High Level Programming Packages in Undergraduate Mechanical Engineering

I. H. Leslie, G. Garcia
New Mexico State University

Abstract

At the start of the 2003-04 academic year Mathcad and Matlab were chosen for the introductory programming course in the Mechanical Engineering Department. Prior to this change, C programming was taught, which had replaced FORTRAN several years earlier. The primary motivation for introducing these high level programming packages is to allow more time to be spent on setting up and solving engineering problems. Faculty responsible for this introductory course as well as a numerical methods course felt that too much time was being spent on the minutia of programming syntax. Both of these packages can be learned sufficiently quickly to tackle meaningful problems in undergraduate engineering within one semester. Each package has its strengths and a role to play in undergraduate engineering instruction. These strengths will be discussed in the paper.

Introduction

The Mechanical Engineering Department at NMSU has a long history of teaching programming at the undergraduate level. As with virtually every other engineering program, the important role of scientific computing was recognized early on. When one of the authors joined the faculty in 1984, FORTRAN was the programming language being taught. This language is still sometimes used for code development at the graduate level. Research level computational work at Los Alamos National Laboratory and Sandia National Laboratories is often done with FORTRAN. Despite FORTRAN's historical importance, C (also C++) programming has become much more wide spread. Thus, several years ago FORTRAN was dropped in favor of C. Because of this change it was possible for ME students to take a programming class from the Computer Science Department as well as through their own department. This scheduling flexibility notwithstanding, C programming was dropped this academic year in favor of the higher level programming available with Mathcad and Matlab.

Once the decision to drop C was made, discussions among the faculty were held to decide which package should be the replacement. Although Mathcad and Matlab can be used to do many of the same things, they are really quite distinctive and have their own strengths. These strengths are discussed latter, but for now it can be said that the conclusion to teach both was based on flexibility, both for the student and the instructor. A numerical methods course may be better suited to Matlab, while a course such as heat transfer, which has not historically relied on numerical computations, may more easily incorporate Mathcad. Mathematica has been used in the graduate engineering analysis course.

Goals for Undergraduates

The two basic goals for our mechanical engineering students at NMSU are:

1. Learn the basics of structured programming.
2. Learn to apply programming to solve engineering problems.

The primary goal is really item 2, with item 1 as a necessary first step to obtain that goal. Programming languages such as FORTRAN, BASIC, C, and C++ have similar programming constructs. These are familiar to programmers and include:

1. Variable types
2. Conditional branching
3. Looping
4. Data input/output
5. Functions – built-in and user-defined
6. Arrays – vectors and matrices

Mathcad, and particularly Matlab, have the programming capabilities listed above. Thus there is no loss regarding programming style. In fact, each of these software packages promotes good programming style by limiting programs to sequences, decisions, and loops[1], the only structures required for a computer program. In addition, indentation for loops and decisions statements is automatic. This makes the program easier to understand, and therefore debug.

Regardless of the program language or package chosen, solving problems with a computer often puts higher demands on the students. The student must understand the problem in detail in order to logically develop the steps and statements needed for the code. One example that actually came up in a class involved root finding. A student was using a hand calculator to find the root of cubic equation. The value that was returned was recognized as incorrect for the problem at hand. The reason was that the calculator was returning only one of the roots and the student did not realize this. When developing a computer program to perform root finding, specifying the initial guess or guesses is an integral part of the problem. This naturally leads to the issue of which root will be found.

The programming skills learned with Mathcad and Matlab can serve an engineer well on their own. These skills can also form a firm foundation for learning C or FORTRAN if the need arises in graduate school or the work place.

Why not C?

Before the introduction of the C programming language many departments used FORTRAN to meet their engineering problem solving needs. Once C was introduced it did not take long before it was realized that C had many advantages and was much more flexible than FORTRAN. It is a given that students must possess programming skills to solve many engineering problems. The main issue departments are usually faced with is which of the available programming languages best fits the needs of the students and the engineering problems they must solve. Many engineering departments chose to replace FORTRAN in the curriculum with C because of the flexibility and the belief that C would be a more valuable programming language to know than FORTRAN. However, FORTRAN is still popular among many engineers and scientists who require number crunching intensive applications.

Over time the high level software packages have evolved to such a degree that they are very engineering problem solving oriented. Companies have catered to the engineering and scientific community and as a result the packages have become much more useful in day to day applications and have become very user friendly. As a result these high level software packages soon grew in popularity in the university environment[2]. It is now common for engineering departments to maintain these high level software packages as common software on departmental computers which are available to students.

As a result of the evolution of the high level software packages and their usefulness to solve day to day engineering type problems students are opting to use these packages over C programming to meet their computing needs. In addition, many companies have adopted these high level software packages and use them to meet their day to day computing needs. It is not uncommon for a potential employer to ask a student if they are familiar with one of these high level software packages. As a result, students are expressing their desire to learn these high level software packages.

In the Department of Mechanical Engineering at NMSU, C replaced FORTRAN several years ago and has been taught to students in our ME 260 course, ME Problem Solving. So the question is why have we abandoned C and adopted Mathcad and Matlab?

The answerer is not a simple one line explanation but a combination of reasons. The C programming language can be considered complex by students who have no programming background. It is not the details of understanding the sequential nature of programming or the various control structures that seem to cause problems for the students. It is frustration with the C structure and the syntax that presents the biggest problems with student performance. This assessment is based upon grading countless numbers of HW assignments and exams. Students generally have the structure of the problem correct but often do not declare variables correctly, or variables are not declared at the correct location in the program, or functions are not declared correctly. Often the programs are littered with syntax errors. The end result is that students perform poorly and, because of the importance of the programming language structure and syntax, teaching problem solving skills takes a back seat to teaching the C programming language.

These frustrations along with the demand by employers to have students knowledgeable in high level software packages has resulted in the Department of Mechanical Engineering at NMSU replacing C with Mathcad and Matlab.

Why Mathcad and Matlab?

Mathcad and Matlab are two quite different software packages with their own strengths and weaknesses. They are very complimentary when it comes to solving engineering type problems. Mathcad has the ability to present text, equations, graphical plots, and graphics, thus making it especially well suited for solution presentation. With Mathcad it is a simple matter to add formatted text and graphs to a document. In addition, many mathematical operations such as integration, differentiation, and summation are presented in a standard notational form. This makes a Mathcad document easy to read and understand, and ideal for a variety of homework style assignments. The development of programs in Mathcad is fairly straightforward. However, the programs written are pretty much confined to the one document. Although programming in Mathcad is fairly easy to understand it is a little difficult for students to grasp for those already familiar with C programming structure.

Matlab can be considered a more powerful programming tool than Mathcad with characteristics quite similar to C and FORTRAN. The sequential programming nature and control structure are virtually the same between Matlab and C. The main differences have to do with declaration of variables and functions, syntax, and ability to generate graphical results with little effort. Examples demonstrating the programming structure of Matlab for a tri-diagonal solver and root finding scheme are provided later on in the paper.

Like Mathcad, Matlab has some symbolic solving capabilities, although the symbolic toolkit must be purchased for Matlab. While Mathcad and Matlab have similar symbolic capabilities the main difference is in presentation. The elegance of Mathcad's symbolic capabilities lies in its ability to present equations in their standard notation, thus making it very easy to understand and follow the flow of a problem.

Although students could solve most problems using either Mathcad or Matlab, it is our belief that the decision to use one or the other lies in the type of problem being solved. Students are taught the capabilities, strengths, and weaknesses of the two software packages and when to use one package over the other. The end result is the ME 260 course now focuses on problem solving skills and in the process students learn how to use these powerful high level software packages.

Required Courses with Mathcad and Matlab

Students are first introduced to Mathcad and Matlab in the sophomore level course, ME 260 as mentioned above. This one semester course is three credit-hours. In this course students spend one hour in class and 2 hours in lab solving problems each week. ME 260 covers the following topics:

1. Engineering Problem Solving	7. Interpolation and Curve Fitting
2. MATLAB Environment	8. Ordinary Differential Equations
3. MATHCAD Environment	9. Symbolic Mathematics using MATHCAD
4. MATLAB Functions	10. Integration and Differentiation using MATHCAD
5. Linear Algebra and Matrices	
6. Solutions to Systems of Linear Equations	11. Special Topics

Although traditional programming topics such as functions, control structures, arrays, input/output, etc. are not listed in the above topics they are still covered in the course. The emphasis of the course is problem solving and in the process of solving engineering problems students learn much of the material taught in a traditional programming course. The main difference is that students learn the material as the need arises. The course was taught for the first time using the high level software packages in the Fall 2003 semester. Since the course is taught every semester it will not be long before the course evolves into one that maximizes the benefits of the high level software packages and, at the same time, teaches students the basics of structured programming.

At the junior level the students take a required three credit-hour numerical methods course, Engineering Analysis II (ME 329). This course covers the following subjects while fostering good programming style.

1. Sources of error	5. Interpolation and curve fitting
2. Root of equations	6. Numerical differentiation and curve fitting
3. Solution of linear system of equations	7. Solution of ordinary differential equations
4. Solution of non-linear system of equations	8. Solution of partial differential equations

Both Mathcad and Matlab have been used for the course. Starting in the Fall 2003 semester the decision was made to use Matlab exclusively. One reason is that Matlab most closely resembles C or FORTRAN in how programs are written. Although not used in the course, Matlab allows both C and FORTRAN to be imported, and for Matlab m-files to be exported as stand alone C programs. Thus, it is quite powerful and, in fact, can be used as a program development tool. With Matlab it is straightforward to develop user-defined functions that can be used as modules in larger programs. For example, students have been introduced to and written a tri-diagonal solver (see Figure 1.). This function has been used to solve a simple 1-D fin problem as well as forming a part of an alternating direction implicit, or ADI[1], scheme for 2-D transient heat transfer. Something similar could be done with Mathcad, but would require cutting and pasting and would be rather contrived unless the DLL creation feature is used.

```
% Tri-diagonal solver using Thomas algorithm.
% Matrix D is n x 4 with lower, main, and upper
% diagonals in columns 1, 2, and 3 respectively.
% Column 4 holds the right-hand-side vector b.

function [x] = tridiag(D)
n = size(D,1);
D(1,4) = D(1,4)/D(1,2);
for i = 2:n
    D(i-1,3) = D(i-1,3)/D(i-1,2);
    D(i,2) = D(i,2) - D(i,1)*D(i-1,3);
    D(i,4) = (D(i,4) - D(i,1)*D(i-1,4))/D(i,2);
end
for i = n-1:-1:1
    D(i,4) = D(i,4) - D(i,3)*D(i+1,4);
end
x = D(:,4);
```

Figure 1. Matlab sample program.

Matlab has many built in capabilities and functions. These higher level capabilities are particularly well suited to other classes and on-the-job needs. ME 260 introduces Matlab and focuses on structured programming using basic control structures i.e., “sequence, selection, and repetition”[1]. This course does not use most of the advanced Matlab capabilities. The numerical methods course, ME 329, continues to stress structured programming, while teaching students numerical techniques. Whether advanced Matlab capabilities are used depends on the objective of a particular numerical method.

An example is the solution of a system of linear equations. If we consider the matrix form $Ax = b$, then with Matlab it is a simple matter to find the vector x with the left division operator (\), $x = A \backslash b$. The actual solver used depends on the structure of the matrix A, and is therefore quite sophisticated. When teaching the students methods for solving linear equations they are not allowed to use this capability except as a check. The logic is that in the future they may need to incorporate a solver into a C or FORTRAN code and that an understanding of how a numerical method works and how that method scales with the number of unknowns is valuable to know.

A counter example is the solution of a system of non-linear equations using Newton’s method. Here a system of linear equations is repeatedly solved while iterating. In this case the method and logic of the program were stressed so the left division operator was allowed. The system of non-linear equations shows the students how to put together a program using modules (functions). The main program calls a function `simul` which performs the iteration on the system of linearized equations. The function `simul` must call a function `jacobian` that determines derivatives using central differences. Both `simul` and `jacobian` call a function `mysytem` that evaluates each of the equations. In this way the students may check out and debug each part of the larger program. The function `simul` is given in Figure 2.

```
% This program uses Newton's method for finding roots
% of simultaneous non-linear equations.
% eps is the convergence criterion

function [sol,iter] = simul(x,eps)
error = 2*eps;
delta = 0.01;
iter = 0;
while error > eps
    J = jacobian(x,delta);
    F = mysystem(x);
    del = -J\F;
    x = x + del;
    error = sqrt(sum(del.*del));
    iter = iter + 1;
end
sol = x;
display(iter);
```

Figure 2. Matlab function for Newton’s method.

Students submit their program electronically via WebCT which puts a time stamp on their submissions. The instructor then uploads the programs to a local computer and grades them.

Student Feedback

No formal survey has been done regarding the switch from C to Mathcad and Matlab. The students in the numerical methods course have been obliged to learn to use these software packages as part of the course. This has put an extra burden on both the students and the instructor. In particular time had to be carved out of the course to instruct the students on how to use Mathcad and now Matlab. The first students to take the numerical methods course that have already passed through the introductory programming course will be in Fall 2004.

Anecdotal information suggests that at least some students have applied Mathcad in other classes under their own volition. At this time there is no formal requirement to use either Mathcad or Matlab in required courses, so this bodes well for wider use. In at least one course during the Fall 2003, Compressible Flow, the students were given a numerical assignment with the option to use any programming method they wished. Mathcad was the most popular choice.

Future Plans

The instructors of both the courses discussed in this paper will continue to monitor the relative preparedness of students as they complete their first course and enter the numerical methods course. Since the primary goal is to enable students to solve engineering problems, a concerted effort will be made to have other faculty “buy-in” to either Mathcad or Matlab. The current plan is to present a seminar to the faculty that demonstrates how both packages can be incorporated into existing courses, and to demonstrate the relative ease in learning these packages. It is imperative that students entering these classes have the necessary foundation and skills so that no time need be spent in class teaching the students programming. In addition, the authors will make themselves available for consultation with any faculty that wish for or require additional help.

The long term goal is make one or both of these packages an integral part of the undergraduate engineering curriculum. Not only will this expand the scope of problems that can be addressed in courses, but will provide the student will valuable skills when they enter the workforce.

References

1. Chapra, S. C. and Canale, R. P., “Numerical Methods for Engineers, 4th Ed.”, McGraw Hill (2002)

2. Brannan, K. P. and Murden, J. A., “From C++ to Mathcad: Teaching an Introductory Programming Course with a Non-Traditional Programming Language”, *Proceedings of the American Society of Engieering Education Conference*, (1998)