→ Find the decimal equivalent of $(1011101)_2$,

	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	64	32	16	8	4	2	1
binary number	1	0	1	1	1	0	1

decimal value: $64 + 16 + 8 + 4 + 1 = 93$

$\therefore (1011101)_2 \iff (93)_{10}$

→ Find the binary equivalent of $(412)_{10}$

	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	256	128	64	32	16	8	4	2	1
binary digit	1	1	0	0	1	1	1	0	0
cummulative sum	256	384	384	384	400	408	412	412	412

$\therefore (110011100)_2 \iff (412)_{10}$

→ Consider a number system with 6 digits - a base 6 system. What is the equivalent of $(412)_{10}$ in this system?

	6^3	6^2	6^1	6^0
	216	36	6	1
base 6 digits	1	5	2	4
cummulative sum	216	396	408	412

$\therefore (412)_{10} \iff (1524)_6$

TOTAL NUMERICAL ERROR

Round-off and Truncation Error, Reduction Schemes, and Tradeoffs

- Define and contrast the terms truncation error and roundoff error.
- What are the primary mechanisms for reducing these two forms of numerical error?
- Discuss the tradeoff involved here and identify the dominant contribution to the total numerical error for most realistic applications.

Defn.

Round-off error — due to the fact that computers can only represent quantities with a finite number of digits. Therefore, floating point arithmetic is not exact on the computer.

Truncation error — due to the fact that many numerical methods employ approximations to represent exact mathematical operations. (a Taylor series approximation to derivatives is a good example)

Error Reduction

Roundoff error — (1) careful problem formulation (avoid subtractive cancellation)
(2) increase # of sig. figures (double precision)

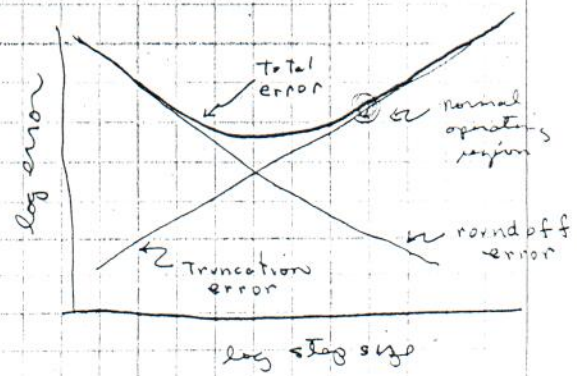
Truncation error — (1) use higher-order formulation
(2) reduce step size

Tradeoffs

In most applications, only a few terms are needed in a Taylor series to get a reasonable approximation. If more accuracy is required, it is usually easier and more efficient to reduce the step size than to increase the order of approximation.

However, because a decrease in step size leads to an increased number of arithmetic computations, as the truncation error decreases the roundoff error increases. (see below)

Truncation error is usually the dominant error mechanism in most well formulated problems (operation is to the right of the optimum in the fig. to the right)



Taylor Series Derivative Approximations

version #2

Given the polynomial function

$$f(x) = x^5 - 2x^4 + 3x^2 - 1$$

(a) Compute the exact first derivative of $f(x)$ at $x = 2.1$

$$f = x^5 - 2x^4 + 3x^2 - 1$$

$$f' = 5x^4 - 8x^3 + 6x$$

$$\therefore \text{ at } x = 2.1 \quad f'\Big|_{x=2.1} = 5(2.1)^4 - 8(2.1)^3 + 6(2.1)$$

$$= 97.2405 - 74.088 + 12.6$$

$$= \boxed{35.7525}$$

(b) Estimate the first derivative of $f(x)$ at $x = 2.1$ using a forward taylor series approximation with the following step sizes

$$h = [\,10^{-4} \quad 10^{-3} \quad 10^{-2} \quad 10^{-1} \quad 10^{0} \quad 10^{1}\,]$$

Using the exact result from Part a, make a log-log plot of the relative error in $f'(x)\big|_{x=2.1}$

From the plot, determine the value of n in the expression:

$$\epsilon = \alpha h^n = \mathcal{O}(h^n)$$

and explain your results!

→ use $f'(x) = \dfrac{f(x+h) - f(x)}{h}$ forward approx.

(c) Repeat Part b using a central taylor series approximation to $f'(x)\big|_{x=2.1}$. What is the order of error in this case? Explain your results!

→ use $f'(x) = \dfrac{f(x+h) - f(x-h)}{2h}$

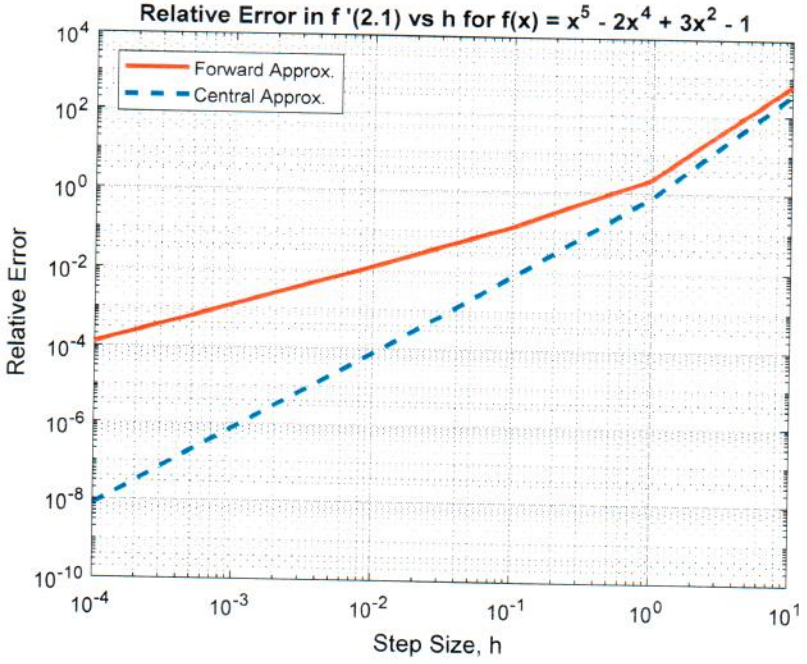
see TS_deriv_2.m and resultant plot and tabular data.
(also see TS_deriv_2results.doc)

Note: Use Matlab's polyval function to evaluate the above polynomial... (see help polyval)

```
>> ts_deriv_2

   Polynomial of interest is: x^5 - 2x^4 + 3x^2 - 1
   The exact derivative at x = 2.1 is :     35.7525

   Approximate Derivatives at x = 2.1
        h       Forward Approx    Rel Error          Central Approx    Rel Error
     1.0e-004        35.7568      1.194e-004            35.7525       7.638e-009
     1.0e-003        35.7952      1.195e-003            35.7525       7.636e-007
     1.0e-002        36.1821      1.202e-002            35.7552       7.636e-005
     1.0e-001        40.3031      1.273e-001            36.0256       7.639e-003
     1.0e+000       115.2425      2.223e+000            64.0525       7.916e-001
     1.0e+001     21692.6525      6.057e+002         12765.7525       3.561e+002
```



Forward Approx:

$h = 10^{-2} \rightarrow 10^{-3}$

$$n = \log(\varepsilon_1/\varepsilon_2) = \log\left(1.202\times10^{-2}/1.195\times10^{-3}\right) \approx 1.003$$

Central Approx:

$h = 10^{-2} \rightarrow 10^{-3}$

$$n = \log(\varepsilon_1/\varepsilon_2) = \log\left(7.636\times10^{-5}/7.636\times10^{-7}\right) \approx 2.000$$

Notes:

Since $\varepsilon = \alpha h^n$, then $\log\varepsilon = \log\alpha + n\log h$. Thus, n is the slope of the curve on a log-log plot.

From the plot, we see that the **forward** approximation has a slope of unity (i.e. a one decade change in step size leads to about a one decade change in relative error -- thus, $n_{forward} \rightarrow 1$.

For the **central** approximation, the slope of the log-log curve is just about a factor of two larger (i.e. a one decade change in h gives a two decade change in ε -- thus, $n_{central} \rightarrow 2$.

And, of course, from the development in the notes, this is exactly what we expected!!!

Also, note that, for very large h, this approximate theory starts to break down (error is just too large when $h > 1$ for this problem)…

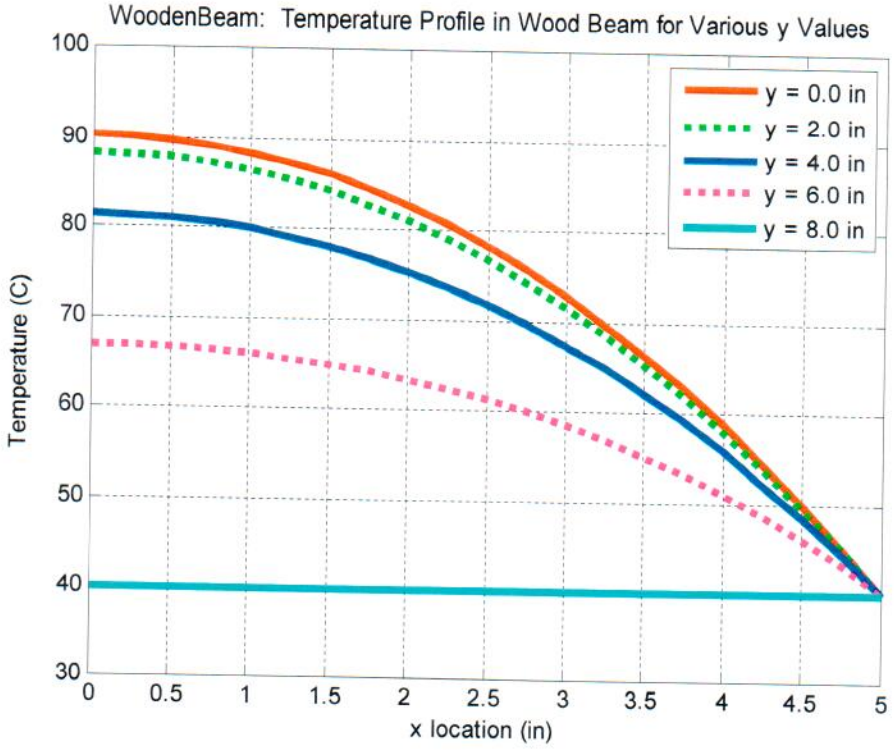
From a more quantitative view, we also have the following relationships:

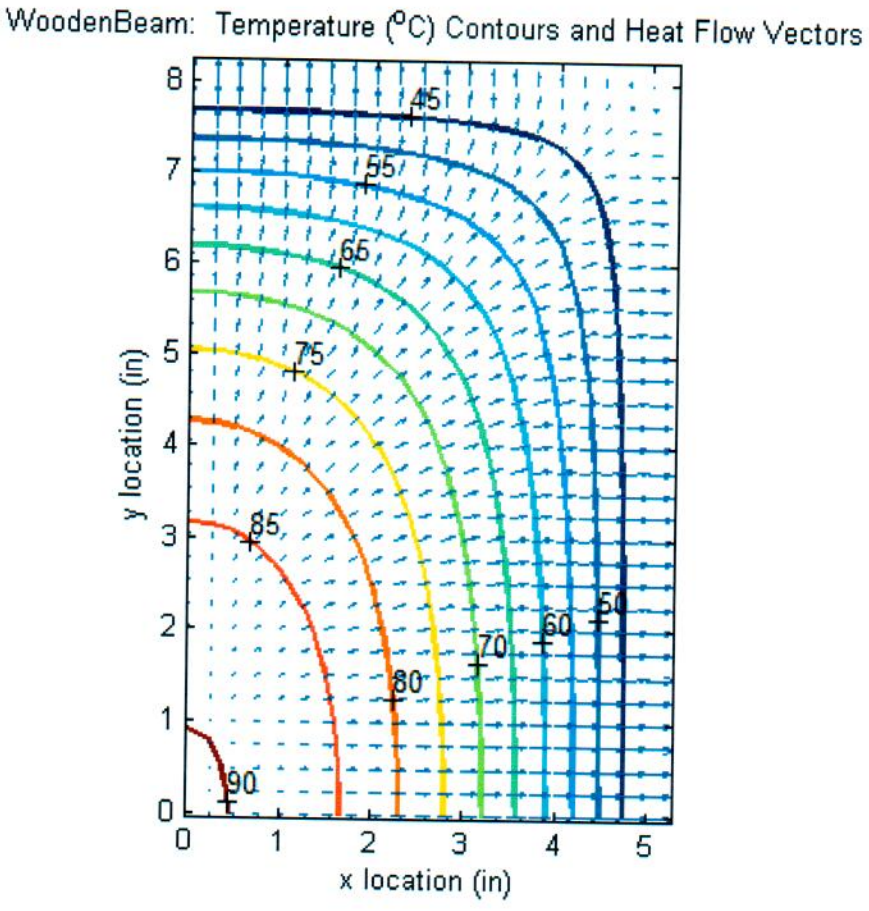
$$\frac{\varepsilon_1}{\varepsilon_2} = \left(\frac{h_1}{h_2}\right)^n \quad \text{or} \quad n = \frac{\log(\varepsilon_1/\varepsilon_2)}{\log(h_1/h_2)}$$

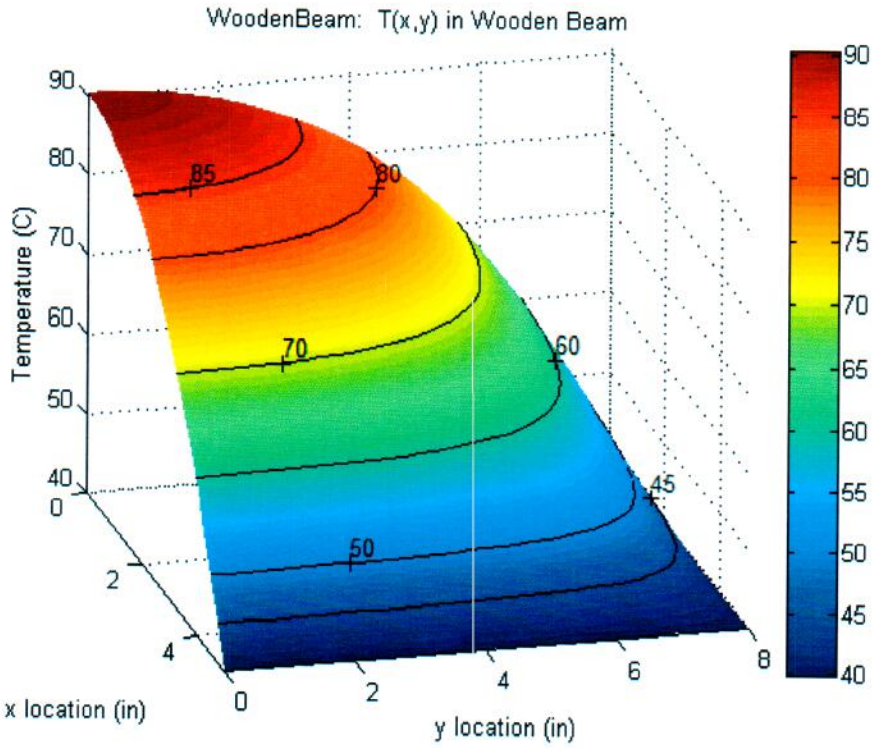
and, **for a change in h of a factor of 10**, n is given by $n = \log(\varepsilon_1/\varepsilon_2)$ [**see above calculations**]

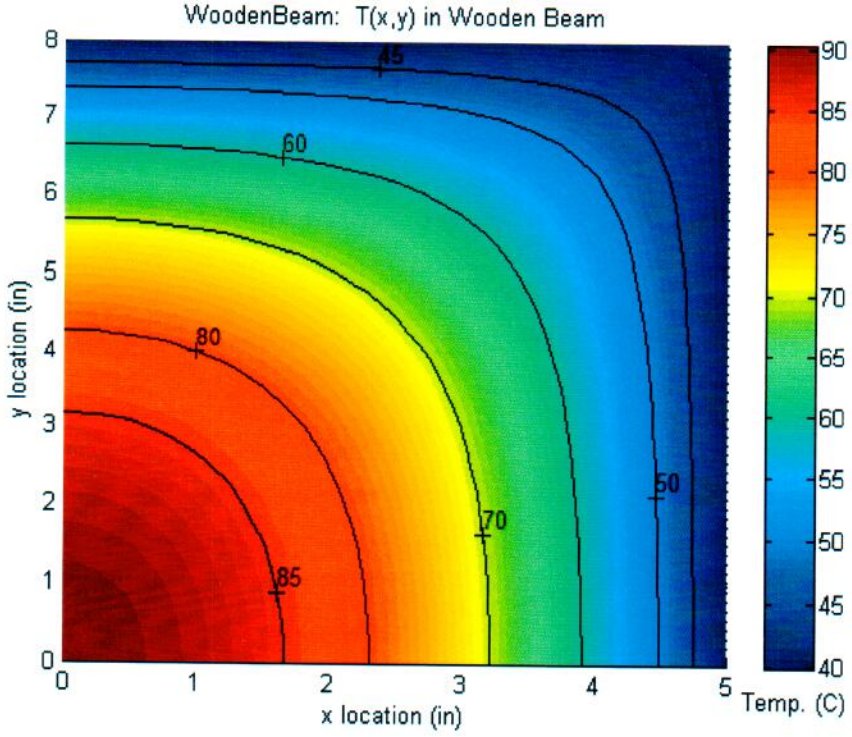
```
%
%   TS_DERIV_2.M   Approximation to the 1st Derivatives of a Simple Function
%
%   This file addresses the error associated with two different approximations
%   to the 1st derivative of a simple polynomial function:
%     f(x) = x^5 - 2x^4 + 3x^2 - 1
%     f'(x) = 5x^4 - 8x^3 + 6x   (exact representation)
%   The error versus step size is plotted and tabulated.
%
%   In evaluating this function, we will use Matlab's description of a
%   polynomial as a vector of coefficients and Matlab's polyval function
%   for efficiently evaluating the polynomial (type "help polyval", as needd).
%
%   File prepared by J. R. White, UMass-Lowell (last update: Oct. 2017)
%

    clear all,  close all,   nfig = 0;
%
%   define polynomial functions and evaluate the 1st derivative at x = 2.1
    fxc = [1 -2 0 3 0 -1];   fpxc = [5 -8 0 6 0];
    x = 2.1;  fpx = polyval(fpxc,x);
    fprintf(1,'\n   Polynomial of interest is: x^5 - 2x^4 + 3x^2 - 1\n');
    fprintf(1,'   The exact derivative at x = 2.1 is :  %10.4f \n\n',fpx);
%
%   evaluate the 1st derivative using forward & central approximations
    Nh = 6;  h = logspace(-4,1,Nh);
    fpxforw = (polyval(fxc,x+h) - polyval(fxc,x))./h;
    fpxcent = (polyval(fxc,x+h) - polyval(fxc,x-h))./(2*h);
%
%   compute relative errors (absolute values)
    reforw = abs((fpxforw-fpx)/fpx);    recent = abs((fpxcent-fpx)/fpx);
%
%   tabulate results
    fprintf(1,'   Approximate Derivatives at x = 2.1  \n');
    fprintf(1,'         h       Forward Approx   Rel Error        Central Approx   Rel
Error \n');
    for i = 1:Nh
      fprintf(1,'  %12.1e     %10.4f   %12.3e         %10.4f   %12.3e \n', ...
            h(i),fpxforw(i),reforw(i),fpxcent(i),recent(i));
    end
    fprintf(1,'\n')
%
%   plot results
    nfig = nfig+1;   figure(nfig)
    loglog(h,reforw,'r-',h,recent,'b--','LineWidth',2), grid
    title('Relative Error in f ''(2.1) vs h for f(x) = x^5 - 2x^4 + 3x^2 - 1');
    xlabel('Step Size, h'),ylabel('Relative Error')
    legend('Forward Approx.','Central Approx.','Location','NorthWest')
%
%   end of program
```


WoodenBeam: Temperature Profile in Wood Beam for Various y Values
Temperature (C)
x location (in)
y = 0.0 in
y = 2.0 in
y = 4.0 in
y = 6.0 in
y = 8.0 in
100
90
80
70
60
50
40
30
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


WoodenBeam: Temperature (°C) Contours and Heat Flow Vectors
y location (in)
x location (in)
45
55
65
75
85
90
80
70
60
50
8
7
6
5
4
3
2
1
0
0
1
2
3
4
5






```
> woodenbeam
Needed 10 terms for convergence to max error of 0.01 C
Needed 20 terms for convergence of vx
Needed 36 terms for convergence of vy
```

Given $T(x,y) = v(x,y) + w(x)$

where $w(x) = T_0 + \frac{S}{2k}\left(L_x^2 - x^2\right)$

$$v(x,y) = \sum_{n=1}^{\infty} B_n \cos(\lambda_n x)\cosh(\lambda_n y)$$

with $\lambda_n = \frac{(2n-1)\pi}{2L_x}$ and $B_n = \frac{2S}{kL_x}\left[\frac{(-1)^n}{\lambda_n^3 \cosh(\lambda_n L_y)}\right]$

Now $\vec{q} = -k\vec{\nabla}T$

$$= -k\left(\frac{\partial T}{\partial x}\hat{i} + \frac{\partial T}{\partial y}\hat{j} + \frac{\partial T}{\partial z}\hat{k}\right)$$

$\rightarrow 0$ for 2-D problem

or $\vec{q} = q_x\hat{i} + q_y\hat{j}$

where $q_x = -k\frac{\partial T}{\partial x} = -k\left(\frac{\partial v}{\partial x} + \frac{dw}{dx}\right)$

$$q_y = -k\frac{\partial T}{\partial y} = -k\left(\frac{\partial v}{\partial y}\right)$$

and $\frac{\partial v}{\partial x} = -\sum_{n=1}^{\infty} B_n \lambda_n \sin(\lambda_n x)\cosh(\lambda_n y)$

$$\frac{\partial w}{\partial x} = -\frac{S}{k}x$$

$$\frac{\partial v}{\partial y} = \sum_{n=1}^{\infty} B_n \lambda_n \cos(\lambda_n x)\sinh(\lambda_n y)$$

These expressions explicitly define $\vec{q}(x,y)$ for the given situation.

Certainly we can plot $q_x(x,y)$ and $q_y(x,y)$ separately.

However, Matlab's quiver command allows one to plot the vector field $\vec{q}(x,y) = q_x\hat{i}$ and $q_y\hat{j}$ and this gives a good visualization of the energy flows in this system.

Let's give this a try...

```
%
%   WOODENBEAM.M  2-D Steady State Heat Transfer Problem with Internal Heating
%
%   Analytical Soln using Separation of Variables to the following problem:
%     Txx(x,y) + Tyy(x,y) = -S/k   with Tx(0,y) = 0   T(Lx,y) = Tr = 40 C
%                                       Ty(x,0) = 0   T(x,Ly) = Tt = 40 C
%
%   Physically, this equation and BCs represent the upper right quadrant of
%   a very long wooden beam heated by microwaves (a uniformly distributed
%   internal heat source, S).
%
%   The goal here is to evaluate the analytical solution (given in notes) and to
%   plot the resultant 2-D temperature distribution, T(x,y).
%   Evaluation and ploting of the heat flux vectors is also done in the last code
%   segment.
%
%   File prepared by J. R. White, UMass-Lowell (last update: Oct. 2017)
%

      clear all,  close all,  nfig = 0;
%
%   problem data
      width = 10*2.54/100;    Lx = width/2;    % half width of beam cross section (m)
      height = 16*2.54/100;   Ly = height/2;   % half height of beam cross section (m)
      To = 40;                                 % right & top temp (C)
      k = 0.16;                                % thermal conductivity of wood (W/m-C)
      S = 1200;                                % internal heat source density (W/m^3)
      maxt = 50;                               % max number of nonzero terms
      tol = 0.01;                              % convergence limit for series expansion
%
%   points to evaluate function (defines upper right quadrant of the wooden beam)
      Nx = 26;    x = linspace(0,Lx,Nx);
      Ny = 41;    y = linspace(0,Ly,Ny);
      [xx,yy] = meshgrid(x,y);
%
%   calc eigenvalues and coefficents in expansion (for n = 1, 2, 3, ... ,maxt)
      dd = 0.5*pi/Lx;       cc = 2*S/(k*Lx);
      ln = zeros(maxt,1);    bn = zeros(maxt,1);
      for n = 1:maxt
        ln(n) = (2*n-1)*dd;      bn(n) = cc*(-1)^n/(ln(n)^3*cosh(ln(n)*Ly));
      end
%%
%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   %   Compute and Plot Temperature Distribution  %
%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   evaluate w(x)  (use 2-D matrix xx for x since this is valid for each y value)
      w = To + (0.5*S/k)*(Lx*Lx - xx.*xx);
%
%   determine spatial distribution for v(x,y)
%   Note: We are using the absolute error here instead of the usual relative error,
%         where the max abs error (tol) = 0.01 C (given in problem statement).
      n = 0;   maerr = 1.0;   v = zeros(Ny,Nx);
      while maerr > tol   &&   n < maxt
        n = n+1;   vn = bn(n)*cos(ln(n)*xx).*cosh(ln(n)*yy);
        v = v + vn;  maerr = max(max(abs(vn)));
```

```
    end
    disp([' Needed ',num2str(n),' terms for convergence to max error of 0.01 C'])
%
%  construct complete solution (both v and w are 2-D arrays of the same size)
    T = v + w;
%
%  plot T(x,y) for a few y values  (quantitative 2-D plot)
    Ncm = 6;   scm = ['r-';'g:';'b-';'m:';'c-';'k:']; % set color and marker code
    conv = 100/2.54;                                   % convert meters to inches
    nfig = nfig+1;  figure(nfig);  jj = 0;
    for j = 1:8:Ny
      jj = jj+1;
      h(jj) = plot(conv*x,T(j,:),scm(jj,:),'LineWidth',3); hold on
      st(jj,:) = sprintf('y = %3.1f in',conv*y(j));
    end
    title('WoodenBeam:  Temperature Profile in Wood Beam for Various y Values')
    grid,xlabel('x location (in)'),ylabel('Temperature (C)')
    legend(h,char(st))
%
%  various 3-D plots
%   nfig = nfig+1;  h = figure(nfig);  set(h,'Renderer','Zbuffer')
    nfig = nfig+1;  figure(nfig);  colormap(jet)
    surf(conv*xx,conv*yy,T), shading interp; colorbar, view(75,25), hold on
    [cc,hh] = contour3(conv*xx,conv*yy,T, [85 80 70 60 50 45]);
    clabel(cc), set(hh,'EdgeColor','k')
    axis('tight')
    title('WoodenBeam:  T(x,y) in Wooden Beam')
    ylabel('y location (in)'),   xlabel('x location (in)')
    zlabel('Temperature (C)'),   hold off
%
    nfig = nfig+1;  figure(nfig);  colormap(jet)
    surf(conv*xx,conv*yy,T), shading interp; colorbar, view(2), hold on
    [cc,hh] = contour3(conv*xx,conv*yy,T, [85 80 70 60 50 45]);
    clabel(cc), set(hh,'EdgeColor','k')
    axis('tight')
    title('WoodenBeam:  T(x,y) in Wooden Beam')
    ylabel('y location (in)'),   xlabel('x location (in)')
    zlabel('Temperature (C)'), gtext('Temp. (C)'),  hold off
%%
%  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  %   Compute and Plot Heat Flux Distribution (this is a vector relationship) %
%  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  evaluate spatial distribution for the various derivatives
    wx = -S*xx/k;
%  Note: here we go back to using a relative error (the usual approach)
    n = 0;   mrerr = 1.0;   tol = 1e-2;  vx = zeros(Ny,Nx);
    while mrerr > tol   &&   n < maxt
      n = n+1;   vn = -bn(n)*ln(n)*sin(ln(n)*xx).*cosh(ln(n)*yy);
      vx = vx + vn;
      i = find(vx);  rerr = vn(i)./vx(i);   mrerr = max(max(abs(rerr)));
   end
    disp([' Needed ',num2str(n),' terms for convergence of vx'])
%
    n = 0;   mrerr = 1.0;   tol = 1e-2;  vy = zeros(Ny,Nx);
    while mrerr > tol   &&   n < maxt
```

```
    n = n+1;    vn = bn(n)*ln(n)*cos(ln(n)*xx).*sinh(ln(n)*yy);
    vy = vy + vn;
    i = find(vy);  rerr = vn(i)./vy(i);    mrerr = max(max(abs(rerr)));
  end
  disp([' Needed ',num2str(n),' terms for convergence of vy'])
%
  qx = -k*(vx + wx);    qy = -k*vy;    qmag = sqrt(qx.*qx + qy.*qy);

%

  nfig = nfig+1;  figure(nfig);  colormap(jet)
  Tc = 45:5:90;      % select contour levels for good visualization
  [c,h] = contour(conv*xx,conv*yy,T,Tc); clabel(c); hold on
  set(h,'Linewidth',2)
  quiver(conv*xx,conv*yy,qx,qy,'LineWidth',1), hold off , axis image
  title('WoodenBeam:  Temperature (^oC) Contours and Heat Flow Vectors')
  ylabel('y location (in)'),    xlabel('x location (in)')
  hold off
%  end of problem
```