1

Introduction

This short introductory chapter is divided into two parts. In the first part there is an overview of the mechanics of fiber-reinforced composite materials. The second part includes a short tutorial on MATLAB.

1.1 Mechanics of Composite Materials

There are many excellent textbooks available on mechanics of fiber-reinforced composite materials like those in [1–12]. Therefore this book will not present any theoretical formulations or derivations of mechanics of composite materials. Only the main equations are summarized for each chapter followed by examples. In addition only problems from linear elastic structural mechanics are used throughout the book.

The main subject of this book is the mechanics of fiber-reinforced composite materials. These materials are usually composed of brittle fibers and a ductile matrix. The geometry is in the form of a laminate which consists of several parallel layers where each layer is called a lamina. The advantage of this construction is that it gives the material more strength and less weight.

The mechanics of composite materials deals mainly with the analysis of stresses and strains in the laminate. This is usually performed by analyzing the stresses and strains in each lamina first. The results for all the laminas are then integrated over the length of the laminate to obtain the overall quantities. In this book, Chaps. 2–6 deal mainly with the analysis of stress and strain in one single lamina. This is performed in the local lamina coordinate system and also in the global laminate coordinate system. Laminate analysis is then discussed in Chaps. 7–9. The analysis of a lamina and a laminate in these first nine chapters are supplemented by numerous MATLAB examples demonstrating the theory in great detail. Each MATLAB example is conducted in the form of an interactive MATLAB session using the supplied MATLAB functions. Each chapter of the first nine chapters has a set of special MATLAB functions

written specifically for each chapter. There are MATLAB functions for lamina analysis and for laminate analysis.

In Chap. 10, we illustrate the basic concepts of the major four failure theories of a single lamina. We do not illustrate the failure of a complete laminate because this mainly depends on which lamina fails first and so on. Finally, Chaps. 11 and 12 provide an introduction to the advanced topics of homogenization and damage mechanics in composite materials, respectively. These two topics are very important and are currently under extensive research efforts worldwide.

The analyses discussed in this book are limited to linear elastic composite materials. The reader who is interested in advanced topics like elasto-plastic composites, temperature effects, creep effects, viscoplasticity, composite plates and shells, dynamics and vibration of composites, etc. may refer to the widely available literature on these topics.

1.2 MATLAB Functions for Mechanics of Composite Materials

The CD-ROM accompanying this book includes 44 MATLAB functions (M-files) specifically written by the authors to be used for the analysis of fiber-reinforced composite materials with this book. They comprise what may be called the MATLAB Composite Materials Mechanics Toolbox. The following is a listing of all the functions available on the CD-ROM. The reader can refer to each chapter for specific usage details.

OrthotropicCompliance(E1, E2, E3, NU12, NU23, NU13, G12, G23, G13)
OrthotropicStiffness(E1, E2, E3, NU12, NU23, NU13, G12, G23, G13)
TransverselyIsotropicCompliance(E1, E2, NU12, NU23, G12)
TransverselyIsotropicStiffness(E1, E2, NU12, NU23, G12)
IsotropicCompliance(E, NU)
IsotropicStiffness(E, NU)

E1(Vf, E1f, Em)
NU12(Vf, NU12f, NUm)
E2(Vf, E2f, Em, Eta, NU12f, NU21f, NUm, E1f, p)
G12(Vf, G12f, Gm, EtaPrime, p)
Alpha1(Vf, E1f, Em, Alpha1f, Alpham)
Alpha2(Vf, Alpha2f, Alpham, E1, E1f, Em, NU1f, NUm, Alpha1f, p)
E2Modified(Vf, E2f, Em, Eta, NU12f, NU21f, NUm, E1f, p)

ReducedCompliance(E1, E2, NU12, G12)
ReducedStiffness(E1, E2, NU12, G12)
ReducedIsotropicCompliance(E, NU)
ReducedIsotropicStiffness(E, NU)
ReducedStiffness2(E1, E2, NU12, G12)
ReducedIsotropicStiffness2(E, NU)

T(theta)
Tinv(theta)
Sbar(S, theta)
Qbar(Q, theta)
Tinv2(theta)
Sbar2(S, T)
Qbar2(Q, T)

Ex(E1, E2, NU12, G12, theta)
NUxy(E1, E2, NU12, G12, theta)
Ey(E1, E2, NU21, G12, theta)
NUyx(E1, E2, NU21, G12, theta)
Gxy(E1, E2, NU12, G12, theta)
Etaxyx(Sbar)
Etaxyy(Sbar)
*Etax*xy(Sbar)
*Etay*xy(Sbar)

Strains(eps_xo, eps_yo, gam_xyo, kap_xo, kap_yo, kap_xyo, z)

Amatrix(A, Qbar, z1, z2)
Bmatrix(B, Qbar, z1, z2)
Dmatrix(D, Qbar, z1, z2)

Ebarx(A, H)
Ebary(A, H)
NUbarxy(A, H)
NUbaryx(A, H)
Gbarxy(A, H)

1.3 MATLAB Tutorial

In this section a very short MATLAB tutorial is provided. For more details consult the excellent books listed in [13–21] or the numerous freely available tutorials on the internet – see [22–29]. This tutorial is not comprehensive but describes the basic MATLAB commands that are used in this book.

In this tutorial it is assumed that you have started MATLAB on your system successfully and you are ready to type the commands at the MATLAB prompt (which is denoted by double arrows "≫"). Entering scalars and simple operations is easy as is shown in the examples below:

```
>> 2 * 3 + 7

ans =
    13
```

```
>> sin(45*pi/180)

ans =

    0.7071

>> x = 6

x =

    6

>> 5/sqrt(2 - x)

ans =

    0 - 2.5000i
```

Notice that the last result is a complex number. To suppress the output in MATLAB use a semicolon to end the command line as in the following examples. If the semicolon is not used then the output will be shown by MATLAB:

```
>> y = 35;
>> z = 7;
>> x = 3 * y + 4 * z;
>> w = 2 * y - 5 * z

w =

    35
```

MATLAB is case-sensitive, i.e. variables with lowercase letters are different than variables with uppercase letters. Consider the following examples using the variables x and X.

```
>> x = 1

x =

    1

>> X = 2

X =

    2
>> x
```

```
x =

    1

>> X

X =

    2
```

Use the help command to obtain help on any particular MATLAB command. The following example demonstrates the use of `help` to obtain help on the `det` command.

```
>> help det

 DET    Determinant.
    DET(X) is the determinant of the square matrix X.

    Use COND instead of DET to test for matrix singularity.

    See also COND.

 Overloaded methods
    help sym/det.m
```

The following examples show how to enter matrices and perform some simple matrix operations:

```
>> x = [1 4 7 ; 3 5 6 ; 1 3 8]

x =

     1     4     7
     3     5     6
     1     3     8

>> y = [1 ; 3 ; 0 ]

y =

     1
     3
     0

>> w = x * y
```

```
w =

     13
     18
     10
```

Let us now solve the following system of simultaneous algebraic equations:

$$\begin{bmatrix} 1 & 4 & 6 & -5 \\ 3 & 1 & 0 & -1 \\ 3 & 7 & 2 & 1 \\ 0 & 1 & 3 & 5 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 1 \\ -2 \\ 0 \\ 5 \end{Bmatrix} \tag{1.1}$$

We will use Gaussian elimination to solve the above system of equations. This is performed in MATLAB by using the backslash operator "\" as follows:

```
>> A = [1 4 6 -5 ; 3 1 0 -1 ; 3 7 2 1 ; 0 1 3 5]

A =

     1     4     6    -5
     3     1     0    -1
     3     7     2     1
     0     1     3     5

>> b = [1 ; -2 ; 0 ; 5]

b =

     1
    -2
     0
     5

>> x = A\b

x =

    -0.4444
    -0.1111
     0.7778
     0.5556
```

It is clear that the solution is $x_1 = -0.4444$, $x_2 = -0.1111$, $x_3 = 0.7778$, and $x_4 = 0.5556$. Alternatively, one can use the inverse matrix of A to obtain the same solution directly as follows:

```
>> x = inv(A) * b

x =

   -0.4444
   -0.1111
    0.7778
    0.5556
```

It should be noted that using the inverse method usually takes longer than using Gaussian elimination especially for large systems.

Finally in order to plot a graph of the function $y = f(x)$, we use the MATLAB command `plot(x, y)` after we have adequately defined both vectors x and y. The following is a simple example.

```
>> x = [ 1 2 3 4 5 6 7 8 9 10]

x =

     1     2     3     4     5     6     7     8     9     10

>> y = x. ^ 3 - 2 * x. ^ 2 + 5

y =

     4     5    14    37    80   149   250   389   572   805
```

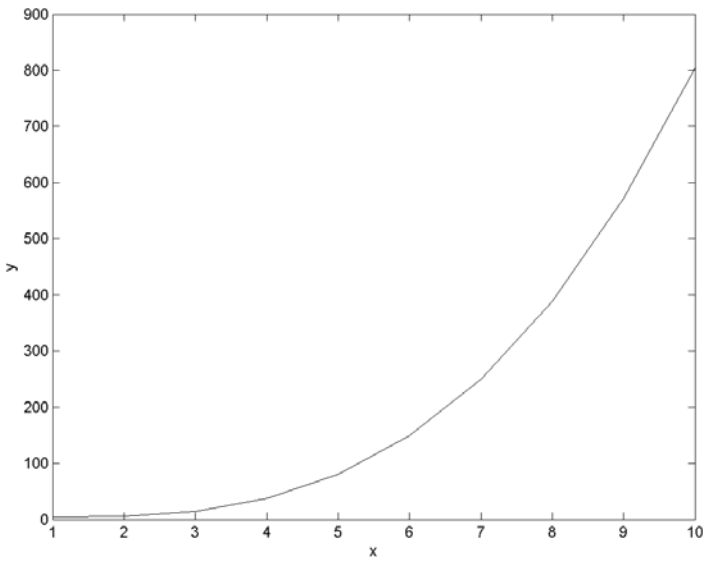

Fig. 1.1. Using the MATLAB `Plot` command

```
EDU >> plot(x, y)
EDU >> hold on;
EDU >> xlabel('x');
EDU >> ylabel('y');
```

Figure 1.1 shows the plot obtained by MATLAB. It is usually shown in a separate graphics window. Notice how the `xlabel` and `ylabel` MATLAB commands are used to label the two axes. Notice also how a "dot" is used in the function definition just before the exponentiation operation to indicate to MATLAB to carry out the operation on an element by element basis.