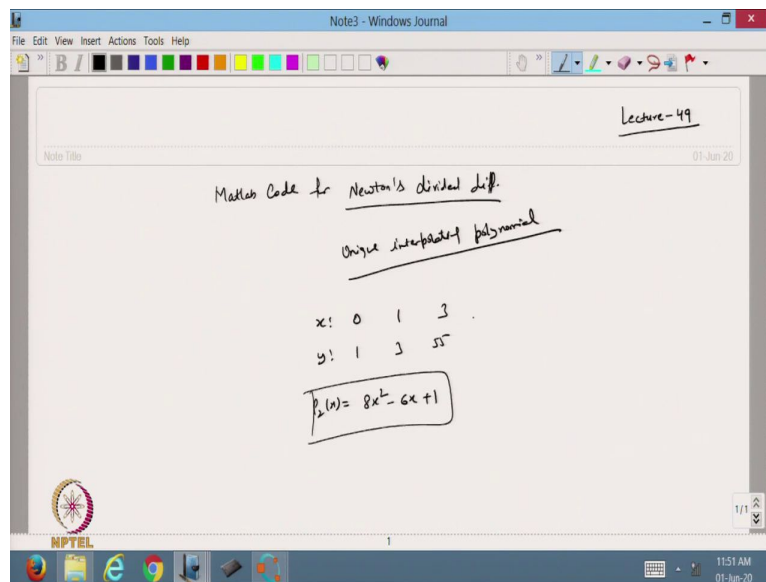


Scientific Computing Using Matlab
Professor Vivek Aggarwal
Professor Mani Mehra
Department of Mathematics
Indian Institute of Technology Delhi
Delhi Technological University
Lecture 49
MATLAB Code for Newton's Divided Difference and Least
Square Approximation

Hello viewers, welcome back to the course on scientific Computing using MATLAB. So, in the previous lecture we have started with the MATLAB code of the Lagrangian polynomial on an octave platform, so we will continue with this one.

(Refer Slide Time: 00:35)



So, today we also discuss the MATLAB code for Newton's divided difference. So, we already know that the polynomial for the given data we have seen that we will be going to have the unique interpolating polynomial, so in the previous example in the previous lecture we have taken the example that x data is given to me, so this, suppose this data is 0 1 and 3 and this y is given to me that is 1 3 55 and I know that from here we will be able to get the second degree

Lagrangian interpolating polynomial and the answer was $8x^2 - 6x + 1$. So, today we are going to solve the same one with the help of Newton divided difference.

(Refer Slide Time: 01:54)

```

Editor
File Edit View Debug Run Help
interpolatingPolyScript.m lagrange.m Laine.m newpoly.m
1 % script file for the data
2 clear all
3 clc
4 % For quadratic polynomial
5 X=[0,1,2];
6 Y=[1,3,55];
7 % For Cubic Polynomial
8 X=[0 0.4 0.8 1.2];
9 Y=[1 0.92106 0.69670 0.36235];
10 X=[0 1 4 5];
11 Y=[11 68 123];
12 % For Lagrange-----
13 [C,L]=lagrange(X,Y);
14 disp('The Lagrange's fundamental coefficients are: x^n x^(n-1)...const. ');
15 L
16 disp('The coefficients of the Lagrange inter. poly. are: ');
17 C
18 % For Newton Divided -----
19 %C
20 %disp('The Divided difference table: ');
21 %D
22 %disp('The coeff. of the Newton interpolating poly. are: x^n x^(n-1)...const. ');
23 %C
24 % square line
25 X=[1,2,3,4,5,6];
26 Y=[1,2,5,4,3,0,-1];

```

```

Editor
File Edit View Debug Run Help
interpolatingPolyScript.m lagrange.m Laine.m newpoly.m
1 function [C,L]=lagrange(X,Y)
2 % input - (X,Y) co-ordinates
3 % C- is a
4 N=length(X);
5 n=N-1;
6 L=zeros(N,N);
7 % Lagrange coefficient polynomial
8 for k=1:N
9     V=[];
10    for j=1:N
11        if k~=j
12            V=conv(V,poly(X(j)))/(X(k)-X(j));
13        end
14    end
15    L(k,:)=V;
16 end
17 % coefficient of the Lagrange interpolating polynomial
18 C=Y*L;
19 endfunction
20

```

```

1 function [C,D]=newpoly(X,Y)
2 % input:- (Xi,Yi) co-ordinates
3 % Output:- C is a vector that contains the coeff. of Newton
4 % interpolating polynomial
5 % D:- is the divided difference table
6 N=length(X);
7 D=zeros(N,N);
8 D(:,1)=Y';
9 % Divided-difference table
10 for j=2:N
11     for k=j:N
12         D(k,j)=(D(k,j-1)-D(k-1,j-1))/(X(k)-X(k-j+1));
13     endfor
14 endfor
15 % To determine the coeff. of the Newton interpolating
16 C=D(N,N);
17 for k=N-1:-1:1
18     C=conv(C,poly(X(k)));
19     m=length(C);
20     C(m)=C(m)+D(k,k);
21 endfor
22 endfunction
23

```

So, let us take this one. So, in the previous one we have discussed the Lagrange, now we will discuss the newton polynomial. Hello, viewers welcome back to the course on scientific computing using MATLAB, so in the previous lecture we have discussed about the Lagrangian the MATLAB code for the Lagrangian interpolating polynomial, so today we will continue from that one and we will try to make the MATLAB code for Newton divided difference formula. So, let us do this one.

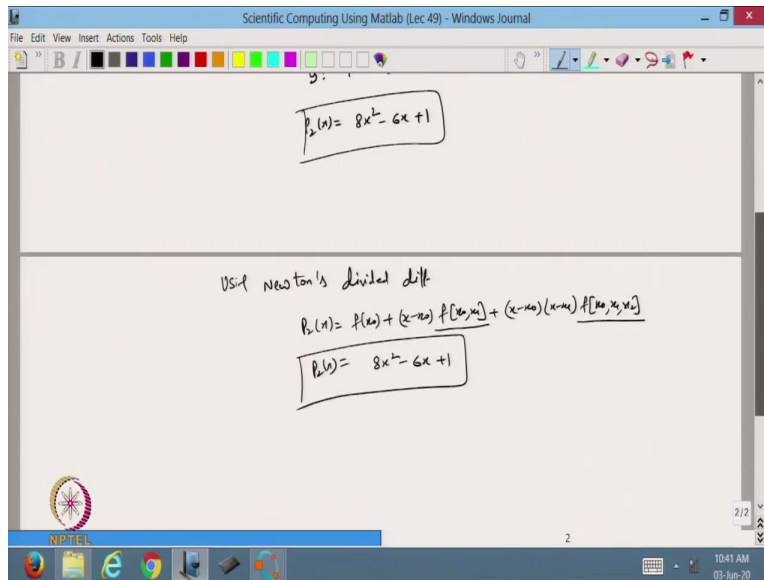
(Refer Slide Time: 02:35)

Lecture-49

Matlab Code for Newton's divided diff.
Unique interpolating polynomial

x:	0	1	3
y:	1	3	55

$$P_2(x) = 8x^2 - 6x + 1$$



So, today we are going to write the MATLAB code for Newton divided difference and I am taking the same example, which we have discussed for Lagrangian polynomials. So, this is suppose my x coordinates and this is the y coordinate and I know that if I take the Newton divided difference formula then my Newton divided difference formula so I will get my $P_2(x)$

, so that will be $P_2(x) = f(x_0) + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2]$

And if I solve this one then I should get the value $P_2(x) = 8x^2 - 6x + 1$ And this is the first order divided difference, this is a second order divided difference, so that we are going to discuss today. So, let us go to the octave.

(Refer Slide Time: 03:53)

```

Editor
File Edit View Debug Run Help

interpolatingPolyscript.m newpoly.m
1 function [C,D]=newpoly(X,Y)
2 % input:- (X,Y) co-ordinates
3 % Output:- C is a vector that contains the coeff. of Newton
4 % interpolating polynomial
5 % D:- Is the divided difference table
6 N=length(X);
7 D=zeros(N,N);
8 D(:,1)=Y';
9 % Divided-difference table
10 for j=2:N
11     for k=j-1:N
12         D(k,j)=(D(k,j-1)-D(k-1,j-1))/(X(k)-X(k-j+1));
13     endfor
14 endfor
15 % To determine the coeff. of the Newton interpolating
16 C=D(N,N);
17 for k=N-1:-1:1
18     C=conv(C,poly(X(k)));
19     m=length(C);
20     C(m)=C(m)+D(k,k);
21 endfor
22 endfunction
23
line: 1 coding: SYSTEM eof: CRLF

```

So, this is the function Newton polynomial I have made for you. So, let us start doing this one, now I am taking the name as Newton and polynomial means poly and I am inputting the X and Y coordinates. So, I am writing that input is X Y coordinates and output, what is the output? So, output I am giving C is a vector that contains the coefficients of the Newton interpolating polynomial and D is the divided difference table. Because we also have to make in this case the divided difference.

(Refer Slide Time: 04:27)

Scientific Computing Using Matlab (Lec 49) - Windows Journal

$P_2(x) = 8x^2 - 6x + 1$

Use Newton's divided diff

$$P_2(x) = f(x_0) + (x-x_0)f[x_0, x_1] + (x-x_0)(x-x_1)f[x_0, x_1, x_2]$$

$P_2(x) = 8x^2 - 6x + 1$

x_k	y_k	1 st DD	2 nd DD
x_0	y_0	$f[x_0, x_1]$	
x_1	y_1		$f[x_0, x_1, x_2]$
x_2	y_2		

So, this is my x_k this is y_k , so this is my $x_0, x_1, x_2, y_0, y_1, y_2$ and then I take the first divided difference, so this is my $f[x_0, x_1]$ and this is my $f[x_1, x_2]$, so that is the first divided difference, then the second divided difference and so on. So, that we are.
(Refer Slide Time: 05:16)

```

1 function [C,D]=newpoly(X,Y)
2 % Input:- (X1,Y1) co-ordinates
3 % Output:- C is a vector that contains the coeff. of Newton
4 % interpolating polynomial
5 % D:- is the divided difference table
6 N=length(X);
7 D=zeros(N,N);
8 D(:,1)=Y';
9 % Divided-difference table
10 for j=2:N
11     for k=j:N
12         D(k,j)=(D(k,j-1)-D(k-1,j-1))/(X(k)-X(k-j+1));
13     endfor
14 endfor
15 % To determine the coeff. of the Newton interpolating
16 C=D(N,N);
17 for k=N-1:-1:1
18     C=conv(C,poly(X(k)));
19     m=length(C);
20     C(m)=(m)+D(k,k);
21 endfor
22 endfunction
23

```

Now, in this case, so whatever the X I am passing I am taking the length of that one, so that is my N, now defining the matrix whose dimension is N cross N and I am defining like a zeros, so it gives you that a D is a matrix of dimension N cross N whose value all values are 0. Now, here because I am going to introduce the divided difference formula, so the first column will be my Y.

So, in this case I am putting D that the matrix I have introduced here, so this is that I am taking all the rows but the column number 1 is 1, so that is equal to Y, so why I am putting dash here because this Y we are giving in the row vector so that Y dash is the column vector, so the first column of this matrix will be Y.

Now, how do I make the divided difference table? Now, I will go from j from 2 to N and also k from j to N, so $D(k,j)$, so this is the kth row and they are jth column, so $D(k,j-1)-D(k-1,j-1)/(X(k)-X(k-j+1))$ so whatever it is there, so this is the way we can define the difference. So, that is the difference of the in the Y coordinates divided by difference in the X coordinates.

So, this is the way we are introducing the divided difference table. Now, j is giving you the columns and k is the row, so if I take $j = 2$ means I am working in the column number 2 and k is moving from 2 to N , so k is moving from 2 to N , then it is giving you the all the corresponding differences in the second column, similarly $j = 3$ that is giving you the second difference in the third column and so on. So, in this way, we are going to introduce the first divided difference, second divided difference and so on.

Now, C is equal to D and N , so D and N is the last element we are going to have, so that is my C . So, why am I taking this one C ? Because C is the vector that contain the coefficient of the Newton interpolating polynomial, so I know that in the end of that column if I keep going in the each of the column that is giving me the first divided difference, second divided difference, third divided difference and suppose it depends upon the number of value, so in the case of suppose, I have three value, then I know that the I can go up to second order difference.

So, in this case if I have the three value my zeros will be 3 by 3, so this is my matrix 3 by 3, so first column will be Y , the second column will be the first divided difference and the third column will be the second divided difference and that difference contains only one value, so this is my $C = D(N,N)$, so I am doing this one. Now, what do I do? I am taking all the coefficients, so I am starting from the $k = N-1$ with the increment of -1 and going up to 1.

Now, I am introducing the same way that is the so the, in the previous class also I have started with the $\text{poly}(X)$, so $\text{poly}(X)$ is the gives me the polynomial having the root $X(k)$ and now I am doing the convolution of this one with this C , so this C will multiply by this polynomial and that will save to the new C , m is equal to the length of C , so whatever the length of C is there so that will give the length of C and $C(m) = C(m) + D(k,k)$, so whatever the coefficient I am taking we are plus the corresponding in the difference we are taking, so that is the difference we are adding and this is the end of this loop and this is the end of the function. So, this way we are able to define the Newton divided difference polynomial.

(Refer Slide Time: 09:33)


```

1 % script file for the data
2 clear all
3 clc
4 close all
5 % For quadratic polynomial
6 X=[0,1,3];
7 Y=[1,3,55];
8 % For Cubic Polynomial
9 X=[0 0.4 0.8 1.2];
10 Y=[1 0.92106 0.69670 0.36235];
11 X=[0 1 4 5];
12 Y=[8 11 68 123];
13 % For Lagrange-----
14 [C,L]=lagrange(X,Y);
15 disp('The Lagrange's fundamental coefficients are: x^n x^(n-1)...const. ');
16 %
17 disp('The coefficients of the Lagrange inter. poly. are');
18 %
19 lpoly=polyval(C,X);
20 plot(X,lpoly,'ko-')
21 % For Newton Divided -----
22 [C,L]=lagrange(X,Y);
23 %
24 disp('The Divided difference table');
25 %
26 % The coeff. of the Newton interpolation poly. are: x^n x^(n-1) ...const. %

```

Now, I will this is my script file, so here I am going to introduce the Newton divided difference, so this is my Newton divided difference. Now, I am taking the same X and Y and from here this is a Newton divided difference, so control V.

(Refer Slide Time: 10:05)

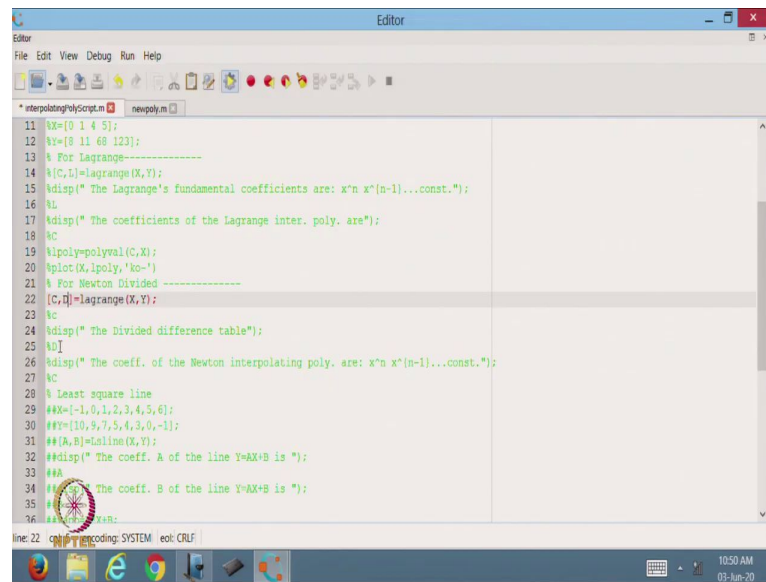
```

1 % script file for the data
2 clear all
3 clc
4 close all
5 % For quadratic polynomial
6 X=[0,1,3];
7 Y=[1,3,55];
8 % For Cubic Polynomial
9 X=[0 0.4 0.8 1.2];
10 Y=[1 0.92106 0.69670 0.36235];
11 X=[0 1 4 5];
12 Y=[8 11 68 123];
13 % For Lagrange-----
14 [C,L]=lagrange(X,Y);
15 disp('The Lagrange's fundamental coefficients are: x^n x^(n-1)...const. ');
16 %
17 disp('The coefficients of the Lagrange inter. poly. are');
18 %
19 lpoly=polyval(C,X);
20 plot(X,lpoly,'ko-')
21 % For Newton Divided -----
22 [C,L]=lagrange(X,Y);
23 %
24 disp('The Divided difference table');
25 %
26 % The coeff. of the Newton interpolation poly. are: x^n x^(n-1) ...const. %

```

And I am taking this as my X and this is my Y, the same Y, I am taking as we have taken for the Lagrange also.

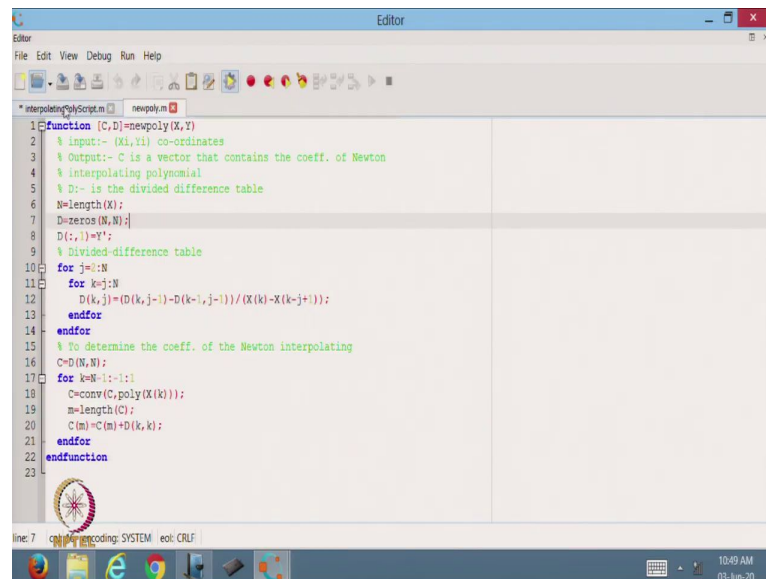
(Refer Slide Time: 10:18)



```
Editor
File Edit View Debug Run Help

* interpolatingPolyScript.m newpoly.m

11 %X=[0 1 4 5];
12 %Y=[8 11 68 123];
13 % For Lagrange-----
14 %C,L=Lagrange(X,Y);
15 %disp(' The Lagrange's fundamental coefficients are: x^n x^{n-1}...const. ');
16 %L
17 %disp(' The coefficients of the Lagrange inter. poly. are ');
18 %C
19 %poly=polyval(C,X);
20 %plot(X,Lpoly,'ko-')
21 % For Newton Divided -----
22 %C,D=Lagrange(X,Y);
23 %C
24 %disp(' The Divided difference table ');
25 %D
26 %disp(' The coeff. of the Newton interpolating poly. are: x^n x^{n-1}...const. ');
27 %C
28 % Least square line
29 %X=[-1,0,1,2,3,4,5,6];
30 %Y=[10,9,7,5,4,3,0,-1];
31 % [A,B]=lsline(X,Y);
32 %disp(' The coeff. A of the line Y=AX+B is ');
33 %A
34 %disp(' The coeff. B of the line Y=AX+B is ');
35 %B
36 %A,B,C,X,Y;
```



```
Editor
File Edit View Debug Run Help

* interpolatingPolyScript.m newpoly.m

1 function [C,D]=newpoly(X,Y)
2 % input:- (X,Y) co-ordinates
3 % Output:- C is a vector that contains the coeff. of Newton
4 % interpolating polynomial
5 % D:- is the divided difference table
6 N=length(X);
7 D=zeros(N,N);
8 D(:,1)=Y';
9 % Divided-difference table
10 for j=1:N
11 for k=j:N
12 D(k,j)=(D(k,j-1)-D(k-1,j-1))/(X(k)-X(k-j+1));
13 endfor
14 endfor
15 % To determine the coeff. of the Newton interpolating
16 C=D(N,N);
17 for k=N:-1:1
18 C=conv(C,poly(X(k)));
19 m=length(C);
20 C(m)=C(m)+D(k,k);
21 endfor
22 endfunction
23
```

So, in this case if you see my output is C and D, so my output is C and then I will write here that is D.

(Refer Slide Time: 10:38)

```

Editor
File Edit View Debug Run Help

* interpolatingPolyScript.m newpoly.m

11 %x=[0 1 4 5];
12 %y=[8 11 60 123];
13 % For Lagrange-----
14 [C,L]=lagrange(X,Y);
15 %disp(" The Lagrange's fundamental coefficients are: x^n x^(n-1)...const.");
16 %L
17 %disp(" The coefficients of the Lagrange inter. poly. are");
18 %C
19 %poly=polyval(C,X);
20 %plot(X,Lpoly,'ko-')
21 % For Newton Divided -----
22 [C,D]=lagrange(X,Y);
23 %C
24 %disp(" The Divided difference table");
25 %D
26 %disp(" The coeff. of the Newton interpolating poly. are: x^n x^(n-1)...const.");
27 %C
28 % Least square line
29 %X=[-1,0,1,2,3,4,5,6];
30 %Y=[10,9,7,5,4,3,0,-1];
31 % [A,B]=LsLine(X,Y);
32 %disp(" The coeff. A of the line Y=AX+B is ");
33 %A
34 %disp(" The coeff. B of the line Y=AX+B is ");
35 %B
36 %Bm:

line: 27  coding: SYSTEM eol: CRLF

```

```

Editor
File Edit View Debug Run Help

* interpolatingPolyScript.m newpoly.m Save File and Run

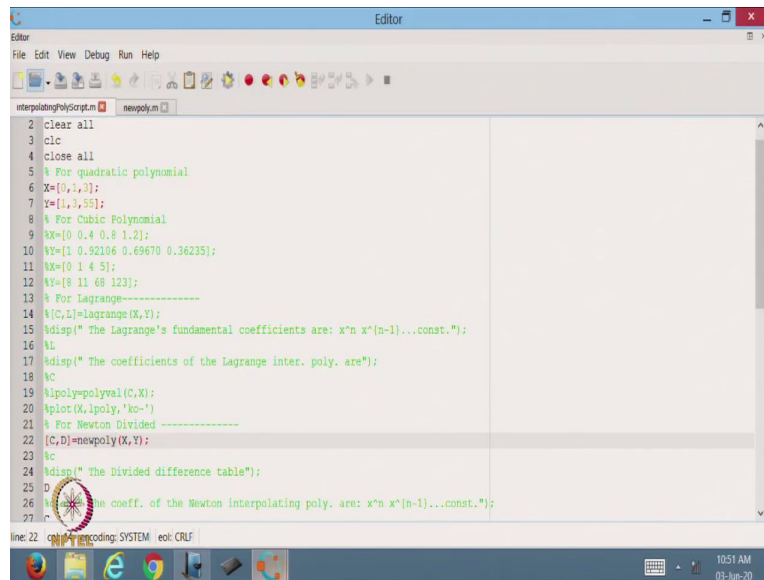
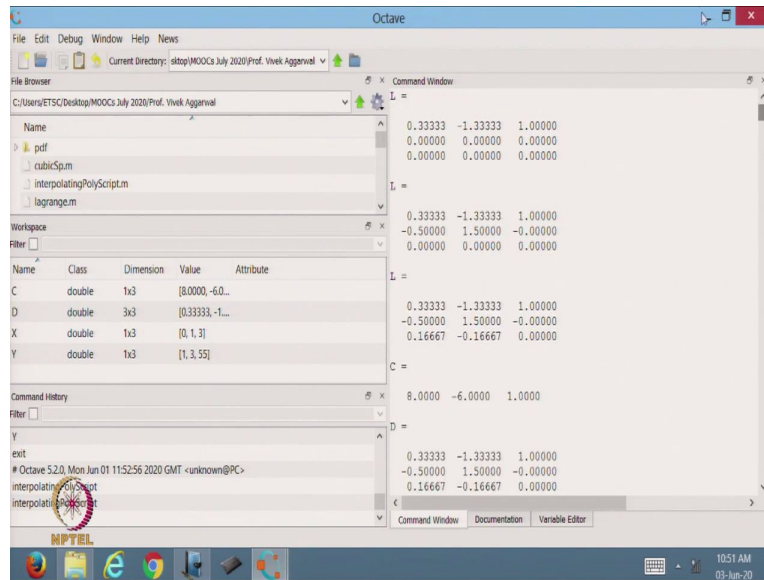
31 % [A,B]=LsLine(X,Y);
32 %disp(" The coeff. A of the line Y=AX+B is ");
33 %A
34 %disp(" The coeff. B of the line Y=AX+B is ");
35 %B
36 %yapp=A*X+B;
37 %plot(X,Y,'bo',X,yapp,'k-');
38 %-----
39 %x=[0,1,2,3];
40 %y=[0,0,0.5,2,0,1.5];
41 %x=[0,1,8,27];
42 %y=[0,1,2,3]; % Y=X^1/3
43 %S=cubicSp(X,Y);
44 %disp(' Rows of S are precisely the coeff. of the cubic spline');
45 %S
46 % Now to plot the cubic spline for 4 points
47 %x1=X(1):.01:X(2); y1=polyval(S(1,:),x1-X(1));
48 %x2=X(2):.01:X(3); y2=polyval(S(2,:),x2-X(2));
49 %x3=X(3):.01:X(4); y3=polyval(S(3,:),x3-X(3));
50 % Evaluating the value at any x
51 % x=21;
52 % App=polyval(S(3,:),x-8)
53 %plot(x1,y1,x2,y2,x3,y3,X,Y,'o');
54
55

line: 53  coding: SYSTEM eol: CRLF

```

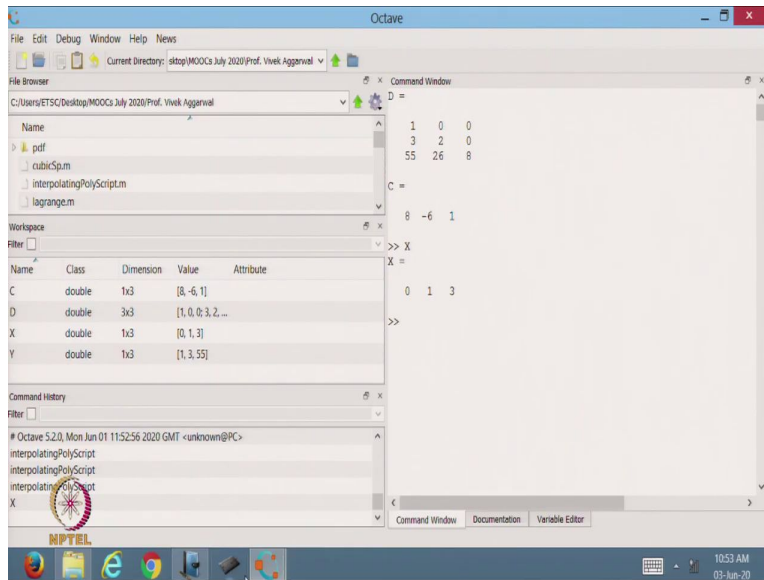
Now, I put on the display, this is my divided difference and this is my, the coefficients. So, let us run this one. So, let us try to run this one.

(Refer Slide Time: 11:08)



So, this is the, so this is my C and this is D I am also writing somewhere I am also, no this is not here, so I am writing this as new, new poly, so that is my function we have to do. So, from here, I will run this one and then I will try to see the output.

(Refer Slide Time: 11:55)



So, see from here the D is a 3 by 3 matrix, the first row gives you first column gives you the value of Y, then this is the difference, so it is 1 3 55 and this is $3-1=2$ because the value of X is given to me, so this is my X, so the difference between this is 1 and this is 2, so $3-1/1=2$ and $55-3/(3-1)=26$. And from here, I am getting the second difference so the second difference will be $26-2/(3)=8$.

So, this is a second order difference. So, based on this one I can find my D and this is the coefficient, so 8 -6, 1 so this will give you $8x^2 - 6x + 1$, so that gives you the same Newton polynomial whatever we are also able to find in the case of a Lagrangian polynomial. So, that is my function the Newton functions for the data that is 3 by 3 data.

(Refer Slide Time: 13:19)

```

Editor
File Edit View Debug Run Help
* interpolatingPolyScript.m newpoly.m
2 clear all
3 clc
4 close all
5 % For quadratic polynomial
6 X=[0,1,3];
7 Y=[1,3,55];
8 % For Cubic Polynomial
9 X=[0 0.4 0.8 1.2];
10 Y=[1 0.92106 0.69670 0.36235];
11 X=[0 1 4 5];
12 Y=[8 11 68 123];
13 % For Lagrange-----
14 [C,L]=lagrange(X,Y);
15 %disp(" The Lagrange's fundamental coefficients are: x^n x^(n-1)...const.");
16 %
17 %disp(" The coefficients of the Lagrange inter. poly. are");
18 %
19 %poly=polyval(C,X);
20 %plot(X,Lpoly,'ko-')
21 % For Newton Divided -----
22 [C,D]=newpoly(X,Y);
23 %
24 %disp(" The Divided difference table");
25 %
26 %disp(" The coeff. of the Newton interpolating poly. are: x^n x^(n-1)...const.");
27 %
line:10

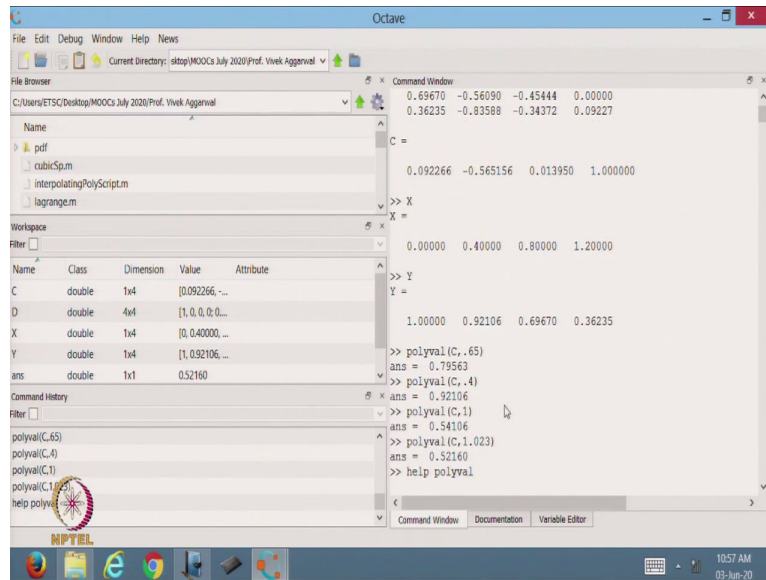
```

So, let us take the other data, the little bit bigger one, now what I do is I will try to take it the bigger one, this is my X and this is my Y and let us try to run this one.
(Refer Slide Time: 13:53)

```

Octave
File Edit Debug Window Help News
Current Directory: /stop/MOOCs July 2020/Prof. Vivek Aggarwal
File Browser
C:\Users\ETSC\Desktop\MOOCs July 2020\Prof. Vivek Aggarwal
Name
> pdf
cubicSp.m
interpolatingPolyScript.m
lagrange.m
Workspace
Filter
Name Class Dimension Value Attribute
C double 1x4 [0.092266, ...
D double 4x4 [1, 0, 0, 0; 0, ...
X double 1x4 [0.40000, ...
Y double 1x4 [1.092106, ...
ans double 1x1 0.92106
Command History
Filter
interpolatingPolyScript
X
Y
polyval(C,65)
polyval(C,4)
polyval(C,4)
>> polyval(C,65)
ans = 0.79563
>> polyval(C,4)
ans = 0.92106
>> polyval(C,4)
ans = 0.92106
>> polyval(C,4)
ans = 0.92106
Command Window Documentation Variable Editor
D =
1.00000 0.00000 0.00000 0.00000
0.92106 -0.19735 0.00000 0.00000
0.69670 -0.56090 -0.45444 0.00000
0.36235 -0.83588 -0.34372 0.09227
C =
0.092266 -0.565156 0.013950 1.000000
>> X
X =
0.00000 0.40000 0.80000 1.20000
>> Y
Y =
1.00000 0.92106 0.69670 0.36235
>> polyval(C,65)
ans = 0.79563
>> polyval(C,4)
ans = 0.92106
>> polyval(C,4)
ans = 0.92106
>> polyval(C,4)
ans = 0.92106

```



So, that is my value. So, in this case my X is this and my Y is this, so X has the this is a difference 4 4 and 0.4, 0.4 and 0.4, so that is the uniform distributed data here that is given to us, now from here, so this is uniform data is given to me and this is the value of Y. Now, the first column is the 1.92, 0.69, 0.36, so this is the first column that gives you the value Y that is the difference between these two, so this is the first divided difference, this is the second divided difference and this is the third divided difference. So, that is why you know that I was able to substitute $C = D(N,N)$, so this is $D(N,N)$, the last divided difference.

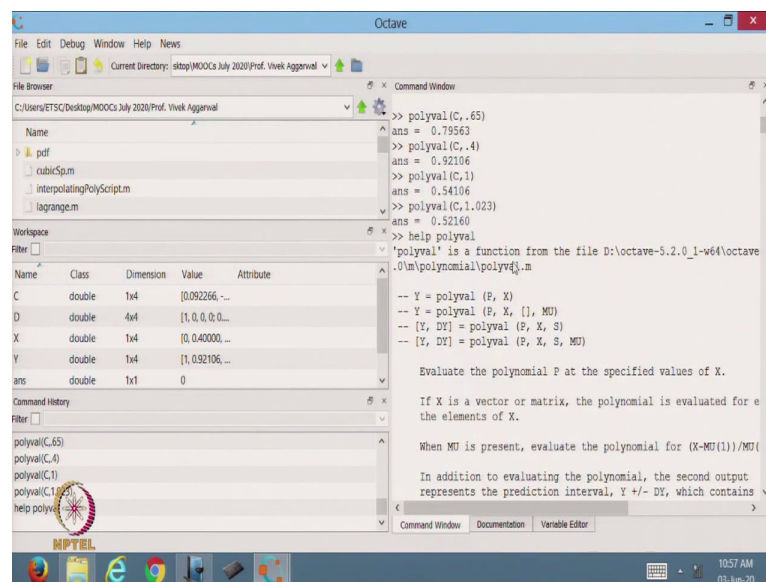
So, from here, I am getting the value of C, so these are the coefficients $0.092266x^3 - 0.565156x^2 + 0.023950x + 1$ so that is the value of this. So, now suppose I this is my Newton divided difference formula for the cubic, now suppose I want to find the value of any approximate the any value of X in between these data, so there is a command for this one, so I will write polyval, so polyval means I am taking the polynomial, that is C and I want to find its value at suppose, so the X is given to me here, so I want to find the values 8.65, so that is the value of this one.

Suppose I want to find this value at 0.4, so at 0.4 this is given 0.92106 and also it is because this is a polynomial interpolating polynomial so it is passing through the points, so that is why its value is coming the same. Similarly, I can define the value of these as suppose I want to find its

value at 1, so that is the value 0.4106.

So, once you know the value of this coefficient, this coefficient means basically this is the polynomial we have. Now once I know the cubic polynomial suppose I want to approximate any values of X in between this values, so I just put the value polyval, so polyval is the polynomial value, so that c is my polynomial and I want to evaluate its value at 0.65 so that gives value, where I want to find 1.023, so that is the value I am going to get. And if you want to see then you can write help polyval. So, that gives you what is the meaning of this polyval.

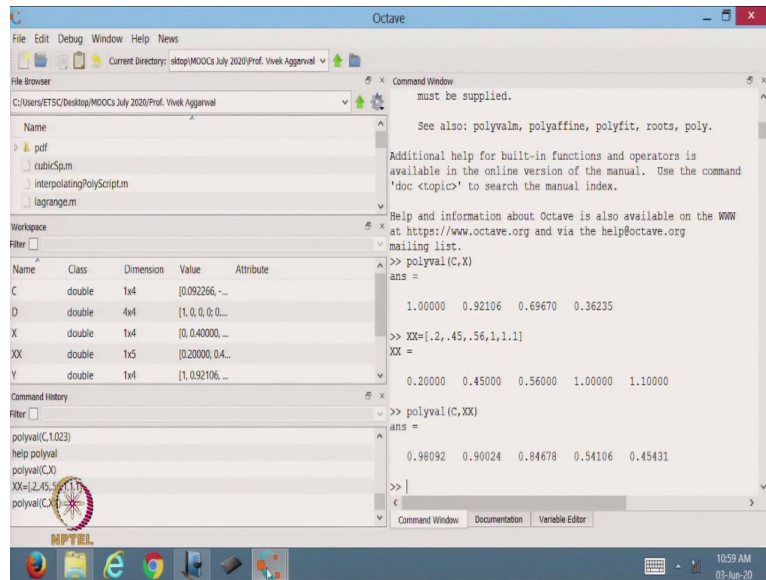
(Refer Slide Time: 17:08)



```
>> polyval(C, 0.65)
ans = 0.79563
>> polyval(C, 4)
ans = 0.92106
>> polyval(C, 1)
ans = 0.54106
>> polyval(C, 1.023)
ans = 0.52160
>> help polyval
'polyval' is a function from the file D:\octave-5.2.0_1-w64\octave
.\0\m\polynomial\polyval.m
-- Y = polyval (P, X)
-- Y = polyval (P, X, [], MU)
-- [Y, DY] = polyval (P, X, S)
-- [Y, DY] = polyval (P, X, S, MU)
Evaluate the polynomial P at the specified values of X.
If X is a vector or matrix, the polynomial is evaluated for e
the elements of X.
When MU is present, evaluate the polynomial for (X-MU(1))/MU(
In addition to evaluating the polynomial, the second output
represents the prediction interval, Y +/- DY, which contains
```

So, polyval is the function form of the file polynomial so Y is a polyval evaluating the polynomial P at a specified value of X. So this is a polynomial P at the specified values of X. So, if X is a vector or matrix, then we also can pass this one directly and it will give you the value of each of the X.

(Refer Slide Time: 17:37)



Like suppose I want to write like this one, I just write `polyval C` and pass this `X`, so let us see. So, it gives the value of a polynomial for complete `X`. So, I put the value of `X` I get the solution and this is my first column, suppose I define another one `X` and suppose I want to find the value at maybe 0.2 then 0.45 then maybe 0.56, 1 and 1.1,

So this is the five values I want to find out, directly, so this is my `X`, `X` I have taken, now I will pass this `X`. `X` to here and that is the value. So, that is the value we are going to get from here using the Newton divided difference. So I use the Newton divided difference and this is the values of the `X` I have approximated for this value of `X`. So, this is the way we can define the Newton divided difference using MATLAB.

(Refer Slide Time: 19:02)

```

1 % script file for the data
2 clear all
3 clc
4 close all
5 % For quadratic polynomial
6 X=[0,1,3];
7 Y=[1,3,55];
8 % For Cubic Polynomial
9 X=[0 0.4 0.8 1.2];
10 Y=[1 0.92106 0.69670 0.36235];
11 X=[0 1 4 5];
12 Y=[8 11 68 123];
13 % For Lagrange-----
14 % [C,L]=lagrange(X,Y);
15 % disp("The Lagrange's fundamental coefficients are: x^n x^{n-1}...const.");
16 %
17 % disp("The coefficients of the Lagrange inter. poly. are");
18 %
19 % polyval(C,X);
20 % plot(X,ipoly,'k-')
21 % For Newton Divided-----
22 [C,D]=newpoly(X,Y);
23 %
24 % disp("The Divided difference table");
25 %
26 % disp("The coeff. of the Newton interpolation poly. are: x^n x^{n-1}...const.");

```

I can also take another example and now somebody asked me that suppose we want to plot this function, so let us try to plot this also.

(Refer Slide Time: 19:19)

Name	Class	Dimension	Value	Attribute
C	double	1x4	[0.092266, ...]	
D	double	4x4	[1, 0, 0, 0; ...]	
X	double	1x4	[0.040000, ...]	
XX	double	1x5	[0.20000, 0.4...	
Y	double	1x4	[1.092106, ...]	

```

>> polyval(C,X)
ans =
    1.00000    0.92106    0.69670    0.36235

>> XX=[.2,.45,.56,1,1.1]
XX =
    0.20000    0.45000    0.56000    1.00000    1.10000

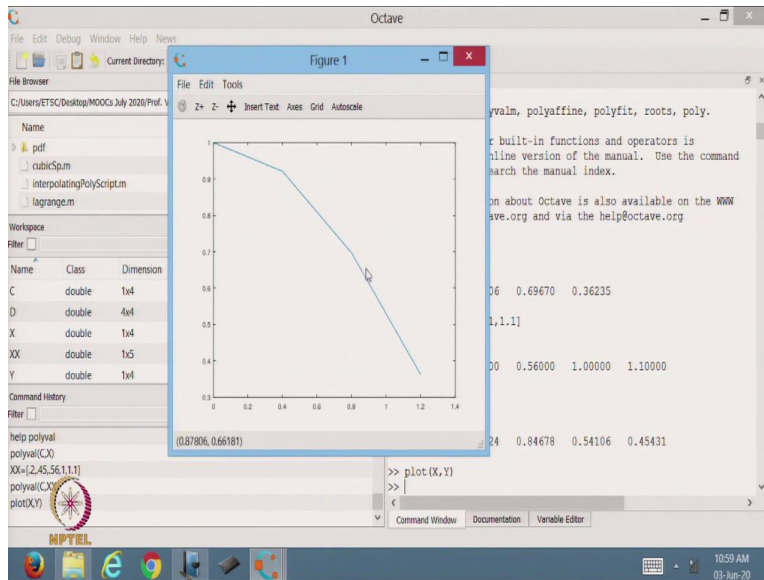
>> polyval(C,XX)
ans =
    0.98092    0.90024    0.84678    0.54106    0.45431

>> plot(X,Y)

```

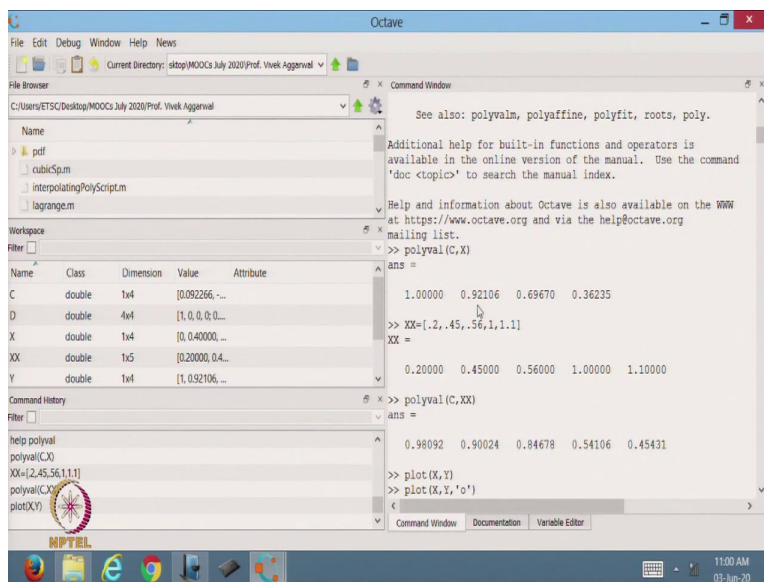
So, this is the value of x, so just I plot this for the corresponding X and corresponding Y.

(Refer Slide Time: 19:27)



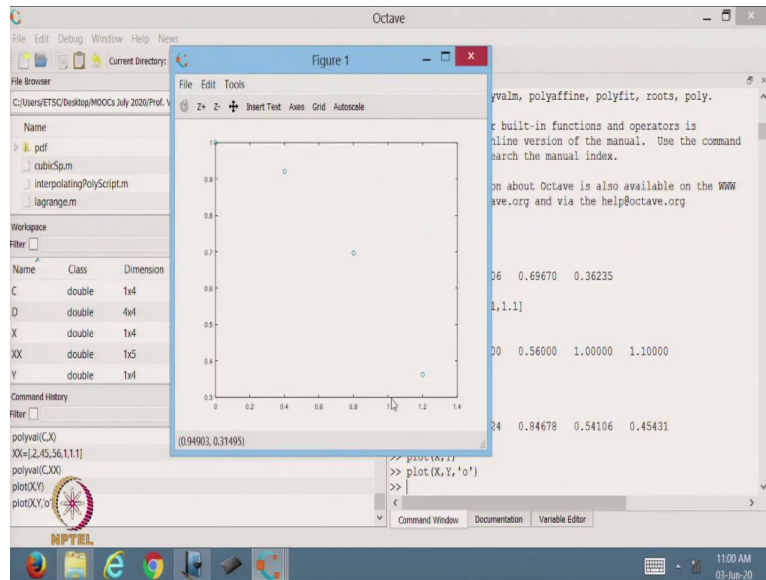
So, that is the value we are getting at, so this is my X this is a value we are going getting, so that is value is this, it is showing only you that that if I have the value of X and Y and if I try to plot that one, then it is giving you this values or maybe I can find this as.

(Refer Slide Time: 19:50)



I will write this plot and then I will just write maybe this.

(Refer Slide Time: 20:02)



So that is the, so this is the data points we got, at 0 its value is 1, 0.4, 0.8 and 1.2, so this is a value we are going to get in this case.

(Refer Slide Time: 20:20)

The screenshot shows the Octave Editor window with a script for polynomial interpolation. The script is as follows:

```

1 % script file for the data
2 clear all
3 clc
4 close all
5 % For quadratic polynomial
6 X=[0,1,3];
7 Y=[1,3,5];
8 % For Cubic Polynomial
9 X=[0 0.4 0.8 1.2];
10 Y=[1 0.69670 0.36235];
11 X=[0 1 4 5];
12 Y=[8 11 68 123];
13 % For Lagrange
14 [C,L]=lagrange(X,Y);
15 %disp('The Lagrange's fundamental coefficients are: x^n x^(n-1)...const. ');
16 %
17 %disp('The coefficients of the Lagrange inter. poly. are');
18 %
19 %polyval(C,X);
20 %plot(X,ipoly,'ko')
21 % For Newton Divided
22 [C,D]=newpoly(X,Y);
23 %
24 %disp('The Divided difference table');
25 D
26 %disp('The coeff. of the Newton interpolation poly. are: x^n x^(n-1)...const. ');

```

After doing this one.

(Refer Slide Time: 20:41)

```

Editor
File Edit View Debug Run Help
interpolatingPolyscript.m newpoly.m Lsline.m
1 % Least Square Line
2 function [A,B]=Lsline(X,Y)
3 %Input:- (X,Y) co-ordinates
4 % Output:- coefficients of the line Ax=B
5 N=length(X);
6 sumx=sum(X);
7 sumy=sum(Y);
8 sum1=sum(X.*X);
9 sum2=sum(X.*Y);
10 % Now we need to write the system Mx2=K of equations
11 M=[sum1,sumx;sumx,N];
12 R=[sum2;sumy];
13 sol=M\R;
14 A=sol(1);
15 B=sol(2);
16 endfunction
17

```

This one, so let us take the next topic and that is a least square line. So, we know that in the least square line, I have the data and that data need not be, the function need not be passing through the data, then how can we approximate that given data with a line? So, that we have done for the least square methods, so in this case also, I will write the name Ls line, so least square line and I am passing X and Y here and from there because in this case I am representing this with a line. (Refer Slide Time: 21:27)

Scientific Computing Using Matlab (Lec 49) - Windows Journal

Use of Newton's divided diff

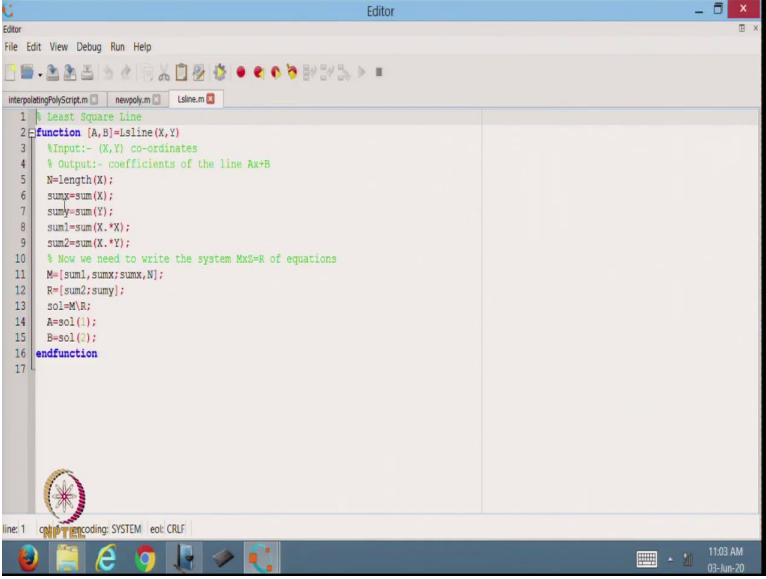
$$P_2(x) = f(x_0) + (x-x_0)f[x_0,x_1] + (x-x_0)(x-x_1)f[x_0,x_1,x_2]$$

$P_2(x) = 8x^2 - 6x + 1$

x_k	y_k	1 st DD	2 nd DD
x_0	y_0	$f[x_0,x_1]$	
x_1	y_1	$f[x_1,x_2]$	
x_2	y_2		

$y = 8x^2 - 6x + 1$

2/2

A screenshot of the MATLAB Editor window. The title bar says 'Editor'. The menu bar includes 'File', 'Edit', 'View', 'Debug', 'Run', and 'Help'. The toolbar contains various icons for file operations, editing, and running. The main editor area shows a script named 'Least Square Line' with the following code:

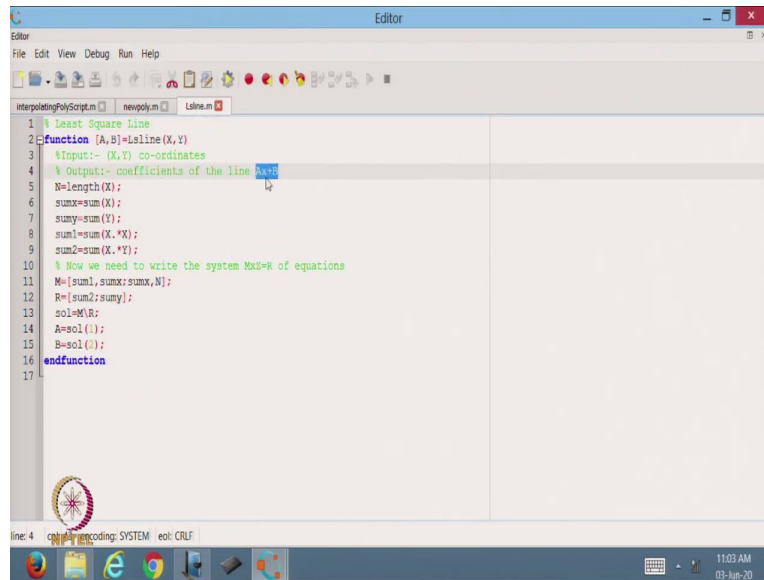
```
1 % Least Square Line
2 function [A,B]=Lsline(X,Y)
3 %Input:- (X,Y) co-ordinates
4 % Output:- coefficients of the line Ax+B
5 N=length(X);
6 sumx=sum(X);
7 sumy=sum(Y);
8 sum1=sum(X.*X);
9 sum2=sum(X.*Y);
10 % Now we need to write the system Mx2=K of equations
11 M=[sum1,sumx;sumx,N];
12 R=[sum2;sumy];
13 sol=M\R;
14 A=sol(1);
15 B=sol(2);
16 endfunction
17
```

The status bar at the bottom shows 'line: 1', 'coding: SYSTEM', 'eol: CRLF', and the system clock '11:03 AM 03-Jun-20'.

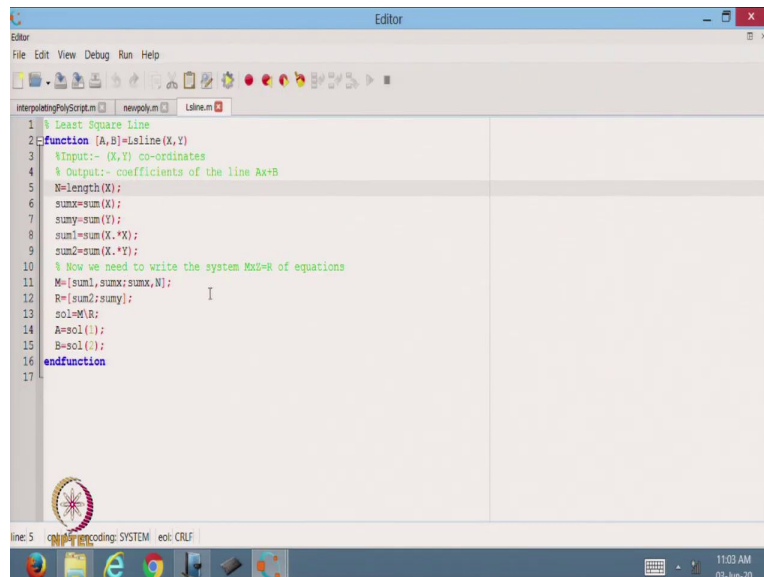
So, suppose this is the line we are going to get I am going to have a line $Y = AX + B$, so I want to find what is this A the coefficient of X and this is the B. So, this one I want to find using the least square line. So, let us try to find out.

Now, what I do in this case is first I find the length of X that and that is capital N, then I find the value X the sum X, so I put it as a x then I put sum Y that is I call it sum Y, then I know that if I am taking the line $AX+B$, then I need to find the sum of X square and sum of XY also, so I define here is, first I define X multiply by X, because X is a vector so that is why I put the dot star. So, this gives you the X square and I put the sum, so I call it sum1. And then I will try to find the sum of X Y, so this is the two vectors I am multiplying and I am taking the sum.

(Refer Slide Time: 22:43)



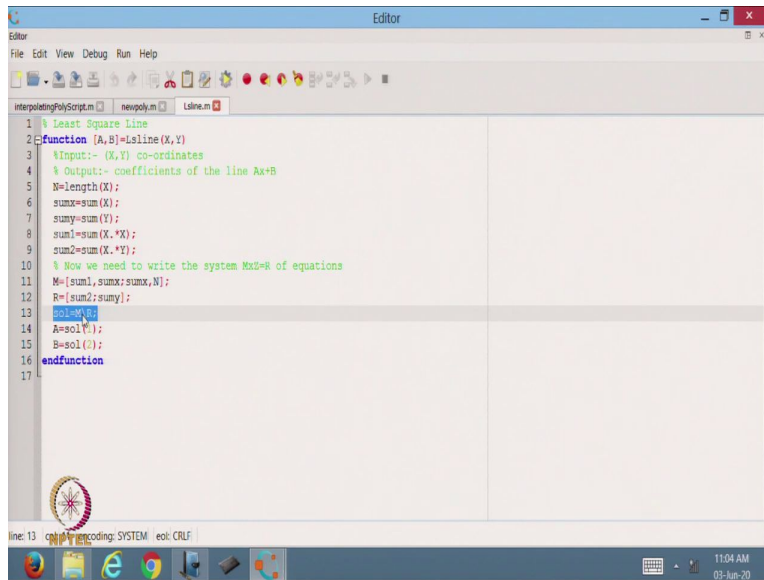
```
1 % Least Square Line
2 function [A,B]=Lsline(X,Y)
3 % Input:- (X,Y) co-ordinates
4 % Output:- coefficients of the line Ax+B
5 N=length(X);
6 sumx=sum(X);
7 sumy=sum(Y);
8 sum1=sum(X.*X);
9 sum2=sum(X.*Y);
10 % Now we need to write the system Mx2=N of equations
11 M=[sum1,sumx;sumx,N];
12 R=[sum2;sumy];
13 sol=M\R;
14 A=sol(1);
15 B=sol(2);
16 endfunction
17
```



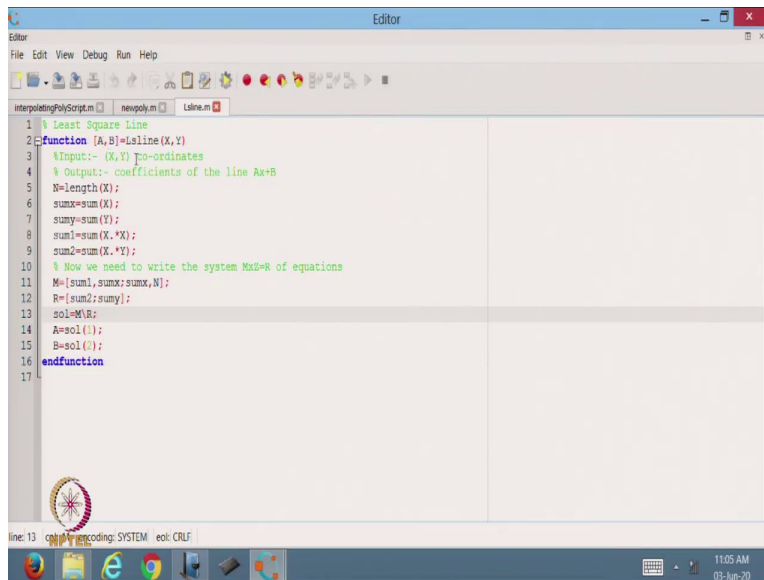
```
1 % Least Square Line
2 function [A,B]=Lsline(X,Y)
3 % Input:- (X,Y) co-ordinates
4 % Output:- coefficients of the line Ax+B
5 N=length(X);
6 sumx=sum(X);
7 sumy=sum(Y);
8 sum1=sum(X.*X);
9 sum2=sum(X.*Y);
10 % Now we need to write the system Mx2=N of equations
11 M=[sum1,sumx;sumx,N];
12 R=[sum2;sumy];
13 sol=M\R;
14 A=sol(1);
15 B=sol(2);
16 endfunction
17
```

So, this is the line I want to find out, so $AX+B$, so this one I want to do, now what I do is that we need to find the so from here we know that we get the system of equation, so the system of equation this is a matrix I get so that is the sum1 as a first element, sum X as a second element, then sum X is the first element of the second row and N is the second column or the value at the last value of this 2 by 2 matrix, on the right hand side I will get sum2, so that is X Y and the sum Y that in Y. So, that is the right hand side vector we get.

(Refer Slide Time: 23:31)



```
1 % Least Square Line
2 function [A,B]=Lsline(X,Y)
3 %Input:- (X,Y) co-ordinates
4 % Output:- coefficients of the line Ax+B
5 N=length(X);
6 sumx=sum(X);
7 sumy=sum(Y);
8 sum1=sum(X.*X);
9 sum2=sum(X.*Y);
10 % Now we need to write the system Mx2=N of equations
11 M=[sum1,sumx;sumx,N];
12 R=[sum2;sumy];
13 sol=sol(M,R);
14 A=sol(1);
15 B=sol(2);
16 endfunction
17
```

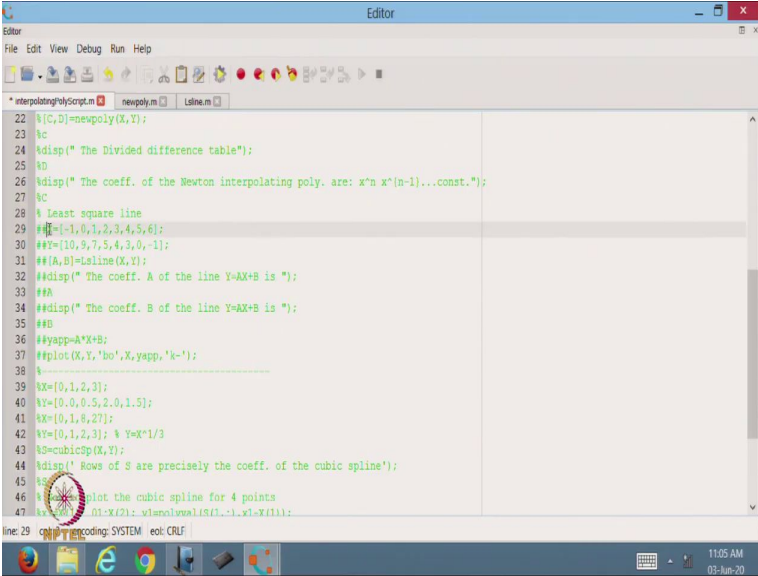


```
1 % Least Square Line
2 function [A,B]=Lsline(X,Y)
3 %Input:- (X,Y) co-ordinates
4 % Output:- coefficients of the line Ax+B
5 N=length(X);
6 sumx=sum(X);
7 sumy=sum(Y);
8 sum1=sum(X.*X);
9 sum2=sum(X.*Y);
10 % Now we need to write the system Mx2=N of equations
11 M=[sum1,sumx;sumx,N];
12 R=[sum2;sumy];
13 sol=M\R;
14 A=sol(1);
15 B=sol(2);
16 endfunction
17
```

Now, this is the solution I am going to get solving this system with a backslash method, so that is the efficient way to solve the system equation, so this is $M \backslash R$, so $M \backslash R$ means it is equal to the inverse M multiply by R , so that we are doing this one, so once I get the solution, I will get solution that will be a vector 2 by 1 vector, so this will be a column vector with the two elements, so I call it first element as a and the second of element as B , this is what we wanted to find out and then I will output this, pass this value to the place where we are going to call this function. So, that is saved as Ls line. Now, let us try to find out the least square

approximation.

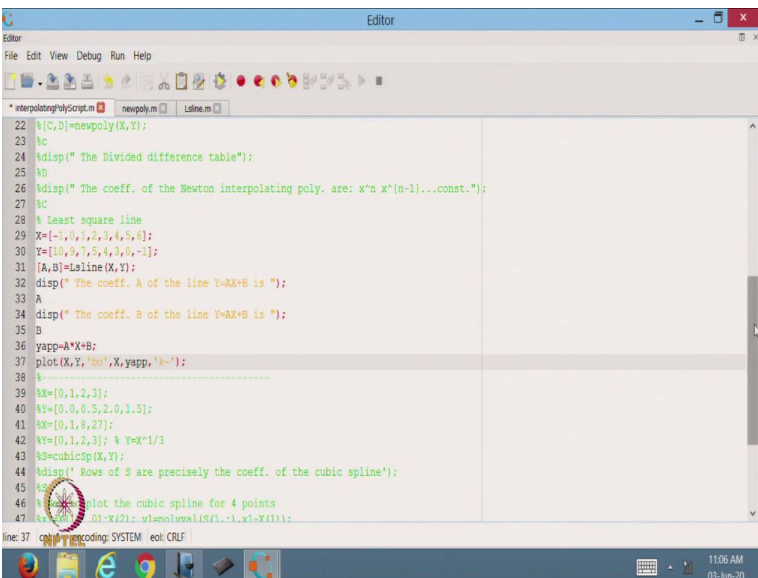
(Refer Slide Time: 24:19)



```
22 %C,D=newpoly(X,Y);
23 %C
24 disp('The Divided difference table');
25 %D
26 disp('The coeff. of the Newton interpolating poly. are: x^n x^{n-1}...const. ');
27 %C
28 % Least square line
29 X=[-1,0,1,2,3,4,5,6];
30 Y=[10,9,7,5,4,3,0,-1];
31 [A,B]=Lsline(X,Y);
32 disp('The coeff. A of the line Y=AX+B is ');
33 %A
34 disp('The coeff. B of the line Y=AX+B is ');
35 %B
36 yapp=A*X+B;
37 plot(X,Y,'bo',X,yapp,'k-');
38 %
39 X=[0,1,2,3];
40 Y=[0,0,0.5,2,0,1.5];
41 X=[0,1,8,27];
42 Y=[0,1,2,3]; % Y=X^1/3
43 %S=cubicSp(X,Y);
44 disp('Rows of S are precisely the coeff. of the cubic spline');
45 %S
46 %X=plot the cubic spline for 4 points
47 %X=X(1:4); Y=Y(1:4); %X=X(1:4); Y=Y(1:4); %X=X(1:4); Y=Y(1:4);
```

So, I will just write and this is a least square approximation we are going to use. So this is a least square approximation.

(Refer Slide Time: 25:04)

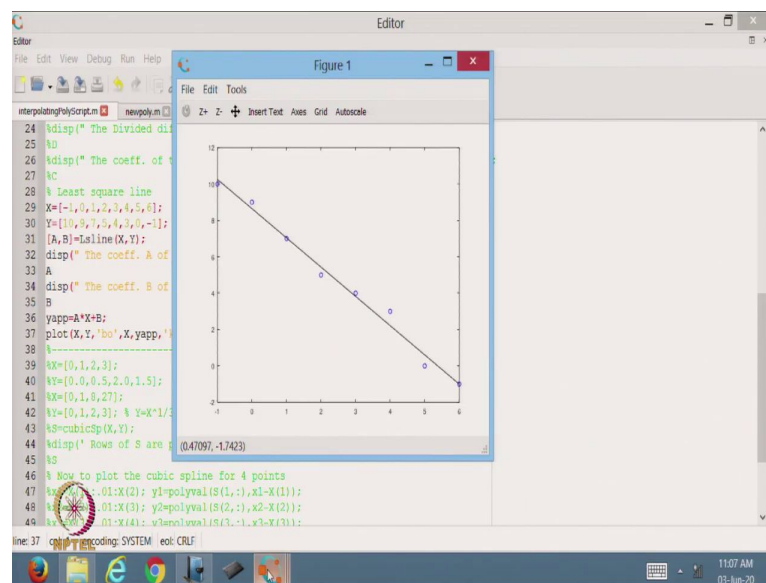


```
22 %C,D=newpoly(X,Y);
23 %C
24 disp('The Divided difference table');
25 %D
26 disp('The coeff. of the Newton interpolating poly. are: x^n x^{n-1}...const. ');
27 %C
28 % Least square line
29 X=[-1,0,1,2,3,4,5,6];
30 Y=[10,9,7,5,4,3,0,-1];
31 [A,B]=Lsline(X,Y);
32 disp('The coeff. A of the line Y=AX+B is ');
33 %A
34 disp('The coeff. B of the line Y=AX+B is ');
35 %B
36 yapp=A*X+B;
37 plot(X,Y,'bo',X,yapp,'k-');
38 %
39 X=[0,1,2,3];
40 Y=[0,0,0.5,2,0,1.5];
41 X=[0,1,8,27];
42 Y=[0,1,2,3]; % Y=X^1/3
43 %S=cubicSp(X,Y);
44 disp('Rows of S are precisely the coeff. of the cubic spline');
45 %S
46 %X=plot the cubic spline for 4 points
47 %X=X(1:4); Y=Y(1:4); %X=X(1:4); Y=Y(1:4); %X=X(1:4); Y=Y(1:4);
```

So, suppose I am taking this X, so this is the X coordinate we are going to take and this is the Y

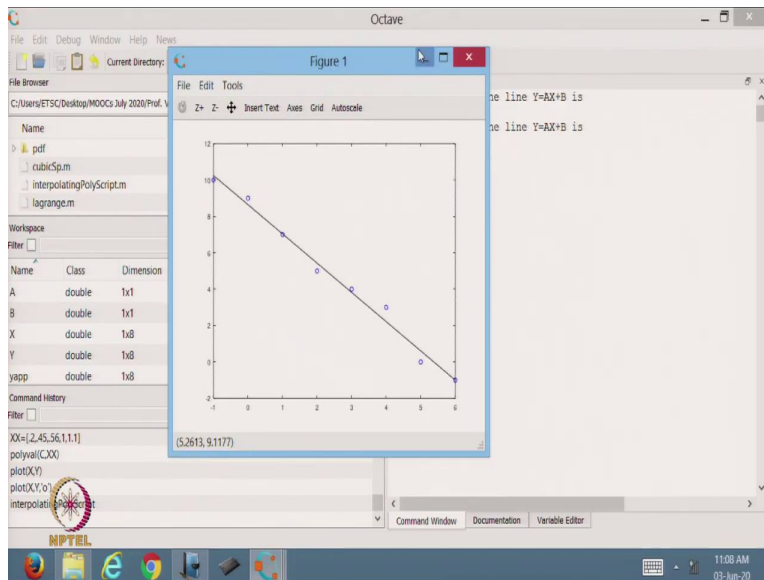
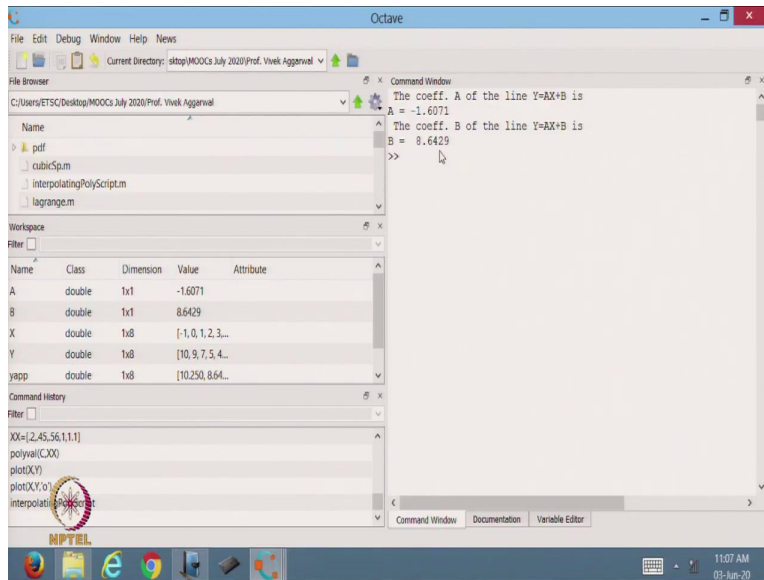
coordinates we are going to take. Now, I call the function `Ls` line from here, so this is the value of A and B we are going to get this is my I want to display what is the value of A and what is the value of B . Now, what I want to find from here, I will get the y ap that is the equation of the line and then I try to plot this line, so this is what we are going to do and let us try to run this one.

(Refer Slide Time: 25:47)



So, that is the graph we are going to get. Now, from here we can see that this dot is the my data points that we have started with and this is the line that is we are getting using the least square approximation method. So, from here you can see that this line is not passing through any of the points except this point, so only one point and these two points, so it passes through two points otherwise it is, so that gives you the minimum error that is going close to all these points. So, this is my least square line and I want to see.

(Refer Slide Time: 26:36)



So, from here I get the value of $A = -1.6071$ and $B = 8.6429$. So, that is the coefficient we are going to get and based on this one. We have plotted this figure also and that figure gives you the line that is using the least square approximation. So, based on this one, we can see very clearly that this line is a least square fit and it is not passing through all these points except only a few points. So, that is the way we can define the least square approximation. Maybe I can define another approximation also.

(Refer Slide Time: 27:23)

```

Editor
File Edit View Debug Run Help

* interpolatingPolyScript.m newpoly.m Lsline.m

24 disp(" The Divided difference table");
25 %
26 disp(" The coeff. of the Newton interpolating poly. are: x^n x^(n-1)...const.");
27 %
28 % Least square line
29 %X=[-1,0,1,2,3,4,5,6];
30 %Y=[10,9,7,5,4,3,0,-1];
31 [A,B]=lsline(X,Y);
32 disp(" The coeff. A of the line Y=AX+B is ");
33 A
34 disp(" The coeff. B of the line Y=AX+B is ");
35 B
36 yapp=A*X+B;
37 plot(X,Y,'bo',X,yapp,'k-');
38 %
39 %X=[0,1,2,3];
40 %Y=[0.0,0.5,2.0,1.5];
41 %X=[0,1,0,27];
42 %Y=[0,1,2,3]; % Y=X^1/3
43 %S=cubicsp(X,Y);
44 %disp(' Rows of S are precisely the coeff. of the cubic spline');
45 %
46 % How to plot the cubic spline for 4 points
47 %X=[0.01:X(2); y1=polyval(S(1,:),x1=X(1));
48 %X(3); y2=polyval(S(2,:),x2=X(2));
49 %X(4); y3=polyval(S(3,:),x3=X(3));

```

```

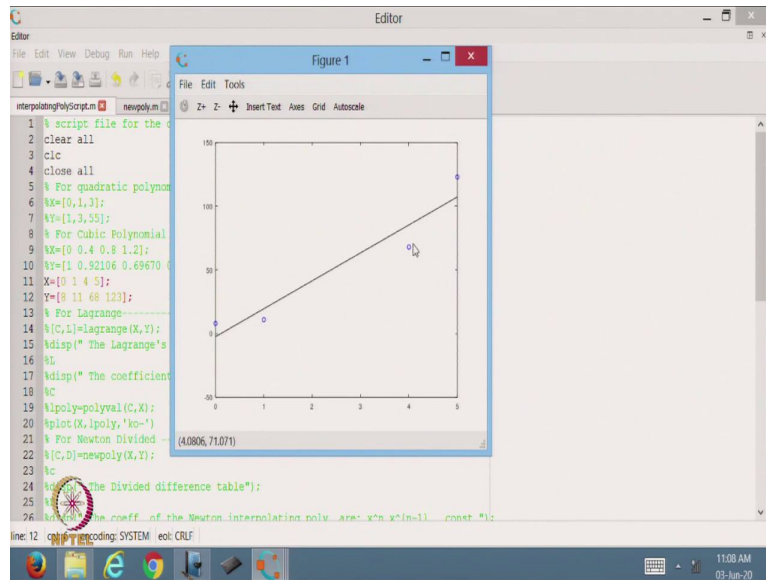
Editor
File Edit View Debug Run Help

* interpolatingPolyScript.m newpoly.m Lsline.m

1 % script file for the data
2 clear all
3 clc
4 close all
5 % For quadratic polynomial
6 %X=[0,1,3];
7 %Y=[1,3,55];
8 % For Cubic Polynomial
9 %X=[0 0.4 0.8 1.2];
10 %Y=[1 0.92106 0.69670 0.36235];
11 X=[0 : 4 : 5];
12 Y=[8 11 68 123];
13 % For Lagrange
14 %C,L=lagrange(X,Y);
15 disp(" The Lagrange's fundamental coefficients are: x^n x^(n-1)...const.");
16 %
17 disp(" The coefficients of the Lagrange inter. poly. are");
18 %
19 %lpoly=polyval(C,X);
20 %plot(X,lpoly,'ko-');
21 % For Newton Divided
22 %C,D=newpoly(X,Y);
23 %
24 disp(" The Divided difference table");
25 %
26 disp(" The coeff. of the Newton interpolation poly. are: x^n x^(n-1)...const.");

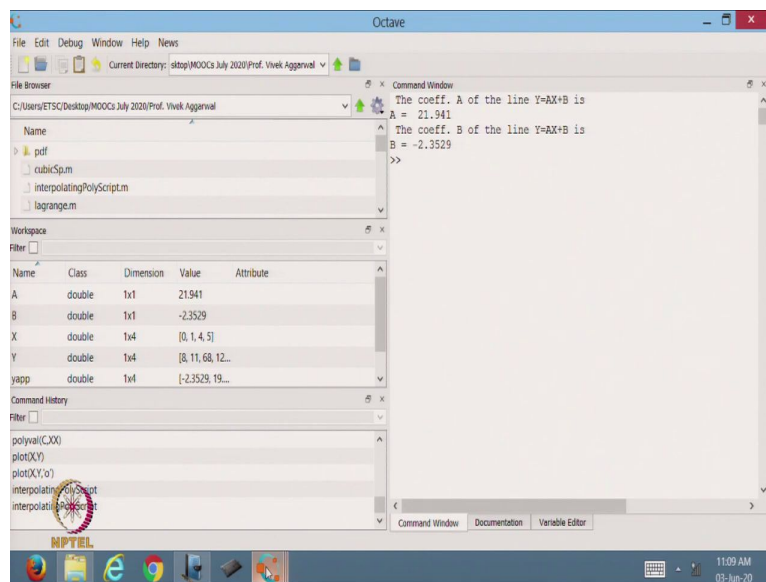
```

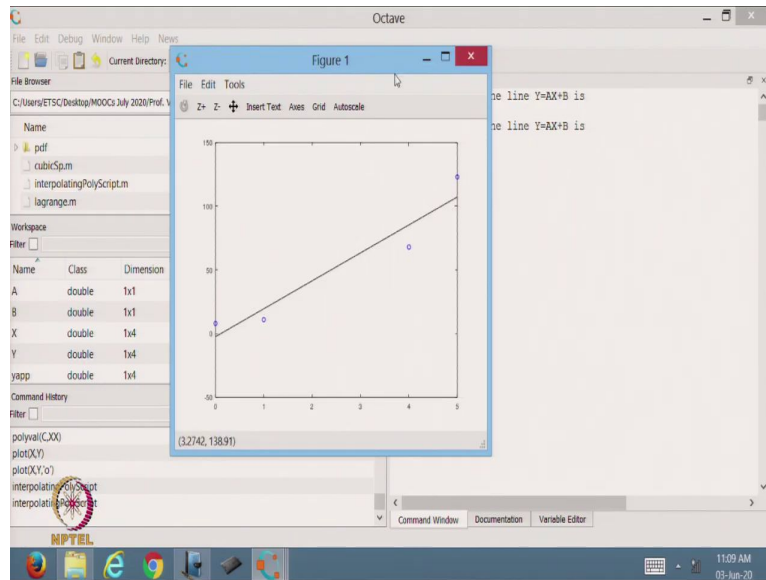
Instead of this X I try to make another line. Let us take this X and this Y.
(Refer Slide Time: 27:48)



So, that is another line. So, in this case I have only four points. At 0 it is 8 11 and then 68 and 123. So, this is the X coordinate and this is the Y coordinates. So, based on this one if I run the code.

(Refer Slide Time: 28:08)





Then I get this value of A and B, so A the coefficient is coming 21, so A you can see that that is the slope of the line, because $Y=AX+B$, so $A = 21$ so slope is 21 and this is the value of B. And from here the equation of line you can see that the slope is 21 and that is passing through, so in this case this line, least square line is not passing through the any of the point, so that is the least square approximation. So, this is the today we will stop here. So, today we have discussed the MATLAB code for Newton divided difference formula and least square approximation. So, we will continue with this one in the next lecture. So, thanks for watching, thanks very much.