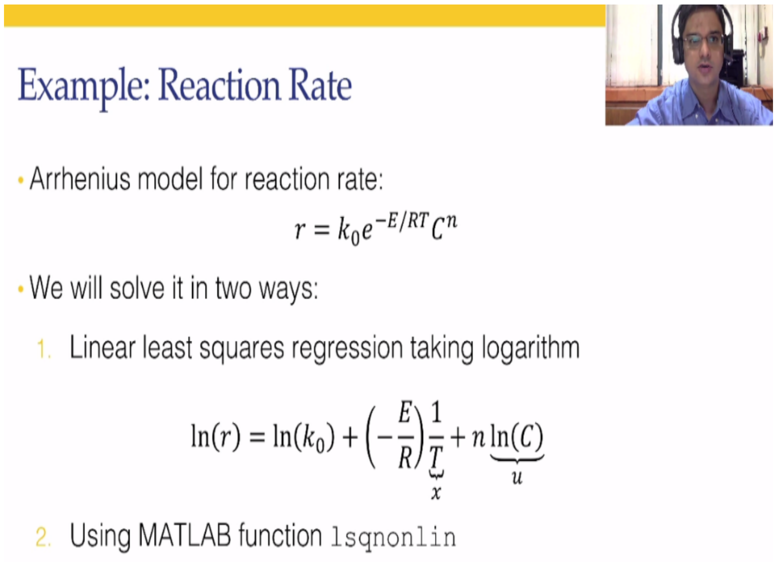
Matlab Programming for Numerical Computation
Dr. Niket Kaisare
Department Of Chemical Engineering
Indian Institute of Technology, Madras

Module No. #06
Lecture No. #6.3
Regression and Interpolation – Functional and nonlinear regression

Hello and welcome to MATLAB programming for numerical computations. We are in module 6. In this module we have been covering regression and interpolation. In the first lecture of this module we covered introduction to regression and interpolation. In the previous lecture we covered linear least squares regression and took up an example and solved it using a matrix method for linear least square.

In this lecture we are going to take a physically relevant example and show how we can do linear and nonlinear regression. We will take example of rate of reaction, the Arrhenius form that we have studied in our eleventh and twelfth grades. We will take an example, for the Arrhenius form and show we can use linear or nonlinear regression in order to obtain the parameters for the rate of reaction.
(Refer Slide Time: 01:07)



Example: Reaction Rate

- Arrhenius model for reaction rate:

$$r = k_0 e^{-E/RT} C^n$$

- We will solve it in two ways:
  1. Linear least squares regression taking logarithm

$$\ln(r) = \ln(k_0) + \left(-\frac{E}{R}\right)\underbrace{\frac{1}{T}}_{x} + n\underbrace{\ln(C)}_{u}$$

  2. Using MATLAB function `lsqnonlin`

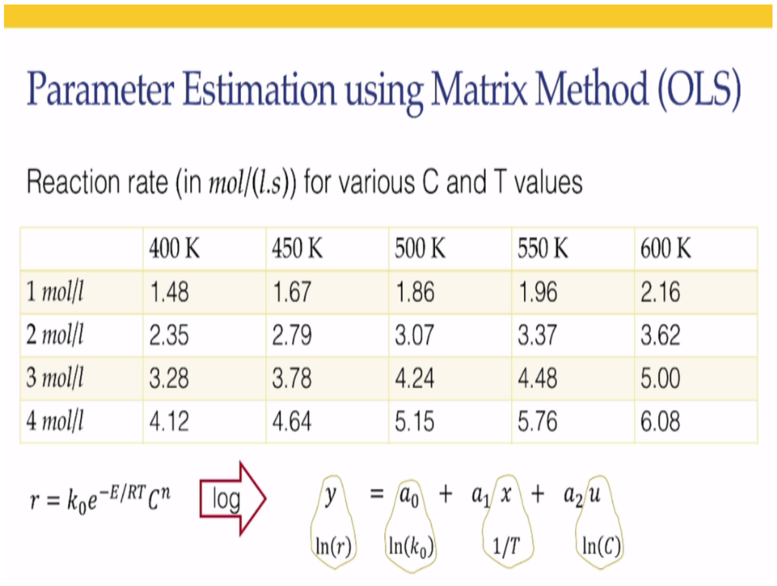
So the Arrhenius model for reaction rate is given by rate r = k0 * e ^ -e /rt. Where, e is the activation energy and r is the gas constant multiplied by concentration c to the power n. So k0 * e

^ -e / rt c ^ n. Now we will solve this problem in 2 different ways. First is using linear least squares regression. Now this is clearly not a linear form of equation. You see, you have your k0, you have your the other parameters is e / r and the third parameter is n.

They are not related to each other in a linear fashion. However if we take a logarithm of this, we will get logarithm of r = logarithm of k0 + negative e/r * 1 / T + n * ln(c). So if we were to write 1 / T = x, we were to write ln(c) = u. We will get this in the form y=a 0+a1 * x+a2 * u. where, our a0 is nothing but log of k0, our a1 is nothing but-e / r and our a2 is nothing but our n okay.

So this has a form which is linear in the parameters that we are interested in finding. Note that when we talked about linear regression it or nonlinear regression, we mean that it is linear in the parameters or nonlinear in the parameters respectively. It does not matter whether it is linear or nonlinear in the measured quantities. That is in this case temperature; concentration and rate of reactions are measured.

It does not matter whether it is linear or nonlinear in these quantities. What is important that is that, it should be linear in a0, a1, a2 and so on. It is indeed linear in a0, a1, a2. So first we will solve this particular problem using linear least squares and then we will solve it using nonlinear least squares using the MATLAB function lsqnonlin. lsqnonlin is a function that is available in MATLAB’s optimization toolbox.
(Refer Slide Time: 03:52)



Parameter Estimation using Matrix Method (OLS)

Reaction rate (in $mol/(l.s)$) for various C and T values

	400 K	450 K	500 K	550 K	600 K
1 mol/l	1.48	1.67	1.86	1.96	2.16
2 mol/l	2.35	2.79	3.07	3.37	3.62
3 mol/l	3.28	3.78	4.24	4.48	5.00
4 mol/l	4.12	4.64	5.15	5.76	6.08

$$r = k_0 e^{-E/RT} C^n \xrightarrow{\log} \underbrace{y}_{\ln(r)} = \underbrace{a_0}_{\ln(k_0)} + a_1 \underbrace{x}_{1/T} + a_2 \underbrace{u}_{\ln(C)}$$

So you will have an access to that function only if you have access to the optimization toolbox. So let us go ahead and solve this problem in 2 ways for the following data. So the data is obtained at 5 different temperatures 400, 450, 500,550 and 600 Kelvin's and 4 different concentration 1 mole per liter, 2 moles per liter, 3 moles per liter and 4 moles per liter and these is the rate of reaction given in moles per liter per second.

If we take logarithm of the rate, we are going to get the expression of the form y=a 0+a1 x+a2 u okay. So y is log (r), x is 1 / t and u is log (c) and that is what we will use in order to do the linear regression. Let us go to MATLAB and do that okay.

(Video Starts: 04:37) So what I have done already is, written down a snippet of code for obtaining the data in concentration and temperature. So let us see what data that we have. So let us, when we run this, I have given a command to display xData and yData and let us go ahead and do that okay. So this is basically our independent variables concentration and independent variable temperature and our dependent variable rate of reaction.

Let us look at the table that we have had so far. 1 moles per liter and 400 kelvins it was 1. That is 48 for 2 moles per liter, 400 Kelvin's. It was 2.35 for 1 and 400, 1.48, 2 and 2 and 400, 2.35 and 3 and 400, 3.28 for 3 and 400 it is 3.28 so on and so forth. So if so the temperatures are given as xData2 and concentrations are given as xData1, the first column is concentrations, the second column is temperatures and yData is nothing but our rate of reactions okay.

So what we want to do is, get our x is nothing but 1/ T. Our u is nothing but log (c). So let us go ahead and do that in MATLAB first okay. So let me delete, sorry let me delete this line, as the instructions. Delete this line before having the code. The reason why I had this line over here is, to display to you what the data looks like okay.

So our concentration is going to be, yeah our concentration is xData column 1, our temperature is xData column 2. What we first need to do, we want to call our x as 1 / T. So x=1./ T and t is xData. All the row’s first, sorry second column okay. That is our x. Our small u is small, u is the first column of xData. Logarithm of that u is logarithm(C).

C is the first column of xData. So u is going to be logarithm of our concentrations. Concentration is nothing but xData all the row’s first column okay. So that is our x and u. Our y is nothing but logarithm of rate. Rate is n yData and our y is nothing but logarithm of rate. So we will put that in log (yData) okay.

So let us run up to the stage and see whether we have xu and y correctly or if there is some error, MATLAB will throw us an error okay. So there is no error, so, xu and y has probably been obtained correctly. Let us look at our x, x is 1/temperature. So we should be getting 1/400, 1/450 and so on. So let us type x and see what the values are. So 1/400 is 0.0025, 1/500 is 0.0020. So this looks like its correct okay.

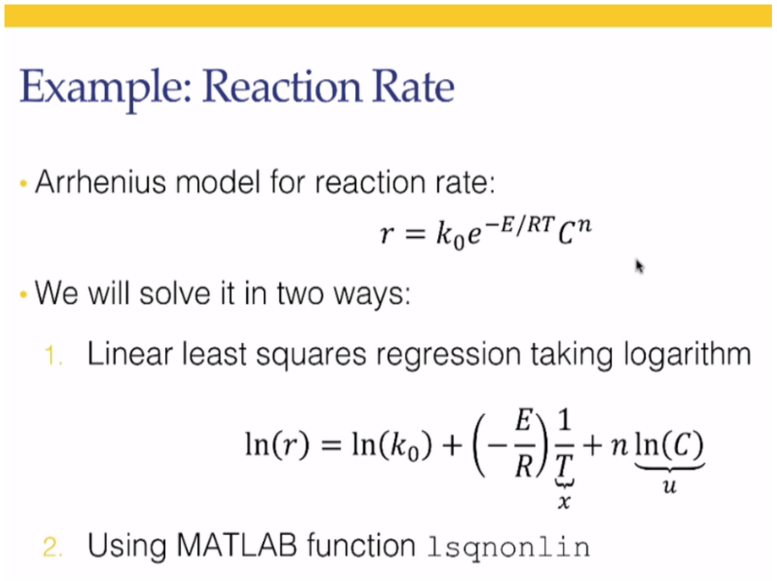
So we have our x, we have our u, we have our y. So let us construct our big matrices. Our x was nothing but ones and that is size of n rows and single column. Then we had our x followed by our u okay. This should give an error. I will come to that in a minute but let us say why nothing but our y okay was. So let us run this and see whether we get an error or not. Yes we get an error and the reason why we got an error was because our variable n was not defined okay.

So again, I am doing this in a step by step manner. So, that this becomes fairly easy for us to debug. So every time I write a few lines, I will go back and check. You do not need to do that, you can write the entire code and then check it. The reason why I am doing this is basically on a video like this, it becomes easier for me to go and show where the errors can possibly be rather than typing out the entire code okay.

So it is up to you how you want to do this. Whether you want to do this in a single shot writing the entire code and then debugging it or writing few lines of code running it and then check whether the results are okay or not and debug whenever you get an error. So anyway the error was that n is not defined. So let us define our variable n over here.

And n=length(y). So n is the number of rows in our vector y okay. So this is our x, this is our y and our phi is going to be nothing but $(x` * x)^{-1} * x^{'}*y$ okay. (Video Ends: 10: 53)

That was how the way we do linear least squares x transpose x inverse x transpose y. So inv of x` multiplied x. Our k0 is exp of phi (0), sorry phi (1), a1, a0 is phi (1) right okay. So this is our phi (1) is going to be nothing but logarithm (k0). So k0 is e ^ e1. e/r is nothing but negative a1. So e/r is –phi (2) and oops! Let us get to power point and our n is nothing but phi (3) okay.
So that is our linear least squares. Let us save and let us run this hopefully. It will run without an error okay. We have run this and our e/r is 480.1, k0 is 4.8 and our end is 0.75 approximately.
(Refer Slide Time: 21:29)



Example: Reaction Rate

- Arrhenius model for reaction rate:

$$r = k_0 e^{-E/RT} C^n$$

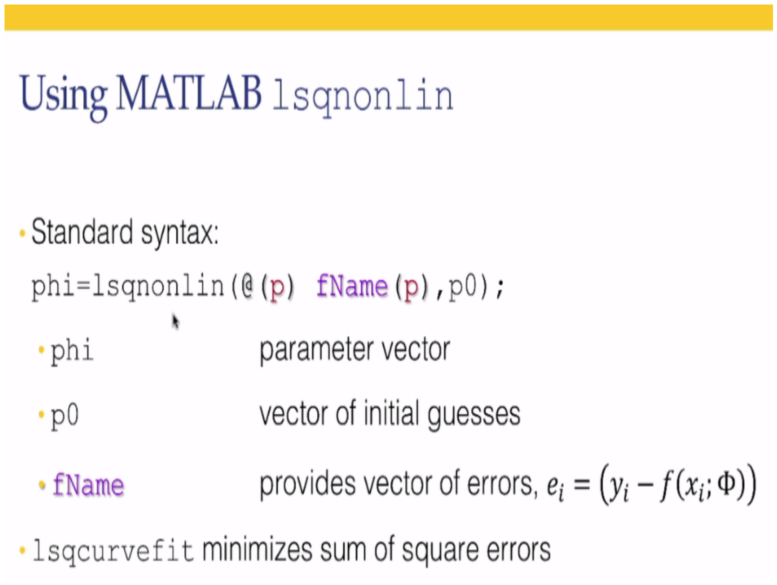
- We will solve it in two ways:
  1. Linear least squares regression taking logarithm

$$\ln(r) = \ln(k_0) + \left(-\frac{E}{R}\right)\underbrace{\frac{1}{T}}_{x} + n\,\underbrace{\ln(C)}_{u}$$

  2. Using MATLAB function `lsqnonlin`

So that is our rate of reaction using linear least square regression. Next what we are going to do is, we are going to use nonlinear regression of this particular function directly using lsqnonlin. So let us look at the syntax for lsqnonlin.

(Refer Slide Time: 12:30)



Using MATLAB `lsqnonlin`

- Standard syntax:

```
phi=lsqnonlin(@(p) fName(p),p0);
```

`phi`	parameter vector
`p0`	vector of initial guesses
`fName`	provides vector of errors, $e_i = \left(y_i - f(x_i;\Phi)\right)$

- `lsqcurvefit` minimizes sum of square errors

The syntax for the lsqnonlin is phi=lsqnonlin @ p space function name p and the initial guess. So let us start with that okay. (Video Starts: 12:39) Let us say our initial guess for phi (0), phi (0) or phi_guess, let us say was 1 was our k0. Let say our e/r was 100 and our n was also 1. So that was our phi_guess okay. Our phi known phiLSQ is going to be lsqnonlin at phi. Let us say rxnFunction phi okay.

So this is the standard syntax. We need to define our function. Function called rxnFunction and our phi_guess. And I think that should be, yeah that should be all that is required now fName. The function f sorry, the function f provides errors in the form of e=y- the model for values for various values of phi. So we basically need to get that in our function. So let us go here, edit, rxnFunction. Yes okay.

So function f error=rxnFunction and our phi. So k0 is phi (1). Our e is phi (2) and our n is phi (3). Rate of reactions is going to be nothing but k0 * exp-e/rt. So –e. So you are here. We are using this e/r so, e/r. /t okay. Multiplied by concentration c ^ n okay. That, the final bracket was not required is just for our sake for human readable sake. That is rate of reaction and our f error is nothing but yData - r, okay.

So to get parameters calculate rate okay. So before that we need to get what c is, c is nothing but our xData all the row's first column, t is xData all the row's second column okay. So again when I run this, I am going to get an error. So let us see what the error is going to be okay. Let us run this and we get an error okay. The error is undefined function or variable xData in rxnFunction. The reason why we get that get this error is xData and yData are defined in our workspace but this is a function. They are not defined in our function. We need to somehow define pass on this information in the function. So let us put this as 2 parameters, 2 additional parameters of the function okay.
So xData and yData need to be passed on to the function by the reaxparam script okay. And the way we do that if we know how to do this, is we have at phi. So that is the variable we need to solve for and an rxnFunction in addition to phi. We pass on all the other parameters that are required xData, yData okay.

Keep in mind xData and yData over here, stand for this variable name and this variable name the xData and yData over here stand for basically the variable name that we are going to use over here. And here they happen to be the same variable names, because it is the exact same data but it need not be that I can actually replace this with just let us say xx and yy and that will also work okay.

xData, yData will also work and xx and yy will also work. So let me just save this and run this and see that it works hopefully. It should without an error, oops! There is an error okay. The error is with the multiplication in line 11 and we wanted an element by element multiplication. So we should have put a dot star there okay. And star is fine because k0 is a scalar. So let us, let's clear all and run this okay.

Okay, so it's run and let us see what phiLSQ gives us. So with phiLSQ we got our k0 as 4.84. Our e/r is 486 and our n is 0.75 okay. So we are getting very close numbers, using our lsqnonlin as well as using our linear regression okay. Let us go over here and see what we have done again okay. lsqnonlin at phi rxnFunction phi, xData, yData.

The name xData over here has to be the same as this name over here. yData has to be the same as this name over here. phi has to be the same as this name over here okay. So let us change this data to xData and yData. So everywhere we have changed this to xData, everywhere we have change this to yData and let us run this okay. And it runs with our phiLSQ as the same values as before.

So as you can see the variable names are dummy as long as this name is exactly the same as this name. And this name is exactly the same as this name. There is actually not going to be any error that you will see okay. (Video Ends: 20:01) So with that we have completed our aims of this particular lecture. Our first aim was to convert this into a linear regression form and use linear least squares and our second aim was to use nonlinear least squares using MATLAB function okay.

Its linear least squares we use matrix method and created that matrix. The y matrix was log(r), the x matrix the first column was ones. The second column was 1/t, the third column was l and c and we use that in order to get our log (k0), our a/r and n. Then we used MATLAB function lsqnonlin, the standard syntax for which was phiLSQ nonlin at p fName p, p not were the initial guesses okay.

And if there were parameters, then we change this to at p fName p, param1, param2 and so on and so forth. fName returns the vector of errors which is y-y model by which y is nothing but r- the right hand side. So this is the error as a vector that fName needs to return okay. So with that we come to the end of lecture 6.3 and I will see you in lecture 6.4. Thank you and goodbye.