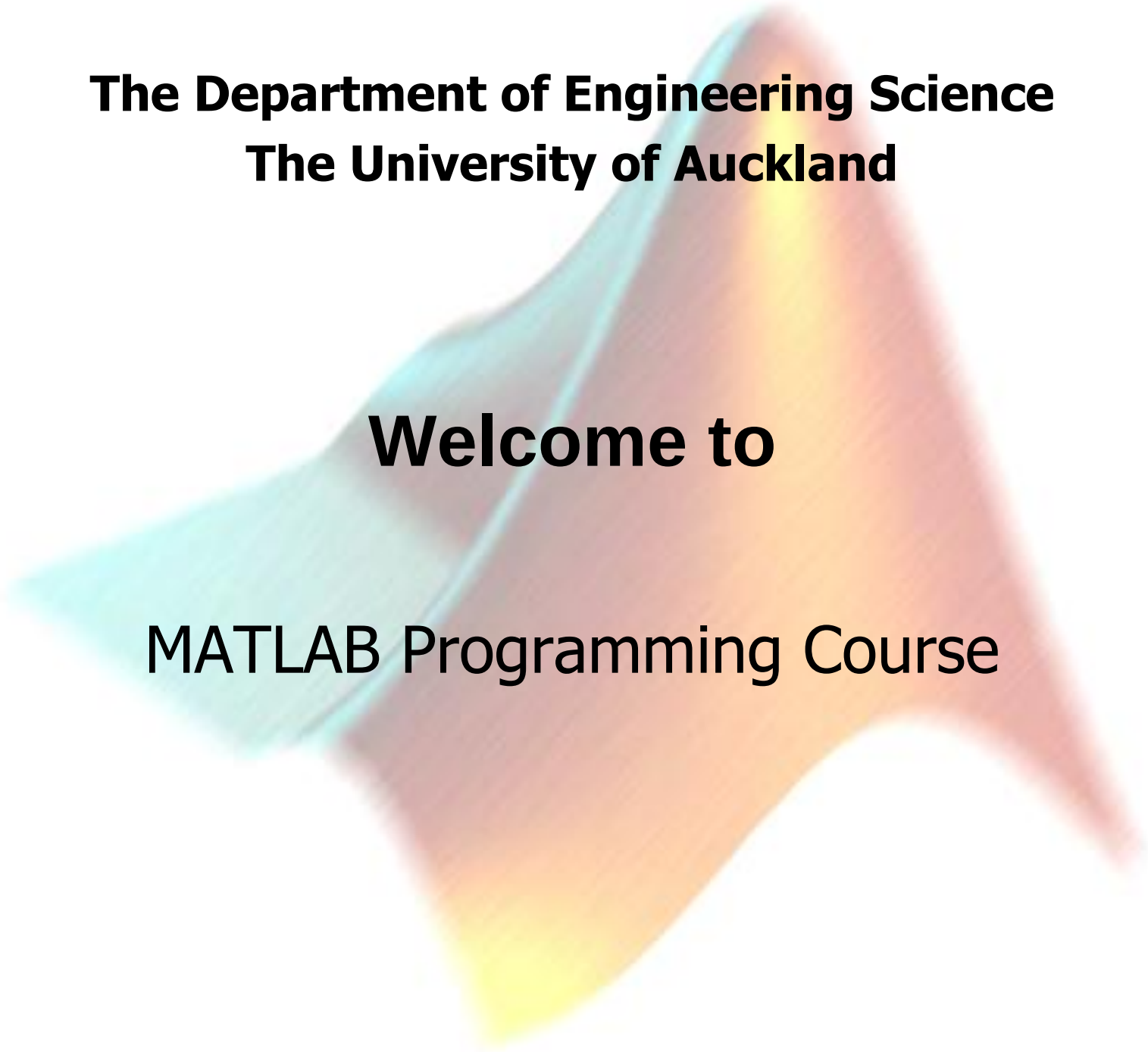
The Department of Engineering Science
The University of Auckland



Welcome to

MATLAB Programming Course

Course Information

- Jim Greenslade– Course Organiser and MATLAB Lecturer
  - Office 439.233 j.greenslade @auckland.ac.nz
- Lab Tutors: PhD students from the Department of Engineering Science.

Course Components

- 4 x 60 minute lectures
- Wednesday AND Thursday
  - 9:00-10:00
  - 1:00-2:00

- 4 x 90 minute labs
- Wednesday AND Thursday
  - 10:30-12 :00
  - 2:30-4:00

Text Books

- MATLAB - Suggested texts:

"A Concise Introduction to Matlab"; or

"Introduction to MATLAB 7 for Engineers"
William J Palm III

- Both books on "desk-copy" in the Engineering Library

Software

- MATLAB
  - Available from the Science Student Resource Centre
    - G16 ground level of the Maths Building 303
- Octave
  - "free software"
  - "mostly compatible with MATLAB"
  - Course staff can give no assistance or assurances

http://www.gnu.org/software/octave

The Department of Engineering Science
The University of Auckland

Chapter 1

An Introduction to MATLAB

Learning outcomes

- Use Matlab as a calculator
- Create and use variables
- Write a script file
- Get input from the user and display output
- Understand the importance of commenting
- Write simple comments

Course Motivation

- Computers are important tools for modern-day engineering.
- Computers allow engineers to perform time consuming tasks and solve problems quickly.
- Computers make visualisation of models possible.

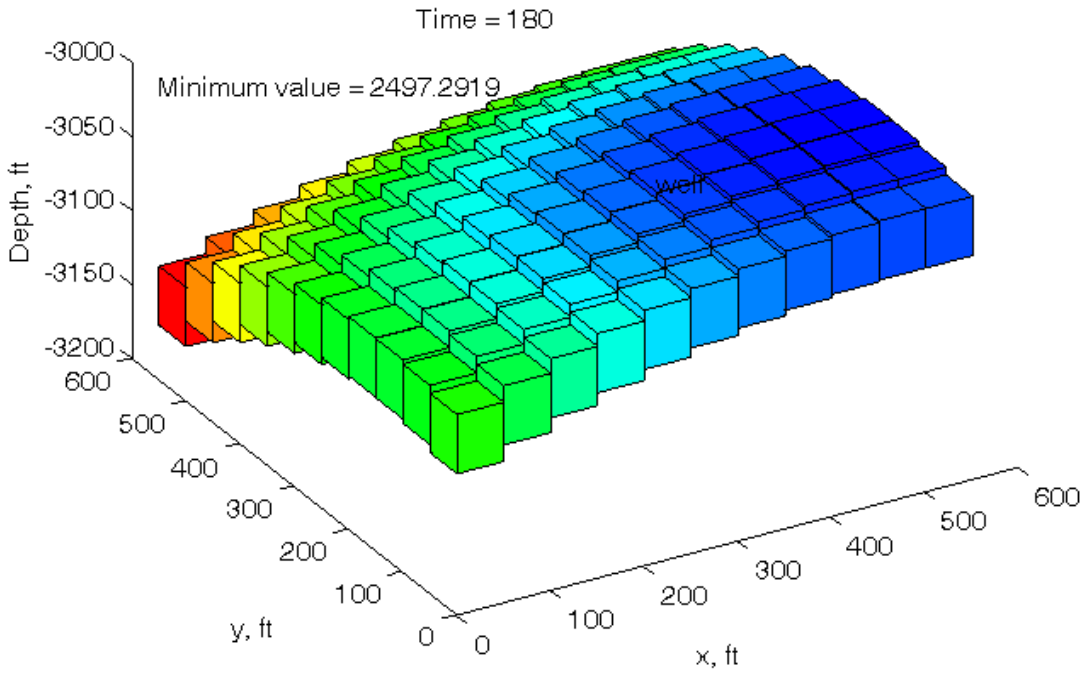


Image: Pressure in an oil reservoir

Solving Equations

- Solving simultaneous equations:

$$2x + y = 4$$

$$x - y = -1$$

- Can solve by hand to get $x = 1, y = 2$

Solving More Equations

- Solving simultaneous equations:

$$\begin{aligned} 2x + y + 2z &= 4 \\ x - y - z &= -1 \\ y - 2z &= 4 \end{aligned}$$

- Can solve by hand to get

$$x = 1.2,\ y = 2.8,\ z = -0.6$$

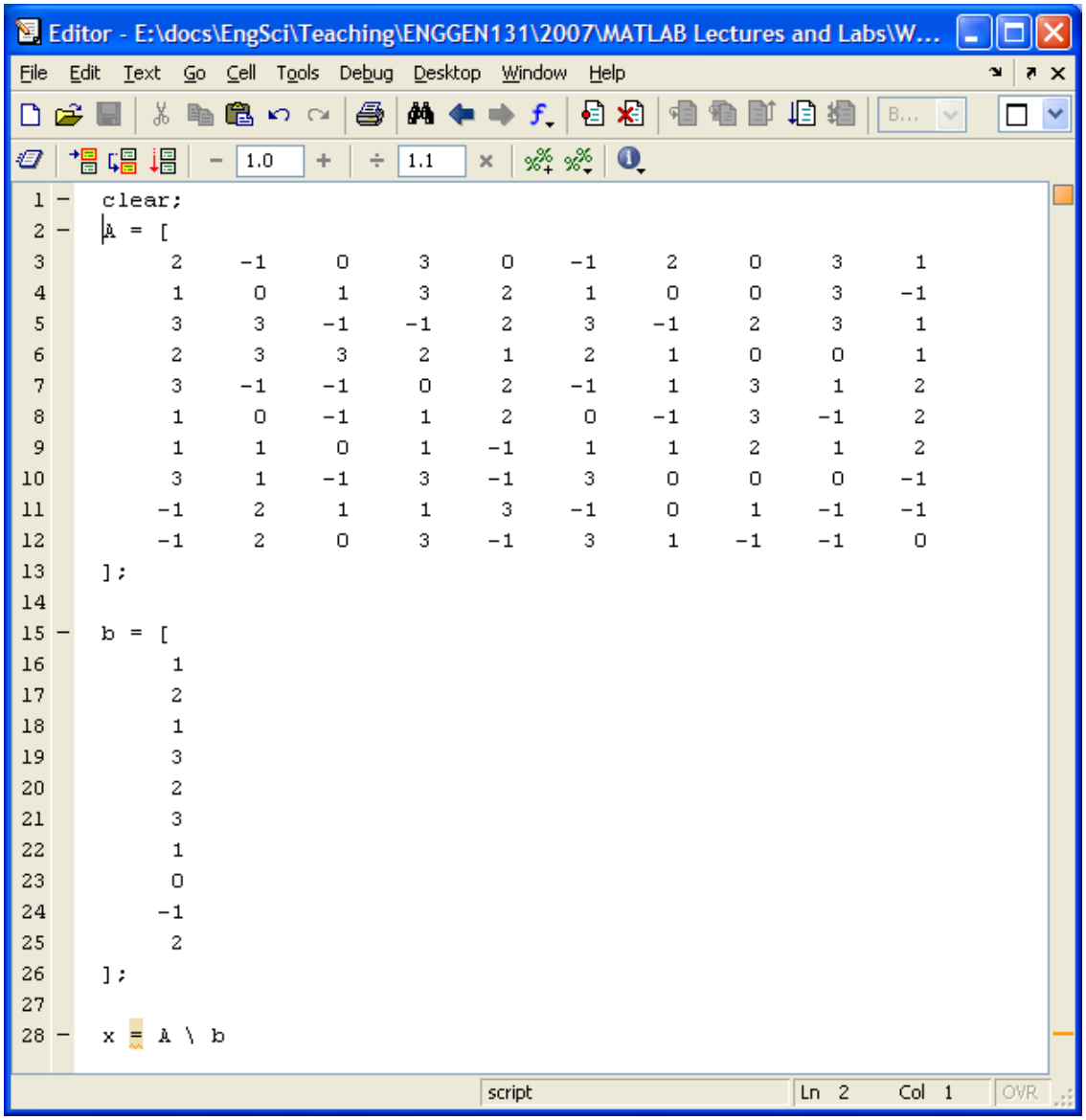
Solving Even More Equations

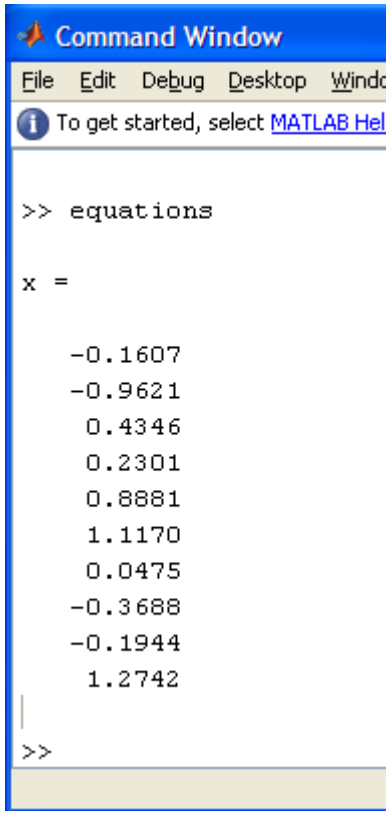
- Solving simultaneous equations:

$$
\begin{array}{rrrrrrrrrrr}
2x_1 & -x_2 & & +3x_4 & & -x_6 & +2x_7 & & +3x_9 & +x_{10} & = 1 \\
x_1 & & +x_3 & +3x_4 & +2x_5 & +x_6 & & & +3x_9 & -x_{10} & = 2 \\
3x_1 & +3x_2 & -x_3 & -x_4 & +2x_5 & +3x_6 & -x_7 & +2x_8 & +3x_9 & +x_{10} & = 1 \\
2x_1 & +3x_2 & +3x_3 & +2x_4 & +x_5 & +2x_6 & +x_7 & & & +x_{10} & = 3 \\
3x_1 & -x_2 & -x_3 & & +2x_5 & -x_6 & +x_7 & +3x_8 & +x_9 & +2x_{10} & = 2 \\
x_1 & & -x_3 & +x_4 & +2x_5 & & -x_7 & +3x_8 & -x_9 & +2x_{10} & = 3 \\
x_1 & +x_2 & & +x_4 & -x_5 & +x_6 & +x_7 & +2x_8 & +x_9 & +2x_{10} & = 1 \\
3x_1 & +x_2 & -x_3 & +3x_4 & -x_5 & +3x_6 & & & & -x_{10} & = 0 \\
-x_1 & +2x_2 & +x_3 & +x_4 & +3x_5 & -x_6 & & +x_8 & -x_9 & -x_{10} & = -1 \\
-x_1 & +2x_2 & & +3x_4 & -x_5 & +3x_6 & +x_7 & -x_8 & -x_9 & & = 2
\end{array}
$$

- Can solve by hand…!?

Using MATLAB





Solving Equations

- Often need to solve systems with 10,000 or 100,000 equations
  - Can be done very quickly using a computer
- This is common in engineering
  - Operations research
  - Mechanics and dynamics
  - Electrical circuits

MATLAB

- MATLAB = MATrix LABoratory
- Extremely useful mathematical software
  - Can be used as an advanced calculator/graphing tool
  - Can be used as a programming language

Why use MATLAB?

- MATLAB is an easy introduction language for programming.
- MATLAB provides a “quick-and-easy” development environment.
- MATLAB is very useful in many engineering contexts.
- MATLAB is used in industry.

Programming with MATLAB

- Programming is a TRANSFERABLE SKILL
  - Programming concepts are common for all languages
  - Syntax may change, but usually similar
- MATLAB is PLATFORM INDEPENDENT
  - Can write software once for many OS
- MATLAB can be linked to other software
  - C/C++, Java, Fortran

MATLAB in Your Degree

- MATHEMATICAL MODELLING 2 and 3
  - You will need to use MATLAB to solve applied mathematical models.
- Other courses
  - structural analysis
  - electrical circuits
  - systems and control
- Plotting results, checking long calculations, etc.

MATLAB is a Marketable Skill

"**Job Description:** Create and maintain steady-state and dynamic thermodynamic system models from conceptual design through the complete design/development process (using industry tools such as MATLAB, Simulink, Altia, etc.). Will also support the design, development, and testing of hardware components and/or subsystems."

from http://www.andrews-space.com/en/employment/career_ops_midlevel_eng_II.htm

Calculations in MATLAB

- MATLAB can be used in a wide range of ways to help you solve engineering problems.

- We will begin by using MATLAB as an advanced calculator:
  - To express mathematics in a form suitable for MATLAB.
  - To use built-in mathematical functions in calculations.
  - To use variables in calculations.

MATLAB as a Calculator

- You can enter expressions at the command line and evaluate them right away.

previous command

```
>> 3 + 5 * 8

ans =

     43

>>
```

next command

The >> symbols indicate where commands are typed.

Mathematical Operators

Operator	MATLAB	Algebra
+	+	5 + 4 = 9
–	-	5 - 4 = 1
×	*	5 * 4 = 20
÷	/	5 / 4 = 1.25
a^b	a^b	5^4 = 625

BEDMAS

B = Brackets
E = Exponentials
D = Division
M = Multiplication
A = Addition
S = Subtraction

```
>> 3*4 + 2

ans =

     14

>> 3*(4+2)

ans =

     18
```

Be careful using brackets – check that opening and closing brackets are matched up correctly.

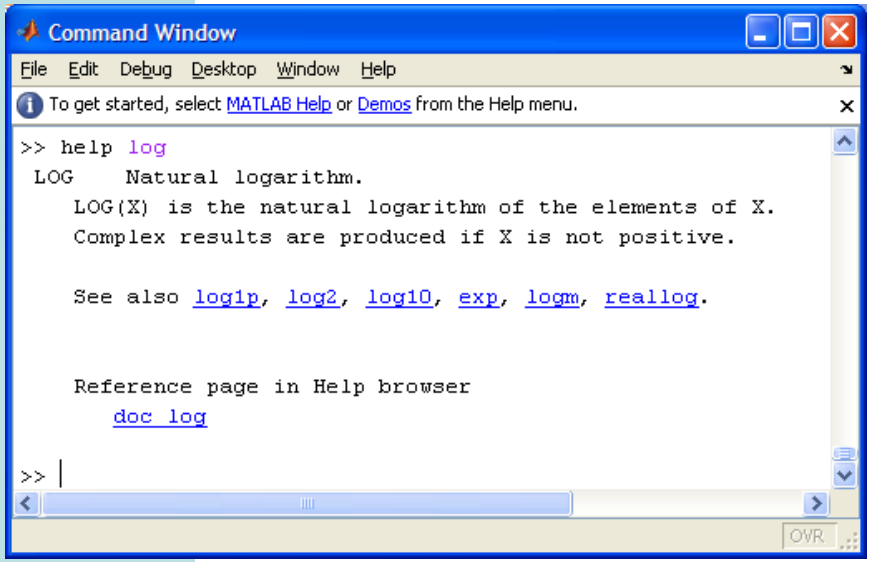
Built-In Functions

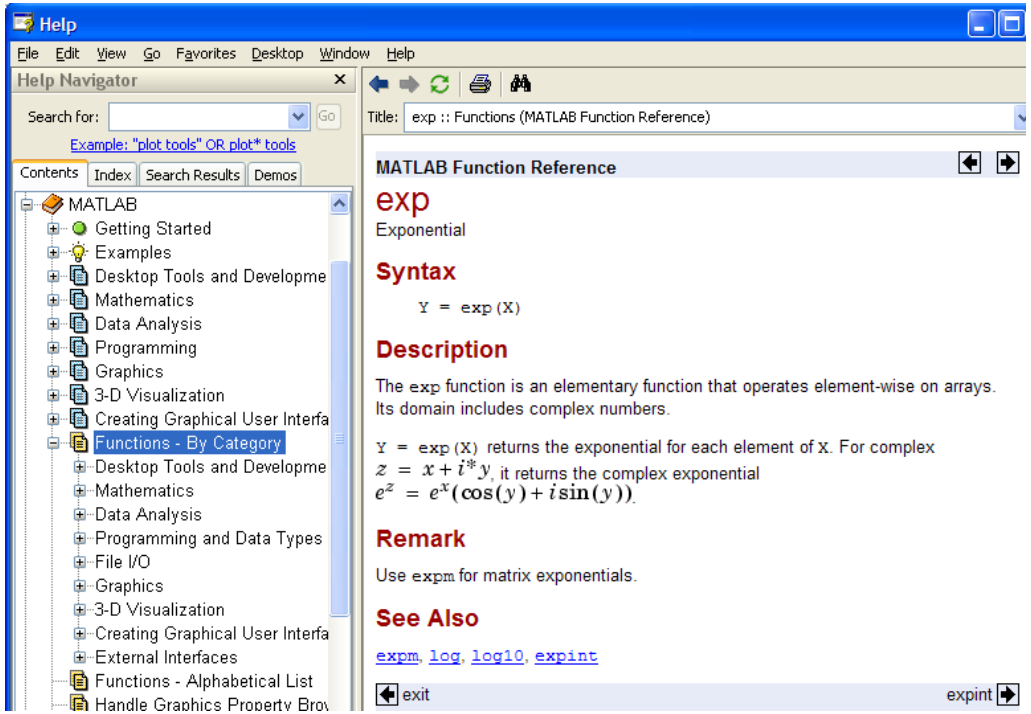
- Like a calculator, MATLAB has many built-in mathematical functions.

```
>> sqrt(4)

ans =

     2

>> abs(-3)

ans =

     3
```

MATLAB Help

- Find out more about functions using MATLAB's help
  - `>> help` – gives command line help
  - `>> doc` – gives GUI help





Variables

- We use variables so calculations are easily represented.

$$C = (F - 32) \times \frac{5}{9}$$

$$F = 100 \Rightarrow C = 37.8$$

$$F = 32 \Rightarrow C = 0$$

- You can think of variables as *named locations in the computer memory in which a number can be stored.*

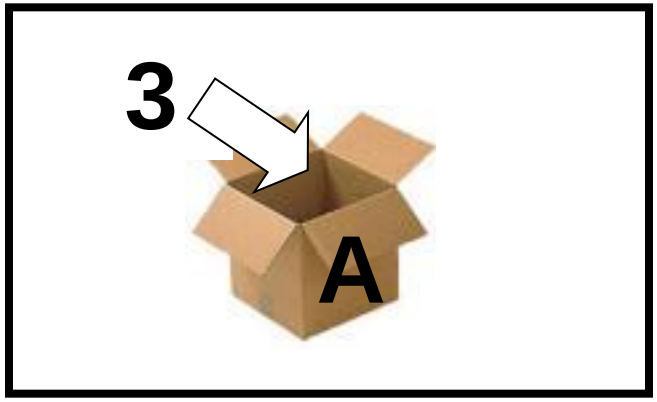
MATLAB Variables

```
>> F = 100
F =
     100
>> C = (F-32)*5/9
C =
    37.7778
>> F = 32
     F = 32
>> C = (F-32)*5/9
C =
     0
```

Memory as a Filing System

- You can think of computer memory as a large set of "boxes" in which numbers can be stored.
- The values can be inspected and changed.

```
>> A = 3

A =

     3
```



- Boxes can be labelled with a variable name.

Assigning Variables

- Either 1) Creates the variable
  - Created in MATLAB Workspace
- Or 2) Changes the variable value
- Always left-to right

`>> a =` expression

calculation

etc

```
>> a = 2

a =

     2

>> 3 = a
??? 3 = a
      |
Error: …
>> b = a

b =

     2
```

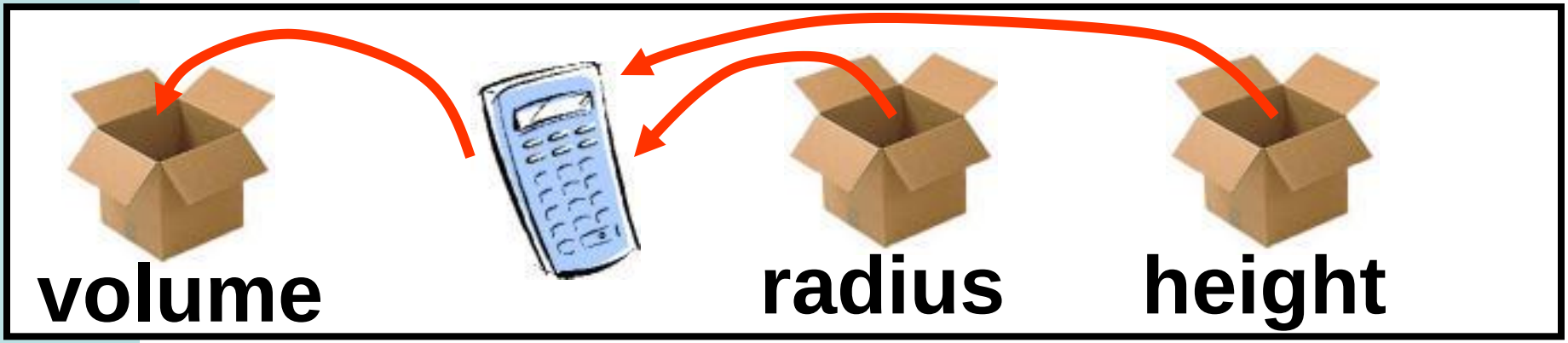
Special Variables

- MATLAB has some special variables:
  - **`ans`** is the result of the last calculation
  - **`pi`** represents π
  - **`Inf`** represents infinity
  - **`NaN`** stands for not-a-number and occurs when an expression is undefined e.g. division by zero
  - **`i, j`** represent the square root of –1 (necessary for complex numbers)

Calculations with Variables

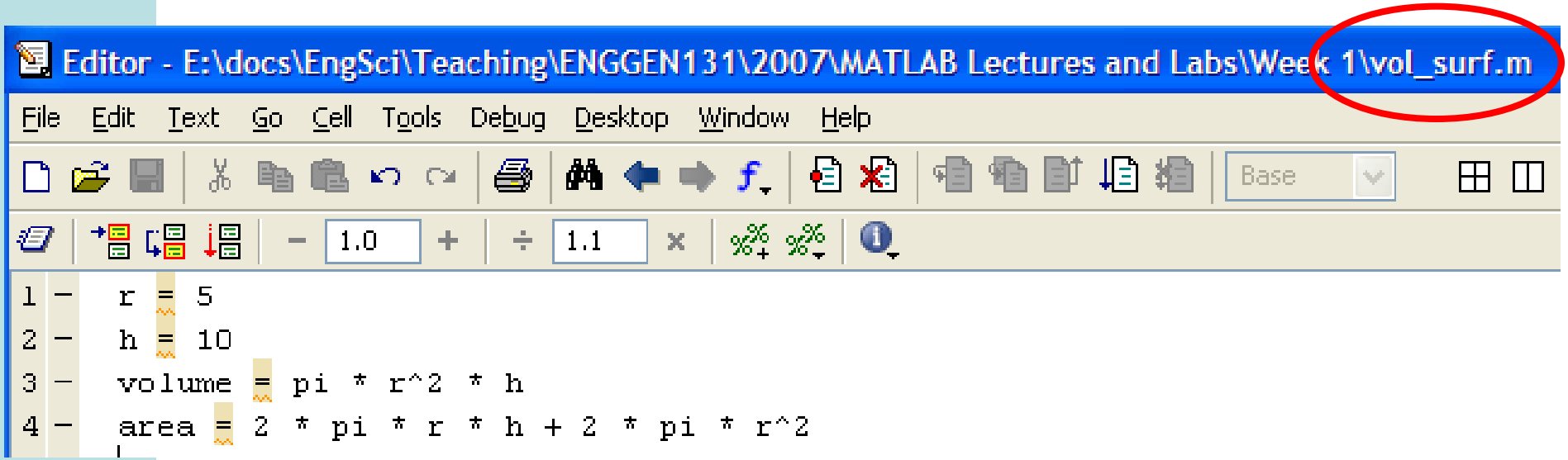
- Suppose we want to calculate the volume of a cylinder.
- It's radius and height are stored as variables in memory.

```
>> volume = pi*radius^2*height
```



Script Files

- You can save a sequence of commands for reuse later
- Each line is the same as typing a command in the command window
- Save the file as *filename*.m



Script Files

- Run sequence of commands by typing

  filename

  in the command window

```
>> vol_surf

r =

     5

h =

     10

volume =

     785.3982

area =

     471.2389

>>
```

Commenting

- Comment lines start with a %
- Not executed by Matlab, just for people reading the code
- Helps people understand what the code is doing and why!
- VERY IMPORTANT
- Good commenting is a huge help when maintaining/fixing/extending code

Header comments

- Every script file should have a header
- Indicates what the purpose of the file is

```
% ConvertTemp.m converts the freezing and boling points for
% water from degrees Celsius (c) to Farenheit (f)
% Author: Peter Bier
```

- Matlab incorporates this header as help

```
>> help ConvertTemp

ConvertTemp.m converts the freezing and boling points for
 water from degrees Celsius (c) to Farenheit (f)
 Author: Peter Bier
```

- No header = no lab mark

Other comments

- Comment anything that is not easy to understand
- Write USEFUL comments, compare the following:

```
% set x to zero
x = 0
% calculate y
y = x * 9/5 + 32
```

```
% Convert freezing point of water from
% celsius to farenheit
c = 0
f = c * 9/5 + 32
```

- No need to go overboard but…
- No comments = no lab mark

Basic user interaction: I/O

- Use input command to get input from user and store in a variable:

```
height = input('Enter the height:')
```

Matlab will display the message enclosed in quotes, wait for input and then store the entered value in the variable

Basic user interaction: I/O

- Use disp command to show something to a user

```
disp('The area of the rectangle is')
disp(area)
```

Matlab will display any message enclosed in quotes and the value of any variable

Optional Reading

Chapter 1 An Introduction to Matlab	Introduction to Matlab 7 for Engineers (2nd ed)		A Concise Introduction to Matlab (1st ed)	
Topic	**Section**	**Pages**	**Section**	**Pages**
Using Matlab as a calculator	1.1	6-17	1.1	2-13
Menus and the toolbar	1.2	17-19	1.2	13-15
Script Files	1.4	29-32	1.4	23-26
Input/Output	1.4	36-38	1.4	26-28
Help	1.5	38-43	1.5	28-31
Help			1.6	32

The Department of Engineering Science
The University of Auckland

Chapter 2

1D Arrays, Problem Solving

Learning outcomes

- Explain what a 1D array is
- Create and manipulate 1D arrays
- Draw plots of 1D arrays
- Use 1D arrays in programs
- Outline the five steps for problem solving
- Use the five steps to solve a problem

MATLAB Arrays

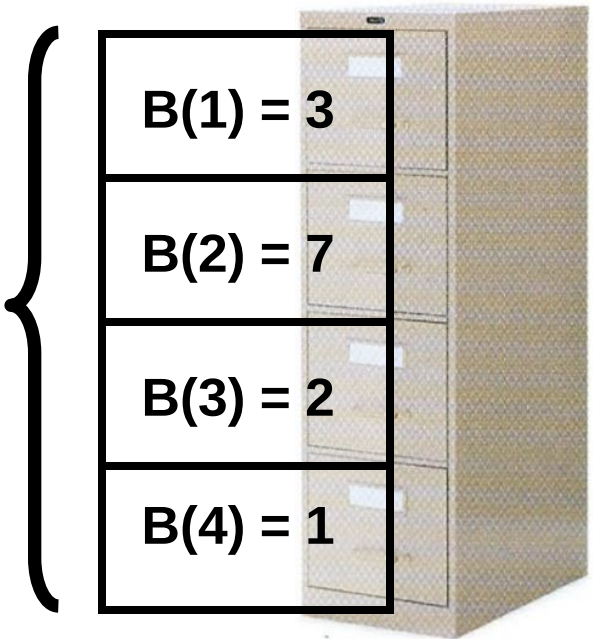
- So far MATLAB variables hold a single value
- Can also create MATLAB arrays that hold multiple values
- Useful for storing lists of values (1D arrays) or tables of values (2D arrays)
- Can be used for dealing with vectors and matrices (Lecture 10)

Array Variables versus Scalars

- If a scalar variable (for a single value) is like a *cardboard box*, a 1D array variable is like a *filing cabinet*

```
>> B=[3, 7, 2, 1]

B =

     3   7   2   1
```

B



Creating 1D arrays

- Create a list of values by enclosing numbers within `[ ]` and separating by `,` or a space.

```
>> dailyHighs = [10, 11, 13, 12, 19, 18, 17]
dailyHighs =
10   11   13   12   19   18   17

>> dailyLows = [3   2   4   1   5   6   4]
dailyLows =
3   2   4   1   5   6   4
```

Accessing Array Elements

- You can access/change a particular array element using ()

```
>> dailyHighs
dailyHighs =
10   11   13   12   19   18   17
>> dailyHighs(2)
ans =
11
>> dailyHighs(2) = 14
dailyHighs =
10   14   13   12   19   18   17
```

Extending arrays

- You can add extra elements by
  - creating them directly `()`
  - concatenating them `[ , ]`

```
>> dailyHighs
dailyHighs =
10   14   13   12   19   18   17
>> dailyHighs(8) = 12
dailyHighs =
10   14   13   12   19   18   17   12
>> dailyHighs = [dailyHighs, 14]
dailyHighs =
10   14   13   12   19   18   17   12   14
```

Default Array Elements

- If you don't assign array elements, MATLAB gives them a default value of 0

```
>> dailyHighs
dailyHighs =
10   14   13   12   19   18   17   12   14
>> dailyHighs(12) = 10
dailyHighs =
10   14   13   12   19   18   17   12   14   0   0   10
```

Using Arrays in Programming

- Main use for arrays in programming is data storage
  - keeping track of the trajectory of a basketball
  - storing the stress along a beam
  - storing pressures inside the heart

Using Arrays in MATLAB

- MATLAB provides lots of special array functionality
- Using arrays and MATLAB functions allows repetitive calculations to be done quickly
- Also allows for compact programs.
- MATLAB originally written for use with arrays
  - very good at dealing with arrays

Automatic 1D Arrays

- Ways to create 1D arrays automatically

```
>> x = 0:10

x =

     0   1   2   3   4   5   6   7   8   9   10

>> t = linspace(0,10,7)

t =

     0   1.6667   3.3333   5.0000   6.6667   8.3333   10.0000

>>
```

This command creates a list of 7 points spaced evenly between 0 and 10

Array Slicing

- It is possible to access several elements of an array at once
- Instead of using a using a single value to index the array we can use another array

```
>> dailyHighs
dailyHighs =
10   14   13   12   19   18   17   12   14   0   0   10
>> dailyHighs([2,4,6])
dailyHighs =
14   12   18
```

Array Slicing

- The colon operator is handy when you want to pull out a sequence of values

```
>> dailyHighs
dailyHighs =
10   14   13   12   19   18   17   12   14   0   0   10
>> dailyHighs(3:5)
dailyHighs =
13   12   19
```

Array Arithmetic

- Arrays of the same length can be added or subtracted to each other.
- Arrays can also be multiplied by scalar constants.

```
>> dailyHighs = [10, 11, 13, 12, 19, 18, 17];
>> dailyLows =  [ 3,  2,  4,  1,  5,  6,  4];
>> dailyRange = dailyHighs - dailyLows
dailyRange =
     7   9   9   11   14   12   13
>> dailyAverage = 0.5 * (dailyHighs + dailyLows)
dailyAverage =
     6.5   6.5   8.5   6.5   12   12   10.5
```

Array Arithmetic

- It is possible to multiply the elements in one array by the corresponding elements in another array.
- To do this we use the dot operator

```
>> heights = [9, 8, 4, 6];
>> widths =  [3, 2, 1, 5];
>> areas = heights .* widths
areas =
     18   16   4   30
     27
```

Array Arithmetic

- We can also do element by element division

```
>> heights = [9, 8, 4, 6];
>> widths =  [3, 2, 1, 5];
>> ratios = heights ./ widths
ratios =
     3   4   4   1.2
```

- Similarly we can do element by element exponentiation

```
>> heights = [9, 8, 4, 6];
>> square = heights.^2
square=
     81   64   16   36
```

Array Functions

- Standard mathematical functions (sin, cos, exp, log, etc) can apply to arrays as well as scalars

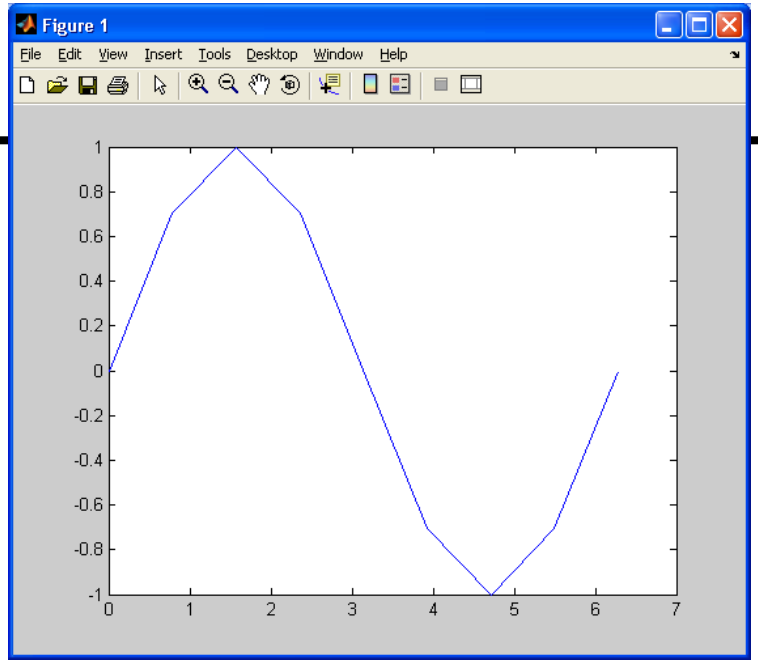
```
>> x = [1, 2, 3];
>> y = sin(x);
```

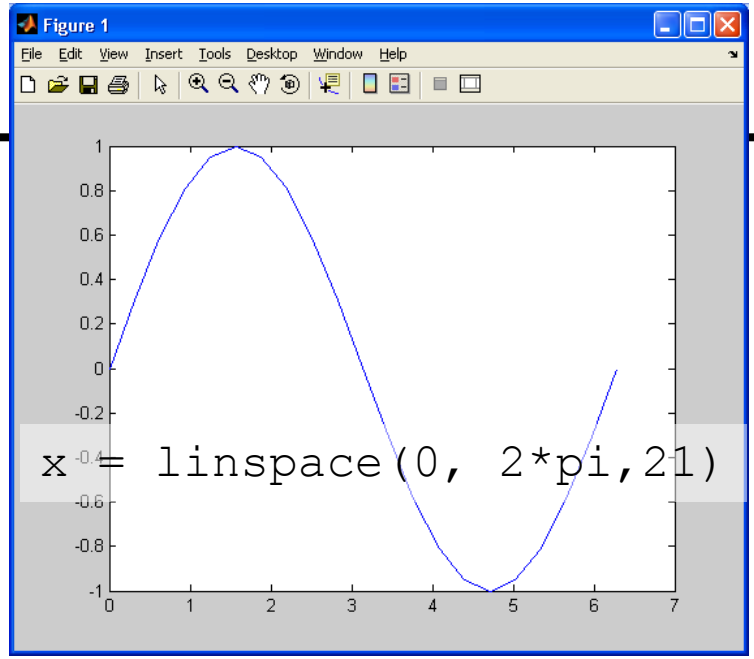
`y` is `[sin(1), sin(2), sin(3)]`

  - When writing functions (Lecture 3) remember input might be an array

Array Functions

```
>> x = linspace(0, 2*pi,9)

x =

         0    0.7854    1.5708    2.3562    3.1416    3.9270    4.7124    5.4978    6.2832

>> y = sin(x)

y =

         0    0.7071    1.0000    0.7071    0.0000   -0.7071   -1.0000      -0.7071
0.0000

>> plot(x,y)
```





Special Array Functions

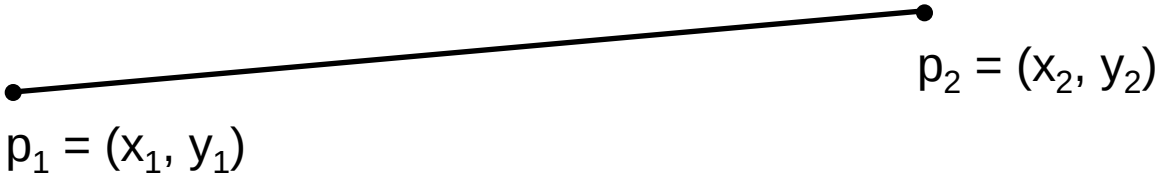
- Some functions are specialised for use with 1D arrays
  - `length(array)` gives the number of elements in `array`
  - `min(array)` gives the minimum value in `array`
  - `max(array)` gives the maximum value in `array`
  - `sum(array)` gives the sum of values in `array`

5 Steps for Problem Solving

1. State the problem clearly
2. Describe the input and output information
3. Work the problem by hand (or with a calculator) for a simple set of data
4. Develop a solution and convert it to a computer program
5. Test the solution with a variety of data

Problem-Solving Worked Example

- We want to compute the distance between two points in a plane



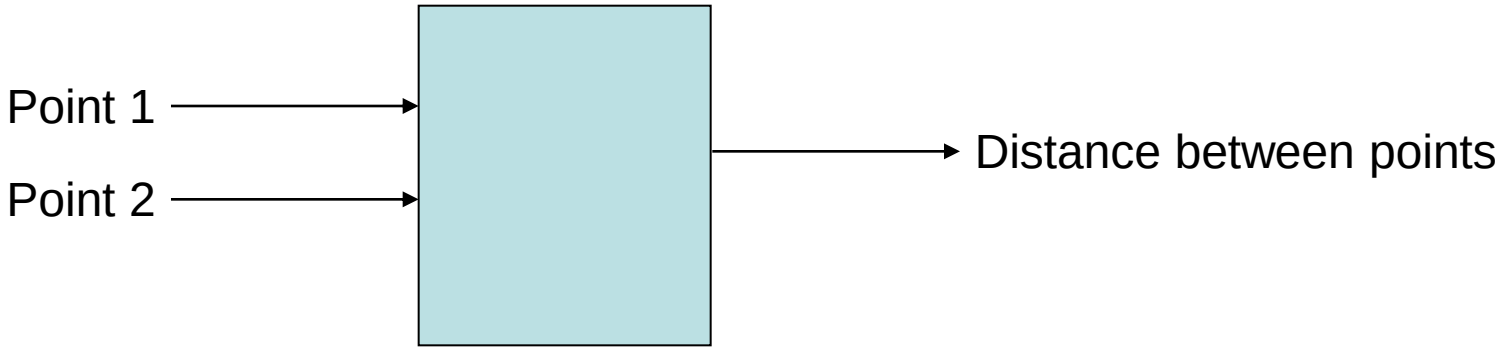
Step 1: Problem Statement

- State the problem clearly

Compute the straight-line distance between two points in a plane.

Step 2: Input/Output Description

- Describe information given to solve problem
  - Input
- Identify values to be computed
  - Output
- I(nput)/O(utput) diagram

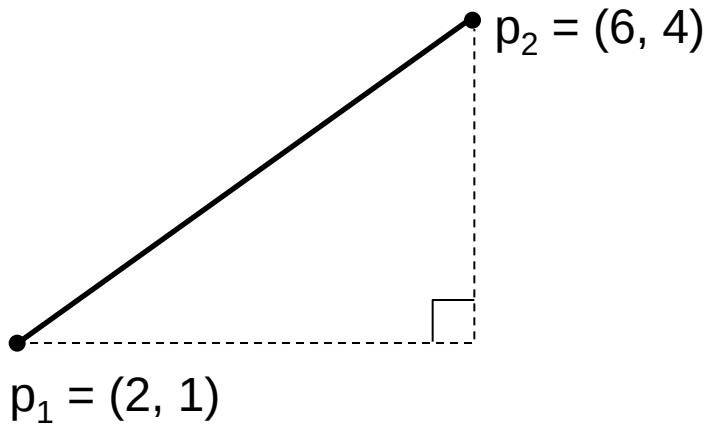


Step 3: Work the problem by hand

- Work problem by hand
  - Use a calculator if necessary
- Very important step
  - Don’t skip even for simple problem
  - If you cannot do this step
    - read problem again
    - consult reference material
- Diagrams can be useful

Step 3:

- Known solution, distance = 5



$$\begin{aligned}\text{distance} &= \sqrt{(\text{side}_1)^2 + (\text{side}_2)^2} \\ &= \sqrt{(6-2)^2 + (4-1)^2} \\ &= \sqrt{4^2 + 3^2} = \sqrt{16+9} = \sqrt{25} \\ &= 5\end{aligned}$$

Step 4: Develop a solution and convert it to a computer program

- Decompose problem into set of steps
  - Simple problems give simple steps
    - Give pseudocode/flowchart for code
  - Complex problems give complex steps
    - Give pseudocode/flowchart for functions
    - Each complex step may require problem-solving process

Step 4:

- Pseudocode
  1. Get x- and y-values for two points
  2. Compute length of side of right angle triangle generated by points
  3. Use hypotenuse calculation to get distance
  4. Return the distance

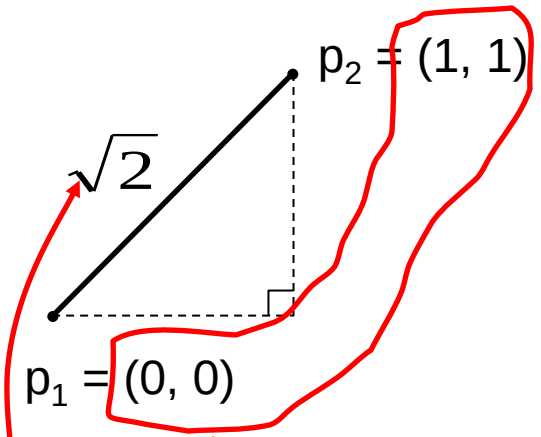
Step 4:

```
% This script file computes the straight-line
% distance between two points in a plane.
% Input: (x1, y1) = coordinates of point 1
%        (x2, y2) = coordinates of point 2
% Output: distance = distance between points

% Get x- and y-values for two points
disp('Enter coordinates for point 1:');
x1 = input('x-coordinate > '); % x-value for p1
y1 = input('y-coordinate > '); % y-value for p1

disp('Enter coordinates for point 2:');
x2 = input('x-coordinate > '); % x-value for p2
y2 = input('y-coordinate > '); % y-value for p2

% Compute length of side of right angle triangle generated by points
side1 = x2 - x1; % Length of the x-side
side2 = y2 - y1; % Length of the y-side

% Use hypotenuse calculation to get distance
distance = sqrt(side1^2 + side2^2);

% Return the distance
disp('The distance is:'), distance
```

Step 5: Test the solution

- Test using hand example
- Test with other data

```
>> getdist
Enter coordinates for point 1:
x-coordinate > 2
y-coordinate > 1
Enter coordinates for point 2:
x-coordinate > 6
y-coordinate > 4
The distance is:

distance =

     5

>>
```




```
>> getdist
Enter coordinates for point 1:
x-coordinate > 0
y-coordinate > 0
Enter coordinates for point 2:
x-coordinate > 1
y-coordinate > 1
The distance is:

distance =

    1.4142

>>
```

Recommended Reading

Chapter 2 1D Arrays, Problem Solving	Introduction to Matlab 7 for Engineers (2nd ed)		A Concise Introduction to Matlab (1st ed)	
Topic	**Section**	**Pages**	**Section**	**Pages**
Arrays	1.3	19-20	1.3	16-17
Arrays	2.1	70-81	2.1	38-48
Element by element operations	2.3	83-97	2.3	49-57
Problem Solving	1.7	52-60		

Lab #1 Preview

- Navigating MATLAB
- MATLAB help system
- Calculations and variables
- Script files
- Commenting
- Simple input and output commands