Chemical Reaction Engineering - Part 7

Richard K. Herz, rherz@ucsd.edu, www.ReactorLab.net

Methods of experimental data analysis

So far we have been given rate equations and values of rate coefficients. For new reactions or old reactions under new conditions or catalyzed by new catalysts, we may have to do experiments and determine rate equations ourselves.

This aspect of CRE is called kinetics or kinetic studies.

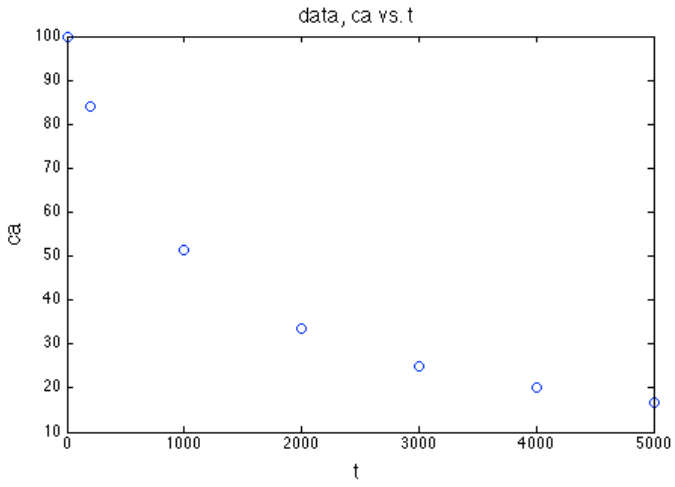
The biggest challenge in doing kinetic studies is choosing a good method to do chemical analysis and determine concentrations or conversion vs. reaction time. The speed at which we must make measurements is a consideration. Reactions that near completion over a few minutes or hours are easier to measure than very fast or very slow reactions. Depending on the system we might have the choice of chromatography, mass spectrometry, spectrophotometry or other method. Affordability is a consideration because some of these methods use expensive instruments. The math, which we focus on here, is the simplest aspect.

Reactor selection is also a consideration. Here we specify use of a well-mixed, isothermal batch reactor. That may be easy to provide for a relatively slow, liquid-phase reaction with low heat of reaction. It will be harder, e.g., for fast, highly exothermic gas-phase reactions over solid catalysts. Controlling heat transfer for energetic reactions in order to provide isothermal conditions will be a challenge. For solid catalysts, batch reactor conditions can be approximated by rapidly pumping a recycle of fluid over a small bed of catalyst.

There are two basic approaches we can take to analyze the data from isothermal batch reactors: the integral method and the differential method.

Example data

We did some experiments in Reactor Lab, Division 1, Lab 1. The reactor is a constant volume, isothermal batch reactor for the reaction A → B. Here is a plot of C_A vs t. There is a small amount of random scatter in the data.



Integral method of kinetic data analysis

In this method, we start with the functional form of a rate equation. The rate equation may be given to us, or we propose a functional form ourselves, e.g., first-order or second-order. Then we put the rate equation into the balance equation and integrate the balance equation. Finally, we determine if the integrated equation can "fit the data" and, if it does fit, determine the values of the rate coefficients. By "fit the data" we mean that the rate equation can predict the how the experimental data change as reaction time or initial concentrations are changed.

1st-order rate equation and integral method

First let's look at a 1st-order reaction. We may already know the reaction is 1st order but not know the value of the rate coefficient, or we may propose this rate equation if we don't know the equation. The reaction is either essentially irreversible, or we take data at early times such that the back rate is negligible.

Specify that we have obtained a table of t_{rxn} and C_A values.

$$\frac{dN_A}{dt} = r_A V = -k C_A V$$

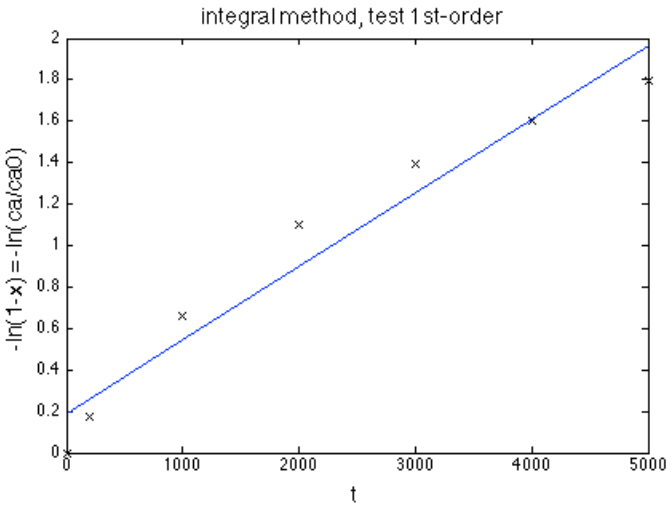
Here specify a constant volume reactor.

$$\frac{dC_A}{dt} = -k C_A \quad ; \quad t=0 \; ; \; C_A(0) = C_{A,0}$$

What's the next step? Integrate - this is the integral method after all.

$$-\ln\left(C_A / C_{A,0}\right) = k\, t_{rxn}$$

Now test to see if this fits the data by plotting $-\ln(C_A/C_{A0}) = -\ln(1 - X_A)$ for constant V, vs. t_{rxn}. If the data points fall along a straight line, then we can conclude that a 1st-order rate equation fits the data. The slope of a line fit through the data has a slope equal to the value of k. In a section below we will discuss a couple methods of fitting straight lines to data. For our example data, it looks like the reaction is not first order.



If the data plot this way can't be fit by a straight line, then we need to try another rate equation.

All experimental measurements will contain some error. Hopefully this error is only random errors or "random scatter" with a relatively narrow distribution about the average value for repeat measurements. So we are looking at the trends in the data on the plot, not that a straight line goes exactly through each point.

We worked the math using concentrations as an example. You also need to be able to do this using conversion.

2nd-order rate equation and integral method

Specify a constant volume reactor.

$$\frac{dC_A}{dt} = -kC_A^2 \quad ; \quad t=0 \; ; \; C_A(0) = C_{A,0}$$

Integrate

$$\left(\frac{1}{C_A} - \frac{1}{C_{A,0}}\right) = k\,t_{rxn}$$

Now test to see if this fits the data by plotting ($1/C_A$- $1/C_{A0}$) vs. t_{rxn}. If the data points fall along a straight line, then we can conclude that a 2nd-order rate equation fits the data. The slope of a line fit through the data has a slope equal to the value of k.

It looks like our example data agrees with 2nd-order kinetics. The slope of the line is k = 1.0 x 10^{-5} m^3/mol/s.

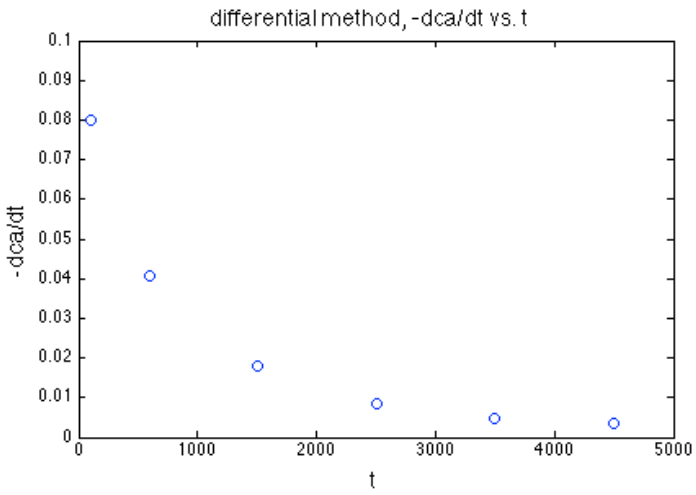
If the data plot this way can't be fit by a straight line, then we need to try another rate equation.

Note that reaction orders may not be integers.

Differential method of kinetic data analysis

In this method, we collect concentration vs. time data, e.g., C_A vs. t_{rxn}, and then differentiate the data. That is, determine the slopes dC_A/dt (or dX_A/dt) at a series of different times. At each of these times, we have measurements (or interpolated measurements) of fluid composition, e.g., C_A.

Here is a plot of estimates of $-dC_A/dt$ vs. t for the example data we used above.



For a constant volume, isothermal batch reactor

$$\frac{dC_A}{dt} = r_A$$

So we now have estimates of the reaction rate at known compositions. For example, we have a table of r_A and C_A values.

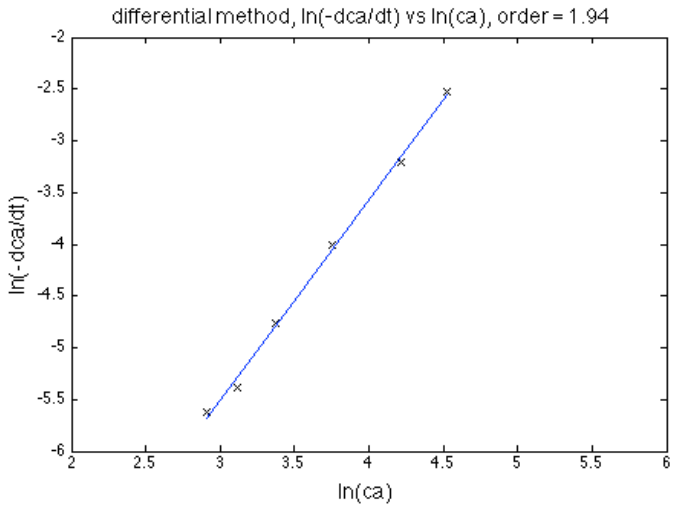
The next step is to find a rate function that can fit the data.

A special case is the power-law rate equation. For an essentially irreversible reaction of order n

$$r_A = kC_A^n$$

$$\ln(r_A) = \ln(k) + n\ln(C_A)$$

Plot points of $\ln(r_A)$ vs. $\ln(C_A)$ then fit a straight line through the points to get $\ln(k)$ and the order n.



The slope of the line is the estimate of the reaction order, n = 1.94. The intercept is ln(k). From this plot, k = 1.2 x 10^{-5} m^3/mol/s. These estimates are close to the results we got using the integral method: n = 2, k = 1.0 x 10^{-5} m^3/mol/s. There is only a small amount of random error in these data.

Comparison of the methods

For this power-law rate equation, it is interesting to compare the integral and differential methods. The integral method requires you to separately test each guessed value of *n* but you don't have to differentiate the data. The differential method allows you to determine *k* and *n* from one plot but, in order to get that plot, you have to process - differentiate - the data. You should know how to do both methods.

If you have a new reaction with an unknown rate equation, you may be able to propose a rate equation from your knowledge of the chemistry. Then you might have luck using the integral method.

For complex reaction kinetics, it might be advisable to use the differential method to see how the rate depends on the concentrations of the different components in the reaction mixture. Here we use an example where only C_A affects the rates but, in general, there may be more components involved in a rate equation.

Data used to generate plots above

t	ca	dca	dt	-dca/dt	cap (ave. ca)
0	100				
		-16	200	0.08	92
200	84				
		-32.5	800	0.0406	67.75
1000	51.5				
		-18.1	1000	0.0181	42.45
2000	33.4				
		-8.6	1000	0.0086	29.1
3000	24.8				
		-4.6	1000	0.0046	22.5
4000	20.2				
		-3.6	1000	0.0036	18.4
5000	16.6				

Matlab script used to generate plots above

Virtual experiments were performed in Reactor Lab, Division 1, Lab 1, Batch reactor. The data obtained were pasted into Matlab and the header info commented out.

```
% D1L1 Batch, Data Set 5b
%
% k*_5b remained constant at    1.00E-6  (value at 300 K)
% Ea_5b remained constant at       40.0  kJ/mol
% n_5b remained constant at       2.00  (dimensionless)
% T_5b remained constant at      350.0  K
% Cin_5b remained constant at      100.0  mol/m3
% vol_5b remained constant at       1.00  m3
% time_5b has units of s
% tau_5b has units of s
% Cout_5b has units of mol/m3
% conv_5b has units of (dimensionless)
%
% time_5b   tau_5b  Cout_5b conv_5b

d = [
200.0   200.0   84.0    0.160
1000    1000    51.5    0.485
2.00E3  2.00E3  33.4    0.666
3.00E3  3.00E3  24.8    0.752
4.00E3  4.00E3  20.2    0.798
5.00E3  5.00E3  16.6    0.834
];

% extract columns from array d
t = d(:,1);
ca = d(:,3);
x = d(:,4);

% add initial conditions
t = [0; t];
ca0 = 100;
ca = [ca0; ca];
x = [0; x];

figure(1), plot(t,ca,'bo')
title('data, ca vs. t')
ylabel('ca')
xlabel('t')

% integral method, test 1st-order
y = -log(1-x);

% fit straight line (1st order polynomial)
coefn1 = polyfit(t,y,1); % get coef of straight line (poly order = 1)
yfit = polyval(coefn1,t);

figure(2), plot(t,y,'kx',t,yfit)
title('integral method, test 1st-order')
ylabel('-ln(1-x) = -ln(ca/ca0)')
xlabel('t')

% integral method, test 2nd-order
y = (1 ./ ca - 1 / ca0); % << note ./ since ca is an array

% fit straight line (1st order polynomial)
coefn2 = polyfit(t,y,1); % get coef of straight line (poly order = 1)
yfit = polyval(coefn2,t);
```

```
figure(3), plot(t,y,'kx',t,yfit)
title('integral method, test 2nd-order')
ylabel('(1/ca - 1/ca0)')
xlabel('t')

% differential method

% do simplest method here, difference between neighbor pairs of points
% using matlab diff() function
dca = diff(ca);
dt = diff(t);
dcadt = dca ./ dt;

% diff() returns differences between neighbor pairs of points
% and so returns one fewer element in array
% get tp and cap (p = prime) at midpoint of each interval
[row col] = size(t);
tp = t(1:(row-1),:);
tp = tp + 0.5*dt;
cap = ca(1:(row-1),:);
cap = cap + 0.5*dca;

figure(4), plot(tp,-dcadt,'bo')
title('differential method, -dca/dt vs. t')
axis([0 5000 0 0.1]) % adjust after seeing plot first time
ylabel('-dca/dt')
xlabel('t')

% fit straight line (1st order polynomial)
coef = polyfit(log(cap),log(-dcadt),1)
yfit = polyval(coef,log(cap));

figure(5), plot(log(cap),log(-dcadt),'kx', log(cap),yfit,'b')
tt = sprintf('differential method, ln(-dca/dt) vs ln(ca), order = %4.2f',...
    coef(1))
axis([2 6 -6 -2]) % adjust after seeing plot first time
title(tt)
ylabel('ln(-dca/dt)')
xlabel('ln(ca)')
hold off
```

Getting activation energy from two or more k values

Knowing the values of a rate coefficient at two temperatures allows us to estimate an activation energy. We can do this for *k* values and also the *K's* in LHHW and Michaelis-Menten rate equations.

$$k = A e^{-E/R/T}$$

$$\frac{k(T_2)}{k(T_1)} = \frac{A e^{-E/R/T_2}}{A e^{-E/R/T_1}}$$

$$k(T_2) = k(T_1) e^{-(E/R)(1/T_2 - 1/T_1)}$$

$$-\frac{E}{R} = \frac{\ln k(T_2) - \ln k(T_1)}{1/T_2 - 1/T_1}$$

Doing this, we are essentially putting a straight line through two points on an "Arrhenius plot," which is a plot of ln(*k*) vs. 1/*T*, to get the slope (-*E*/*R*).

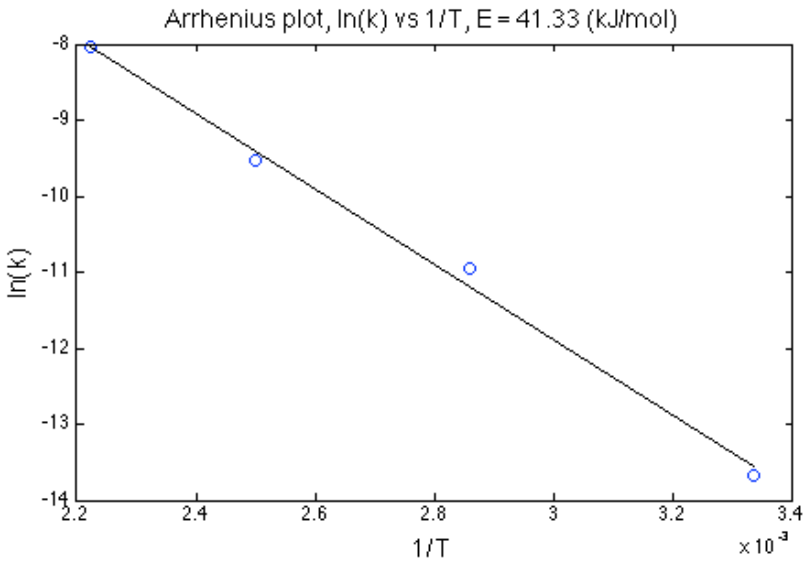
Once we have a value for *E* we can then estimate the value of the rate coefficient at a third temperature.

$$k(T_3) = k(T_1)e^{-(E/R)(1/T_3 - 1/T_1)}$$

Because of random scatter in the experimental data, you should use more than two *k* values. With multiple *k* and *T* values over the T-range of interest, make an Arrhenius plot, fit a straight line to a section of data that can be fit by a straight line, and determine *E* from the slope and *A* from the intercept on the y-axis (for x-axis $1/T \rightarrow 0$).

Here is an Arrhenius plot that was generated using the same lab in Reactor Lab and input values as above. The left-most point is at 450 K and the right-most point is at 300 K.

Note that if you estimate *E* from *k* at only two temperatures, you would get a different estimate of the *E* value.



Here is the Matlab script that was used to generate this plot

```
% Arrhenius plot
% k*_5b remained constant at    1.00E-6  (value at 300 K)
% Ea_5b remained constant at       40.0  kJ/mol
R = 8.3145' % gas constant
T = 300:50:450;
k = 1.0e-6*exp(-(40e3/R)*(1./T - 1/300));
% add some random scatter to k values to simulate real experiments
kr = k .* (1 + rand(1,length(k)));
lnk = log(kr);
invT = 1./T;
% fit straight line (polynomial of order 1)
coef = polyfit(invT,lnk,1);
% 1st-order polynomial coef(1) is slope (-E/R)
E = -coef(1)*R; % J/mol
E = E * 1e-3; % convert to kJ/mol
```

```
lnkp = polyval(coef,invT); % calc model values (p = prime)
plot(invT, lnk,'bo',invT,lnkp,'k')
tt = sprintf('Arrhenius plot, ln(k) vs 1/T, E = %4.2f (kJ/mol)',E)
title(tt), ylabel('ln(k)'), xlabel('1/T')
```

During analysis of experimental data, it may be preferable to first determine a value of k with a high confidence at a reference temperature in the middle of the range of interest, and then use this equation and values of k determined at other temperatures in a regression algorithm to estimate E. This procedure minimizes parameter-correlation problems in estimates of A and E when fitting data with the Arrhenius equation.

Fitting a straight line to experimental data

What does this mean? Often we use Matlab's `polyfit()` function or add "trendlines" in Excel. But what is going on inside the computer when we do that? Or when we "eye ball" it by hand, for that matter?

Consider that we have n pairs (data points) of a and b data. Here, a could be $(1/T)$ values in an Arrhenius plot, and b could be $\ln(k)$ values. Or a and b could be $\ln(C_A)$ and $\ln(-dC_A/dt)$ values in the differential method analysis of a power-law rate equation.

We want to find the values of the slope and intercept of the straight line equation that minimize that the sum of the squared differences between the line's (model's) prediction of a b value and the experimental b value at each a value.

$$\text{minimize} \sum_{i=1}^{i=n} \left(b_i^{\text{model}} - b_i^{\text{exp.}} \right)^2$$

This is only one possible measure of goodness of fit but it is a common one. It is commonly referred to as a "least squares fit." We square the differences or "errors" so that negative errors don't cancel out positive errors and give a sum near zero in the presence of large errors. Squaring also weights larger errors more than smaller errors.

What if we only have two a,b pairs (two data points)? Two points exactly define a straight line, so we can find the slope and intercept that give a sum of squared errors of zero.

The trouble is that most experimental data has some random error in it, even if the underlying chemistry and physics can be described by a straight line. So we want to take more than two data points - many more if possible. The number we need depends on the amount of error in the data and is the subject of a course on probability and statistics. We won't go into too much detail here.

Here is the equation for a straight line. The slope is x_1 and the intercept is x_2 .

$$a_i x_1 + x_2 = b_i \quad \text{or} \quad a_i^1 x_1 + a_i^0 x_2 = b_i \quad \text{since} \quad a_i^0 = 1$$

For n = 2 data points (a,b pairs), we can write this in vector-matrix form

$$\begin{bmatrix} a_1^1 & a_1^0 \\ a_2^1 & a_2^0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$\begin{bmatrix} a_1^1 & 1 \\ a_2^1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$\mathbf{A}_{2,2}\vec{x} = \vec{b}$$

This is so simple that we could use Gaussian elimination. But the general linear algebra solution is

$$\vec{x} = \mathbf{A}_{2,2}^{-1}\vec{b}$$

Now write the matrices for $n > 2$

$$\begin{bmatrix} a_1^1 & a_1^0 \\ a_2^1 & a_2^0 \\ . & . \\ a_n^1 & a_n^0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ b_n \end{bmatrix}$$

$$\mathbf{A}_{n,2}\vec{x} = \vec{b}$$

With more than two data points, **A** isn't be square and can't be inverted. And Gaussian elimination won't work.

This solution for vector *x* gives the minimum sum-of-squared errors between the computed model *b* values and experimental *b* data. That is, it gives a least squares fit.

$$\vec{x} = \left[\mathbf{A}_{2,n}^T \mathbf{A}_{n,2}\right]^{-1}\left[\mathbf{A}_{2,n}^T \vec{b}\right]$$

The result of matrix-multiplying the transpose of **A** times **A** is a square matrix (2x2 in this case) and is therefore invertible.

Here is the Matlab script for this equation. An apostrophe is the transpose operator. The standard function `inv` returns the inverse of the argument. The `*` multiplication operators without preceding dots specify matrix multiplication (not element-by-element array multiplication specified by `.*`).

```
x = inv(A'*A)*A'*b
```

A better way to find the solution in Matlab is the use the divide from left operator \ , which chooses from a set of algorithms and can solve problems which may be difficult to invert.

```
x = A\b
```

Here we discuss straight line fits but this procedure can be used for fitting any other type of polynomial. Remember that a straight line is a polynomial of order 1. A polynomial of order two or higher is a nonlinear equation... in the *a* data, that is. But we know the values of the *a* data. In terms of the coefficients *x* of the polynomial, a polynomial equation is linear (*x*'s raised to power only 1)

$$a_i^n x_1 + a_i^{n-1} x_2 + ... + a_i^1 x_n + a_i^0 x_{n+1} = b_i$$

and the unknown coefficients *x* can be solved for using linear algebra.

Matlab's `polyfit()` function returns the coefficients x_i in this order: coefficient x_1 = `x(1)` of highest order term to the "constant" x_{n+1} = `x(n+1)`.

This is the principle of the math that is going on behind the scenes when you use Matlab's `polyfit()` function or add "trendlines" in Excel.

LHHW equation and integral method

Specify a constant volume reactor. LHHW equations are used over solid "heterogeneous "catalysts. The LHHW rate equation used as an example here has the same functional form as a Michaelis-Menton equation for enzyme-catalyzed reactions. These types of equations were introduced in Part 1 of these notes. This is one example.

$$\frac{dC_A}{dt} = \frac{-kC_A}{1 + K_A C_A}$$

$$\left(\frac{1}{C_A} + K_A\right) dC_A = -k\,dt \quad ; \quad t = 0 \; ; \; C_A(0) = C_{A,0}$$

$$-\ln\left(C_A / C_{A,0}\right) + K_A\left(C_{A,0} - C_A\right) = k\,t_{rxn}$$

$$\frac{-\ln\left(C_A / C_{A,0}\right)}{\left(C_{A,0} - C_A\right)} = k\,\frac{t_{rxn}}{\left(C_{A,0} - C_A\right)} - K_A$$

$$\text{plot } \frac{-\ln\left(C_A / C_{A,0}\right)}{\left(C_{A,0} - C_A\right)} \text{ vs. } \frac{t_{rxn}}{\left(C_{A,0} - C_A\right)}$$

The slope is *k* and the y-intercept is $-K_A$.

We can also use linear algebra to solve for the unknowns. Let's say we have made *n* measurements. We have this equation for each data point (measurements at one t_{rxn}).

$$k\,t_{rxn} + (-K_A)(C_{A,0} - C_{A,}) = -\ln\left(C_A / C_{A,0}\right)$$

The matrix equation is

$$\begin{bmatrix} t_{rxn,1} & (C_{A,0}-C_{A,})_1 \\ t_{rxn,2} & (C_{A,0}-C_{A,})_2 \\ . & . \\ t_{rxn,n} & (C_{A,0}-C_{A,})_n \end{bmatrix} \begin{bmatrix} k \\ -K_A \end{bmatrix} = \begin{bmatrix} -\ln(C_A/C_{A,0})_1 \\ -\ln(C_A/C_{A,0})_2 \\ . \\ -\ln(C_A/C_{A,0})_n \end{bmatrix}$$

$$\mathbf{A}_{n,2}\vec{x}=\vec{b}$$

We can solve it using the method described above.

LHHW rate equation and the differential method

For an isothermal, constant volume batch reactor

$$\frac{dC_A}{dt}=r_A=\frac{-kC_A}{1+K_AC_A}$$

$$\frac{1}{-r_A}=\left(\frac{1}{k}\right)\frac{1}{C_A}+\left(\frac{K_A}{k}\right)$$

A plot of inverse rate vs. inverse concentration will fall along a straight line if this rate equation fits the data. The slope is (1/*k*) and the intercept is (K_A/*k*).

Additional resources

Schmidt's book - on library reserve & online via `http://roger.ucsd.edu/record=b7179459~S9`
Chapter 2 - sections 18 and 19

Levenspiel's book - on library reserve
Chapter 3

Fogler's *Essentials* book - on library reserve
Chapter 7

Also see Fogler's web resources at `http://www.umich.edu/~essen/html/07chap/frames.htm`