances of the samples as well as the expectation and variance of the random variable X:

```
plot(m,'.')
hold on
plot(1:100,EX)
plot(v,'.r')
plot(1:100,VarX,'r')
xlabel('Sample size/10')
hold off
```

As you can see, the blue dots for the sample mean move around the expectation of X. Most of them get closer to the expectation of X the larger the sample size gets, but exceptions keep cropping up, as you would expect for random samples. Similarly, the red dots for the sample variance move around the variance of X. Most of them get closer the larger the sample size gets, but there keep being exceptions.

5.2 Random variables

The random variable X is associated with throwing a fair 6-sided die once and recording the outcome, the random variable Y is associated with throwing a fair 4-sided die once and recording the outcome . The random variable Z is associated with throwing both the fair 6-sided die and 4-sided one. If the 4-sided die was even, we record the difference of the outcomes of the dice, otherwise we record the sum of the outcomes. We use matlab to find the pmf for Z. First we define vectors `X` and `Y` with the values for the corresponding random variables. After this, we construct a matrix `Z` which has the outcomes.

```
X = 1:1:6
Y = 1:1:4
for j=1:4 if mod(Y(j),2)==0,
             for i=1:6 Z(i,j)=X(i)-Y(j); end;
          else for i=1:6 Z(i,j)=X(i)+Y(j); end;
          end;
end;
Z                     % matrix with all outcomes
```

Next we determine the pmf. The vector `values` will contain all different values in `Z` and the vector `p` has the probability of each value.

```
values = [];
for i=1:6 values = union(values,Z(i,:)); end;
p=[];
for i=1:length(values) temp = (Z==values(i)) ;
    p(i)=sum(sum(temp))/(6*4);
end;
values               % list with all different values
p                    % probability for each value
sum(p)               % Check that the sum is 1
```

So now we conclude that the pmf is (check this!)

x	-3	-2	-1	0	1	2	3	4	5	6	7	8	9
$P(X=x)$	0.0417	0.0417	0.0833	0.0833	0.0833	0.1250	0.0833	0.1250	0.0833	0.0833	0.0833	0.0417	0.0417

Finally we determine the expectation and variance.

```
EZ = values*p'
VarZ = (values-EZ).^2*p'
% Outcomes
EZ = 3
VarZ = 10.1667
```

5.3 Exercises

1. (a) Consider a biased six-sided die where the probability of the even number is twice as high as the probability of the odd numbers. Give the probability mass function for the random variable associated with throwing this die. (Hint: call the probability of an odd number p, then the probability of an even number is $2p$. Find p.) Find the expectation and variance of the random variable.

   (b) Use the random generator `unidrnd` to generate samples to correspond to this process. One possibility could be to generate random numbers between 1 and 9. The number 1 gets linked to 1, the numbers 2,3 to 2, 4 with 3, etc. Make samples of size 100, 200, ... 10000 and compare the mean and variance of the samples with the expectation and variance of the random variable.

2. Consider three independent random processes, denoted by X, Y and Z. Their expectation and variance are:

$$E(X) = 3,\ \mathrm{Var}(X) = 2; \quad E(Y) = -2,\ \mathrm{Var}(Y) = 4; \quad E(Z) = 0,\ \mathrm{Var}(Z) = 5.$$

   Find the expectation and variance of $A = Y - 2X + 4$, $B = Z - 3X + Y - 9$, and $C = 4X - 5Y - 8Z - 7$.

3. The random variable X is associated with throwing a fair 3-sided die once and recording the outcome, the random variable Y is associated with throwing a fair 4-sided die once and recording the outcome . The random variable Z is associated with throwing both the fair 3-sided die and 4-sided one. If the 4-sided die was even, we record the product of the outcomes of the dice, otherwise we record the sum of the outcomes.

   (a) Give the pmf of X, Y and $X \cdot Y$. Find the expectation and variance of X, Y and $X \cdot Y$. Are X and Y independent random variables?

   (b) Find the pmf of Z and its expectation and variance.

   (c) Are $X \cdot Y$ and $X + Y$ independent?

   (d) Are Z and $X \cdot Y$ independent?

   (e) Are Z and $X + Y$ independent?

Lab Tutorial 6 - The Binomial Distribution

5.1 Binomial Probabilities

We start by plotting the binomial probability mass function for three binomial distributions namely $B(10, 0.25)$, $B(10, 0.50)$ and $B(10, 0.75)$. First split the graphics window into three equal areas use the `subplot` command. Then use the `binopdf` to calculate the pmfs as shown:

```
>> subplot(3,1,1); plot(0:10,binopdf(0:10,10,0.25))
>> subplot(3,1,2): plot(0:10,binopdf(0:10,10,0.50))
>> subplot(3,1,3); plot(0:10,binopdf(0:10,10,0.75)).
```

How do the three plots differ? Why?

Similarly the `binocdf` calculates the cumulative distribution functions for the binomial distributions. To plot the cumulative distribution functions of $B(10, 0.25)$, $B(10, 0.50)$ and $B(10, 0.75)$ use the following:

```
>> subplot(3,1,1); plot(0:10,binocdf(0:10,10,0.25))
>> subplot(3,1,2): plot(0:10,binocdf(0:10,10,0.50))
>> subplot(3,1,3); plot(0:10,binocdf(0:10,10,0.75)).
```

How do these plots differ? Why?

Now using commands such as

```
>> plot(0:20,dbinom(0:20,20,0.25))
>> plot(0:20,pbinom(0:20,20,0.25))
```

to produce plots of the probability mass functions and the cumulative distribution functions for the distributions $B(20, 0.25)$, $B(20, 0.50)$ and $B(20, 0.75)$.

We can calculate specific probabilities of a random variable with a binomial distribution having a certain outcome. For example, for $X \sim B(10, 0.9)$, we use `binopdf(6,10,0.9)` to calculate $P(X = 6)$ and `binocdf(6,10,0.9)` to calculate $P(X <= 6)$. Note that to calculate $P(X < 6)$, we use `binocdf(5,10,0.9)` as $P(X < 6) = P(X \leq 5)$. Thus in matlab:

```
>> binopdf(6,10,0.9)     % P(X = 6)
>> binocdf(6,10,0.9)     % P(X <= 6)
>> binocdf(5,10,0.9)     % P(X < 6)
```

5.2 Random Samples From Binomial Distributions

The function `binornd` generates random data from a Binomial distribution. For example,

```
>> sample = binornd(20,0.6,10,1)
```

will generate a 10×1 matrix of random samples from a $B(20, 0.6)$ distribution and store it in the variable *sample*. Having generated a random sample, we proceed by producing a graphical representation of the data using:

```
>> hist(sample)
```

and then calculate the mean and variance of `sample` with the commands:

```
>> mean(sample)
>> var(sample)
```

Now use similar commands to produce a random sample of size 50 from $PB(20, 0.6)$. Call the vector containing the random sample `sample50`. Produce a histogram of the sample and find the mean and variance.

In the same way, now generate random samples of sizes 100 and 1000. The mean of the $B(20, 0.6)$ distribution is $20 \times 0.6 = 12$. The variance is $20 \times 0.6 \times 0.4 = 4.8$. The mean and variance of the samples tend to get closer to the distribution mean and variance as the sample size increases.

Next, generate random samples of various sizes from a $B(10, 0.3)$ distribution. Calculate the mean and variance of each sample and compare with the theoretical values.

5.3 Exercises

1. Calculate the following for a $B(12, 0.35)$ distribution:
   (a) $P(X = 4)$;
   (b) $P(X = 7)$;
   (c) $P(X \leq 5)$;
   (d) $P(X > 6)$;
   (e) $P(X \leq 7)$;
   (f) $P(2 < X \leq 5)$.
2. A commuter travels to work by train. The train is late with probability 0.15. Over the next 4 weeks (ie 20 work days) calculate the probability that the train is late:
   (a) exactly 3 times;
   (b) no more than 5 times;
   (c) exactly 5 times;
   (d) no more than 8 times.
3. A multiple choice paper contains 20 questions and each question has 5 possible answers. We want to determine the probabilities of obtaining different numbers of correct answers if the questions are answered entirely by guesswork. Produce a matrix for which:

   - first column = number of correct answers obtained by guesswork;
   - second column = probability mass function;
   - third column = distribution function,

   by using the following sequence of commands:

   ```
   >> correct=0:20;
   >> prob=dbinom(0:20,20,0.2);
   >> dist=pbinom(0:20,20,0.2);
   >> M=[correct;prob;dist];
   ```

   From the matrix M we can read off probabilities. For example the probability of scoring 3 correct answers is 0.2054 and the probability of scoring 9 or fewer is 0.9974.

   (a) Suppose that a pass is obtained by scoring 10 or more correct answers. What is the probability of passing the paper by guesswork alone?
   (b) An alternative paper is suggested by the board of examiners. This paper has 30 questions with four possible answers to each question. A pass is obtained if 15 or more correct answers are given. A distinction is gained for 25 or more correct answers.
      i. What is the probability of passing this paper by guesswork alone?
      ii. What is the probability of gaining a distinction by guesswork?

4. Generate a random sample of size m from a $B(n, \theta)$ distribution, and compare sample mean and variance to that of the distribution, for
   (a) $m = 10\,,\ n = 20\,,\ \theta = 0.4\,;$
   (b) $m = 20\,,\ n = 15\,,\ \theta = 0.2\,;$
   (c) $m = 100\,,\ n = 20\,,\ \theta = 0.3\,.$
5. Answer questions 1 and 4 in section 4.11 of the blue booklet.

Lab Tutorial 7: The Poisson Distribution

7.1 Poisson Probabilities

We will start by plotting the Poisson probability mass function and distribution function for each of the Poisson distributions $Po(5)$, $Po(10)$ and $Po(15)$. To view all six plots together we will use the `subplot` command. The command for the probability mass function is `poisspdf` and the command for the distribution function is `poisscdf`. The plot for the probability mass function for $Po(5)$ can be obtained using the command:

```
>>    subplot(3,2,1); plot(0:30,poisspdf(0:30,5))
```

Note that this command produces a plot giving the 31 probabilities $P(X = 0)$, $P(X = 1), \ldots, P(X = 30)$ where X is a random variable with a $Po(5)$ distribution. Similarly, the plot for the distribution function for $Po(5)$ can be obtained using the command:

```
>> subplot(3,2,2); plot(0:30,poisscdf(0:30,5))
```

Note that this command produces a plot giving the 31 probabilities $P(X = 0)$, $P(X \leq 1)$, $P(X \leq 2), \ldots, P(X \leq 30)$ where X is a random variable with a $Po(5)$ distribution.

The corresponding plots for the $Po(10)$ and $Po(15)$ distributions can be obtained using the following commands:

```
>> subplot(3,2,3); plot(0:30,poisspdf(0:30,10))
>> subplot(3,2,4); plot(0:30,poisscdf(0:30,10))
>> subplot(3,2,5); plot(0:30,poisspdf(0:30,15))
>> subplot(3,2,6); plot(0:30,poisscdf(0:30,15))
```

Note that although Poisson random variables can take values larger than 30, for the three distributions chosen, the probabilities of values larger than 30 are negligible.

If you want to keep this figure on screen and also plot some more graphs, use the command

```
>> figure;
```

to start a new figure window, while keeping the original one.

We can calculate specific probabilities of a random variable with a Poisson distribution having a certain outcome. For example, if you wish to calculate $P(X = 6)$ for a $Po(5)$ distribution use `poisspdf(6,5)`. To calculate the probability of getting a 7 or an 8 from a random variable with a $Po(5)$ distribution use `poisspdf(7,5)+poisspdf(8,5)`. Similarly, for $P(X \leq 6)$ use `poisscdf(6,5)`, for $P(X < 4)$ use `poisscdf(3,5)` and for $P(X > 7)$ use `1-poisscdf(7,5)`.

7.2 Random Samples From Poisson Distributions

Random samples can be generated from poisson distributions using the `poissrnd` command. Use the following commands to generate a random sample of size 10×1 from a $Po(5)$ distribution called `sample`. We also produce a histogram and calculate the mean and variance of `sample`.

```
>> sample=poissrnd(5,10,1);
>> hist(sample)
>> mean(sample)
>> var(sample)
```

Generate random samples of sizes 100 and 1000. Note that for the Poisson distribution with parameter 5 the mean and variance are both 5. How do the sample values compare with the distribution values of mean and variance?

Next, generate random samples of various sizes from a Poisson distribution with parameter 14. Calculate the mean and variance of each sample and compare.

7.3 Dependent and indepent events

Next we explore the difference between $3X$ and $X+X+X$. In $3X$, we add the same event X to itself three times. In $X+X+X$, we add three independent events, all with the same distribution. Below we define three samples from the same poisson distribution $X \sim Po(7)$, called `X1, X2, X3`. Next we define `Y` such that it is a sample of the distribution $3X$ and `Z` to be a sample of the distribution $X+X+X$.

```
>> X1 = poissrnd(7,400,1);
>> X2 = poissrnd(7,400,1);
>> X3 = poissrnd(7,400,1);
>> Y = 3*X1;
>> Z = X1+X2+X3;
```

Next we plot the histograms of `Y` and `Z` in one figure, while keeping the previous figure on screen.

```
figure;
set(gca,'Fontsize',18)
subplot(2,1,1);hist(Y);
xlabel('Outcome of Y'); ylabel('Frequency');
title('Histogram of Y');
subplot(2,1,2);hist(Z);
xlabel('Outcome of Z'); ylabel('Frequency');
title('Histogram of Z');
```

From the theory, we know that if A and B are independent and $A \sim Po(\lambda)$ and $B \sim Po(\mu)$, then $A+B \sim Po(\lambda+\mu)$. Thus $X+X+X \sim Po(3 \cdot 7) = Po(21)$, but we do not expect that $3X$ has this property. Since `X1, X2, X3` are independent, we expect that Z is a sample of a $Po(21)$ distiribution. But we do not expect Y to have this property. To check this, we replot the histograms and put the scaled pdf of $Po(21)$ on top of it.

```
subplot(2,1,1);[n,xout] = hist(Y);
maxY=max(Y); maxhist=max(n);
bar(xout,n);
hold on;
xlabel('Outcomes'); ylabel('Frequency');
title('Analysis of Y');
PY=poisspdf(1:maxY,20);PYmax=max(PY);
plot(1:maxY,PY*maxhist/PYmax,'r');
hold off;
subplot(2,1,2);[n,xout] = hist(Z);
maxZ=max(Z); maxhist=max(n);
bar(xout,n);
hold on;
xlabel('Outcomes'); ylabel('Frequency');
title('Analysis of Z');
PZ=poisspdf(1:maxZ,20);PZmax=max(PZ);
plot(1:maxZ,PZ*maxhist/PZmax,'r');
hold off;
```

As you can see, in case of `Z`, there is a reasonable correspondance between the poisson curve and the sample, while in case of `Y`, there is quite a difference. So this illustrates the difference between dependent and independent distributions and that it is really necessary to have independent distributions to have that $A+B \sim Po(\lambda+\mu)$ if $A \sim Po(\lambda)$ and $B \sim Po(\mu)$.

7.4 Binomial distribution and Poisson distribution

Next we visually check the difference between the probability mass function of a binomial distribution $B(n, \theta)$ and the pmf for a Poisson distribution $Po(n \cdot \theta)$. We should see that the difference gets small if n is large and θ is small. First we keep $\theta = 0.1$ fixed and increase n:

```
theta=0.1;n=10;
B10 = binopdf(1:n,n,theta);
P10 = poisspdf(1:n,n*theta);
subplot(4,1,1); plot(1:n,B10,'*',1:n,P10,'ro');
title('\theta=0.1, n=10')
n=50;
B50 = binopdf(0:n,n,theta);
P50 = poisspdf(0:n,n*theta);
subplot(4,1,2); plot(0:n,B50,'*',1:n,P50,'ro');
title('\theta=0.1, n=50')
n=100;
B100 = binopdf(0:n,n,theta);
P100 = poisspdf(0:n,n*theta);
subplot(4,1,3); plot(0:n,B100,'*',1:n,P100,'ro');
title('\theta=0.1, n=100')
n=500;
B500 = binopdf(0:n,n,theta);
P500 = poisspdf(0:n,n*theta);
subplot(4,1,4); plot(0:n,B500,'*',1:n,P500,'ro');
title('\theta=0.1, n=500')
```

Finally we plot the errors:

```
figure;
subplot(4,1,1); plot(0:7,P10(0:7)-B10(0:7));
title('Errors for n=10, \theta=0.1'); ylabel('Error')
subplot(4,1,2); plot(0:15,P50(0:15)-B50(0:15));
title('Errors for n=50, \theta=0.1'); ylabel('Error')
subplot(4,1,3); plot(0:25,P100(0:25)-B100(0:25));
title('Errors for n=100, \theta=0.1'); ylabel('Error')
subplot(4,1,4); plot(25:75,P10(25:75)-B10(25:75));
title('Errors for n=500, \theta=0.1'); ylabel('Error')
```

Note that we didn't plot the full range as the errors are getting really small, so no more information can be read off in the outer region(s).

Next we keep $n = 500$ fixed and decrease θ;

```
figure;
n=500;theta=0.8
B8 = binopdf(0:n,n,theta);
P8 = poisspdf(0:n,n*theta);
subplot(4,1,1); plot(0:n,B8,'*',1:n,P8,'ro');
theta=0.5;
B5 = binopdf(0:n,n,theta);
P5 = poisspdf(0:n,n*theta);
subplot(4,1,2); plot(1:n,B5,'*',1:n,P5,'ro');
theta=0.2;
B2 = binopdf(0:n,n,theta);
```

```
P2 = poisspdf(0:n,n*theta);
subplot(4,1,3); plot(0:n,B2,'*',1:n,P2,'ro');
theta=0.1;
B1 = binopdf(0:n,n,theta);
P1 = poisspdf(0:n,n*theta);
subplot(4,1,4); plot(0:n,B1,'*',1:n,P1,'ro');
```

Make an error plot yourselves.

7.5 Exercises

1. Suppose that a random variable Y has a $Po(6)$ distribution. Calculate the following probabilities:
   (a) $P(Y = 0)$;
   (b) $P(Y = 8)$;
   (c) $P(Y \leq 3)$;
   (d) $P(Y \geq 4)$;
   (e) $P(5 \leq Y < 7)$;
2. The number of vehicles passing a checkpoint per hour can be modelled by a Poisson random variable with parameter 18.
   (a) What is the probability that exactly 17 cars pass the checkpoint in an hour?
   (b) What is the probability that 18 or 19 cars pass the checkpoint in an hour?
   (c) What is the probability that 15 cars or fewer pass the checkpoint in an hour?
   (d) What is the probability that less than 15 cars pass the checkpoint in an hour?
   (e) What is the probability that more than 20 cars pass the checkpoint in an hour?
3. Generate a random sample of size m from a $Po(\lambda)$ distribution, and compare sample mean and variance to that of the distribution, for
   (a) $m = 10,\ \lambda = 0.4$;
   (b) $m = 20,\ \lambda = 0.2$;
   (c) $m = 100,\ \lambda = 0.3$.
4. Answer questions 2 and 5 in section 4.10 of the blue booklet.

Lab Tutorial 8: The Normal Distribution

8.1 Plotting and calculations

We will start by plotting the Normal probability density function for each of the four Normal distributions $N(25,36)$, $N(25,9)$, $N(25,81)$ and $N(25,225)$. To view all four plots together use the `hold` command to enable easy comparison to be made between the shapes of the distributions. We also set the ranges for x and y axes to be the same for each plot.

The command for the probability density function is `normpdf`. The four plots can be obtained using the following commands (notice that in MATLAB the parameters are the mean and the standard deviation rather than the mean and the variance - so, for example in the first plot we use the parameters 25 and 6 rather than 25 and 36):

```
>> x=-20:70;
>> hold on
>> plot(x,normpdf(x,25,6)); set(gca,'XLim',[-20,70],'YLim',[0,0.14]);
>> plot(x,normpdf(x,25,3)); set(gca,'XLim',[-20,70],'YLim',[0,0.14]);
>> plot(x,normpdf(x,25,9)); set(gca,'XLim',[-20,70],'YLim',[0,0.14]);
>> plot(x,normpdf(x,25,15)); set(gca,'XLim',[-20,70],'YLim',[0,0.14]);
>> hold off
```

Alternatively, the following commands will plot them all at once in different colours:

```
>> x=-20:70
>> N1=normpdf(x,25,6); N2=normpdf(x,25,3); N3=normpdf(x,25,9); N4=normpdf(x,25,15);
>> plot(x,N1,x,N2,x,N3,x,N4)
>> legend('N1','N2','N3','N4')
```

Note that each distribution has its peak at 25 but that larger variances give rise to flatter distributions and that smaller variances give rise to 'taller distributions.

We can calculate specific probabilities. For example, to calculate the probability mass function $f(18)$ for a $N(25,36)$ distribution, we use `normpdf(18,25,6)`. To calculate $P(X \leq 18)$ for a $N(25,36)$ distribution use `normcdf(18,25,6)`. To calculate $P(20 < X < 27)$ for a $N(25,9)$ distribution, we use `normcdf(27,25,3)-normcdf(20,25,3)`. To calculate $P(X > 18|X > 15)$ for a $N(25,81)$ distribution use `(1-normcdf(18,25,9))/(1-normcdf(15,25,9))`. Note that for a continuous distribution like the normal distribution, symbols $<$ and $\leq$ can be interchanged and similarly for $>$ and $\geq$. So, for example, $P(X < 7)$ is the same as $P(X \leq 7)$ where X has a normal distribution.

8.1.1 Exercise

The weight of a species of bat can be modelled by a normal random variable with mean 30 and variance 49.

1. Plot the probability density function for the weights of the population of bats.
2. A bat is chosen at random. What is the probability that the bat is lighter than 26g?
3. What is the probability that the bat is heavier than 40g?
4. What is the probability that the bats weight is between 25g and 35g?
5. Given that the bat is lighter than 22g, what is the probability that it is lighter than 17g?

8.2 Random Samples From Normal Distributions

The function `normrnd` generates random data from a Normal distribution. Generate random samples of size 800 from normal distributions N(5,36) and N(17,64) using the commands:

```
>> y1=normrnd(5,6,1,800);
>> y2=normrnd(17,8,1,800);
```

Calculate the mean, standard deviances and variances of those samples (the means should be close to 5 and 17 and the variances should be close to 36 and 64) by using:

```
mean(y1)
std(y1)
var(y1)
mean(y2)
std(y2)
var(y2)
```

Now create a new vector of random observations by adding elements of `y1` and `y2` in pairs:

```
y3 = y1+y2;
mean(y3)
std(y3)
var(y3)
```

Finally in one window (use the `subplot` command) produce a plot of the probability density function for the distribution N(22,100) on the top and a histogram of the random sample in `y3` the bottom. The shapes should be similar. Can you explain why this is so? To visualise the relation more clearly, we will scale the normal distribution pdf and plot it on top of the `y3` histogram.

```
figure;
[n,yout]=hist(y3);
freq_max=max(n);
normal_max = normpdf(22,22,10);  % Remember that a normal distribution
                                % is symmetric about the mean, hence
                                % the maximum is at the mean
hist(y3);
hold on;
spacing = yout(2)-yout(1);
yy = yout(1)-spacing/2 : (yout(end)-yout(1)+spacing)/100 : yout(end)+spacing/2;
plot(yy,normpdf(yy,22,10)*freq_max/normal_max,'r-');
ylabel('Frequency');
legend('y3','N(22,100)');
hold off
```

8.3 The Central Limit Theorem

To see how the central limit theorem works, we will look at some random samples of means of various distributions.

8.3.1 Means of binomial distribitions

First we consider the binomial distribution. We will make a list with 100 numbers called `binomset`. This numbers in this list are the means of 200 observations from a $B(80, 0.25)$ distribution. The list is made as follows

```
binomset = [];
for i=1:100
  binomset = [binomset,mean(binornd(80,0.25,1,200))];
end;
```

The binomial distribution $B(80, 0.25)$ has mean $\mu = 80 \cdot 0.25 = 20$ and variance $\sigma^2 = 80 \cdot 0.25 \cdot 0.75 = 15$. According to the central limit theorem, the distribution of the means should be like a normal distribution $N(20, 15/200)$. First we check the mean and variance are close to the predicted ones

```
mu= mean(binomset)              % mean and standard deviation of binomset
sigma = std(binomset)
sigma^2
```

Are the means and variance of binomset indeed close to 20 respectively 15/200?

Next we plot a histogram of set `binomset` and a scaled normal distribution in the same graph. The scaling is such that the top of the histogram (which we will call top) coincides with the top of the normal distribution. We will plot the normal distribution for the same values are the histogram is plotted, i.e., the values in `binomset`.

```
[n,bino_out] = hist(binomset);     % get the input for the histogram
top = max(n);                      % find the maximal frequency
spacing = bino_out(2)-bino_out(1); % find width of the bins
xmin = bino_out(1)-spacing/2;
xmax = bino_out(end)+spacing/2;
xx = xmin:(xmax-xmin)/100:xmax;
figure;
hist(binomset);
hold on;
plot(xx,normpdf(xx,20,sqrt(15/200))*top/normpdf(20,20,sqrt(15/200)),'r-');
plot(xx,normpdf(xx,mu,sigma)*top/normpdf(mu,mu,sigma),'g-');
legend('binomset','N(20,15/200)','N(\mu,\sigma^2)');
hold off
```

Another way to check if the means indeed satisfy this normal distribution is by using some commands we have seen in the beginning of the course. We plot the theoretical quantiles of the normal distribution against the sample quartiles

```
figure;
normplot(binomset)
```

8.3.2 Means of uniform distribitions

Next we consider the uniform distribution. First again the list of means of random samples of U[70,150] and a histogram:

```
uniset = []
for i = 1:100
  uniset = [uniset,mean(unifrnd(70,150,1,200))];
end;
mu= mean(uniset)
sigma = std(uniset)
var(uniset)
mean_expect = (150+70)/2
var_expect = (150-70)^2/12/200
```

Are the mean and variance as expected? Next we plot the figures

```
[n,uni_out] = hist(uniset)
top = max(n);                          % find the maximal frequency
spacing = uni_out(2)-uni_out(1);   % find width of the bins
xmin = uni_out(1)-spacing/2;
xmax = uni_out(end)+spacing/2;
xx = xmin:(xmax-xmin)/100:xmax;
figure;
hist(uniset);
hold on;
plot(xx,unifpdf(xx,110,sqrt(8/3))*top/unifpdf(110,110,sqrt(8/3)),'r-');
plot(xx,normpdf(xx,mu,sigma)*top/normpdf(mu,mu,sigma),'g-');
legend('binomset','N(110,8/3)','N(\mu,\sigma^2)');
hold off
```

Finally, the theoretical quantiles of the normal distribution against the sample quartiles

```
figure;
normplot(uniset)
```

8.4 Exercises

1. Repeat the steps above with means of random samples of a Po(50) distribution distribution.
2. Repeat the steps, using 500 samples instead of 200 samples for all three distributions.

Lab Tutorial 9: Regression modelling

9.1 Fitting Linear Models: Linear Regression

In this section, we will fit a straight line model to paired data using the method of least squares. First load the data set 'peanuts' from the shared drive

\\shared.surrey.ac.uk\Shared_Labs\NetworkExamples

by the command

```
load peanuts
```

This gives a vector `X` with average levels of alfatoxin of a batch of peanuts and a vector `Y` with the corresponding percentage of non-contaminated peanuts in this batch.

Plot the data by using the command `scatter` and find the correlations and rank correlation of this data using

```
corr(X,Y)
corr(X,Y,'Type','Spearman')
```

Is the correlation positive or negative?

Next we fit a straight line model of the form

$$Y = a + b\,X.$$

Hence X is the explanatory variable and Y the response variable.

```
myfit = polyfit(X,Y,1)
```

Note that the order of the vectors is important, the first vector is the explanatory variable, the second one the response variable. The third entry (here the number 1) is the order of the polynomial that we are trying to fit. Here we fit a first order polynomial.

The result of polyfit are two numbers. The first number is the slope b and the second number the intercept a. Here we get

```
-0.0029  100.0021
```

Thus the linear model is

$$Y = 100.0021 - 0.0029\,X.$$

We plot this line on top of the scatter data:

```
scatter(X,Y)
hold on
Y_approx = polyval(myfit,X);
plot(X,Y_approx)
hold off
```

To check that the assumptions made in fitting the model are reasonable we need to carry out two checks:

1. Check that the residuals are randomly scattered with no obvious relationship to the explanatory variable.
2. Check that the residuals are approximately normally distributed, as we assumed that the noise is normally distributed.

```
residuals = Y-Y_approx;
mean(residuals), var(residuals)
figure
scatter(X,residuals)
hold on
plot(X,zeros(length(X)))
figure
qqplot(residuals)
```

Check that the scatter plots doesn't show an obvious relation between the explanatory variable `X` and the `residuals`. Furthermore verify that the qqplot show the points close to the straight line.

9.2 Quadratic

In this section we look at fitting a quadratic curve to date. First we load the data set `anaerob` from the shared drive.

```
load anaerob
```

This data set is related to someone performing an exercise, gradually increasing the level of effort. The vector `x` has the oxygen uptake, the vector `y` the expired ventilation. Plot the data by using the command `scatter` and find the correlations and rank correlation of this data using

```
corr(x,y)
corr(x,y,'Type','Spearman')
```

Is the correlation positive or negative? As you can see, the data looks like a quadratic curve, so we will fit a quadratic model of the form

$$y = a + b\,x + c\,x^2.$$

Hence x is the explanatory variable and y the response variable.

```
myfit2 = polyfit(x,y,2)
```

The result gives three numbers, the first is c, the second b and the last one a. It seems that $c = 0$, which suggest that a straight line is fitted. However, if you look at the values on the x and y axis, you see that the x values are of order 10^3, while the y values are of order $10 - 10^2$. So we should ask matlab to give the outcomes in the long format

```
format long
myfit2
```

Now we get

```
0.000008902016323  -0.013441200930183  24.270395202631583
```

Thus the quadratic model is

$$y = 24.27 - 1.344\,10^{-2}\,x + 8.902\,10^{-6}\,x^2$$

We plot this line on top of the scatter data:

```
scatter(x,y)
hold on
y_approx = polyval(myfit2,x);
plot(x,y_approx)
hold off
```

To check that the assumptions made in fitting the model are reasonable we need to carry out the two checks that the residuals don't have an obvious relationship with x and that they are normally distributed.

```
residuals = y-y_approx;
mean(residuals), var(residuals)
figure
scatter(x,residuals)
hold on
plot(x,zeros(length(x)))
figure
qqplot(residuals)
```

Check that the scatter plots doesn't show an obvious relation between the explanatory variable `x` and the `residuals`. Furthermore verify that the qqplot show the points close to the straight line.

9.3 Multiple regression

Finally, we consider a data set with two explanatory variables and one response variable.

Load the `crab` data set (on the shared drive). The matrix `X` contains the explanatory variables: carapace width (cm) and weight (kg). The response variable is the number of male crabs (satellites) surrounding the nesting female/male couple. This is saved in the `satellites` vector.

We will fit a linear multiple regression model

$$\texttt{satellites} = a + b\,\texttt{carapace\_width} + c\,\texttt{weight}.$$

```
load crab
scatter3(X(:,1),X(:,2),satellites); hold on;
XX=[ones(length(satellites),1),X(:,1),X(:,2)];
r=regress(satellites,XX)
```

This gives three numbers, the first one is a, the second b and the last is c.

$$\texttt{satellites} = -3.5885 + 0.0868\,\texttt{carapace\_width} + 1.7332\,\texttt{weight}.$$

Next we will plot the resulting plane in our scatter plot.

```
xfit = min(X(:,1)):(max(X(:,1))-min(X(:,1)))/100:max(X(:,1));
yfit = min(X(:,2)):(max(X(:,2))-min(X(:,2)))/100:max(X(:,2));
[Xfit,Yfit] = meshgrid(xfit,yfit);
Zfit = r(1) + r(2)*Xfit + r(3)*Yfit;
mesh(Xfit,Yfit,Zfit)
xlabel('Carapace width')
ylabel('Weight')
zlabel('Satellites')
hold off;
```

To compare the model with the observed data, we look at

```
prediction = r(1)+r(2)*X(:,1)+r(3)*X(:,2);
figure
scatter(satellites,prediction)
hold on
m1 = max(min(satellites),min(prediction));
M1 = min(max(satellites),max(prediction));
```

```
plot([m1,M1],[m1,M1])
xlabel('Observed satellites')
ylabel('Predicted satellites')
hold off
```

Finally we consider the residuals:

```
residuals = satellites-prediction;
figure
scatter(satellites, residuals)
plot([min(satellites),max(satellites)],[0,0])
figure
qqplot(residuals)
```

Note that both plots indicate a bias in the system, hence the linear regression is not appropriate in this case.

9.4 Exercises

1. Load the `prestige` data set (on the shared drive). Fit a linear and quadratic model for the data in `education` and `prestige` vectors.

Lab Tutorial 10: Time Series

In this practical session you will learn the method for entering a set of time series data into MATLAB and for examining this time series using MATLAB. The data we are examining comprise a single time series of quarterly sales of a company in agricultural markets over a period of 12 years. Quarters are labelled Qtr., running from 1 to 4 within each year, with Year running from 1973 to 1984. The data are contained in the table below:

Qtr\Yr	73	74	75	76	77	78	79	80	81	82	83	84
1	8.48	8.94	9.20	9.13	9.23	9.49	9.37	9.56	9.71	9.72	9.82	10.11
2	8.70	8.86	9.11	9.23	9.21	9.54	9.66	9.98	9.60	9.88	9.90	9.90
3	8.09	8.45	8.69	8.65	8.68	9.06	9.03	9.19	9.18	9.11	8.87	9.47
4	8.58	9.00	8.87	8.84	9.20	9.35	9.44	9.50	9.53	9.49	9.38	9.47

This data can also be found in the file sales.txt on the website and shared drive

\\shared.surrey.ac.uk\Shared_Labs\NetworkExamples

Import this data by using "Import data": click on $\boxed{\text{File}} \mapsto \boxed{\text{Import Data}}$ and select the downloaded file. MATLAB will recognize the space-separated data, so simply click on Next. Then click on $\boxed{\text{Finish}}$. The data is in a matrix `sales`, check it on your screen. In this matrix, the columns representing Quarters and the rows represent years. To manipulate the data, we first put it into one long vector, reading column by column. And we define a vector for the quarters.

```
Sales = sales(:,1);
for i = 2:length(sales)
    Sales = [Sales;sales(:,i)];
end;
quarters=[1973:1/4:1985-1/4];
```

A plot of the time series data is obtained using the following commands:

```
plot(quarters,Sales,'o-');
set(gca,'FontSize',14);
ylabel('Sales'); xlabel('Quarters');
set(gca,'Xlim',[1973,1985])
title('Time Series of Quarterly Sales 1973-1984');
```

10.1 The autocorrelation function (ACF)

To get the ACF, we use the following commands

```
SALES(:,1)=Sales;
for i=2:length(Sales)
    SALES(:,i)=[Sales(i:end);Sales(1:(i-1))];
end;
acorr=corr(SALES);
figure;
bar([0:length(acorr(:,1))-1],acorr(:,1),0.2);
set(gca,'FontSize',14);
title('ACF for Quarterly Sales 1973-1984');
xlabel('Lag');ylabel('ACF');
set(gca,'XLim',[-1,25])
```

Thus the j-th column of the matrix `SALES` has the data for the $j-1$-th lag. The first column of the matrix `acorr` gives the ACF. To plot this, we have to remember that lag i is in the $i+1$-st entry of the ACF function.

Both from the plot of the time series and the ACF plot, we can tell that the series is highly seasonal, following the natural annual pattern of farming activities.

10.2 The trend

To find the model, we are first going to find the trend by using smoothing to remove the seasonal features. For this quarterly data a centred 4-point moving average is appropriate.

```
mask = ones(5,1)/4; mask(1)=mask(1)/2; mask(end)=mask(end)/2;
salesMAtemp = filter(mask,1,Sales);
salesMA = salesMAtemp(5:end);     % first four values are not relevant
quartersMA = quarters(3:end-2);     % remove quarters for which there is no MA
figure;
plot(quartersMA,salesMA,'r-');
hold on;
plot(quarters,Sales,'o-');
title('Orginal time series and centered 4-point average for Sales data')
xlabel('Quarters');
ylabel('Sales')
set(gca,'XLim',[1973,1985])
```

Does this look as a trend, i.e., fairly smooth but with some features?

10.3 The seasonal effect

In the following the series is decomposed into an underlying level (trend) and the seasonal component, consisting of 4 quarterly effects. First we detrend the data (take out the trend)

```
sales_detrend=Sales(3:end-2)-salesMA;
```

This vector has the seasonal component, but on average, it doesnt go up or down too much. Note that this vector starts with the 3rd quarter in 1973.

In order to work out the monthly effects (i.e. the S_t values), we take the mean of the detrended values for each month in the MATLAB commands below and store them in a MATLAB vector called `sales_monthly`.

```
for i=1:4
    sales_offset(i)=mean(sales_detrend(i:4:end));
end;
sales_monthly=[sales_offset(3:4),sales_offset(1:2)]
```

This should give

$$0.20680.2347 - 0.3935 - 0.0457$$

The following plot gives the estimated quarterly effects.

```
figure;
plot(1:4,sales_monthly,'o-');
hold on;
plot([1,4],[0,0],'g');
title('Average Monthly Effects for Sales');
xlabel('Quarter index'); ylabel('Seasonal sales');
set(gca,'XLim',[0,5])
```

Does this seem reasonable compared to the quarters that they represent?

10.4 Residuals

Next construct and examine the residuals, which should look like a random sample from a normal distribution.

```
for i=1:length(sales_detrend)
    sales_resids(i) = sales_detrend(i) - sales_offset(mod(i-1,4)+1);
end;
figure;
subplot(3,1,1);scatter(1:length(sales_resids),sales_resids);
hold on; plot([1,length(sales_resids)],[0,0],'g');
title('Plot of Residuals');
xlabel('Quarters'); ylabel('Residuals');
subplot(3,1,2);hist(sales_resids);
title('Histogram of Residuals');
xlabel('Residuals'); ylabel('Frequency');
subplot(3,1,3); qqplot(sales_resids);
title('QQplot of Residuals');
```

10.5 Checking the model

Finally, to judge the adequacy of the model, a plot of the fitted data superimposed onto the original data can be obtained, the fitted data being the sum of the underlying level and the quarterly effects.

```
sales_fitted = salesMA' + repmat(sales_offset,1,length(salesMA)/4);
figure
plot(quarters, Sales); hold on
plot(quartersMA,sales_fitted,'r--');
legend('observed sales data','model sales values','location','NorthWest');
title('Raw Data versus Model');
xlabel('Quarters');
ylabel('Sales');
set(gca,'XLim',[1973,1985])
```

Lab Tutorial 11: Test and MapleTA

Today's practice test and the test (worth 60% of the marks) will both be carried out using MapleTA. To access MapleTA and the test related to this module in the Austen Pearce computer labs, do the following:

- Open a web browser and type in the web address
  `http://mapleta6.eps.surrey.ac.uk`
  You might want to add this to your favorites for easy access in future.
- Log on to MapleTA using the same login name and password as you used to log on to the computer.
- Click on Find classes open for registration. (*In order to see this link, you may need to first click on the words Maple TA at the top left of the screen or refresh the page*). You have to do this only once and you can skip this step in future.
  - Tick the box next to Maths Modelling for Computing 2 and then click on Register.
  - Click to confirm that you do want to register on "Mathematical Methods for Computing 2".
- You should now have a list of modules for which you are registered. For most people, the only module that appears is "Mathematical Methods for Computing 2". Click on Mathematical Methods for Computing 2
- You should now see a menu that includes the item Lab Practice Test 2010. Don't click on this yet.

11.1 The Lab session practice test

Some information about the Lab session practice test.

- There are 16 practice questions to answer. To make it as much as possible as the test, you may use pen, paper and calculator but NOT notes, books or MATLAB.
- Answer the questions by typing a numerical answer in the box, or selecting the correct answer in multiple choice questions. Click on Next at the top of the page to move on to the next question. Previous questions can be viewed using the Back and Question number buttons.
- When you have answered all the questions, click on Grade. This will end your tests and you can not go back to the questions anymore, so only do this when you are really finished. If you have left some questions unanswered the computer will tell you this. You can either go back and answer these questions or, if you are happy to leave the questions unanswered, click on the Grade button a second time.
- You will get a mark. The grade is automatically stored, but this test is just for practice and the mark does not count towards any assessment. By clicking on View details you will be able to see the questions and your answers with some guidance.
- When you have finished looking at your results, click on Quit and Save at the top of the page. *You must click on the* Grade *button BEFORE clicking on* Quit and Save.
- Click on Logout to leave MapleTA.

Now you are ready to start the practice test, just click on Lab Practice Test 2010.

11.2 Final test

The final test will be similar to the lab practice test, but longer and timed. You will have 90 minutes to answer the questions. To help you prepare for this, a mock test will be available from the beginning of January. You will be informed by email when it is available.

You can take the mock test as often as you want, though it is best to do it after you have done a good amount of revision. Some questions will change, so you will not be taking the same test twice. *If you want to access the mock test off-campus, you have to go to* `http://remote.surrey.ac.uk` *first.*