AC 2007-456: IMPROVING PROBLEM-SOLVING SKILLS THROUGH ADAPTING PROGRAMMING TOOLS

Linda Shaykhian, NASA

Linda H. Shaykhian Linda Shaykhian is a computer engineer with the National Aeronautics and Space Administration (NASA), Kennedy Space Center (KSC). She is currently co-lead of the Information Architecture team for the Constellation Program's Launch Site Command and Control System Proof of Concept project. She was lead of the Core Technical Capability Laboratory Management System project, which is currently used for resource management and funding of KSC Core Technical Capability laboratories. She was the Software Design Lead and Software Integrated Product Team Lead for the Hazardous Warning System on KSC's Checkout and Launch Control System Project. Linda has been involved in various other software projects at Kennedy Space Center and also has a background in quality assurance. She has earned both Bachelor of Science and Master of Science degrees in Computer Science.

Gholam Ali Shaykhian, NASA

Gholam "Ali" Shaykhian Gholam Ali Shaykhian is a software engineer with the National Aeronautics and Space Administration (NASA), Kennedy Space Center (KSC), Engineering Directorate. He is a National Administrator Fellowship Program (NAFP) fellow and served his fellowships at Bethune Cookman College in Daytona Beach, Florida. Ali is currently pursing a Ph.D. in Operations Research at Florida Institute of Technology. He has received a Master of Science (M.S.) degree in Computer Systems from University of Central Florida in 1985 and a second M.S. degree in Operations Research from the same university in 1997. His research interests include object-oriented methodologies, design patterns, software safety, and genetic and optimization algorithms. He teaches graduate courses in Computer Information Systems at Florida Institute of Technology's University College. Mr. Shaykhian is a senior member of the Institute of Electrical and Electronics Engineering (IEEE) and is the Vice-Chair (2005-2007), Education Chair (2003-2007) and Awards Chair of the IEEE Canaveral section. He is a professional member of the American Society for Engineering Education (ASEE), serving as the Program Chair and Web Master for the Minorities in Engineering Division of ASEE (2006-2008). He was an assistant professor and coordinator of the Information Systems program at the University of Central Florida prior to his full time appointment at NASA KSC.

Improving Problem Solving Skills through Adapting Programming Tools

Introduction

There are numerous ways for engineers and students to become better problem-solvers. The use of command line and visual programming tools can help to model a problem and formulate a solution through visualization. The analysis of problem attributes and constraints provide insight into the scope and complexity of the problem. The visualization aspect of the problem-solving approach tends to make students and engineers more systematic in their thought process and help them catch errors before proceeding too far in the wrong direction. The problem-solver identifies and defines important terms, variables, rules, and procedures required for solving a problem. Every step required to construct the problem solution can be defined in program commands that produce intermediate output.

This paper advocates improved problem solving skills through using a programming tool. MatLab, created by MathWorks, is an interactive numerical computing environment and programming language. It is a matrix-based system that easily lends itself to matrix manipulation and plotting of functions and data. MatLab can be used as an interactive command line or a sequence of commands that can be saved in a file as a script or named functions. Prior programming experience is not required to use MatLab. MatLab visual and command programming are presented here. The GNU Octave, part of the GNU project, a free computer program for performing numerical computations, is comparable to MatLab.

MatLab Programming Language

The easy-to-use computing environment of MatLab, along with the availability of thousands of built-in functions for mathematics, statistics, simulation, data analysis and general purpose programming, has rendered it a popular software tool for many engineers and scientists in various fields. MatLab is a dynamically typed language [8]. In a dynamically typed language, impending errors are caught at run-time. MATLAB is delivered as part of an integrated computing environment facilitating programming in three different ways, known as shell, script and function. Shell, a single command line typed directly from a keyboard, uses hundreds of pre-built mathematical functions. MatLab programming constructs (decision statements, loop statements, etc.) and pre-built

mathematical functions stored in a plain text file (script file) are executed by typing the script file name directly from the MatLab command prompt. MatLab functions are similar to Fortran subroutines, or "C" functions.

MatLab is an interpreted programming language; the interpreter translates each command line to machine code every time it is executed, whereas compiled code performs the translation once to perform the desired function. Memory allocated for variables is slower in an interpreted programming language because the mapping of the variables to the actual storage location is done repeatedly during run time. A variable's data type is defined by its corresponding value; in the same code, a variable can take on different data values, thereby changing its data type. For example, a variable K1 can take on the integer value 87, float value 63.45, string value "ASEE" or character value 'A' all in the same code. Running interpreted code is slower than running compiled code because of run time memory allocation and program statement transactions each time the code is executed.

A MatLab function can have both input and output parameter lists. The input parameters are those values sent to the function for processing, and the output parameters are the expected results. When calling a function, the caller provides the function with input parameters (variables or values), and the function calculation result is passed back to the caller in a list. A MatLab program file name must end with extension ".m", commonly known as an m-file. Although is possible to store several functions in a single m-file, we suggest that each m-file contain exactly one MatLab function; this will lead to having a collection of reusable functions.

The MatLab *help* command displays a list of functions for which on-line help is available; the command *help function-name* displays information about a specific function. The command *help sort* will give information about sorting an array of elements in ascending or descending order. In command *sort(A)*, *A* can be a vector, in which case the sort command will sort the elements of *A*. *A* can be a matrix, in which case, the sort command will sort each column of *A*. To sort *A* in descending order, the sort command is written as *sort(A, 'descend')*.

Problem-solving skills

A problem solution can be a single command line code (that sorts a vector, for instance), or may involve several functions (for example, evaluating the shortest path in the traveling salesman problem [9]). For the latter, the solution may require going through several steps for planning and analysis of the problem, elaborating the problem by defining the tasks involved [1, 6]. These steps must address: What is exactly required? What must be done first? What can be left until later? What is already known to approach the current problem? This strategy can help characterize the problem and visualize how to achieve the target solution by measuring progress made in each step. Once this is done, then options for different solutions can be evaluated. Working towards the best solution is facilitated by knowing what would make a 'best possible solution'. How far is this feasible under the circumstances [6]? Hidden advantages and flaws, and previous

attempts and failures should be examined. Are the right resources available for each potential solution? What feedback has been received from others? What does this reveal about the solution's performance?

It is important to clearly understand the problem definition and generate alternative solutions. It is advisable to first discard the implausible features of the problem, and come up with a list of possible options that will lead to an improved solution. The decision of which option to implement should be examined for selecting a realistic solution. Once all of the options have been evaluated, and one that seems to accomplish the goals has been selected, it is time to test it. During this stage, should continued assessment of the chosen solution and the degree to which it is "solving" the problem is recommended [6].

Programming in MatLab

Input/Output is one of the most desired programming features. The keyboard is usually the interactive standard input device. Handling the standard input in MatLab in done by applying the *input* command. The *input* command has two parts: (1) a prompt to notify the user that input is waiting and (2) a variable assigned to the value of the user's input. By default, the standard output of a program is the screen. MatLab *disp* command is used to display a string or a variable.

```
City_name = input('Which city will host the 2008 ASEE Annual Conference?')
disp(City_name)
```

The C language conversion specifications control the notation, alignment, significant digits, field width and other features of the display format. The *sprintf* command can be used to format the display information. The conversion specification syntax uses the percent sign followed with a conversion character designating the conversion, for example, %8.4e, uses 8 columns for the field width, 4 precision, the character 'e' (exponential notation) is the conversion character.

A MatLab program can open and use data files for reading or writing. The ability to read from or write to a file depends on the file mode permission specified for the file opening. The file input/output uses the MatLab fopen command. The fopen command general syntax for opening a file is:

```
file_identifier = fopen(filename)
file_identifier = fopen(filename, file_permission_mode)
```

The file identifier is used as the first argument to the subsequent file input/output routines, the filename is a string enclosed in single quotes. The characters '*r*', '*w*', '*a*', '*r+*', '*w+*' and '*a+*' are used to represent the file permission mode (default permission mode is to open a file for reading – character '*r*'). The following script demonstrates opening a file for input and output. Comments are preceded by %.

```
% create a new file, for writing; discard existing contents, if any.
fid = fopen('afile.txt','w');    % 'w' means open file to write text
if (fid < 0)
   error('could not open file "afile.txt"');
```

```
end;

for x = -pi:0.01:pi
    fprintf(fid,'%8.3g %8.3g\n',x,sin(x)); % write some stuff to file
 end;
fclose(fid); %  always close the file when done
fid = fopen('afile.txt','r');     % 'r' means open file to read from

% read from file into table with 2 rows and 1 column per line
data = fscanf(fid,'%g %g',[2 inf]) % It has two rows, inf-reads to eof.
data = data'; % transpose the matrix
data
fclose(fid);
```

Problem Solving Methodology

Problem 1: The following illustrates steps involved in using the MatLab programming tool to solve a problem and conduct data analysis. Problem statement: Develop a program to calculate the average, standard deviation, max and min weight of 100 students. Step 1: The weight data is loaded from a weight data file into a matrix variable or alternatively, is input from the keyboard using the *input* command. This known weight data is required to solve the problem. Step 2: The MatLab *mean*, *std*, *max*, and *min* commands are used to calculate the average, standard deviation, max and min weight of 100 students. A different solution option to perform data analysis could be to write a custom function to calculate the average, standard deviation, max and min weight of 100 students. These two options should be evaluated for efficiency and accuracy. The optimal solution is selected by the problem-solver. In this example, there is no value added by writing a custom function to replace MatLab commands. Step 3: To perform data analysis, use MatLab graphing capabilities are used. The MatLab program code is shown below; the *plot* and *hist* commands produce the line graph and histogram as shown in Figure 1.

```
% Script listing to calculate the average, standard deviation, max and min weight of
% 100 students and produce both line graphs and histogram.
% This script loads (reads) a data file, then plots the data
% Data file: Weight of 100 students
% clear removes all variables from the workspace. This frees up system memory.
clear

load weight.dat; % load weight.dat file,weight becomes matrix variable

% Find the mean, standard deviation, max and min
average_weight = mean(weight);
standard_deviation = std(weight);
maximum_weight = max(weight);
minimum_weight = min(weight);
```

```
disp(sprintf('\n\tAverage Weight: %6.2f', average_weight));
disp(sprintf('\n\tStandard Deviation: %6.2f', standard_deviation));
disp(sprintf('\n\tMaximum Weight: %6.2f', maximum_weight));
disp(sprintf('\n\tMinimum Weight: %6.2f', minimum_weight));

subplot(1,2,1) % set the plot area- subplot: row=1, column=2, pane=1

plot(weight) % plot (linear 2-D plot)of weight data

title('Weight of Freshman Class Men')
xlabel('Student Number')
ylabel('Weight, lbs')
grid on

% subplot: row=1, column=2, pane=2
subplot(1,2,2)

hist(weight) % plot (histogram plot)of weight data
xlabel('Weight, lbs')
ylabel('Number of students')
title('Weight of Freshman Class Men')

disp(sprintf('\n\tDone'));
```

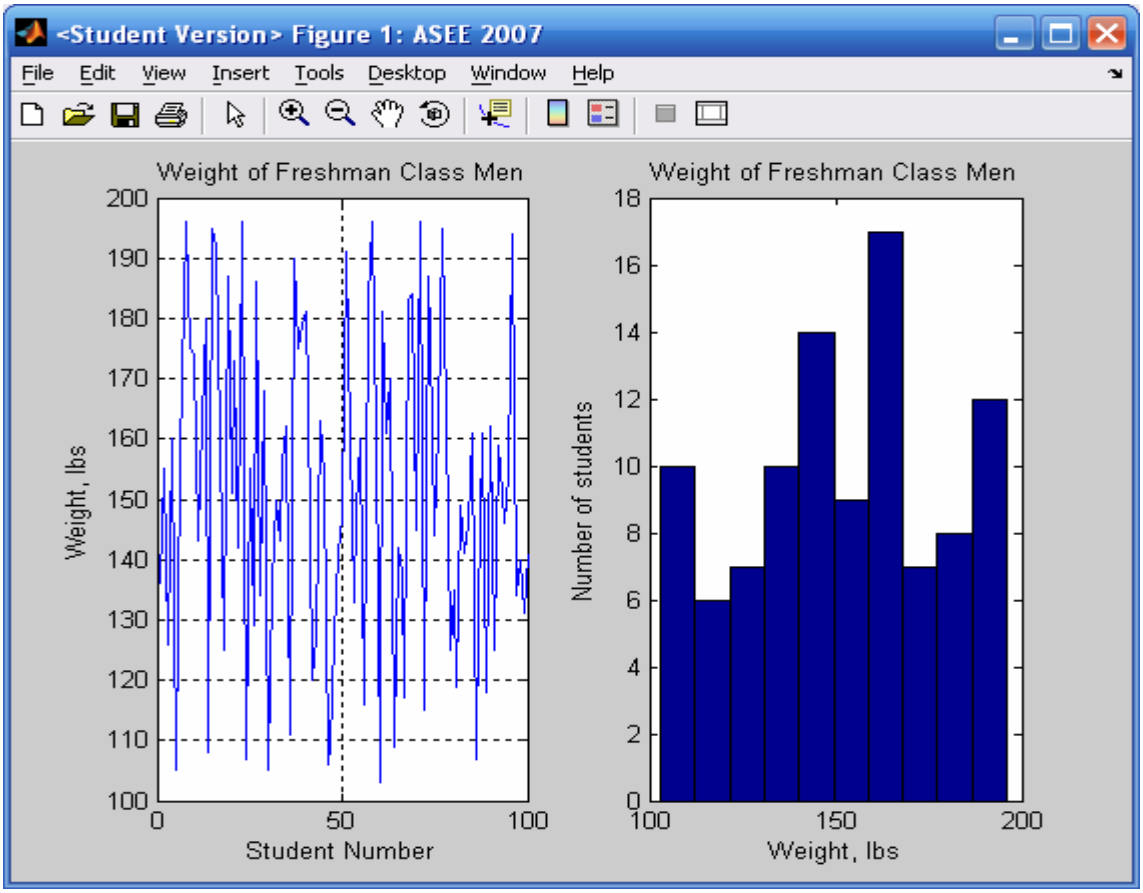

Figure 1

The MatLab *disp* command shows the following output on the screen. To format the output, the *sprintf* command is used. The MatLab command prompt is ">>", the word "EDU" indicates that the MatLab educational version is used for the preparation of this script.

EDU>> script1

Average Weight: 151.47

Standard Deviation: 26.34

Maximum Weight: 196.00

Minimum Weight: 103.00

Done

A run of this script produces an average weight of 151.47 pounds, a standard deviation of 26.34 pounds, a maximum weight of 196.00 pounds, and a minimum weight of 103.00 pounds. This particular problem uses weight data as the only parameter that may be used to produce different solutions. The weight data file and graph of the experimental data shown in Figure 1 can easily be changed; a new graph and numerical analysis

(calculations of average, standard deviation, maximum and minimum weights) is produced by re-running the script, this will take only a few keystrokes.

Problem 2: By definition, π is the ratio of the circumference of a circle to its diameter. The constant π is an irrational real number, which is approximately equal to 3.14159. Two equations are shown in Table 1 for calculating the approximate values of π. Two solution options are performed to evaluate the approximate values of π; the analysis of the data reveals information as to the quality of the solutions. These two options should be evaluated for efficiency and accuracy. The optimal solution is selected by the problem-solver. The data shown in the right columns [11] of Table 1 reveal that just after 6 terms, a convergent is obtained. The data in the left column [10] shows that the same accuracy is obtained after 14 terms.

i	$\pi = 3\sum_{i=0}^{\infty} \frac{(2i)!}{i!^2\,(2i+1)2^{4i}}$	$\pi=\sum_{i=0}^{\infty}\frac{1}{16^i}\left(\frac{4}{8i+1}-\frac{2}{8i+4}-\frac{1}{8i+5}-\frac{1}{8i+6}\right)$
0	3.0000000000	3.1333333333
1	3.1250000000	3.1414224664
2	3.1390625000	3.1415873903
3	3.1411551339	3.1415924576
4	3.1415111723	3.1415926455
5	3.1415767158	3.1415926532
6	3.1415894253	3.1415926536
7	3.1415919824	3.1415926536
8	3.1415925112	3.1415926536
9	3.1415926229	3.1415926536
10	3.1415926469	3.1415926536
11	3.1415926521	3.1415926536
12	3.1415926533	3.1415926536

13	3.1415926535	3.1415926536
14	3.1415926536	3.1415926536
15	3.1415926536	3.1415926536

Table 1

```
% Script listing to calculate approximate values of π
pi1=0; pi2=0;
for i=0:15
  pi1= pi1+ 3*factorial(2*i)/(factorial(i)^2*(2*i+1)*2^(4*i));
  pi2= pi2+ (1/16^i)*(4/(8*i+1)-2/(8*i+4)-1/(8*i+5)-1/(8*i+6));
  disp(sprintf('%3d %20.10f %20.10f\n',i,  pi1, pi2));
end
```

Conclusion

As discussed in the paper, considerable details about a problem can be exploited through the use of a programming tool. MatLab is best suited for scientists and engineers, has hundreds of standard functions supporting systems of linear equations, Eigenvalue and Eigenvector computations, matrix factoring, data analysis and much more. We briefly examined the MatLab graphical capabilities to demonstrate the analysis of problem attributes and constraints. The visualization aspect of this problem-solving approach provides real insight into internal problem mechanisms and the performance of the problem solution.

Bibliography

1. Problem Solving Skills, http://www.athealth.com/Consumer/disorders/problemsolving.html

2. Zywno, M., Instructional Technology, Learning Styles and Academic Achievement, Proceedings of the 2002 American Society for Engineering Education Annual Conference & Exposition.

3. Matušu, R. and Prokop, R., User-friendly MATLAB Program for Control and Robust Stability Analysis of Systems with Parametric Uncertainties, Proceedings of the 13th Mediterranean Conference on Control and Automation Limassol, Cyprus, June 27-29, 2005.

4. Daku, B., Jeffrey, K., An Interactive Computer-Based Tutorial for MATLAB, 30th ASEE/IEEE Frontiers in Education Conference, October 18-21, 2000.

5. Wirth, M., Kovesi, P., MATLAB as an Introductory Programming Language, 2006 Wiley Periodicals Inc.

6. Teaching Problem-Solving Skills, Prepared for the TRACE Workshop, "Teaching Problem-Solving Skills," June 17, 2003.

7. Navaee, S., Das, N., Utilization of MATLAB in Structural Analysis, Proceedings of the 2002 ASEE/SEFI/TUB Colloquium.

8. Wikipedia encyclopedia, http://en.wikipedia.org/wiki/Dynamically_typed_language.

9. Sen, S., and Shaykhian, G. Scope of various random number generators in ant system approach for TSP, Proceedings of the 2007 American Society for Engineering Education Annual Conference & Exposition.

10. http://en.wikipedia.org/wiki/Pi

11. Bailey, D., Borwein1and, P., Plouffe, S., On The Rapid Computation Of Various Polylogarithmic Constants, 1991 Mathematics Subject Classification, 11A05 11Y16 68Q25.

Appendix

Additional examples are provided in the Appendix section to demonstrate MatLab functionalities. The MatLab user guide and the online help system provide examples for hundreds of commands that are readily available.

A MatLab built-in or user defined function is entered from the command prompt. The command prompt is the default mode, it is designated with ">>" or "EDU>>" prompt. The example below shows a single MatLab *sin* command entered to evaluate the *sin* of 30 degree (pi/6) to produce 0.50:

*EDU>> sin(*pi/6*)*

Multi Plot Script: A script is used here to demonstrate the multi plot features of MatLab. A mesh plot, a surface plot, a contour plot, and a combination of surface and contour plot are shown all in one screen in Figure 2. Placing these plots next to each other should help to better visualize the problem.

```
clear % clear removes all variables from the workspace.
clc %clear screen

% The linspace function generates linearly spaced vectors. It is similar to the colon
% operator ":", but gives direct control over the number of points.
% y = linspace(a,b,n) generates a row vector y of n points linearly spaced between and
% including a and b.  x=linspace(0,10*pi,1000);
%
```

```
% The Colon Operator (:) is used to create a vector containing:
% -2.0000   -1.8000   -1.6000  ..... 1.6000    1.8000    2.0000 has 21 cells
x=[-2:0.2:2];
y=[-2:0.2:2];

% meshgrid - Generates X and Y matrices for three-dimensional plots
% [X,Y] = meshgrid(x,y) transforms the domain specified by vectors x and y into arrays
% X and Y, which can be used to evaluate functions of two variables and three-
% dimensional mesh/surface plots. The rows of the output array X are copies of the
% vector x; columns of the output array Y are copies of the vector y.
%
% For example, if the [X,Y] = meshgrid(1:3,10:14) are:
%
% X =
%     1     2     3
%     1     2     3
%     1     2     3
%     1     2     3
%     1     2     3
%
% Y =
%    10    10    10
%    11    11    11
%    12    12    12
%    13    13    13
%    14    14    14
[X,Y]=meshgrid(x,y);

% .* operator is used to perform an element by element multiplication
Z=X.*exp(-X.^2-Y.^2);
subplot(2,2,1)

% mesh(X,Y,Z) draws a wireframe mesh with color determined by Z so color is
% proportional to surface height.
mesh(X,Y,Z)
title('Mesh Plot')
xlabel('x-axis'), ylabel('y-axis'), zlabel('z-axis')

subplot(2,2,2)
% surf(X,Y,Z) creates a shaded surface using Z for the color data as well as surface
% height.
surf(X,Y,Z)
title('Surface Plot')
xlabel('x-axis'), ylabel('y-axis'), zlabel('z-axis')

subplot(2,2,3)
```

```
contour(X,Y,Z) % contour(X,Y,Z)draws contour plot of Z
title('Contour Plot')
xlabel('x-axis'),  ylabel('y-axis'), zlabel('z-axis')

subplot(2,2,4)
surfc(X,Y,Z) % surfc(...) draws a contour plot beneath the surface.
title('Combination Surface and Contour Plot')
xlabel('x-axis'), ylabel('y-axis'), zlabel('z-axis')
```

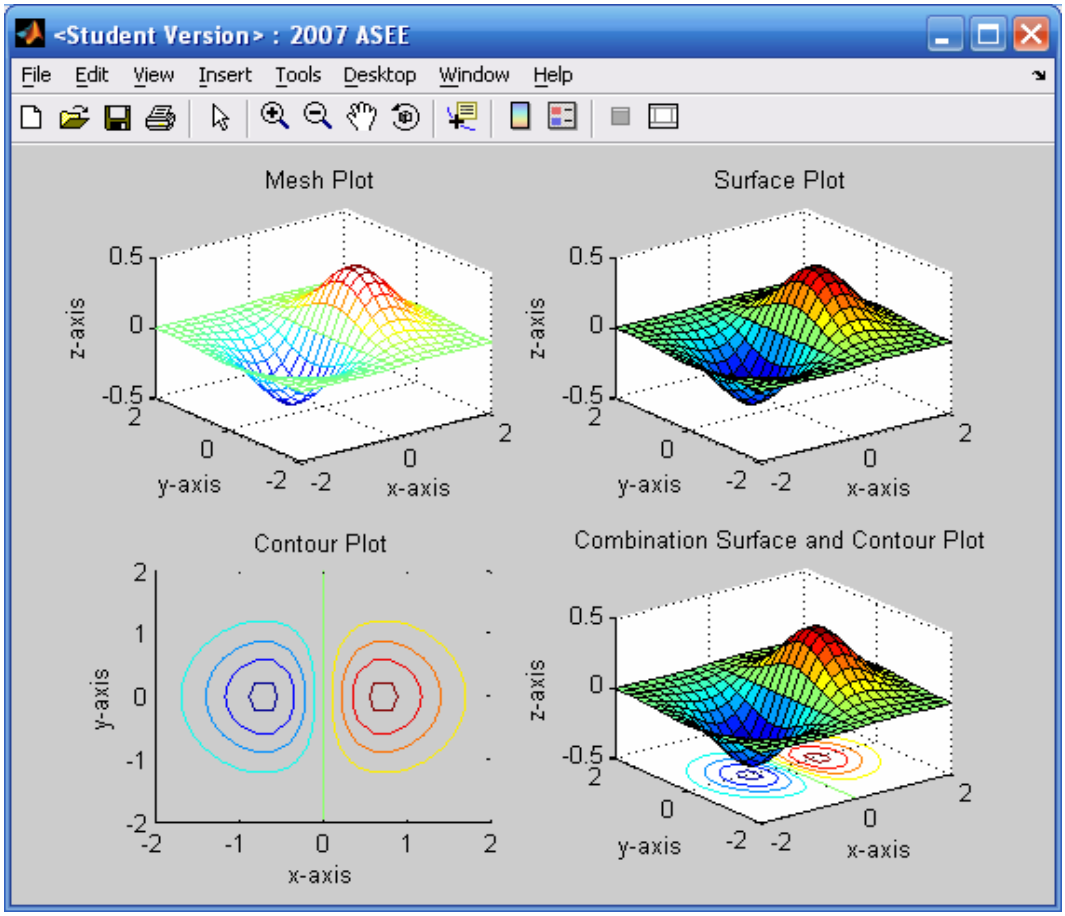

Figure 2

Plot 3 Script: The MatLab command *plot3* is used to produce a three dimensional graph. x, y and z axes are labeled using *xlabel*, *ylabel* and *zlabel* commands. The script below constructs a 3-dimensional graph for a spring, the graph is titled using the MatLab *title* command. The *linspace* command (function) takes three arguments (*a*, *b*, *n*), and generates linearly spaced row vectors, the third argument indicates *n* points linearly spaced between and including *a* and *b*. Figure 3 shows the spring graph.

```
clear % clear removes all variables from the workspace.
clc  % clean screen
```

```
% The linspace function generates linearly spaced vectors. It is similar to the colon
% operator ":", but gives direct control over the number of points.  y = linspace(a,b,n)
% generates a row vector y of n points linearly spaced between and including a and b.

x=linspace(0,10*pi,1000);
y=cos(x); z=sin(x); plot3(x,y,z)
% comet3(x,y,z) - Generates an animated version of plot3
grid
xlabel('angle'), ylabel('cos(x)'), zlabel('sin(x)')
title(' Spring Demo')
```

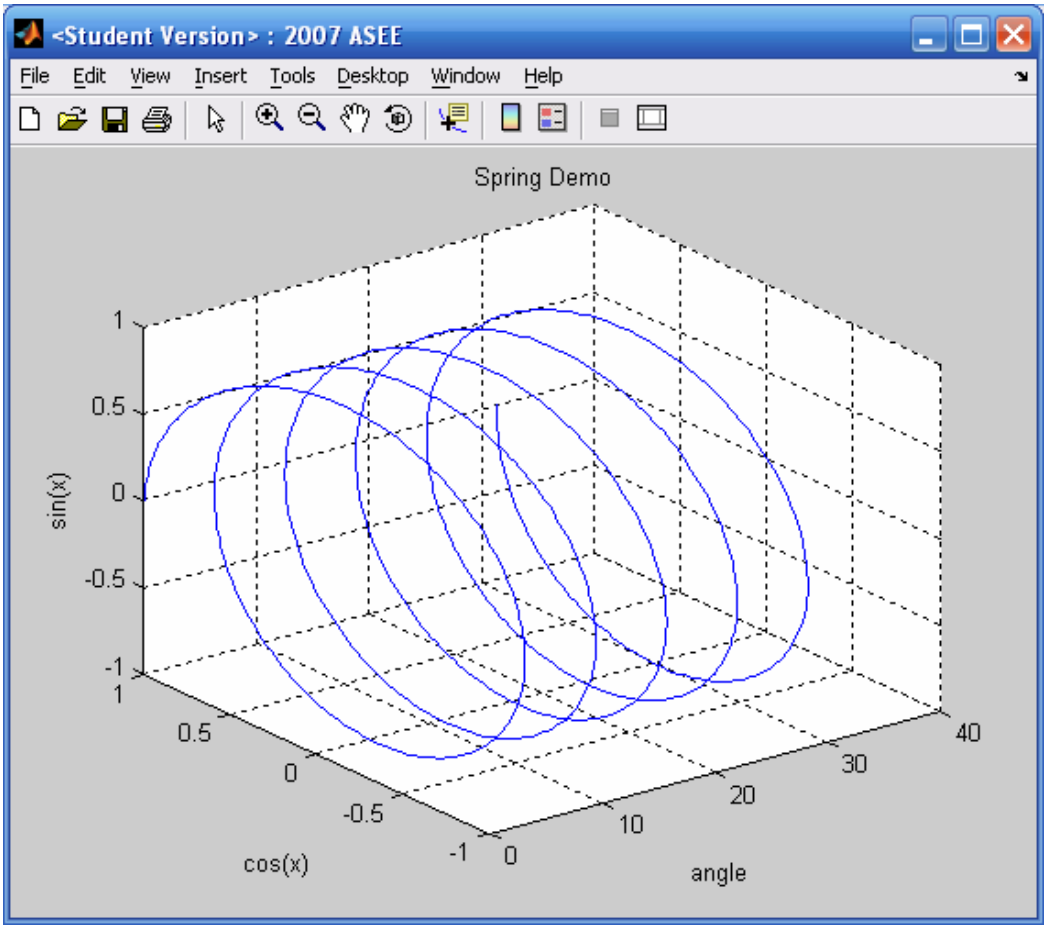

Figure 3

User Defined Functions: MatLab user defined functions are demonstrated next. Three user defined functions are provided to explore the easy to write feature of user defined functions. It is expected that the reader easily able to write his/her own functions with minimal effort.

```
function returnValue = square(inputArgument)
%
%  This function calculates the square of the given number 'inputArgument'
%
%  inputArgument    (input)   value to be squared
%  returnValue      (output)  the square result

  returnValue = inputArgument^2;

% end of the function

function returnValue = power(X,N)
%
%  This function calculates the X^N
%
%
%  Input arguments  X,N
%
%  returnValue      (output)  the power result

  returnValue = X^N;

% end of the function
```

```
EDU>> power(5,3)

ans =
   125
```

```
function display_utility(aString, aCount)
%
% This utlity function displays a string aCount times
%
% Input Arguments:
%
%    aString          (input)  to print the value of a variable
%
%    aCount           (input)  use aCount for loop counter
%
%
  for i=1:aCount
```

```
  fprintf(1, '%s\n', aString);
end
```

```
EDU>> display_utility('2007 ASEE',4)
2007 ASEE
2007 ASEE
2007 ASEE
2007 ASEE
```