Math 2400 - Numerical Analysis

Homework # 4 Solutions

In this assignment, we will examine nonlinear and linear least squares. You will need some data to test your program with, so go to the course web-page and download the data file hw4data. Once you have the file in your directory, you can type `load -mat hw4data` at the Matlab command prompt. This will load two variables t and y. We will assume that $y = ce^{kt}$ for some c and k. The data contains errors, so we use least-squares to find the best c and k.

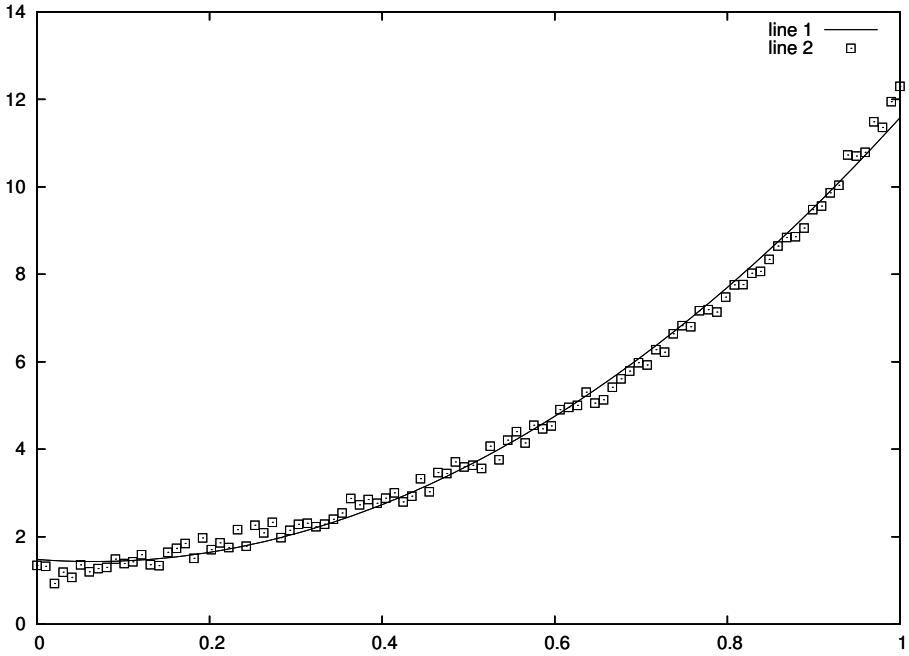
1. Use linear least squares to find the best fit quadratic $y = c_1 + c_2 t + c_3 t^2$ for the given data.

   We use the three basis functions $\phi_1 = 1$, $\phi_2 = t$ and $\phi_3 = t^2$ to find the best fit quadratic. To find the c_i's we will solve $A\vec{c} = b$. The components are given by $(A)_{ij} = \sum_{k=1}^{m} \phi_i(t_k)\phi_j(t_k)$ and $b_i = \sum_{k=1}^{m} y_k \phi_i(t_k)$. Here is the session I used to find the $c_i's$.

```
octave:2> load -m hw4data
octave:3> A=zeros(3)
A =

  0  0  0
  0  0  0
  0  0  0

octave:4> A(1,1)=lenght(t)
error: 'lenght' undefined near line 4 column 8
error: evaluating assignment expression near line 4, column 7
octave:4> A(1,1)=length(t)
A =

  100    0    0
    0    0    0
    0    0    0

octave:5> for i=1:100
> A(1,2)=A(1,2)+t(i);
> A(1,3)=A(1,3)+t(i)^2;
> A(2,3)=A(2,3)+t(i)^3;
> A(3,3)=A(3,3)+t(i)^4;
> end
octave:6> A
A =

  100.00000   50.00000   33.50168
    0.00000    0.00000   25.25253
    0.00000    0.00000   20.30337

octave:7> A(2,1)=A(1,2);
octave:8> A(2,2)=A(1,3);
octave:9> A(3,1)=A(1,3);
octave:10> A(3,2)=A(2,3);
octave:11> A
A =

  100.000   50.000   33.502
   50.000   33.502   25.253
   33.502   25.253   20.303

octave:12> b=zeros(3,1)
b =
```

```
  0
  0
  0

octave:13> for i=1:100
> b(1)=b(1)+y(i);
> b(2)=b(2)+y(i)*t(i);
> b(3)=b(3)+y(i)*t(i)^2;
> end
octave:14> b
b =

  461.54
  316.63
  247.19

octave:15> c=inv(A)*b
c =

    1.4809
   -1.5084
   11.6076

octave:16> plot(t,c(1)+c(2).*t+c(3).*t.^2)
octave:17> hold on
octave:18> plot(t,y,'x')
octave:19> gset output "hw4071a.eps"
octave:20> gset term postscript eps
octave:21> replot
```

Below is the graph produced by the session.



2. Use non-linear least squares to find a c and k which minimizes $\sum_{i=0}^{m}(y_i - ce^{kt_i})^2$. You may use Gauss-Newton

iterations in your non-linear solver. Plot the exponential cure and the data on the same graph. Mark the data points with an `x`.

First we need subroutines to evaluate the vector r of errors and its Jacobian with respect to c and k. They are given by

$$r = \begin{pmatrix} y_0 - ce^{kt_0} \\ \vdots \\ y_m - ce^{kt_m} \end{pmatrix}, \quad Dr = \begin{pmatrix} -e^{kt_0} & -ct_0e^{kt_0} \\ \vdots & \vdots \\ -e^{kt_m} & -ct_me^{kt_m} \end{pmatrix}.$$

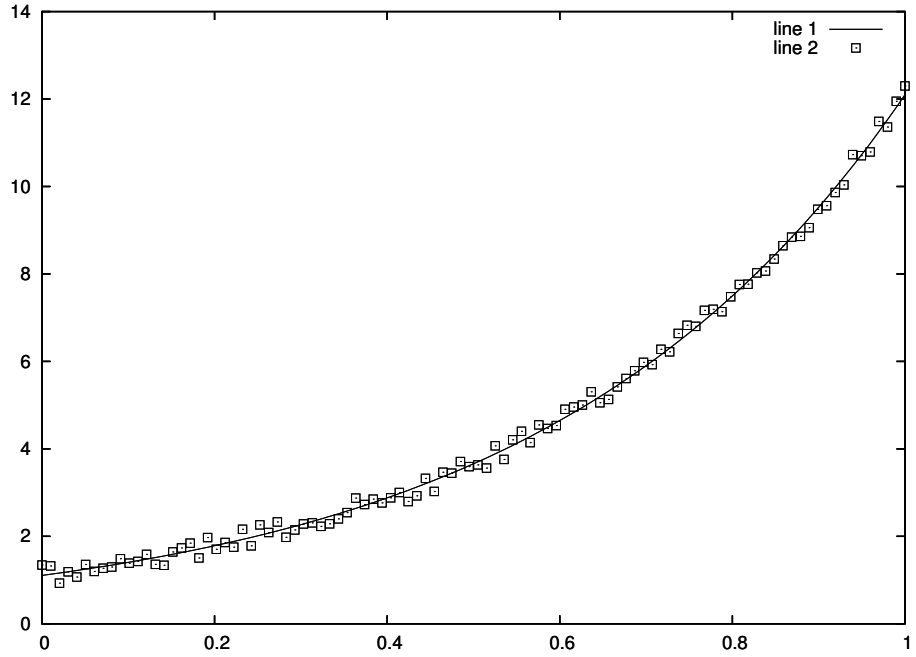
Here are the Matlab routines I used to evaluate these expressions.

```
function x=r(t,y,c,k)

m=length(y);
x=zeros(m,1);

for i=1:m
  x(i)=(y(i)-c*exp(k*t(i)));
end

function J=Dr(t,y,c,k)

m=length(y);
J=zeros(m,2);

n=2;

for i=1:m
   J(i,1)=-exp(k*t(i));
   J(i,2)=-c*t(i)*exp(k*t(i));
end
```

Here is the Matlab session

```
octave:2> load hw4data
octave:3> m=length(t)
m = 100
octave:4> c0=[1;1]
c0 =

  1
  1

octave:5> for i=1:50
> delc=-inv(Dr(t,y,c0(1),c0(2))'*Dr(t,y,c0(1),c0(2)))*Dr(t,y,c0(1),c0(2))'*r(t,y,c0(1),c0(2));
> c0(1)=c0(1)+delc(1);
> c0(2)=c0(2)+delc(2);
> end
octave:6> c0(1)
ans = 1.1068
octave:7> c0(2)
ans = 2.3915
octave:8> plot(t,y,'x')
octave:9> hold on
octave:10> plot(t,c0(1).*exp(c0(2).*t))
octave:11> gset output "hw4061.eps"
octave:12> gset terminal postscript eps
octave:13> diary off
```

Here is the graph the session produced.



3. In this section we will use linear least squares on transformed data.

   (a) Transform the data with the `log(x)` function to an appropriate form for linear least squares.
   If we take the logarithm of the model, we get

$$\begin{aligned} \ln(y) &= \ln(ce^{kt})\,, \\ \ln(y) &= \ln(c) + \ln(e^{kt})\,, \\ \ln(y) &= \ln(c) + kt\,. \end{aligned}$$

   So for this model, we only need to take the logarithm of the data and not the times. This is one of the key differences between this exponential model and the previous power law model. Here is the Matlab session I used to find the line of best fit going through $(t, \ln(y))$.

   (b) Use linear least squares to determine a straight line which best approximates the transformed data.
   Here is the Matlab session to determine the line of best fit.

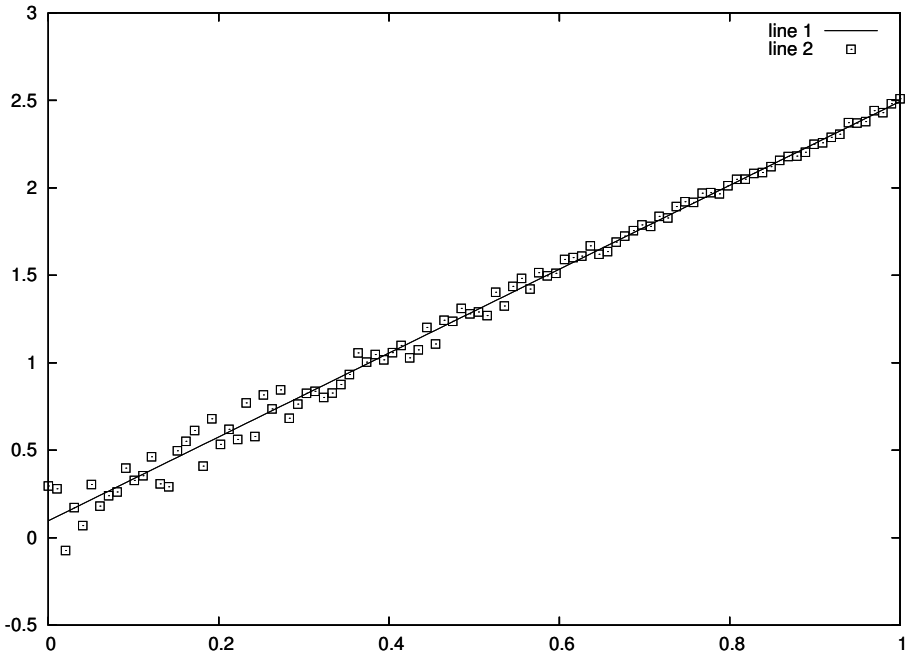
```
octave:32> y1=log(y);
octave:33> A=zeros(2)
A =

  0  0
  0  0

octave:34> A(1,1)=length(t)
A =

  100    0
    0    0

octave:35> for i=1:100
> A(1,2)=A(1,2)+t(i);
> A(2,2)=A(2,2)+t(i)^2;
```

```
> end
octave:36> A(2,1)=A(1,2)
A =

  100.000   50.000
   50.000   33.502

octave:37> b=zeros(2,1)
b =

  0
  0

octave:38> for i=1:100
> b(1)=b(1)+y1(i);
> b(2)=b(2)+y1(i)*t(i);
> end
octave:39> newcs=inv(A)*b
newcs =

  0.096756
  2.396874

octave:40> plot(t,y1,'x')
octave:41> hold on
octave:42> plot(t,newcs(1)+newcs(2).*t)
```

So the best line of best fit for the modified data is given by

$$\ln(y) = 0.096756 + 2.396874t$$

Here is a graph of the line of best fit with the transformed data



(c) Use your result to find c and k

We can see from the transformed model, the slope of the line of best fit is our rate constant. To find the initial concentration, we must invert the logarithm.

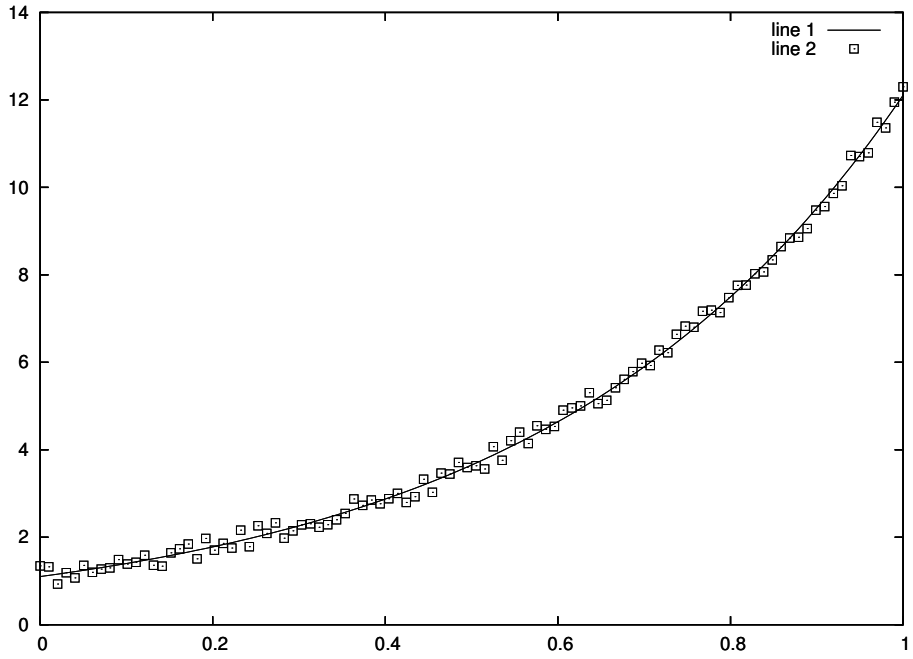
```
octave:40> c1=exp(newcs(1))
c1 = 1.1016
octave:41> k1=newcs(2);
octave:42> plot(t,y,'x')
octave:43> hold on
octave:44> plot(t,c1.*exp(k1.*t))
octave:45> gset output "hw4-2b.eps"
octave:46> gset terminal postscript eps
octave:46> diary off
```

So the best fit exponential we get is

$$y = 1.1016e^{2.396847t}.$$

Not very different from the nonlinear least squares result.

(d) plot the resulting curve and the data points on the same graph. Mark the data points with an `x`.



4. Compare the accuracy of the two methods by calculating

$$\sum_{i=0}^{m}(y_i - ce^{kt_i})^2$$

for both cases and compare the results. Give a possible explanation for any differences found.

Here is the Matlab session to calculate the sum of the errors. Sum1 will be for the nonlinear least squares calculation and sum2 will be for the linear least squares calculation.

```
octave:111> sum1=0
sum1 = 0
octave:112> sum2=0
sum2 = 0
octave:113> for i=1:100
> sum1=sum1+(y(i)-c(1)+c(2)*t(i)+c(3)*t(i)^2)^2;
> sum2=sum2+(y(i)-c0(1)*exp(c0(2)*t(i)))^2;
> sum3=sum3+(y(i)-c1*exp(k1*t(i)))^2;
> end
octave:114> sum1
sum1 = 5.0011
```

```
octave:115> sum2
sum1 = 2.1428
octave:116> sum3
sum2 = 2.1464
octave:117> diary off
```

It is clear that the quadratic gives the poorest result. The reason is that the underlying data follows an exponential relationship and a quadratic can not be expected to approximate the cures as well as an exponential. However, if you look at the graph, you can see that we get a reasonable result. So if you are not going to use the quadratic for long-time interpolation and are not too concerned with accuracy then the quadratic would be a suitable approximation.

So you can see that the other two approaches result in a similar error, but the nonlinear squares method has a slightly lower error. Since both are minimizing the error, we would expect them to both to have the same error. Why are they different.

The answer is that the logarithmic transform affects the error. Consider the error at time t_i,

$$y_i = ce^{kt_i} + \varepsilon_i \,.$$

Now we take the transform of both sides,

$$\ln(y_i) = \ln(ce^{kt_i} + \varepsilon_i) \,.$$

Now we expect the error to be small relative to the data, so we expand the transform as a first order Taylor polynomial about $\varepsilon_i = 0$ to get the following relation.

$$\ln(y_i) = \ln(ce^{kt_i}) + \frac{1}{ce^{kt_i}}\varepsilon_i$$

or

$$\ln(y_i) = \ln(c) + kt_i + \frac{1}{ce^{kt_i}}\varepsilon_i$$

So instead of minimizing $\sum_{i=0}^{m} \varepsilon_i^2$, we are minimizing $\sum_{i=0}^{m} \left(\frac{1}{ce^{kt_i}}\varepsilon_i\right)^2$. If we assume k is positive, then this transform places more weight on data for small t. If you look at the graph of the transformed data, it is clear that errors for small time data are greatly exaggerated. If $k < 0$ then more weight is placed on large t data.

Finally I note that in theory the nonlinear least squares should always give a better result for errors which are additive, but one also has to consider the error introduced by Newton's method. In this case both solutions where very close, however there are times using a transform on data to allow the use of a linear least squares method can be disastrous.

Hint: To see the effect of the transform to a linear system, consider the effect of the transform on the error of the i^{th} step.

$$y_i = ce^{kt_i} + \epsilon_i$$

We take the transform and we get

$$\ln(y_i) = \ln(ce^{kt_i} + \epsilon_i)$$

To see what is happening use a tayor expansion about $\epsilon = 0$