An Introduction to Maple™

Daniel S. Stutts, Ph.D.

February 26, 2001

1 Introduction

Maple™ is a *computer algebra system* developed at the University of Waterloo, Ontario, Canada, and has subsequently been commercialized and widely available since 1990. Along with its primary competitors, Mathematica, and Macsyma, Maple has evolved greatly as compared to the first computer algebra systems developed in the 1970s. In essence, a computer algebra system performs analytical calculations automatically. All of the commercially available systems also perform a wide variety of numerical computations as well, and each has its own programming language so that very complex algorithms may be implemented. Hence, these systems are much more than just an analytical calculator, although they work perfectly well in that limited capacity. The following is a very brief description of some of Maple's capabilities. An online tutorial is available at:

2 Basic Commands and Syntax

Maple is available on almost all computer platforms, and may be found on most of the PC's in UMR's Computer Learning Centers (CLCs). It is also available on the UNIX system, where it is invoked at the command prompt by entering *xmaple*. There are two principal versions of Maple avaliable: (1) the interactive version, and (2) the command-line version. The latter is a holdover from the days before windowing operating systems, but has the advantage of requiring less computer resources, so it runs in less memory, and also executes commands faster.

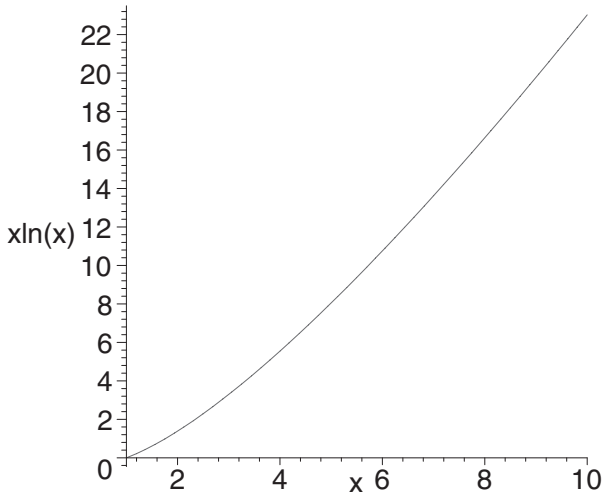
A session on the interactive Maple may be saved in a *Maple worksheet*. The worksheet may be a whole Maple program, and may be executed later on either line by line, or by using the *Edit* menu, and selecting the *Execute Worksheet* command. On the interactive Maple, the only one discussed here, every line is an input line, and commands are executed simply by pressing *Enter* or *Return*. All commands must be terminated by the semicolon (;) or the colon (:), except for the beginning statement in a control loop, such as a *for* or *while* statement. Another exception is when the question mark (?) is used to invoke *help* on a specific command. If one wanted help on the integration command *int*, one would type at the prompt:

```
>?int
```

Comments in Maple may be entered in two ways: (1) by selecting the *Text* button, or (2) by beginning the input line with the pound (#) sign. Terminating an input line with the colon (:) suppresses the default echo of the output of the executed function. The colon is used when output is not needed. The assignment operator in Maple is colon followed by the equal sign (:=). This is not to be confused with the logical comparison operator (=). A complete list of operators and their precedence is given by entering ?operator. Maple is best understood by interactively by example, so several examples from a Maple session will follow.

```
Here is an example of some text that may be used to comment a Maple work sheet.
> #Here is another way to comment using the # sign
[Next, let's do some actual math!
> f:=x*ln(x);

                              f := x ln(x)

[Now let's integrate f:
> int(f,x);
```

```
                              2              2
                         1/2 x  ln(x) - 1/4 x
[Let's take the derivative of f:
> diff(f,x);

                                ln(x) + 1

[Let's plot f:
plot(f,x=1..10,labels=[`x`,`xln(x)`]);
```



It is also possible to send the plot to a file of various types such as encapsulated postscript (.eps) and JPEG (.jpg). The plot shown above was saved as an eps file with several options by issuing the plotsetup command as follows:

```
plotsetup(ps,plotoutput=`C:\Classes\MaplePrimer\xlnx.eps`,plotoptions=
`leftmargin=1in,width=3in,height=3in,portrait,noborder`);
plot(f,x=1..10,labels=[`x`,`xln(x)`]);
> plotsetup(default);
```

In the above, the plotsetup was returned to the default (to the screen) setting after the plot was saved – otherwise, all other plots would go to the same file. Unfortunately, as of Maple V, version 5.1, EPS files generated by Maple often have errors in them, and, as in the above case, must be taken into a graphics program such as Adobe Illustrator and re-saved. In this example, the EPS file generated by Maple initially showed up in this LaTeXdocument all the way over on the right margin. The solution was to open it up in Illustrator, copy it, paste it into a new Illustrator document, and save it as an Illustrator EPS file.

In the interest of space, the output of the Maple commands demonstrated in this document will be suppressed except where needed. Plotting in Maple is a rather large subject, requiring much more explanation than will be given here, but the help system in Maple is full of examples, and with some trial and error, it is usually possible to obtain fairly nice plots.

3 Calculus in Maple

Maple has a very rich set of mathematical functions and operations, and only a few basic ones will be addressed here. You have already seen examples of integration and differentiation, and these operations will be examined in more detail here.

3.1 Differentiation and Integration

There are several forms of differentiation, and three main forms of integration. Differentiation may be done using the *diff* function, which is the functional form of differentiation as illustrated above. It may also be done using an operational form using the *D* function. Both integration and differentiation also have inert forms which produce an unevaluated operation except when forcing numerical integration. The purpose of the inert form, which is output in Maple *pretty print* format, to produce nicely formatted equations which are easier to read, and may be exported, to LaTeXfor example, as will be demonstrated later. The inert forms use a capital first letter: *Diff*, and *Int* instead of *diff* and *int*. Some examples follow:

```
diff(sin(x),x);
                                  cos(x)

> Diff(sin(x),x);

                                 d
                                 -- sin(x)
                                 dx

> D(sin)(x);

                                  cos(x)

> D(sin@x);

                                 cos@x D(x)

> D(sin)@x;

                                  cos@x

> D(sin);

                                   cos

> Int(sin(x),x=0..Pi/2);

                              1/2 Pi
                             /
                            |
                            |        sin(x) dx
                            |
                           /
                            0

> evalf(Int(sin(x),x=0..Pi/2,3));

                                   1.00
```

Note that the asci character integral shown above is beautifully formated in the Maple worksheet, but when copied and pasted into a text editor, such as was used to create the LaTeXsource file for this document, looks exactly like the above. This is the same style of output produced by the command-line version of Maple. You should execute all of the commands discussed here to see how they actually appear in Maple. Note the last integration was forced to evaluate numerically using the *evalf* function. The name stands for: "evaluate as floating point," and the *Int* function optionally takes additional arguments to specify the number of digits to output (three in this example), and

a specific integration scheme to use – this example used the default numerical scheme. The @ sign is the composition operator in Maple. The @@ operator is the multiple composition operator. For example:

```
>h:=(x)->sin((sin@@4)(x));
```

$$x \mapsto \sin(\left(\sin^{(4)}\right)(x))$$

```
>h(0);
                                    0
evalf(h(Pi/2));
                                .6275718321
```

In the above example, the abbreviated form of a procedure was used. Here, the "maps to" notation was borrowed from mathematics: $(x) -> f(x)$. Literally, x maps to $f(x)$.

3.2 Example of Fourier Expansion

In this section, the use of the *sum()* function will be demonstrated through the use of a procedure. Maple also has *Sum()* which is the inert form of the sum function which simply typesets the output.

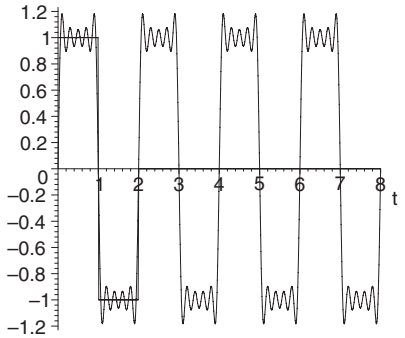
```
s:=Sum((-1)^n*x^n/n!,n=0..infinity);
```

$$s := \sum_{n=0}^{\infty} \frac{(-1)^n x^n}{n!}$$

Below is my procedure to demonstrate fourier expansion of a function. The key features of this procedure are that it demonstrates the sum() function, and it allows "extra" plot arguments to change the appearence of the plot. A better version of this procedure would allow the function and Fourier expansion of the function to be plotted in different colors. This is no big deal, but I will leave it up to the reader to figure out how to do it!

```
# Procedure: fourierplot3
>   #
>   # Author: Dr. Daniel S. Stutts
>   # Associate Professor of Mechanical Engineering and
>   # Engineering Mechanics, University of Missouri-Rolla.
>   #
>   # Function: Plots the function, 'func', and its n-term Fourier expansion.
>   #
>   # func = function to be plotted
>   # xrange = Period of function -- a..b
>   # plotrange = range of Fourier expansion to plot -- a..b
>   # Given f(x), both xrange and plotrange are entered as x=a..b,
>   # where a and b are the respective endpoints.
>   # n = number of fourier terms to be used in expansion
>   #
>   # Features: allows additional plot arguments -- arguments 5 through nargs.
>   #
>   # Based on example in Monagan, et al. "Maple V Programming Guide," 1996, Springer,
```

```
>   # pages 318-319.
>   # Modified to allow plotting of the periodic Fourier extension 1/18/00 by DSS.
>   # Modified to use the sum() function instead of a do loop 2/25/01 by DSS.
>
>   fourierplot3:=proc(func,xrange::name=range, plotrange::name=range,n::posint)local x, a, b, l, i, pa
>    a:=lhs(rhs(xrange));
>    b:=rhs(rhs(xrange));
>    l:=b-a;
>    x:=2*Pi*lhs(xrange)/l;
>    partsum:=1/l*int(func,xrange);
>
>    fsum:=partsum+sum(2/l*int(func*sin(i*x),xrange)*sin(i*x)
>    +2/l*int(func*cos(i*x),xrange)*cos(i*x),i=1..n);
>
>    plot({func,fsum},plotrange,color=black,args[5..nargs]);
>    end;
>     Warning, `fsum` is implicitly declared local

  fourierplot3 := proc(func, xrange::(name = range),

plotrange::(name = range), n::posint)
local x, a, b, l, i, partsum, fsum;
    a := lhs(rhs(xrange));
    b := rhs(rhs(xrange));
    l := b - a;
    x := 2*Pi*lhs(xrange)/l;
    partsum := int(func, xrange)/l;
    fsum := partsum + sum(
        2*int(func*sin(i*x), xrange)*sin(i*x)/l
         + 2*int(func*cos(i*x), xrange)*cos(i*x)/l,
        i = 1 .. n);
    plot({func, fsum}, plotrange, color = black,
        args[5 .. nargs])
end

> fourierplot3(F(t),t=0..2,t=0..8,9,numpoints=500);
```



4 Linear Algebra in Maple

When working with an array or matrices, the *Map* command must be used, and the linear algebra library must be loaded. Instead of loading the whole *linalg* library by executing the *with(linalg)* command, individual linear algebra

functions may be executed by prepending the linalg keyword as shown in the following example:

```
>A:=linalg[matrix](2,2,[sin(x), x^2+x+3, exp(x), cos(x^2)]);
```

The output looks like the following:

$$A := \begin{bmatrix} \sin(x) & x^2+x+3 \\ e^x & \cos(x^2) \end{bmatrix}$$

Lets say we want to take the derivative of A. Then we must use the *Map* command as follows:

```
>B:=map(diff, A, x);
```

The output looks like:

$$B := \begin{bmatrix} \cos(x) & 2\,x+1 \\ e^x & -2\,\sin(x^2)x \end{bmatrix}$$

Now lets say we want to multiply A and B. We must use the matrix multiply operator &$*$, and force the evaluation as a matrix by using $evalm()$ as follows:

```
>C:=evalm(A&*B);
```

The output looks like:

$$C := \begin{bmatrix} \sin(x)\cos(x) + \left(x^2+x+3\right)e^x & \sin(x)\left(2\,x+1\right) - 2\,\left(x^2+x+3\right)\sin(x^2)x \\ e^x\cos(x) + \cos(x^2)e^x & e^x\left(2\,x+1\right) - 2\,\cos(x^2)\sin(x^2)x \end{bmatrix}$$

Now, lets take the determinant of C, and create a function using the *unapply*() command:

```
>detr:=linalg[det](evalm(C)):
>DET:=evalf(unapply(detr,x)):
```

In the above, the $evalf()$ command is an attempt to force Maple to simplify the output to a floating point number. Actually, Maple will evaluate the result as a floating point number if the argument is a floating point number, but leave the result unevaluated if the argument is an integer. Lets test this:

```
>evalf(DET(-1));
                              -1.990208157
>DET(-1);
```

$$-2\,\left(\sin(1)\right)^2\left(\cos(1)\right)^2 - 3\,\left(e^{-1}\right)^2 - 7\,\sin(1)\cos(1)e^{-1}$$

```
>DET(-1.);
                              -1.990208157
```

Incidentally, another handy function for those of you who use LaTeXis the *latex()* command which was used to create all of the typeset equations in this document. The *latex()* function belongs to a family of functions which convert the output of Maple commands. The other two commands in this group are the *fortran*, and the C command which output FORTRAN and C commands respectively. Note that the C command would not work in this example because we have already defined an expression named C! Hence, care must be taken not to redefine Maple's function especially if you plan to use them. Maple will allow such overloading of definitions without complaining.

Here are some examples using the *latex()* and *fortran()* commands. Note that the *fortran()* command takes optional arguments.

```
> latex(C);
\left [\begin {array}{cc} \sin(x)\cos(x)+\left ({x}^{2}+x+3\right ){e^
{x}}&\sin(x)\left (2\,x+1\right )-2\,\left ({x}^{2}+x+3\right )\sin({x
}^{2})x\\\noalign{\medskip}{e^{x}}\cos(x)+\cos({x}^{2}){e^{x}}&{e^{x}}
\left (2\,x+1\right )-2\,\cos({x}^{2})\sin({x}^{2})x\end {array}
\right ]
>fortran(C);
C(1,1) = sin(x)*cos(x)+(x**2+x+3)*exp(x)
C(1,2) = sin(x)*(2*x+1)-2*(x**2+x+3)*sin(x**2)*x
      C(2,1) = exp(x)*cos(x)+cos(x**2)*exp(x)
      C(2,2) = exp(x)*(2*x+1)-2*cos(x**2)*sin(x**2)*x
>fortran(C,optimized);
      t1 = sin(x)
      t2 = cos(x)
      t4 = x**2
      t5 = t4+x+3
      t6 = exp(x)
      t10 = 2*x+1
      t12 = sin(t4)
      t18 = cos(t4)
      C(1,1) = t1*t2+t5*t6
      C(1,2) = t1*t10-2*t5*t12*x
      C(2,1) = t6*t2+t18*t6
      C(2,2) = t6*t10-2*t18*t12*x
```

5 Ordinary Differential Equations in Maple

The following example illustrates the use of *dsolve()*, the function for solving ordinary differential equations. There are numerous solution options to choose from, but in the following, the *Laplace* option was chosen.

```
>de:=diff(u(t),t$2)+2*zeta*omega*diff(u(t),t)+omega^2*u(t);
```

$$de := \frac{d^2u(t)}{dt^2} + 2\,\tilde{\zeta}\,\tilde{\omega}\,\frac{du(t)}{dt} + \tilde{\omega}^2 u(t)$$

```
[Now lets make some assumptions about zeta and omega:
> assume(zeta,constant);
> assume(omega,constant);
> assume(zeta,real);
> assume(omega,real);
> assume(zeta<1);
[Now lets solve the ode!
>uu:=dsolve({de=10*Heaviside(t),u(0)=0,D(u)(0)=0},u(t),method=laplace):
```

Now, lets make use of the *subs()* function to substitute some numerical values for ζ and ω.

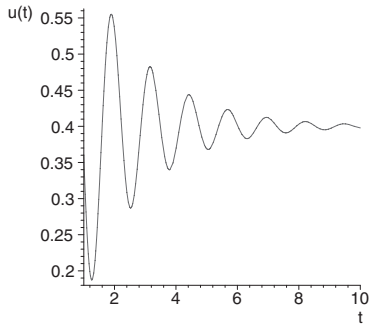
```
>uu2:=subs(omega=5,zeta=.1,uu):
```

Before we can plot the above, we must select the right hand side of the equation using the *rhs()* function. We could also plot the solution using the *odeplot()* function.

```
uu3:=rhs(uu2):
```

Now, lets plot uu3:

```
>plot(uu3,t=0..10);
```

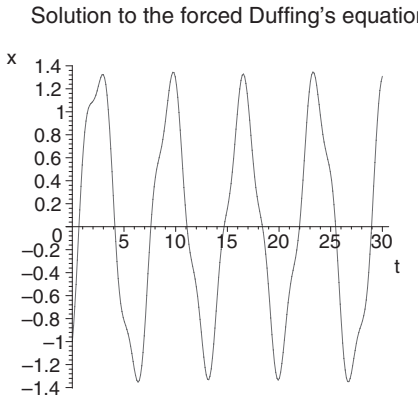


In the above examples, the output was suppressed using the colon (:) to save space. One shortcoming of Maple is that it is difficult to control the form of the output. Maple often choses to output solutions in the form of exponentials, and one has to use the *convert()* function to convert to sines and cosines. The only problem with this conversion is that it converts the exponentials real argument to cosh and sinh. Its rather difficult to factor out the exponentials with negative real part, and then express the complex exponentials as sines and cosines – at least, I have not been able to figure out how to do it!

6 Numerical Solution of ODEs

The following example illustrates numerial solution of ODEs, and in the process, demonstrates the use of the *op()* function.

```
Definition of ODE to solve:
> de1 := diff(x(t),t$2) + x(t)-(1/6)*x^3 = 1.5*cos(2.85*t);

                / 2      \
                |d       |               3
         de1 := |--- x(t)| + x(t) - 1/6 x  = 1.5 cos(2.85 t)
                |  2     |
                \dt      /

Solution using dsolve with the Gear's method using Bulirsh/Stoer extrapolation approach:
> p:=dsolve({de1,x(0)=-1,D(x)(0)=1},x(t),type=numeric,method=gear[bstoer]);

                    p := proc(x_gear)  ...  end


Enabling plot functions:
> with(plots):
Plotting initial solution:

> odeplot(p,[t,x(t)],0..30,labels=[t,x],title=`Solution to the forced
Duffing's equation`,numpoints=250);


Extracting the second operand in the numerical solution -- x(t).
> P:=t->rhs(op(2,p(t)));

                     P := t -> rhs(op(2, p(t)))
```

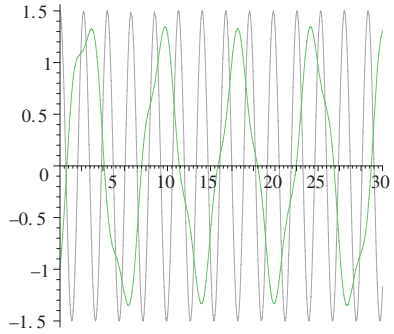


Extracting the first operand in the numerical solution, t, and defining the forcing function of t:

```
> F:=t->1.5*cos(2.85*rhs(op(1,p(t))));

                F := t -> 1.5 cos(2.85 rhs(op(1, p(t))))
```

Plotting both the solution and the forcing function with respect to time:

```
> plot([P,F],0..30);
```



[Note order 1/3 sub-harmonic solution in the above.

7 Example From Vibration of Shells and Plates

The following is some Maple code which was written to model the vibration of an annular plate. The *unapply()* command is used liberally, and a very powerful matrix forming command is used to form the coefficient matrix from the boundary conditions in order to take the determinant, and form the eigenvalue or characteristic equation. The solution is built up from its basic components starting with the radial solution, then the azimuthal (θ) solution. The solution is then operated on by differential operators, and combined into the proper form for the boundary equations – in this case, fixed-free conditions.

```
# Radial component
>   #
>   R:= unapply( A1_ * BesselJ( n_, lambda_ * r_ ) +
>                A2_ * BesselI( n_, lambda_ * r_ ) +
>                A3_ * BesselK( n_, lambda_ * r_ ) +
>                A4_ * BesselY( n_, lambda_ * r_ ),
>                n_, r_, lambda_, A1_, A2_, A3_, A4_ ):
```

```
>   #
>   # Azimuthal component:
>   #
>   Theta:= unapply( cos( n_ * ( theta_ + alpha_ ) ), n_, theta_, alpha_ ):
>   #
>   #  Separation of variables solution to biharmonic equation
>   #  in cylindrical cdts.:
>   U3[w]:= unapply( R(n_,r_,lambda_,A1_,A2_,A3_,A4_) * Theta(n_,theta_,alpha_),
>                 r_, theta_, n_, lambda_, alpha_, A1_, A2_, A3_, A4_ ):
```

The Bessel functions are part of Maple's core function library.

```
#
>   #Slope in radial direction:
>   #
>   dR := unapply(diff(R(n_,r_,lambda_,A1_,A2_,A3_,A4_),r_ ) ,
>    n_,r_,lambda_,A1_,A2_,A3_,A4_ ):
>   #
>   # Curvature in radial direction:
>   #
>   ddR := unapply(diff(R(n_,r_,lambda_,A1_,A2_,A3_,A4_),r_$2) ,
>    n_,r_,lambda_,A1_,A2_,A3_,A4_ ):
>   #
>   #Change in curvature in radial direction:
>   #
>   dddR:= unapply(diff(R(n_,r_,lambda_,A1_,A2_,A3_,A4_),r_$3 ) ,
>    n_,r_,lambda_,A1_,A2_,A3_,A4_):
```

Then next step is to form the boundary condition functions for the fixed-free plate:

```
#
>   # Defining the moment and shear functions
>   #
>   Mr:= unapply(
>     ddR(n_,r_,lambda_,A1_,A2_,A3_,A4_)
>     + nu*(1/r_*dR(n_,r_,lambda_,A1_,A2_,A3_,A4_)-
> n_^2/r_^2*R(n_,r_,lambda_,A1_,A2_,A3_,A4_)),
>     n_,r_,lambda_,A1_,A2_,A3_,A4_):
>
>
>   Vr:=
> unapply(dddR(n_,r_,lambda_,A1_,A2_,A3_,A4_)+1/r_*ddR(n_,r_,lambda_,A1_,A2_,A3_,A4_)
>   -1/r_^2*((2-nu)*n_^2+1)*dR(n_,r_,lambda_,A1_,A2_,A3_,A4_)+
>   n_^2/r_^3*(3-nu)*R(n_,r_,lambda_,A1_,A2_,A3_,A4_),n_,r_,lambda_,A1_,A2_,A3_,A4_):
>
```

Then next step is to form the boundary condition functions for the fixed-free plate:

```
  # Defining the characteristic matrix and the characteristic equation
>
>   print(‘All unapply definitions complete, now defining chrmatrix...‘);
>
>   #  Characteristic matrix prototype function:
>
>   chrmatrix:= (ri_,ro_,n_,lambda_) ->
```

```
>             (genmatrix([eval(R(n_,ri_,lambda_,
>                                A1_,A2_,A3_,A4_))=0,
>                           eval(dR(n_,ri_,lambda_,
>                                A1_,A2_,A3_,A4_))=0,
>                           eval(Mr(n_,ro_,lambda_,
>                                A1_,A2_,A3_,A4_))=0,
>                           eval(Vr(n_,ro_,lambda_,
>                                A1_,A2_,A3_,A4_))=0],
>                               [A1_,A2_,A3_,A4_]) ):
>
>   print(`First chrmatrix defined.  Now defining CHRMX...`);
>
>
>
>   CHRMX_n:= chrmatrix(ri_w/ro,1,NW,lambdaNORMED): # Evaluate at the
>
> # azimuthal  mode of interest, NW.
>
>   CHRMX_0:= chrmatrix(ri_w/ro,1,0,lambdaNORMED):
>
>   print(`CHRMX defined.  Now computing determinant...`);
>
>   Time:= time()-start_time;
>
>   chreqn_n := det(CHRMX_n):
>
>   chreqn_0 := det(CHRMX_0):
>
>   print(`Determinant defined.`);
```

In the above, the variables with appended underscore character, ri_,ro_,n and _,lambda_ are being used as dummy variables, so the underscore character makes sure that no previously defined variables make their way into the function prototype. This was a convention that one of my graduate students adopted, and it seems like a pretty good one. With the underscore character, there is no chance that one of the variables in a function definition was previously defined – as long as you don't assign any variables with the underscore appended to them!

The *genmatrix()* command was used to form the coefficient matrix out of the boundary equations. This is certainly not the only way to do this, nor especially the most efficient, but it does make the code easier to read and debug. Note that the first two equations in this function are the displacement and slope conditions at the fixed inner radius. The second two are the moment and shear conditions on the free outer radius.

After the chreqn_n, and chreqn_0 are defined, these functions must be solved for their roots. Dr. James Friend (Ph.D. UMR, 1998), has written a nice routine in Maple for this purpose, and it will be made available later, but for now, it is possible to simply plot the function, and then use *fsolve()* to find the root at the visually determined axis crossing.

Once the eigenvalues are determined, they may be substituted back into the boundary conditions, and three of the four constants may be solved for in terms of the fourth which is then set to unity. This process is demonstrated below:

```
for i from 1 to n_roots do
>
>    lambda:= eig_vals_n[i]/r_a: # Rescale lambdaNORMED for each eigenvalue.
>
> CCinv:= inverse( delrows( delcols( chrmatrix(ri, ro,
```

```
>            NW, lambda), 4..4 ), 4..4 ) );
>
> col4:= matrix(3, 1, [
>   [-col( chrmatrix(ri, ro,  NW, lambda), 4) [1] ],
>   [-col(chrmatrix(ri, ro, NW, lambda), 4) [2] ],
>   [-col( chrmatrix(ri, ro, NW, lambda), 4) [3] ]
>                          ]):
>
>     solnVec:= multiply(CCinv, col4):
>
>  #  Evaluation of constants:
>
>     print(‘Values of the constants in the radial solution:‘);
>
>   A1:= col( solnVec, 1 )[1]:
>   A2:= col( solnVec, 1 )[2]:
>     A3:= col( solnVec, 1 )[3]:
>    A4:= 1.0;
>
>  AA1[i]:=A1;
>  AA2[i]:=A2;
>  AA3[i]:=A3;
>  AA4[i]:=A4;
>
>   RR_n[i]:= R(NW, r, lambda, A1, A2, A3, A4):# Radial Modeshape for each n = 3 mode.
>
>  od:
```

In the above, the *inverse()* function is used to compute the inverse of the matrix reduce by deleting the fourth column and row using the *delcol()* command. The resulting inverse is then multiplied times the fourth column, which is negative since it was moved to the right hand side of the equations, yielding the solution vector containing the unknowns A1, A2, and A3. The resulting solution vector is then normalized with respect to A4.

Note that the eigen vector solution is carried out over the number of roots (eigenvalues), n_roots, determined from the solution of the characteristic equation. The result is a set of eigen vectors (the A_i) for each mode of the solution which may then be used in a forced solution, or simply plotted to observe the mode shapes.

8 Conclusions and Recommendations

As you can see, Maple is quite a powerful and complex system. Like any large software application, it takes considerable time to totally master Maple – if that is even humanly possible! However, progress can be made rapidly, and Maple can be used immediately to solve simple problems.

I recommend that you, gentle reader, use the help system to print out the descriptions of every function used in this document. Then, simply staple the function descriptions to the back of this document as an appendix. That way, you will have many of the tricks used herein at your finger tips! Good luck, and have fun!