Int. J. Engng Ed. Vol. 16, No. 4, pp. 362–368, 2000
Printed in Great Britain.

0949-149X/91 \$3.00+0.00

Using a Computer Algebra System to Teach the Finite Element Method*

SHIRLEY POMERANZ
Department of Mathematical and Computer Sciences, The University of Tulsa, Tulsa, OK 74104-3189, USA. E-mail: pomeranz@euler.mcs.utulsa.edu

This paper describes how the computer algebra system, Mathematica, can be used to introduce students to the finite element method. Typical students are juniors, seniors, and beginning graduate students in mathematics, computer science, and various engineering disciplines. Students were given template code. They were instructed to modify the code in order to solve two-dimensional elliptic boundary-value problems and to verify the correctness of their numerical solutions.

INTRODUCTION

THE FINITE ELEMENT method (FEM) is one of the topics introduced in our upper-level undergraduate course, Math 4503, Numerical Methods. During the Fall 1999 semester, we used a computer algebra system (CAS), *Mathematica*, to assist in teaching an introduction to FEM. The textbook used for this course [1] presents an introductory section on finite elements, and we supplemented this section.

A *Mathematica* notebook (program) that implements a basic two-dimensional FEM using linear, triangular elements was made available to students at the website, http://euler.mcs.utulsa.edu/ma4503/index.html [2]. The use of this FEM notebook assumes that students are fairly familiar with *Mathematica.* This is a major requirement, since it generally takes a good deal of exposure to *Mathematica* to become comfortable using it at the level required here (use of palettes, templates, and the help browser in *Mathematica Version 4* alleviates some of this requirement). At TU, we introduce our students to *Mathematica* in second semester calculus. Thereafter, many of our mathematics courses, including third semester calculus, differential equations, mathematical modeling, and numerical methods, use *Mathematica*. One of our faculty members has written a set of *Mathematica* tutorials to assist students and faculty in using this software effectively. These tutorials are available on the web [3].

FINITE ELEMENT NOTEBOOK AND ASSIGNMENT

When the FEM notebook is opened, it initially appears in outline form, as shown in Fig. 1. Each section of the notebook is closed, and only the topic heading from the section appears; that is, the cell groupings are closed. Groups of *Mathematica* cells can be either *open* or *closed*. When a cell group is open, all the cells are visible to the user. When a cell group is closed, only the first or *heading* cell in the group is visible. Students could double-click on any closed cell-bracket (or use the menu option) to open the cell group and view and access the enclosed code corresponding to a specific portion of the notebook. This feature of *Mathematica* essentially modularizes any large program. Students can see the overall structure of the method, i.e., the steps are shown explicitly, and then can keep more focused on the specific step of interest.

Students were given the following assignment. They were to work in teams of three students per team, access the FEM notebook (copy it onto a disk), and use it to solve a suitable problem of their choice. Suitable problems, i.e., those that the *Mathematica* program could handle, are as described in the documentation in the 'Statement of problem' section of the notebook. The recommended type of problem was a basic second-order linear partial differential equation (e.g., Poisson's or Helmholtz's equation) with Dirichlet boundary conditions specified on $\square$, a rectangle in the plane:

$$-a(\partial^2 u/\partial x^2 + \partial^2 u/\partial y^2) + cu = f(x, y) \text{ in } \Omega$$

$$u = g(x, y) \text{ on } \partial\Omega$$

This could represent, for example, a steady-state heat conduction problem in which the unknown, u, represents temperature. The problem type is restricted, yet general enough so students can ask 'what if' questions. Students could run a sequence of cases to study the effect that varying some parameter had on the behavior of the FEM solution. The equation coefficients, nonhomogeneous term, domain size, and grid are among the quantities that could be varied. More advanced students

* Accepted 15 Febraury 2000.

MA4503 Introduction to Numerical Methods
Finite Element Method Program

- ❑ Introduction:
- ❑ Statement of problem:
- ❑ Preliminary commands:
- ❑ Enter data (including Dirichlet boundary conditions); compute grid spacings:
- ❑ Construct grid and set remaining grid parameters:
- ❑ Construct grid diagram:
- ❑ Construct FEM stiffness matrix and right-hand side vector (proceed element-by-element):
- ❑ Apply Burnett's method of enforcing essential boundary conditions [4] (essentially this decreases size of FEM linear system and uses only actual unknowns):
- ❑ Solve FEM linear system:
- ❑ Form FEM solution on each element (triangle):
- ❑ Construct global piecewise linear FEM solution and present graphics:
- ❑ Use 3D ViewPoint selector under Input menu:
- ❑ FEM error estimates- Compute discrete sup norm of error and plot related graphs:

Fig. 1. Outline of *Mathematica* finite element method notebook.

could investigate rewriting the code to make it more efficient or easier to understand.

Instead of providing a program in which students would simply supply input parameters, students were provided a template program in which a specific boundary-value problem was solved. This was done intentionally. One goal was for students to experience a situation that occurs in practice; that is, code that has been intended to solve a specific problem is used to solve a different, but related problem. In this case, the code itself must be adapted to the problem of interest. Students must really understand what parts of the code are relevant and what operations these parts of the code perform.

Students were to modify the code as necessary for their problems, solve their problems, and verify the correctness of their FEM solutions. These verifications could be done using analytic, numerical, and/or graphical comparisons with analytic solutions or numerical solutions obtained by other methods, or by refining their grids and verifying that their FEM solutions behaved as expected. Since this template code was provided, the only commands that required updating were in the section, 'Enter data (including Dirichlet boundary conditions); compute grid spacings', and, additionally, one command in the section, 'FEM error estimates—Compute the discrete sup norm of the error and plot related graphs'.

This latter change involved a parameter that became too large for moderately refined grids. The author had inadvertently written a command in a form that was essentially grid-dependent. However, the way that some of the students reacted to the resulting difficulties showed much about their programming skills and their approaches to dealing with the types of difficulties encountered in using software to solve engineering problems. This issue will be addressed in the section, Discussion and Conclusion.

STRUCTURE OF FINITE ELEMENT NOTEBOOK

The *Mathematica* notebook implements a basic finite element method (FEM) program, using linear, triangular elements, to solve a second-order partial differential equation boundary-value problem (pde-bvp) on a rectangular region in the plane. Procedural programming was used because this programming style is easier for most students to follow (i.e., compared to functional programming or rule-based programming). The material in Fig. 2, which includes text, *Mathematica* commands, and graphics, is a summary of the notebook itself. For many of the *Mathematica* commands, output is suppressed in Fig. 2 in order to present this material more concisely. The entire *Mathematica* notebook is available at the website [2].

DISCUSSION AND CONCLUSION

The FEM is an especially good candidate for CAS instruction since the method is very computationally intensive. Yet there is a well-defined sequence of steps to be performed. *Mathematica* takes care of the tedious computations, and students are able to concentrate on the general steps of the method. Students are not sidetracked by tedious hand computations that may obscure the overall nature of the method [5]. This computer technology is used to complement the lectures/textbook portion of the course. It is used to make the course more relevant

1. Introduction:
This *Mathematica* notebook implements a basic finite element method (FEM) program, using linear triangular elements, to solve a second-order partial differential equation boundary-value problem (pde-bvp) on a rectangular region in the plane.

2. Statement of problem:
Solve the following pde-bvp using the FEM:

$$-\nabla \circ [a\ \nabla u] + b\ \partial u/\partial y + c\ u = \text{in}\ \Omega = [0,L] \times [-H1,H2],$$
$$u = g(x,y) \quad \text{on}\ \partial\Omega,$$

where a > 0, b, and c are constants; f and g are bounded functions.

3. Preliminary commands:
(This section includes some *Mathematica*-specific commands, e.g., calls to *Mathematica* packages. This section is not presented in this paper.)

4. Enter data (including Dirichlet bcs) and compute grid spacings:
The current test problem is from **Numerical Analysis**, 6th edition, Burden and Faires, Brooks/Cole, 1997. See Example #1, page 676, with exact solution u(x,y) = 400 x y. Note that the exact solution is bilinear in the variables x and y, so this program will not yield the exact solution.

nxnodes and **nynodes** are the number of nodes in the x-direction and y-direction, respectively.
hx and **hy** are the x- and y-grid spacings, respectively.

```
L = 1/2;
H1 = 0;
H2 = 1/2;
nxnodes = 6;
nynodes = 6;

hx = L/(nxnodes - 1)
hy = (H2 + H1)/(nynodes - 1)

a = 1;
b = 0;
c = 0;
f[x_,y_] = 0;
```

Set Dirichlet boundary conditions (bcs):
g(x,-H1) = 0, g(x,H2) = 200x, g(0,y) = 0, and g(L,y) = 200y, for $0 \le x \le L$, $-H1 \le y \le H2$.
ufem gives the FEM nodal values.

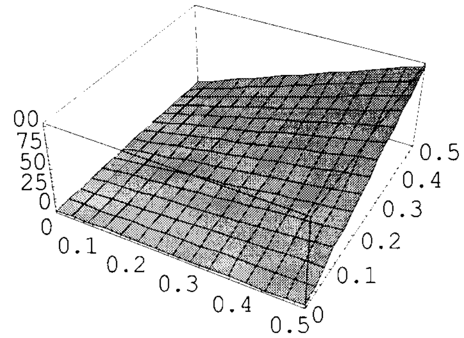
```
Table[ufem[i] = 0, {i, 1, nxnodes}]
Table[ufem[nxnodes*(nynodes - 1) + i] = 200*(0 + (i - 1)*hx), {i, 1, nxnodes}]
Table[If[Mod[i, nxnodes] == 1, ufem[i] = 0], {i, 1, nxnodes*nynodes, nxnodes}]
Table[If[Mod[i,nxnodes]==0, ufem[i]=200*(0+(i-nxnodes)/nxnodes)*(hy)],
      {i,nxnodes,nxnodes*nynodes,nxnodes}]
```

Check that the Dirichlet boundary conditions are applied correctly:

```
Table[ufem[i],{i,nxnodes*nynodes}];

uTestExact[x_, y_] = 400*x*y
Plot3D[uTestExact[x, y], {x, 0, L}, {y, -H1, H2}];
```

Fig. 2. Summary if *Mathematica* finite element method notebook.



Exact Solution, **uTestExact**

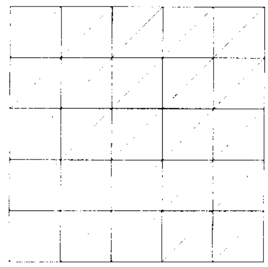
5. Construct grid and set remaining grid parameters:

(This section is not presented in this paper.)

6. Construct grid diagram:

triangle is a graphics object that is used to depict the grid.
For example, note this structure, and use it to construct the following loop and mesh diagram:

```
Table[meshPoints1[[1,k]],{k,1,3}];
triangle[k_]:=Graphics[Line[
     { meshPoints1[[k,1]], meshPoints1[[k,2]], meshPoints1[[k,3]],meshPoints1[[k,1]] } ] ]
Show[Table[triangle[k],{k,1,ntriangles}], AspectRatio->Automatic];
```



Grid diagram

7. Construct FEM stiffness matrix and right-hand side vector (proceed element-by-element):

```
elementContribution[elementc_]:=
        Module[{xList, yList,area,aa,bb,cc,diag,phi},
        {xList,yList}={gxcoord[elementc],gycoord[elementc]};
         area=(1/2)Det[{{1,xList[[1]],yList[[1]]},{1,xList[[2]],yList[[2]]},{1,xList[[3]],yList[[3]]}}];
aa=RotateLeft[xList]RotateLeft[yList,2] - RotateLeft[xList,2]RotateLeft[yList];
bb=RotateLeft[yList]-RotateLeft[yList,2];
cc=RotateLeft[xList,2]-RotateLeft[xList];
phi[x_,y_]=(1/(2*area))(aa+bb*x+cc*y);
diag[x_]=InterpolatingPolynomial[{{xList[[1]],yList[[1]]}, {xList[[3]],yList[[3]]}},x];
SetCoordinates[Cartesian[x,y,z]];
  Do[  Do[
          stiffmat[ elementc[[i]],elementc[[j]] ]=
          stiffmat[ elementc[[i]],elementc[[j]] ]+Integrate[ (a*Grad[phi[x,y][[i]]].Grad[phi[x,y][[j]]] +
          b*D[phi[x,y][[j]],y]*phi[x,y][[i]]+c*phi[x,y][[i]]*
          phi[x,y][[j]]) ,{x,xList[[1]], xList[[2]]},{y,yList[[1]],diag[x]}],
        {i,1,3}];
```

Fig. 2. Continued.

rhs[elementc[[j]]]=rhs[elementc[[j]]] + Integrate[f[x,y]*phi[x,y][[j]], {x,xList[[1]], xList[[2]]},{y,yList[[1]],diag[x]}],
{j,1,3}]]

Initialize stiffness matrix and rhs vector to be zeros. Then loop over all elements to compute their contributions and store in the appropriate locations:
Caution: Be sure that **stiffmat** and **rhs** are zeroed out appropriately.
elementc stores the x- and y-coordinates of the three nodes that define a specific triangular element.

```
Timing[ Table[stiffmat[i,j]=0,{i,nunknowns},{j,nunknowns}];
Table[rhs[i]=0,{i,nunknowns}];
Do[
 elementc=elemconnectivity[[k]];
 elementContribution[elementc],
{k,ntriangles}] ]
```

8. Apply Burnett's method of enforcing the essential boundary conditions (essentially this decreases the size of the FEM linear system, and consequently we are working with actual unknowns):

(This section is not presented in this paper.)

9. Solve the FEM linear system:

(This section is not presented in this paper.)

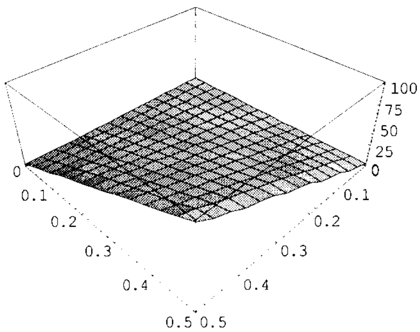
10. Form FEM solution on each element (triangle):

(This section is not presented in this paper.)

11. Construct global piecewise linear FEM solution and present graphics:

Following are plots of the finite element approximation (**femApproximation**) and the corresponding exact solution, **uTestExact**. 2D-plots, 3D-plots and contour plots are given.

```
Plot3D[femApproximation[x,y],{x,0,L},{y,-H1,H2},
ViewPoint->{1.2,1.2,1.2}];
```



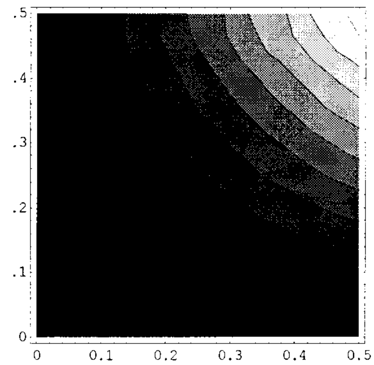
3-D plot of finite element approximation, **femApproximation**

```
ContourPlot[femApproximation[x,y],{x,0,L},{y,-H1,H2}];
```

Fig. 2. Continued.

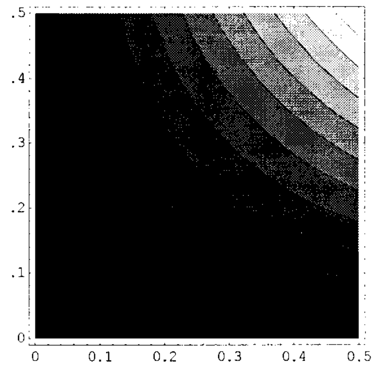
(hands-on), make the material easier to learn, and because, for most of these students, this is the context in which they will eventually use numerical methods.

As numerical methods have tradeoffs, so do methods of teaching numerical methods. This use of *Mathematica* with a template FEM program is more of a 'middle of the road' approach [6]. Students have access to the code, should they wish to modify the program; however, on the other hand, this template can more or less be used as a 'black box' in which case students update only the required commands. In either scenario, the notebook provides students with the



Contour plot of finite element approximation, **femApproximation**

ContourPlot[uTestExact[x,y],{x,0,L},{y,-H1,H2}];



Contour plot of exact solution, **uTestExact**

12. FEM error estimates-
Compute the discrete sup norm of the error and plot related graphs:

(This section is not presented in this paper.)

Fig. 2. Continued.

capability to interactively explore and observe how the numerical solution depends on the values of certain parameters.

This approach enabled the author to see how some of the students reacted when a particular command took too long to evaluate. One command in the section, 'FEM error estimates—Compute the discrete sup norm of the error and plot related graphs', involved a parameter that became too large for moderately refined grids. The author had inadvertently written a command in a form that was grid-dependent. *Mathematica* was performing a doubly nested loop in sampling the absolute value of the error at selected points. In the original code, 50 sampling points were used per grid step, both in the x-direction and in the y-direction. Thus, for the choice of 6 nodes in the x-direction and similarly in the y-direction, there were $250^2 = 62{,}500$ sampling and comparison operations. However, when the students ran their problems with more refined grids, *Mathematica* got hung-up on this command. And this makes sense, because several hundred thousand (or more) operations were required. But instead of looking at the code and determining what might be the cause of this difficulty, some of the students just left *Mathematica* running all night! We have been introducing technology to our students as labor saving tools, and apparently the students interpreted this as saving all mental labor as well. This episode did generate a discussion with the students about looking at the code (in this case, the nested loops) and attempting to understand the cause of the difficulty and determining more effective 'fixes'.

Particularly with respect to the topic of finite elements, care must be taken in introducing this topic at the undergraduate level [7]. CASs can play a unique role in rendering this topic suitable for this level of presentation. If students have the mathematical background provided by three semesters of calculus, an introduction to linear algebra, and some exposure to partial differential equations (for the application presented in this paper), CASs can facilitate in presenting the salient aspects of this method [8, 9].

Acknowledgement—The author acknowledges The University of Tulsa Faculty Development Summer Fellowship (1999), which enabled her to write the *Mathematica* finite element method notebook.

REFERENCES

1. R. L. Burden and J. L. Faires, *Numerical Analysis*, 6th edition, Brooks/Cole, Pacific Grove (1997) pp. 707–722.
2. S. B. Pomeranz, Department of Mathematical and Computer Sciences (MCS), The University of Tulsa, Tulsa, OK, (1999), http://euler.mcs.utulsa.edu/ma4503/index.html/.
3. D. R. Doty, MCS, The University of Tulsa, Tulsa, OK, (1999), http://euler.mcs.utulsa.edu/ma7103/mathematica.html/.
4. D. Burnett, *Finite Element Analysis*, Addison-Wesley, Reading, MA (1987) pp. 108–114.
5. W. Gander and D. Gruntz, Derivation of numerical methods using computer algebra, *SIAM Review* **41,** 3, (1999) pp. 577–593.
6. C. Pozrikidis, Software approach aims for the middle ground, *Scientific Computing World*, (August/September 1999) pp. 23–24.
7. J. Matthews and S. Jahanian, Computer applications through interdepartmental engineering students through finite element method, ASEE *CoED Journal* **ix,** 4, (1999) pp. 46–48.
8. J. Milton-Benoit, I. R. Grosse, C. Poli and B. P.Woolf, The multimedia finite element modeling and analysis tutor, *ASEE J. Eng. Educ.*, (1998 Supplement) pp. 511–517.
9. B. Cabell, V, J. Rencis and H. Grandib Jr., Using Java to develop interactive learning material for the world-wide web, *Int. J. Eng. Educ.*, **13,** 6, (1997) pp. 397–406.

Shirley Pomeranz is an Associate Professor of Mathematics in the Department of Mathematical and Computer Sciences at The University of Tulsa. She is active in the American Society for Engineering Education (ASEE) Mathematics Division and is a member of the Editorial Advsiroy Board for *The International Journal of Engineering Education*. Her interests include support of women in mathematics, engineering mathematics, and the finite element method.