MSE 510
Final Examination Solutions
May 5, 2015

1. Single Variable Nonlinear Optimization

Download the data located at
http://utkstair.org/clausius/docs/mse510/data/mse510_xm02_p01.txt
This data represents the results of a set of experiments intended to measure the variance in the particle size of a crystallization process. The data is essentially a histogram with the first column corresponding to variance, *x*, and the second column corresponding to probability, *f*. From theory this data should follow the chi-squared distribution, given by

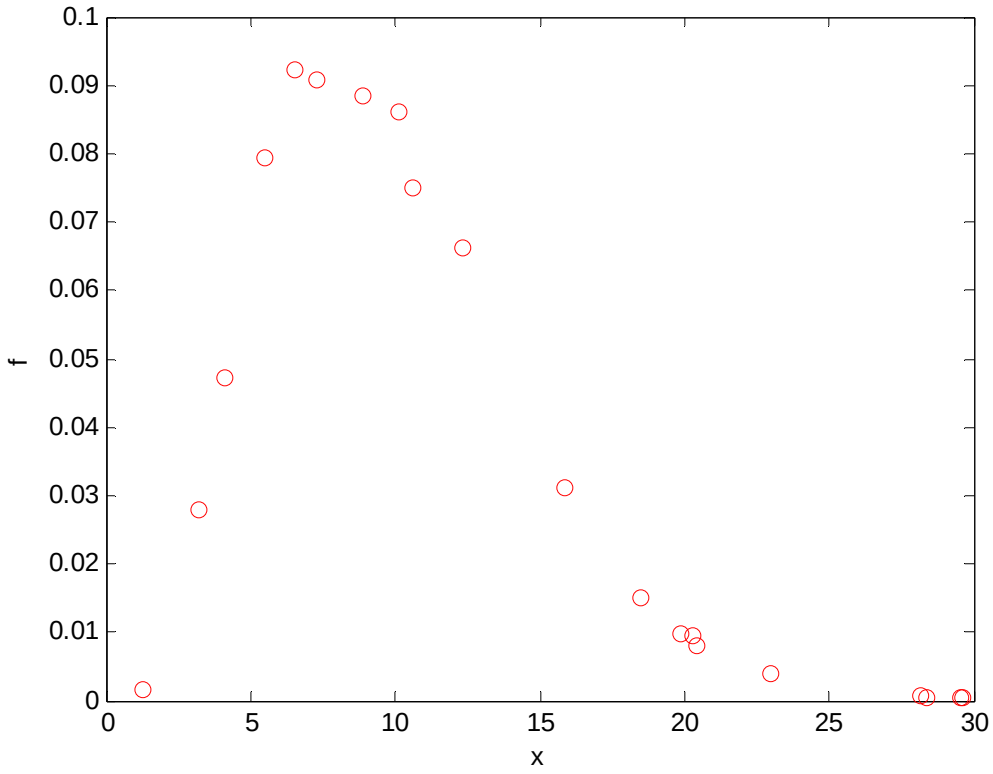
$$f_{\chi^2}(x;\nu) = \begin{cases} \dfrac{1}{2^{\nu/2}\Gamma(\nu/2)} x^{\nu/2-1} e^{-x/2} & \text{for } x > 0 \\ 0 & \text{elsewhere} \end{cases} \qquad (1)$$

The chi-squared distribution has one parameter, *v*, the degrees of freedom. ($\Gamma(x)$ is the gamma function, an intrinsic function in Matlab, gamma(x).) Perform a single variable nonlinear optimization in order to fit the data to this model. Report the optimal value of *v*. As a good initial guess, consider that the population mean of the chi-squared distribution is *v*.

Solution:

I decided to use Brent's line minimization method to solve this 1-D nonlinear optimization problem.

I first plotted the data to get a feel for its shape.



It appears to me that there is a peak about 8.

I created the following objective function, which corresponds to the sum of the square of the errors (SSE) between the data and the model in funkeval.m.

```
function f = funkeval(x);
v = x;
datamat = [    4.13607015 0.04732715
  6.53404815 0.09233015
  ...
  20.40535115 0.00804915
  15.83540515 0.03117115  ];
xdata = datamat(:,1);
fdata = datamat(:,2);
ndata = length(xdata);
fac = 1.0/(2.0^(v/2.0)*gamma(v/2.0));
for i = 1:1:ndata
    fmod(i) = fac*xdata(i)^(v/2.0-1.0)*exp(-xdata(i)/2.0);
end
f = 0.0;
for i = 1:1:ndata
    f = f + (fmod(i) - fdata(i))^2;
end
```

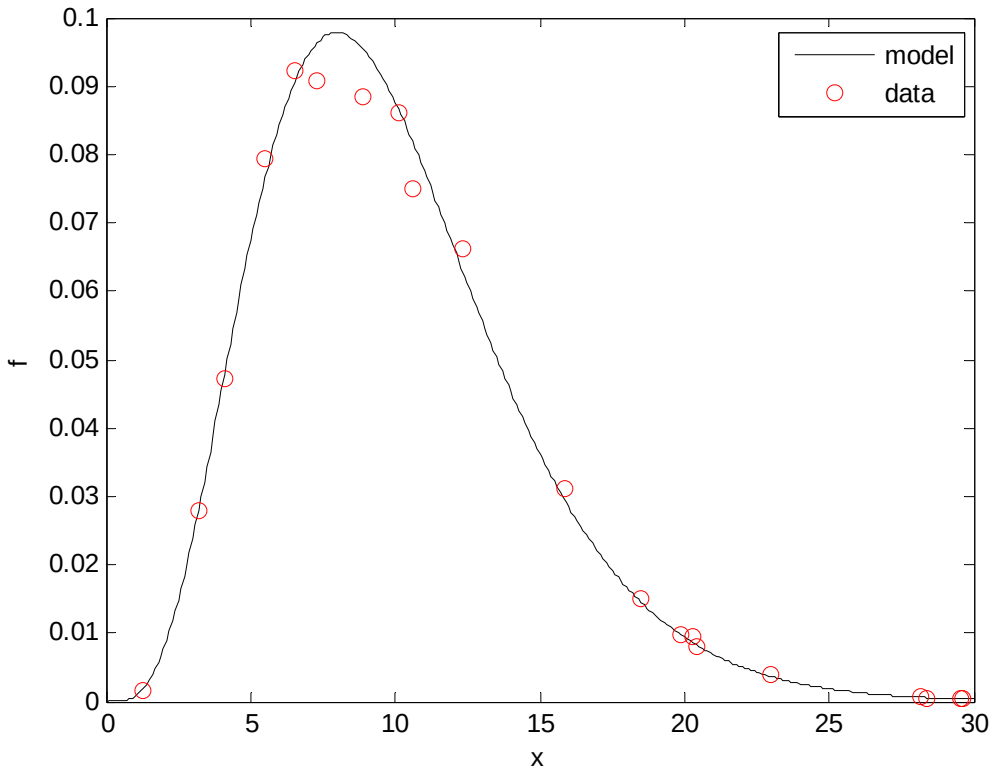
I created the brackets and called Brent's line minimization with the following two commands.

```
>> [ax, bx, cx] = mnbrak1(1,8,'min');
>> [f,vmin] = brent(ax,bx,cx,1.0e-6,1,'min');
```

This generated the following output.

```
boundary 1:  a =  9.98e+00 f(a) =  1.56e-04
boundary 2:  b =  9.98e+00 f(b) =  1.56e-04
best guess:  x =  9.98e+00 f(x) =  1.56e-04
2nd best guess:  w =  9.98e+00 f(w) =  1.56e-04
previous w:  v =  9.98e+00 f(v) =  1.56e-04
most recent point:  u =  9.98e+00 f(u) =  1.56e-04
error =  3.04e-06 iter =    28

ANSWER =  9.976226e+00
```

The method converged in 28 iterations. The values of *v* is given by 9.98. A plot (not required for the solution) of the data and the model is provided below.



2. Single Non-Linear Parabolic PDE

The one-dimensional heat equation can describe heat transfer in a material with both heat conduction and radiative heat loss.

$$\frac{\partial T}{\partial \mathrm{t}} = \frac{k}{\rho C_p}\frac{d^2T}{dz^2} - \frac{\varepsilon\sigma S}{\rho C_p}\left(T^4 - T_s^4\right)$$

where the following variables [with units] are given as

temperature in the material T [K]
surrounding temperature $T_s = 77$ [K]
axial position along material z [m]
thermal conductivity $k = 401$ [J/K/m/s] (for Cu)
mass density $\rho =$ 8960 [kg/m^3] (for Cu)
heat capacity $C_p = 384.6$ [J/kg/K] (for Cu)
Stefan–Boltzmann constant $\sigma = 5.670373x10^{-8}$ [J/s/m^2/K^4]
gray body permittivity $\varepsilon = 0.15$ (for dull Cu)
surface area to volume ratio $S = 80$ [m^{-1}] (for a cylindrical rod of diameter 0.05 m)

A cylindrical Cu rod of diameter 0.05 m and length 0.5 m is initially at $T(z, t=0) = 800$ K. One end of the rod is maintained at $T(z=0, t) = 900$ K. The other end of the rod is insulated,

$$\left.\frac{dT}{dz}\right|_{z=0.3} = 0 \text{ K/m.}$$

(a) Plot the transient behavior.
(b) Find the approximate steady-state temperature in the material at z=0.5 m.

Solution:

This is a single non-linear parabolic PDE with one spatial dimension and a Dirichlet boundary condition at z=0 and a Neumann boundary condition at z=0.5. To solve this problem, I will use the code `parapde_1_anyBC.m`.

I modified the input functions in `parapde_1_anyBC.m` as follows.

I assigned the appropriate type of boundary conditions.

```
BC(1) = 'D';
BC(2) = 'N';
```

I set the final time to 1000 seconds and chose dt to be 1.0 seconds, so I had 1,000 temporal intervals.

```
% discretize time
```

```
to = 0;
tf = 1.0e+3;
dt = 1.0e+0;
```

The rod spans from 0 to 0.5 meter. I set dx to be 0.05 m, so I had 10 spatial intervals.

```
% discretize space
xo = 0;
xf = 0.5;
dx = 5.0e-2;
```

I defined the PDE in the following function.

```
%
%  function defining PDE
%
function k = pdefunk(x,t,y,dydx,d2ydx2);
%
Temp = y;
% rho = density [kg/m^3]
rho = 8960.0;
% Cp = heat capacity [J/kg/K]
Cp = 384.6;
% k = thermal conductivity [W/m/K]
k = 401.0;
%  alpha = thermal diffusivity
alpha = k/rho/Cp;
% length of rod [m]
L = 0.5;
% diameter in [m]
radius = 0.025;
diameter = 2.0*radius;
% surface Area in [m^2]
Area = pi*diameter*L;
% Volume in [m^3]
Volume = pi/4*diameter^2*L;
% surface area to volume ratio
S = Area/Volume;
%  Temperature of the surroundings [K]
Tsurround = 77.0;
% Stefan-Boltzmann constant [J/s/m^2/K^4]
sigma = 5.670373e-8;
% gray body permittivity [dimensionless]
eps = 0.15;
fac = eps*sigma*S/(rho*Cp);
k = alpha*d2ydx2 - fac*(Temp^4 - Tsurround^4);
```

I defined the IC and BCs in the functions below.

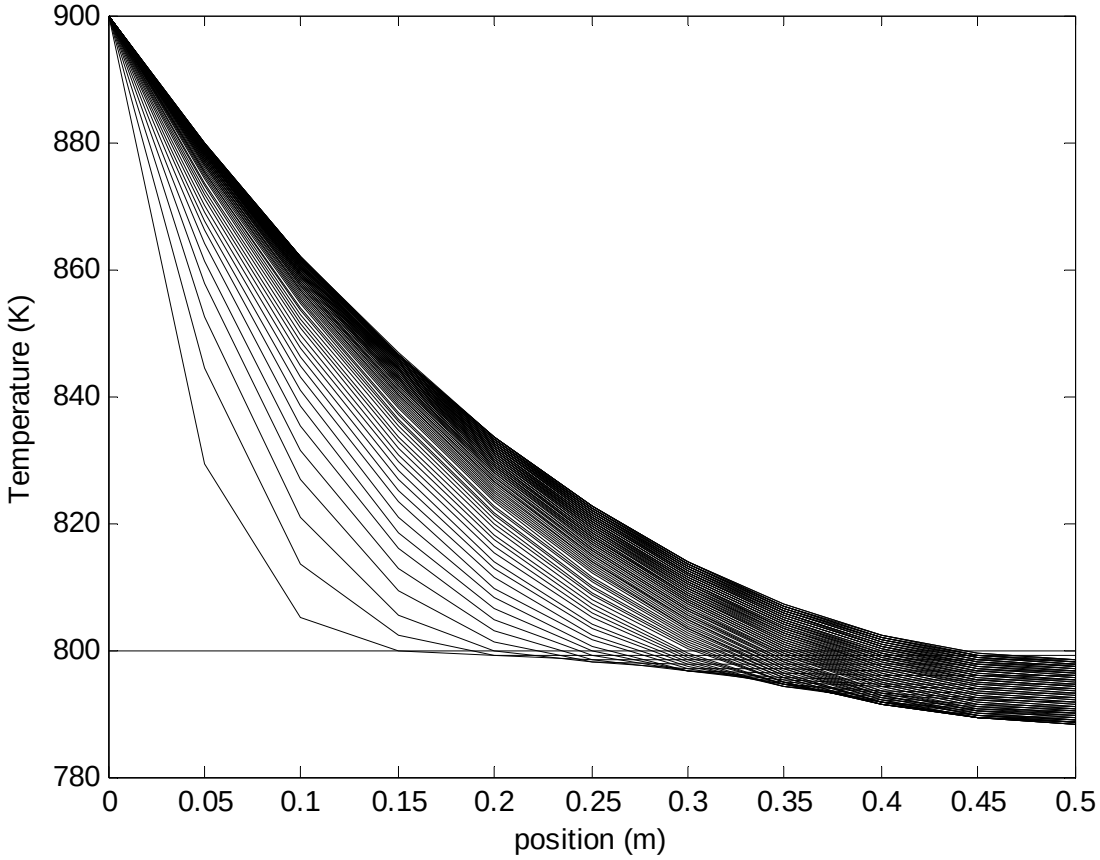
```
%
%  function defining initial condition
%

function ic = icfunk(x);
ic = 800;

%
```

```
%  functions defining LHS boundary condition
%

function f = aBCo(t);
f = 1;

function f = bBCo(t);
f = 0;

function f = cBCo(t);
f = -900;

%
%  functions defining RHS boundary condition
%

function f = aBCf(t);
f = 0;

function f = bBCf(t);
f = 1;

function f = cBCf(t);
f = 0;
```

At the command line prompt, I typed

```
[xvec,tvec,Tmat] = parapde_1_anyBC;
```

This command generated the following plot.



To find the last value at x = 0.5 m, I confirmed that I knew the correct spatial and temporal indices.

```
>> xvec(12)
ans =    0.5000

>> tvec(1001)
ans =  1000

>> Tmat(12,1001)
ans =  798.5048
```

Therefore the temperature at the end at 1000 seconds is 798.5 K.

I don't know that this is steady state. I can run the simulation longer. If I change nothing but the final time to 5000 seconds, then I generate the data point

```
>> [xvec,tvec,Tmat] = parapde_1_anyBC;
>> Tmat(12,5001)
ans =  804.1653
```

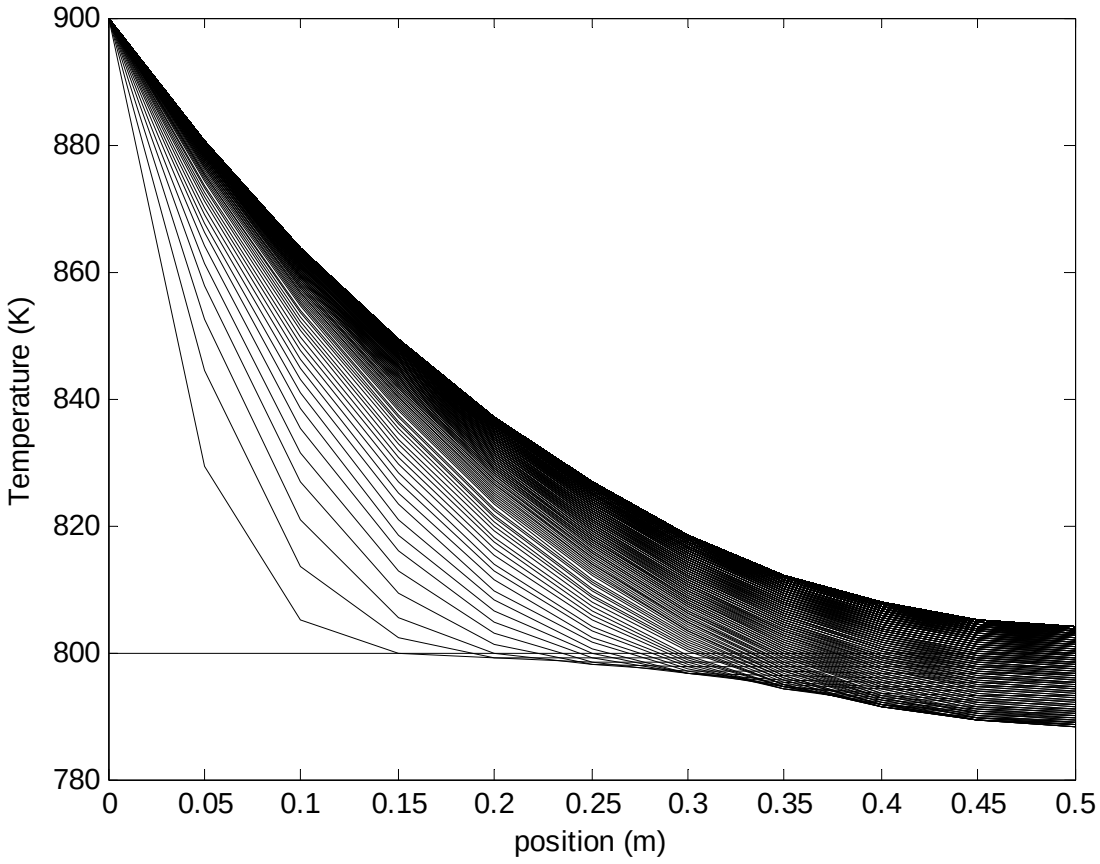
Therefore the temperature at the end at 5000 seconds is 804.2 K.

The two answers are quite different, so we are not at steady state. If I change nothing but the final time to 10,000 seconds, then I generate the data point

```
>> [xvec,tvec,Tmat] = parapde_1_anyBC;
>> Tmat(12,10001)
ans =  804.1757
```

Therefore the temperature at the end at 10,000 seconds is 804.2 K. This is pretty close to the steady state temperature.

Below is the corresponding plot.



3. ODE Boundary Value Problem

Consider the following boundary value problem, which represents the steady state profile in Problem 2.

$$0 = \frac{k}{\rho C_p}\frac{d^2T}{dz^2} - \frac{\varepsilon\sigma S}{\rho C_p}\left(T^4 - T_s^4\right)$$

with the boundary conditions

$$T(z=0) = T_o = 900 \text{ K}$$
$$\frac{dT}{dz}(z=0.5) = T_f' = 0 \text{ K/m}$$

where all of the parameters are given in problem 2.

(a) Convert this single second-order ODE, to a system of two first-order ODEs.
(b) Plot the solution.
(c) What is the temperature gradient at $z = 0$?
(d) What is the temperature at $z = 0.5$?

Solution:
(a) Convert this single second-order ODE, to a system of two first-order ODEs.

Follow the three step process. First, define new variables.

$$y_1 = T \qquad\qquad y_2 = \frac{dT}{dz}$$

Second write the ODEs in the new variables.

$$\frac{dy_1}{dz} = y_2$$

$$\frac{dy_2}{dz} = \frac{\varepsilon\sigma S}{k}\left(y_1^4 - T_s^4\right)$$

Third, write the conditions in terms of the new variables.

$$y_1(z=0) = T_o = 900$$
$$y_2(z=0.5) = T_f' = 0$$

To solve this BVP, I will use both the code for Newton-Raphson method with Numerical Derivatives for 1 equation (`nrnd1.m`) and the classical 4th-order Runge-Kutta method for N equations (`rk4n.m`).

I modified the input function in `nrnd1.m` as follows:

```
function f = funkeval(x)
xo = 0;
xf = 0.5;
yo_1 = 900;
yo_2 = x;
yf_2 = 0.0;
n = 1000;
[x,y]=rk4n(n,xo,xf,[yo_1,yo_2]);
yf_2_calc = y(n+1,2);
f = yf_2_calc-yf_2;
```

I entered the ODEs in the input file for `rk4n.m` as follows

```
function dydx = funkeval(x,y);
k = 401.0;
L = 0.5;
radius = 0.025;
diameter = 2.0*radius;
Area = pi*diameter*L;
Volume = pi/4*diameter^2*L;
S = Area/Volume;
Tsurround = 77.0;
sigma = 5.670373e-8;
eps = 0.15;
fac = eps*sigma*S/k;
dydx(1) = y(2);
dydx(2) = fac*(y(1)^4 - Tsurround^4);
```

At the command line prompt, I typed

```
>> [x0,err] = nrnd1(-100)
```

where -100 K/m was my initial guess for the initial slope, based on my solution to problem 2. This command generated the following output.

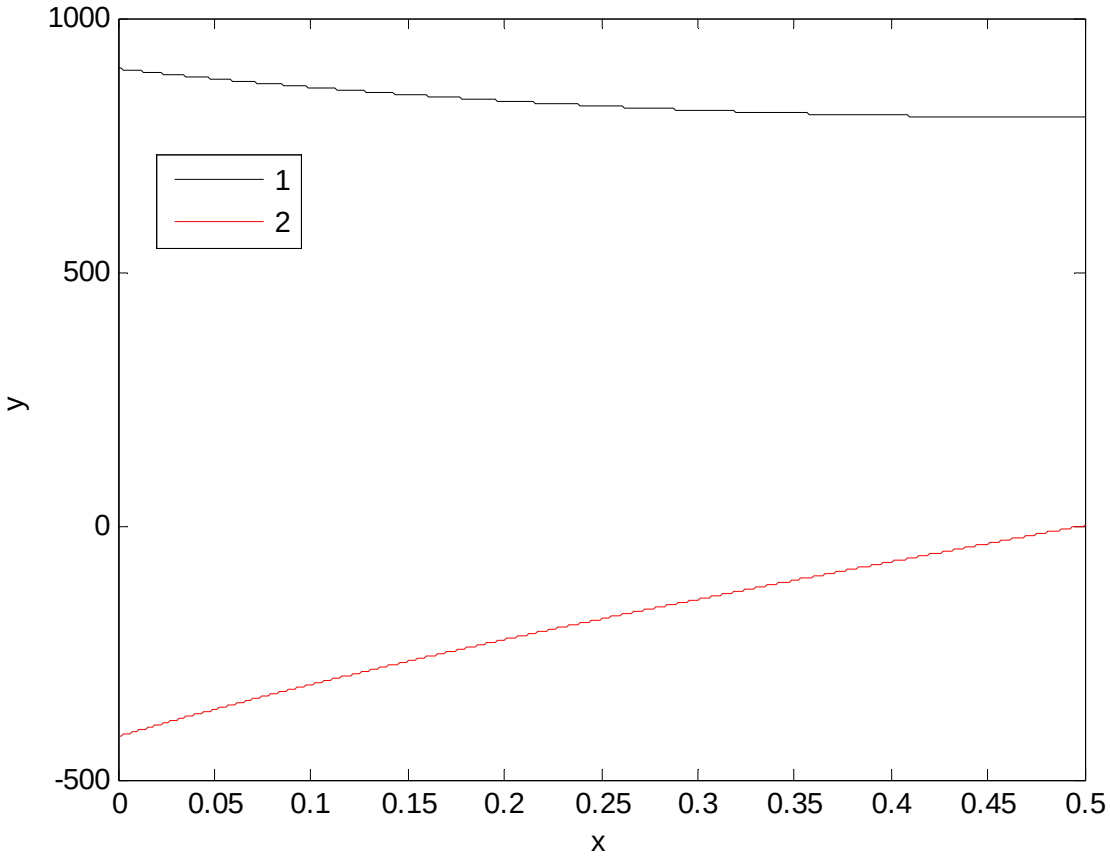
```
>> [x0,err] = nrnd1(-100);
icount = 1 xold = -1.000000e+02 f = 5.158883e+02 df = 1.781830e+00 xnew = -3.895273e+02  err = 1.000000e+02
icount = 2 xold = -3.895273e+02 f = 3.928538e+01 df = 1.525552e+00 xnew = -4.152789e+02  err = 6.201033e-02
icount = 3 xold = -4.152789e+02 f = 2.417708e-01 df = 1.506880e+00 xnew = -4.154393e+02  err = 3.862048e-04
icount = 4 xold = -4.154393e+02 f = 9.594935e-06 df = 1.506766e+00 xnew = -4.154393e+02  err = 1.532812e-08

x0 = -415.4393
err =   1.5328e-08
```

The Newton Raphson method converged in four iterations. The initial slope is -415.4393 K/m.

The converged solution can be viewed by typing the following command,

```
>>  [x,y]=rk4n(1000,0,0.5,[900, -415.4393]);
```

The resulting figure is provided below.



We can confirm that this is the solution to the boundary condition by checking the value of *y* at the final value of *x*.

```
>> >> y(1001,1)

ans =  804.1158 K
```

This solution agrees with our solution of the PDE given above.