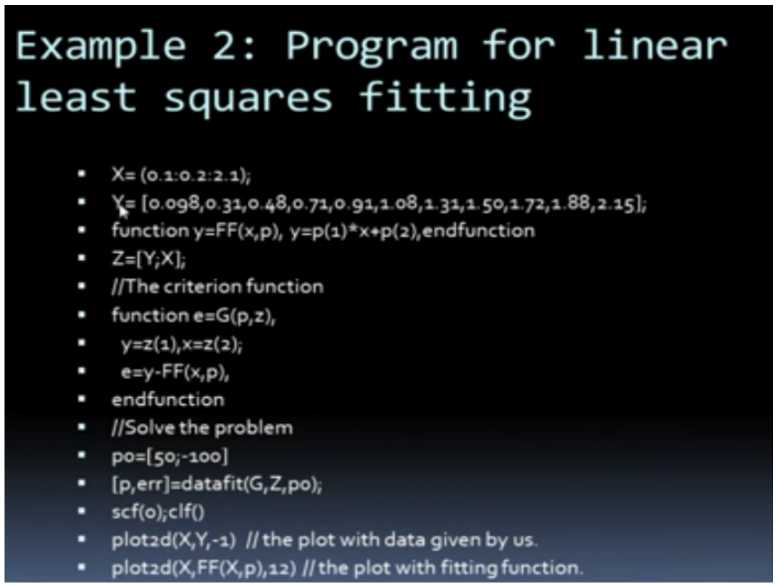
Computational Chemistry & Classical Molecular Dynamics

Prof. B. L. Tembe
Department of Chemistry
Indian Institute of Technology – Bombay

Lecture - 26
Scilab-4: Curve Fitting and Execution of Scilab Programs

Hello and welcome to this lecture. This is going to be a continuation lecture for Scilab. In the last discussion, we were talking of fitting using Scilab. These are all the commands for fitting.

(Refer Slide Time: 00:31)



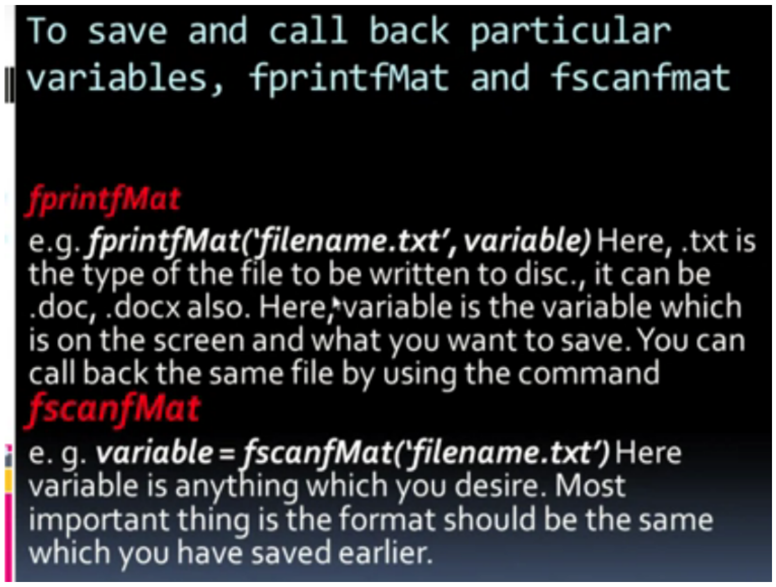
Example 2: Program for linear least squares fitting

```
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
function e=G(p,z),
  y=z(1),x=z(2);
  e=y-FF(x,p),
endfunction
//Solve the problem
po=[50;-100]
[p,err]=datafit(G,Z,po);
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 what are these commands for fitting, define the value of x, define the value of y, define a fitting function, that fitting function will have this is a linear least square fit, so I will have y=mx+c, that is my straight line. In my case, p1x+p2, then exactly copy these commands as they are, okay. Then, finally execute the data fit. Data fit is the fitting function. Then plot this on the screen. So this is what we have to do for fitting a straight line through a set of data points.

Instead of a straight line, suppose I want a quadratic function, p1x square+p2x+p3. So I can give 3, I can give a different fitting function, there a quadratic, then define all the values of p1, p2, p3 and execute all these in the same way, okay. So that is how you will do. Then, before we start our execution, there are 2 more things I want to comment on.

(Refer Slide Time: 01:34)



To save and call back particular variables, fprintfMat and fscanfmat

fprintfMat

e.g. ***fprintfMat('filename.txt', variable)*** Here, .txt is the type of the file to be written to disc., it can be .doc, .docx also. Here, variable is the variable which is on the screen and what you want to save. You can call back the same file by using the command

fscanfMat

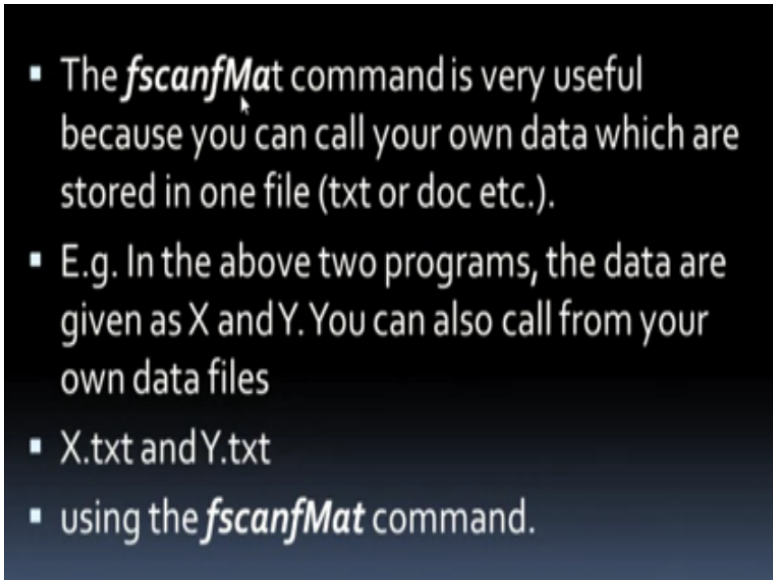
e. g. ***variable = fscanfMat('filename.txt')*** Here variable is anything which you desire. Most important thing is the format should be the same which you have saved earlier.

The next one is how do I write variables into a disk. Remember in Fortran I used open unit=11, open unit=12 and associated those units with some filenames. So in Scilab, there are no units. You directly write a variable into a file, okay. Say for example, suppose I want to write a particular variable into some file, so what do I do? That function is f print mat, fprintfMat, so fprint fMat allows you to write a variable into that file called filename.text.

So not only you can write 1 variable, you can write several things into your filename.text and as you execute your Scilab program, keep on writing those variables, different variables into this file and at the end, you may want now to read those from that particular file. So how will you read. So fprint fMat is more like a write statement. The read statement would be fscanfMat, okay. So this is my read statement.

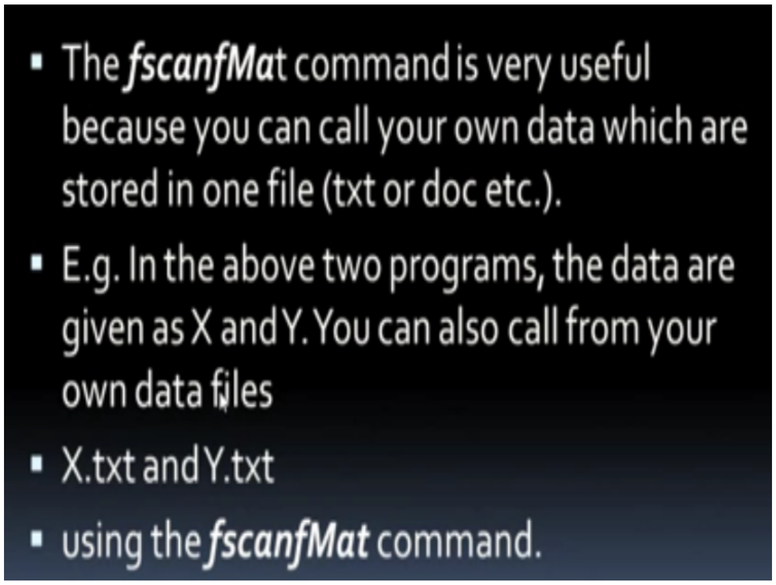
Suppose I want to read the variable from this file, I will now say variable=fscanfMat ("filename.text"). So this is how I will read from the particular file and this is how I will write from the file. So these are the 2 commands that allow me to write something into a file and read from a file. So these are the read and write statements, okay.

(Refer Slide Time: 03:14)


- The *fscanfMat* command is very useful because you can call your own data which are stored in one file (txt or doc etc.).
- E.g. In the above two programs, the data are given as X and Y. You can also call from your own data files
- X.txt and Y.txt
- using the *fscanfMat* command.

Now fscanfMat, okay, you can actually, you may have saved many things in your text file or doc file, you can read it from fscanfMat, okay. So the data files could be, it need not be always text, it could have any other extension. It could have any other extension. So you can read and write using Scilab. So let me summarize what all we have done using Scilab, okay.

(Refer Slide Time: 03:41)



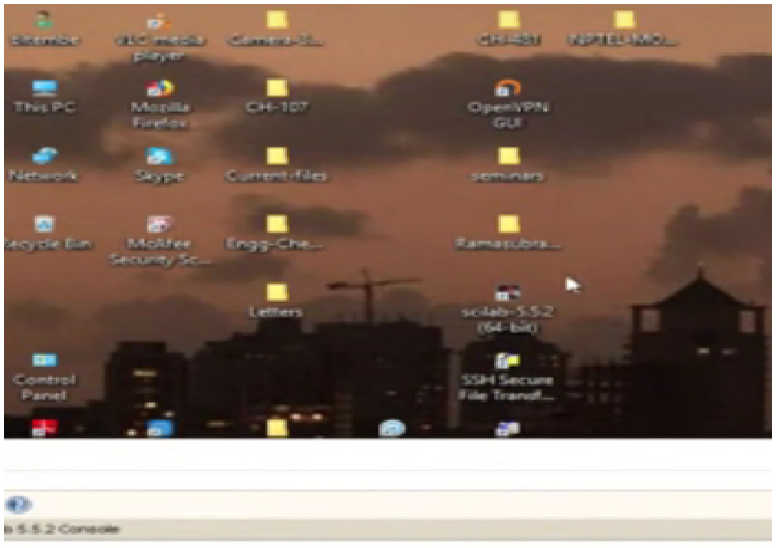
- The *fscanfMat* command is very useful because you can call your own data which are stored in one file (txt or doc etc.).
- E.g. In the above two programs, the data are given as X and Y. You can also call from your own data files
- X.txt and Y.txt
- using the *fscanfMat* command.

We just, we did not do everything. We just used some of the operations, 1 was a set of matrix operations that we did using Scilab. Another 1, we solved differential equations, then we solved roots of equations, we fitted, we plotted many functions, okay. We also fitted function. We discussed Scilab programming, rudiments of Scilab programming and we also discussed storage and execution of programs.

So personally, I have really enjoyed this Scilab because whatever took me so much time using Fortran programs, I can do the same thing in less than 1 day with Scilab programs, okay, but there are also advantages of knowing all the programming yourself because many of the functions that Scilab is executing, we do not know the logic. Whereas when you do your own Fortran programming, you know the logic of every step.

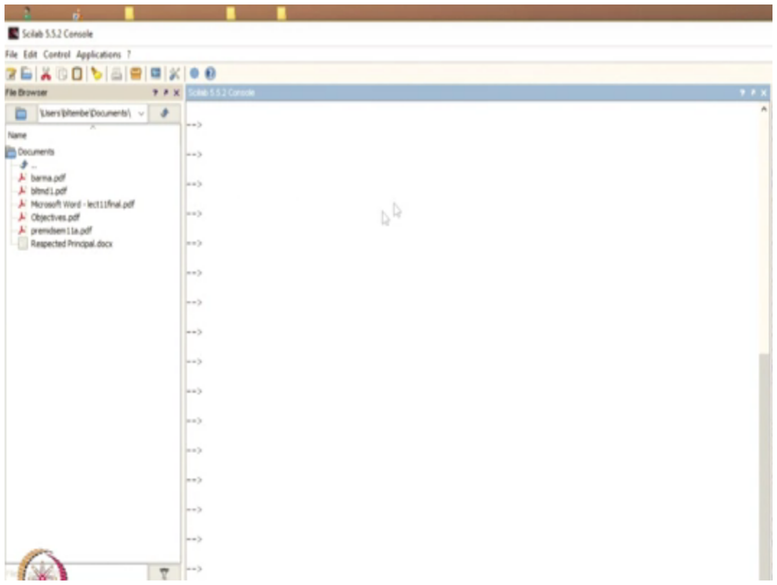
So enjoy both Scilab programming as well as computing. So now we will execute some things using Scilab. So what I will do? So I want to now. There is a Scilab environment. You can see on the screen.

(Refer Slide Time: 05:02)



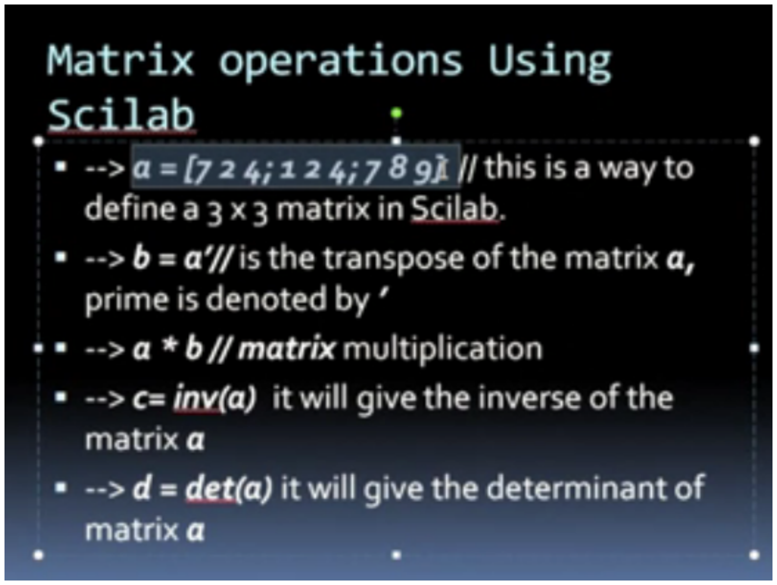
There is a Scilab 5.5.2, when I double click on this particular thing, this whole menu appears.

(Refer Slide Time: 05:10)



This whole menu appears, so this is my Scilab environment. So in this Scilab environment, I want to execute many, many functions that I already described to you during my course of this lectures. So let us start with matrix operations. Let us start with matrix operations. So I will go to my lecture, whichever commands I had actually used, I shall use identical commands now to execute Scilab, okay.

(Refer Slide Time: 05:47)



Matrix operations Using Scilab

- --> *a = [7 2 4; 1 2 4; 7 8 9]* // this is a way to define a 3 x 3 matrix in Scilab.
- --> ***b = a'*** // is the transpose of the matrix ***a***, prime is denoted by ***'***
- --> ***a * b // matrix*** multiplication
- --> ***c= inv(a)*** it will give the inverse of the matrix ***a***
- --> ***d = det(a)*** it will give the determinant of matrix ***a***

So I take this A=, this particular matrix. Let me want to take this A, so I will, what I have done? I have just taken it. I have taken the, let me paste it. So I have got this here. So what I have done. I took. It shows at the bottom of the screen. I took this particular matrix A, 7, 2, 4;1, 2, 4;7, 8, 9.

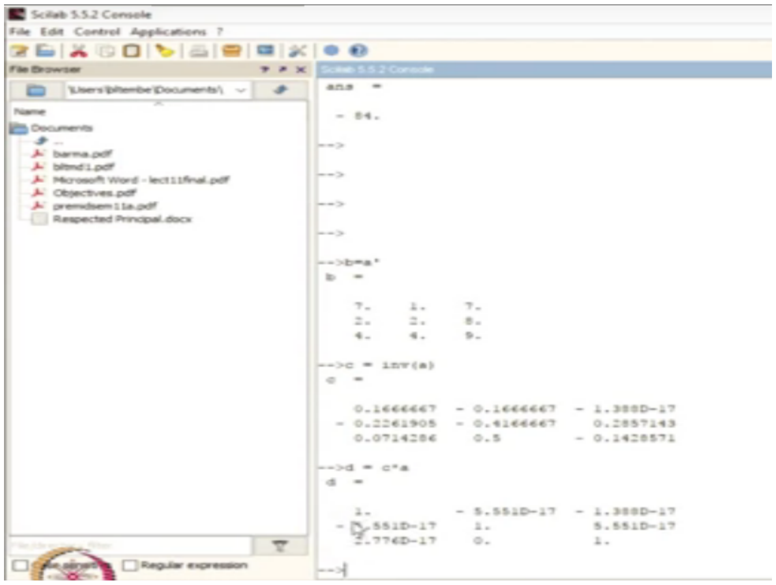
So I took this matrix and I enter. So when I type A and enter, it prints me the whole matrix. It prints me the whole matrix. So now, let us say I want to calculate the determinant of A.

What was the command? Det(A), so this gives me the determinant, the answer is -64, that is the determinant of A. So let me also calculate A prime now. What is A prime? A prime was a transpose of A. So let me say B=A prime, so when I enter, so remember what was my matrix? My matrix was 7, 2, 4;1, 2, 4;7, 8, 9. So the transpose is, the transpose of the matrix, the column will be the row, 7, 1, 7; 2, 2, 8; and 4, 4, 9. So I got the transpose.

So now I want to inverse this matrix. So how do I inverse the matrix, that was very straight forward, so let us say C=inverse of A, okay. So when I do that, so what I got? This set is the set of elements of the inverse of that matrix. So I got C was the inverse of. So what I can do now. How do I know it is really the inverse? The best way would be I multiply C by the matrix A and see whether I get the identity matrix.

So let us say D=C*A. So I am multiplying C, which is the inverse of A and I am multiplying by A. So let us see what I get.

(Refer Slide Time: 08:34)

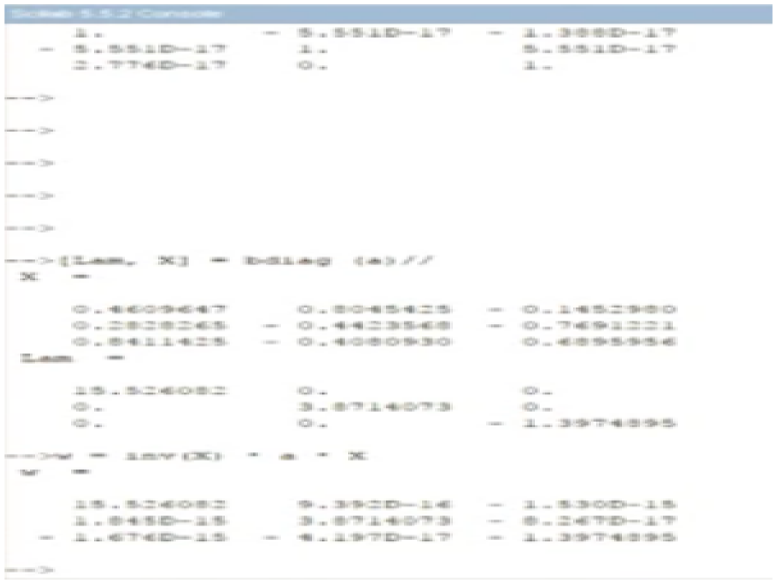


```
 ans  =

  - 64.

-->

-->

-->

-->

-->b=a'
 b  =

    7.    1.    7.
    2.    2.    8.
    4.    4.    9.

-->c = inv(a)
 c  =

    0.1666667  - 0.1666667  - 1.388D-17
  - 0.2261905  - 0.4166667    0.2857143
    0.0714286    0.5        - 0.1428571

-->d = c*a
 d  =

    1.         - 5.551D-17  - 1.388D-17
  - 5.551D-17    1.           5.551D-17
    2.776D-17    0.           1.

-->
```

Now see this is my D. So D is the product of the matrix and it's inverse. So you will see that what are the elements, 1-5*10-7-1*10-7 again -5*10-17, 15*10-17, 2*10-17, 01. So this is really an identity matrix, because in a computer 5*10-17 is a number where there are 16 0s followed by that 5. So this is as good as 0 on my computer. So it is an extremely small number all the half diagonal elements are small and diagonal elements are 1.

So you saw that using Scilab in a matter of seconds, I can execute the matrix operation. Then we also want to do the Eigen values and Eigen vectors. Remember that was the very important command. So let us do this. I will take this lambda x=b diagonal a, okay. So I want to take this command and execute in Scilab. So all I am doing now, I am taking these commands and executing in Scilab.

(Refer Slide Time: 09:57)



Scilab 5.5.2 Console

```
      1.              - 5.551D-17   - 1.388D-17
    - 5.551D-17        1.             5.551D-17
      2.776D-17        0.             1.

-->

-->

-->

-->

-->

-->[Lam, X] = bdiag (a)//
 X  =

      0.4609647       0.8045425   - 0.1452980
      0.2828265     - 0.4423548   - 0.7491221
      0.8411425     - 0.4080930     0.4855954
 Lam  =

      15.526082       0.             0.
      0.              3.8714073      0.
      0.              0.           - 1.3974595

-->w = inv(X) * a * X
 w  =

      15.526082       9.392D-14   - 1.530D-15
      1.845D-15       3.8714073   - 8.247D-17
    - 1.676D-15     - 4.197D-17   - 1.3974595

-->
```

So I paste those elements. So I do not know whether it has gone. So it did not paste or it has pasted, okay. See the last line here, because I am not able to reduce this screen the way I want. So I have typed lam, just look at the last line here lam, x=b diagonal a, okay. So actually I have typed it twice. So I will eliminate it once, okay. If you do it twice, it will give lot of problems, though it was a command card anyway. So what I want to do?

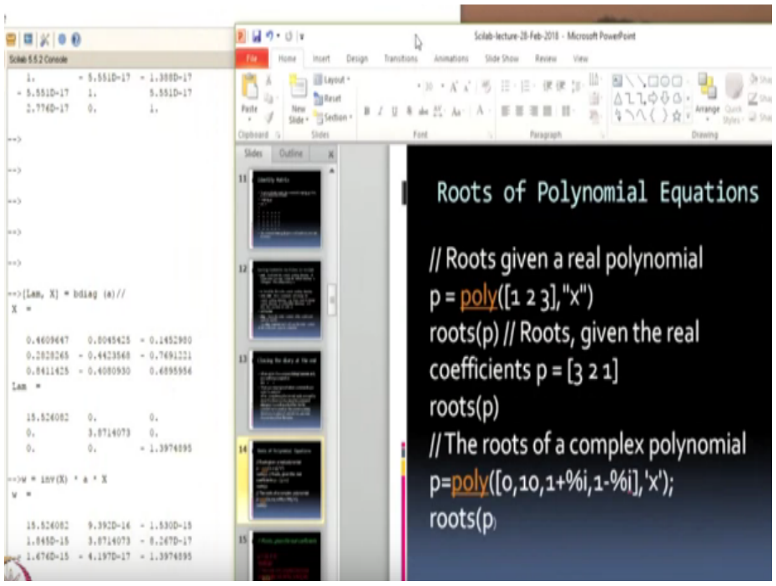
My matrix was actually small a, so instead of A, I will replace by a, okay. So my lam, x, lambda would be my Eigen values and x would be Eigen vectors, okay. So I will execute it. So when I execute, look at what has happened. Lambda are my Eigen values. So they are given in the diagonal rows. First Eigen value, second Eigen value, third, Eigen value, this is my Eigen value and x is a set of Eigen vectors. X is a set of Eigen vectors.

So to make sure that remember how is the diagonal matrix obtained from the original matrix, x inverse ax is my diagonal matrix. So let us see whether I get the same thing. So what I will do, I will say w=I want to say x inverse ax, so what I will do inverse of x. So this is the inverse matrix. This is the inverse matrix of x, multiply that by a, then I multiply that by x. So what I am doing w=x inverse a*x, so let us see whether it gives me the identity matrix, okay.

So low and behold, okay I did x inverse ax, so this is the matrix I have got. So this is exactly my diagonal matrix. Why do I say this is my diagonal matrix? Look at these values of, okay, I already have it. Remember, so my Eigen values were 15.5, 3.87, and -1.39. So when I did inverse of x multiplied by a*x, I have got 15.52, same as this diagonal value, 3.87 same as this value, and 1.3974895, 1.3974895, exactly the same as the lambda values, okay.

So we saw that using this simple command lam x=ba diagonal a, I am able to get the Eigen values and Eigen vectors. So 1 thing I must tell you that I am not going to do all the very elementary operations in Scilab that we did like adding numbers, multiplying numbers. So we did Eigen values and Eigen vectors. So what I want to do? I want to do the polynomials. So let me do the polynomials first. So how do I do that?

(Refer Slide Time: 13:33)



```
   1.          - 5.551D-17  - 1.388D-17
 - 5.551D-17     1.           5.551D-17
   2.776D-17     0.           1.

-->

-->

-->

-->

-->

-->[Lam, X] = bdiag (a)//
 X  =

   0.4609647    0.0045425  - 0.1452980
   0.2828265  - 0.4423568  - 0.7691221
   0.8411425  - 0.4080930    0.6895956
 Lam  =

   15.524082    0.           0.
   0.           3.8714073    0.
   0.           0.         - 1.3974895

-->w = inv(X) * a * X
 w  =

   15.524082    9.392D-16  - 1.530D-15
   1.845D-15    3.8714073  - 8.267D-17
   1.676D-15  - 4.197D-17  - 1.3974895
```

Roots of Polynomial Equations

// Roots given a real polynomial
p = poly([1 2 3],"x")
roots(p) // Roots, given the real
coefficients p = [3 2 1]
roots(p)
// The roots of a complex polynomial
p=poly([0,10,1+%i,1-%i],'x');
roots(p

Look at the bottom of the screen, okay. So I will say p is a polynomial. These are the coefficients = so 1 should be sure whether it is square brackets, so you remember here there are square brackets. So whenever it is a square brackets, you have to use square brackets. So square bracket 1, 2, 1. So I will use it. So what is this my function. I expect this function to be x square+2x+1, so I want to solve this equation, okay.

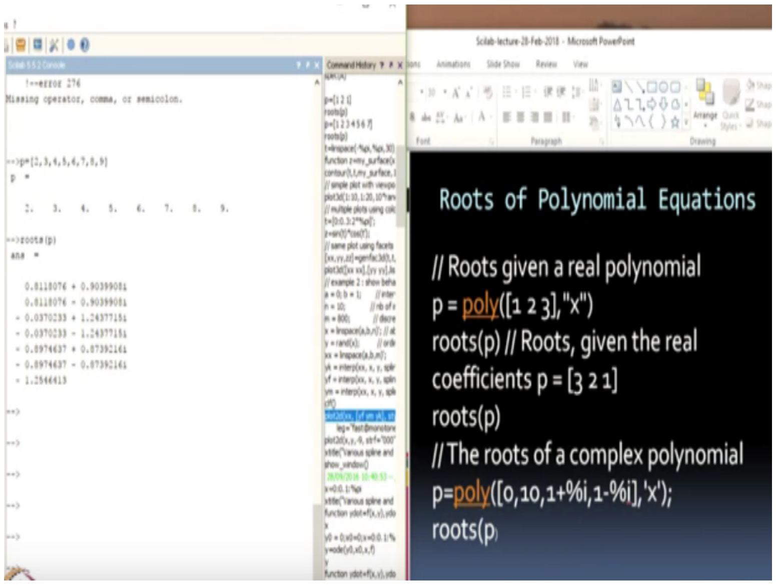
So these are the coefficients of my polynomial x square+2x+1. Now by 12 find the roots*(p) remember here I gave square brackets 1, 2, 1. Here I give round brackets, so when I say roots p, so it has given these roots. Remember when it is x square +2x+1, that function is nothing but x+1*x+1 and for that function, the roots are -1 and 1, okay, these are the roots. So you see that it very easily gives you the roots. Now let us do slightly more interesting problem.

So let us say p=let me take a more complicated polynomial p= I will give a long polynomial 3, 2. So I have taken a large polynomial now. It has many coefficients 2, 3, 4, 5, 6, 7, 8, 9. There are 8 coefficients now. So when there are 8 coefficients, so I had given some problem here. See what was the mistake? I did not give an equal to sign. So I just made p2, 3, 4. So I do, I will go back. So I should say p=all that. So I have done.

So now it gave me a polynomial with these coefficients. So since there are 8 coefficients, I will read it like this 2x to the 7th power+3x to the 6th power, 4*x5, 5*x4, 6*x cube, 7*x square, 8*x+9. So this is my polynomial which is a 7th order polynomial. Now I will get roots*(p). So it should give me 7 roots. So let us see how it is giving 7 roots, okay. Now these are the 7 roots and you know that whenever I have roots of these polynomials, the root and its complex conjugate appear as pairs.

So the first 1 is the complex root and the second root is a complex conjugate of the first root. Third 1 is a complex root, fourth 1 is a complex conjugate of that root, the fifth and sixth are again complex conjugates of each other and finally the 7th roots. So you will see that I am able to calculate all the roots of equations using this very simple command. Now I also want to execute, I will also put the Scilab slide.

(Refer Slide Time: 16:49)



```
    !--error 276
Missing operator, comma, or semicolon.


-->p=[2,3,4,5,6,7,8,9]
 p  =

    2.    3.    4.    5.    6.    7.    8.    9.

-->roots(p)
 ans  =

    0.8118076 + 0.9039908i
    0.8118076 - 0.9039908i
  - 0.0370233 + 1.2437715i
  - 0.0370233 - 1.2437715i
  - 0.8974637 + 0.8739216i
  - 0.8974637 - 0.8739216i
  - 1.2546413

-->
```

Roots of Polynomial Equations

// Roots given a real polynomial

p = poly([1 2 3],"x")

roots(p) // Roots, given the real coefficients p = [3 2 1]

roots(p)

// The roots of a complex polynomial

p=poly([0,10,1+%i,1-%i],'x');

roots(p)

So suppose I want to just express a polynomial. I want to express this side p=poly[1, 2, 3]*x. I want to do this. So let us see what it gives me, p=poly*([1, 2, 3 let me use a more involved one 1, 2, 3, 4, 5. Let me give a more complicated function], and “x”), okay, round bracket. So when I type it, so it gave me, it gave me a 5th order polynomial. Remember it was 1, 2, 3, 4, 5. So I have a 5th order polynomial, which I have generated in this way. So this is my polynomial.

And this polynomial. These are really the roots. When I give in this form, it takes 1, 2, 3, 4, 5 as roots and then gives the expression of the polynomial. Now if I want to, let me reverse the problem. How will I reverse the problem? I will say q=, okay q=now I will give these are the coefficients, remember for the 5th power, my coefficient was 1. For the fourth power, my coefficient was -15, and for the third power it was -85.

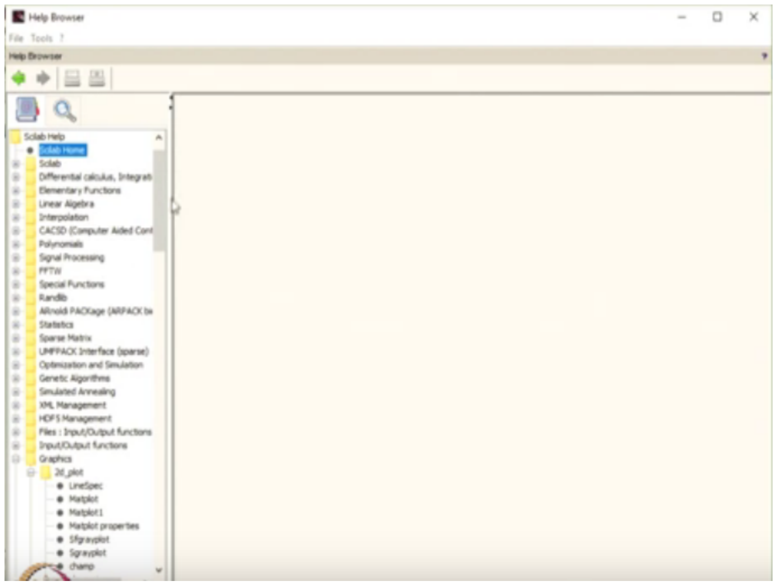
Then for the next one is -225, then it is -274, then it is -120, okay. These are my coefficients now. I am now fitting the coefficients of the polynomial*q, okay. So there is some invalid factor. Remember I gave a round bracket in place of a square bracket, so I have to correct that. So this you have to be very careful. If you give a wrong, any character that is wrong, it will mess up the whole thing. So these are my coefficients.

So whatever I did earlier, I am just giving these as coefficients. Now I will find roots cube. Roots cube, now I should recover my 5 roots, which I gave, let us see what it will do, okay. It has given different roots. I did not give me exactly what I started with. I should have recovered these original 5 roots, okay. 1, 15, 85, these are my coefficients, maybe I have done something wrong. So even if you give one coefficient wrong, it will give you wrong results, okay.

1, -15, -85, so this was +85. See in place of +85, I have given -85, so I got different results in place of +274, I have given minus, so I have got these results. Now let me correct that, okay. I will go back. So this was +85, so let me give it 85 as plus, -120, that is correct, okay. 5th power was, 4th power was -15, 3rd power was +85, 2nd power was -225, this was +274, I should change this sign as well +274, and then -120. So let us execute.

So this is my q, okay. Now so you see when I gave the correct coefficients, I got all my roots, 1, 2, 3, 4, 5. So this really tells me that Scilab is an extremely good software to do many of these things. Now I will do the next thing I want to do. I want to solve differential equations plotting and all that, but before I do that let me use a help command. So that it will tell you that because if you know how to use the help command very well, even if you do not know, you will be able to learn from the software itself.

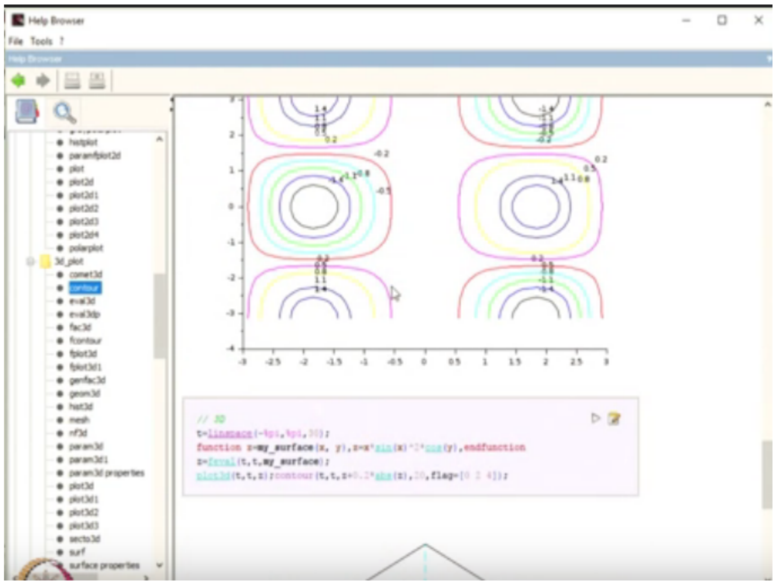
(Refer Slide Time: 21:52)



So when I typed help, so it gave me, see on the left side it gave me a large number of files here. You see there are large number of files, okay. So these are, so it tells you lots of details about Scilab, you know. It tells you what is new in Scilab. It tells you other sources. So what I want to do now, I just want to go to help command and do some good graphics, okay. So let us see in graphics I have 2D plot. I have a 3D plot.

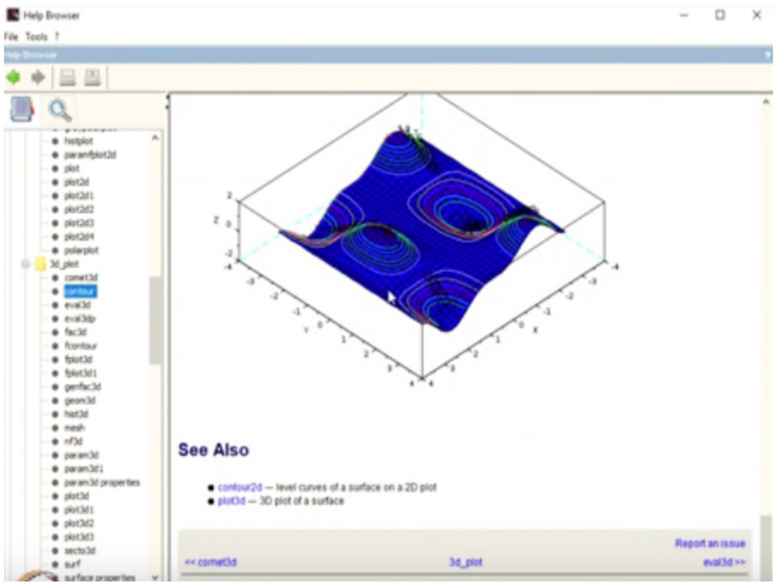
So let me go to 3D, these are all 3D plots plotting subroutines. What I want to do? Just go to that help command, whatever is there in the Scilab help, I will copy on my screen and plot it. So let us say I want to do a contour plot, okay. This contour plot gives me level curse of a 3D surface, okay. So it also tells me all kinds of example. So it is giving these examples, okay.

(Refer Slide Time: 22:53)



```
// 3D
t=linspace(-%pi,%pi,30);
function z=my_surface(x, y),z=x*sin(x)^2*cos(y),endfunction
z=feval(t,t,my_surface);
plot3d(t,t,z);contour(t,t,z+0.2*abs(z),20,flag=[0 2 4]);
```

(Refer Slide Time: 22:54)



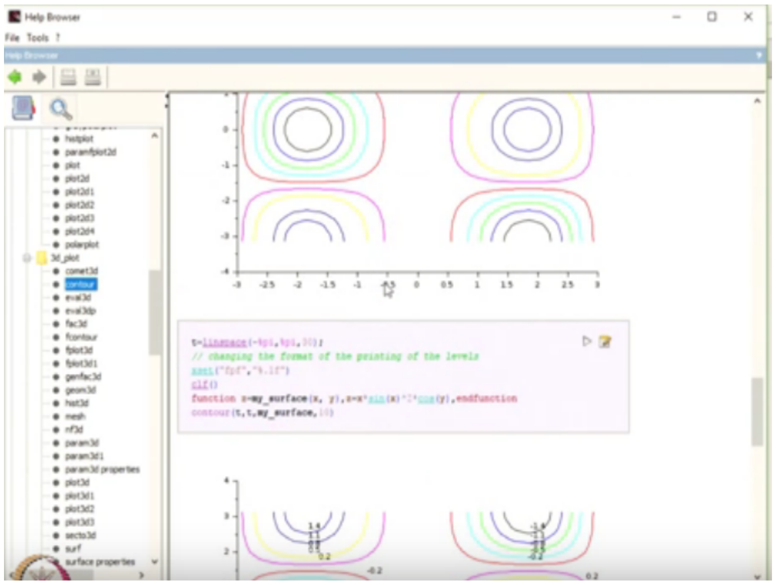
See Also

- contour2d — level curves of a surface on a 2D plot
- plot3d — 3D plot of a surface

Report an issue

<< comet3d 3d_plot eval3d >>

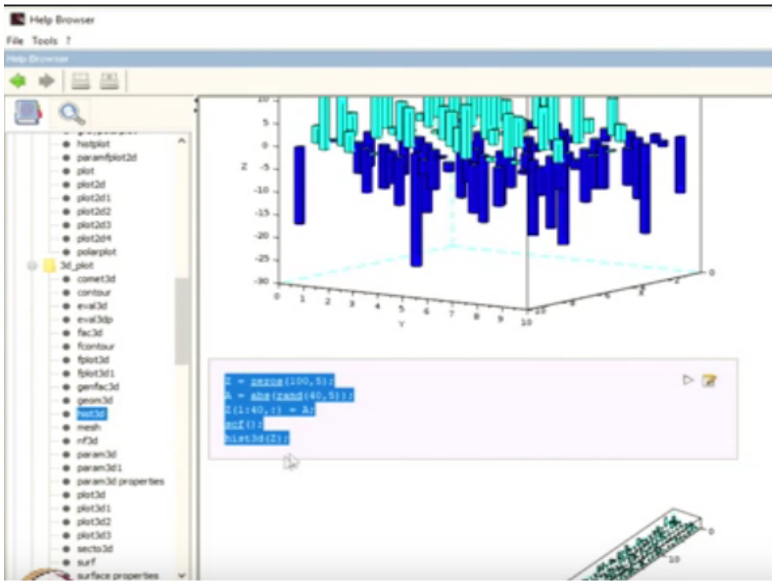
(Refer Slide Time: 22:55)



```
t=linspace(-%pi,%pi,30);
// changing the format of the printing of the levels
xset("fpf","%.1f")
clf()
function z=my_surface(x, y),z=x*sin(x)^2*cos(y),endfunction
contour(t,t,my_surface,10)
```

So to execute this, all I have to do, I have to copy, paste the lines in the Scilab directory. So this is what I will do. I will take this. I will take, where did my help go, okay. I will take, several graphs are there. I want to just copy.

(Refer Slide Time: 23:43)



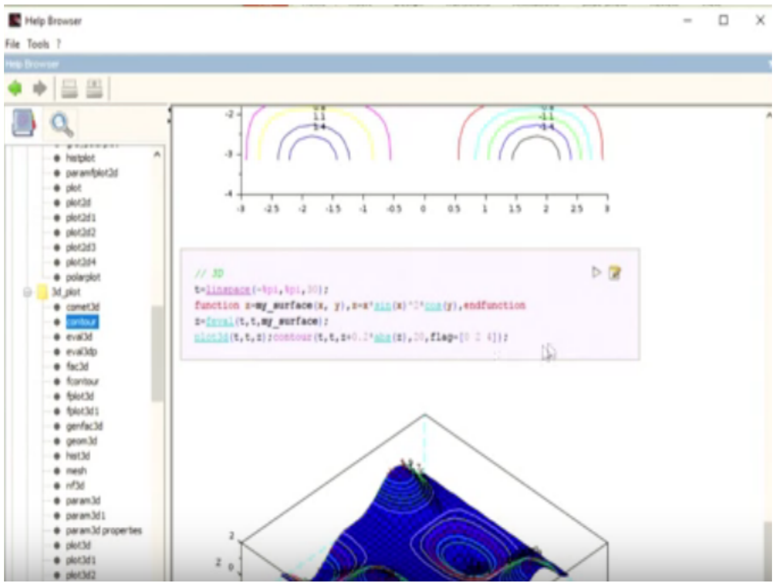
```
Z = zeros(100,5);
A = abs(rand(40,5));
Z(1:40,:) = A;
scf();
hist3d(Z);
```

Instead of what I wanted, I got these particular lines now. So z=a=and I want to apply it, so I will do copy this into Scilab, okay. So I got the Scilab console, I have pasted. Whatever lines were there in the help command, I pasted. So let us see what it is doing. So you see printed exactly what that help command said. It is plotting some graph like this, okay. So I will, let us try some other command now, okay. Let us try some other command. So I go to the help browser.

Actually I plotted this particular graph using that, I have plotted this. Now let us say I want to, these are all for 3 histograms, okay. Instead of that let us say I wanted a contour plot because they are very useful in chemistry. So let us take these contours. I want to plot these contours. These are 2 dimensional contours; these are like our wave functions for a particle in a 2 dimensional box. These are wave functions in a 2 dimensional box.

In fact, you can look at the solutions and give those solutions to plot this particular graph. So right now, I want to do a 3 dimensional plot, so I just want to copy these lines, okay.

(Refer Slide Time: 25:24)

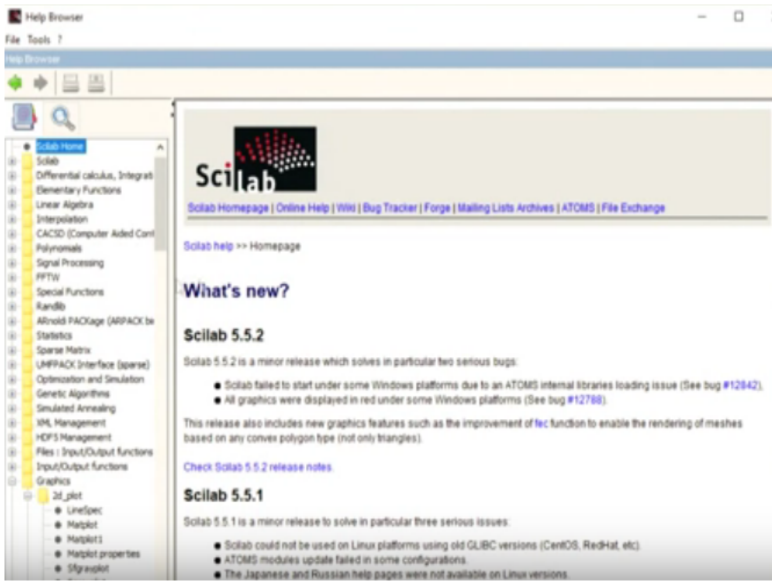


```
// 3D
t=linspace(-%pi,%pi,30);
function z=my_surface(x, y),z=x*sin(x)^2*cos(y),endfunction
z=feval(t,t,my_surface);
plot3d(t,t,z);contour(t,t,z+0.2*abs(z),20,flag=[0 2 4]);
```

So I took these lines. I will copy into Scilab environment and execute, so I have gone to Scilab environment and paste. I have pasted all those commands and now I will execute, okay. I will execute. You saw that, I exactly got the entire 3 dimensional plot that I had seen in that help command. So you will see that in addition to whatever few statements that we have given, you can also go to the Scilab help directory, okay.

Scilab help directory and you will be able to plot many, many things. You know, in fact, let me try that help again, so that you will see, okay. So I have this help browser. So let us go to this help browser again. So let us see what are all the options available, okay. In the Scilab, you have differential calculus integration.

(Refer Slide Time: 26:31)



Scilab help >> Homepage

What's new?

Scilab 5.5.2

Scilab 5.5.2 is a minor release which solves in particular two serious bugs:

- Scilab failed to start under some Windows platforms due to an ATOMS internal libraries loading issue (See bug #12842).
- All graphics were displayed in red under some Windows platforms (See bug #12788).

This release also includes new graphics features such as the improvement of fec function to enable the rendering of meshes based on any convex polygon type (not only triangles).

Check Scilab 5.5.2 release notes.

Scilab 5.5.1

Scilab 5.5.1 is a minor release to solve in particular three serious issues:

- Scilab could not be used on Linux platforms using old GLIBC versions (CentOS, RedHat, etc).
- ATOMS modules update failed in some configurations.
- The Japanese and Russian help pages were not available on Linux versions.

We have done some of this. We have elementary functions, linear algebra, interpolation, computer aided control, system design, then polynomial, signal processing, folio transforms, special functions, random library. I urge you to go to this random library and generate random numbers and see if these random numbers are better than what we generated using our programs, then you have some package, statistics, pars matrix, okay.

You have some interface optimization and simulation, curve fitting really is part of this optimization, genetic algorithm, simulated annealing, okay, xml management, some other management, input-output functions. We discussed this with. Then graphics, in graphics you have a very large number of ways to do the graphics. Then annotation, these are histograms, color management, these things are there, so some more graphics, okay.

Then in addition to graphics, you have data structures, parameters, Boolean, remember that =t and +%t, %t and %false we already discussed in our earlier thing. This is the Boolean. We have integer arithmetic strings, all programming you have done so far is only with numbers. So you can also manipulate strings and Scilab allows you to do that, okay. Time, there are output functions, okay, these are just some of these functions.

So you have already seen now how many functions are there in Scilab. So I will conclude this lecture today saying that we have discussed some of the operations using Scilab. So few more things I want to do. I want to follow a differential equation and a curve fitting. That I will do in the next slide, but by seeing all these you must be really now convinced that it is an extremely powerful tool to do many, many things that you want to do.

In fact, if I wanted to write myself all the programs that are involved in this Scilab, it will almost be impossible for an individual, because this has been created by a large number of people working together and creating this software, so use it and enjoy different aspects of Scilab. In the next class, I will consider 2 more applications, then we will go to molecular dynamic simulations. I will conclude my present lecture here. Thank you.