Non-Linear Regression

So far we've focussed on linear regression: problems where the model is linear in the unknown modelling parameters. This is convenient, but not really necessary! Suppose we have the non-linear model:

$$\bar{b} = f(\underset{\sim}{x}, t)$$

where $\underset{\sim}{x}$ is an array of modelling functions.

We may still define the residual between observed and fitted values:

$$\underset{\substack{\uparrow \\ \text{residual}}}{r_i} = \underset{\substack{\uparrow \\ \text{meas}}}{b_i} - f(\underset{\substack{\uparrow \\ \text{model}}}{\underset{\sim}{x}}, t_i)$$

Thus we can form the sum of squares:

$$F(\underset{\sim}{x}) = \sum_{i=1}^{N} (r_i)^2$$

The best fitted values for $\underset{\sim}{x}$ are obtained via finding the minimum of $F(\underset{\sim}{x})$!

Be Wary: Non-linear optimization problems may have local minima which can trap the solver! Also, make sure you have set up the parameters so that $F(\underset{\sim}{x})$ is well conditioned. The dependence on each parameter should be of similar magnitude. Let's look at an example:

Arrhenius Kinetics

$$\text{rate} \sim K_0 e^{-E/RT}$$

In senior lab you will measure the rate of oxidation of methane as a function of temperature, and use it to try to get the activation energy E for the catalyst. If we could measure this rate, holding everything but T constant, we could use linear regression. Unfortunately, the rate <u>also</u> depends on concentration, and <u>that</u> depends (experimentally) on T as well!

The exp't is written up in Chem Eng. Education, <u>36</u> (1) p. 34-40. Suppose we feed a reactor (well-mixed) a concentration of Methane C_{r_0} at flow rate q_r. We measure some outlet concentration C_r. The subscript "r" is because the reactor is <u>hot</u>: due to expansion, the <u>actual</u> concentration is $\frac{C}{C_r} = \frac{T_r}{T}$ from the ideal gas law - where T_r is some ref. temp (°K).

The system is further complicated be cause the reaction is <u>fractional order</u>, e.g.

$$rate \sim C^n$$

where $n \neq 1$ (not first order)

we need to figure out n too!

From a mass balance, we get the rate of rxn per gram of catalyst:

$$\frac{q_r}{m}\left(C_{ro}-C_r\right) = \left(C_r \frac{T_r}{T}\right)^n k_o e^{-\frac{E}{RT}}$$

where m is the mass of catalyst.

We can also look at the conversion ratio given by:

$$x = 1 - \frac{C_r}{C_{r_o}}$$

which yields (after rearrangement):

$$\frac{x}{(1-x)^n} = \frac{m}{q_r} C_{or}^{n-1}\left(\frac{T_r}{T}\right)^n k_o e^{-\frac{E}{RT}}$$

If you were to fix T and plot x vs. C_{or}, you find it decreases w/ C_{or}, thus $n<1$ for this system.

Ok, suppose we vary C_{ro} and T, and measure C_r. How do we get the unknowns n, k_o, and E?

The classic approach is to do it in two steps:

1) Fix T and plot

$$\ln\left\{\frac{q_r}{m}(c_{or}-c_r)\right\} \text{ vs. } \ln\{C_r\}$$

You should get a straight line with slope n!

2) With this in hand, you can then do an Arrhenius Plot of:

$$\ln\left\{\frac{q_r}{m}\frac{(C_{or}-C_r)}{C_r^{\,n}}\left(\frac{T_r}{T}\right)^{-n}\right\} = \ln K_o - \frac{E}{R}\frac{1}{T}$$

Thus, the *slope* (vs. $\frac{1}{T}$) is just E/R and the intercept is $\ln k_o$!

This works fine *if* the data is good. Unfortunately, this is *not* usually the case. The problem is that c_r is a *much* stronger function of T than c_{r_0}, and thus you get large errors in the first step (n), leading to large errors in the second step (E & K_o). The fitting parameters are also strongly biased by errors at low conversions.

We can avoid this by using non-linear regression! We simply define the deviation from the model Δ:

$$\Delta = C_r - C_{r_0} + \frac{m}{q_r} C_r^n k_0 e^{-\frac{E}{RT}} \left(\frac{T_r}{T}\right)^n$$

and minimize the objective function:

$$F(k_0, E, n) \equiv \sum \Delta_i^2$$

over all the experiments! The sensitivity of the fitting parameters to error can then be easily calculated using the non-linear error propagation/gradient method, accounting for the error in both C_r and T. Note that you should work with n, $\ln k_0$, and $\frac{E}{RT_r}$ as fitting parameters so that the optimization problem is well-conditioned!

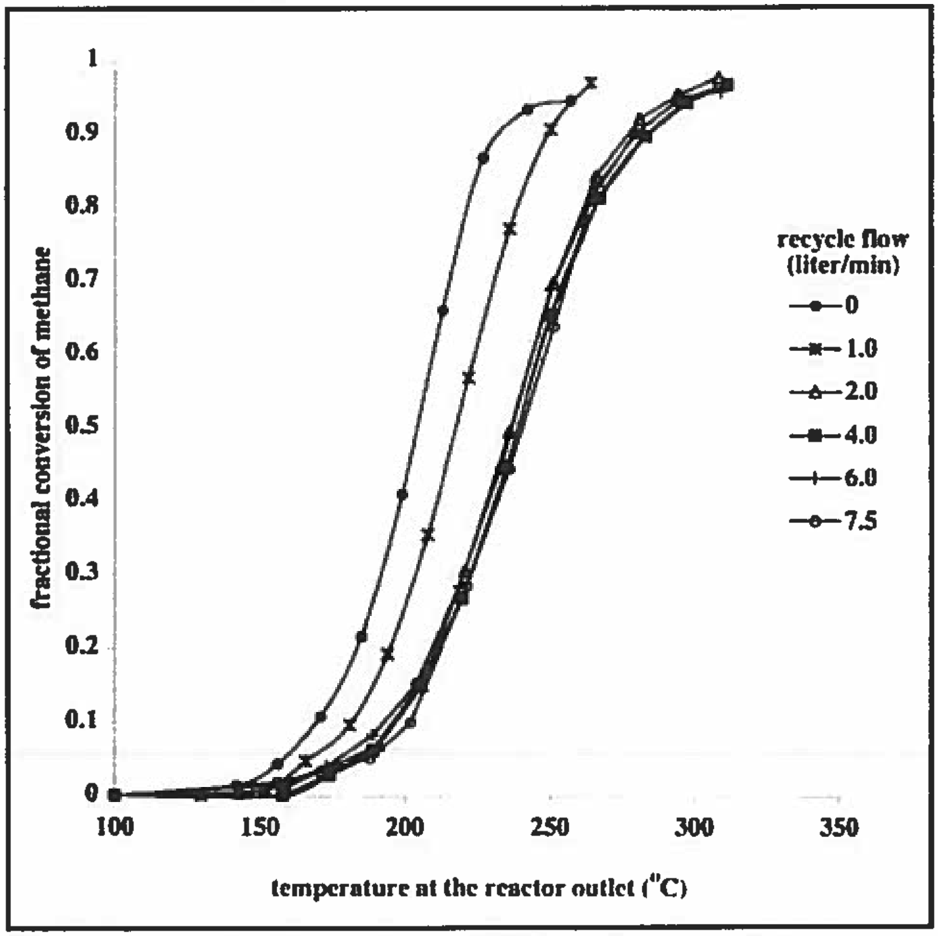


Figure 2. *The effect of recycle flow on methane conversion for a feed concentration, C_{or}, of $1.976x10^{-4}$ mol/liter.*

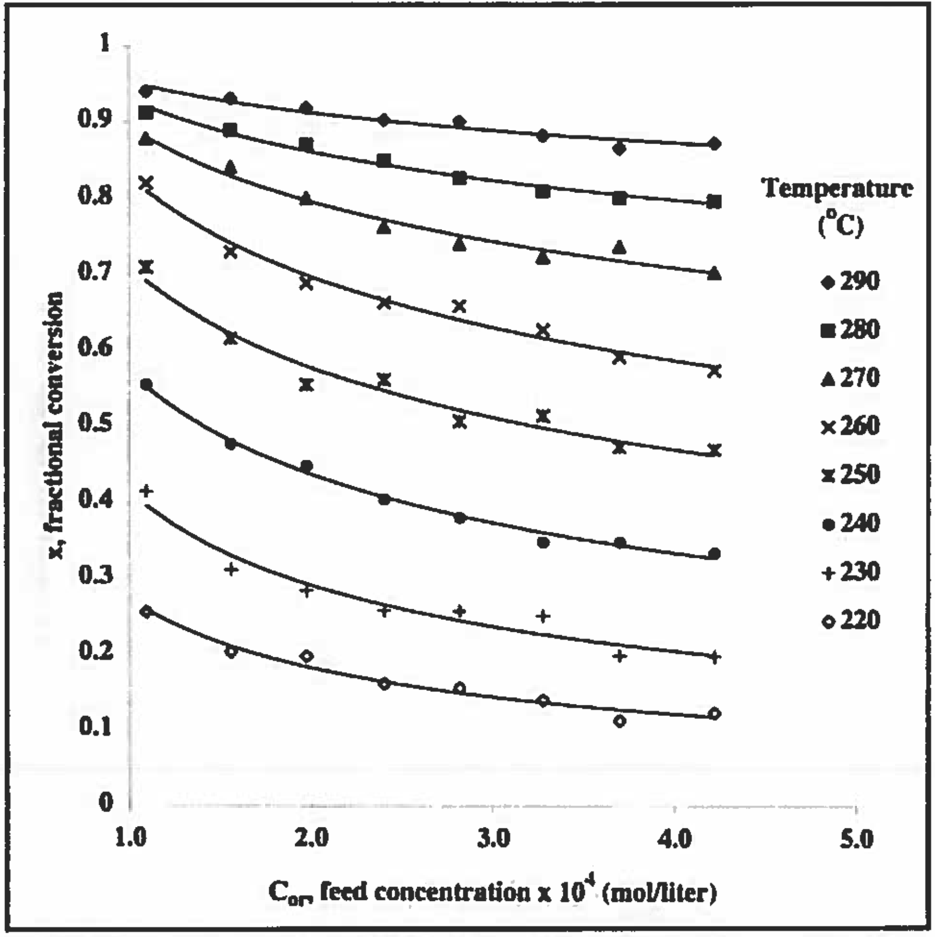


Figure 3. *The effect of feed concentration on methane conversion under gradientless conditions. The curves are least-squares polynomial representations of the data.*

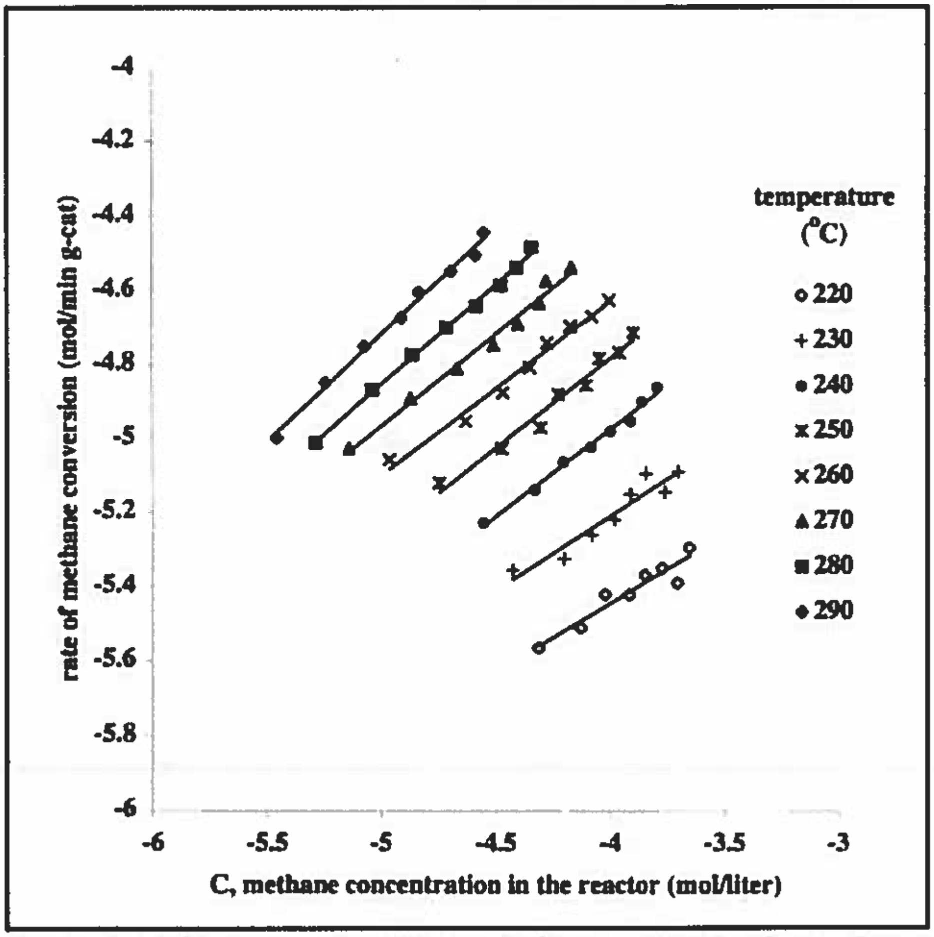


Figure 4. *Methane conversion rates. The straight lines are least-squares representations of the data shown. The numbers on the coordinates are exponents of 10.*

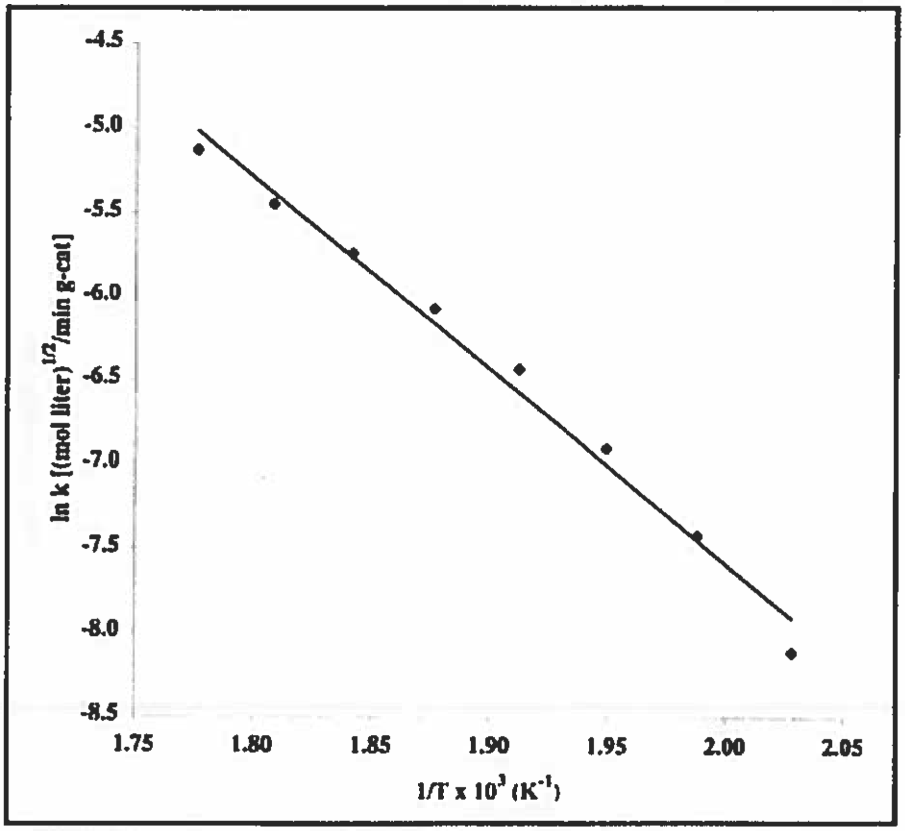
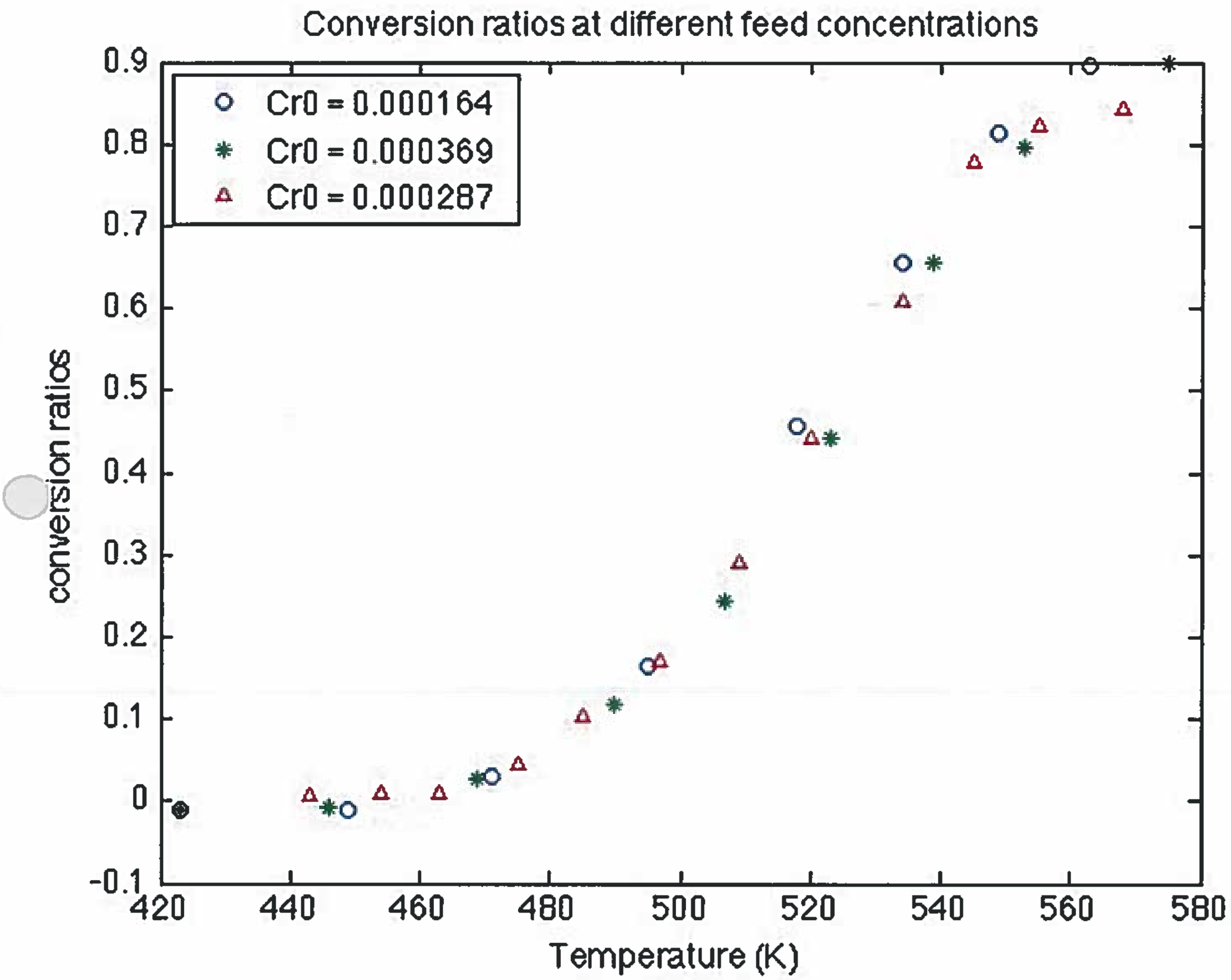


Figure 5. *An Arrhenius plot. The values of k_o and E/R, determined from the least-squares line shown, are given in Eqs. (6) and (7).*


Conversion ratios at different feed concentrations
Cr0 = 0.000164
Cr0 = 0.000369
Cr0 = 0.000287
conversion ratios
Temperature (K)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
420
440
460
480
500
520
540
560
580

```
function y=delta(guess)
%This function takes in guesses for the unknown parameters n, ln(k0) and
%E/RTr, and returns the deviation between the model and the data.  We bring
%the data in through the "global" command:
global crpass cr0pass Tpass

Tr=298; %The reference temperature.
qr=0.1; %The flow rate (liters/min)
m=1; %The amount of catalyst (g)

n=guess(1); %The first parameter
k0=exp(guess(2)); %The second parameter
ERTr=guess(3); %The third parameter

miss=crpass-cr0pass+m/qr*crpass.^n*k0.*(Tr./Tpass).^n.*exp(-ERTr*Tr./Tpass);

%OK, the question is how to weight each of the data points.  As it is
%currently written, it tends to accentuate the weighting at higher
%temperatures where the conversion is the largest.  This is because cr is
%lowest, and it is multiplied by a large value to make it balance cr0.  It
%also places a stronger weight on the runs with higher intial
%concentrations.  On the other hand, for lower cr we should have more
%accurate measurements (if the fractional error is fixed, for example).
%Different weightings will yield different "solutions" for optimal parameters.

%A reasonable choice is to weight each of the runs with the inverse of the
%initial concentration.  This is essentially equivalent to assuming an error
%proportional to the concentration measured, and each data point should be
%of O(1).  Thus:
miss=miss./cr0pass;

%and thus we get the objective function:
y=sum(miss.*miss);
```

Ex. 18

```
clear
echo on
%In this example we analyze the catalytic oxidation data obtained by a
%group of students in senior lab.  Under CSTR conditions they measured outlet
%concentrations for three different reactor feed concentrations.  The data
%for the three feeds cr0 are given below:

Ta=[423 449 471 495 518 534 549 563];
cra=[1.66E-04   1.66E-04    1.59E-04    1.37E-04    8.90E-05    5.63E-05    3.04E-05
cr0a=1.64E-4;

Tb=[423 446 469 490 507 523 539 553 575];
crb=[3.73E-04   3.72E-04    3.59E-04    3.26E-04    2.79E-04    2.06E-04    1.27E-04
cr0b=3.69e-4;

Tc=[443 454 463 475 485 497 509 520 534 545 555 568];
crc=[2.85E-04   2.84E-04    2.84E-04    2.74E-04    2.57E-04    2.38E-04    2.04E-04
cr0c=2.87e-4;
pause

%We can calculate conversion ratios for these three runs:
xa=1-cra/cr0a;
xb=1-crb/cr0b;
xc=1-crc/cr0c;

%and we can plot them up:
figure(1)
plot(Ta,xa,'o',Tb,xb,'*',Tc,xc,'^')
xlabel('Temperature (K)','FontSize',14)
ylabel('conversion ratios','FontSize',14)
legend(['Cr0 = ',num2str(cr0a)],['Cr0 = ',num2str(cr0b)],['Cr0 = ',num2str(cr0c)],'I
title('Conversion ratios at different feed concentrations','FontSize',14)
set(gca,'FontSize',14)
pause

%Looking at this plot, we can immediately see why the standard technique
%for analyzing the reaction data will run into trouble: Even with
%interpolation, it will be very hard to get accurate values of the
%conversion at different concentrations for fixed temperatures.  To use
%non-linear regression to get at the fitting parameters, we will have to
%define an objective function for minimization, as well as some initial
%guesses for the parameters.  We can pass the data into the objective
%function using the "global" meat axe:

global crpass cr0pass Tpass

Tpass=[Ta,Tb,Tc];
crpass=[cra,crb,crc];
cr0pass=[cr0a*ones(size(Ta)),cr0b*ones(size(Tb)),cr0c*ones(size(Tc))];

%and you will have to save the function "delta.m" which returns the
%objective function to be minimized.
pause
```

```
%OK, let's do it.  We have the initial guesses:
guess=zeros(3,1);
guess(1)=.5; %This is the guess for n
guess(2)=15; %This is the guess for ln(k0)
guess(3)=38; %This is the guess for E/RTr

%And we go:
guess=fminsearch('delta',guess)
pause
%Looking at these values, they aren't too far off of those in the
%literature.  In particular, the exponent is quite close to the
%expected value of 0.5, and the activation energy is only off by 7%!
pause

%We can plot the model up too.  We need the other parameters
%(both here and in the function delta.m).
Tr=298; %The reference temperature.
qr=0.1; %The flow rate (liters/min)
m=1; %The amount of catalyst (g)

n=guess(1);
k0=exp(guess(2));
ERTr=guess(3);

Trange=[min(Tpass):max(Tpass)]; %A plotting range
xmodel=m/qr*(Tr./Trange).^n*k0.*exp(-ERTr*Tr./Trange);

xafn=xa./(1-xa).^n/cr0a^(n-1);
xbfn=xb./(1-xb).^n/cr0b^(n-1);
xcfn=xc./(1-xc).^n/cr0c^(n-1);

%OK, we've got the model and the data for the function x/(1-x)^n/cr0^(n-1).
%It should be independent of the concentration. Let's plot it up:
pause
figure(2)
plot(Ta,xafn,'o',Tb,xbfn,'*',Tc,xcfn,'+',Trange,xmodel)
xlabel('Temperature (K)','FontSize',14)
ylabel('x/(1-x)^n/cr0^(n-1)','FontSize',14)
legend(['Cr0 = ',num2str(cr0a)],['Cr0 = ',num2str(cr0b)],['Cr0 = ',num2str(cr0c)],'m
title(['Comparison of data to model, n = ',num2str(n)],'FontSize',14)
set(gca,'FontSize',14)
%Which shows that we get pretty much perfect collapse of the data.
pause

%Now we turn to the trickier error calculations.  First, we need to get a
%measure of the uncertainty in the concentration measurements.  We can get
%this from the magnitude of the "miss" in the data:
miss=crpass-cr0pass+m/qr*crpass.^n*k0.*(Tr./Tpass).^n.*exp(-ERTr*Tr./Tpass);

%We must adjust this to account for the relative weighting of the data.  In
%this case, a rough correction for the actual fractional deviation in cr
%is given by:
miss=miss./cr0pass.*(crpass./cr0pass);

%Thus we get the fractional standard deviation (assuming randomness) of:
crstdev=norm(miss)/(length(Tpass)-3)^.5
```

```
%Which yields a fractional error of around 3% - not too bad.  Note that these
%deviations could have been due to errors in the temperature just as readily!
pause

%OK, it is always important to plot up the residuals to see if the error is
%really random.  It is useful to plot up the actual cr's and predicted
%cr's.  Alas, we have an implicit equation for the predicted cr's which
%cannot be solved analytically.  Instead, we shall use the "miss" from the
%minimization routine.  We need the range of indices corresponding to each
%data set:
a=[1:length(Ta)];
b=[max(a)+1:max(a)+length(Tb)];
c=[max(b)+1:max(b)+length(Tc)];

figure(3)
plot(Ta,miss(a),'o',Tb,miss(b),'*',Tc,miss(c),'+')
hold on
plot(Trange,zeros(size(Trange)))
hold off
xlabel('Temperature (K)','FontSize',14)
ylabel('Residual (dimensionless fractional deviation)','FontSize',14)
title('Plot of Residuals','FontSize',14)
legend(['Cr0 = ',num2str(cr0a)],['Cr0 = ',num2str(cr0b)],['Cr0 = ',num2str(cr0c)],'
set(gca,'FontSize',14)
pause

%As you can see, there is a pretty significant systematic deviation between
%the model and the data.  That means that the random error in cr is
%-overestimated- (it's less than 3%) while assuming the data to be
%dominated by independent random error will lead to errors in fitting parameters
%to be underestimated!  It also means that there is something going on which is
%not captured by the model - not a big surprise.
pause

%OK, we still want to measure the sensitivity of the calculated values to
%errors in measured concentrations.  This can be done by taking the
%derivative of the fitted values with respect to each of the data points
%(not forgetting the initial concentrations, which have error too!).
%First, let's do the cr's:

crkeep=crpass;
gradfcr=zeros(3,length(Tpass)); %The array where we stuff the gradient.
for j=1:length(Tpass)
    crpass=crkeep;
    ep=crpass(j)*crstdev; %The amount we change the j'th data point by.
    crpass(j)=crpass(j)+ep;
    %Now we calculate new values of the fitted parameters:
    newguess=fminsearch('delta',guess);
    gradfcr(:,j)=(newguess-guess)/ep; %The gradient.
    echo off
end
echo on

%And we do the same for the initial concentrations:
crpass=crkeep;
```

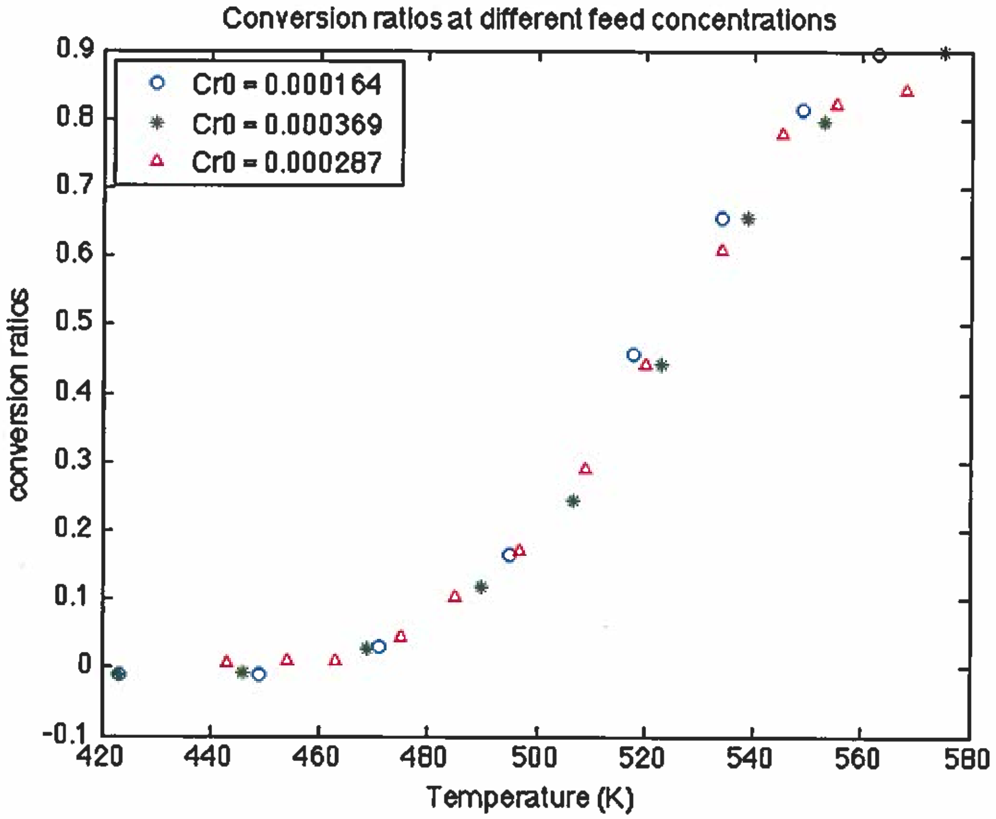
```
gradfcr0=zeros(3,3); %We had three initial concentrations.

cr0pass=[cr0a*(1+crstdev)*ones(size(Ta)),cr0b*ones(size(Tb)),cr0c*ones(size(Tc))];
gradfcr0(:,1)=(fminsearch('delta',guess)-guess)/(crstdev*cr0a);

cr0pass=[cr0a*ones(size(Ta)),cr0b*(1+crstdev)*ones(size(Tb)),cr0c*ones(size(Tc))];
gradfcr0(:,2)=(fminsearch('delta',guess)-guess)/(crstdev*cr0b);

cr0pass=[cr0a*ones(size(Ta)),cr0b*ones(size(Tb)),cr0c*(1+crstdev)*ones(size(Tc))];
gradfcr0(:,3)=(fminsearch('delta',guess)-guess)/(crstdev*cr0c);

%and we put cr0pass back again:
cr0pass=[cr0a*ones(size(Ta)),cr0b*ones(size(Tb)),cr0c*ones(size(Tc))];

pause

%That gives us the sensitivity gradients.  To complete the problem, we need
%to determine the matrix of covariance of the concentration measurements.
%This is a little more "iffy" because we -know- that they are not really
%random!  Still, if we make the randomness assumption, we can get an
%estimate of the matrix of covariance of the fitting parameters.

varcr=diag((crstdev*crpass).^2);
varcr0=diag(([cr0a,cr0b,cr0c]*crstdev).^2);
%Note that we multiply by the value of cr, etc., as crstdev was an
%estimate of the -fractional- standard deviation!
pause

%These will both contribute to the uncertainty.  We can look at each
%separately.  First, from cr:
var1=gradfcr*varcr*gradfcr'

%and now from cr0:
var2=gradfcr0*varcr0*gradfcr0'
pause

%Note that the error due to the initial concentrations is actually greater
%than that due to all the reactor outlet measurements put together!  That's
%because it is used in every calculated value of the fitting parameters,
%while the "randomness" of the outlet measurements gets averaged out.

%Putting these two together yields an estimate of the variance:
var=var1+var2

%and the fitting parameters + error:
[guess,diag(var).^.5]

%which, I would guess, gives a reasonable measure of the random uncertainty
%in the values.  This is because the uncertainty in cr0 (which is probably
%overestimated) dominates the calculation, while the "randomness
%assumption" underestimates the contribution due to error in cr.  The total
%error will be greater due to other contributions not considered here.  In
%particular, calibration errors will yield systematic error not observable
%from the residuals!
pause
```

```
%So, in conclusion, the fractional exponent lies within approximately two
%standard deviations of the literature value of 0.5, and the activation
%energy is also within two sigma of 38.5 (at Tr=298K).  Error estimates
%could be improved by having independent estimates of the uncertainty in
%cr0 (the dominant source), by modeling the matrix of covariance in cr,
%and by studying the effect of errors in the other parameters in the
%problem, such as T, qr, and m.  You can also study how the modeling
%parameters change if you leave cr0 as an additional adjustable parameter
%in the model, and simply add its normalized deviation from the measured
%value as an additional contribution to the objective function.  That would
%decrease the model dependence on these few particular data points, and
%might actually decrease the parameter error bars and reduce the systematic
%deviation in the residual.  To get the most out of your data, you need to
%think about the analysis procedure: what assumptions and relative
%weighting it is putting on particular data points.  You will have fun with
%this experiment senior year!
echo off
```

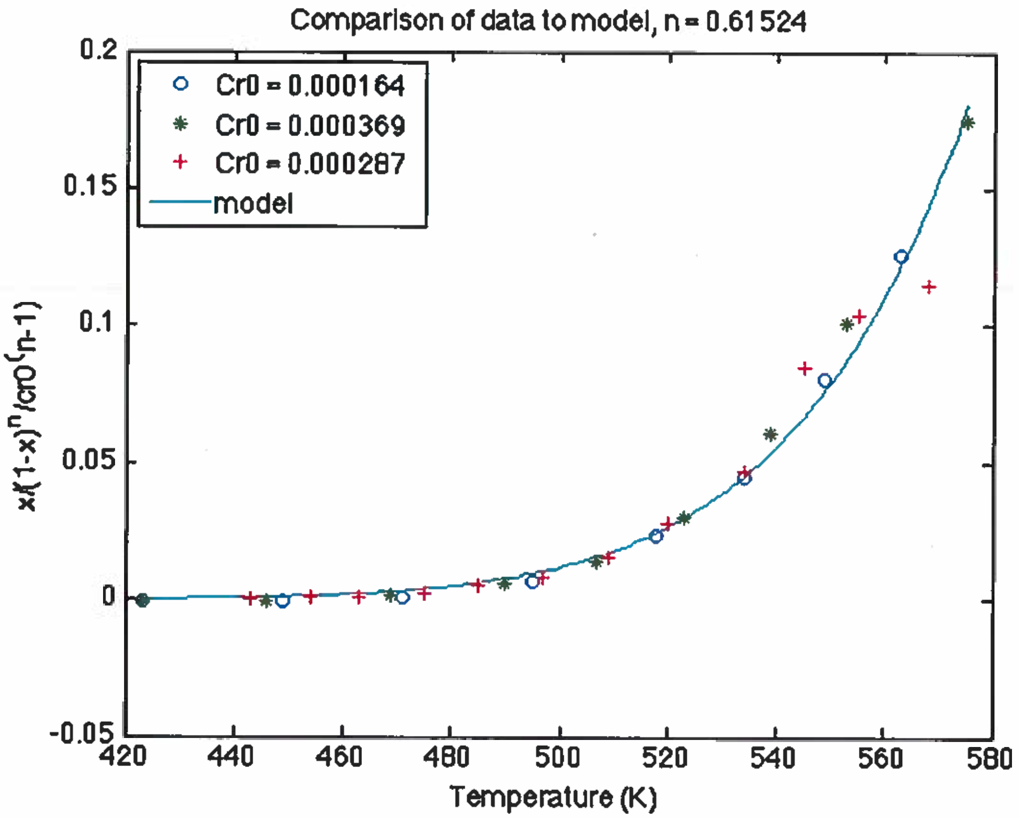
```
%In this example we analyze the catalytic oxidation data obtained by a
%group of students in senior lab.  Under CSTR conditions they measured outlet
%concentrations for three different reactor feed concentrations.  The data
%for the three feeds cr0 are given below:
Ta=[423 449 471 495 518 534 549 563];
cra=[1.66E-04   1.66E-04   1.59E-04   1.37E-04   8.90E-05   5.63E-05   3.04E-05
cr0a=1.64E-4;

Tb=[423 446 469 490 507 523 539 553 575];
crb=[3.73E-04   3.72E-04   3.59E-04   3.26E-04   2.79E-04   2.06E-04   1.27E-04
cr0b=3.69e-4;

Tc=[443 454 463 475 485 497 509 520 534 545 555 568];
crc=[2.85E-04   2.84E-04   2.84E-04   2.74E-04   2.57E-04   2.38E-04   2.04E-04
cr0c=2.87e-4;
pause

%We can calculate conversion ratios for these three runs:
xa=1-cra/cr0a;
xb=1-crb/cr0b;
xc=1-crc/cr0c;

%and we can plot them up:
figure(1)
plot(Ta,xa,'o',Tb,xb,'*',Tc,xc,'^')
xlabel('Temperature (K)','FontSize',14)
ylabel('conversion ratios','FontSize',14)
legend(['Cr0 = ',num2str(cr0a)],['Cr0 = ',num2str(cr0b)],['Cr0 = ',num2str(cr0c)],'I
title('Conversion ratios at different feed concentrations','FontSize',14)
set(gca,'FontSize',14)
pause

%Looking at this plot, we can immediately see why the standard technique
%for analyzing the reaction data will run into trouble: Even with
%interpolation, it will be very hard to get accurate values of the
%conversion at different concentrations for fixed temperatures.  To use
%non-linear regression to get at the fitting parameters, we will have to
```

```
%define an objective function for minimization, as well as some initial
%guesses for the parameters.  We can pass the data into the objective
%function using the "global" meat axe:
global crpass cr0pass Tpass

Tpass=[Ta,Tb,Tc];
crpass=[cra,crb,crc];
cr0pass=[cr0a*ones(size(Ta)),cr0b*ones(size(Tb)),cr0c*ones(size(Tc))];

%and you will have to save the function "delta.m" which returns the
%objective function to be minimized.
pause

%OK, let's do it.  We have the initial guesses:
guess=zeros(3,1);
guess(1)=.5; %This is the guess for n
guess(2)=15; %This is the guess for ln(k0)
guess(3)=38; %This is the guess for E/RTr

%And we go:
guess=fminsearch('delta',guess)
guess =
    0.6152
   15.0920
   36.0888
pause
%Looking at these values, they aren't too far off of those in the
%literature.  In particular, the exponent is quite close to the
%expected value of 0.5, and the activation energy is only off by 7%!
pause

%We can plot the model up too.  We need the other parameters
%(both here and in the function delta.m).
Tr=298; %The reference temperature.
qr=0.1; %The flow rate (liters/min)
m=1; %The amount of catalyst (g)

n=guess(1);
k0=exp(guess(2));
ERTr=guess(3);

Trange=[min(Tpass):max(Tpass)]; %A plotting range
xmodel=m/qr*(Tr./Trange).^n*k0.*exp(-ERTr*Tr./Trange);

xafn=xa./(1-xa).^n/cr0a^(n-1);
xbfn=xb./(1-xb).^n/cr0b^(n-1);
xcfn=xc./(1-xc).^n/cr0c^(n-1);

%OK, we've got the model and the data for the function x/(1-x)^n/cr0^(n-1).
%It should be independent of the concentration. Let's plot it up:
pause
figure(2)
plot(Ta,xafn,'o',Tb,xbfn,'*',Tc,xcfn,'+',Trange,xmodel)
xlabel('Temperature (K)','FontSize',14)
ylabel('x/(1-x)^n/cr0^(n-1)','FontSize',14)
legend(['Cr0 = ',num2str(cr0a)],['Cr0 = ',num2str(cr0b)],['Cr0 = ',num2str(cr0c)],'n
title(['Comparison of data to model, n = ',num2str(n)],'FontSize',14)
set(gca,'FontSize',14)
%Which shows that we get pretty much perfect collapse of the data.
pause

%Now we turn to the trickier error calculations.  First, we need to get a
%measure of the uncertainty in the concentration measurements.  We can get
%this from the magnitude of the "miss" in the data:
miss=crpass-cr0pass+m/qr*crpass.^n*k0.*(Tr./Tpass).^n.*exp(-ERTr*Tr./Tpass);

%We must adjust this to account for the relative weighting of the data.  In
%this case, a rough correction for the actual fractional deviation in cr
%is given by:
miss=miss./cr0pass.*(crpass./cr0pass);

%Thus we get the fractional standard deviation (assuming randomness) of:
crstdev=norm(miss)/(length(Tpass)-3)^.5
crstdev =
    0.0312

%Which yields a fractional error of around 3% - not too bad.  Note that these
%deviations could have been due to errors in the temperature just as readily!
pause

%OK, it is always important to plot up the residuals to see if the error is
%really random.  It is useful to plot up the actual cr's and predicted
%cr's.  Alas, we have an implicit equation for the predicted cr's which
%cannot be solved analytically.  Instead, we shall use the "miss" from the
%minimization routine.  We need the range of indices corresponding to each
%data set:
a=[1:length(Ta)];
b=[max(a)+1:max(a)+length(Tb)];
c=[max(b)+1:max(b)+length(Tc)];

figure(3)
plot(Ta,miss(a),'o',Tb,miss(b),'*',Tc,miss(c),'+')
hold on
plot(Trange,zeros(size(Trange)))
hold off
xlabel('Temperature (K)','FontSize',14)
ylabel('Residual (dimensionless fractional deviation)','FontSize',14)
title('Plot of Residuals','FontSize',14)
legend(['Cr0 = ',num2str(cr0a)],['Cr0 = ',num2str(cr0b)],['Cr0 = ',num2str(cr0c)],'n
set(gca,'FontSize',14)
pause

%As you can see, there is a pretty significant systematic deviation between
%the model and the data.  That means that the random error in cr is
%-overestimated- (it's less than 3%) while assuming the data to be
%dominated by independent random error will lead to errors in fitting parameters
%to be underestimated!  It also means that there is something going on which is
%not captured by the model - not a big surprise.
pause

%OK, we still want to measure the sensitivity of the calculated values to
%errors in measured concentrations.  This can be done by taking the
```

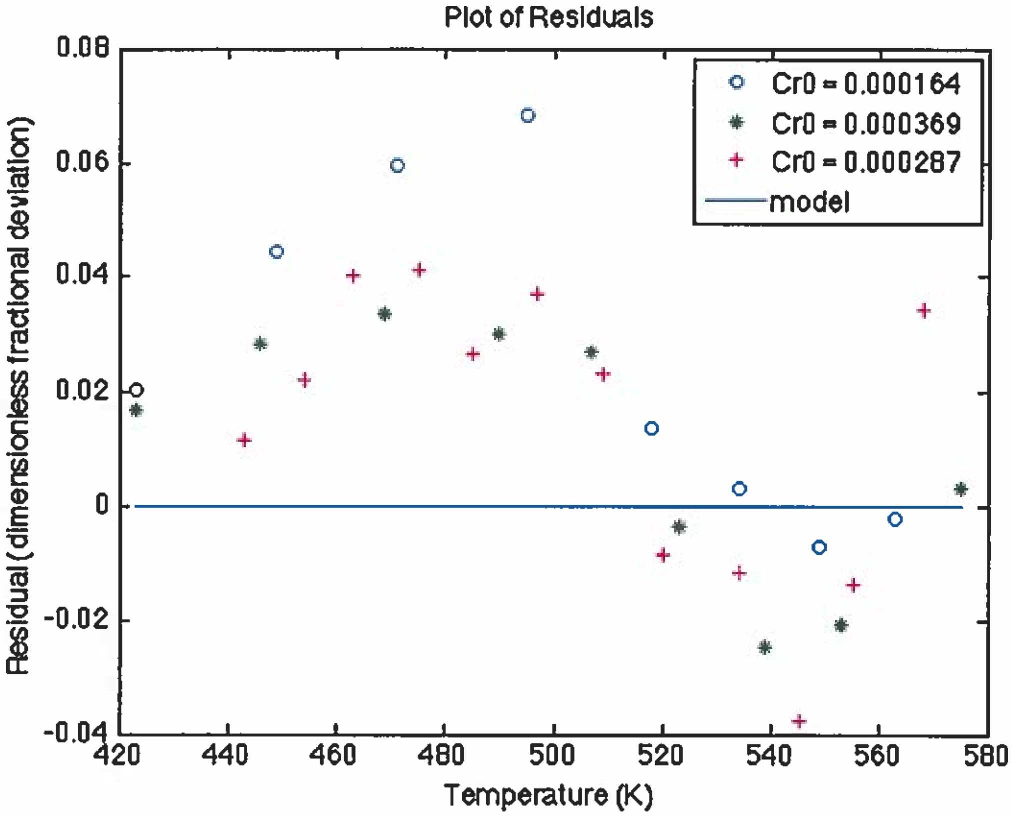
```
%derivative of the fitted values with respect to each of the data points
%(not forgetting the initial concentrations, which have error too!).
%First, let's do the cr's:
crkeep=crpass;
gradfcr=zeros(3,length(Tpass)); %The array where we stuff the gradient.
for j=1:length(Tpass)
    crpass=crkeep;
    ep=crpass(j)*crstdev; %The amount we change the j'th data point by.
    crpass(j)=crpass(j)+ep;
    %Now we calculate new values of the fitted parameters:
    newguess=fminsearch('delta',guess);
    gradfcr(:,j)=(newguess-guess)/ep; %The gradient.
    echo off

%And we do the same for the initial concentrations:
crpass=crkeep;
gradfcr0=zeros(3,3); %We had three initial concentrations.

cr0pass=[cr0a*(1+crstdev)*ones(size(Ta)),cr0b*ones(size(Tb)),cr0c*ones(size(Tc))];
gradfcr0(:,1)=(fminsearch('delta',guess)-guess)/(crstdev*cr0a);

cr0pass=[cr0a*ones(size(Ta)),cr0b*(1+crstdev)*ones(size(Tb)),cr0c*ones(size(Tc))];
gradfcr0(:,2)=(fminsearch('delta',guess)-guess)/(crstdev*cr0b);

cr0pass=[cr0a*ones(size(Ta)),cr0b*ones(size(Tb)),cr0c*(1+crstdev)*ones(size(Tc))];
gradfcr0(:,3)=(fminsearch('delta',guess)-guess)/(crstdev*cr0c);

%and we put cr0pass back again:
cr0pass=[cr0a*ones(size(Ta)),cr0b*ones(size(Tb)),cr0c*ones(size(Tc))];

pause

%That gives us the sensitivity gradients.  To complete the problem, we need
%to determine the matrix of covariance of the concentration measurements.
%This is a little more "iffy" because we -know- that they are not really
%random!  Still, if we make the randomness assumption, we can get an
%estimate of the matrix of covariance of the fitting parameters.
varcr=diag((crstdev*crpass).^2);
varcr0=diag(([cr0a,cr0b,cr0c]*crstdev).^2);
%Note that we multiply by the value of cr, etc., as crstdev was an
%estimate of the -fractional- standard deviation!
pause

%These will both contribute to the uncertainty.  We can look at each
%separately.  First, from cr:
var1=gradfcr*varcr*gradfcr'
var1 =
    0.0005    0.0117    0.0127
    0.0117    0.3550    0.4275
    0.0127    0.4275    0.5431

%and now from cr0:
var2=gradfcr0*varcr0*gradfcr0'
var2 =
    0.0027    0.0697    0.0759
    0.0697    1.8849    2.1491
    0.0759    2.1491    2.5335
pause

%Note that the error due to the initial concentrations is actually greater
%than that due to all the reactor outlet measurements put together!  That's
%because it is used in every calculated value of the fitting parameters,
%while the "randomness" of the outlet measurements gets averaged out.
%Putting these two together yields an estimate of the variance:
var=var1+var2
var =
    0.0032    0.0814    0.0886
    0.0814    2.2400    2.5766
    0.0886    2.5766    3.0765

%and the fitting parameters + error:
[guess,diag(var).^.5]
ans =
    0.6152    0.0565
   15.0920    1.4966
   36.0888    1.7540

%which, I would guess, gives a reasonable measure of the random uncertainty
%in the values.  This is because the uncertainty in cr0 (which is probably
%overestimated) dominates the calculation, while the "randomness
%assumption" underestimates the contribution due to error in cr.  The total
%error will be greater due to other contributions not considered here.  In
%particular, calibration errors will yield systematic error not observable
%from the residuals!
pause

%So, in conclusion, the fractional exponent lies within approximately two
%standard deviations of the literature value of 0.5, and the activation
%energy is also within two sigma of 38.5 (at Tr=298K).  Error estimates
%could be improved by having independent estimates of the uncertainty in
%cr0 (the dominant source), by modeling the matrix of covariance in cr,
%and by studying the effect of errors in the other parameters in the
%problem, such as T, qr, and m.  You can also study how the modeling
%parameters change if you leave cr0 as an additional adjustable parameter
%in the model, and simply add its normalized deviation from the measured
%value as an additional contribution to the objective function.  That would
%decrease the model dependence on these few particular data points, and
%might actually decrease the parameter error bars and reduce the systematic
%deviation in the residual.  To get the most out of your data, you need to
%think about the analysis procedure: what assumptions and relative
%weighting it is putting on particular data points.  You will have fun with
%this experiment senior year!
echo off
```



Conversion ratios at different feed concentrations



Comparison of data to model, n = 0.61524



Published with MATLAB® 7.12