Solving a Traditional Shell and Tube Heat Exchanger Problem

A Computer Project Applying the Ability to Numerically Solve Systems of Partial Differential Equations

Advanced Engineering Mathermatics
ChE 505
Department of Chemical Engineering
University of Tennessee
Knoxville, TN

Project Designed by:
Dr. David Keffer

Solution Submitted by:
Austin Newman
12/5/01

Table of Contents

Lists of Tables and Figures

Introduction

Engineers have studied heat transfer phenomena for many years. There are many resources available describing models which define heat transfer phenomena. In the study of heat exchangers, there is a certain method available which describes the heat transfer rate in a 1-1 heat exchanger with 1 shell pass and one tube pass. However, in this method there is a specific term used which describes an "average temperature" over the entire length of the apparatus. While the derivation of this "log mean temperature" is straightforward, it is desired to make a comparison of the log mean temperature to another method of solving the same problem. Therefore, a numerical experiment has been designed which will attempt to describe the accuracy of the log mean temperature. In this experiment, partial differential equations will be used to model a 1-1 shell and tube heat exchanger. Using the partial differential equations, the system will be modeled and the inlet and outlet temperatures of the heat exchanger will be determined. Additionally, the heat transfer rate will be determined from the energy balances. Next, the system will be analyzed utilizing the traditional method, which utilizes the log mean temperature. A comparison will then be made of the heat transfer rates obtained from both methods.

Assumptions

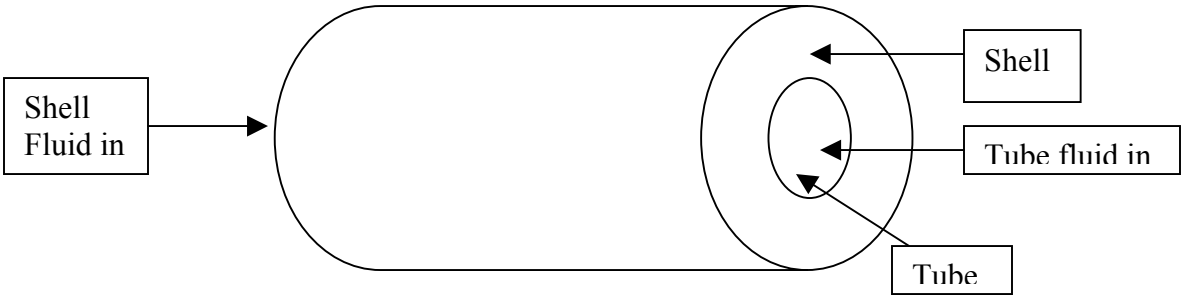
Many assumptions have been made in order to simplify this numerical experiment. These assumptions include:

- Counterflow through the heat exchanger
- The outer tube, or shell, is perfectly insulated
- Water is flowing through both tubes and will be considered an incompressible fluid
- No phase change occurs in this model
- The water is a liquid in both tubes
- The water has a constant density throughout the system
- U, the overall heat transfer coefficient, is constant throughout the system
- The heat capacity is not a function of temperature

System Parameters

The system being modeled is a simple shell and tube heat exchanger (Figure 1.).

Figure 1. Diagram of a shell and tube counterflow heat exchanger



It is important to note fluid flow in the tube occurs in the opposite direction of the fluid flow in the shell.

The operating parameters of the heat exchanger are:

The length, L, of the apparatus is 4 m and the overall heat transfer coefficient, U, is 1700 W/m^2 K.

Table 1. Heat exchanger operating parameters

Parameter	Tube	Shell
T_{in}	498.15 K	298.15 K
ρ	1001 kg/m^3	1001 kg/m^3
C_p	4184 J/kg K	4184 J/kg K
V	0.01 m/s	0.01 m/s
D	1 m	2 m

Background: Shell and Tube Heat Exchanger (traditional solution)
In order to determine the amount of heat transfer occurring in tube, one may be inclined to utilize the relationship:

$$q = UA\Delta T \quad (1)$$

where q is the heat transfer rate, U is the overall heat transfer coefficient, A is the area the heat transfer is occurring over, and ΔT is:

$$\Delta T = T_{in} - T_{out} \quad (2)$$

for either the tube or the shell. However, in utilizing this relationship, an assumption is made that the temperature drop is constant throughout the entire apparatus. In actuality, the temperature drop actually varies with position over the heat exchanger. While this may provide a good relationship, engineers are constantly seeking out better ways to describe natural phenomena. Therefore, a mean temperature must be used to better approximate the heat transfer occurring over the length of the apparatus. Conducting a heat balance and solving a differential equation yields:

$$\Delta T_{lm} = \frac{\Delta T_2 - \Delta T_1}{\ln\left(\frac{\Delta T_2}{\Delta T_1}\right)} \quad (3)$$

where ΔT_{lm} is the log mean temperature. Now, the log mean temperature can be substituted into equation (1) and will yield:

$$q = UA\Delta T_{lm} \quad (4)$$

Equation (4) is valid for a 1-1 exchanger with 1 shell pass and 1 tube pass in parallel or counterflow.

Background: Shell and Tube Heat Exchanger (partial differential equations)
In the analysis of a heat exchanger, or any heat transfer problem, one must begin with an energy balance. Typically, an energy balance has the form:

$$\text{accumulation} = \text{in} - \text{out} + \text{generation} - \text{consumption} \quad (5)$$

However, in this specific application, equation (5) will reduce to:

$$\text{accumulation} = \text{in} - \text{out} + \text{generation} \quad (6)$$

Now, an energy balance must be written specifically for this system. In this system, an enthalpy balance will be conducted. The initial equation which is written is as follows:

$$\rho V \frac{\partial H}{\partial T} = H\dot{m}\big|_x - H\dot{m}\big|_{x+\Delta x} - UA\left(T_{in} - T_{out}\right) \quad (7)$$

In order to continue the derivation, V, $\dot{m}$, and A will be replaced with the appropriate terms for this physical system which will result in:

$$\rho\pi r^2 \Delta x \frac{\partial H}{\partial T} = H\rho v\pi r^2\big|_x - H\rho v\pi r^2\big|_{x+\Delta x} - U2\pi r\Delta x\left(T_{in} - T_{out}\right) \quad (8)$$

The next step is to divide the equation by $\rho\pi r^2\Delta x$ which will result in:

$$\frac{\partial H}{\partial T} = \frac{Hv}{\Delta x}\Big|_x - \frac{Hv}{\Delta x}\Big|_{x+\Delta x} - \frac{U2(T_{in} - T_{out})}{\rho r} \quad (9)$$

The above equation will simplify to:

$$\frac{\partial H}{\partial T} = v\left(\frac{H_x - H_{x+\Delta x}}{\Delta x}\right) - \frac{U2(T_{in} - T_{out})}{\rho r} \quad (10)$$

where

$$-\left(\frac{H_x - H_{x+\Delta x}}{\Delta x}\right) \quad (11)$$

is the definition of the derivative of H. Using this relationship, equation (10) can be rewritten as:

$$\frac{\partial H}{\partial T} = -v\frac{\partial H}{\partial x} - \frac{U2(T_{in} - T_{out})}{\rho r} \quad (12)$$

In working with a real system, enthalpy is usually not the most convenient property to use. A very easy property to measure is temperature. Luckily, thermodynamics provides a convenient approximation for enthalpy as a function of temperature, which can be defined as:

$$\hat{H}(T) = \hat{C}_p T \quad (13)$$

Equation (13) can be substituted into equation (12) to produce:

$$\hat{C}_p \frac{\partial T}{\partial T} = -v\hat{C}_p \frac{\partial T}{\partial x} - \frac{U2(T_{in} - T_{out})}{\rho r} \quad (14)$$

Next, for convenience, the equation is divided by $\hat{C}_p$ to produce:

$$\frac{\partial T}{\partial T} = -v\frac{\partial T}{\partial x} - \frac{U2\pi(T_{in} - T_{out})}{\rho\hat{C}_p r} \quad (15)$$

The equation for the shell will differ slightly from the equation for the tube. The two main differences are:

1. The term V will be different
2. The generation term will be positive because the shell is gaining heat

$$\rho\pi(r_{shell}^2 - r_{tube}^2)\Delta x\frac{\partial H}{\partial T} = H\rho v\pi(r_{shell}^2 - r_{tube}^2)\big|_x + H\rho v\pi(r_{shell}^2 - r_{tube}^2)\big|_{x+\Delta x} + U2\pi r_{tube}\Delta x(T_{in} - T_{out})$$

(16)

The rest of the derivation will follow the same steps as outlined above and will result in:

$$\frac{\partial T}{\partial T} = -v\frac{\partial T}{\partial x} + \frac{U2\pi r_{tube}(T_{in} - T_{out})}{\hat{C}_p\rho(r_{shell}^2 - r_{tube}^2)} \quad (17)$$

The heat transfer in the heat exchanger can now be described by two partial differential equations (equations (15) and (17)). In order to solve partial differential equations, there must be an initial condition and two boundary conditions for each equation. For the tube the conditions are:

Initial Condition
T(z=0 to z=4) = 298.15 K (18)

Tube Boundary Conditions
T(x=0,t) = 473.15 K (19)
T(x=4,t) = $\frac{dT}{dt} = 0$ (20)

The shell can be described as:
Initial Condition:
T(x=0 to x=4) = 298.15 K (21)

Shell Boundary Conditions
T(x=0,t) = $\frac{dT}{dt} = 0$ (22)
T(x=4,t) = 298.15 K (23)

In order to solve the differential equations, it would be best to utilize a numerical technique. Dr. David Keffer has written a routine utilizing MATLAB that will solve a system of partial differential equations utilizing Runge-Kutte. The above equations along with the boundary conditions where input into this MATLAB routine in order to solve the equations.

After obtaining the temperature profile, an energy balance will be conducted in order to determine the total amount of heat transfer occurring in the heat exchanger.

Results
Utilizing MATLAB, a solution was obtained for the heat exchanger system. However, one must know what to look for in the solution. MATLAB cannot describe when the solution reaches a "steady state" operation. While running the code, the user must specify the length of time the heat exchanger is to operate. In doing so, the user must use an iterative process to determine when the system has reached a steady state. After 350 seconds of operation, one can observe the temperature profile in Figure 2. After 800 seconds of operation (Figure 3), the temperature profile appears to be very similar to that of the steady state temperature profile (Figure 4). However, at 800 seconds, the outlet temperatures have not yet reached a steady state. After reaching steady state, the outlet temperatures of each of the streams were determined as:

$T_{Tube\ Out}$ = 394.07 K
$T_{Shell\ Out}$ = 324.61 K

The total amount of heat transfer was calculated as

q_{Total} = 2636 kW

Figure 2. Temperature profile after 350 seconds of operation

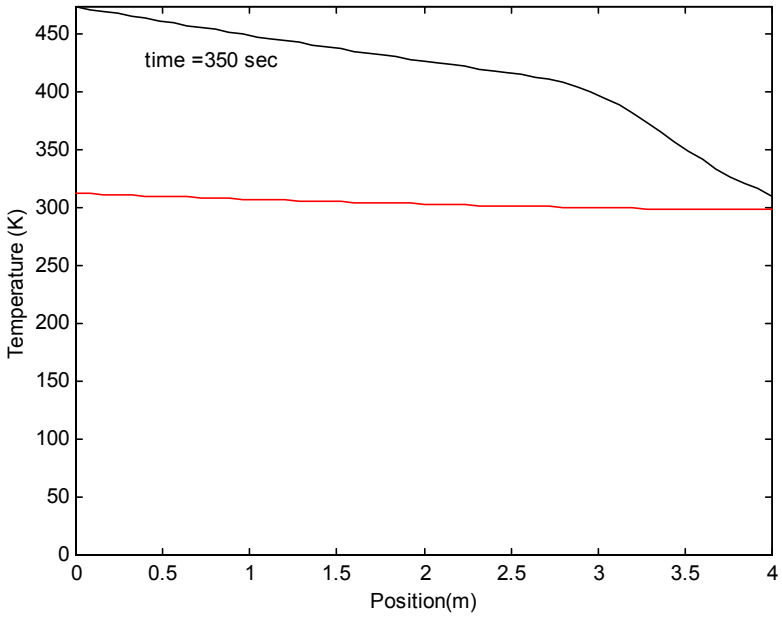


Figure 3. Temperature profile after 800 seconds of operation

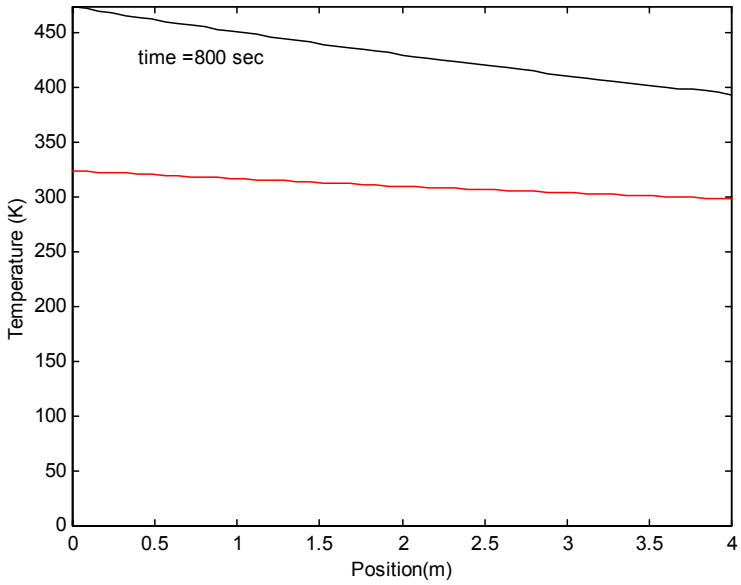
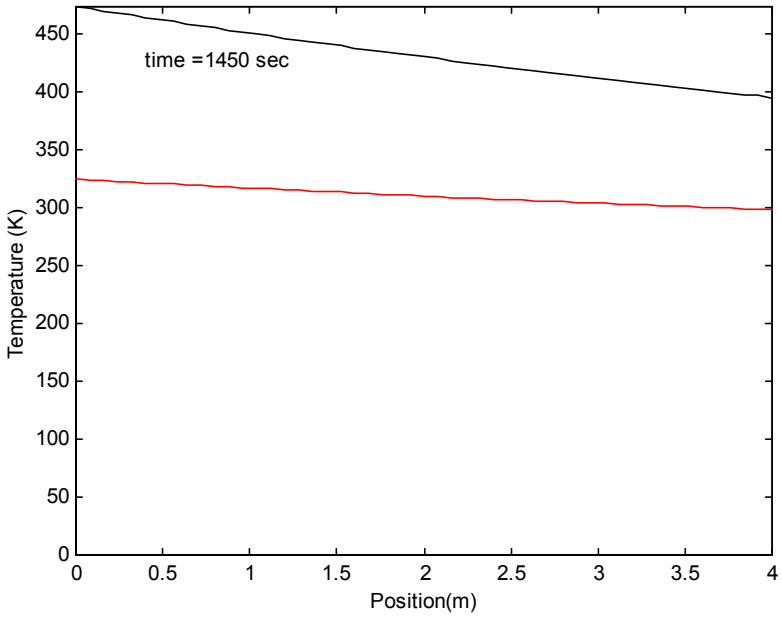


Figure 4. Steady State Temperature Profile of Heat Exchanger



The heat transfer was calculated by using the relationship:

$$q_{Total} = U 2\pi r \Delta x \left(T_{Tube} - T_{Shell}\right) \quad (18)$$

When utilizing numerical techniques to solve partial differential equations, the time and space dimensions are discretized into a grid. A distance of Δx separates each of the temperature intervals. Therefore, when the final solution to the system of equations is obtained, the temperature profile is available as a function of distance over the entire apparatus. In calculating the total amount of heat transfer, the temperature difference was calculated at each node of the grid. Each of the local temperature differences was then utilized to calculate the heat transfer occurring at that point of the apparatus. Each of the local heat transfer values was then summed over the entire length of the apparatus to obtain the total amount of heat transfer taking place.

Now that the outlet temperatures are known, the traditional method (utilizing the log mean temperature) was utilized to solve for the total amount of heat transfer. The amount of heat transfer calculated utilizing this method was:

q_{Total} = 2572 kW

Discussion of Results

Ideally, experimental values from an actual heat exchanger would be available so a comparison could be made to actual values. Unfortunately, experimental values are not available for the heat exchanger described here. If actual results were available, further experimentation could be conducted in solving the partial differential equations. The spatial grid and time grid parameters could be optimized to replicate the results of a real

system. Of course, in optimizing these parameters, there would be a sacrifice in the amount of time the code takes to run. As the number of points in the grid increases, the computation time also increases.

However, it is still interesting as to how close the traditional approximation was to the approximation obtained utilizing the partial differential equations. In reviewing the results obtained from each method, it is comforting to know they are within 4% of each other.

Many simplifications were made in this model. In the future, it would be very interesting to refine it at an attempt of obtaining more accurate results. In doing this, a good place to start would be with the value of the overall heat transfer coefficient. While it is convenient to assume this coefficient is constant, it usually varies over the length of the heat exchanger. A more accurate model would account for the changes which occur with U along the length of the heat exchanger.

Discussion of Numerical Techniques

There were several difficulties encountered with solving this particular system of partial differential equations utilizing numerical techniques. In determining the solution of this system of equations, a Runge-Kutta second order technique was utilized. Runge-Kutta is very robust and very efficient (compared to other numerical techniques) which makes it a very popular method to solve differential equations. However, the particular system of equations encountered here, was very sensitive to the parameters utilized in determining the numerical solution. In order to add stability to this technique a " thermal diffusion" term was added to each of the heat balance equations which gives:

$$\frac{\partial T}{\partial T} = -v\frac{\partial T}{\partial x} - \frac{U2\pi(T_{in} - T_{out})}{\rho\hat{C}_p r} + D\frac{\partial^2 T}{dx^2} \quad (19)$$

and

$$\frac{\partial T}{\partial T} = -v\frac{\partial T}{\partial x} + \frac{U2\pi r_{tube}(T_{in} - T_{out})}{\hat{C}_p \rho(r_{shell}^2 - r_{tube}^2)} + D\frac{\partial^2 T}{\partial x^2} \quad (20)$$

In actuality, there is actually thermal diffusion occurring in this process. However, to stabilize the solution, a very large diffusion coefficient had to be used. An actual diffusion coefficient would be approximately 2.39E-7. In order to stabilize the solution, a coeffecient of 2.39E-4 had to be utilized.

Another method of stabilizing the solution would be to use a finer mesh. However, as the size of the mesh increases, the computation time greatly increases. Experiments were conducted in utilizing a finer mesh. For the purpose of this experiment, a decision was made to utilize an "unreal" thermal diffusivity value. However, if further work was conducted on this system, especially, a comparison to an "actual" heat exchanger, it

would be necessary to utilize a finer mesh to better model the heat exchanger. It would also be suggested to utilize a programming language which is more efficient than MATLAB. It is believed utilizing FORTRAN would greatly reduce the amount of time required to solve this problem.

Appendix I. Nomenclature

ρ	Density [kg/m^3]
ΔT	Change in Temperature [K]
ΔT	Log Mean Temperautre [K]
V	Volume [m^3]
A	Area [m^2]
q	Heat Transfer Rate [W]
U	Overall Heat Transfer Coefficient [W/m^2 K]
T	Temperature [K]
D	Diameter [m]
v	Velocity of stream [m/s]
$\dot{m}$	Mass Flow Rate [kg/s]
$\hat{C}_p$	Heat Capacity [J/kg K]
H	Enthalpy [kJ/kg]
x	Spatial Coordinate [m]
r	Radius [m]

Acknowledgements
This project was assigned by Dr. David Keffer, University of Tennesee, Knoxville for Chemical Engineering 505, Advanced Numerical Techniques. While there was a "standard class project" available, Dr. Keffer, offered students the flexibility to design their own project. As I have always been interested in heat transfer phenomena, I requested permission to study heat transfer phenomena for this project. Dr. Keffer suggested determining the total heat transfer occurring in a shell and tube heat exchanger utilizing differential equations. He thought it would be interesting to compare this calculation to that utilizing the log mean temperature difference.

I would like to thank Dr. Keffer for offering me the opportunity to complete this project. He spent extra time in getting me off on the right foot with the derivation of the energy balances. Additional time was also spent in debugging the software because of the problems mentioned in the "Discussion of Numerical Techniques" section. At times, completion of this project was very frustrating. In the end however, I thoroughly enjoyed working on a project in which I was interested in the subject matter.

References

Geankopolis, Christie, *Transport Processes and Unit Operations,* Third Edition, Prentice Hall, 1993.

Keffer, David, *Application and Solution of the Heat Equation in One-and Two-Dimensional Systems Using Numerical Methods*, http://clausius.engr.utk.edu (under Courses: ChE 240, March, 1999.

Frank P. Incropera, David P. Dewitt, *Fundamentals of Heat and Mass Transfer*, Third Edition, Wiley, 1990.

Appendix II. MATLAB Code

```
function syspde_para
clear all;
close all;
% n_eq = number of equations
n_eq = 2; %INPUT

% m_int = number of spatial intervals
m_int = 50; %INPUT
xo = 0.0;
xf = 4.0;
n_int = 1000;
%n_int = 100;
to = 0.0;
tf = 1450;
%  BC is a vector of boundary condition type definitions
%  0 = Dirichlet
%  1 = Neumann
BC = [0 1; 1 0; 0 1; 1 0]; %INPUT

%  code must create files
% function y_out = syspde_boundary_conditions(k_eq,k_bc,x,t,y1,i_x,
dx);
% function y_out = syspde_initial_conditions(k_eq,x);




%
%  create spatial grid
%
%  mx = number of spatial gridpoints
%  (this grid has imaginary points for Neumann BCs)
%  (it still works if the problem has only Dirichlet BCs)
mx = m_int + 3;
dx = (xf-xo)/m_int;
dxi = 1/dx;
xgrid = [(xo-dx):dx:(xf+dx)]';
%  create temporal grid
%  nt = number of temporal gridpoints
nt = n_int + 1;
dt = (tf-to)/n_int;
tgrid = [(to):dt:(tf)]';
%
%  dimension solution matrix
%
y = zeros(n_eq,nt,mx);
%
%  define some limitations on indices for where solution is not given
by ICs and
%  where the derivatives need to be calculated using finite difference
techniques
%
for k_eq = 1:1:n_eq
```

```
   for i_BC = 1:1:2
      if (i_BC == 1)
         if (BC(k_eq, i_BC) == 0)
            i_x_v(k_eq,i_BC) = 3;
         else
            i_x_v(k_eq,i_BC) = 2;
         end
      else
         if (BC(k_eq, i_BC) == 0)
            i_x_v(k_eq,i_BC) = mx-2;
         else
            i_x_v(k_eq,i_BC) = mx-1;
         end
      end
   end
end
%i_x_v
%i_x_d
%
%  input initial conditions into solution matrix
%
j_t = 1;
for k_eq = 1:1:n_eq
   for i_x = 2:1:mx-1
      x = xgrid(i_x);
      y(k_eq,j_t,i_x) = syspde_initial_conditions(k_eq,x);
   end
end
%
%  input Dirichlet Boundary conditions into solution matrix for all
times
%
y1 = zeros(n_eq,mx);
for k_eq = 1:1:n_eq
   for k_bc = 1:1:2
      if (BC(k_eq,k_bc) == 0)
         if (k_bc == 1)
            i_x = 2;
         else
            i_x = mx - 1;
         end
         x = xgrid(i_x);
         for j_t = 1:1:nt
            t = tgrid(j_t);
            for m_eq = 1:1:n_eq
               y1(m_eq,:) = y(m_eq,j_t,:);
            end
%            fprintf(1,'D: k_eq= %i kbc= %i i_x = %i x = %f t = %f
\n',k_eq, k_bc,i_x,x,t)
            y(k_eq,j_t,i_x) =
syspde_boundary_conditions(k_eq,k_bc,x,t,y1,i_x,dx);
         end
      end
   end
end
%
%  input initial Neumann Boundary conditions into solution matrix
```

```
%
j_t = 1;
t = tgrid(j_t);
for k_eq = 1:1:n_eq
   y1(k_eq,:) = y(k_eq,j_t,:);
end
for k_eq = 1:1:n_eq
   for k_bc = 1:1:2
      if (BC(k_eq,k_bc) == 1)
         if (k_bc == 1)
            i_x = 1;
            i_x2 = 2;
            i_x3 = 3;
         else
            i_x = mx;
            i_x2 = mx-1;
            i_x3 = mx-2;
         end
         x = xgrid(i_x);
%          fprintf(1,'N: k_eq= %i kbc= %i i_x = %i x = %f t = %f
\n',k_eq, k_bc,i_x,x,t)
         y(k_eq,j_t,i_x) =
syspde_boundary_conditions(k_eq,k_bc,x,t,y1,i_x,dx);
      end
   end
end
%
%  call Classical second order Runge-Kutta method
%
dydt_1 = zeros(n_eq,mx);
dydt_2 = zeros(n_eq,mx);
for j_t = 2:1:nt
   %  obtain first half of approximation to derivative
   t = tgrid(j_t-1);
   for k_eq = 1:1:n_eq
      y1(k_eq,:) = y(k_eq,j_t-1,:);
   end
   dydt_1 = syspde_para_input(n_eq,i_x_v,y1,t,dxi,mx);
   %  obtain second half of approximation to derivative
   t = t + dt;
      %  first obtain intermediate values of y in interior nodes
   for k_eq = 1:1:n_eq
      for i_x = i_x_v(k_eq,1):1:i_x_v(k_eq,2)
         y1(k_eq,i_x) = y1(k_eq,i_x) + dt*dydt_1(k_eq,i_x);
      end
   end
   %  second obtain intermediate values of y from Dirichlet BCs
   for k_eq = 1:1:n_eq
      for k_bc = 1:1:2
         if (BC(k_eq,k_bc) == 0)
            if (k_bc == 1)
               i_x = 2;
            else
               i_x = mx-1;
            end
            x = xgrid(i_x);
            y1(k_eq,i_x) = y(k_eq,j_t,i_x);
```

```
         end
      end
   end
   % third obtain intermediate values of y from Neumann BC's
   t = tgrid(j_t);
   for k_eq = 1:1:n_eq
      for k_bc = 1:1:2
         if (BC(k_eq,k_bc) == 1)
            if (k_bc == 1)
               i_x = 1;
            else
               i_x = mx;
            end
            x = xgrid(i_x);
            y1(k_eq,i_x) =
syspde_boundary_conditions(k_eq,k_bc,x,t,y1,i_x,dx);
         end
      end
   end
   dydt_2 = syspde_para_input(n_eq,i_x_v,y1,t,dxi,mx);
   for k_eq = 1:1:n_eq
      for i_x = 1:1:mx
         y(k_eq,j_t,i_x) = y(k_eq,j_t-1,i_x) + 0.5*dt*(dydt_1(k_eq,i_x)
+ dydt_2(k_eq,i_x) );
      end
   end
   % compute new values of unknown from Neumann BC's
   t = tgrid(j_t);
   for k_eq = 1:1:n_eq
      for k_bc = 1:1:2
         if (BC(k_eq,k_bc) == 1)
            if (k_bc == 1)
               i_x = 1;
            else
               i_x = mx;
            end
            x = xgrid(i_x);
            for m_eq = 1:1:n_eq
               y1(m_eq,:) = y(m_eq,j_t,:);
            end
            y(k_eq,j_t,i_x) =
syspde_boundary_conditions(k_eq,k_bc,x,t,y1,i_x,dx);
         end
      end
   end
end
%
%  write solution
%
%y
%
%  plot solution
%
% define plot/movie parameters
T=y;
nframe = 40;
if (n_int > nframe)
```

```
   nskip = round(n_int/nframe);
else
   nskip = 1;
end
lplot = 1;
fps = 3;
%  define plot line colors (for up to 10 eqns)
plc{1} = 'k-';
plc{2} = 'r-';
plc{3} = 'b-';
plc{4} = 'g-';
plc{5} = 'm-';
plc{6} = 'k:';
plc{7} = 'r:';
plc{8} = 'b:';
plc{9} = 'g:';
plc{10} = 'm:';
if (lplot == 1)
   fpsinv = 1.0/fps;
   newplot;
   Tmax=max(max(max(T)));
   Tmin=min(min(min(T)));
   xtext = xo + 0.1*(xf-xo);
   ytext = Tmin +0.9*(Tmax-Tmin);
   for j_t=1:nskip:nt
      for k_eq = 1:1:n_eq
         for i_x = 2:1:mx-1
            vector_plot(i_x) = T(k_eq,j_t,i_x);
         end
         plot(xgrid(2:mx-1) , vector_plot(2:mx-1),plc{k_eq} );
         hold on

      end
      axis([xo xf Tmin Tmax]);
      XLABEL('Position(m)')
      YLABEL('Temperature (K)')
      temps = char(strcat('time = ', num2str(tgrid(j_t)), ' sec '));
      text(xtext, ytext,temps)
      hold off
      pause(fpsinv);
   end
end
%Output the final outlet temperatures
for ii=1:n_eq
   fprintf('Inlet/Outlet Temperaure for %2.0f equation is %6.2f and
%6.2f \n',ii, y1(ii,2),y1(ii,mx-1));
end
%Calculate the amount of heat transferred utilizing the spatial grid
qdiff=0;
rho_tube = 1000.1;
v_tube = 1e-2;
Cp_tube = 1000.0*4.184;
%System variables from syspde_para_input
h_tran_coeff = 1700;
r_tube = 0.5;
Asurf = 2.0*pi*r_tube*dx;
for ii=2:mx-1
```

```
   qdiff = h_tran_coeff*Asurf*(y1(1,ii)-y1(2,ii)) + qdiff;
end

%Calculate the amount of heat transferred utilizing the log mean
temperature
Asurf = 2.0*pi*r_tube*4;
delT1 = y1(1,2) - y1(2,2);
delT2 = y1(1,mx-1) - y1(2,mx-1);
Tlogmean = (delT1-delT2)/(log(delT1/delT2));
qlogmean= h_tran_coeff*Asurf*Tlogmean;

fprintf('The total heat transfer utilizing differential equations is
%12.2f \n',qdiff);
fprintf('The total heat transfer utilizing the log mean temperature is
%12.2f \n',qlogmean);



function y_out = syspde_boundary_conditions(k_eq,k_bc,x,t,y1,i_x, dx);
if (k_bc == 1)
   if (k_eq == 1)
      y_out = 473.15;
   elseif (k_eq == 2)
      y_out = y1(k_eq, i_x+2)- 2*dx*(0.0);
   elseif (k_eq == 3)
      y_out = 2;
   else
      y_out = y1(k_eq, i_x+2)- 2*dx*(-1.0);
   end
else
   if (k_eq == 1)
      y_out = y1(k_eq, i_x-2)- 2*dx*(0.0);
   elseif (k_eq == 2)
      y_out = 298;
   elseif (k_eq == 3)
      y_out = y1(k_eq, i_x-2)- 2*dx*(0.0);
   else
      y_out = 3;
   end
end
%fprintf(1,'BC: k_eq= %i kbc= %i i_x = %i x = %f t = %f yout = %f
\n',k_eq, k_bc,i_x,x,t,y_out)

function y_out = syspde_initial_conditions(k_eq,x);
if (k_eq == 1)
   y_out = 298;
elseif (k_eq == 2)
   y_out = 298;
elseif (k_eq == 3)
   y_out = 300;
else
   y_out =  100 + 100*(2-x)/(2);
end
```

```
function dydt = syspde_para_input(n_eq,i_x_v,y1,t,dxi,mx)
% evaluate first and second spatial derivatives for all variables
for k_eq = 1:1:n_eq
   for i_x = i_x_v(k_eq,1):1:i_x_v(k_eq,2)
      dydx_matrix(k_eq, i_x) = 0.5*( y1(k_eq,i_x+1) - y1(k_eq,i_x-1)
)*dxi;
      d2ydx2_matrix(k_eq,i_x) = ( y1(k_eq,i_x+1) - 2.0*y1(k_eq,i_x) +
y1(k_eq,i_x-1) )*dxi^2;
   end
end
%  evaluate temporal derivative for all equations at interior nodes
dydt = zeros(n_eq,mx);
%  input heat transfer parameters
      % L = length of rod
      L = 4.0;
      % for copper
      % rho = density [kg/m^3]
      rho_tube = 1000.1;
      rho_shell = 1000.1;
      % Cp = heat capacity [J/kg/K]
      Cp_tube = 1000.0*4.184; % [J/kg]
      Cp_shell = 1000.0*4.184; % [J/kg]
      % k = thermal conductivity [W/m/K]
      k_tube = 1000;
      k_shell = k_tube;
      %  alpha = thermal diffusivity
      alpha_tube = k_tube/rho_tube/Cp_tube;
      alpha_shell = k_shell/rho_shell/Cp_shell;
      %  heat transfer coefficient in [W/m^2/K]
      h_tran_coeff = 1700;
      % tube radius [m]
      %r_tube = 1.0;
      r_tube = 0.5;
      %r_shell = sqrt(2.0)*r_tube;
      r_shell = 1;
      % diameter in [m]
      d_tube = 2.0*r_tube;
      d_shell = 2.0*r_shell;
      % Area in [m^2]
      A_tube = pi*(r_tube^2);
      A_shell = pi*(r_shell^2 - r_tube^2);
      % Volume in [m^3]
      V_tube = A_tube*L;
      V_shell = A_shell*L;
      %Area of the heat transfer surface
      Asurf = 2.0*pi*r_tube*L;
      %  Temperature of the surroundings in Kelvin
      Tsurround = 200.0; %[K]
      %factor = h_tran_coeff*Area/(rho*Cp*V);
      afac_tube = h_tran_coeff*Asurf/(rho_tube*Cp_tube*V_tube);
      afac_shell = h_tran_coeff*Asurf/(rho_shell*Cp_shell*V_shell);
      ffac_tube = afac_tube*Tsurround;
      ffac_shell = afac_shell*Tsurround;
      % Velocity of Fluid
      v_tube = 1.0e-2;
      v_shell = -v_tube;
% for k_eq = 1
```

```
%  perfectly insulated rod
k_eq = 1;
   for i_x = i_x_v(k_eq,1):1:i_x_v(k_eq,2)
      dydt(k_eq,i_x) =  -v_tube*dydx_matrix(k_eq,i_x) +
alpha_tube*d2ydx2_matrix(k_eq,i_x) ...
         - afac_tube*(y1(1,i_x) - y1(2,i_x)) ;
   end
 k_eq = 2;
   for i_x = i_x_v(k_eq,1):1:i_x_v(k_eq,2)
      dydt(k_eq,i_x) =  -v_shell*dydx_matrix(k_eq,i_x) +
alpha_shell*d2ydx2_matrix(k_eq,i_x) ...
         + afac_shell*(y1(1,i_x) - y1(2,i_x)) ;
   end
```