A.J.M. Ferreira

Solid Mechanics and its Applications

MATLAB Codes for Finite Element Analysis

Solids and Structures

Springer

MATLAB Codes for Finite Element Analysis

SOLID MECHANICS AND ITS APPLICATIONS
Volume 157

Series Editor: G.M.L. GLADWELL
Department of Civil Engineering
University of Waterloo
Waterloo, Ontario, Canada N2L 3GI

Aims and Scope of the Series

The fundamental questions arising in mechanics are: *Why?, How?,* and *How much?* The aim of this series is to provide lucid accounts written by authoritative researchers giving vision and insight in answering these questions on the subject of mechanics as it relates to solids.

The scope of the series covers the entire spectrum of solid mechanics. Thus it includes the foundation of mechanics; variational formulations; computational mechanics; statics, kinematics and dynamics of rigid and elastic bodies: vibrations of solids and structures; dynamical systems and chaos; the theories of elasticity, plasticity and viscoelasticity; composite materials; rods, beams, shells and membranes; structural control and stability; soils, rocks and geomechanics; fracture; tribology; experimental mechanics; biomechanics and machine design.

The median level of presentation is the first year graduate student. Some texts are monographs defining the current state of the field; others are accessible to final year undergraduates; but essentially the emphasis is on readability and clarity.

MATLAB Codes for Finite Element Analysis

Solids and Structures

A.J.M. Ferreira

Universidade do Porto
Portugal

Springer

A.J.M. Ferreira
Universidade do Porto
Fac. Engenharia
Rua Dr. Roberto Frias
4200-465 Porto
Portugal
ferreira@fe.up.pt

ISBN 978-1-4020-9199-5 e-ISBN 978-1-4020-9200-8

Library of Congress Control Number: 2008935506

Cover illustration: WMXDesign GmbH

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

This book intend to supply readers with some MATLAB codes for finite element analysis of solids and structures.

After a short introduction to MATLAB, the book illustrates the finite element implementation of some problems by simple scripts and functions.

The following problems are discussed:

- Discrete systems, such as springs and bars
- Beams and frames in bending in 2D and 3D
- Plane stress problems
- Plates in bending
- Free vibration of Timoshenko beams and Mindlin plates, including laminated composites
- Buckling of Timoshenko beams and Mindlin plates

The book does not intends to give a deep insight into the finite element details, just the basic equations so that the user can modify the codes. The book was prepared for undergraduate science and engineering students, although it may be useful for graduate students.

The MATLAB codes of this book are included in the disk. Readers are welcomed to use them freely.

The author does not guarantee that the codes are error-free, although a major effort was taken to verify all of them. Users should use MATLAB 7.0 or greater when running these codes.

Any suggestions or corrections are welcomed by an email to ferreira@fe.up.pt.

Porto, Portugal, *António Ferreira*
2008

Contents

1 Short introduction to MATLAB 1
1.1 Introduction 1
1.2 Matrices 1
1.3 Operating with matrices 2
1.4 Statements 3
1.5 Matrix functions 3
1.6 Conditionals, if and switch 4
1.7 Loops: for and while 5
1.8 Relations 6
1.9 Scalar functions 7
1.10 Vector functions 8
1.11 Matrix functions 9
1.12 Submatrix 10
1.13 Logical indexing 12
1.14 M-files, scripts and functions 13
1.15 Graphics 14
1.15.1 2D plots 14
1.15.2 3D plots 15
1.16 Linear algebra 16

2 Discrete systems 19
2.1 Introduction 19
2.2 Springs and bars 19
2.3 Equilibrium at nodes 20
2.4 Some basic steps 21
2.5 First problem and first MATLAB code 21
2.6 New code using MATLAB structures 28

3 Analysis of bars 33
3.1 A bar element 33
3.2 Numerical integration 36

3.3 An example of isoparametric bar . . . 37
3.4 Problem 2, using MATLAB `struct` . . . 41
3.5 Problem 3 . . . 44

4 Analysis of 2D trusses . . . 51
4.1 Introduction . . . 51
4.2 2D trusses . . . 51
4.3 Stiffness matrix . . . 52
4.4 Stresses at the element . . . 53
4.5 First 2D truss problem . . . 53
4.6 A second truss problem . . . 58
4.7 An example of 2D truss with spring . . . 63

5 Trusses in 3D space . . . 69
5.1 Basic formulation . . . 69
5.2 A 3D truss problem . . . 69
5.3 A second 3D truss example . . . 73

6 Bernoulli beams . . . 79
6.1 Introduction . . . 79
6.2 Bernoulli beam problem . . . 81
6.3 Bernoulli beam with spring . . . 85

7 2D frames . . . 89
7.1 Introduction . . . 89
7.2 An example of 2D frame . . . 91
7.3 Another example of 2D frame . . . 95

8 Analysis of 3D frames . . . 103
8.1 Introduction . . . 103
8.2 Stiffness matrix and vector of equivalent nodal forces . . . 103
8.3 First 3D frame example . . . 104
8.4 Second 3D frame example . . . 108

9 Analysis of grids . . . 113
9.1 Introduction . . . 113
9.2 A first grid example . . . 116
9.3 A second grid example . . . 119

10 Analysis of Timoshenko beams . . . 123
10.1 Introduction . . . 123
10.2 Formulation for static analysis . . . 123
10.3 Free vibrations of Timoshenko beams . . . 130
10.4 Buckling analysis of Timoshenko beams . . . 136

11 Plane stress 143
11.1 Introduction 143
11.2 Displacements, strains and stresses 143
11.3 Boundary conditions 144
11.4 Potential energy 145
11.5 Finite element discretization 145
11.6 Interpolation of displacements 145
11.7 Element energy 146
11.8 Quadrilateral element Q4 147
11.9 Example: plate in traction 149
11.10 Example: beam in bending 152

12 Analysis of Mindlin plates 161
12.1 Introduction 161
12.2 The Mindlin plate theory 161
12.2.1 Strains 162
12.2.2 Stresses 163
12.3 Finite element discretization 163
12.4 Example: a square Mindlin plate in bending 165
12.5 Free vibrations of Mindlin plates 182
12.6 Buckling analysis of Mindlin plates 192

13 Laminated plates 203
13.1 Introduction 203
13.2 Displacement field 203
13.3 Strains 203
13.4 Strain-displacement matrix B 205
13.5 Stresses 205
13.6 Stiffness matrix 207
13.7 Numerical example 208
13.8 Free vibrations of laminated plates 225

References 231

Index 233

Chapter 1
Short introduction to MATLAB

1.1 Introduction

MATLAB is a commercial software and a trademark of The MathWorks, Inc., USA. It is an integrated programming system, including graphical interfaces and a large number of specialized toolboxes. MATLAB is getting increasingly popular in all fields of science and engineering.

This chapter will provide some basic notions needed for the understanding of the remainder of the book. A deeper study of MATLAB can be obtained from many MATLAB books and the very useful help of MATLAB.

1.2 Matrices

Matrices are the fundamental object of MATLAB and are particularly important in this book. Matrices can be created in MATLAB in many ways, the simplest one obtained by the commands

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

Note the semi-colon at the end of each matrix line. We can also generate matrices by pre-defined functions, such as random matrices

```
>> rand(3)
ans =
    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575
```

Rectangular matrices can be obtained by specification of the number of rows and columns, as in

```
>> rand(2,3)
ans =
    0.9649    0.9706    0.4854
    0.1576    0.9572    0.8003
```

1.3 Operating with matrices

We can add, subtract, multiply, and transpose matrices. For example, we can obtain a matrix $C = A + B$, by the following commands

```
>> a=rand(4)
a =
    0.2769    0.6948    0.4387    0.1869
    0.0462    0.3171    0.3816    0.4898
    0.0971    0.9502    0.7655    0.4456
    0.8235    0.0344    0.7952    0.6463
>> b=rand(4)
b =
    0.7094    0.6551    0.9597    0.7513
    0.7547    0.1626    0.3404    0.2551
    0.2760    0.1190    0.5853    0.5060
    0.6797    0.4984    0.2238    0.6991
>> c=a+b
c =
    0.9863    1.3499    1.3985    0.9381
    0.8009    0.4797    0.7219    0.7449
    0.3732    1.0692    1.3508    0.9515
    1.5032    0.5328    1.0190    1.3454
```

The matrices can be multiplied, for example $E = A * D$, as shown in the following example

```
>> d=rand(4,1)
d =
    0.8909
    0.9593
    0.5472
    0.1386
>> e=a*d
e =
    1.1792
    0.6220
    1.4787
    1.2914
```

The transpose of a matrix is given by the apostrophe, as

```
>> a=rand(3,2)
a =
    0.1493    0.2543
    0.2575    0.8143
    0.8407    0.2435
>> a'
ans =
    0.1493    0.2575    0.8407
    0.2543    0.8143    0.2435
```

1.4 Statements

Statements are operators, functions and variables, always producing a matrix which can be used later. Some examples of statements:

```
>> a=3
a =
     3

>> b=a*3
b =
     9

>> eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
```

If one wants to cancel the echo of the input, a semi-colon at the end of the statement suffices.

Important to mention that MATLAB is case-sensitive, variables a and A being different objects.

We can erase variables from the workspace by using clear, or clear all. A given object can be erased, such as clear A.

1.5 Matrix functions

Some useful matrix functions are given in table 1.1

Table 1.1 Some useful functions for matrices

eye	Identity matrix
zeros	A matrix of zeros
ones	A matrix of ones
diag	Creates or extract diagonals
rand	Random matrix

Some examples of such functions are given in the following commands (here we build matrices by blocks)

```
>> [eye(3),diag(eye(3)),rand(3)]
ans =
    1.0000         0         0    1.0000    0.9293    0.2511    0.3517
         0    1.0000         0    1.0000    0.3500    0.6160    0.8308
         0         0    1.0000    1.0000    0.1966    0.4733    0.5853
```

Another example of matrices built from blocks:

```
>> A=rand(3)
A =
    0.5497    0.7572    0.5678
    0.9172    0.7537    0.0759
    0.2858    0.3804    0.0540
>> B = [A, zeros(3,2); zeros(2,3), ones(2)]
B =
    0.5497    0.7572    0.5678         0         0
    0.9172    0.7537    0.0759         0         0
    0.2858    0.3804    0.0540         0         0
         0         0         0    1.0000    1.0000
         0         0         0    1.0000    1.0000
```

1.6 Conditionals, if and switch

Often a function needs to branch based on runtime conditions. MATLAB offers structures for this similar to those in most languages. Here is an example illustrating most of the features of if.

```
x=-1
if x==0
    disp('Bad input!')
elseif max(x) > 0
    y = x+1;
else
    y = x^2;
end
```

If there are many options, it may better to use switch instead. For instance:

```
switch units
    case 'length'
        disp('meters')
    case 'volume'
        disp('cubic meters')
    case 'time'
        disp('hours')
    otherwise
        disp('not interested')
end
```

1.7 Loops: for and while

Many programs require iteration, or repetitive execution of a block of statements. Again, MATLAB is similar to other languages here. This code for calculating the first 10 Fibonacci numbers illustrates the most common type of for/end loop:

```
>> f=[1 2]
f =
     1     2
>> for i=3:10;f(i)=f(i-1)+f(i-2);end;
>> f
f =
     1     2     3     5     8    13    21    34    55    89
```

It is sometimes necessary to repeat statements based on a condition rather than a fixed number of times. This is done with while.

```
>> x=10;while x > 1; x = x/2,end
x =
     5
x =
    2.5000
x =
    1.2500
x =
    0.6250
```

Other examples of for/end loops:

```
>> x = []; for i = 1:4, x=[x,i^2], end
```

```
x =
     1
x =
     1     4
x =
     1     4     9
x =
     1     4     9    16
```

and in inverse form

```
>> x = []; for i = 4:-1:1, x=[x,i^2], end
x =
    16
x =
    16     9
x =
    16     9     4
x =
    16     9     4     1
```

Note the initial values of `x = []` and the possibility of decreasing cycles.

1.8 Relations

Relations in MATLAB are shown in table 1.2.

Note the difference between '=' and logical equal '=='. The logical operators are given in table 1.3. The result if either 0 or 1, as in

```
>> 3<5,3>5,3==5
```

Table 1.2 Some relation operators

<	Less than
>	Greater than
<=	Less or equal than
>=	Greater or equal than
==	Equal to
~=	Not equal

Table 1.3 Logical operators

&	and
|	or
~	not