Problem Solving Using Programming Language.

O. Shoewu, M.Sc.

Department of Electronics and Computer Engineering,
Lagos State University, Epe Campus, Epe, Nigeria.

E-mail: engrshoewu@yahoo.com

ABSTRACT

The second-year electronics and computer engineering curricula of Lagos State University does include a course in programming in a high level language (such as FORTRAN, BASIC, etc.). Such courses have traditionally been justify as a teaching logical thinking and problem solving, while providing the student with a tool that they will use as practicing engineers. However, these goals can be achieved more readily nowadays due to the use of modern mathematical software tools such as MATLAB®. These tools are powerful and more easier to master than a traditional programming language, and it allows the undergraduate electronic computer engineering student to solve interesting and challenging problems earlier in the course. Consequently, there is little reason to teach programming languages to undergraduate engineering students and many reasons not to , and all these shall be highlighted in the ensuring discussion.

(Keywords: problem solving, programming language, MATLAB®, computer logical thinking)

INTRODUCTION

During the 1998/1999 academic session, Lagos State University reorganized the second year curriculum for engineers. The most significant changes were the addition of computer tools course, and the restructuring of the existing programming languages. The change affects approximately 300 students per year.

Before the fall of 2000, electronic engineering student in LASU were required to complete a three-semester-hour programming course in either the C or FORTRAN programming language during their second year. We taught ECE211, "Introduction to basic programming language." The course had a well-deserved reputation as being difficult and time consuming.

In the fall of 2000 (Dept. of ECE, LASU, 1999), this course was replaced by two-hour courses, the first of which is ECE 211, Introduction to programming languages". The course begins by introducing students to the operating system that they will be using during their stay at LASU (Windows) with particular emphasis on information exchange and communications (Press et al, 1988). After this brief introduction to Windows, the emphasis of the course shifts to engineering tools: MATLAB® and a spreadsheet. Once the students have successfully completed ECE 211, they are then required to take a 2-hour programming course in either C or FORTRAN.

NEED FOR COMPUTER PROGRAMMING

Almost all of the engineering schools in the country require their students to learn at least one high- level programming language such as C, FORTRAN, or Pascal (Shoewu et al., 2003) . In discussion with colleagues, we have heard a number of arguments in favor of this practice:

- "Programming teaches students to think logically"
- "Programming allows students to solve technical problems"
- "There is need for on the job programming".

We shall consider each of these arguments one after the other.

LOGICAL THINKING

At first glance, the concern for logical thinking might appear to justify teaching traditional

programming languages. Certainly we want our students to be able to attack problems systematically, and computer programming would seem to provide a vehicle for teaching this. But logical thinking is also required to use MATLAB® (Shampine et al., 2003) or any other computer tool successfully. A student will not get very far trying to solve problems in MATLAB® by randomly pressing the keys. A variation of the " logical thinking" argument is that students should understand the basic principles of programming-flow of control, program modularity, and so on. Those who advance this view often say it is not important which language the students are taught (Davis et al, 1995). In that case, MATLAB® would serve just as well as C or FORTRAN, since MATLAB® offers the usual programming constructs: loops (for and while), selection structures (if-else), functions, arrays, strings, and so on.

PROBLEM SOLVING

Every problem that can be solved by MATLAB® can also be solved in C or FORTRAN. The difference is that the MATLAB® solution requires considerably less time to implement. In other words, MATLAB® is more powerful than C or FORTRAN. Let us consider, for example, the solution of the system of equations Ax = b by LU decomposition. The MATLAB® solutions require some line of code: X = A/b. in contrast, a C will carry out the same computation required about 70 lines of code.

1) A skilled programmer (Shoewu, 2001) would require many hours to design, code, test, and debug such a program is likely to contain error.

2) To make matter worse, most of our electronic engineering students are not skilled programmers. Over the last decade, in fact, programming language is not taught in secondary schools. The fraction of our electronic and computer engineering student who have taken a course in high-level programming language before entering the higher institution has reduced below 50%.

3) Therefore, undergraduate-programming classes must be taught on the beginning level, with the result that instruction tends to focus on the details of syntax. More importantly, the problems used in instruction must be chosen for their simplicity and ease of demonstration of a particular programming construct. We have found that it takes twelve to fifteen weeks to cover enough C or FORTRAN.

4) (On the other hand, 64% reported "frequent" use of spread sheets). Of course, computer and software engineers will continue to need high-level programming languages for example, it would not make sense to write a campier or develop a CAD package using MATLAB®. However, the vast majority of engineers are not in the business of writing software. For them, MATLAB® would be a more useful tool than C or FORTRAN.

EXAMPLE OF EIGEN VALUE PROBLEMS

This section describes the solutions of some eigen value PDE problems. The problems are solved using the graphical use interface (Shoewu, 2001) and the command-line function of PDE Toolbox (LeBold, et al., 1995).

EIGEN VALUE AND EIGEN FUNCTION FOR THE L-SHAPED MEMBRANE

The problem of finding the eigenfunctions of an L-shaped membrane is of interest to all MATLAB® users, since the plot of the first eigenfunction are the logo of the Math Work. In fact, you can compare the PDE Toolbox to the ones produced by the MATLAB® function membrane.

The problem to computer all eigen mode with eigen values< 100 for the eigen mode PDE problems. $-\Delta=\lambda\mu$ on the geometry of the L-shaped membrane. U = 0 on the boundary (Dirichlet condition).

USING THE GRAPHICAL USER INTERFACE

With the PDE Tool GUI active, check that the current mode is generic scalar. Then draw the L-shaped as a polygon with the corners in (0,0), (-1,0), (-1,-1), (1,-1), (1,1), (0,1).

There is no need to define any boundary conditions for this problem since the default condition –u = 0 on the boundary is the correct one. Therefore, you can continue to the next step: to initialize the mesh. Refine the initial mesh twice. Defining the eigen value PDE problem is also easy. Open the PDE specification dialog box and select eigenmodes . the default values for the PDE coefficients, c=1, a=0, d=1, all match the problem description, so you can exit the PDE coefficients, c=1, a=0, d=1, so you can exit the PDE specification dialog box by pressing the OK button.

Open the solve parameters dialog box by selecting parameters … from the solve menu. The dialog box contain s an edit box for entering the eigen value search range. The default entry in [0,100]. Which in just what you want. Finally, solve the L-shaped membrane problem by pressing the = button. The solution first displayed is the first (smallest) eigen value is also displayed. You find the number of eigen values on the information line at the bottom of the GUI. You can open the plot selection dialog box and choose which eigen function to plot by selecting from a pop-up menu of the corresponding eigen values.

USING COMMAND-LINE FUNCTIONS

The geometry of the L-shaped membrane is described in the file ishapeg.m and the boundary conditions in the file ishapeb.m. first, initialize the mesh and refine it twice using the command line functions at the MATLAB® prompt:

“[p,e,t]=initmest(1shapeg’);
“[p,e,t]=refinemest(1shapeg’p,e,t);
“[p,e,t]=refinemest(1shapeg’p,e,t);

recall the general eigen value PDE problem description:

$-\nabla\ (c\ \nabla\ u) + au = \lambda\ du$

This means in this case you have c=1, a=0, d=1. The syntax of pdeeig, the eigenvalue solver in the PDE Toolbox is [V,1]= pdeeig (b, p, e, t ,c, a, d, r).

The input argument r is a two-element vector indication the interval on the real axis where pdeeig searches for eigen values. Here you are looking for eig sen values< 100, so the interval you use is [0,100].

Now you can call pdeeig and se how many eigen values you find, [V,1]=pdeeig (Ishaped”m,p,e,t,l,[0,100]);

There are 19 eigen values smaller than 100. Plot the first eigen mode and compare it to MATLAB’s membrane function:

,,pdesurf (p,t,v (:,1)
,, figure
,, membrane (1, 20, 9, 9)
Membrane can produce the first 12 eigen function for the L-shaped membrane
Compare also the 12th eigenmodes;
,, figure
,, pdesurf (p,t,v (:,12()
,, figure
,, membrane (12, 20,9,9)

Looking at the following eigenmodes, you can see how numbers of oscillation increases. The eigen functions are symmetric or anti-symmetric around the diagonal from (0,0) to (1,1), which divides the L-shaped membrane into toe mirror images. In a practical computation, you could take advantage of such symmetries in the PDE problem, and solve over a region half the size. The eigen values of the full L-shaped membrane are the union of those of the half with the dirchlet boundary condition along the diagonal (eigenvalues 2,4,7,11,13,16 and 17) and those of the Neumann boundary condition (eigenvalues 1,3,5,6,10,12,14, and 15).

The eigen values λ8 make up a double eigen values for the PDE at around 49, 64. Also, the eigen values λ8 and λ18 and λ19 make u another double eigen value at around 99.87. you may have gotten two different but close values. The default triangulation made by intmesh is not symmetric around the diagonal, but a symmetric grad gives a matrix with true double eigen value. Each of the eigen functions U8 and U9 consist of three copies of eigen functions over the square unit obtained the zero values along a diagonal of the square- any line through the centre of the may be computed.

This shows a general fact about multiple eigen values for symmetric matrices; namely that any

vector in the invariant subspace is equally valid as an eigenvector. The two eigen functions U8 and U9 are orthogonal to each other if the dividing lines make right angles. Check your solution for that.

Actually, the eigen values of the square can be computed exactly. They are (m2 + n2) π 2 (e.g. the double eigen value) λ18 and λ19 is 10π 2 which pretty close to 100.if you compute the FEM approximation with only one refinement, you would only find 16 eigen values, and obtain the wrong solurin to the problem. You can of course check for this situation by computing the eigen values over a slightly larger range than the original problem. You get some information from the print out in the MATLAB® command window that is printed during the computation. For this problem , the algorithm computed a new set of eigen value approximation and tested for convergence every third step. In the output, you step number, the time in seconds since the tart of the eigen values both inside and out side the interval counted.

Here is what MATLAB® wrote :

Basis= 10 time=2.70, New con Eig =0
Basis =13 time =3.50, New con Eig =0
Basis =16 time =4.36 New con Eig =0
Basis =19 time =5.34 New con Eig =1
Basis =22 time =6.46 New con Eig =2
Basis =25 time =7.61 New con Eig =3
Basis =28 time =8.86 New con Eig =4
Basis =31 time 10.23 New con Eig =5
Basis =34 time =11.69 New con Eig =6
Basis =37 time =13.28 New con Eig =7
Basis =40 time =14.97 New con Eig =8
Basis =43 time 16.77 New con Eig =9
Basis =46 time=18.70 New con Eig =10
Basis =52 time=22.90 New con Eig =11
Basis =55 time=25.13 New con Eig =12
Basis =58 time =27.58 New con Eig =13
Basis=61 time=30.13 New con Eig =14
Basis =64 time=32.83 New con Eig =15
Basis =67 time=35.64 New con Eig =16
Basis =70 time=38.62 New con Eig =17
End of sweep :basis=32 time =38.62 New con Eig =18
Basis =35 time=43.29 New con Eig =19
Basis =38 time =44.70 New con Eig =20
Basis =41 time=46.22 New con Eig =21

Basis =44 time=47.81 New con Eig =22
Basis=47 time=49.52 New con Eig =0
Basis=50 time=51.35 New con Eig =0
Basis =53 time=53.27 New con Eig =0
End of sweep: basis =53 time =55.30 New con Eig =0

You can see two Arnoldi runs were made. In the first ,22 eigen values converged after a basis of size 70 was computed; in the second ,where the vectors were orthogonal zed against all the 22 converged vectors, the smallest eigen value stabilized at a value[0,100], so the algorithm signalled convergence of the 22 converged eigen values, 19 were inside the search interval.

All of the material necessary to solve the problem is converged in four weeks of MATLAB® instruction. Although the students may not know how to calculate the inverse of a matrix, they are able to understand the solution of a system of linear equation given in MATLAB® as x=in(A) *b or x=A/b. we tell the student that they will learn more about the theory of linear algebra in future courses. For now, they can concentrate on solving the numerous problems (mathematical and engineering) that involve systems of linear equation. Through problems such as this , we are able to expose our students to problem that do not have one single correct solution. Although they find this frustrating at times (they often want rules to apply a particular curve-fitting method or for deciding which interpolation method to use), it provides an opportunity of use to explain that in "real life" situation engineers need to depend on professional judgments, and that the instructions they desire do not exist.

CONCLUSIONS

MATLAB® offers a number of significant advantages over traditional high-level programming languages such as FORTRAN or C:

- More power: a student or engineer can solve difficult problems in less time and with less effort.
- Greater ease of use: MATLAB® is easier to master than a high-level programming language. As a result, the students have the tool to solve significant engineering problems much earlier in the semester.
- Emphasis on problem solving: rather than becoming entangled in language issues, students can concentrate more

fully on the problem to be solved. Students can tackle more time to analyse their result.

- Fewer errors; when using MATLAB®, students make fewer errors. Moreover, they are better able to find and correct the errors themselves, requiring less help from instructors and teaching assistants (Peterson, 1995).

REFERENCES

1. Press, W.H., B.P. Flannery, S.A, Teukolsky, and W.T. Vetterling. 1988. *Numerical Recipes in C*. Cambridge University Press: Cambridge, UK.
2. Peterson, I. 1995. *Fatai Defect*. Time Books: New York, NY.
3. Le Bold, W.K., Ward, S.K., and Garling, A. 1995. "Freshman Evaluating Report". Department of Freshman Engineering, Purdue University: Philadelphia, PA.
4. Davis, J.S., G.E. Blau, and G.V Reklaitis. 1995. "Computer in Undergraduate Chemical Engineering Education." *Chemical Engineering Education*. 29(1):50-55.
5. Department of Electronic and Computer Engineering, Lagos State University Undergraduate. 1999. *Programme Handbook*. LSU: Lagos, Nigeria.
6. Shoewu, O., S.K. Kassim, and A.A. Ajasa. 2003. "Introduction to Computer Applications". ADMASS Publishing Company.
7. Shoewu, O. 2001. "Understanding Computer Graphics". Dec. 2001.
8. Shampine, L.P., Gladwell, I. and Thompson, S. 2003. "Mathlab Programming for Engineers, C++ solving Odes with Mathlab". Cambridge University: Cambridge, UK.

ABOUT THE AUTHOR

O. Shoewu, M.Sc., B.Sc.(Hons), MNSE, serves as a Lecturer at the Lagos State University. He earned his M.Sc. and B.Sc. from the Lagos State University and the University of Lagos in 1992 and 1995, respectively. He is presently a postgraduate student working towards a Doctoral degree in Electronics. His research interests are in the areas of electronics, computers, and communications technology.

SUGGESTED CITATION

Shoewu, O. 2009. "Problem Solving Using Programming Language". *Pacific Journal of Science and Technology.* 10(2):408-412.

Pacific Journal of Science and Technology