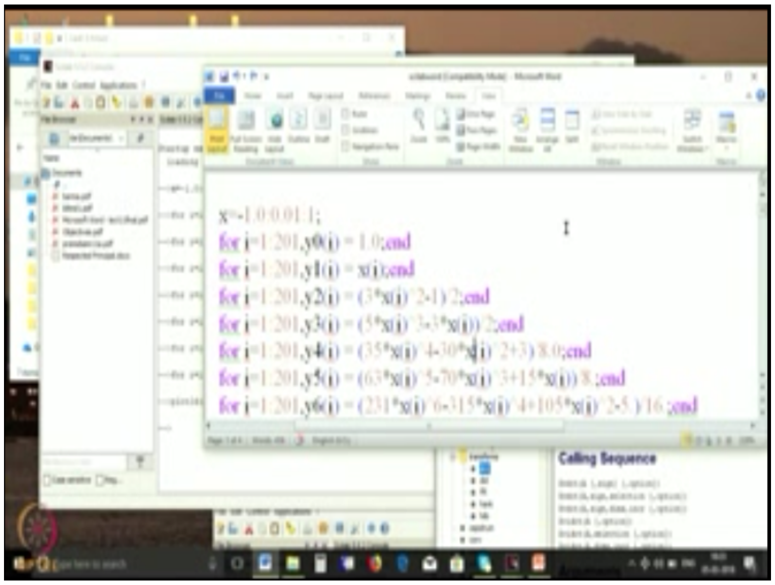
Computational Chemistry & Classical Molecular Dynamics
Prof. B. L. Tembe
Department of Chemistry
Indian Institute of Technology – Bombay

Lecture - 27
Scilab-5: Legendre Polynomials, Multiple Plots and Curve Fitting

Hello and welcome to this lecture. What we will do today is to continue our discussions on Scilab and consider several applications. The first application I want to study is plotting several Legendre polynomials and then I will consider the fitting function. Remember we had done the fitting function using the Fortran program. Scilab also allows you to do the same fitting in a very quick and it also allows you to plot all the fitted functions.

Then after that we will consider a very important topic that of Fourier and Laplace transforms. I will only illustrate the Fourier transforms and leave Laplace transforms as an exercise. Now look at your screen.

(Refer Slide Time: 00:59)



```
x=-1.0:0.01:1;
for i=1:201,y0(i) = 1.0;end
for i=1:201,y1(i) = x(i);end
for i=1:201,y2(i) = (3*x(i)^2-1)/2;end
for i=1:201,y3(i) = (5*x(i)^3-3*x(i))/2;end
for i=1:201,y4(i) = (35*x(i)^4-30*x(i)^2+3)/8.0;end
for i=1:201,y5(i) = (63*x(i)^5-70*x(i)^3+15*x(i))/8 ;end
for i=1:201,y6(i) = (231*x(i)^6-315*x(i)^4+105*x(i)^2-5 )/16 ;end
```

So what I have done, there are many, many programs that I would like to execute. So what I do rather than typing directly on the screen. I shall first enter all those commands in a word file and then copy, paste whatever programs I want from the word file into the Scilab environment. So you will see this word environment here and at the background is the Scilab environment. I have a Scilab environment.

I have already opened the Scilab executor, Scilab program and here is my word program. So if you are using the Linux operating system, then you may use that editor in that Scilab command. When you do an editor in a Scilab command, you can enter all the lines and then you can execute that edited file. That is an SCF file or if you want, you can just edit a normal file and save all your commands there and then copy, paste all those commands into your Scilab environment.

What is the advantage of this copy, pasting? See if you make some mistake, you can go back to your word file or an edited file and correct it and then go back to the Scilab environment. This is far more effective. So now look at the first thing I want to execute. I want to execute a graphics program, where I want to calculate a large number of Legendre polynomials. Remember these Legendre polynomials are nothing but your angular parts of your wave functions of a hydrogen atom. So now look at this word file.

I starts here, x=-1:0.001:1;. What this means, it is a do loop x values go from -1 spacing of 0.01:1; so the do loop calculates values of x from -1 to 1 with a spacing of 0.01. So there will be 201 points. So in Scilab, as I mentioned before, when you do not say anything, the variables are assumed to be an array variable. So in Fortran or in C, or other languages, you have to be very careful about your array variable. Here that array nature is implicit in all the variables.

So I define my x values 201 values from -1 to 1. These x values are nothing but cos theta values. The Legendre polynomials are functions of cos theta, cos theta goes from -1 to 1 and have generated 201 points on the x axis or abscissa. Now on the ordinate or the y axis, I want to calculate several Legendre polynomials. The first one is your first Legendre polynomial, which is a constant function. So how do I generate that constant function for i=1:201.

This is your do loop now. Remember in Scilab the do loop is i=1:201, not comma. So many times even I make this mistake, because I am used to Fortran. So everything you type, every character is important, i=1:201, y0i=1.0; end. So what this does? It executes a do loop for i going from 1 to 201. It calculates the values of this y0, y0 is my 0th Legendre polynomial, okay. So that is a constant. So the whole function is a constant.

It calculates the value of the constant for the first function. Now the second function is yi=x, so that is the second function is the same as x, that is first function is a constant, second function is a cos theta function, our x itself was cos theta. So the function is cos theta. So the second function or the first Legendre polynomial with this is l=1, that is cos theta. So the next one, this is like your pz orbital. The next one is your dz square orbital.

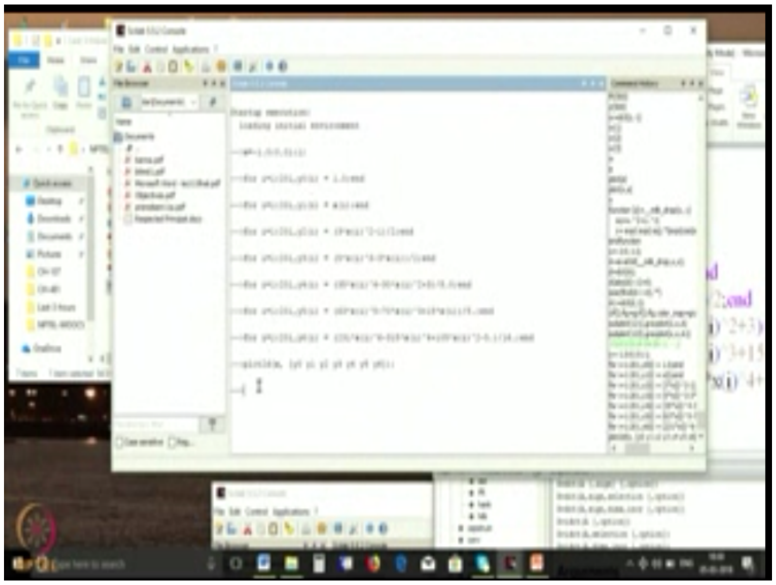
What was that dz square orbital, 3cos square theta-1/2. So my y2 function is a function 3 cos square theta-1/2. Here also I have a do loop. So what is my do loop, i going from 1 to 201, my y2 is defined as 3 cos square theta-1. So that is my next function. The function next to that is 5cos cube theta-3cos theta/2. So that is my y3 function. So similarly y4, y5, y6, so I have in this particular set of lines, I have those 6 functions I have created.

I have created the 6 functions. Then these first y0, y1, y2, y3, y4, y5, y6, actually 7 functions. Now I want to plot all the 7 functions. So I want to plot all the 7 functions on the screen. So what is the command for plotting now. If I plot 2D, actually I can say just plot, but plot 2D means plot in 2 dimensions. There is a plot 3D as well. So we are not going to do the 3D plotting as yet, so plot 2D.

If I say plot 2D(x,y0), so it will just plot just x and y0, but now I want to plot not only y0, y1, I want to plot all the 7 variables. So how do I give that plot statement. Plot 2D (x, this is a round bracket and now this is a square bracket. So the square bracket will have all the y coordinates that I want to plot, y0 separated by 1 space, y1 separated by a space, y2 separated by space, y3, y4, y5, y6, no comma.

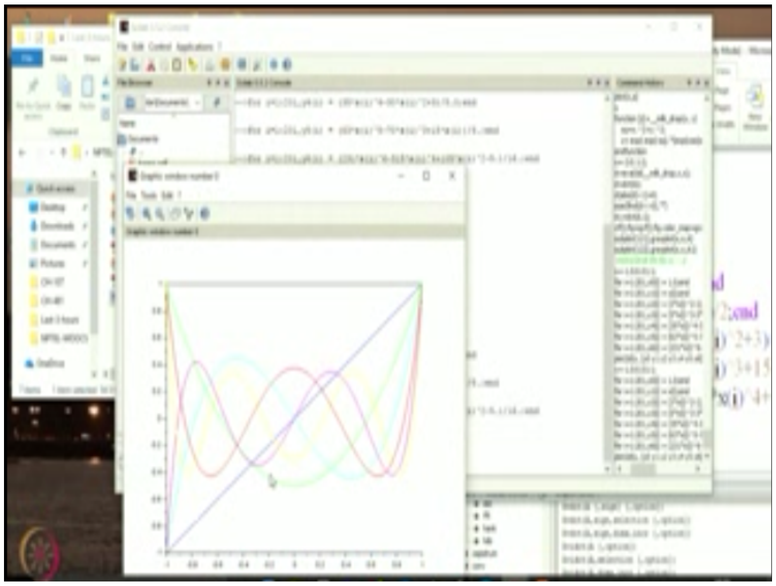
Just separate them by space, square bracket complete, angular bracket complete; so these are all my commands to calculate all the Legendre polynomials, the 7 of them, so I copy this from my word file, then I go and paste it in the Scilab executable file.

(Refer Slide Time: 07:39)



This is a Scilab environment. The arrow means a Scilab environment. I will go to that point and paste all my commands. See what it has done.

(Refer Slide Time: 07:53)



Now look at what Scilab has done, okay. Let us see how did Scilab write. I had given all the lines and pasted them. First is executed x from -1 to 1, spacing of 0.01, then it calculated y0, y1, y2, all the way up to y6, then we give this execution, plot this command, so it has executed all these, then finally when it says plot 2D(x, [y0 to y6]); it plots your functions. Now these are now the functions of the plot. So let us now see whether it has done alright.

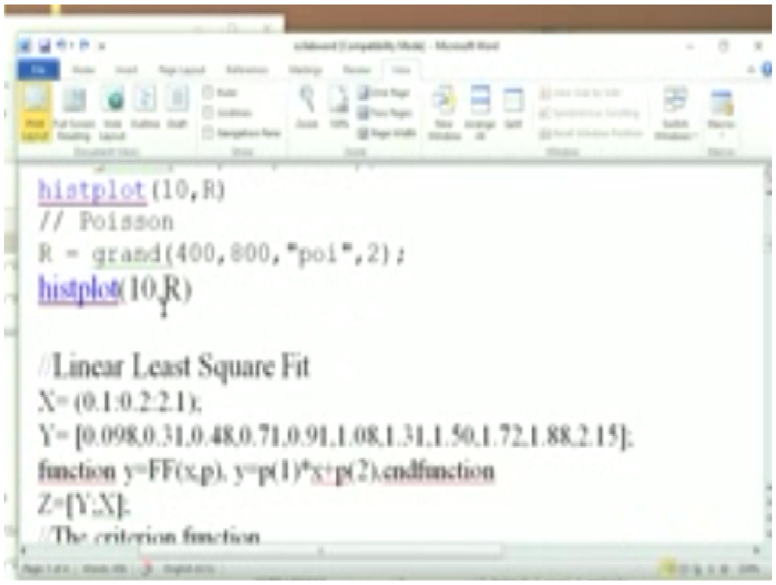
So the first function was a constant function, okay, y was equal to 1. This is the first function, which is a constant. No dependence on cos theta. Second function was exactly y=x, so this is my second function, ok this is my second function. So what is my third function. Third function is 3 cos square theta-1/2. This is my third function. So you will see that the third function has one oscillation. The fourth function will have two oscillation, and so on.

So it has plotted all the 7 Legendre polynomials here and the advantage of Scilab is you can get a plot instantly. So plotting is instantaneous, so you can really visualize whatever work you are doing. In Fortran, it is a lot more difficult, you have to do all the calculations, save it in a file, take that file, somewhere else and plot it. There are many softwares in Linux, which you may want to practice, so I shall mention to that. There is a software called Xmgrace.

You may want to download that software Xmgrace into your Linux system and Xmgrace allows you to plot very, very nicely. So I shall not do that Xmgrace in this course. So try to see your YouTube or try to go to the Google and search for your Xmgrace, download it and execute it. That I will not do in this course, because we have now only a few lectures left, so I will just illustrate the applications of Scilab.

So what we did now, we executed multiple plots using the Scilab. Now I shall close this window. So now what I want to do. I want to execute the fitting functions.

(Refer Slide Time: 10:29)



```
histplot(10,R)
// Poisson
R = grand(400,800,"poi",2);
histplot(10,R)

//Linear Least Square Fit
X= (0.1:0.2:2.1);
Y= [0.098,0.31,0.48,0.71,0.91,1.08,1.31,1.50,1.72,1.88,2.15];
function y=FF(x,p), y=p(1)*x+p(2),endfunction
Z=[Y;X];
//The criterion function
```

Remember we have done several lectures on fitting function. So what I want to do. This is that execute Legendre.sci, so if you have saved this file using a Scilab editor, you can execute that file. So what we did, we copied from word, so you can also do it this alternate way. So now what I want to do. I want to execute a linear least square fit. So this fitting is a very important operation. So we will use the Scilab program to do the linear least square fit, okay.

So look at these lines carefully. So there are some 10 or 12 lines. So you have to write it properly, so that you can execute it. So do not worry too much about the algorithm in Scilab, because Scilab is a very advanced software whereas our Fortran programs were very straight forward, linear least square fit, okay or polynomial least square fit. They had unique solution. So we have shown you how to do the programming.

Now we will show you how to do the same linear least square fit using Scilab. So in this case, my data x varies from 0.1 up to 2.1 spacing of 0.2. So how do I do it, x= (0.1:0.2:2.1), so colon gives a range, do not give a comma here. It will be, if you give a comma, there will be confusion. Here there will be 11 points for the value x. So now y is the data. What is my data? First point is 0.098, second point is 0.31, all the way up to 2.15.

These are my 11 points, which is a dependent variable, x is an independent variable. So I am just going to show you all the commands which are useful for plotting as well as fitting, fitting first and then plotting. So now I want to define a function using Scilab. What is my definition? So this is how you define a function in a Scilab, function y=ff(x,p), okay. This is needed for my fitting, y=ff(x, p), y=p1x+p2, end function. So my fitting function is a straight line.

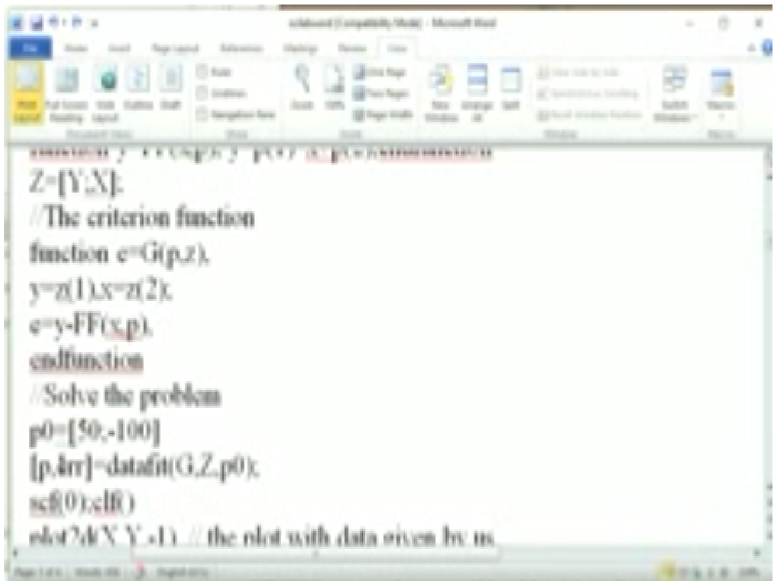
What is a straight line? Y=mx+c, so what is my m, p1 is my m and p2 is my constant C. So p1x, p1*x+p2. So this is my, I have defined a function. So now the remaining commands now, they are all required for the Scilab fitting program. So we do not ask too many questions on that. We just have to repeat exactly as these are. So after I define a function, so the next one, this is the criterion function. So just put //, so after this ff, I have to define z=y;x, okay.

That should be in square brackets, that is the requirement of this software, then the criterion function. This is a comment card, then I define function e=G(p, z). I have to define a function e, okay. I have to define a function e, G, z, okay, then y=z1, x=z2, so just you have to give this. Whatever fitting you want to do, this is a common thing, then e=y-ff(x,p). So y is your dependent variable, and ff is the fitted function.

Remember you are trying to fit a function, you are trying to fit a function, okay p1x+p2. So that is my fitted function, y is my original function, okay. So then end function, because I have defined a new function e, end function. Wherever I define a function, it should be end function. Then I have this given a command line, solve the problem. So when you solve the problem, you have to give an initial value of the coefficients p, remember.

Your p1 and p2 were two coefficients, to find out the fitted coefficients p, p is a vector now, p1 and p2. So to start my fitting, I need to give some initial values. So these 2 are my initial values, p0=(50;-100;). So this is p1, this is p2, so p1 and p2, okay. Remember the semicolon is important. If you do not use semicolon, there will be a problem.

(Refer Slide Time: 15:12)



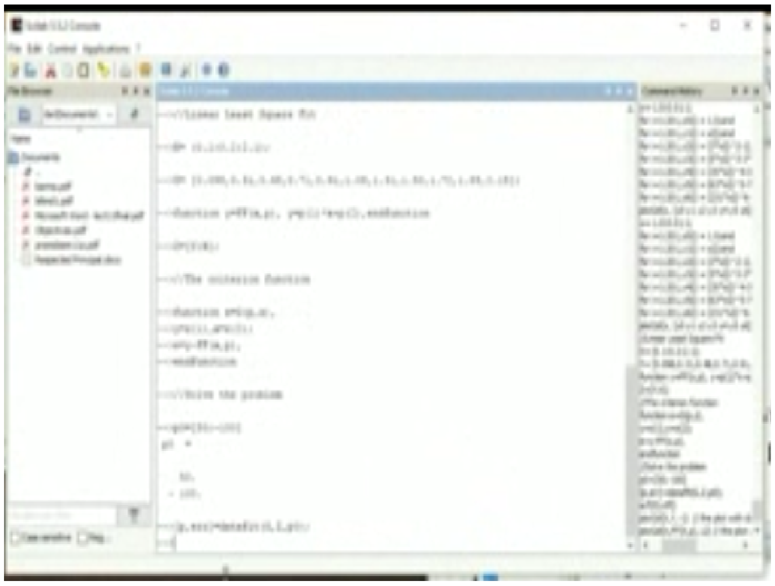
```
[illegible]
Z=[Y;X];
//The criterion function
function e=G(p,z),
y=z(1),x=z(2);
e=y-FF(x,p),
endfunction
//Solve the problem
p0=[50;-100]
[p,err]=datafit(G,Z,p0);
scf(0);clf()
plot2d(X,Y,-1) // the plot with data given by us
```

So then, I define this again a Scilab function p error, then you write p:error=data fit g, z, p0. So p0 is my starting data, z is the matrix you have defined, g is also a function I have defined, gpz, so this is my, it will execute, then these are again some standard commands. So when you come up to this point, it has already executed. When you come out of this, it has executed, so what am I going to do now. I am going to plot. I am going to plot x, y, -1.

So this is the data, which we had given, x and y were the data we had given. So it will plot x and y, -1 tells me what is the nature of the points, okay, you can vary this third variable here, so this tells you whether your points will be dots, stars, plus signs, and so on. So the third point, the third variable in a plot statement tells you the nature of the graphics. So first I will plot x and y, y is my actual data, then I will plot the fitted data, fitted data is ff, ff is my fitted function.

So that I will use some other notation. So if -1 is a star, 12 will be some other notation. So you practice all these, okay. So this is my function to fit a straight line. So what I will do, very simple. I will copy all the things up to this plot 2D. So this should allowing me to execute my linear least square fit program. So I will copy this, then I will go to Scilab environment.

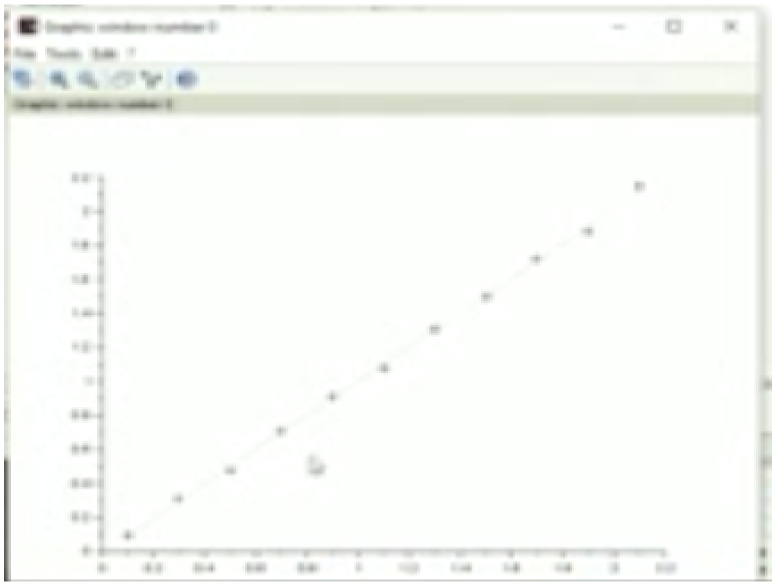
(Refer Slide Time: 17:03)



```
-->//Linear Least Square fit

-->X= [illegible]

-->Y= [illegible]

-->function y=FF(x,p), [illegible], endfunction

-->Z=[illegible];

-->//The criterion function

-->function e=G(p,z),
-->[illegible]
-->e=y-FF(x,p),
-->endfunction

-->//Solve the problem

-->p0=[50;-100]
p0  =

    50.
  - 100.

-->[p,err]=datafit(G,Z,p0);
-->
```

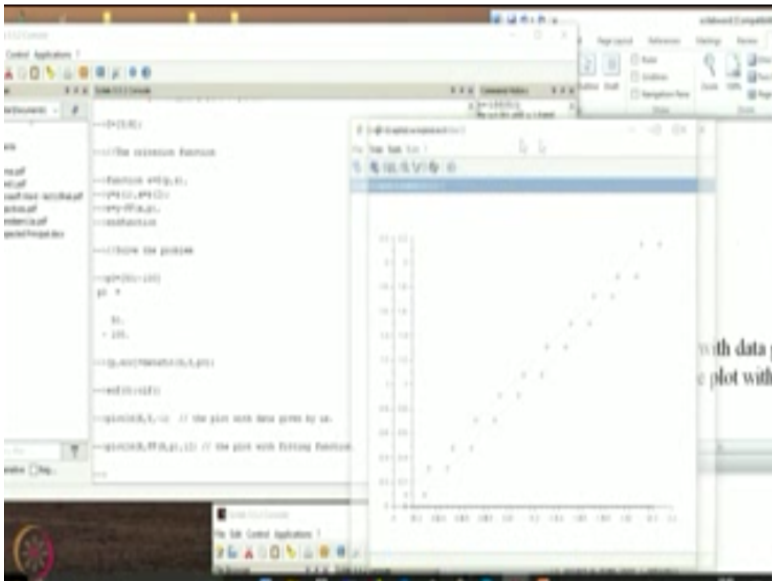
I will go to Scilab environment. I will paste all the lines, so I have pasted all the lines, so you see.

(Refer Slide Time: 17:10)



This is the execution. So it executed every so you can look through, you execute it yourself, look through all that, p0 was defined as 50; -100. This is my initial thing. So when you give p error, it does the fitting through this, then plots. So let us look at the plots now.

(Refer Slide Time: 17:30)

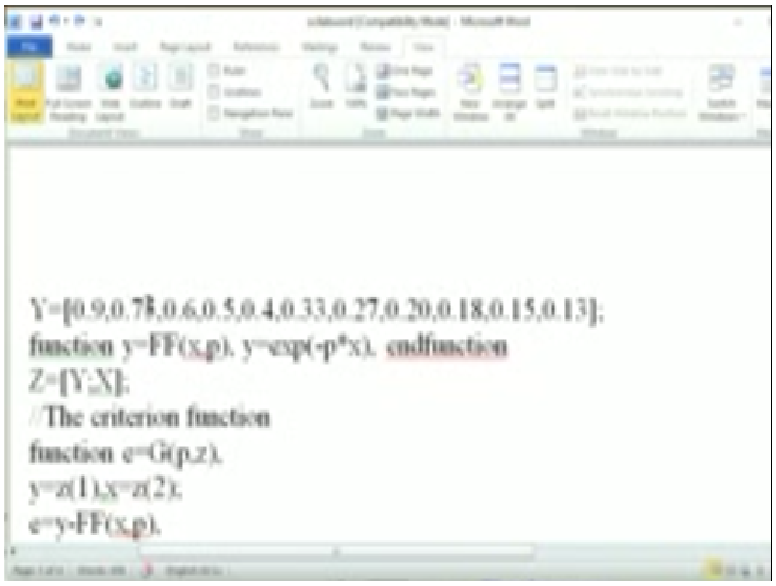


```
-->Z=[Y;X];

-->//The criterion function

-->function e=G(p,z),
-->y=z(1),x=z(2);
-->e=y-FF(x,p),
-->endfunction

-->//Solve the problem

-->p0=[01;-100]
 p0  =

    0.
  - 100.

-->[p,err]=datafit(G,Z,p0);

-->scf(0);clf();

-->plot2d(X,Y,-1)  // the plot with data given by us.

-->plot2d(X,FF(X,p),12) // the plot with fitting function.

-->
```

So the plus points are my original data point. The plus points are my original data points and the blue straight line is a fitted function through the Scilab. So you see that just by following the statements, which were given. So I am able to do this fitting almost in about 1 or 2 minutes. So to me, you should know the logic of fitting the functions, you should know the program logic, but when you want to do something quickly, I feel Scilab is a much better option.

So this is my linear least square fit. So what I now want to do, I want to fit some other function. I want to fit some other function. So let us see what that, so I will close this window. So I want to do 2 more fitting calculations, so the next one is an exponential fit.

(Refer Slide Time: 18:23)



```
Y=[0.9,0.78,0.6,0.5,0.4,0.33,0.27,0.20,0.18,0.15,0.13];
function y=FF(x,p), y=exp(-p*x), endfunction
Z=[Y;X];
//The criterion function
function e=G(p,z),
y=z(1),x=z(2);
e=y-FF(x,p),
```

So what is an exponential fit? My x is the same, same as the old data. Then my y is some function. I have taken some arbitrary number of points. I have taken an arbitrary 11 points. So I am taking this point, these data points, they are decreasing with x. So I have given it almost like a exponential decay. I am giving the decaying function. So the only change now here is that y=f(x,p), now y=exp(-p*x). So in the previous program, my p had 2 values, p1 and p2.

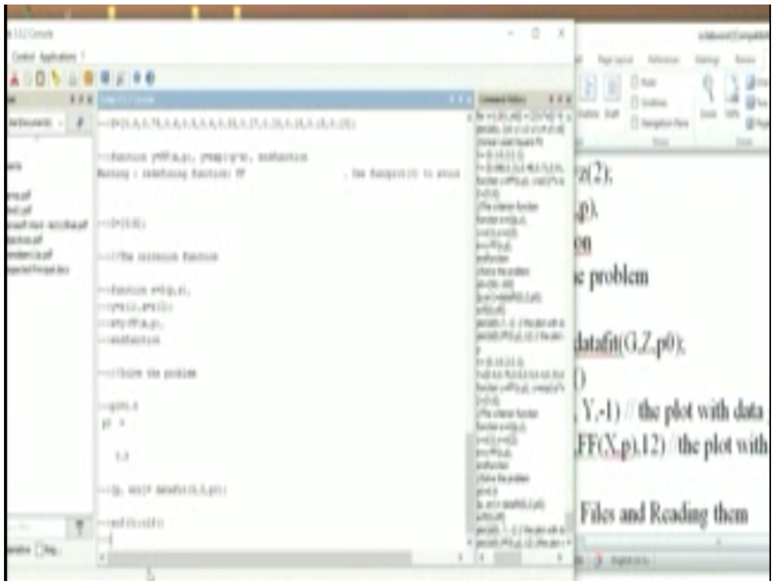
Now this p will have only 1 value. This is not a vector. It is not a column vector. So it is just an individual number p*x. So again criterion function, z is the same function, e this is the same, error is the same. The only difference now that p0 is a single value. Earlier remember I had p1 and p2. Now I have exactly a single value. This is the value of the exponent, initial value of the exponent, okay. Then p, error=data fit g, z, p0, p0 is a single number now.

Now plot 2D. I have used the same exactly the same commands. So I will copy these and paste into my Scilab and see what it is going to do. Copy up to that plot 2D. I will go to the Scilab. So before I execute, let me ask you in the previous program it was p1 and p2, is not it? It calculated p1 and p2. So if I just say p and return, it gives these values. Remember it was y=mx+c. The previous program p was a column vector.

This is y=mx+c, m=1, 0.01 and constant was -0.007. So now I shall execute the second program. This is the exponential fit. So just to clarify, this p was a column vector in the previous program, in my next program p is the single value of the exponent. So it will understand. The moment you give a new program; it will understand that you have changed the variable. What I will do? I will just copy. Now I should actually paste it.

So let us see whether I paste, there is nothing. Because I go back, copy all my commands, then I paste it in the Scilab environment, paste.

(Refer Slide Time: 21:03)

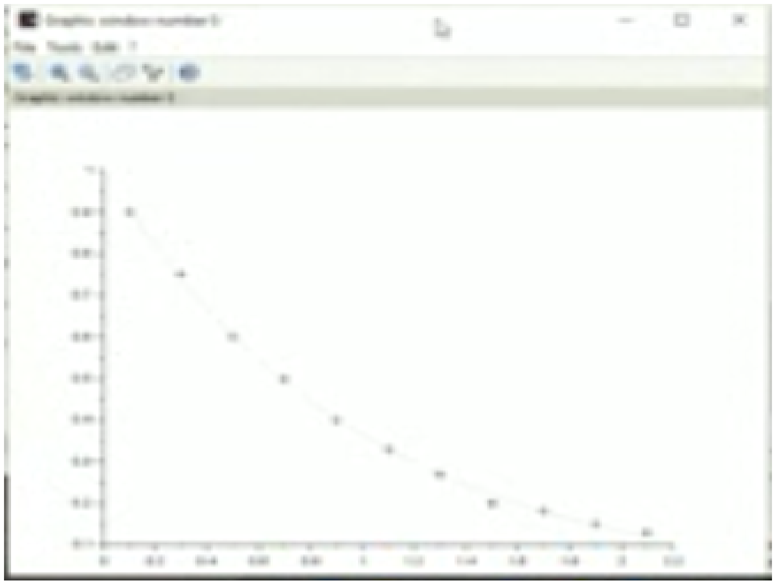


```
-->[illegible]
-->function y=FF(x,p), [illegible], endfunction
Warning : redefining function: FF [illegible]
-->[illegible]
-->//The criterion function
-->function e=G(p,z),
-->[illegible]
-->e=y-FF(x,p),
-->endfunction
-->//Solve the problem
-->p0=1.0
 p0  =
   1.0
-->[p, err]= datafit(G,Z,p0);
-->[illegible]
-->
```

```
z(2);
p)
on
e problem
datafit(G,Z,p0);
)
, Y,-1) // the plot with data
FF(X,p),12) // the plot with
Files and Reading them
```

So it is executed.

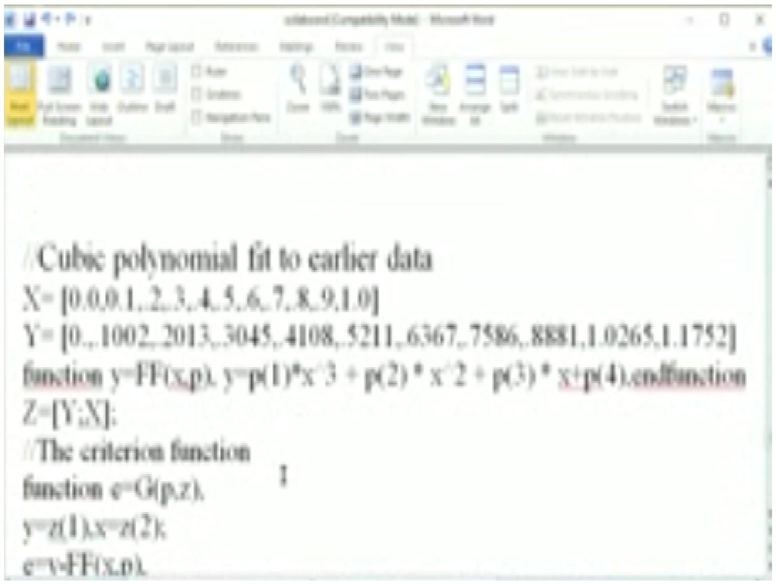
(Refer Slide Time: 21:05)



See it has run the execution. The plus signs were my original data. I had purposely given a set of decreasing data points. I fitted an exponential, so you see it is beautifully fitting through those plus points. So this is my exponential fit. So instead of this, there was no constant to that, you see. There was no constant, just it was e to the –p, x. Suppose you want to fit a+e to the –px, so you have to give two variables, just as I had p1 and p2. So I can say p1+e to the –p2x.

So when I give 2 parameters, my initial p0 should also be 2 parameters. What I mean is, see if instead of y=e to the –px, suppose I want y=p1+exponential-p2x, 2 parameters, p1 and p2. So here when I define my p0. I should define 2 parameters, okay, we then it will solve this problem. So the next problem I want to do. I want to remember the executed cubic polynomial fit. We executed a cubic polynomial fit. So I want to use that cubic polynomial fit.

(Refer Slide Time: 22:28)



```
//Cubic polynomial fit to earlier data
X= [0.0,0.1,.2,.3,.4,.5,.6,.7,.8,.9,1.0]
Y= [0.,.1002,.2013,.3045,.4108,.5211,.6367,.7586,.8881,1.0265,1.1752]
function y=FF(x,p), y=p(1)*x^3 + p(2) * x^2 + p(3) * x+p(4),endfunction
Z=[Y;X];
//The criterion function
function e=G(p,z),
y=z(1),x=z(2);
e=y-FF(x,p),
```

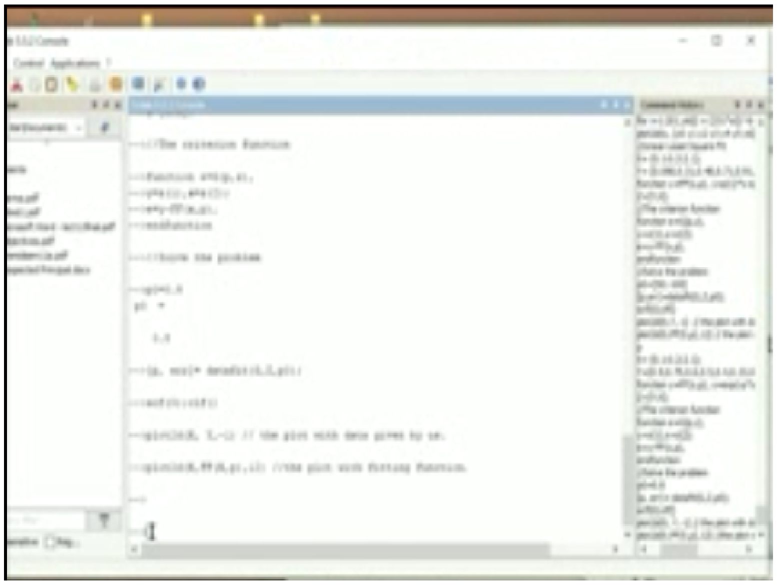
So this is the fitting function, my x values went from 0 up to 1. I think there are 11 points. 1, 2, 3, 4, 5, 6, 7,8,9, 10, 11, 11 data points, my y had also 11 data points. Now I want to fit through this data points y, cubic polynomial. What is my cubic polynomial? So here my fitting function y=f(x, p), this is the cubic polynomial, what is that? y=p1*x cube. So in Scilab, the cubed is x hat 3. So this is how I write cubes in Scilab.

In Fortran, you would write it as x**3 or if you want to be plain, you will say x*x*x. So it is x cube. So y fitting function is y=p1*x cube+p2*x square+p3*x+p4; sorry, end function. So this is my fitting function. So the rest of the commands are the same y=zx, criterion function e, everything is the same, y=z1, exactly copy this. You do not have to do anything. So now the only thing you have to change is my p0.

Because I had 4 parameters, since I had 4 parameters, my initial set of parameter should also be 4. So [p0=0.1;-0.1;0.5;0.1], this semicolon is very important. I tried many times with the comma, it did not work, because p should be a column vector. So you get a column vector, when you give semicolons. When you give semicolons, you get a column vector. So I give my initial value of p that is p0, then data fit, g, z, p0, p: error, execute.

Let us now, I will copy all these into my Scilab, okay. Let us execute, copy. Then go to Scilab.

(Refer Slide Time: 24:36)



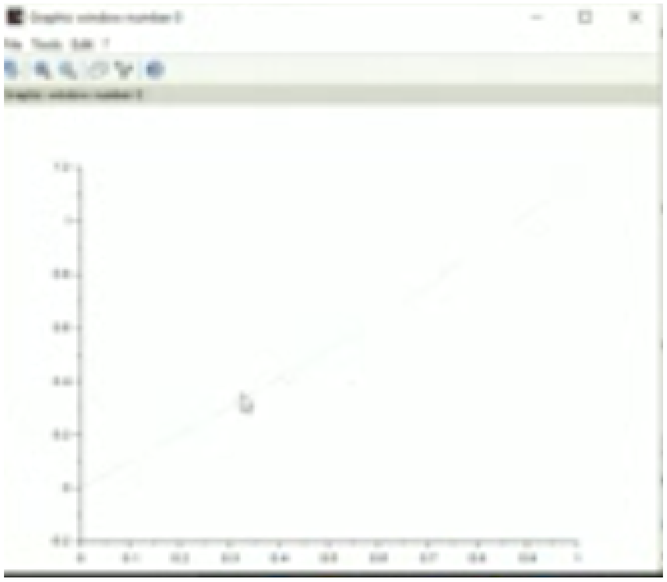
```
-->//The criterion function

-->function e=G(p,z);
-->y=z(1),x=z(2);
-->e=y-FF(x,p);
-->endfunction

-->//Solve the problem

-->p0=1.0
 p0  =

    1.0

-->[p, err]= datafit(G,Z,p0);

-->scf(0);clf();

-->plot2d(X, Y,-1) // the plot with data given to us.

-->plot2d(X,FF(X,p),12) //the plot with fitting function.

-->

-->
```

Scilab environment, so I paste. I have paste my new things, so it did give some problems, okay. So it has given some warning, so warning should not matter. So it has given me a plot. It has given a nice fit, yesterday. So let us see whether it has given. So what I will do, since it is executed, let us try to print that p. There will be 4 value of p, so you see. That p1, p2, p3, p4, it is exactly the same program that we did in the class earlier.

Why it has not printed, I do not know. So let me try this again. So exactly it has given the same coefficient that you obtained through a least square fit, cubic least square fit, exactly as in our earlier class, the coefficient differ a little bit, but it is exactly the same. So let me try this plot again. I still do not know why it did not plot. It should have plotted my fitted function. So I shall copy and see whether it will plot it, so it should plot.

So if it has not plotted, so the plot will be somewhere in the screen. So let me see. I think it has plotted. This is my fit. So it has plotted. It just appeared somewhere else or maybe it is the least square, so let me try that again. So one thing you have to do keep trying it again. So it has plotted.

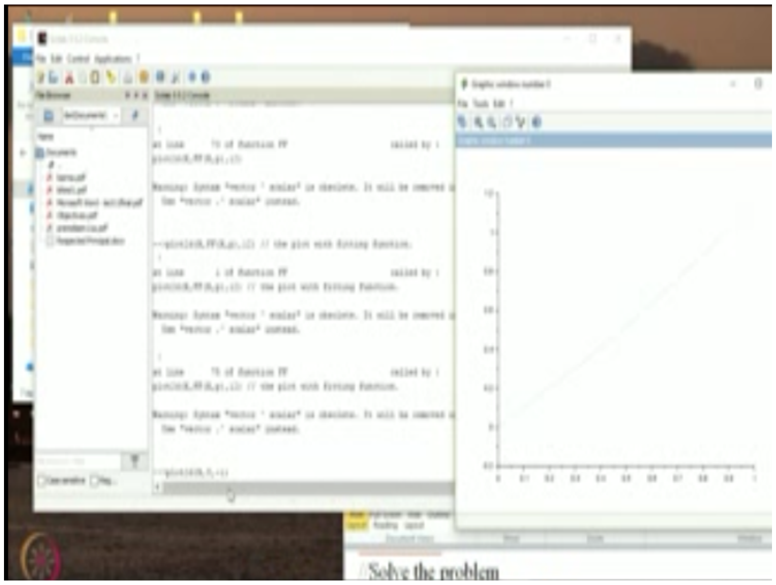
(Refer Slide Time: 26:40)



It has plotted, this is the fitted function, okay. Now let me also plot, this is the fitted function. What was my original function? Original function was x,y, okay. Fitted function, so this is the plot, you see on the right hand side, it is the fitted function. So let me try again, paste. It just gave a warning, because I think what this warning says is that you just see here. In Scilab 6, they are going to make some alternations from my original Scilab 5.5.

So you will have a better version when you download. So as far as what I will do now is just execute. This was -1, this is my execute that -1, close my bracket. So instead of f(x,p), I will just say y here because I want to plot my original function. So already on the right side, you have seen that it has plotted the fitted function. So now I want to plot the y data as well. So you see low and behold.

(Refer Slide Time: 27:57)


```
at line    73 of function FF                    called by :
plot2d(X,FF(X,p),13)

Warning: Syntax "vector ' scalar" is obsolete. It will be removed [illegible]
  Use "vector .' scalar" instead.


-->plot2d(X,FF(X,p),13) // the plot with fitting function.

at line     1 of function FF                    called by :
plot2d(X,FF(X,p),13) // the plot with fitting function.

Warning: Syntax "vector ' scalar" is obsolete. It will be removed [illegible]
  Use "vector .' scalar" instead.


at line    73 of function FF                    called by :
plot2d(X,FF(X,p),13) // the plot with fitting function.

Warning: Syntax "vector ' scalar" is obsolete. It will be removed [illegible]
  Use "vector .' scalar" instead.


-->plot2d(X,Y,-1)
```

It plotted the y data as well. So I had, this is the fitting function. So what I will do in the next lecture, I will explain about Fourier transforms. So what I have done in the present lecture? Plotting multiple functions, we have seen and fitting several functions using the Scilab fitting programs. This is going to be extremely useful because many of you will be doing lot of experiments.

You want to fit curves through your data points and Scilab gives a very, very nice algorithm as well as a set of statements to fit curves through your data points. So I will conclude here and in the next lecture, I will start on Fourier transforms and then also begin molecular dynamics. Thank you.