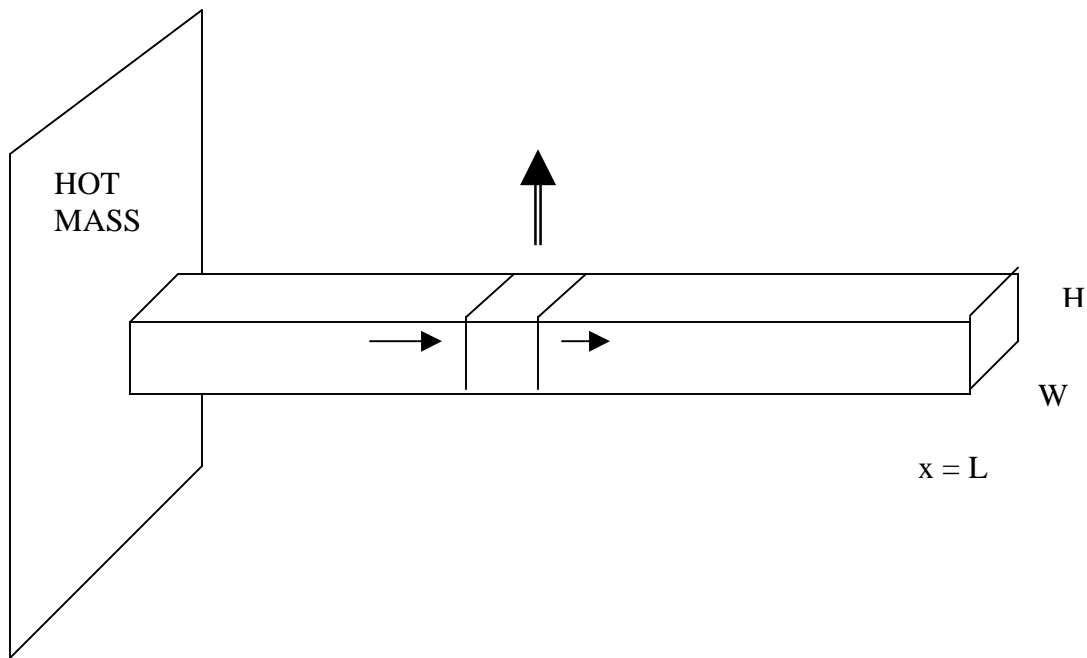


Parameter Identification for Steady State Heat Diffusion

Consider heat diffusing from a hot mass down a thin rod and into a surrounding cooler region. Suppose this rod has length equal to L in the x direction, and it is rectangular with other dimensions equal to H and W where both these are much smaller than L .



Change in the heat = $0 =$ (diffusion from the left)

- (diffusion to the right)

- (heat loss to the surrounding region)

Let $u(x)$ be the steady state temperature in the long thin rod at position x .

The Fourier heat law gives

$$0 = Ku_{xx} + c(2H + 2W)/(WH)(u_{\text{sur}} - u)$$

$$0 = u_{xx} + C(u_{\text{sur}} - u) \quad \text{where}$$

$$C = (c/K)(2H + 2W)/(WH)$$

u_{sur} = given surrounding temperature.

K = thermal conductivity which may be known

c = unknown proportionality constant in heat loss to surrounding region.

So, C is not known. Two other unknowns are the heat diffusion rates at $x = 0$ and L :

$$-u_x(0) = q_1/K = Q_1 \text{ and } u_x(L) = q_2/K = Q_2.$$

In summary, C , Q_1 and Q_2 are not known, $u = u(x, C, Q_1, Q_2)$ must satisfy:

$$0 = -u_{xx} + C(u_{sur} - u) \text{ for } x \text{ between } 0 \text{ and } L$$

$$-u_x(0) = q_1/K = Q_1 \text{ and } u_x(L) = q_2/K = Q_2.$$

One can show that $u(x, C, Q_1, Q_2)$ must have the form

$$u = C_1 e^{\sqrt{C}x} + C_2 e^{-\sqrt{C}x} + u_{sur} \text{ where}$$

$$Q_1 = -C_1 \sqrt{C} + C_2 \sqrt{C}$$

$$Q_2 = C_1 \sqrt{C} e^{\sqrt{C}L} + C_2 \sqrt{C} e^{-\sqrt{C}L}.$$

In order to find C , Q_1 , Q_2 , we must make a number of observations for the steady state temperatures, d_i , as a function of space, x_i . Then we can form the residuals

$$r_i = d_i - u(x_i, C, Q_1, Q_2).$$

Choose C , Q_1 , Q_2 so that $r^T r = \sum_i (d_i - u(x_i, C, Q_1, Q_2))^2$ is a minimum. Because of the

complicated nature of $u(x_i, C, Q_1, Q_2)$, the normal equations are not applicable.

Fortunately, Matlab has an implementation of a search algorithm called the Nelder-Mead simplex method, and this is in the Matlab command `fminsearch.m`. A very sketchy

outline of this method for three unknowns when $f = r^T r$ and the three variables are

$x = (C, Q_1, Q_2)$ is as follows.

Step 1: Pick four points to form a tetrahedron.

Step 2: Order the points so that

$$f(x^0) \leq f(x^1) \leq f(x^2) \leq f(x^3).$$

(The solution may be nearest to the plane formed by the first three points)

Step 3: Choose a new fourth point starting with some point on the line from the old fourth point and going through the center $x = (x^0 + x^1 + x^2)/3$ of the plane. Here there are a number of additional steps.

Step 4: After the new fourth point has been selected repeat steps 2-3.

Outline of Parameter Identification via Matlab's fminsearch.

1. Function definitions.
 - a. Find the function and its parameters....you may need to use dsolve.
 - b. Create a Matlab function file, sol, for this as a of function its independent variable and its parameters.
 - c. Create a Matlab function file, lsfct, for the least squares function
2. Execute fminsearch.
3. Use the output, the numerical values of the parameters, of fminsearch.
 - a. Put the parameters into the function in 1a.
 - b. Create a table of values for this function.
 - c. Create a graph of this function.

1a. EDU>> dsolve('D2y + C*(21.47 - y)=0','Dy(0)= Q1', 'Dy(70) = 0')
ans =
2147/100+Q1/C^(1/2)*sinh(C^(1/2)*t)-
Q1*cosh(70*C^(1/2))/C^(1/2)/sinh(70*C^(1/2))*cosh(C^(1/2)*t)

1b. Contents of sol.m
function r =sol(x1,x2,t)
r = -1/100*(-
2147*exp(140*x1^(1/2))*x1^(1/2)+2147*x1^(1/2)+100*x2*exp(x1^(1/2)
*t)+100*x2*exp(-x1^(1/2)*(-140+t)))/x1^(1/2)/(exp(140*x1^(1/2))-1);

1c. Contents of lsfct.m
function lsr=lsfct(xx)
x = [10 14 18 22 26 30 34 38 42 46 50 54 58 62 66];
temp = [115.95 101.94 90.18 80.01 71.37 63.75 56.94 51.97 48.21 44.48
40.66 38.51 37.81 36.11 35.18];
lsr = 0;
for i=1:15
lsr = lsr + (temp(i)-sol(xx(1),xx(2),x(i)))^2;
end

2. Execute fminsearch
EDU>> format long
EDU>> [xx fctval] = fminsearch('lsfct', [.0012 -4.3])
xx =
0.00184474393991 -6.25728450461340
fctval =
7.27302530421425

So, C = .0018447 and Q1 = -6.257284, and for these the least square function has minimum equal to 7.2730253.

3a. EDU>> tempfct = dsolve('D2y + 0.0018447*(21.47 - y)=0','Dy(0)=-6.257284',
'Dy(70) = 0')
tempfct =
2147/100-
142211/419250*sinh(1/10000*184470^(1/2)*t)*184470^(1/2)+142211/41
9250*184470^(1/2)*(exp(7/1000*184470^(1/2))^2+1)/(exp(7/1000*1844
70^(1/2))^2-1)*cosh(1/10000*184470^(1/2)*t)

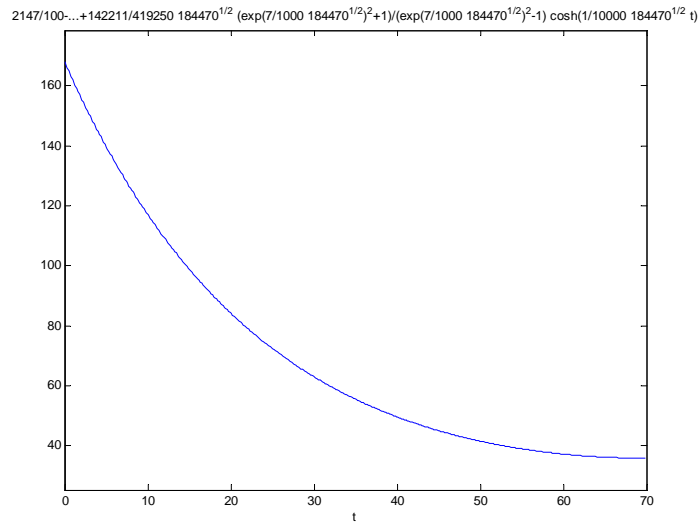
3b. Form the table of location, computed temperatures and temperature data

```
EDU>> location = 10:4:66;
EDU>> for i = 10:4:66
temp(i) = subs(tempfct,'t',i);
end
EDU>> temp_data = [ 115.95 101.94 90.18 80.01 71.37 63.75 56.94 51.97
48.21 44.48 40.66 38.51 37.81 36.11 35.18];
EDU>> format short
EDU>> table = [location' temp(10:4:66)' temp_data']
```

table =

10.0000	117.0702	115.9500
14.0000	102.1691	101.9400
18.0000	89.6557	90.1800
22.0000	79.1597	80.0100
26.0000	70.3707	71.3700
30.0000	63.0285	63.7500
34.0000	56.9160	56.9400
38.0000	51.8522	51.9700
42.0000	47.6873	48.2100
46.0000	44.2982	44.4800
50.0000	41.5846	40.6600
54.0000	39.4660	38.5100
58.0000	37.8800	37.8100
62.0000	36.7794	36.1100
66.0000	36.1319	35.1800

3c. Graph the temperature function



The accuracy of this modeling process can be critiqued at a number of points. First, the measured data may have errors; the reader may find it interesting to experiment with some variations of the measured data. Second, we assumed that the right side boundary condition was set equal to zero, $u_x(70) = 0$; see the files `sol1.m`, `lsfct1.m` for the case where this not zero. Third, `fminsearch` is an implementation of the Nelder-Mead search algorithm, which may not always give accurate "solutions"; the interested reader should consult the Matlab help file on `fminsearch`. Fourth, there is the possibility of a number of choices of the parameters that could minimize the least squares function. The following Matlab code, `graphlsfct.m`, will generate 3D and contour plots for the about least squares function as a function of the two parameters. Note the long valley in the graph of this function, that indicates a number of possible minimizers.

```

begin1 = .0016;
end1 = .0021;
begin2 = -7;
end2 = -6;
n1 = 30;
n2 = 30;
for i = 1:n1
    x1(i) = begin1 +(i-1)*(end1-begin1)/n1;
    for j = 1:n2
        x2(j) = begin2 +(j-1)*(end2-begin2)/n2;
        u(i,j) = lsfct([x1(i) x2(j)]);
    end
end
end
mesh(x1,x2,u)
%contour(x1,x2,u',[6 10 14 18 22 26 30 34 38])

```

