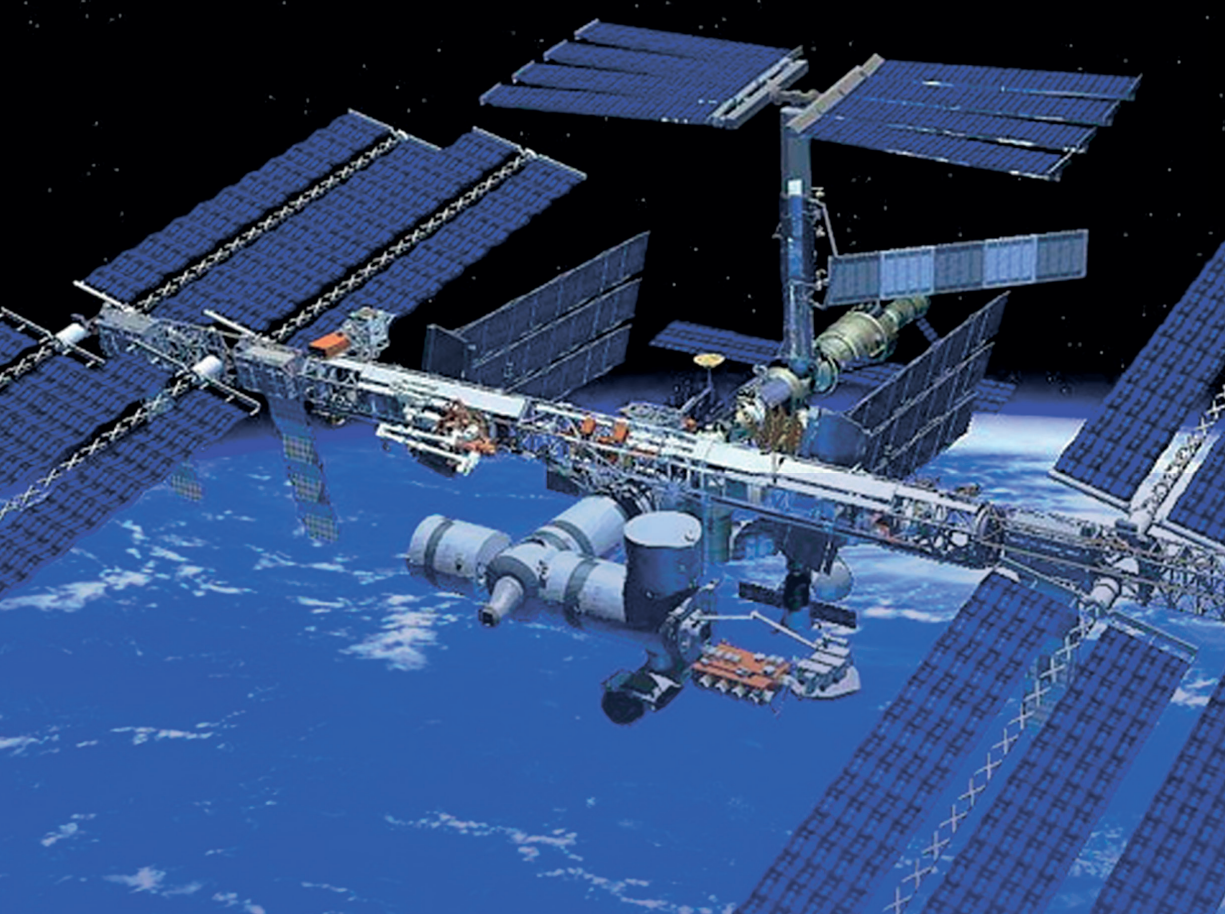
cupress

Numerical Methods & Computer Aided Engineering

with MATLAB & ANSYS



Pramote Dechaumphai

Numerical Methods and Computer Aided Engineering

with MATLAB & ANSYS

Numerical Methods and Computer Aided Engineering

with MATLAB & ANSYS

Pramote Dechaumphai

Chulalongkorn University press

2023

390.-

Dechaumphai, Pramote

Numerical Methods and Computer Aided Engineering with MATLAB & ANSYS / Pramote Dechaumphai

1. Computer-aided engineering. 2. Engineering -- Data processing. 3. MATLAB.

4. ANSYS (Computer system).

620.00420285

ISBN (e-book) 978-974-03-4210-6

CUP. 2598

Knowledge to All

www.cupress.chula.ac.th

Copyright © 2023 by Chulalongkorn University Press

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system,

Or transmitted in any form or by any means, i.e. electronic, mechanical, photocopying recording, etc.

Without prior written permission of the publisher.

Published by **Chulalongkorn University Press**

First edition 2023

www.cupress.chula.ac.th [CUB6512-001K]

Tel. 0-2218-3562-3

Managing editor : Prof. Dr.Aran Hansuebsai Mrs.Orathai Nanthanadisai

Academic Editorial Department : Emeritus Professor Dr.Piyanart Bunnag

Assoc. Prof. Dr.Pimpan Dachakupt

Assoc. Prof.Chitsanu Pancharoen

Assoc. Prof. Dr.Vimolvan Pimpan

Coordinator : Wasana Sumsen

Proof reader : Professor Dr.Pramote Dechaumphai, Tassanee Phewkham

Artwork and Cover : Kaiumporn Phongkhachorn

Contact : **Chulalongkorn University Book Center**

Phyathai Road, Pathumwan District, Bangkok 10330, Thailand

http://www.chulabook.com

Tel: 08-6323-3703-4

customer@cubook.chula.ac.th, info@cubook.chula.ac.th

Apps: CU-eBook Store

Preface

Numerical Methods and Computer Aided Engineering (CAE) play important parts in product designs and analyses nowadays. Therefore, fundamental understandings of engineering mathematics and computational methods are required. Engineers should then have a solid background of numerical methods and CAE prior to fully benefits from any engineering analysis software.

This book "Numerical Methods and Computer Aid Engineering" contains fourteen chapters starting from an overview of numerical methods and CAE to various engineering applications. Key MATLAB topics and coding examples are also introduced to aid readers in comprehending underlying computational processes in CAE programs. This includes details of finite difference method to solve the boundary and initial value problems, and finite element method to analyze one- and two-dimensional problems. Besides computation processes and programing, the book also presents examples using CAE softwares namely, 1) the finite element toolbox embedded in MATLAB to solve elliptic, parabolic and hyperbolic differential equations, and 2) software to analyze the heat transfer, structural and fluid flow problems. Procedures for using ANSYS are demonstrated, step by step, in details. With ranges of topics and depth, the aim of this book is to allow readers to gain sounded understanding of CAE.

The author would like to thank Dr. Edward Dechaumphai for proof-reading and Miss Kaiumporn Phongkhachorn for preparing the manuscript. He would like also to thank the Chulalongkorn University Press for publishing this book.

Pramote Dechaumphai

Contents

Part I

CAE Fundamentals

Chapter 1

Computer Aided Engineering

1.1 Introduction

Computer Aided Engineering (CAE) plays an important role in engineering education nowadays. This is because CAE are being used to analyze multitude of problems with wide ranges of complexities as compared to the traditional analytical methods. CAE, therefore, becomes an integral part of university curriculum to help students develop practical skills in solving problems in solid mechanics, heat transfer, fluid flows, electromagnetics, etc. Designers and engineers in many industrial sectors employ CAE software packages in their product design processes to shorten design cycle time and prototyping costs while increasing the productivity and product reliability. The trial-and-error process, based solely on intuition of designers and engineers, can be minimized or eliminated.

CAE is based on engineering mathematics together with the application of numerical methods. Results of the computed solutions are converted and displayed graphically so that they can be understood easily. However, fundamentals of CAE are required for engineers and designers to have confident and correctness in the validity of output solutions.

1.2 What is CAE

CAE is mainly based on the knowledge of mathematics and numerical methods. Basic mathematical and engineering understandings such as familiarity with governing differential equations that represent the problem is required. For an example of a fluid flow problem, mass and momentums must be conserved at any location in the flow field. Such conservations are expressed in form of the partial differential equations, which is the fundamentals in fluid mechanics. This means users should have a solid background in understanding mathematics and governing equations, along with their physical meanings. By employing a numerical method, these partial differential equations are transformed into a large set of algebraic equations. A computer program then solves these algebraic equations for the flow solutions and solutions can be displayed graphically.

Similarly, the equilibrium equations must be understood prior to analyzing a solid mechanic problem. These equilibrium equations are again in form of the partial differential equations as seen in many solid mechanic textbooks. A numerical method transforms these differential equations into their corresponding algebraic equations. A computer program then solves such algebraic equations for the deformed shapes and stresses that occur in the structure.

CAE users should have good backgrounds in mathematics and physics of the problem being solved. This includes having a good background of the numerical methods prior to use any CAE software package. Engineers and designers can then evaluate the correctness and reliability of the solutions generated by the

software. This is one of the main reasons that most universities are offering the CAE course to engineering students.

Methods for finding solutions can be categorized into two types:

(a) *Analytical Method*. The analytical method herein refers to a mathematical technique used to find an exact or analytical solution for a given problem. The technique can provide solutions only for simple problems as taught in courses where differential equations, boundary conditions and geometries are not complicated. Most problems are limited to one dimensional problems so that their governing equations can be simplified from partial to ordinary differential equations.

(b) *Numerical Method*. If the differential equations, boundary conditions and geometry of a given problem are complicated, solving with analytical method is not feasible. An approximate solution from a numerical method can be obtained instead. There are many numerical techniques for finding solutions to complex problems. The popular techniques widely used are the finite element and finite volume methods. This is mainly because both techniques can handle problems with complex geometry effectively.

The numerical methods transform the governing differential equations into algebraic equations. In this process, many numerical techniques are needed. The techniques include solving a large set of algebraic equations, applying the interpolation function concepts, determining derivatives and integrations of functions numerically, etc. Details of these techniques are taught in numerical method courses and can be found in many introductory numerical method textbooks.

1.3 CAE Software

There are many CAE software packages today that are widely used in universities and industries by students, researchers and engineers. Popular software packages are ANSYS, Nastran,

SolidWorks, COMSOL, Altair, LS-DYNA, Abaqus and COSMOS. General procedures employed in these CAE packages are as follows:

(a) *Pre-processing*. The first step is to create domain or geometry of the problem. This step normally requires a lot of time especially when the domain is complex such as the automotive engine or domain of flow in an office. The constructed domain is discretized into a number of cells or elements. Proper boundary and initial conditions are then applied onto the domain.

(b) *Analysis*. With a discretized model and applied conditions, the analysis of a model can be performed. There are many analysis methods that could be employed depending on the problems. For instance, the simplest analysis method is the finite difference method. The finite element method is more popular and convenient when the domain geometry is complicated. The finite volume method is frequently used to analyze fluid flow problems. The computational time for analyzing a problem depends on the analysis method, the problem size and the differential equations that govern the problem.

(c) *Post-processing*. The last step is to display the results. The computed solutions from the second step are in form of numbers. These numbers are converted into colors and displayed directly on the computer screen so that they can be understood easily.

The general procedures above suggest that users must understand basic framework underlying the CAE software packages in order to verify the solutions and realize the limitations of different methods.

1.4 Types of Governing Equations

Some backgrounds of mathematics are required prior to using the CAE software packages. Results from the software packages are obtained by solving the differential equations that govern the problems. For examples, the deflection and stresses that occur in a frame structure are obtained by solving the equilibrium equations of the structure. These equilibrium equations are in form of the partial differential equations appearing in many structural

textbooks. Another example is the flow behavior of air passing over buildings are obtained by solving the Navier-Stokes equations that represent conservation of mass and momentums in fluid mechanics. The temperature distribution in a fin is obtained by solving the energy equation representing the conservation of energy learned in the heat transfer course. Most problems are governed by many differential equations while some problems are governed by a single differential equation. To study how CAE works, it is easier to start with a single differential equation. Governing differential equations can be classified into three different types as followed:

(a) Elliptic equation,

$$-\left(\frac{\partial}{\partial x}\left(c\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(c\frac{\partial u}{\partial y}\right)\right)+au = f$$

where c, a and f are constants or function of x and y. The unknown u depends on the domain coordinates x and y. The elliptic equation describes many engineering problems such as steady-state heat transfer, potential flow, stresses in solids, etc.

(b) Parabolic equation,

$$d\frac{\partial u}{\partial t}-\left(\frac{\partial}{\partial x}\left(c\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(c\frac{\partial u}{\partial y}\right)\right)+au = f$$

where d is a constant or function of x and y. The variable u depends on the domain coordinates x, y and time t. The parabolic equation describes many engineering problems such as transient heat transfer, unsteady flow, etc.

(c) Hyperbolic equation,

$$d\frac{\partial^2 u}{\partial t^2}-\left(\frac{\partial}{\partial x}\left(c\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(c\frac{\partial u}{\partial y}\right)\right)+au = f$$

The hyperbolic equation describes many engineering problems such as vibration of motors, bridges, frame structures, etc. Note that the forms of the parabolic and hyperbolic equations are quite similar but their solution behaviors are completely different.

Solutions obtained from solving the above differential equations also depend on the boundary conditions which may consist of:

(a) Dirichlet condition by prescribing the value of u on the boundary,

$$h\,u \;=\; r$$

where h and r are constants or function of x and y.

(b) Neumann condition by prescribing the gradient of u on the boundary,

$$c\frac{\partial u}{\partial n} + q\,u \;=\; g$$

where c, q and g are constants or function of x and y. In the above equation, n is the unit normal vector to the boundary.

Solving the parabolic equation requires the initial condition of $u(x, y, 0)$ at time $t = 0$, while the initial conditions of both $u(x, y, 0)$ and $\partial u/\partial t(x, y, 0)$ at time $t = 0$ are required for solving the hyperbolic equation.

For a one-dimensional problem, the elliptic partial differential equation above is reduced to the ordinary differential equation,

$$-\frac{d}{dx}\left(c\frac{du}{dx}\right) + a\,u \;=\; f$$

where $u = u(x)$ depends only on the x-coordinate. Such differential equation is normally used as the first step towards learning CAE. This is because the algebraic equations can be easily derived by using the finite difference, finite element or the finite volume method. In addition, exact solutions are available for most problems subjected to various types of boundary conditions. Thus, accuracy of the approximate solutions obtained from the numerical methods can be quantified and compared. Studying on how to solve the ordinary elliptic differential equation is highly recommended as the first step prior to using the CAE software packages to analyze other complex problems.

1.5 Numerical Methods and Computer Programming

In the process of transforming the differential equations into the corresponding algebraic equations, the knowledge of the finite difference, finite element or finite volume methods is required. These methods also based on the understanding of numerical method fundamentals, such as the concept of interpolation functions, numerical differentiation and integrations, etc. Depending on the size of problems, the set of algebraic equations arise from these methods may consist of a large number of equations. These equations are solved for the solutions at the discretized grid points. Solving a large set of algebraic equations, some numerical method backgrounds such as the Gauss elimination or conjugate gradient methods are required so that these equations can be solved effectively. In addition, if the governing differential equations are nonlinear, such as those arise in fluid flow problems, an iterative method for solving a set of nonlinear algebraic equations is also needed.

Experience in computer programming can significantly help reducing the effort for solving these problems. There are many computer languages that could be used for developing computer programs. Popular computer languages normally used for solving CAE problems are Fortran, C, MATLAB, Mathematica and Python languages. Some of these languages contain built-in functions that can be incorporated directly into the programs, where the embedded built-in functions are efficient and can reduce computational times. They can also reduce the debugging time when developing the programs.

1.6 Advantages of CAE

CAE software packages are widely used by scientists and engineers all over the world for analyzing various types of problems. Examples of problems are as follows.

(a) Stress analyses of large-scale structures such as bridges, ships, trains, aircrafts, automobiles and buildings. Structural

analysis for small-scale products are such as automotive and electronic parts, furniture, machine equipment, etc.

(b) Vibration and dynamic analyses of high-voltage power transmission towers, expressway signs under strong wind, crash simulation of automobiles, turbine blades operating under high pressure and temperature, etc.

(c) Fluid analyses of air flows over cities, air ventilation in large halls, inside offices, cleanrooms, computer cases, etc.

(d) Electromagnetic analyses around power transmission lines, electric motors, sensitive electronic devices, etc.

(e) Bio-mechanic analyses of blood flow in human hearts and veins, artificial joints and bones, etc.

(f) Analyses of other problems in which experiments are dangerous to human or too costly for multiple tests, such as hazardous chemical reaction in gas chambers, prediction of bomb explosion phenomena, flow field around hypersonic aerospace plane, etc.

With the applications of CAE, many engineering institutes recommend CAE as a required course so that students will have good backgrounds on this important tool and able to use it correctly to analyze engineering problems after graduation.

1.7 Conclusion

This chapter explains the need of CAE for education and industrial applications. CAE is widely used to provide better understanding of the problem behavior. CAE can analyze different types of problems for solutions that could not be done by the analytical method. Many universities have thus included CAE as a required course for engineering students.

CAE is based on mathematics and numerical methods. The finite difference method is one of the early CAE methods which is easy to understand. The finite element and finite volume methods are more complicated and widely used nowadays. Several CAE software packages have been developed and commercialized. Some of them are popular and widely used in both academic and industrial sectors. Results from CAE software packages are in form of colors representing the problem solutions such as the stresses, temperature or flow behavior. Without understanding mathematics and numerical methods behind the software, users might not be able to verify the results obtained. It is thus important to have good background of mathematics and numerical methods prior to using any CAE software package.

The following chapters explain fundamentals of CAE and numerical methods so that users will be able to use the software with confidence. Essential mathematics and numerical methods necessary for learning CAE will be provided. Popular and easy-to-use software packages will be introduced and explained. Readers are encouraged to follow examples as well as to practice with exercises at the end of the chapters. This will improve understanding of the theories behind CAE software and increase experience in using the software packages. The goal of this book is to layout core understanding of CAE for various applications including basics of mathematics, engineering, MATLAB and ANSYS.

Chapter 2

MATLAB Essentials

2.1 Introduction

MATLAB software is widely used and suitable for learning CAE. New computer programs can be developed and debugged easily. MATLAB software contains many built-in functions of the numerical method techniques, so there is no need to redevelop them. These built-in functions are quite efficient and can be incorporated directly into the programs. MATLAB also includes high-level graphic capability, the computed solutions can be displayed conveniently.

In this chapter, essential MATLAB commands needed for developing CAE programs are explained. The commands are for operating vectors and matrices, plotting graphs in two and three dimensions, developing new programs, solving a large set of algebraic equations, finding numerical solutions of the differential equations. Symbolic mathematic capability in MATLAB software is also introduced.

2.2 Commands and Basic Functions

MATLAB can be used as a calculator to add (+), subtract (-), multiply (*), divide (/) and raise a power (^) of numbers as in following examples,

```
>> 78.16 + 12.45
ans =
   90.6100
>> 42.53 - 61.87
ans =
  -19.3400
>> 91.23 * 45.89
ans =
   4.1865e+03
>> 91.12 / 65.34
ans =
    1.3946
>> 5 ^ 4
ans =
   625
```

New variable can be defined and used later, e.g.,

```
>> a = 5
a =
     5
>> a + 3
ans =
     8
```

Several variables can be defined on the same line separated by comma (,),

```
>> a = 3, b = 4, c = 5
a =
     3
b =
     4
c =
     5
>> a + b + c
ans =
    12
```

Note that the capital and lower case of a letter are different,

```
>> d = 7, D = 2
d =
     7
D =
     2
>> d + D
ans =
     9
```

The semicolon (`;`) is used to suppress the output,

```
>> e = 4; g = 6;
>> e * g
ans =
    24
```

The order of mathematical operations follows the standard way, i.e., the priority is from raising a power, multiplication and division, addition and subtraction, respectively. As an example,

```
>> 10 - 3 ^ 2 * 2 / 3 + 7
ans =
    11
```

Parentheses can help clarifying the order of mathematical operations, e.g.,

$$5^3\left(\frac{3}{5}+\frac{9}{2^3}\right)$$

```
>> (5^3)*((3/5) + (9/(2^3)))
ans =
  215.6250
```

MATLAB contains a large number of mathematical functions, such as the trigonometric, logarithmic and root functions. As an example, the sine function of a 30 degree angle is determined by using the command,

```
>> sin(pi/6)
ans =
    0.5000
```

where `pi` represents the value of pi. Note that the angle appeared in the argument above must be in radian,

```
>> cos(pi/6)
ans =
    0.8660

>> tan(pi/4)
ans =
    1.0000
```

The angle can be input in degrees by using slightly different function names,

```
>> sind(30)
ans =
    0.5000

>> cosd(90)
ans =
     0

>> tand(45)
ans =
    1.0000
```

The inverse trigonometric functions can also be determined conveniently, e.g.,

```
>> asin(0.5)
ans =
    0.5236

>> acos(-0.3)
ans =
    1.8755

>> atan(1.5)
ans =
    0.9828
```

Similarly, the hyperbolic functions can be determined, e.g.,

```
>> sinh(3)
ans =
   10.0179

>> cosh(2)
ans =
    3.7622
```

```
>> tanh(4)
ans =
    0.9993
```

Other mathematical functions normally encountered in computation are shown by the following examples. Starting from the exponential function,

```
>> exp(3)
ans =
   20.0855
```

The natural logarithmic function,

```
>> log(1)
ans =
     0
```

and the logarithmic function with base 10,

```
>> log10(10)
ans =
     1
```

Square root of a number is determined by the command `sqrt`, e.g.,

```
>> sqrt(7)
ans =
     2.6458
```

Also the root of other orders, e.g.,

```
>> 8^(1/3)
ans =
     2
```

Many solutions in the above examples contain four decimal points. If more decimal points are needed, the `format long` command must be first declared,

```
>> format long
```

Solutions onward will contain 15 decimal points, as an example,

```
>> sqrt(6)
ans =
    2.449489742783178
```

The `format short` command is used if four decimal point solution is preferred,

```
>> format short
```

Numbers in the engineering format can also be displayed,

```
>> format short e
>> sqrt(6)
ans =
    2.4495e+000
```

```
>> format long e
>> sqrt(6)
ans =
    2.449489742783178e+000
```

In addition, rational numbers can be acquired, e.g.,

```
>> format rat
>> 32.5*28.56
ans =
    4641/5
```

2.3 Vectors and Matrices

MATLAB stands for "MATrix LABoratory" since the variables are stored in the matrix form inside the software. As an example, if `a = 4`, the variable `a` is stored as a matrix with dimensions of 1×1 containing only one element of `4`.

There are two types of vectors, row and column vector. A row vector of `a` containing three elements can be defined as an example,

```
>> a = [1 2 3]
a =
     1     2     3
```

Similarly, a column vector is defined by adding semicolon (`;`) between elements,

```
>> b = [4; 5; 6]
b =
     4
     5
     6
```

A vector can be transposed by simply including the prime symbol (`'`) at the end of the vector,

```
>> c = [7 8 9]'
c =
     7
     8
     9
```

A row vector containing many elements can be created easily. As an example, creating a vector that contains values from 1 to 17 with increment of 2,

```
>> d = [1:2:17]
d =
    1    3    5    7    9   11   13   15   17
```

Note that the increment can be negative or a decimal number,

```
>> e = [15:-3:1]
e =
    15    12     9     6     3

>> f = [5:-0.25:3]
f =
  Columns 1 through 6
    5.0000 4.7500 4.5000 4.2500 4.0000 3.7500
  Columns 7 through 9
    3.5000 3.2500 3.0000
```

Elements in a vector can be selected for further use, e.g.,

```
>> b(2) + 5
ans =
    10
```

Number of elements in a vector can be determined by the `length` command,

```
>> length(v)
ans =
     5
```

The `min`, `max` and `mean` commands are used to determine the minimum, maximum and mean values of elements in a vector, respectively,

```
>> min(v)
ans =
     7

>> max(v)
ans =
    15

>> mean(v)
ans =
    11
```

A matrix with dimensions of $m \times n$ can be assigned, as examples,

```
>> A = [1 3 5; 7 2 6; 4 9 8]

A =
     1     3     5
     7     2     6
     4     9     8

>> B = [1 4 7 3 9 0; 3 2 6 4 8 5]
B =
     1     4     7     3     9     0
     3     2     6     4     8     5
```

The `zero` command is used to create an $n \times n$ matrix for which all elements are zero,

```
>> C = zeros(3)
C =
     0     0     0
     0     0     0
     0     0     0
```

The `ones` command is used to create an $n \times n$ matrix where all elements are one,

```
>> D = ones(5)
D =
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
```

An identity matrix with dimensions of $n \times n$ is created by the `eye` command,

```
>> E = eye(4)
E =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

Elements in a matrix can be selected,

```
>> B(2,3)
ans =
     6
```

Elements for the entire row or column of a matrix can be selected too,

```
>> A(1,:)
ans =
     1     3     5
>> A(:,3)
ans =
     5
     6
     8
```

Vector operations are performed in the standard ways. As examples,

```
>> u = [1 5 6];
>> v = [3 7 2];
```

Vector addition and subtraction,

```
>> w = u + v
w =
     4    12     8
>> w = u - v
w =
    -2    -2     4
```

Dot and cross products of vectors,

```
>> w = dot(u,v)
w =
    50
>> w = cross(u,v)
w =
   -32    16    -8
>> w = cross(v,u)
w =
    32   -16     8
```

Matrix operations of addition, subtraction and multiplication are performed in the same ways. Here, the matrices `M` and `N` both have dimensions of 2×3, while the matrix `P` has dimensions of 3×2.

```
>> M = [3 4 6; 8 7 2];
>> N = [1 0 5; 9 6 3];
>> P = [2 3; 1 5; 4 9];
```

Matrices `M` and `N` can be added or subtracted together. Such operations can't be performed between matrices `M` and `P` because they have different dimensions.

```
>> M + N
ans =
     4     4    11
    17    13     5
>> N - M
ans =
    -2    -4    -1
     1    -1     1
>> M + P
??? Error using ==> plus
Matrix dimensions must agree.
```

Multiplication of two matrices must obey the standard rule as shown in the following examples.

```
>> P * M
ans =
    30    29    18
    43    39    16
    84    79    42
>> M * P
ans =
    34    83
    31    77
>> M * N
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

The `det` command is used to determine the determinant of a matrix,

```
>> F = [1 4 7; 2 5 6; 3 4 8];
>> det(F)
ans =
  -25.0000
```

Matrix inversion can be determined conveniently by using the `inv` command,

```
>> inv(F)
ans =
   -0.6400    0.1600    0.4400
   -0.0800    0.5200   -0.3200
    0.2800   -0.3200    0.1200
```

2.4 Plotting Graphs

MATLAB contains graphic commands that can be used to plot graphs in various ways. The computed numerical solutions from a CAE software can be displayed directly on computer screen so that they can be understood quickly. The solutions can be displayed in forms of the *x*-*y*, contour line and color fringe plots.

To perform an *x*-*y* plot, let's consider the two vectors containing *y* data at different *x* locations,

x = [0 1 2 3 4 5 6]

and y = [0 1.175 1.576 1.141 0.577 0.707 1.721]

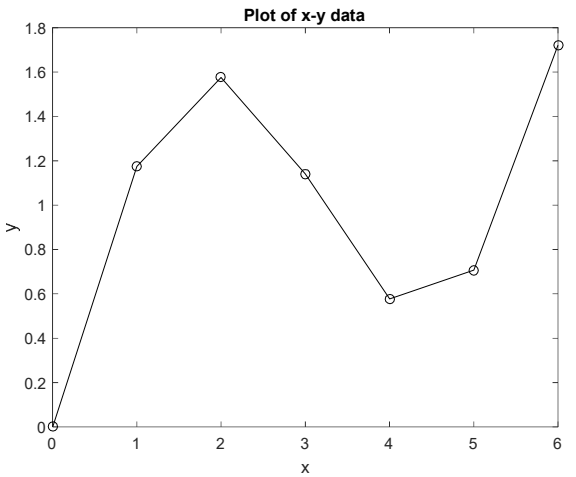
An *x*-*y* plot can be created by using the commands as follows,

```
x = [0 1.000 2.000 3.000 4.000 5.000 6.000];
y = [0 1.175 1.576 1.141 0.577 0.707 1.721];
plot(x, y, '-ok')
title('Plot of x-y data')
xlabel('x'); ylabel('y');
```

leading to the plot as shown in the figure,

The key command for generating the *x*-*y* plot in the figure is,

```
plot(x, y, '-ok')
```



The `plot` command uses the data provided in the *x* and *y* vectors to place points in the graph and connect these points by straight lines. The symbols `-ok` inside the quote `' '` indicate the use of black (`k`) straight lines (`-`) between open circles (`o`). The line pattern can be changed from `-` to `--`, `:` and `-.`, if preferred. The open circles can also be changed to other styles such as `+`, `*`, `X`, `s` (square) and `d` (diamond). Line color in black (`k`) can also be changed to red, green, blue, magenta and yellow by using the symbols `r`, `g`, `b`, `m` and `y`, respectively. In addition, the line thickness and symbol size can be increased or reduced through the commands `linewidth` and `markersize`.

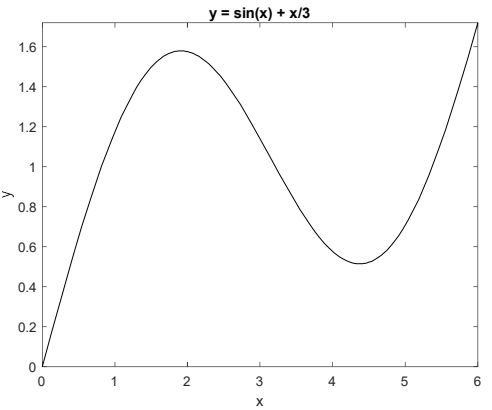
It is noted that data in the *x* and *y* vectors above are generated from the equation,

$$y = \sin x + \frac{x}{3} \qquad 0 \le x \le 6$$

MATLAB contains the `fplot` command that can be used to plot a function directly,

```
fplot(@(x)sin(x)+x./3, [0,6], 'k')
title('y = sin(x) + x/3');
xlabel('x'); ylabel('y');
```

leading to the plot as shown in the figure.



Data points and smooth curve can be displayed on the same graph,

```
fplot(@(x)sin(x)+x./3, [0,6], 'k')
title('y = sin(x) + x/3');
xlabel('x'); ylabel('y');
hold on
x = [0:1:6]
y = sin(x) + x./3
plot(x, y, 'ok')
```

The `hold on` command retains the former plot so that a new plot can be superimposed on it.