Introduction to Finite Difference Methods

Since most physical systems are described by one or more differential equations, the solution of differential equations is an integral part of many engineering design studies. Your introductory course in Differential Equations has shown you how to solve many simple ODEs, and we have already reviewed some of the analytical methods studied in Differential Equations as part of this course -- and we will do several more examples before the end of the semester. In addition, if time permits, we will also use Lesson 8 to illustrate two specific numerical techniques for solving ODEs within Matlab. In particular, we will use Matlab's ***ode23*** or ***ode45*** routine to directly solve initial value problems (IVPs), and we will use this same ODE solver within an iterative method, called the Shooting Method, to solve a wide range of boundary value problems (BVPs). However, these methods are only useful for differential equations with a single independent variable -- that is, they can only solve ordinary differential equations (ODEs).

For problems involving more than one independent variable -- that is, governed by partial differential equations (PDEs) -- the most common solution method involves discretizing the original differential equation (DE) and converting it into a series of recursive and/or algebraic equations. Basically, this requires that one discretize each of the independent variables, essentially breaking the problem into a number of finite volumes or elements. Then, the continuous DE is evaluated at each discrete point or node in the system. Finally, upon approximating the discrete derivatives in the latter equations with finite difference approximations (as developed in the Lesson 4 Lecture Notes), one can completely convert the original continuous differential equations into a series of difference equations -- and it is the discrete difference equations that are evaluated on the computer. This so-called Finite Difference (FD) method can be applied to both ODEs and PDEs; thus, it is a popular method that is used in a wide range of practical applications.

There are several variations of this basic theme that gives rise to a number of specific methods that are used in different engineering applications. Unfortunately, we do not have time to elaborate on any of these methods, since doing so could easily take the rest of the semester. However, with the introduction of the Taylor Series approximations for the discrete 1st and 2nd order derivatives in Lesson 4, we can, at least, take a short time to introduce the basic concepts and to illustrate the key differences that occur in solving IVPs versus BVPs via the FD method. And, if time permits, we will show in Lesson 8 how the FD methods compare to the other numerical techniques mentioned above (i.e. the adaptive Runge-Kutta methods implemented within ***ode23*** and ***ode45*** for IVPs and the Shooting Method technique for BVPs with a single independent variable).

To illustrate some of the ideas mentioned above, we will apply the basic Finite Difference method to two problems, an IVP and a BVP, as follows:

Case 1: Pendulum Dynamics via the FD Method (see the Lesson 1 Lecture Notes and the **pendulum_dynamics.pdf** file for background).

Case 2: Heat Transfer in a Rectangular Fin via the FD Method (see the **rect1d_fin_1.pdf** file that was studied after Lesson 3 for the development of the pertinent equations).

We have already solved both these problems via analytical techniques, so we are already familiar with both these systems. The pendulum dynamics problem is an IVP and it gives rise to a single recursive equation upon discretization. The rectangular fin heat transfer problem, on the other hand, is a BVP, and this type of problem leads to a system of simultaneous equations. One of the main goals of this set of examples is to illustrate the difference between the computer solution of IVPs and BVPs. Thus, let's set up each problem separately to illustrate the above comments in more concrete terms:

Case 1: Pendulum Dynamics via the FD Method

The continuous form of the equation of motion for the linearized pendulum model (assumes small angular displacements) is given by

$$\frac{d^2}{dt^2}\theta + \frac{c}{m}\frac{d}{dt}\theta + \frac{g}{L}\theta = 0 \qquad \text{with} \quad \theta(0) = \theta_o \quad \text{and} \quad \omega(0) = \left.\frac{d}{dt}\theta\right|_{t=0} = \omega_o \tag{1}$$

This 2nd order linear ODE can be solved analytically to give the angular position vs. time, $\theta(t)$, as

$$\theta(t) = e^{\alpha t}\left(c_1 \cos\beta t + c_2 \sin\beta t\right) \tag{2}$$

$$\text{with} \qquad \alpha = -\frac{c}{2m} \qquad \text{and} \qquad \beta = \sqrt{\frac{g}{L} - \left(\frac{c}{2m}\right)^2} \tag{3}$$

where we have assumed that the pendulum does indeed have damped oscillatory behavior. The c_1 and c_2 coefficients in the general solution can be evaluated by applying the specific initial conditions given above to give

$$c_1 = \theta_o \qquad \text{and} \qquad c_2 = \frac{\omega_o - \alpha c_1}{\beta} \tag{4}$$

Equations (2) – (4) represent an analytical solution to the IVP given in eqn. (1).

Now, instead of solving the problem analytically, let's solve the pendulum dynamics problem using a simple Finite Difference scheme -- as an illustration of how to use the FD method for IVPs. For this method, we start by discretizing the time variable by defining a small time interval, Δt, which we will keep fixed for simplicity. Then, $t \to t_i$, $t+\Delta t \to t_{i+1}$, etc., where

$$t_{i+1} = t_i + \Delta t \tag{5}$$

with i being a discrete time index (with $t_1 = t_o = 0$ for this problem).

Now, to discretize the continuous ODE, we simply evaluate every time dependent term in eqn. (1) at discrete time point t_i, or

$$\left.\frac{d^2}{dt^2}\theta\right|_{t_i} + \left.\frac{c}{m}\frac{d}{dt}\theta\right|_{t_i} + \left.\frac{g}{L}\theta\right|_{t_i} = 0 \tag{6}$$

Now, using central FD approximations for both derivatives evaluated at t_i, we have (see eqns. (12) and (13) in the Lesson 4 Lecture Notes),

$$\frac{\theta_{i-1}-2\theta_i+\theta_{i+1}}{\Delta t^2}+\frac{c}{m}\frac{\theta_{i+1}-\theta_{i-1}}{2\Delta t}+\frac{g}{L}\theta_i=0 \tag{7}$$

Multiplying this recursive equation by Δt^2 gives

$$\theta_{i-1}-2\theta_i+\theta_{i+1}+\frac{c}{m}\frac{\Delta t}{2}\left(\theta_{i+1}-\theta_{i-1}\right)+\frac{g}{L}\Delta t^2\theta_i=0$$

and collecting terms gives

$$\left(1+\frac{c\Delta t}{2m}\right)\theta_{i+1}=\left(2-\frac{g\Delta t^2}{L}\right)\theta_i+\left(\frac{c\Delta t}{2m}-1\right)\theta_{i-1}$$

To simplify this a little, we can define some constants

$$a=\left(1+\frac{c\Delta t}{2m}\right) \qquad b=\left(2-\frac{g\Delta t^2}{L}\right) \qquad d=\left(\frac{c\Delta t}{2m}-1\right) \tag{8}$$

and write the final recurrence relationship as

$$\theta_{i+1}=\frac{b}{a}\theta_i+\frac{d}{a}\theta_{i-1} \tag{9}$$

Now, if we know θ_1 and θ_2, then eqn. (9) can be used to estimate θ_3. Knowing θ_2 and θ_3 then leads to θ_4, and so on -- this is why eqn. (9) is said to be a recursive equation. Thus, to simulate the dynamics of the linear pendulum for any desired time interval, all we need is two starting positions, θ_1 and θ_2.

The first point, θ_1, is given directly as part of the initial conditions in eqn. (1). For the second point, θ_2, we need to use the initial condition on $d\theta/dt$ and a forward FD approximation (see eqn. (10) in the Lesson 4 Lecture Notes), as follows:

$$\left.\frac{d}{dt}\theta\right|_{t_1=0}=\omega_o\approx\frac{\theta_2-\theta_1}{\Delta t} \tag{10}$$

and, solving this for θ_2 gives

$$\theta_2=\theta_1+\omega_o\Delta t \tag{11}$$

Thus, eqns. (1) and (11) give us the desired starting points, so repeatedly evaluating eqn. (9) for $i = 2, 3, \ldots, N-1$ (where N is the number of small time intervals) gives the desired simulation.

This FD method for the pendulum dynamics was implemented in Matlab program **pendulum_2.m**. In addition, for ease of comparison and for validation of the FD method, we also evaluated the analytical solution given in eqns. (2) – (4). These two solution schemes can be seen in the listing of **pendulum_2.m** provided in Table 1. After setting the variable parameters (m, c, L, and g) and the ICs for the problem, we first evaluate the analytical solution (note that this is the identical problem that was solved in **pendulum_1.m** from Lesson 1). Then, the equation constants and the starting points for the numerical solution are set and eqn. (9) is evaluated in a simple ***for … end*** loop. The result of this operation for all time points is an estimate of the pendulum's angular position at each discrete time point. Finally, both the analytical and FD solutions are plotted, with the results shown in Fig. 1. As apparent, the analytical solution is exactly as developed previously in Lesson 1 and, the FD method, with the Δt chosen ($\Delta t = 0.05$ seconds), gives pretty good agreement to the analytical result. We know, of course, that the error associated with the FD method used here is roughly $O(\Delta t^2)$, since we used central approximations for the discrete derivative approximations (except for the calculation of θ_2). However, here we just wanted to show the basic scheme for the solution of IVPs via the FD method -- additional analysis and accuracy comparisons can wait for some other day!!!

Table 1 Listing of the pendulum_2.m program.

```
%
%   PENDULUM_2.M  Evaluate and plot the dynamics of a simple linear pendulum
%                 via the FD method (compared to the analytical solution)
%
%   This example evaluates and plots the angular position of a pendulum with a point
%   mass at the end.  The focus is on the solution of IVPs using the finite difference
%   (FD) method.  For comparison, we also plot the analytical solution.  The equations
%   for the FD method are developed in the Lecture Notes following Lesson 4 (see
%   fd_intro.pdf), with the basic mathematical model and analytical solution given in
%   the Lesson 1 Lecture Notes (also see pendulum_dynamics.pdf).
%
%   A similar program, pendulum_1.m, focuses only on evaluating and plotting the
%   analytical solution.
%
%   File prepared by J. R. White, UMass-Lowell (last update: Oct. 2017)
%

      clear all;   close all;   nfig = 0;
%
%   define problem parameters
      m = 1;                    % pendulum mass (kg)
      c = 2;                    % friction coeff (kg/s)
      L = 1;                    % length of pendulum arm (m)
      g = 10;                   % gravitational acceleration (m/s^2)
      poso = 30*pi/180;         % initial position (degrees converted to radians)
      velo = 0;                 % initial velocity (radians/s)
%
%       *************************
%       *  ANALYTICAL SOLUTION  *
%       *************************
%
%   evaluate roots of characteristic equation and compute equation coefficients
      a = -c/(2*m);   b = sqrt(g/L - a^2);
      c1 = poso;          c2 = (velo - a*c1)/b;
      d1 = b*c2 + a*c1;   d2 = a*c2 - b*c1;
%
%   define discrete time domain variable (with time in seconds)
      Nt = 101;   to = 0;   tf = 5;   t = linspace(to,tf,Nt);
%
%   evaluate analytical expressions for angular position
      posa = exp(a*t).*(c1*cos(b*t) + c2*sin(b*t));
%
%       *****************
%       *  FD SOLUTION  *
%       *****************
%
%   evaluate equation coefficients and starting points
%   (can vary Nt to see the sensitivity of the numerical soln to value of dt)
      N = Nt-1;  dt = (tf-to)/N;  % dt is consistent with the time vector from above
      a = 1 + c*dt/(2*m);   b = 2 - g*dt*dt/L;   d = c*dt/(2*m) - 1;
      posn = zeros(1,Nt);   posn(1) = poso;    posn(2) = posn(1) + velo*dt;
%
%   now loop over all remaining discrete time points
      for i = 2:N
```

```
         posn(i+1) = b*posn(i)/a + d*posn(i-1)/a;
       end
%
%        ************************
%        *  PLOT BOTH SOLUTIONS  *
%        ************************
%
       nfig = nfig+1;   figure(nfig)
       plot(t,posa*180/pi,'r-',t,posn*180/pi,'b:','LineWidth',2),grid
       title('PENDULUM\_2: Pendulum Dynamics  --  \theta(t) vs t')
       xlabel('Time (sec)'),ylabel('Angular Position (degrees)')
       legend('Analytical Solution','FD Solution')
%
%    end of problem
```

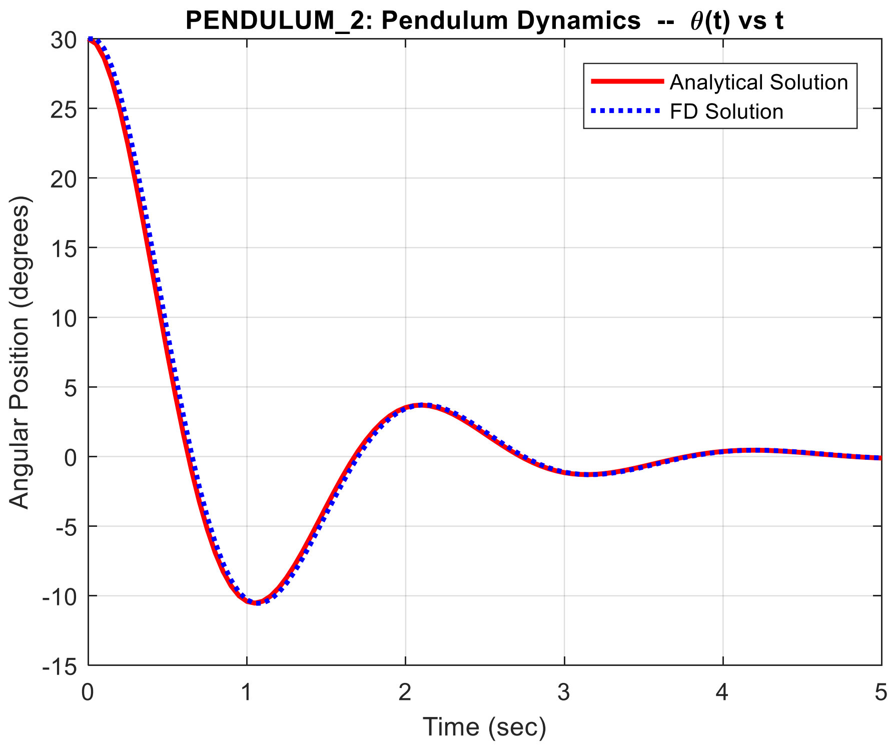
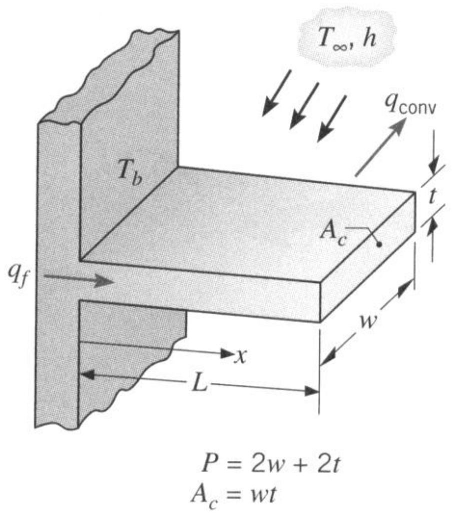


Fig. 1 Pendulum dynamics -- comparison of the analytical and FD methods.

Case 2: Fin Heat Transfer via the FD Method

Solving BVPs via finite difference methods is somewhat more difficult than solving IVPs. We follow the same basic steps, but each step needs to be done carefully and, in the end, we have a system of simultaneous equations (one for each unknown in the problem) rather than a single recurrence relation. To see how all this comes about, let's reconsider the rectangular fin problem that was solved at the end of Lesson 3. The basic problem geometry of interest is shown in the sketch to the right.



The governing ODE for this problem is given by (see **rect1d_fin_1.pdf** for details)

$$\frac{d^2T}{dx^2} - m^2(T - T_\infty) = 0 \qquad \text{with} \qquad m^2 = \frac{hP}{kA_c} \tag{12}$$

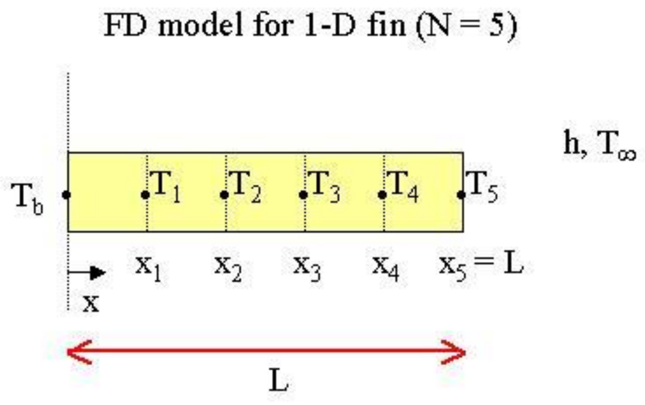
where the specific BCs for this problem include a fixed temperature on the left (at x = 0) and convection to the environment on the right (at x = L), or

$$T(0) = T_b \qquad \text{and} \qquad -k\left.\frac{dT}{dx}\right|_{x=L} = h\left.(T - T_\infty)\right|_{x=L} \tag{13}$$

The analytical solution to eqn. (12) with the BCs given in eqn. (13) was developed in detail in **rect1d_fin_1.pdf** and it can be written as

$$T(x) = T_\infty + \frac{\cosh m(L-x) + \frac{h}{mk}\sinh m(L-x)}{\cosh mL + \frac{h}{mk}\sinh mL}\left(T_b - T_\infty\right) \tag{14}$$

Now, to develop a numerical solution to this problem using the FD method, we again start by discretizing the independent variable, x. Here, we need to be careful to number only the nodal points where the temperature is to be determined -- for example, here T(0) = T_b is known, but T(L) = T_L is an unknown temperature that will have to be determined as part of the problem. As an example, let's say N = number of unknowns = 5. In this case, a side view of the fin geometry would give the sketch shown to the right, and we can compute the discrete spatial increment, Δx, as



$$\Delta x = \frac{L-0}{N} = \frac{L}{N} \tag{15}$$

In this case, the vector of discrete spatial points that specify the location of the unknown temperatures to be computed can be written as **x** = [x_1 x_2 x_3 x_4 x_5] which, in Matlab, can be easily generated with the use of the colon operator, **x = dx:dx:L**, where dx = Δx. Note that the first value of **x** is not zero, but instead, $x_1 = \Delta x$!

Now, with the nodal arrangement defined, we discretize the continuous ODE, or

$$\left.\frac{d^2T}{dx^2}\right|_{x_i} - m^2 (T - T_\infty)\Big|_{x_i} = 0 \tag{16}$$

Again, using a second order central approximation for the 2nd derivative, we have

$$\frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2} - m^2 (T_i - T_\infty) = 0$$

or

$$T_{i-1} - \left(2 + m^2 \Delta x^2\right) T_i + T_{i+1} = -m^2 \Delta x^2 T_\infty \tag{17}$$

Now, it is important to note that this expression is only valid for interior nodes (i = 2:N-1) -- since we used a central approximation for d^2T/dx^2 (recall that a central difference approximation cannot be applied at the end points). Thus, we always need to treat the end nodes as special cases!!!

For i = 1, eqn. (17) can be used directly if we note that $T_{i-1} = T_0 = T_b$, the fin's base temperature. Thus, for i = 1, we have

$$-\left(2 + m^2 \Delta x^2\right) T_1 + T_2 = -m^2 \Delta x^2 T_\infty - T_b \tag{18}$$

For i = N, things are not so easy, since nothing is known to the right of node N -- that is, T_{N+1} is not defined. There are several ways to handle this situation, however. Probably the easiest method is to simply generate a backward approximation to the desired derivative at x = L. To do this, let's write T_N'' as follows,

$$T''\Big|_{x_N} = T_N'' = \left.\frac{d}{dx}\left(T'\right)\right|_{x_N} \approx \frac{T_N' - T_{N-1}'}{\Delta x} \tag{19}$$

Now, we can write a central approximation for the 1st derivative at point N-1, or

$$T_{N-1}' = \frac{T_N - T_{N-2}}{2\Delta x} \tag{20}$$

and, for T_N', we can directly use the given BC at x = L,

$$-kT_N' = h(T_N - T_\infty) \tag{21}$$

that represents convection to the environment on the right face of the fin.

Substitution of eqns. (20) and (21) into eqn. (19) gives

$$T_N'' = \frac{-\frac{h}{k}\left(T_N - T_\infty\right) - \frac{T_N - T_{N-2}}{2\Delta x}}{\Delta x} = -\left(\frac{h}{k\Delta x} + \frac{1}{2\Delta x^2}\right)T_N + \frac{h}{k\Delta x}T_\infty + \frac{1}{2\Delta x^2}T_{N-2} \qquad (22)$$

Finally, putting this expression into eqn. (16) for i = N gives a proper discrete balance equation for the last node in the fin's discrete geometry representation,

$$-\left(\frac{h}{k\Delta x} + \frac{1}{2\Delta x^2}\right)T_N + \frac{h}{k\Delta x}T_\infty + \frac{1}{2\Delta x^2}T_{N-2} - m^2\left(T_N - T_\infty\right) = 0$$

or

$$T_{N-2} - \left(2m^2\Delta x^2 + 1 + \frac{2h\Delta x}{k}\right)T_N = -\left(\frac{2h\Delta x}{k} + 2m^2\Delta x^2\right)T_\infty \qquad (23)$$

Together, eqns. (17), (18), and (23) give a system of N equations with N unknowns -- the unknown temperature at each discrete x_i location. These equations can be written in matrix form,

$$\mathbf{AT} = \mathbf{b} \qquad (24)$$

and easily solved in Matlab for the desired temperature vector, **T**. Note that this is done in Matlab with the backslash operator, or

```
T = A\b
```

We will discuss this approach to solving a matrix equation in detail in Lesson 6. For now, it simply allows us to solve the system of linear equations given by eqn. (24), with the resultant discrete temperatures stored in vector **T**.

The rectangular fin heat transfer problem is solved in Matlab file **rect1d_fin_2.m**, whose listing is given in Table 2. The program is broken into five key steps, as follows:

1. set problem parameters
2. compute analytical solution as given by eqn. (14)
3. set up the coefficient matrices, **A** and **b**, in eqn. (24) for the FD solution [obtained from eqns. (15), (17), (18), and (23)]
4. solve the resultant system of equations using Matlab's backslash operator, **T** = **A\b**
5. plot both the analytical and FD solutions to verify the FD methodology

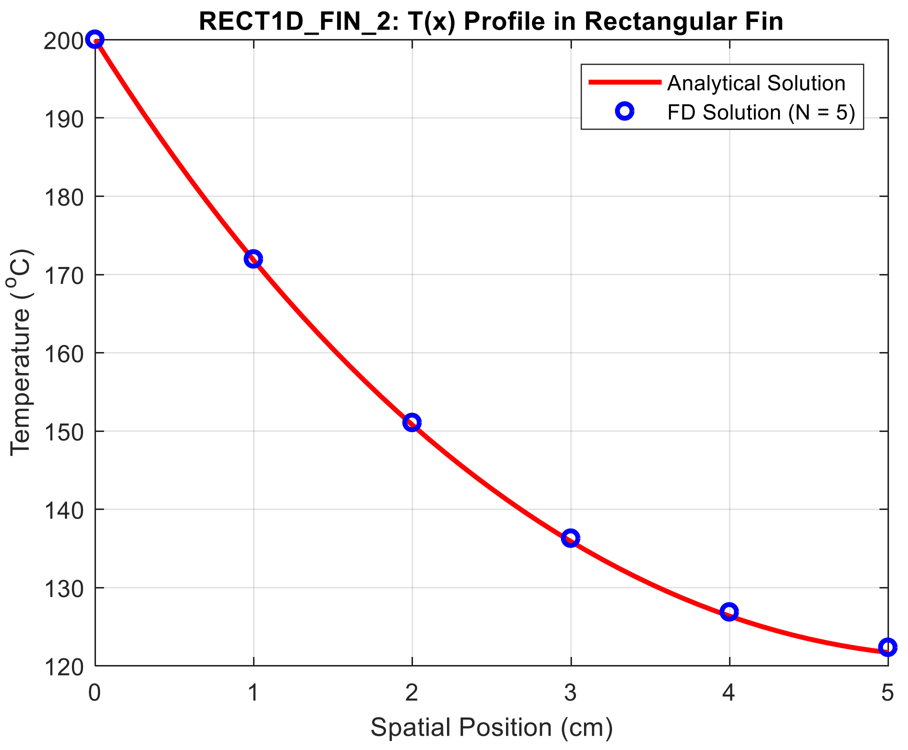
Note that Step #3 represents the only new aspect of this program. We first compute the desired Δx based on a user-specified value for N and set the proper discrete **x** vector. Then, the coefficient equations for the ith node are evaluated, with the non-zero values stored in the proper locations of the **A** and **b** matrices. Note that, when working with the balance equation for node i (which is equation i in the system of N equations), the subscripts i-1, i, and i+1 represent nearest neighbor coupling in the system and, from a mathematical view, these coefficients are placed in columns i-1, i, and i+1 of row i in the **A** matrix. Thus, it is easy to set up the appropriate matrix equations for an arbitrary number of unknowns, N, in the problem by simply defining to the non-zero coupling coefficients in the ith balance equation. Since, for a 1-D BVP problem, node 1 and node N are always special cases (i.e. the boundary nodes), these are treated separately, with the remaining interior nodes all treated within the single ***for … end*** loop. In this way, it is pretty easy to define the full **A** and **b** coefficient matrices, and then simply let Matlab solve the system of equations for the desired temperature vector, **T**.

Table 2 Listing of the rect1d_fin_2.m program.

```
%
%   RECT1D_FIN_2.M   Introduction to FD Methods for Solving BVPs
%                Heat Transfer Analysis of a Rectangular Fin Arrangement
%
%   This file illustrate the finite difference (FD) method for solving BVPs.
%   We solve the same rectangular fin heat transfer problem that was treated in
%   program rect1d_fin_1.m.  Here, the focus is on comparing the FD solution
%   to the analytical solution for the temperature profile, T(x), along the
%   length of the fin.  The development of the FD equations programmed here is
%   contained within the Lecture Notes (see fd_intro.pdf).
%
%   File prepared by J. R. White, UMass-Lowell  (last update: Oct. 2017)
%

      clear all,  close all,  nfig = 0;
%
%   identify basic problem data
      w = 1;                             % unit width of fin (m)
      thk = 0.01;                        % fin thickness (m)
      L = 0.05;                          % fin length  (m)
      Tb = 200;                          % fin base temperature (C)
      Tinf = 30;                         % environment temperature (C)
      h = 500;                           % heat transfer coeff  (W/m^2-C)
      k = 200;                           % fin thermal conductivity (W/m-C)
%
%   compute some derived parameters
      P = 2*w + 2*thk;                   % perimeter
      Ac = w*thk;                        % cross section area (conduction area)
      m = sqrt(h*P/(k*Ac));              % constant in derived equations (see notes)
      m2 = m*m;                          % constant in derived equations (see notes)
      hmk = h/(m*k);                     % constant in derived equations (see notes)
      bot = cosh(m*L) + hmk*sinh(m*L);  % constant in derived equations (see notes)
%
%       *************************
%       *  Analytical Solution  *
%       *************************
%
      Nx = 51;  xa = linspace(0,L,Nx)';  % independent spatial variable
      Ta = Tinf + (Tb - Tinf)*(cosh(m*(L-xa)) + hmk*sinh(m*(L-xa)))/bot;
%
%       *****************
%       *  FD Solution  *
%       *****************
%
%   Setup and solve matrix eqn for discrete temperatures
      N = input('Enter number of unknowns (N) for problem: ');
      dx = L/N;    dx2 = dx*dx;   xn = dx:dx:L;   xn = xn';
      A = zeros(N,N);     b = zeros(N,1);
%   node 1
      A(1,1) = -(2 + m2*dx2);  A(1,2) = 1;  b(1) = -m2*dx2*Tinf - Tb;
%   interior nodes
      for i = 2:N-1
        A(i,i-1) = 1;  A(i,i) = -(2 + m2*dx2);  A(i,i+1) = 1;  b(i) = -m2*dx2*Tinf;
      end
%   last node
      A(N,N-2) = 1;  A(N,N) = -(2*m2*dx2 + 1 + 2*h*dx/k);
      b(N) = -(2*h*dx/k + 2*m2*dx2)*Tinf;
%   solve
      Tn = A\b;
%   add on the known base temperature to the solution vector
```

```
      xn = [0; xn];   Tn = [Tb; Tn];
%
%       *************************
%       *  Plot both solutions  *
%       *************************
%
      nfig = nfig+1;   figure(nfig)
      plot(xa*100,Ta,'r-',xn*100,Tn,'bo','LineWidth',2),grid
      title('RECT1D\_FIN\_2: T(x) Profile in Rectangular Fin')
      xlabel('Spatial Position (cm)')
      ylabel('Temperature (^oC)')
      legend('Analytical Solution',['FD Solution (N = ',num2str(N),')'])
%
%   end of problem
```



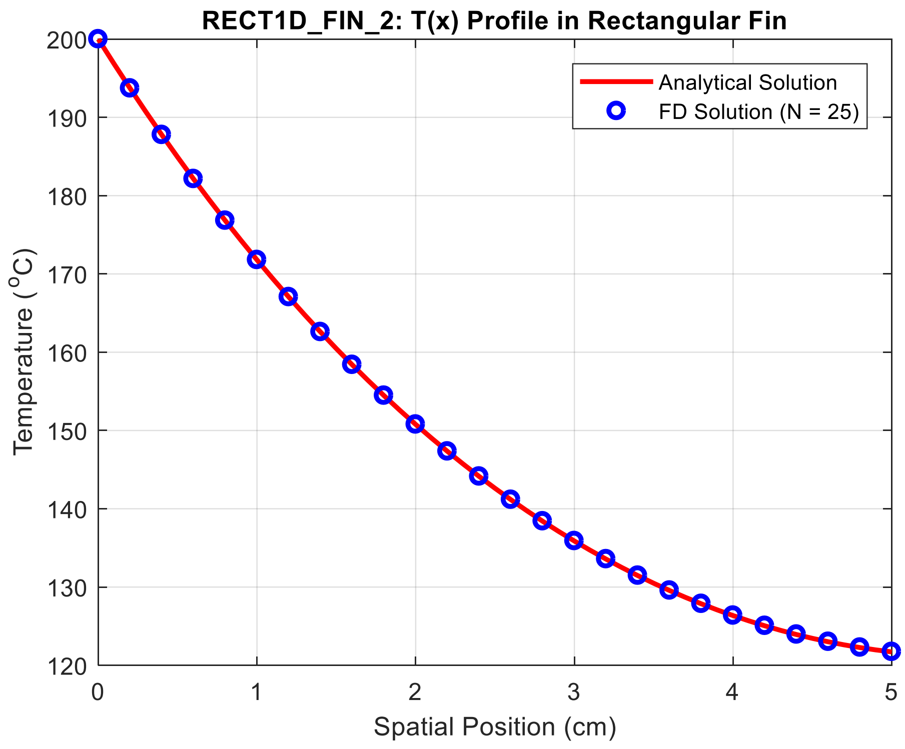


Fig. 2 Fin temperature profile -- comparison of the analytical and FD methods.

Note that, in the final step, just before plotting the FD solution, we include the known boundary temperature, T_b, at x = 0 (the fin's base) into the solution vector. Recall that this point was excluded in the FD solution, since it was known. However, for visualization, it is important to include the full domain of interest, including any boundary temperatures that may be known a priori. This simple procedure then allows us to plot the temperature throughout the fin (including the fixed base temperature).

The results from the **rect1d_fin_2.m** program are summarized in Fig. 2, which compares the temperature profile obtained from the FD method with the analytical result. This was done with two different step sizes (a different N leads to a different Δx) and, as apparent from the comparisons given here, this problem is not very sensitive to the choice of mesh spacing -- since even a relatively coarse mesh (N = 5) gives a good FD approximation for the temperature distribution in the fin.

Well, it is time to bring this brief introduction to Finite Difference (FD) methods for solving differential equations to closure. The two examples given illustrate the overall basic FD approach, and they also highlight the difference between initial value problems (IVPs) and boundary value problems (BVPs). The IVP leads to a simple recurrence relation because enough initial condition information is available to compute the dependent variable at node i+1, y_{i+1}, in terms of known values at two previous nodes, y_i and y_{i-1}. This can be represented mathematically as

$$y_{i+1} = f(y_i, y_{i-1}) \tag{25}$$

for a 2nd order difference equation. Since two initial conditions are needed for a 2nd order IVP, we have enough information to compute y_1 and y_2 to start the recursive expression given above.

Now, for a 1-D 2nd order BVP, there is only one condition given at each end point -- which means that we do not have sufficient information to get the recursive algorithm given by eqn. (25) started. Thus, for BVPs, the 2nd order difference equation is usually written in the following form,

$$f(y_{i-1}, y_i, y_{i+1}) = 0 \tag{26}$$

and, since there are N equations of this type (one for each node in the system), the result is a system of N equations with N unknowns -- which must be solved simultaneously.

Thus, the BVP is a much more challenging problem computationally, since it is not uncommon to have a very large value for N in practical engineering applications ($N > 10$ million is fairly routine in multidimensional problems). Thus, there is a real need to have the capability to solve very large systems of coupled algebraic equations (this will motivate our studies in Lesson 6). And, since we are always interested in solving more and more complex models, we are always pushing the computer's ability to the limit. As the computational power of the available computer hardware and software continues to grow, so will our ability to solve larger, and more complex problems. Today, we routinely solve problems on the PC that were not even possible on large mainframe computers 20 years ago -- and the available power just continues to increase! Who knows what will be possible in another 20 years…