FORTRAN90, TKSOLVER and EXCEL - - A Comparison

Michael H. Gregg, Assistant Professor
Division of Engineering Fundamentals
Virginia Polytechnic Institute and State University
greggmh@vt.edu

Virginia Polytechnic Institute and State University currently requires its incoming freshman or transfer students to take two introductory engineering courses - - EF1005 and EF1006. These classes introduce and expand upon a number of engineering topics, but both culminate in projects.

EF1005, a course in engineering problem solving, teaches pencil-and-paper problem solving skills, introduces students to TK-Solver and its capabilities, and then teaches students how to generate computer solutions via their own FORTRAN programs. The class includes a semester long project which incorporates all of these problem solving skills.

This paper compares the use of FORTRAN 90 (Essential Lahey FORTRAN, ELF90), TK SOLVER 3.0 and EXCEL 5.0 in the solution of various types of freshman level engineering problems. The advantages and disadvantages of each in the creation of graphs, iterative solutions, back solving, and direct solutions are examined. Costs and software and hardware requirements are addressed.

Introduction

Virginia Tech, a land grant institution, is well into its second century. Its objectives are education, research, and community service. Virginia Tech has grown over the past 126 years from an institution with a student body of 43 to a current enrollment of about 25,000, comprising about 5000 new freshman each year. Of these entering freshman, roughly 1500 will enter the Engineering program.

Virginia Tech's engineering program puts an emphasis on immediately involving its entering engineering students in engineering topics. These introductory topics are structured to give these students a taste of engineering curricula and to expose them to problem solving techniques. Early involvement in engineering problem solving helps stimulate, refresh and/or retain the interest that these students have already shown in the engineering profession. Virginia Tech's success in maintaining a respected and rigorous engineering program is combined with an exceptionally high retention rate of its freshman in engineering . The latest figures indicate that nearly 70 percent of freshman engineering students graduate from Virginia Tech with an

engineering degree; an additional 20 percent graduate from Virginia Tech with other than an engineering degree, and the remainder transfer to another institution or take on other challenges. Although admission standards to the engineering program are reasonably high (average 1200 combined SAT and top 10% of class), we feel our high retention rate is partially attributable to the exposure the students gain in their first engineering classes. At Virginia Tech these classes are EF1005 and EF1006.

EF1005

This class is the students' first exposure to engineering at Virginia Tech. The faculty take this opportunity to present the students with a myriad of engineering topics. Although a two credit class, EF1005 meets for three 50 minutes classes each week. Sections are purposely kept small (maximum of 32) to engender a personal relationship among classmates and with their EF instructor/advisor. For ten years Virginia Tech has required entering freshman in engineering to purchase their own computer. This EF1005 class is, therefore, an introduction for many of these students to ownership, maintenance, and use of a computer.

EF1005's syllabus includes an introduction to the profession, examination of ethics and ethical theories and traditional pencil-and-paper problem solving techniques. These pencil-and-paper techniques are then translated to computer solutions via TK SOLVER. Exposure to TK SOLVER provides the student with needed familiarity with the computer and helps to prepare the student for FORTRAN programming. Although 'C' is also offered by the EF department, FORTRAN is the language used in our introductory courses. FORTRAN is a high level language and moderately easy to learn. The learning of a programming language by our students helps reinforce the logic and step-by-step approach required for all engineering problem solving. The use of subprograms -- subroutines and functions -- reinforces the need to break large problems into smaller, digestible pieces.

The FORTRAN project in EF1005 represents a significant level of programming -- both in quantity and structure. As a large project, it requires planning, not procrastination. Students learn through this process to plan their time, to design solution algorithms prior to writing code, to take one step or problem element at a time. EF faculty introduce upper class engineering topics as part of the project -- statics, heat transfer, queue theory as well as other selected topics. The majority of these students have not been exposed to engineering problems -- multi-faceted word problems -- prior to coming to Virginia Tech. The FORTRAN project represents their first real foray into engineering problem solving. Students passing EF1005 have gained an appreciation for their computers, have developed a more formal approach to problem solving, and have been exposed to some advanced engineering topics.

Scholarship students, those with elevated SAT scores or evidence of previous scholarly activities, are enrolled in special EF1005 sections, taught by the director of the Division of Engineering Fundamentals. These students receive instruction is spreadsheet usage (Microsoft Excel) as well as instruction in traditional problem solving methods, TK Solver, FORTRAN90 and C++.

This paper compares the usage of traditional methods, TK Solver, FORTRAN90 and EXCEL 5.0 in the solution of the Fall 1996 EF1005 project. Note that for this project students did not have the option of using spreadsheets, but were required to use TK-Solver, FORTRAN, and pencil and paper techniques.

The Project:

Although the semester project presented to students in EF1005 varies year to year and instructor to instructor, the introduction of statics is often a common element. Roughly four out of every ten freshman either drop their statics class or earn a grade less than a C-. Statics, in particular statics limited to two dimensional analysis, can be grasped readily by most freshman engineering students. The semester project given in the author's classes for the Fall semester of 1996 involved the use of 2-D statics, and was in part intended to help introduce the concept to the students one semester before they officially take the class. An abbreviated form of the project statement as supplied to the students is included as follows:

> A cantilever beam is shown in the attached diagram. The beam is supported at the right end by horizontal and vertical forces and by an internal moment. Up to five (5) planar point loads may be applied to this beam, each acting at some angle with the horizontal. Beam data are furnished (by the instructor) in one data file whose name is entered by the user at runtime (execution). The structure for this data is as follows:
>
> | | |
> |---|---|
> | record1: | beam length in feet, "L" |
> | record2: | beam depth, "b" = "c"*2 |
> | record3: | beam cross sectional area, "A" |
> | record4: | beam moment of inertia, "I" |
> | record5: | beam modulus of elasticity, "E" |
>
> The applied load data is supplied in a second data file, whose name will also be entered by the user during execution. The data in this file is as follows:
>
> | | |
> |---|---|
> | columns 21-22 | Integer number of load |
> | columns 23-32 | Magnitude of load in pounds force, "P" |
> | columns 33-38 | Direction of applied load, Cartesian degrees |
> | columns 1-4 | application point, inches from left end |
>
> Note the following:
>
> 1. We are assuming the principle of "superposition". That is, the effect of the combined loads is the sum of the effects of the individual loads.
> 2. The beam is weightless.
> 3. The vertical shear in the beam, "V" is equal to zero from "x" equal "0" to "a". The shear is the vertical component of the applied load from "a" to support.
> 4. The bending moment "M" is the area under the shear curve. From "0" to "a" this equals zero; from "a" to "a+b" (the right end of the beam) this equals: $V_R*(x-a)$

5. The bending stress, sigma, equals (M*c)/I.

6. The deflection is as given in the attachment. We are assuming that deflections are small, the beam is homogeneous, and stresses are within the proportional limit of the material.

You are to do the following:

1. Use load #1 and solve by hand for the reaction forces and reaction moment.

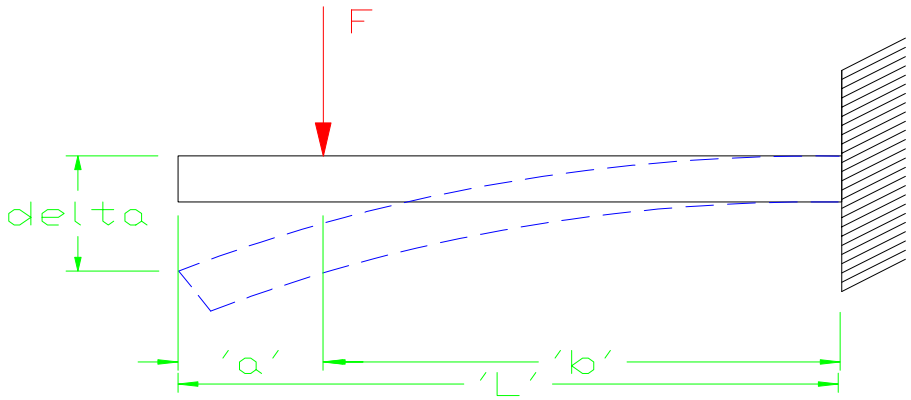
2. Use TK-Solver to solve for the following:
   Shear, "V" ; Bending Moment,"M"; Bending Stress, "sigma" and Beam deflection, "delta"

Determine these values in 1 inch increments along the beam, and plot the values. Load data will be from the FORTRAN input file. List the maxima and minima from the curves. Use user defined rule functions in TK to find the horizontal and vertical components of the applied loads, the shear, moment and deflection values.

3. Write a FORTRAN main program which will read the input data files and output tables of Shear, Bending Moment, Bending Stress, and deflection vs. position on beam, using 1 inch increments from left end. Your FORTRAN program should isolate maxima and minima from these calculations. Use individual function subprograms or subroutine subprograms to calculate these values of Shear, Bending Moment, and deflection. Use subscripted variables (2-D arrays) to store calculated values.

Your program is to ask the user if they wish output to the screen (console), the printer (LPT1) or a data file whose name the user will enter. All output should be presented clearly and neatly in the appropriate format. The program and subprograms are to be fully documented - use a separate sheet of paper for each program/subprogram. No printing should cover the perforations on the printer paper , and no folded output will be accepted. "Print-screen" output will not be accepted. Output to the screen must be controlled so that values do not “scroll” off the screen.

This is to be a user-friendly program. It should allow data entry from either the keyboard or the data files, at the users discretion. You must supply an exportable .EXE file in addition to your .F90 files on a virus-free diskette. No other files should be on this diskette. A virus detectable by the current version of McAfee will result in a 50 point penalty.



The solution approach here is direct. Given the formulas for shear, moment, stress and deflection, it is a fairly simple case of plugging in the necessary physical data via some repetition structure. No iteration or 'guessing' is required, however each solution 'tool' examined here - EXCEL, TKSolver, ELF90 - is capable of iterative solving.

Direct Solving:

FORTRAN and EXCEL require formulas to be entered in the form of an 'assignment' statement. This involves a single (unknown) variable on the left side of the equation, and some expression (algebraic, logical, procedural) or the right side. In this instance (as with most traditional engineering solutions) the engineer must resolve a series of equations into a series of assignment statements, and solve for the unknowns. By repeating the process in a sequential nature, the final assignment statement may be solved for the final (ultimate) unknown quantity. This may involve the algebraic manipulation of formulas in order to put them in the form of y = f(x).

TKSolver, in contrast, allows the user to enter equations as equalities - an algebraic expression on the left side of an equals sign and an algebraic equation on the right side. The traditional form of y = f(x) - an unknown variable equaling an algebraic expression - is not required. The user need not manipulate any of the given relationships, but merely list them (on the 'RULE' sheet.) In addition, TKSolver is not a sequential solver .... it attempts to satisfy all the relationships, equations, or 'rules' simultaneously. In contrast, FORTRAN and EXCEL require that formulas be addressed sequentially, one step at a time. Additionally, TKSolver 'backsolves'; that is, it will solve for any of the variables, presuming the relationships among variables are not over- or under-constrained, and also assuming that any other necessary variables have been defined. With FORTRAN or spreadsheet applications, an assignment statement would have to be written specifically with that variable on the left hand side of the equality.

For EXCEL, the calculation of the deflection of a cantilever beam appears as follows:

[current cell value] = deflection(A6,A15,B6,C6,D6,E6,F6)

where 'deflection' is a procedure function (called a 'module' in EXCEL) which incorporates the argument list in parenthesis.

```
Function deflection(BLEN, X, FY, E, I, A, C)
   B = BLEN - A
   If X < A Then

      deflection = FY / (6 * E * I) * (-B ^ 3 + 3 * BLEN * B ^ 2 - 3 * X * B ^ 2)
   End If
   If X >= A Then

      deflection = FY / (6 * E * I) * ((X - A) ^ 3 - 3 * (X - A) * B ^ 2 + 2 * B ^ 3)
  End If
End Function
```

The formula as listed in the EXCEL 'cell' is not intuitive....it must be interpreted as a function of the values stored in cells identified by an alphanumeric code. The EXCEL module, in this case a function, is very similar to a 'rule function' in TK, or to typical lines of FORTRAN code. However, writing this module in EXCEL would therefore require an understanding of the sequential and decision structures involved.

The formula as listed in the EXCEL 'cell' represents a single deflection value. In order to calculate values across the beam (from 'free' end to 'fixed' end), this formula would have to copied to additional cells. Commonly, a complete row or column would be copied, representing increments of an independent variable (such as location on beam) and the results would be updated automatically.

For TKSOLVER the calculation of deflection takes on the following form:

CALL VMSD(A,L,F,ANGLE,X,C,I;V5,M5,S5,D5,FY)

where the function VMSD is as follows:

```
IF X<=A THEN V=0 ELSE V =-FY     ;V=SHEAR, A=LOADPOSITION, FY = VERT COMP
M = V*(X-A)                      ; M = MOMENT
S = M*C/I        ;S=STRESS, C= HALF BEAM DEPTH, I = BEAM MOMENT OF INERTIA
IF X<=A THEN D=FY/(6*E*I)*(-B^3 + 3*B^2*L - 3*B^2*X)    ;E=MOD OF ELAS, B =L-A
IF X>A THEN D = FY/(6*E*I)*( (X-A)^3 - 3*B^2*(X-A)  + 2*B^3)
B=L-A      ;B = INCHES FROM LOAD TO SUPPORT, A = DISTANCE FROM END TO LOAD.
FY=F*SIND(ANGLE)        ;VERTICAL COMPONENT OF LOAD
```

Accessing the function in TK is more intuitive for the user. The independent variables that make up the argument list have variable names that may be descriptive of their values. EXCEL's function references cell addresses. The TK user, as with EXCEL, must understand the sequential and decision structures of the software packages in order to write the correct code. The same logic applies to the FORTRAN code:

```
      DO X=0,INT(BLEN),1
!           ! Set some local variables...A and B where A is the length from the free
!           ! end of the beam to the load, and B is the length from the load to support.
!           ! using A and B simplifies the equations. P is the vertical load component.
          A = LOAD(L,1)
          B = BLEN - A
          P = LOAD(L,3)*SIN( LOAD(L,4)*PI/180.)
          IF( X<=INT(A) ) THEN        !NOTE: a 3-D array could have been used
             V(L,X) = 0.0          !for example, VALUE(L,Z,X) where L is the load
             M(L,X) = 0.0          !Z is shear, moment, stress or deflection
             S(L,X) = 0.0          !and X is the position on the beam
             DELTA(L,X) = P/(6*E*I)*(-B**3 + 3*BLEN*B**2 - 3*X*B**2)
!                 From Nyhoff and Leestma:
!                 Delta = P/(6EI)*(-b^3+3b^2*L -3b^2*X) end of beam PAST load
!                 Delta = P/(6EI)[(x-a)^3 - 3b^2*(X-a) + 2b^3] load to support
            V(6,X) = V(6,X) + V(L,X)        !RUNNING TOTALS
            M(6,X) = M(6,X) + M(L,X)
            S(6,X) = S(6,X) + S(L,X)
            DELTA(6,X) = DELTA(6,X) + DELTA(L,X)
```

This code provides repetition or looping as well. The dependent and independent variables may be given names which are descriptive. Repetition in this case involves the use of multiple subscripts, requiring a mastery of the 'array' concept by the user. This code, as well as its TK counterpart, incorporates 'comments' about the code and the programmers logic along with the assignment statements. With EXCEL, comments can be imbedded in the background and viewed when desired.

Iterative Solving

Both TKSolver and EXCEL provide a direct route to an iterative solution. TKSolver allows the user to 'guess' a value for the answer, and then request a solution via the 'direct solve' method. If the solution is not reached (within some error factor) in the required number of iterations, the process may be repeated until a solution is reached.

Excel requires setting up the formulas with the unknown value set to a cell address. This is a somewhat more cumbersome process than that employed by TK Solver. ELF90, logically,

requires the user (programmer) to write the necessary iterative structure to find the root or solution. For each separate variable which must be found iteratively, a structure must be written to satisfy that search. Solving iteratively in FORTRAN (or in most other programming languages) is much more complicated than iterative solving with a software 'package'. On the other hand, the iterative solver that is written in a programming language can be tailored for the specific problem. It may be optimized to reduce processing time, or to avoid the problem of non-convergence of the solution.

Units:

Unit conversions may be readily imbedded in FORTRAN and EXCEL, however TKSolver recognizes specific units as ASCII character strings and will convert from one set of units to another. Conversion is accomplished either through the TKSolver Library, or through user-defined definitions. Changes in input or output can be incorporated immediately by changing the units specified for these values. To incorporate different units in a FORTRAN program, for example, the programmer would have to provide the proper decision structure to query the user as to the intended input units and the intended output units. EXCEL requires a conversion as part of the original function, or as a separate cell to do the conversions.

Repetition Structure:

TKSolver and EXCEL both rely on lists for repetition structures. In terms of the 1996 VT FORTRAN project, the unknown values can be calculated in one inch increments across the beam in EXCEL by copying the formula cells (rows). Each new row represents an increment of the location value (x), and the corresponding values of shear, moment, stress and deflection.. TKSolver addresses this problem via 'list solving'. This involves applying the same TK rules to successive values of an input variable, producing an output list (or lists). In addition, TK allows the user to define specific procedure functions and can address individual list elements via subscripts.

FORTRAN, as expected, will perform whatever repetition structure is provided by the programmer. This provides additional flexibility at the expense of added user programming effort. The effort by the programmer reduces the flexibility of the overall FORTRAN program, but eases the process for the user of the FORTRAN executable code.

Input/Output:

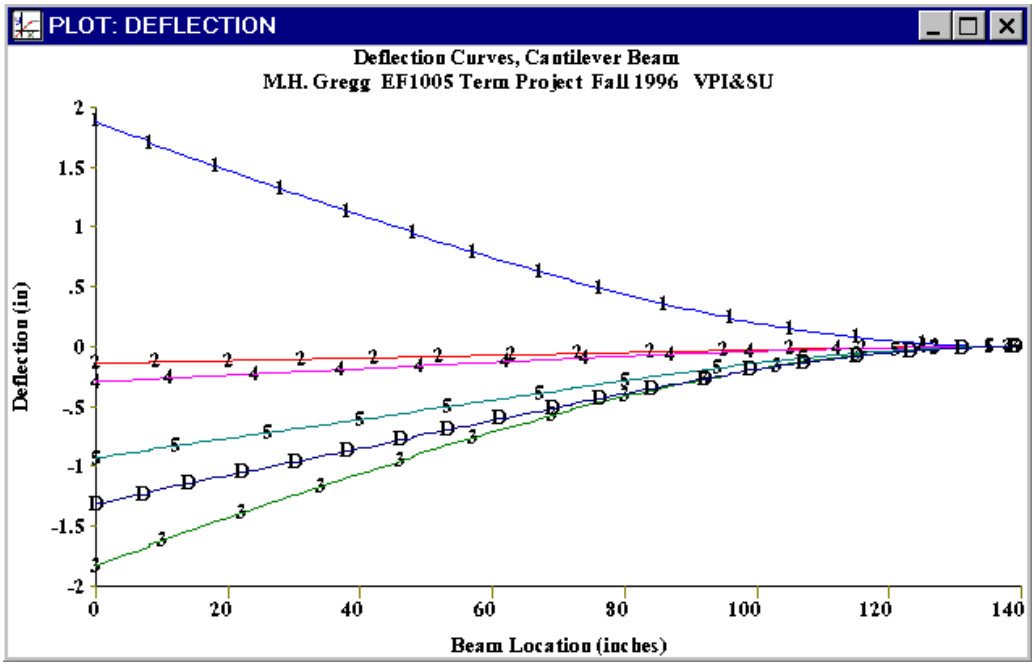
FORTRAN provides whatever input/ the programmer is willing to incorporate in the code - real or integer values, SI or English units, output to the monitor, hard drive, floppy, modem or other device. Format of the output is determined by the limitations of the output device and the ingenuity of the programmer. TK and EXCEL have limited output options. In the simplest case they both provide output to the screen, and this output can be manipulated to allow different

units, format and location. TK's output may be sent to a file, but its structure is limited. Input is almost solely limited to user input from the keyboard into the different cells (for EXCEL) or into different variables or lists (TK).

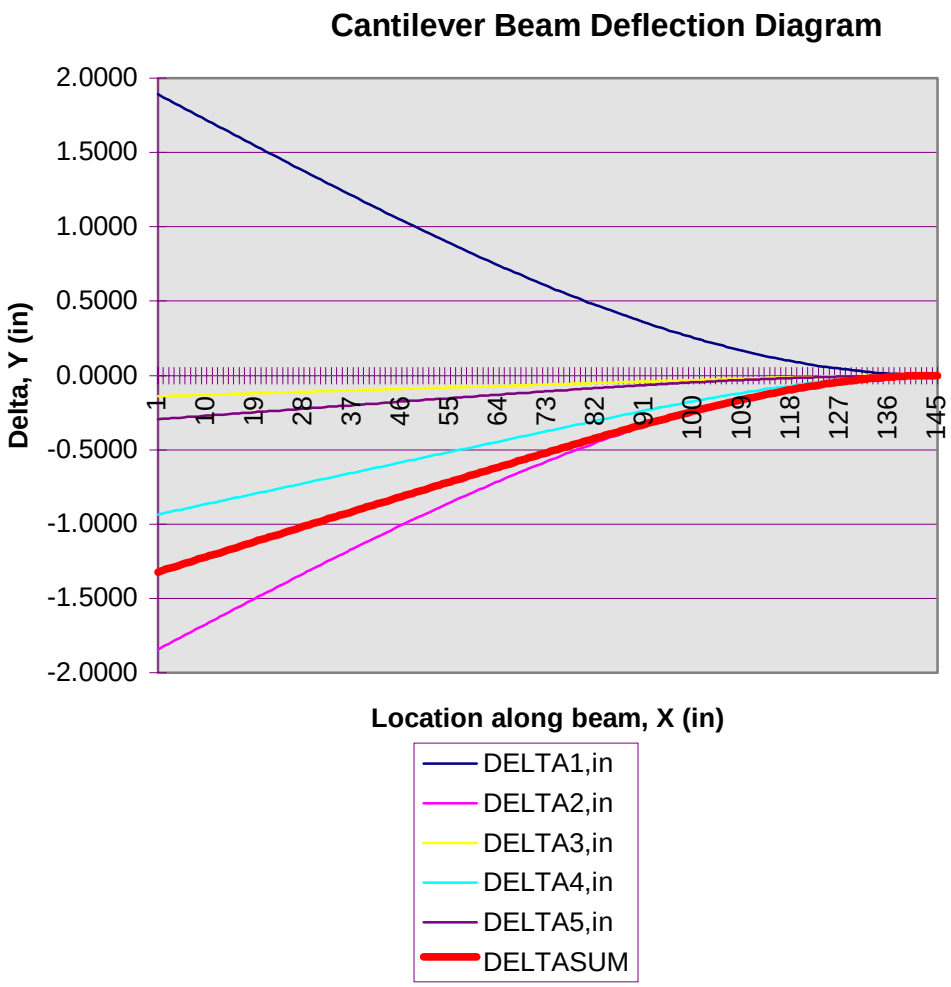
Graphing:

TK and EXCEL both provide excellent graphing capabilities. I find it somewhat easier to set up the abscissa and ordinate values in TK, rather than EXCEL. In FORTRAN, graphing is accomplished by the 'brute force' method - that is, export the values to a properly constructed data file, import those values into TK and use TK's plot capabilities to view the data graphically. It is certainly possible to generate plot directly from FORTRAN, but unless the code is complex, these are highly rasterized plots, whether on the screen or from the printer. Labeling the axes and graduating and calibrating those axes in FORTRAN becomes much more complicated.

TKSOLVER Graph



EXCEL Graph



Overall:

FORTRAN programs, once written, provide a uniform, potentially 'idiot-proof' means of repeating a calculation sequence time after time. FORTRAN programs would generally be written for situations where the same program was to be re-run numerous times with different data sets. For the engineering problem that need be solved only once, writing the FORTRAN code for it would be time consuming. The same problem (in most cases) could be readily solved through a TK Solver model or an EXCEL spreadsheet. In each of these cases, the user must be intimately familiar with the language or software package in order to effect a solution.

In terms of the semester project my students addressed at Virginia Tech, TKSolver produces results, both in text and graphically, faster and with less 'learning curve' that FORTRAN. On the other hand, if the end product is to be used repetitively (perhaps by a structural engineering firm), the FORTRAN end code requires no programming knowledge by

the user. If the same problem were to be solved repetitively in either TK or EXCEL, the user would have to be familiar with those packages. The FORTRAN codes allows the end program to be interactive - to query the user as to input and output and what-ifs. Both TK and EXCEL are not interactive....the user must re-enter the program and modify values in order to see new results. The user must also know how to interpret the results of those changes ....in what cells or on what 'sheets' will the output be available.

Pedagogically, I prefer the use of FORTRAN. Because of the sequential nature of the programming language the student addresses each line of code, or each sub program, as the solution to a problem step. By breaking a complicated problem into smaller pieces, the students recognize and learn to appreciate this approach to problem solving. I feel the student must have a deeper understanding of the underlying problems to effect a FORTRAN solution. With TKSolver, the user may add multiple rules (or equations/relationships), in any order, provide values for those known variables, and either directly solve or iteratively solve for an answer. Although the answer may be easier to obtain, the process may have become obscured. EXCEL requires the sequential approach of FORTRAN, but lacks the input/output flexibility. FORTRAN deals easily with subscripted variables (multi-dimensional arrays), whereas TK and EXCEL are more cumbersome in their approach to these topics. The software packages provide convenience to the user, and fine graphical output. FORTRAN provides a programmer-determined level of flexibility at extra effort.

Costs:

Each of these packages require essentially the same hardware. IBM compatible machine, 486 or better processor, 16 MB of machine memory, about 20 MB of hard disk space (minimally). ELF90 is 32 bit code, and will run better under WINDOWS95 or WINDOWS-NT. Educational list pricing is as follows:

ELF90 and Editor:

Elf90 2.00	195.00
ED for Windows	99.00

Elf90 System Requirements

486DX, Pentium, or Pentium Pro
DOS 3.3 or higher and Windows 3.1x, Windows 95, Windows NT
8MB RAM, 16MB recommended, 32MB recommended with Windows 95 or Windows NT
12MB hard disk space

TK Solver Release 3 for Windows:

$49 TK Solver for Students
$99 TK Solver for Faculty

TK Solver will run on any computer capable of running Windows95.
Requires about 10 MB disk space.

MICROSOFT EXCEL

EXCEL $97 OFFICE $178 (Tech bookstore)
Microsoft Excel 97 Spreadsheet $339.00 (Microsoft)

Personal or multimedia computer with a 486 or higher processor
Microsoft Windows 95 operating system or Microsoft Windows NT 8 MB of memory for use on Windows 95; 16 MB for Windows NT
36 MB of available hard-disk space required; typical
CD-ROM drive VGA or higher-resolution video adapter
Microsoft Mouse, Microsoft IntelliMouse

Bibliography:

TKSolver Home Page

MicroSoft EXCEL Home Page

Lahey FORTRAN Home Page

TKSolver Release 3 User's Guide, Universal Technical Systems, Rockford, Ill, 1994

GOTTFRIED, BRYAN S., Spreadsheet Tools for Engineers, EXCEL5.0 Version, McGraw-Hill, New York, 1996

ELF90 - Essential Lahey Fortran 90, Lahey Computer Systems, Inc., Incline Village, NV, 1996

ELLIS, T.M.R., I.R. Phillips, T.M.Lahey, Fortran 90 Programming, Addison-Wesley, New York, 1994