AC 2007-3055: TEACHING OF ESSENTIAL MATLAB COMMANDS IN APPLIED MATHEMATICS COURSE FOR ENGINEERING TECHNOLOGY

Ganapathy Narayanan, University of Toledo

Teaching of Essential MATLAB Commands in Applied Mathematics Course for Engineering Technology

Abstract

The teaching of applied mathematics for students in the Engineering Technology (ET) curriculum is always a challenge in terms of imparting the essential mathematical knowledge for use in changing technological environments. In this paper, essential MATLAB commands in the applied mathematics course for ET students are emphasized. Of several useful commands available in MATLAB, only a select few, important and essential, commands are introduced to enhance the understanding of the applied mathematical subject. In addition, this introduction of essential MATLAB commands provides useful knowledge for ET students of the current and the future technological approach to solve similar applied mathematical problems. The MATLAB commands that are taught in the applied mathematics course are identified in this paper. In addition, this paper gives notes on the use of each of these MATLAB commands through several mathematical examples. The effects on the student's understanding of and on the enthusiasm developed with the subject by such a teaching of these essential MATLAB commands are also discussed in this paper.

Introduction

In this paper, the subject of MATLAB commands pertaining to the Applied Mathematics course is discussed. Several textbooks exist for teaching Applied Mathematics[1] and the use of MATLAB[2-5]. The teaching of the use of a select few MATLAB commands in conjunction with the subject matter of the Applied Mathematics is undertaken in the 'Applied Engineering Mathematics' course offered to the Engineering Technology (ET) students at the University of Toledo, Toledo, Ohio. The discussion is limited to a select few MATLAB commands as applicable to this applied mathematics course, and as to how these commands help in imparting the subject of applied mathematics for ET students.

Several difficulties exist, both for undergraduate ET students in terms of understanding the applied mathematics subject, and for the instructor in imparting the applied mathematical knowledge required, to make students understand mathematical concepts and applied differential equations with its solutions, within the semester time-frame. The 30 class hours (one semester) are not sufficient for the instructor to make ET students understand and digest the subject matter in order to make them proficient in applied mathematical analysis. Such a difficulty due to time shortage cannot be avoided even if the course teaching hours are extended. This course is both symbolic-intensive and numerical-intensive in terms of the applied mathematical analysis. If symbolic tools and other computational aids are not made available, all students get bogged down usually with many intermediate steps while solving for even the simplest ordinary differential equation, and/or while producing appropriate solution plots manually that are of interest. The MATLAB software, with its applied mathematical commands in its tool-box, rescues the instructor and students in this course with many advantages for both, especially for the ET student. Of course, some special virtual laboratory time need to be spent by students to get trained in the use of these select MATLAB applied mathematics commands.

Thus, the ET student is trained in solving any applied mathematical problem without getting distracted by many intermediate symbolic steps and/or by numerical calculations, and subsequently, the student's comprehension of the subject material is much better. The student can then concentrate efforts in the overall design/analysis of an applied problem/system, leaving symbolic/computational crunching to the virtual computer program (MATLAB). This increases the student's confidence and understanding of the applied mathematical subject, as is pertinent in engineering. In addition, the benefits to the ET student after graduation are quite a lot from this course in the sense that the knowledge of the use of MATLAB (software) as an assistant will help the ET student in applied analysis, design and other areas of study, without extra efforts or training, and especially, in his professional career.

There have been many successful attempts in several universities to teach the mathematical subject using the symbolic software such as Maple or Mathematica. Of course, several texts do exist in applied engineering mathematics with a parallel work shown using Maple[6]. In this paper, a comparison is not done of all other software commands associated with these MATLAB commands.

Select MATLAB Commands

The select few MATLAB commands taught in this applied mathematics course are classified below according to eight important subject areas, for convenience of teaching. The eight areas are chosen specifically to increase the students' ability to remember MATLAB commands, and to minimize any teaching difficulties of the applied mathematics subject in certain areas.

The eight categories of select MATLAB commands are:

a) Function Commands;
b) Plot Commands;
c) Symbolic Differentiation Command;
d) Symbolic Integration Command and its applications;
e) Taylor Series Command;
f) Laplace Transform Commands.
g) Ordinary Differential Equation Solver Command;
h) Partial Differential Equation Solver Command.

In this paper, only a few MATLAB output figures are shown. Other similar output figures can be obtained easily by the reader or by the student, by typing in the MATLAB commands and its examples on a computer.

Function Commands

The following basic types of functions are suggested to be studied, in detail, using MATLAB in this applied mathematics course:

1. Exponential Function, ae^{bx}
2. Logarithmic Function, $a\ln(bx)$, $a\log_{10}(bx)$
3. Trigonometric functions, $a\sin^n(bx+\theta)$, $a\cos^n(bx+\theta)$ and $a\tan^n(bx+\theta)$

4. Power Functions, ax^n
5. Polynomial Functions, $\sum_n A_n x^n$

First, it is suggested in the virtual lab to understand the five 'simple' functions obtained by using $a=1, b=1, \theta=0, n=1$ & $A_n=1$ for all n. Then, the values of constants 'a' and 'b' are changed in each of the five basic functions to see how the simple functions changes in shape. In general, it is known that an engineering technology student will use these functions 80% of time in his professional career. So, his understanding of these functions and its x-y plots will help clarify his understanding of derivatives and integrals of these functions.

The input commands for computing the five basic functional values for given values of 'a' and 'b' at 'x' are seen in MATLAB Command Window, as shown in Figure 1. The MATLAB functional (f1, f2, f31, f32, f33, f4, f5) values output are shown in Figure 2. One can change the values of parameters 'a', 'b' and 'n' in these functions as well as for the variable 'x' value.

```
Command Window
>> %Figure 1 - Basic Functional Values
>> % MATLAB commands for computing Basic Functional Values at x
>> % Assume a=1; and b=2; for all functions below
>> a=1; b=2; n=1; x=5;
>> f1=a*exp(b*x); %For Exponential Function say f(x)=a*exp(b*x)
>> f2=a*log(b*x); %For Logarithmic Function say f(x)=a*ln(b*x)
>> % Use 'log' command for base 'e' and 'log10' for base '10'
>> x=pi/4;n=1;f31=a*(sin(b*x))^n; %For Trigonometric 'Sine' Function
>> x=pi/4;n=1;f32=a*(cos(b*x))^n; %For Trigonometric 'Cos' Function
>> x=pi/4;n=1;f33=a*(tan(b*x))^n; %For Trigonometric 'Tan' Function
>> f4=a*x^n; %For Power Function say f4(x)=a*x^n
>> a0=1;a1=-2;a2=3; f5=a0+a1*x+a2*x^2; %For Polynomial Function
>> % Here, this is quadratic polynomial example
>> % Output function values for f1,f2, f31,f32,f33, f4,f5 are
>> f1, f2, f31, f32, f33, f4, f5
```

Figure 1: MATLAB input commands for computing Basic Functional Values

Plot Commands

In this course, MATLAB 2D plot commands are emphasized. These plot commands are, namely, 'plot', 'semilogx' and 'semilogy'. The 'plot' command produces the graphic plot of tabular vector-valued variables x and y, with x-values along the 'linear' x-axis and y-values along the 'linear' y-axis. One can define these x-variable and y-variable vectors either directly from the ordered-pair values (table) or from computed y-functional values for the corresponding x-values. The use of 'plot' command will be illustrated in this paper. The 'semilogx' and 'semilogy' commands usage is similar to the 'plot' command. The 'semilox' command produces a plot with x-values along x-log axis. The 'semilogy' command creates a plot with y-values along the y-log axis. Figures 3 & 4 give the 'plot' command usage to generate an x-y plot. The 'plot' command needs a minimum of two input arguments, namely x- and y-variables with vector values in them. Only one function plot is generated here for illustration. In the class, virtual lab exercises to plot all the five basic functions for various values of 'a' and 'b' are given.

```
Command Window
>> % Figure 2 - MATLAB Output
>> % MATLAB Basic Functional Values Output
>> f1, f2, f31, f4, f5

f1 =

  2.2026e+004


f2 =

    2.3026


f31 =

     1


f4 =

    0.7854


f5 =

    1.2798
```

Figure 2: MATLAB Basic Functional Values Output

```
Command Window
>> % Figure 3-A: Table Plot Example
>> % Vector Variable x with 7 values,
>> % Vector Variable y with 7 values,
>> % space used to delimit values
>> x = [1 2 3 5 7 8 10];
>> y = [2 6.5 7 7 5.5 4 8];
>> % plot x vs y with x-vector along x-axis
>> % and y-vector values along y-axis
>> plot(x,y,'-o')
>> % Insert x-axis, y-axis labels and Plot Title
>> xlabel('x-values')
>> ylabel('y-values')
>> title('Figure 3-A: Table Plot Example')
>> grid on
>>
```

Figure 3-A: MATLAB Table Plot Example

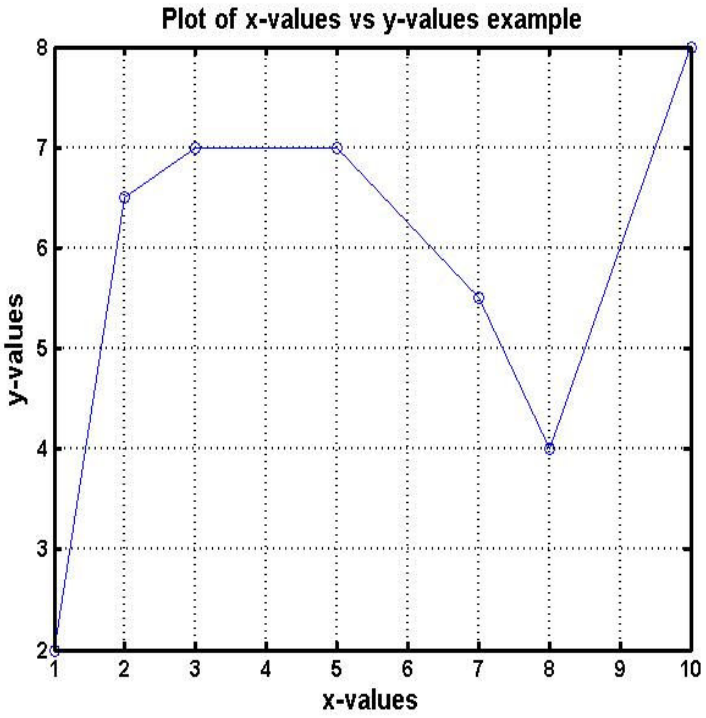

Figure 3-B: MATLAB Plot with two discrete valued vectors (Table Plot)

Command Window

```
>> % Figure 4-A: Functional Plot Example
>> % Example Plot of y = f(x),
>> % say y(x) = 4*exp(-0.2*x)*cos(4*x)
>> % Define x-vector variable in defined domain
>> x = [-2:0.01:4];
>> % Compute y-vector variable
>> y = 4.0*exp(-0.2*x).*cos(4*x);
>> plot(x,y,'-o') % Plot x vs y
>> % Label axes and plot title
>> xlabel('x-values')
>> ylabel('y-values')
>> title('Figure 4-a: Functional Plot Example')
>> grid on
>>
```

Figure 4-A: MATLAB Functional Plot Example

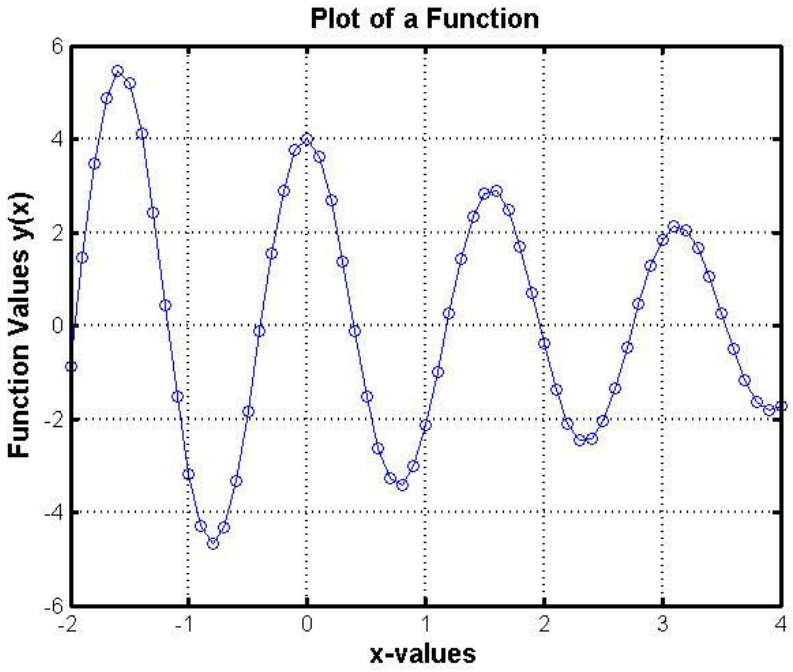

Figure 4-B: MATLAB Plot of a Function

Symbolic Differentiation command

The MATLAB command for the symbolic differentiation is ‘diff’. This is a very powerful command that is useful for taking derivatives of basic and composite functions of one or more independent variables. The 'diff' command requires a minimum of two input arguments, namely, the function to be differentiated as the first argument, and the variable in the function used for differentiation. The MATLAB software does allow nesting of one 'diff' command inside of the other 'diff' command. Mostly, in this applied course, only two independent variables are considered. If double differentiation is needed of the same variable, then a third argument with a value of '2' can be given. Figure 5 shows sample MATLAB example of taking symbolic derivatives of functions of two variables. Of course, the derivatives of functions are also functions of one or two variables, and they can be plotted, as explained above.

Symbolic Integration Command and its applications

The MATLAB command for the symbolic integration is ‘int’. This is also a very powerful command that is useful in performing integration of basic and composite functions with respect to one or two independent variables. The indefinite integral of 'int' command needs only two input arguments, namely, the function to be integrated as the first argument, and the variable in the function used for integration as the second. If the lower and upper limits of integration are

```
Command Window
>> % Figure 5-A: MATLAB 'diff' command usage
>> % Example Function: f(x,y) = (x^2)*y+(y^2)*x
>> % Define symbolic variables
>> syms x y
>> % Determine derivative of f(x,y) wrt x
>> diff((x^2)*y+(y^2)*x,x)

ans =

2*x*y+y^2


>> % Determine derivative of f(x,y) wrt y
>> diff((x^2)*y+(y^2)*x,y)

ans =

x^2+2*x*y
```

Figure 5-A: Use of MATLAB diff command

```
Command Window
>> % Figure 5-B: MATLAB 'diff' command usage
>> % Example Function: f(x,y) = (x^2)*y+(y^2)*x
>> % Define symbolic variables
>> syms x y
>> % Determine derivative of f(x,y) wrt x, twice
>> diff((x^2)*y+(y^2)*x,x,2)

ans =

2*y


>> % Determine derivative of f(x,y) wrt y, twice
>> diff((x^2)*y+(y^2)*x,y,2)

ans =

2*x
```

Figure 5-B: Use of MATLAB diff command

known, they are specified as the third and the fourth arguments, respectively. The MATLAB software allows the nesting of one 'int' command inside of the other 'int' command. This nesting of 'int' commands makes it easy to compute multiple integrals of a function. Figure 6 shows a sample MATLAB application by using the 'int' command in a nested setup to determining the center of area coordinates and the moment of inertia of areas. The example used in Figure 6 determines the centroid and moment of inertia of area formed by two intersecting curves.

```
Command Window
>> % Figure 6-A: Double Integration Example
>> % This is Textbook (Ref 1) Example on Page 1316
>> % Problem: Find Centroid (xcg,ycg) of the region
>> % bounded by lines 'y = x' and 'y = 6-2x', and
>> % the y-axis
>> % Define Symbolic variables
>> syms x y
>> % Find Area of the region
>> area = int( int(1,y,x,(6-2*x)),x,0,2);
>> % Find Static Moment Mx and My
>> Mx   = int( int(y,y,x,(6-2*x)),x,0,2);
>> My   = int( int(x,y,x,(6-2*x)),x,0,2);
>> % Determine Centroid of area
>> xcg = My / area

xcg =

2/3


>> ycg = Mx /area

ycg =

8/3
```

Figure 6-A: Centroid of a Region Example using MATLAB

Taylor Series Command

The MATLAB command for performing series expansion of a function is 'taylor'. This command does both Maclaurin and Taylor series expansion of functions. Figure 7 shows both Maclaurin and Taylor series expansion examples of sample functions using 'taylor' command.

Laplace Transform Commands

The mathematical use of 'Laplace Transform' on the differential equation of motion, as developed for an engineering problem, is discussed thoroughly in many mathematics and engineering texts[6,7], and hence, this mathematical theory is discussed in class, but not discussed in this paper. However, the use of MATLAB 'Laplace Transform' command is discussed in this paper.

The MATLAB ‘laplace’ command can transform a given time function into its corresponding Laplace transform function. Similarly, the ‘ilaplace’ command can determine the inverse Laplace transform of many useful functions defined in the Laplace domain. The use of ‘laplace’ and ‘ilaplace’ commands is shown in Figure 8. Both these commands require three input arguments. The 'laplace' command requires the function as the first argument with 't' and 's' as the second and third arguments. The 'ilaplace' command requires the transformed function as the first argument with 's' and 't' as the second and third arguments.

```
Command Window
>> % Figure 6-B: Moment of Inertia Example
>> % This is Textbook (Ref 1) Example on Page 1319
>> % Find Moment of Inertia (Ixx, Iyy) and
>> % Radii of Gyration (rx, ry) for a region
>> % bounded by line 'y = 2x' and and 'y = x^2'
>> syms x y
>> % Determine area of region
>> area = int( int(1,y,0,x^(1/3)),x,0,8);
>> % Compute Ixx and Iyy
>> Ixx = int( int(y^2,y,0,x^(1/3)),x,0,8);
>> Iyy = int( int(x^2,y,0,x^(1/3)),x,0,8);
>> % Compute rx and ry
>> rx = sqrt(Ixx/area);
>> ry = sqrt(Iyy/area);
>> Ixx , Iyy , rx , ry
```

Figure 6-B: Moment of Inertia Example using MATLAB

Ordinary Differential Equation Solver Command;

In MATLAB, it is possible to solve the ordinary differential equation (ODE) of any system numerically for the system variable(s) in the time or the space domain. However, the solver command 'dsolve' can be used to obtain 'symbolic' solutions of many ODE systems. Both 'direct' and 'indirect' solutions of 'time' approaches of solving ODE are discussed in class. Of course, deriving the 'direct' symbolic 'time' or 'space' solution of first and second order ODE is discussed in detail, as mandated by the syllabus. Such mathematical details of solution are in texts[1,6,7].

MATLAB has several 'direct' ODE commands[8] to solve for the system variable time or space function, in a numerical form, with an associated numerical solution plot. Though these commands are very useful for solving ODE numerically, such commands are not discussed in

this class. The numerical solutions of ODE are not part of the syllabus for this applied math course, and hence, the numerical solution commands of ODE are not discussed in this paper.

Command Window

```
>> % Figure 7-A: Use of MATLAB 'taylor' command
>> % Maclaurin Series Expansion Exaxmple
>> % Function '1/(5 + 4*cos(x))' used
>> syms x
>> f=1/(5+4*cos(x));
>> taylor(f,8)

ans =

1/9+2/81*x^2+5/1458*x^4+49/131220*x^6


>> % Note this series includes terms upto order 8
>> % but does not include eighth order
>>
```

Figure 7-A: Use of MATLAB taylor command

Command Window

```
>> % Figure 7-B: Use of MATLAB 'taylor' command
>> % Taylor Series Expansion Example
>> % Function 'exp(x)' used
>> % Taylor expansion about x = 1
>> syms x
>> g = exp(x);
>> % three input arguments: func, order #, (x=)1
>> taylor(g, 10, 1);
>>
```

Figure 7-B: Use of MATLAB taylor command

```
Command Window
>> % Figure 8-A: Laplace Tranform using MATLAB
>> % To determine Laplace Transform of Function
>> % Function f(t) = sin(a*t) used
>> % Define symbolic variables, t, s, & a
>> syms t s a
>> % Define Function f(t)
>> f = sin(a*t);
>> % Use 'laplace' command
>> laplace(f,t,s)

ans =

a/(s^2+a^2)
```

Figure 8-A: Use of MATLAB laplace command

```
Command Window
>> % Figure 8-B: Inverse Laplace Tranform using MATLAB
>> % To determine Inverse Laplace Transform of Function
>> % Function F(s) = a/(s^2 + a^2) used
>> % Define symbolic variables, t, s, & a
>> syms t s a
>> % Define Transform Function F(s)
>> F = a/(s^2 + a^2);
>> % Use 'ilaplace' command
>> ilaplace(F,s,t)

ans =

sin(a*t)
```

Figure 8-B: Use of MATLAB ilaplace command

However, the use of MATLAB ‘dsolve’ command for obtaining the symbolic solution is discussed in this paper. The ‘dsolve’ command can be used on any ODE system to obtain for the system 'time' or 'space' solution function in a symbolic (analytic) form. Such analytic function(s) can be plotted by the use of MATLAB ‘plot’ command. Figure 9 illustrates the use of ‘dsolve’ command to obtain the time or space solution in a symbolic (analytic) form.

On the other hand, the Laplace or Fourier Transform approach is generally used to obtain the 'indirect' solution using the transform domain solution. The Fourier Transform approach is not elaborated in this course, except as a passing mention that Fourier Transform is a special case of

Command Window

```
>> % Figure 9-A: Use of MATLAB 'dsolve' command
>> % To get general solution of second order
>> % Homogeneous ODE (Initial Conditions not used)
>> % ODE 'D2y + 10*Dy + 100*y = 0' used
>> % use 'dsolve' with two input arguments only
>> dsolve('D2y + 10*Dy + 100*y = 0')

ans =

C1*exp(-5*t)*sin(5*3^(1/2)*t)+C2*exp(-5*t)*cos(5*3^(1/2)*t)


>> % Note: C1 & C2 are constants of integration
>>
```

Command Window

```
>> % Figure 9-B: Use of MATLAB 'dsolve' command
>> % To get solution of second order
>> % Homogeneous ODE (Initial Conditions used)
>> % ODE 'D2y + 10*Dy + 100*y = 0' used
>> % Initial Conditions are: at time t=0, y(0)=1 & Dy(0)=0
>> % use 'dsolve' with three input arguments
>> dsolve('D2y + 10*Dy + 100*y = 0','y(0)=1','Dy(0)=0')

ans =

1/3*3^(1/2)*exp(-5*t)*sin(5*3^(1/2)*t)+exp(-5*t)*cos(5*3^(1/2)*t)


>> % Note: C1 & C2 are determined from Initial Conditions
>>
```

Command Window

```
>> % Figure 9-C: Use of MATLAB 'dsolve' command
>> % To get Total solution of second order
>> % Non-Homogeneous ODE (Initial Conditions used)
>> % Forcing function (RHS): f(t)=3sin(2t) used
>> % ODE 'D2y + 10*Dy + 100*y = 0' used
>> % Initial Conditions are: at time t=0, y(0)=1 & Dy(0)=0
>> % use 'dsolve' with three input arguments
>> dsolve('D2y + 10*Dy + 100*y = 3*sin(2*t)','y(0)=1','Dy(0)=0');
>> % Note: C1 & C2 are determined from Initial Conditions
>> % Also, Total solution contains particular solution for f(t)
>> % remove ';' on dsolve command to see solution output
>>
```

Figure 9: Use of MATLAB dsolve command

the Laplace Transform. The Laplace Transform approach of obtaining the solution for the first and second order ODE is discussed in detail in the class.

Partial Differential Equation (PDE) Solver Command

This is a more advanced topic for the undergraduate engineering technology student, and hence, it is not discussed in the course. However, this topic is left as an open topic for an advanced or graduate student who can pursue this PDE subject when he needs it, either as an independent study or take another course on PDE, with an associate training in the use MATLAB for solving PDEs. There are many senior undergraduate and graduate courses offered at UT for the student to advance his training in this direction. In any case, the commands that he learned in this applied mathematics course will help him with his advanced training for solving partial differential equation using the 'dsolve' or 'pde' commands, with ease.

MATLAB Resources: Additional MATLAB User Application resources are available at www.mathworks.com/matlabcentral .

Classroom Use of select MATLAB commands

The above select MATLAB commands have been offered in the MET Applied Mathematics course as a set of virtual laboratory exercises. There are about three computer laboratory sessions used to present these select MATLAB commands, and the class room materials are presented according to schedule such that the virtual laboratory work is integrated for student experience of these subject materials.

The testing of understanding of the applied mathematics materials with the introduction of the MATLAB laboratory exercises showed an increased interest and improvement among students' understanding of the subject materials. There is definitely a success in the use of above materials in the applied mathematics course. But a rigorous research was not done to assess the percent of success in the applied mathematics course, with the introduction of above select MATLAB commands virtual laboratory exercises. A comparative analysis of grades before and after using this approach would give some perspective of the "goodness" of the technique, in quantitative terms. Also, the impact of this approach of using select MATLAB commands on the "deep" learning of ET students has not been studied.

Conclusions

The paper has been confined to the use of select MATLAB commands in the applied mathematics course useful for the ET student. The MATLAB help facility can provide additional information on the use of these individual commands. The paper has concentrated on the application of these select commands as is taught in the applied mathematics course for ET student. The elimination of the distraction of intermediate symbolic steps in the subject helps all ET students concentrate and understand the important mathematical concepts taught in the course. An additional advantage of learning the MATLAB control commands is with regard to his increased ability to analyze/design any engineering system in his professional career, without getting bogged down with many symbolic or numerical calculations. Of course, the use of

MATLAB commands did boost his ability to perform in obtaining symbolic solutions and/or other calculations needed in other areas of his engineering study.

References

1) Peterson, J.C., "Technical Mathematics with Calculus, 3rd Edition", Thomson Delmar Learning, 2004
2) Gilat, Amos, "MATLAB – An Introduction with Applications", Second Edition, John Wiley, 2005.
3) Chapman, S. J., "MATLAB Programming for Engineers", Second Edition, Brooks/Cole, 2002
4) Etter, D. M., Kuncicky, D. C., and Hull, D., "Introduction to Matlab 6", Prentice Hall Engineering Source, 2002
5) Kuncicky, D. C., "MATLAB Programming", Prentice Hall Engineering Source, 2004
6) Lopez, R.J., "Advanced Engineering Mathematics", Addison-Wesley, 2001.
7) Kreyszig, E.,"Advanced Engineering Mathematics- 9th Edition", John Wiley, 2006
8) Polking, J.C., Arnold, D.,"Ordinary Differential Equations Using MATLAB", Prentice Hall, 2004
9) Farlow, S.J.,"Partial Differential Equations for Scientists and Engineers", Dover, 1993

Dr. G.V. Narayanan teaches at the University of Toledo, Toledo, Ohio. He can be contacted by email at 'nara@utoledo.edu'.