Numerical Methods

Introduction

Time Commitment

Lecture	21 hrs
Practicals	15 hrs
Total	**36 hrs**

Lecturer

General Lecture	Dr. Thai Minh Quan : - 081.555.9669 - Thai-minh.quan@usth.edu.vn
Practical work (computer lab)	Dr. Thai Minh Quan (AE class)
	Dr. Phan Thanh Hien (SA + EN + MST class)
	Dr. Giang Anh Tuan (ICT1 class)
	Dr. Giang Anh Tuan (ICT2 class)

Evaluation

Attendance/Attitude	10%
Exercises	10%
Mid-term test	30%
Final exam	50%
Total	**100%**

Content

No	Contents	Hours			Ref./Resources
		Lect.	Exr.	Prc.	
1	Introduction:	3		1	[1]: Chap 1-2
2	Roots of Non-linear equations	3		1.5	[1] Chap 6
3	Systems of linear equations	3		2	[1] Chap 9
4	LU decomposition	2		2	[1] Chap 10
5	Curve fitting	1		1	
6	Linear Programming	1		1	[1] Chap 15
7	Numerical Differentiation and Integration	2		3	[1] Chap 21-23

References

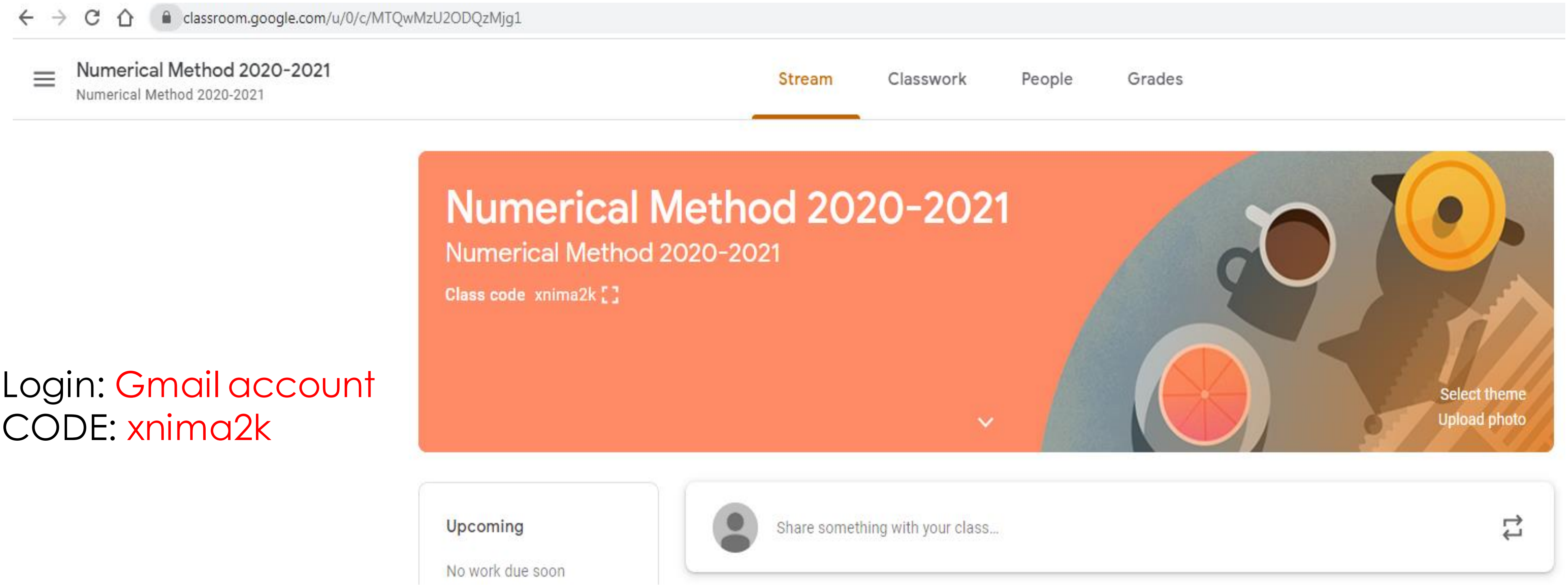
Book: S.C Chapra et al., "Numerical methods for Engineers“, 6th Edition, MacGraw-Hill, 2006

Book: Introduction to Numerical Methods and Matlab Programming for Engineers. - *Todd Young and Martin J. Mohlenkamp*

Lecture notes – *Thai Minh Quan in Classroom*

https://classroom.google.com

References



Login: Gmail account
CODE: xnima2k

Motivation

Analytical method

$$x-3=0$$

$$x^2-2x-3=0$$

$$x^3-2x^2-2x+3=0$$

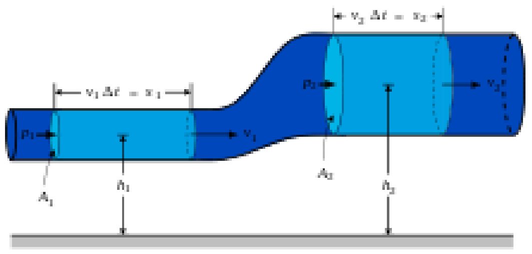
$$x^4+x^3-2x^2-3x+3=0$$

$$x^5-x^4+x^3-2x^2-3x+3=0$$

$$a_n x^n + a_{n-1}x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0 = 0$$

Numerical methods

Motivation



fluid flow

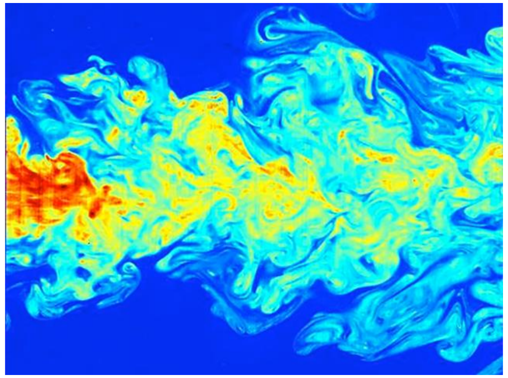
Or…

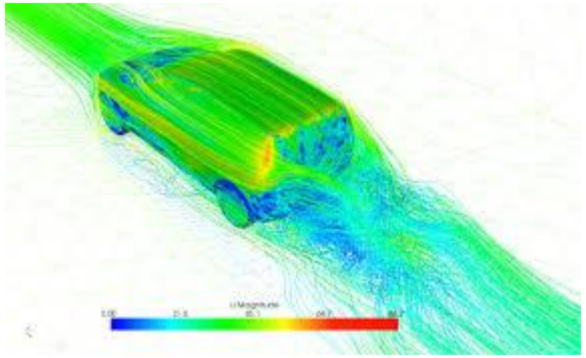
Navier-Stokes momentum equation *(convective form)*

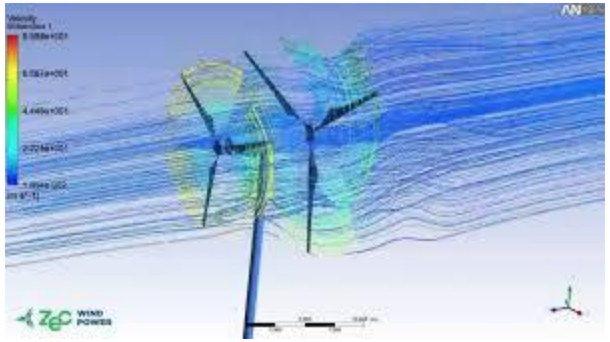
$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla \bar{p} + \nu \nabla^2 \mathbf{u} + \tfrac{1}{3}\, \nu \nabla (\nabla \cdot \mathbf{u}) + \mathbf{g}.$$

Analytical methods ?

Numerical methods





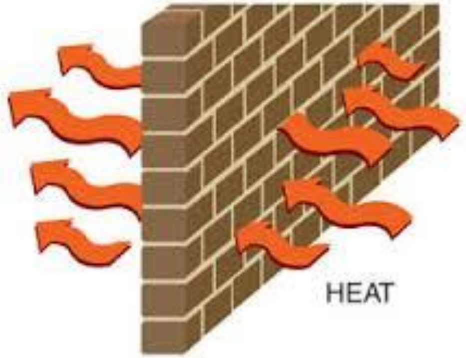


Motivation

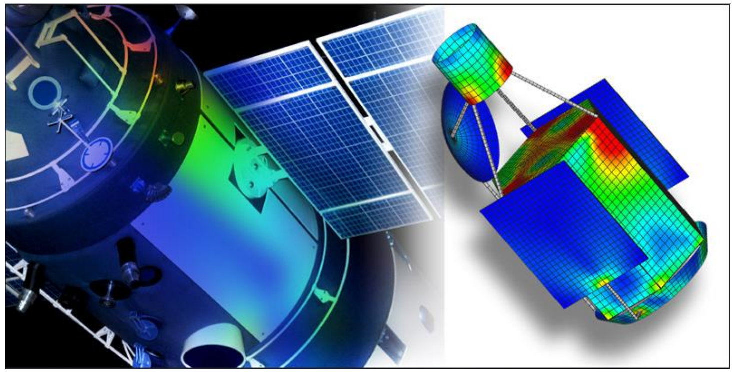
Or…

Fourier's law

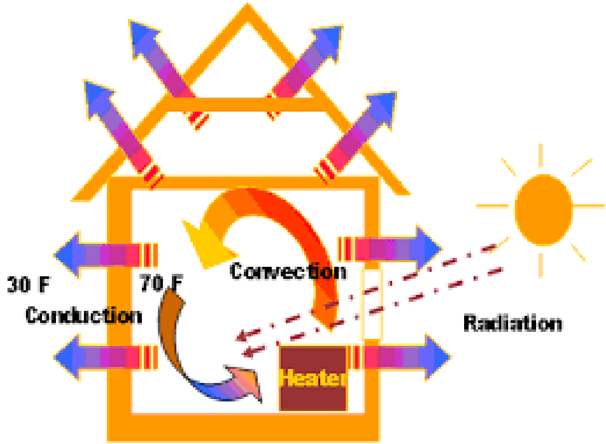
$$\boxed{\vec{q} = -k\nabla T}$$



Analytical methods ?



Numerical methods



Reasons to study

- Solve problems with no analytic solution

  Non-linear equations

  Complex behaviors

- Understand these methods

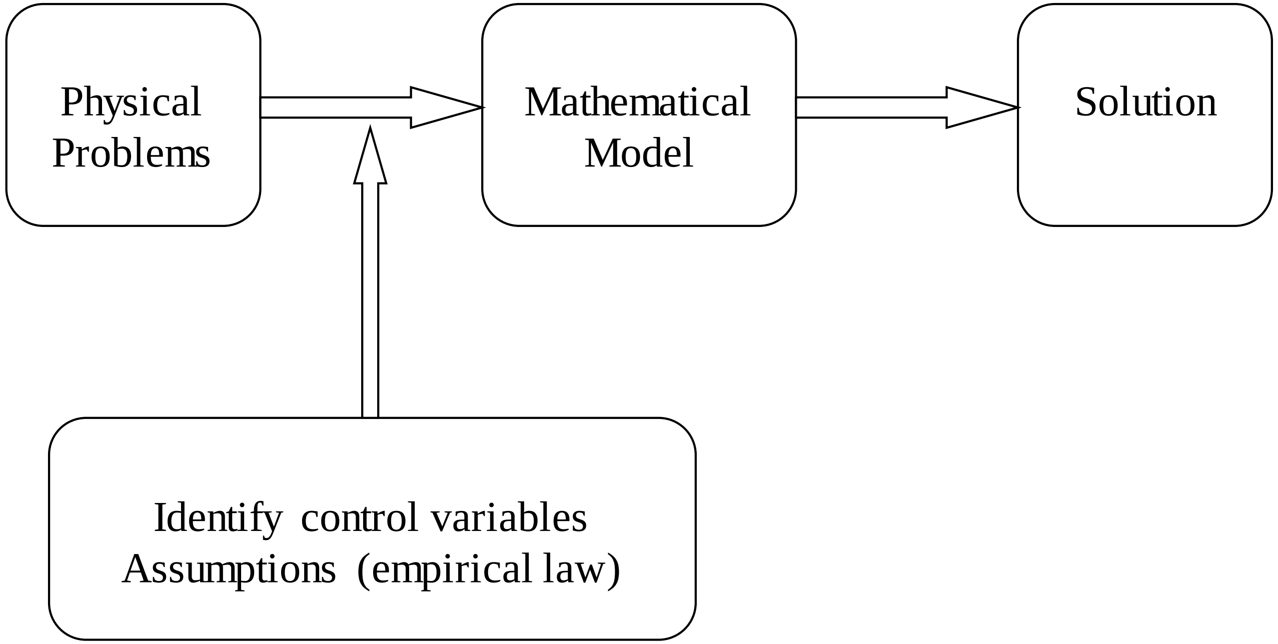
  - Gain familiarity with common algorithms

  - Computing realities and calculations in principle

What is Numerical Method

- Numerical method is an approach for solving complex mathematical problem using only simple arithmetic operations
- The approach involves formulation of model of physical situations that can be solved with arithmetic operations
- Step involved
  1. Formulation of mathematical model
  2. Construction of an appropriate numerical method
  3. Implementation of the method to obtain a solution
  4. Validation of the solution

Mathematical model



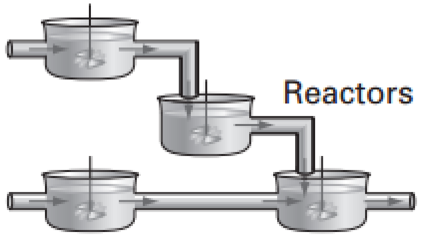
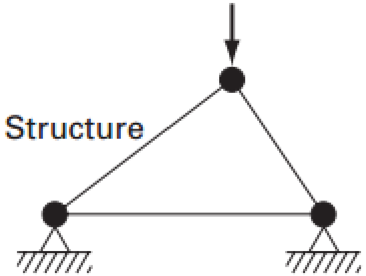
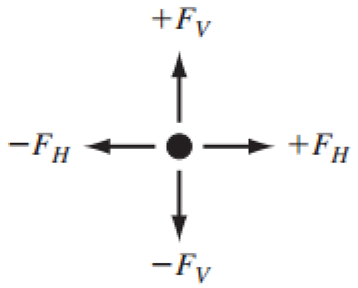
Analysis Method

- Engineering Analysis
  - Classical method
    - Exact Solution
    - Approximate Solution
  - Numerical method
    - Energy
    - Boundary element
    - Finite Difference
    - Finite element

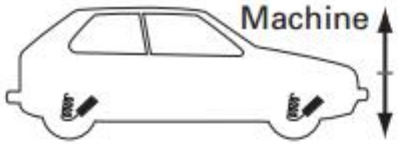
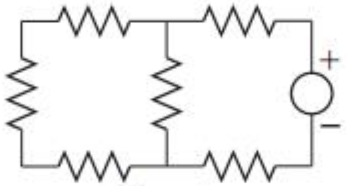
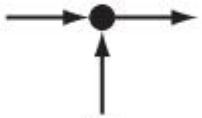
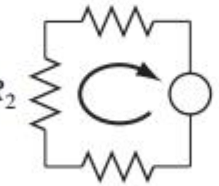
Characteristics of Numerical Methods

1. The solution procedure is iterative, with the accuracy of the solution improving with each iteration
2. The solution procedure provides only an approximation to the true, but unknown, solution
3. The algorithm is simple and can be easily programmed
4. The solution procedure may occasionally diverge from rather than converge to the true solution

Some applications

Field	Device	Organizing Principle	Mathematical Expression
Chemical engineering		Conservation of mass	Mass balance:  Over a unit of time period Δmass = inputs − outputs
Civil engineering		Conservation of momentum	Force balance:  At each node Σ horizontal forces $(F_H) = 0$ Σ vertical forces $(F_V) = 0$

Some applications

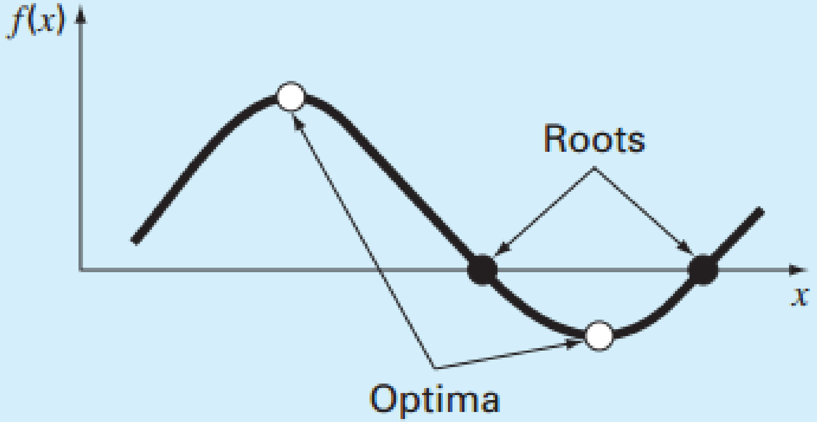
Mechanical engineering		Conservation of momentum	Force balance: Upward force, $x = 0$, Downward force $m \frac{d^2x}{dt^2}$ = downward force − upward force
Electrical engineering	 Circuit	Conservation of charge	Current balance:  For each node Σ current $(i) = 0$
		Conservation of energy	Voltage balance:  Around each loop Σ emf's − Σ voltage drops for resistors = 0 $\Sigma \xi - \Sigma iR = 0$

Some applications

(a) ***Part 2*: Roots and optimization**

Roots: Solve for x so that $f(x) = 0$

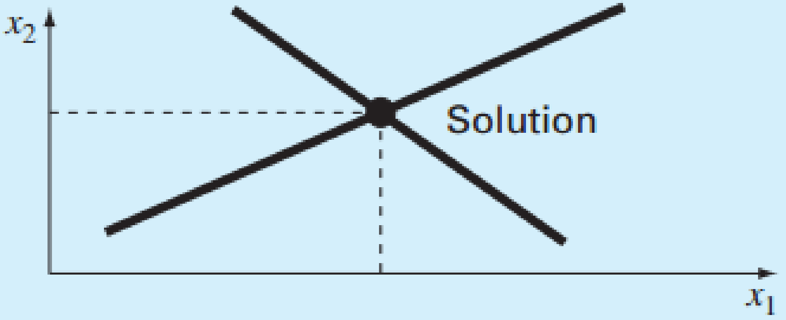
Optimization: Solve for x so that $f'(x) = 0$



(b) ***Part 3*: Linear algebraic equations**

Given the a's and the b's, solve for the x's

$$a_{11}x_1 + a_{12}x_2 = b_1$$
$$a_{21}x_1 + a_{22}x_2 = b_2$$

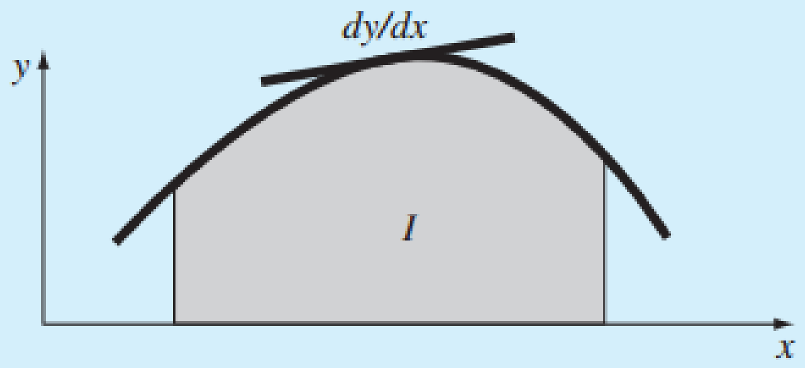


Some applications

(d) *Part 5*: Integration and differentiation

Integration: Find the area under the curve

Differentiation: Find the slope of the curve



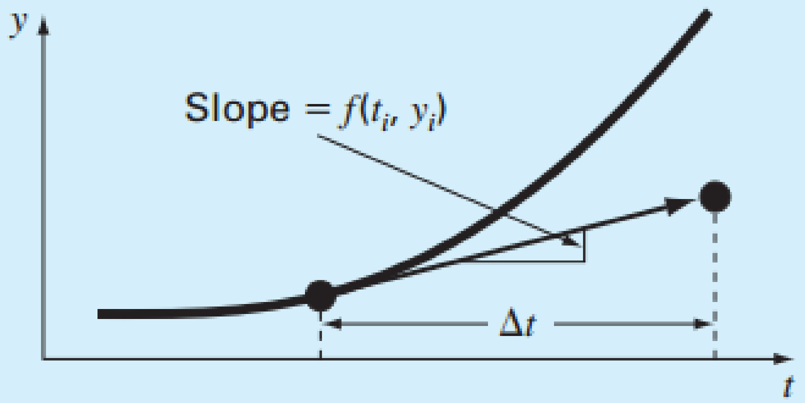
(e) *Part 6*: Differential equations

Given

$$\frac{dy}{dt} \approx \frac{\Delta y}{\Delta t} = f(t, y)$$

solve for y as a function of t

$$y_{i+1} = y_i + f(t_i, y_i)\Delta t$$



Example

Problem Statement. A bungee jumper with a mass of 68.1 kg leaps from a stationary hot air balloon. Use Eq. (1.9) to compute velocity for the first 12 s of free fall. Also determine the terminal velocity that will be attained for an infinitely long cord (or alternatively, the jumpmaster is having a particularly bad day!). Use a drag coefficient of 0.25 kg/m.

$$v(t) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}}t\right)$$

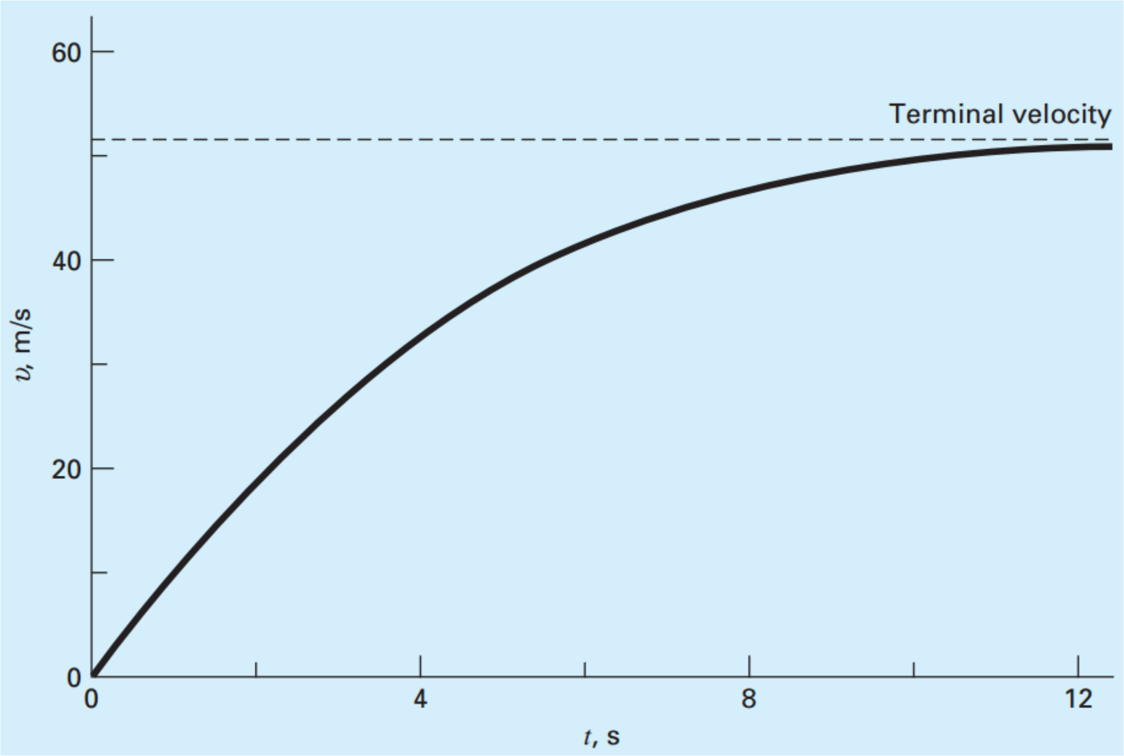
$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

Analytical method

$$v(t) = \sqrt{\frac{9.81(68.1)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{68.1}}t\right) = 51.6938 \tanh(0.18977t)$$

which can be used to compute

t, s	v, m/s
0	0
2	18.7292
4	33.1118
6	42.0762
8	46.9575
10	49.4214
12	50.6175
∞	51.6938

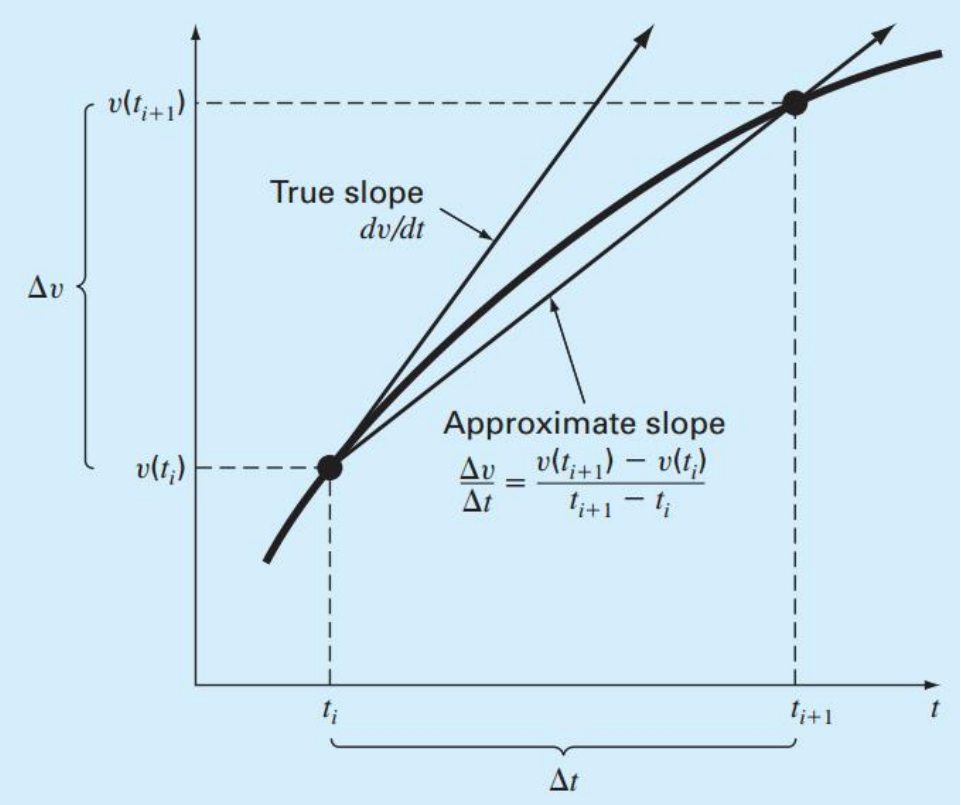


Numerical method

$$v(t) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}}t\right)$$

$$\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i}$$

$$\frac{dv}{dt} = \lim_{\Delta t \to 0} \frac{\Delta v}{\Delta t}$$



Numerical method

$$v(t) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right)$$

$$\frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} = g - \frac{c_d}{m} v(t_i)^2$$

This equation can then be rearranged to yield

$$v(t_{i+1}) = v(t_i) + \left[g - \frac{c_d}{m} v(t_i)^2\right](t_{i+1} - t_i)$$

$$v_{i+1} = v_i + \frac{dv_i}{dt} \Delta t$$

This approach is formally called *Euler's method*

Numerical method

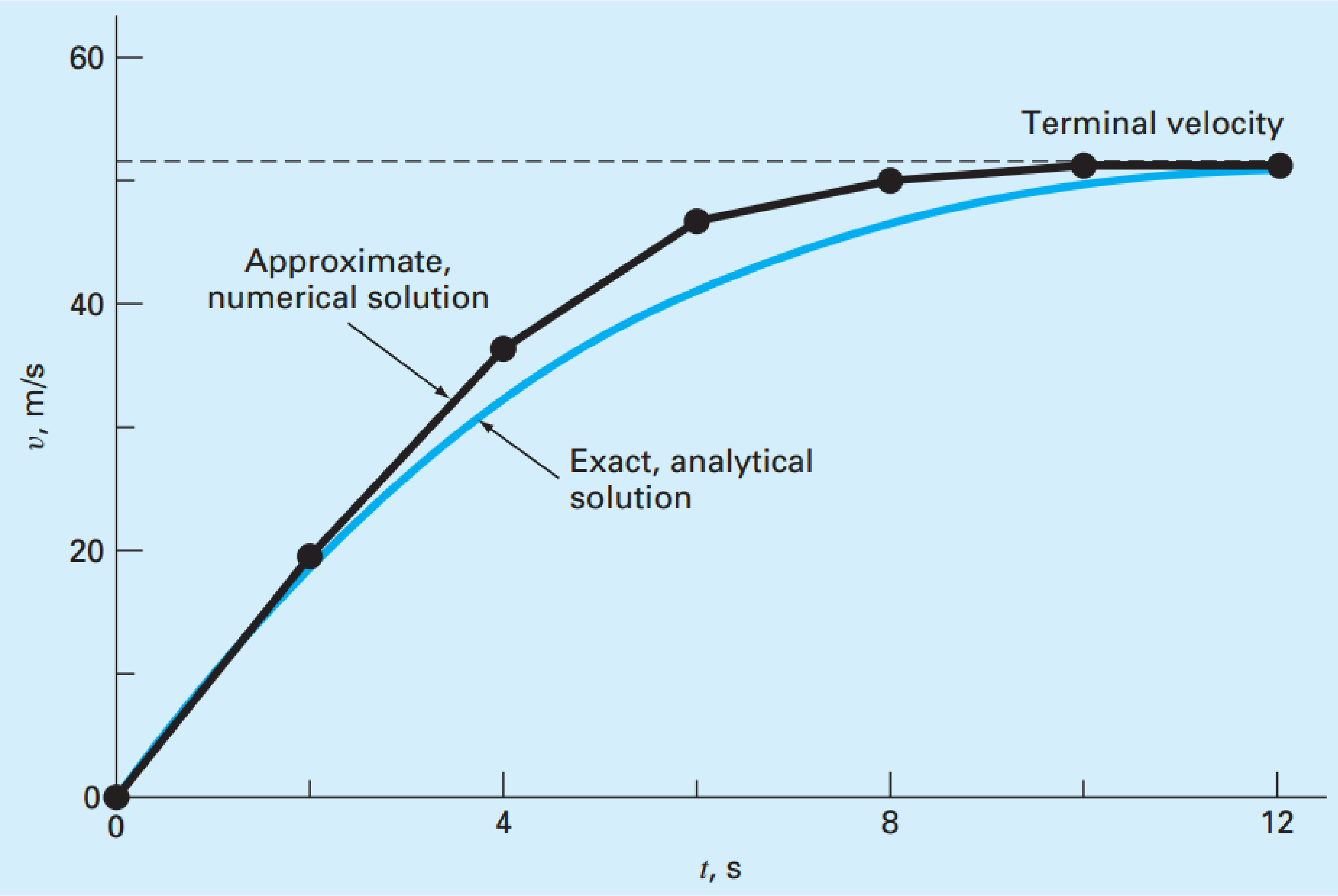
$$v(t_{i+1}) = v(t_i) + \left[g - \frac{c_d}{m} v(t_i)^2\right](t_{i+1} - t_i)$$

at $t_1 = 2$ s $\quad v = 0 + \left[9.81 - \frac{0.25}{68.1}(0)^2\right] \times 2 = 19.62$ m/s

For the next interval (from $t = 2$ to 4 s)

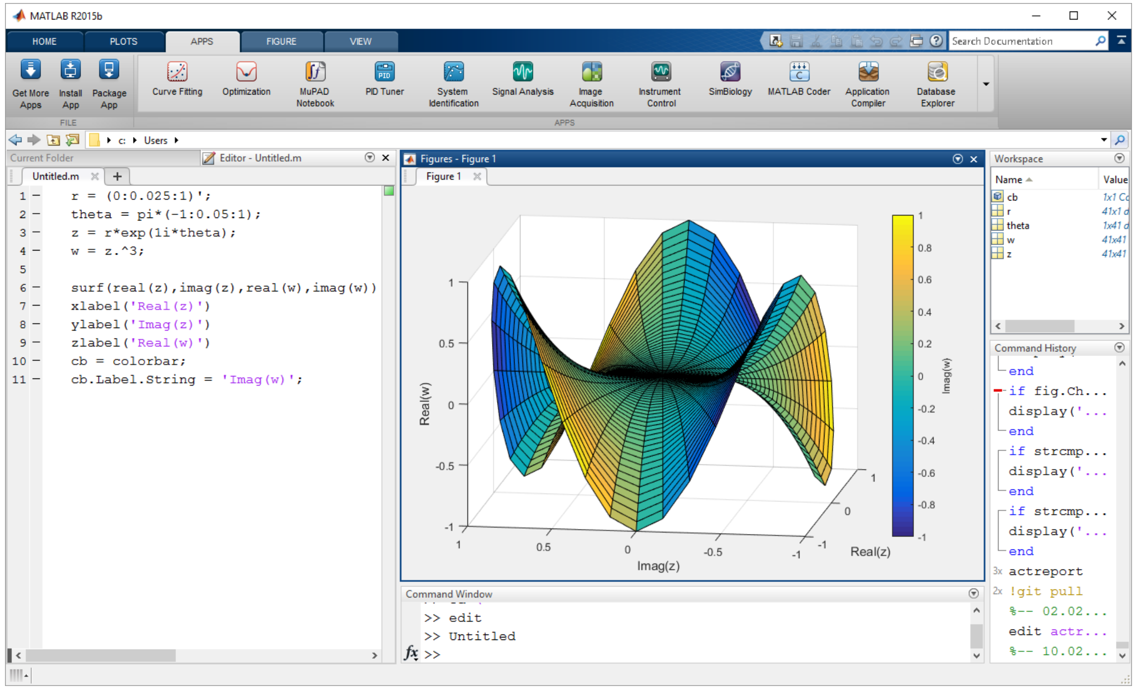
$$v = 19.62 + \left[9.81 - \frac{0.25}{68.1}(19.62)^2\right] \times 2 = 36.4137 \text{ m/s}$$

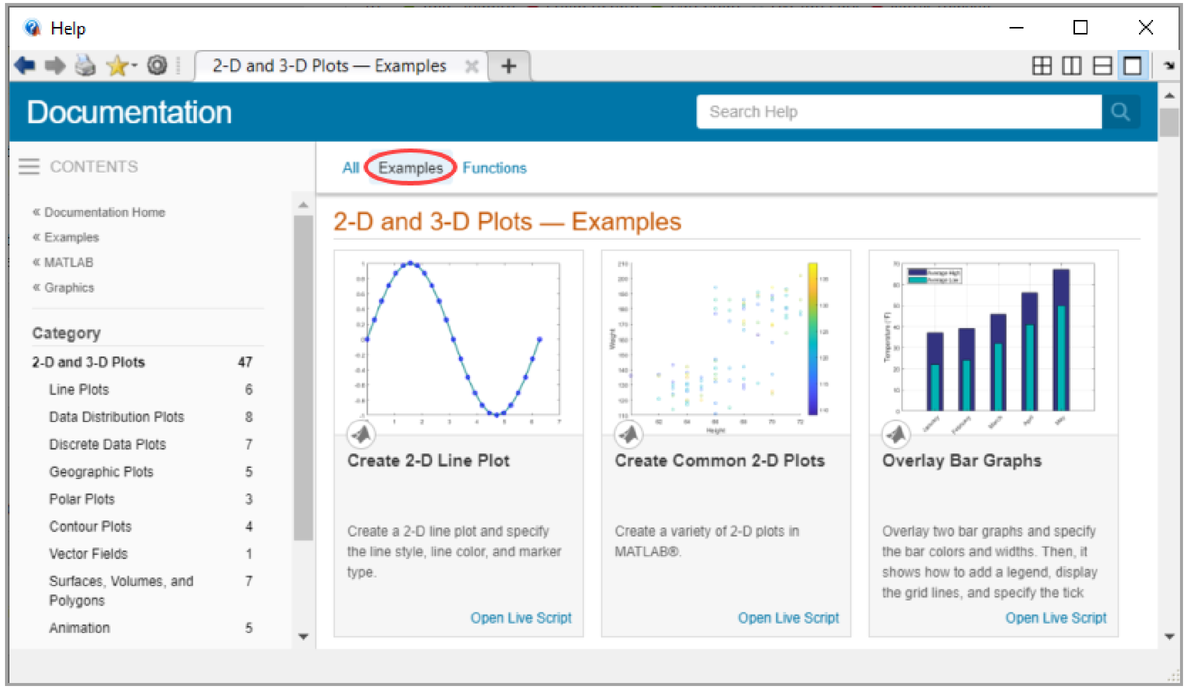
Numerical method



Comparison of the numerical and analytical solutions
for the bungee jumper problem

Matlab introduction





Matlab introduction

- Stands for MATrix LABoratory
- Interpreted language
- Scientific programming environment
- Very good tool for the manipulation of matrices
- Great visualisation capabilities
- Loads of built-in functions
- Easy to learn and simple to use

Matlab introduction

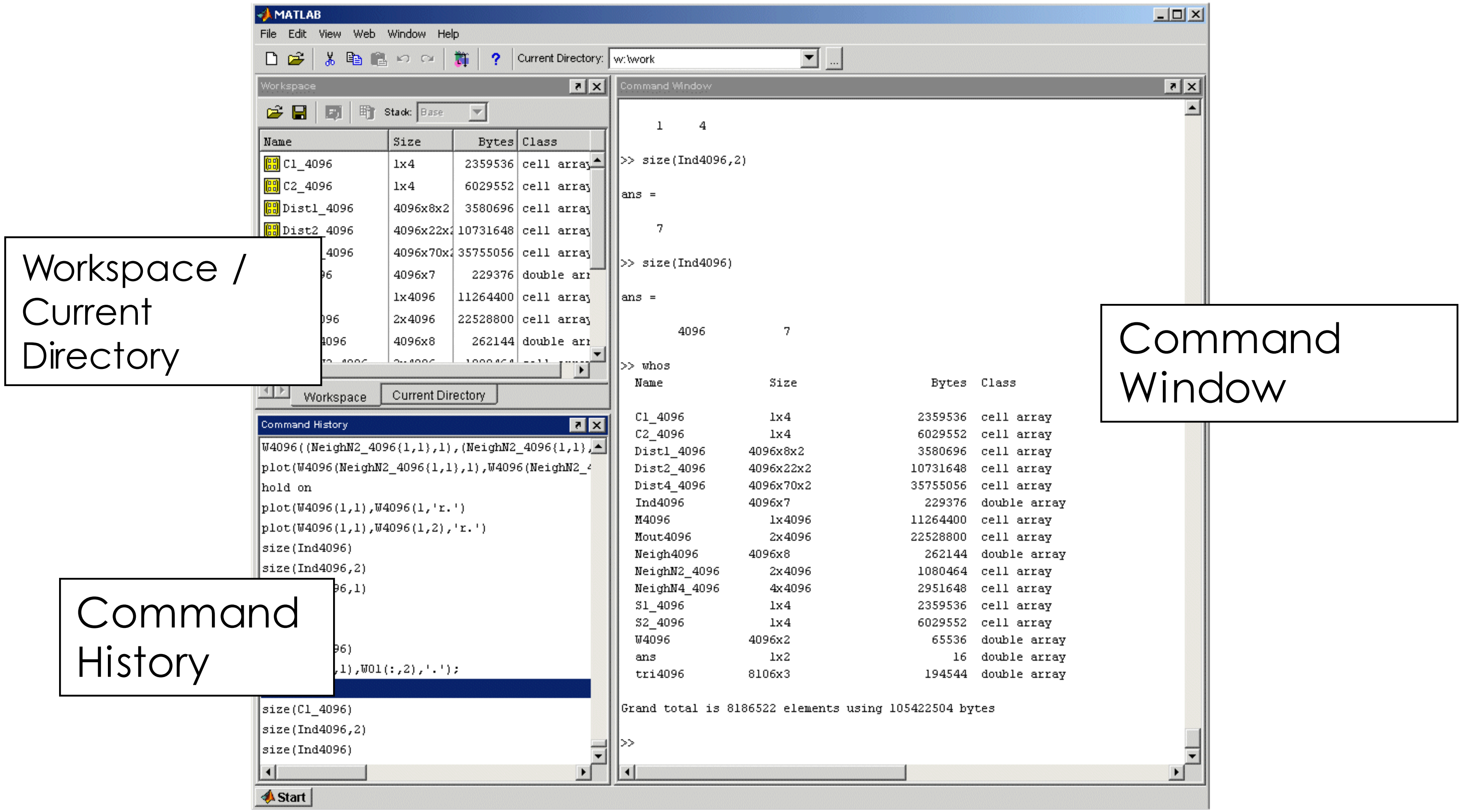
MATLAB software

- A convenient environment for performing many types of calculations
- A very nice tool to implement numerical methods

MATLAB uses three primary windows:

- Command window. Used to enter commands and data.
- Graphics window. Used to display plots and graphs.
- Edit window. Used to create and edit M-files.

Matlab Desktop



Explore the Matlab Desktop

Variables

- Don't have to declare type
- Don't even have to initialise
- Just assign in command window

```
>>
>> a=12; % variable a is assigned 12
```

Matlab prompt

assign operator

suppress command output

comment operator

Try the same line without the semicolon and comments

Variables (continued …)

- View variable contents by simply typing the variable name at the command prompt

```
>> a
a =
      12
>>
>> a*2
a =
      24
>>
```

Matlab introduction

Mathematical operations

- Type 2+3 after the >> prompt, followed by **Enter** (press the **Enter** key) as indicated by <**Enter**>:
  >> 2+3 < **Enter**>
- Commands are only carried out when you enter them. The answer in this case is, of course, 5. Next try
  >> 3–2 < **Enter**>
  >> 2*3 < **Enter**>
  >> 1/2 < **Enter**>
  >> 2ˆ3 < **Enter**>
  >> 2\1 < **Enter**>

^	Exponentiation
–	Negation
* /	Multiplication and division
\	Left division[2]
+ –	Addition and subtraction

Workspace

- The workspace is Matlab's memory
- Can manipulate variables stored in the workspace

```
>> b=10;
>> c=a+b
c =
   22
>>
```

Workspace (continued ...)

- Display contents of workspace

```
>> whos
 Name      Size                Bytes  Class
  a         1x1                 8  double array
  b         1x1                 8  double array
  c         1x1                 8  double array
Grand total is 3 elements using 24 bytes
>>
```

- Delete variable(s) from workspace

```
>> clear a b; % delete a and b from workspace
>> whos
>> clear all; % delete all variables from workspace
>> whos
```

Matlab help commands

- help

>> help whos % displays documentation for the function whos

>> lookfor convert % displays functions with convert in the first help line

- Start Matlab help documentation

>> helpdesk

Matlab introduction

- **Vector**

```
>> a = [1 2 3 4 5] or >> a = 1:5
```

and

```
>> a = [1 ; 2 ; 3 ; 4 ; 5] or   >> a = [1 2 3 4 5]'
```

- **Matrix**

```
>> A = [1 2 3; 4 5 6; 7 8 9]
>> E = zeros(2,3)
>> u = ones(1,3)
```

Matrices

- Don't need to initialise type, or dimensions

```
>>A = [3 2 1; 5 1 0; 2 1 7]
A =
    3    2    1
    5    1    0
    2    1    7
>>
```

square brackets to define matrices

semicolon for next row in matrix

Manipulating Matrices

A =

3	2	1
5	1	0
2	1	7

- Access elements of a matrix

```
>>A(1,2)
ans=
2
```

indices of matrix element(s)

- Remember Matrix(row,column)
- Naming convention Matrix variables start with a capital letter while vectors or scalar variables start with a simple letter

The : operator

- VERY important operator in Matlab
- Means 'to'

```
>> 1:10
ans =
    1    2    3    4    5    6    7    8    9    10
>> 1:2:10
ans =
    1    3    5    7    9
```

Try the following

```
>> x=0:pi/12:2*pi;
>> y=sin(x)
```

The : operator and matrices

```
>>A(3,2:3)
ans =
     1     7
>>A(:,2)
ans =
     2
     1
     1
```

```
A =
     3     2     1
     5     1     0
     2     1     7
```

What'll happen if you type A(:,:) ?

Manipulating Matrices

A =

3	2	1
5	1	0
2	1	7

B =

1	3	1
4	9	5
2	7	2

```
>> A '       % transpose
>> B*A       % matrix multiplication
>> B.*A      % element by element multiplication
>> B/A       % matrix division
>> B./A      % element by element division
>> [B A]     % Join matrices (horizontally)
>> [B; A]    % Join matrices (vertically)
```

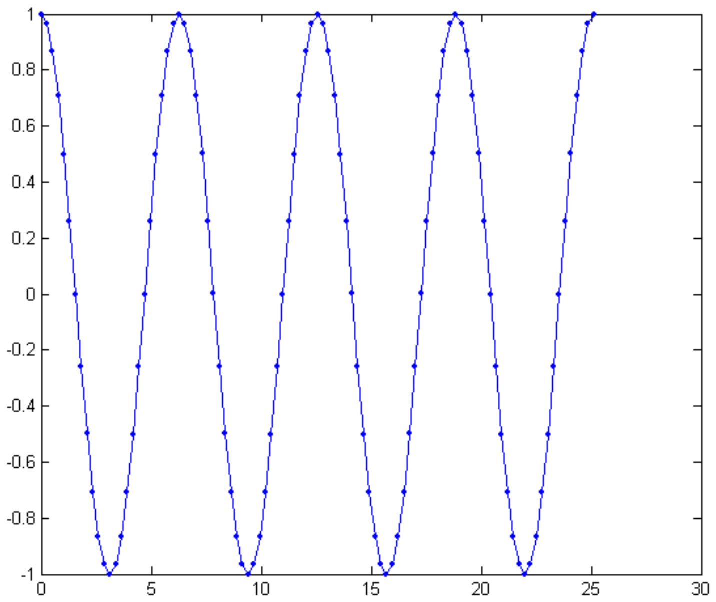
Enter matrix B into the Matlab workspace

Create matrices A and B and try out the the matrix operators in this slide

Visualisation - plotting data

```
>> figure   % create new figure
>> t=0:pi/12:8*pi;
>> y=cos(t);
>> plot(t,y,'b.-')
```

Plot style



Investigate the function
>> y=A*cos(w*t+phi);
for different values of phi (eg: 0, pi/4, pi/3, pi/2), w (eg: 1, 2, 3, 4) and A (eg: 1, 0.5, 2). Use the **hold on** Matlab command to display your plots in the same figure. Remember to type **hold off** to go back to normal plotting mode. Try using different plot styles (**help plot**)

A = amplitude
phi = phase
w = angular frequency = 2*pi*frequency

```matlab

pop_size = 1e4; % Population size

% Generate population distribution through code (can also be done using randtool GUI):
% pop_in = rand(pop_size,1); % Uniform distribution over interval (0,1)
% pop_in = 100*rand(pop_size,1); % Uniform popuation over interval (0,100)
pop_in = random('Rayleigh',2,pop_size); % Rayleigh distribution with size parameter 2
% pop_in = randn(pop_size,1); % Standard normal distribution

N1 = 5; % Sample size 1
N2 = 15; % Sample size 2
N3 = 30; % Sample size 3
N4 = 60; % Sample size 4

M = 200; % How many times to draw random samples from the population

means_N1 = zeros(M,1); % For N1
for i = 1:M % Loop over repeat drawing of samples
    tempdraw = pop_in(randperm(pop_size, N1));
    means_N1(i) = mean(tempdraw);
end

means_N2 = zeros(M,1); % For N2
for i = 1:M % Loop over repeat drawing of samples
    tempdraw = pop_in(randperm(pop_size, N2));
    means_N2(i) = mean(tempdraw);
end

means_N3 = zeros(M,1); % For N3
for i = 1:M % Loop over repeat drawing of samples
    tempdraw = pop_in(randperm(pop_size, N3));
    means_N3(i) = mean(tempdraw);
end

means_N4 = zeros(M,1); % For N4
for i = 1:M % Loop over repeat drawing of samples
    tempdraw = pop_in(randperm(pop_size, N4));
    means_N4(i) = mean(tempdraw);
end

figure, hist(pop_in,30), title('Population Distribution')
figure
subplot(2,2,1), hist(means_N1,30), title('Distribution of the mean - N = 5')
subplot(2,2,2), hist(means_N2,30), title('Distribution of the mean - N = 15')
subplot(2,2,3), hist(means_N3,30), title('Distribution of the mean - N = 30')
subplot(2,2,4), hist(means_N4,30), title('Distribution of the mean - N = 60')
```

Debugging

- Set breakpoints to stop the execution of code

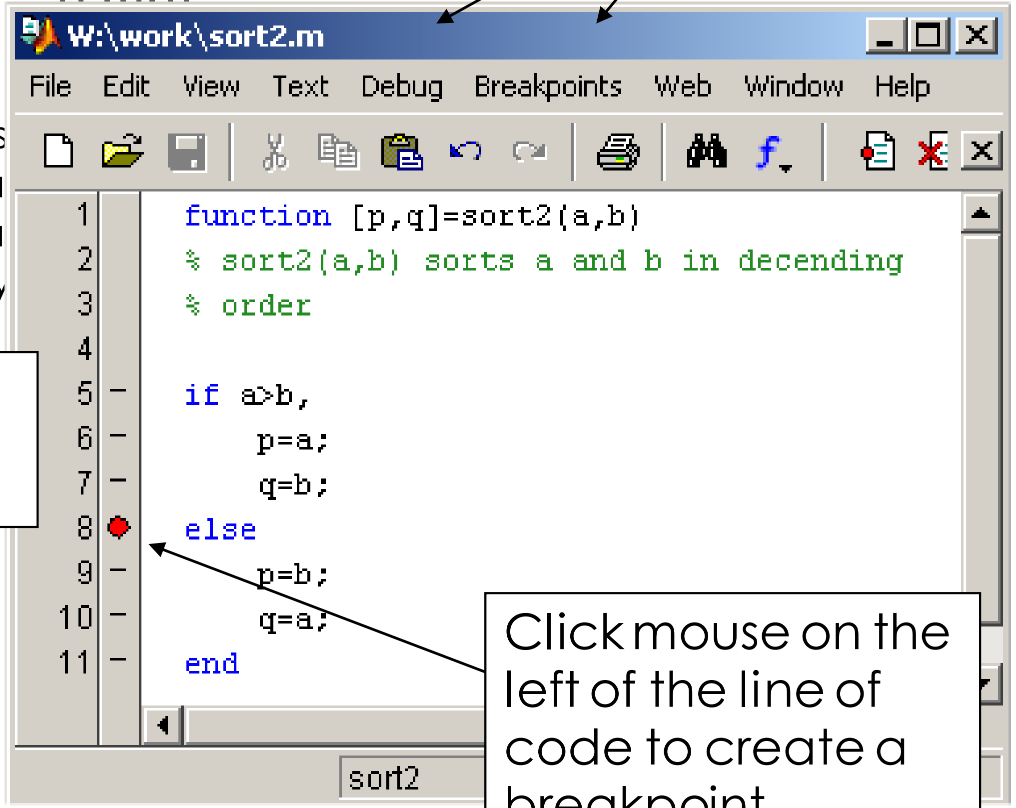
```
>> [i j]=sort2(2,4)
K>>
K>> whos
 Name      Size            Bytes Class
  a         1x1             8  double a
  b         1x1             8  double a
Grand total is 2 elements using 16 by
K>> a
a =
    2
K>> return
i =

    4
j =
    2
```

Debug menus

local function workspace

exit debug mode



Click mouse on the left of the line of code to create a breakpoint

Matlab introduction

- Some commonly used tools and commands:
  – ↑ (up arrow) returns last command input, can be repeated
  – clc – clears the screen
  – clear all – clears variables
  – close all – closes figures

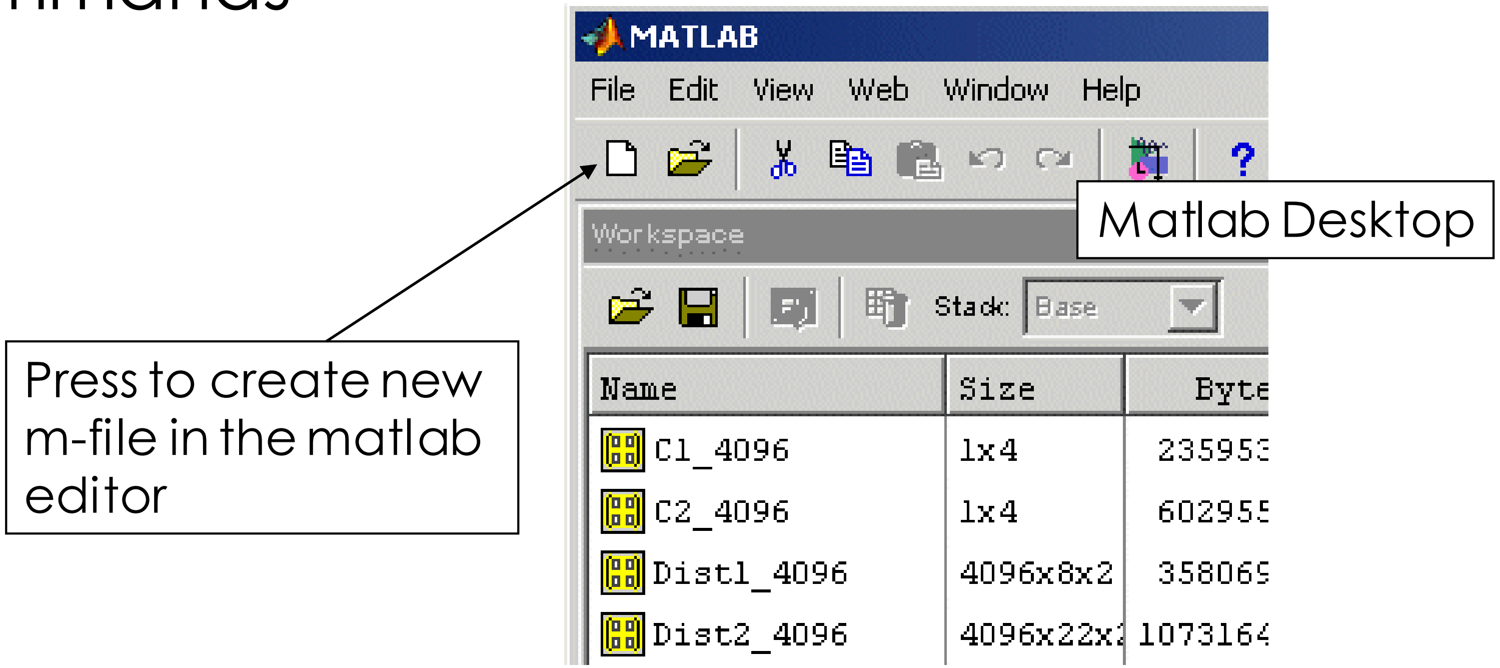
M-FILES

- An M-file consists of a series of statements that can be run all at once.
- M-files are stored with a .m extension
- M-files come in two flavors: script files and function files
  - A script file is merely a series of MATLAB commands that are saved on a file
  - Function files are M-files that start with the word *function*. In contrast to script files, they can accept input arguments and return outputs.

  languages such as *Fortran*, *Visual Basic* or C.

Scripts

- Matlab editor
- Use scripts to execute a series of Matlab commands



Function files

- The syntax for the function file can be represented generally as

function outvar = funcname(arglist)

% helpcomments

statements

outvar = value;

The M-file should be saved as funcname.m

Function files

- **Example**

function [mean,stdev] = stat(x)

n = length(x);

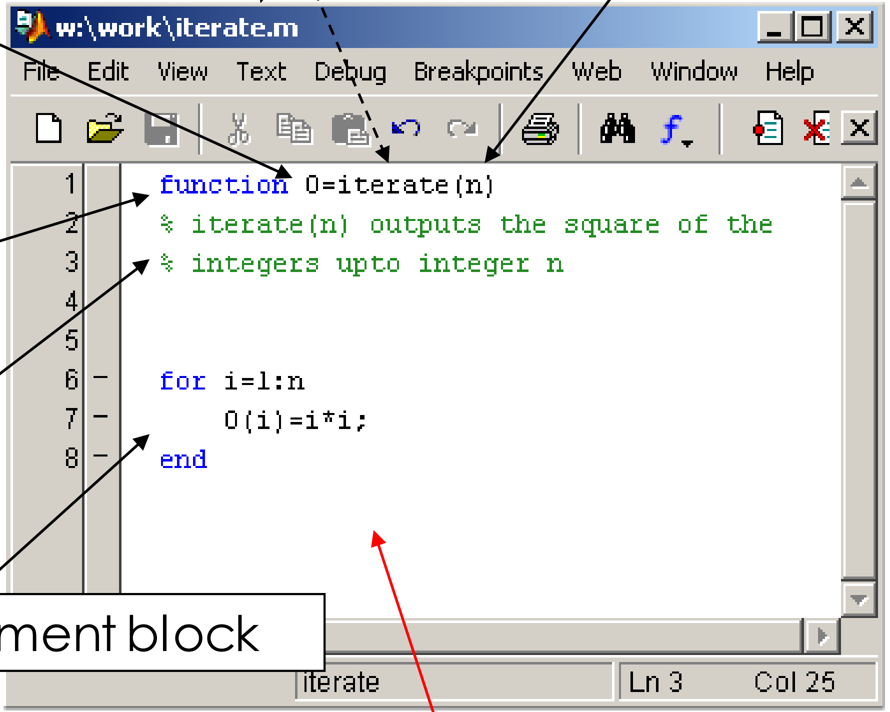
mean = sum(x)/n;

```
clc
close all
clear all
%%
x = 1:5

[mean] = stat(x)
```

Functions (continued)

```
>> I=iterate(5)
I =
     1     4     9    16    25
```



Make sure you save changes to the m-file before you call the function!

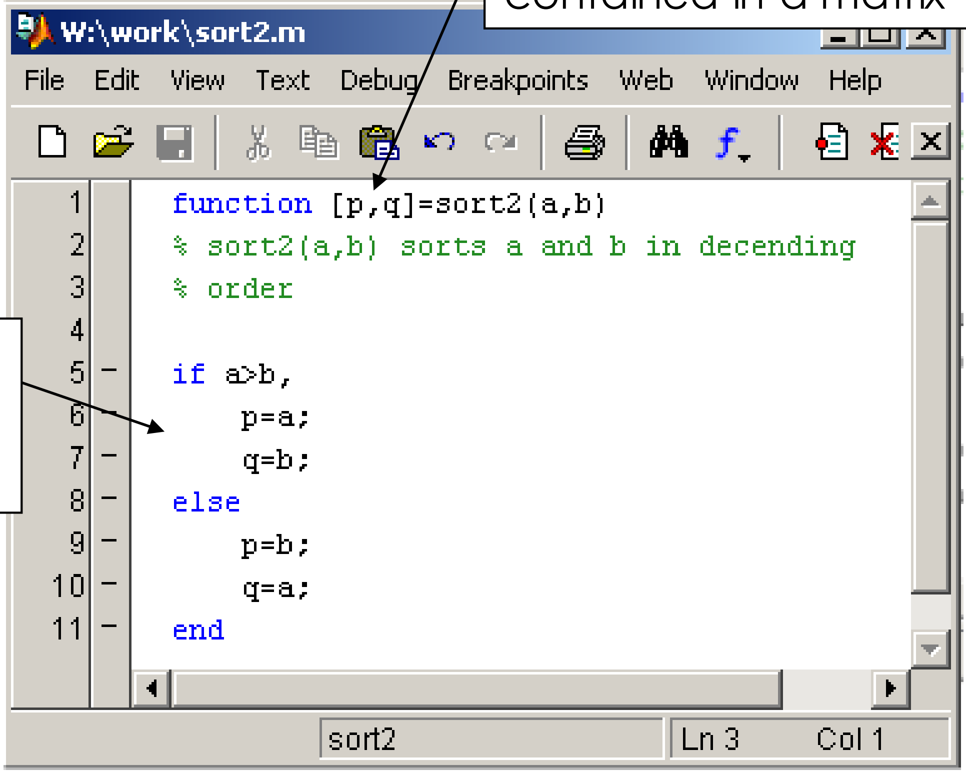
Access the comments of your Matlab functions
>> help iterate

Functions (continued)

```
>> [i j]=sort2(2,4)
i =
    4
j =
    2
>>
```

Functions can have many outputs contained in a matrix

if statement block



Remember to use the Matlab help command for syntax
>> help if

The if...else Structure

- **Structure**

if condition
Statements 1
else
Statements 2
end

```
% Example
attendance = 5;
grade_average = 15;
if ((attendance >= 6) && (grade_average >= 10))
   pass = 1
else
   fail = 1
end;
```

The for loop

- **Structure**

for index = j: k

statements

end

```
% Example
a = 0;
for i=1:10
    a=a+i;
end
```

The While loop

- **Structure**

while condition

statements

end

% Example

```
n = 10;
f = n;
while n > 1
    n = n-1;
    f = f+n;
end
```

Case study

1. Write a program to solve linear equation in matlab

$$ax + b = 0$$

Case study

- Function file

```
function x = linear_eq(a,b)
x=-b/a;
```

- Script file

```
clc
clear all
close all
%%
a = 2;
b = 6;
if a == 0
    fprintf('Coefficient of "a" must not be zero!');
else
    x = linear_eq(a,b)
end
```

Exercises

1. Write a program to solve quadratic equation in matlab

$$ax^2 + bx + c = 0$$

2. And cubic equation

$$ax^3 + bx^2 + cx + d = 0$$

3. Write a program to produces a vector containing the first *n* Fibonacci numbers
   - Group 1 : using While loop
   - Group 2+3 : using For loop

$$1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89,\ 144,\ \ldots$$

Method of solution

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Dividing $ax^3 + bx^2 + cx + d = 0$ by a and substituting $t - \frac{b}{3a}$ for x we get the equation

$$t^3 + pt + q = 0$$

where

$$p = \frac{3ac - b^2}{3a^2},$$
$$q = \frac{2b^3 - 9abc + 27a^2 d}{27a^3}.$$

Method of solution

- Iterative method
  1. Bracketing method
     - Bisection Method
     - Regula Falsi Method
  2. Open end method
     - Newton Raphson Method
     - Secant Method
     - Fixed point Method

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Dividing $ax^3 + bx^2 + cx + d = 0$ by a and substituting $t - \frac{b}{3a}$ for x we get the equation

$$t^3 + pt + q = 0$$

where

$$p = \frac{3ac - b^2}{3a^2},$$
$$q = \frac{2b^3 - 9abc + 27a^2 d}{27a^3}.$$

1. Make Sure the Function Name Matches the File Name

You establish the name for a function when you write its function definition line. This name should always match the name of the file you save it to. For example, if you create a function named curveplot,

function curveplot(xVal, yVal)

then you should name the file containing that function curveplot.m

If not: error!

"Undefined function or variable curveplot"

Octave introduction

Octave online:

- https://rextester.com/l/octave_online_compiler
- https://www.tutorialspoint.com/execute_matlab_online.php