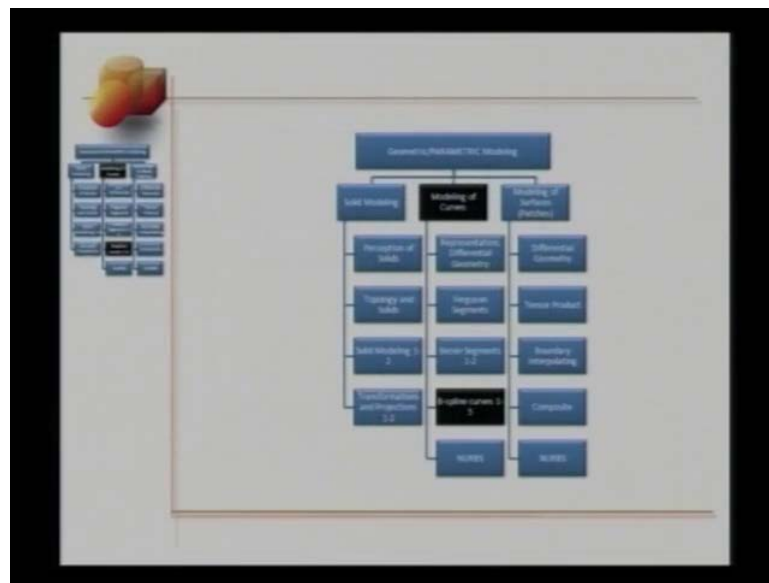


Computer Aided Engineering Design
Prof. Anupam Saxena
Department of Mechanical Engineering
Indian Institute of Technology, Kanpur

Lecture - 31
Implementation and Coding

Hello and good morning we continue with B-spline segments and curves.

(Refer Slide Time: 00:21)



(Refer Slide Time: 01:05)

Normalized B-splines

$N_{i,j}(t) = \delta_j$ such that $\delta_j = 1$ for $t \in [t_{i-1}, t_i)$
 $= 0$, elsewhere

$$N_{k,j}(t) = \frac{t - t_{i-k}}{t_{i-1} - t_{i-k}} [N_{k-1,j-1}(t)] + \frac{t_i - t}{t_i - t_{i-k+1}} N_{k-1,j}(t)$$

This is lecture number 31, over the last 10 or, so lectures we have learnt a great deal about B-spline segments and curves. How to design B-spline basis functions and how to use them to further design B-spline curves. Today we will be working on coding to understand how B-spline basis functions segment and curves an essentially related concepts can be implement.

Let us revisit normalized basis splines, a normalized basis spline of order one is given by $N_{1,i}$, where t_i is a last knot over, which $N_{1,i}$ would stand. This is a function of parameter value t and this is equal to $\delta_{i,j}$ such that $\delta_{i,j}$ is equal to 1 for parameter value t belonging to the interval $t_{i-1} \leq t < t_i$, the left hand side is close the right hand side is open.

Otherwise $N_{1,i}$ of t is 0 elsewhere, in general the recursion formulae is given by $N_{k,i}$ of t equals $t_{i-k} \leq t < t_{i-k+1}$ over $t_{i-k} \leq t < t_{i-k+1}$ times $N_{k-1,i-k}$ of t plus $t_{i-k+1} \leq t < t_{i-k+2}$ over $t_{i-k+1} \leq t < t_{i-k+2}$ times $N_{k-1,i-k+1}$ of t . The first goal for us would be to write a function, to determine any B-spline basis function of order k or which the last knot is t_{i-k} . You can use any mode to perform your coding, I am comfortable with MATLAB, so I am going to be demonstrating the implementation using that software.

(Refer Slide Time: 02:49)

```

1
2 function [val] = nb_spline_basis(m,i,T,t)
3
4 % m: order of the B-spline basis function
5 % i: INDEX of the last knot over which nb_spline_basis(m
6 % T: the KNOT vector
7 % t: parameter value
8
9 % case 1: m = 1
10 if m == 1
11     if { t >= T(i-1) } & { t < T(i) }
12         val = 1;
13     else
14         val = 0;
15     end
16
17 % case 2: m > 1
18 else

```

So, I am going to be writing function to determine B-spline basis function, function some value is equal to let me call this function n b underscore spline basis and will be passing

some parameters into this function. Let me pass a few parameters say m i capital T and small t , m would be the order of the B-spline basis function, i is the index. I am writing this thing in capital letters, index of the last knot over which, this spline function stands I am just going to copy and paste, capital T is the knot vector and small t is the parameter value at which this spline function is to be computed.

Let us start with a coding we have a few cases, case 1 is when m is equal to 1 and then case 2 is when m is greater than 1. Let us write the code for these 2 cases if m is equal to 1 and something else this would be the case, where m is greater than 1 and then will have some arguments will finally, end this escape. Here I am going to be next thing another escape, if the parameter value t is smaller than rather is greater than or equal to T i minus 1.

And if t is smaller than T i , then value should be equal to 1, else value should be equal to 0 end, this is where we compute order one basis spline functions. So, clearly the parameter value t has to lie in between the knot span capital T i minus 1 and capital T i . Next consider the case where m is greater than 1, we expressively imposes case if m is greater than 1 be end this it is statement, we going to be using some arguments in between here.

(Refer Slide Time: 06:22)

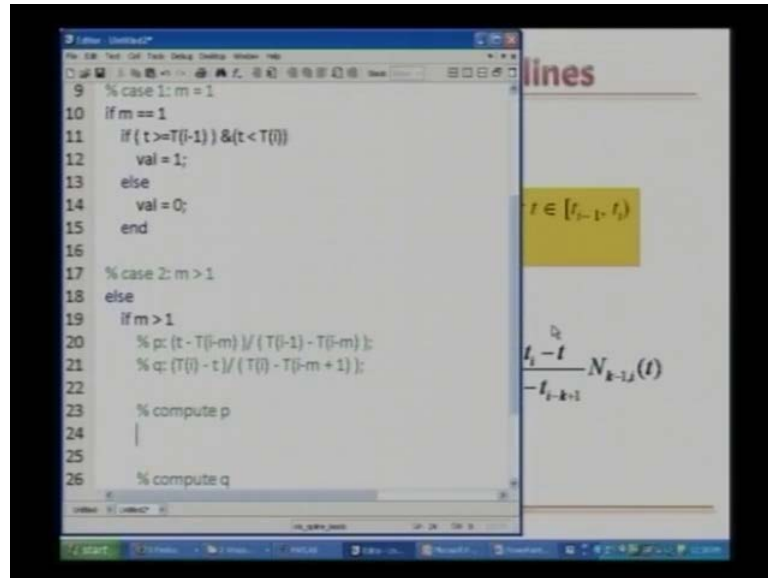
Normalized B-splines

$N_{i,j}(t) = \delta_{ij}$ such that $\delta_{ij} = 1$ for $t \in [t_{i-1}, t_i)$
 $= 0$, elsewhere

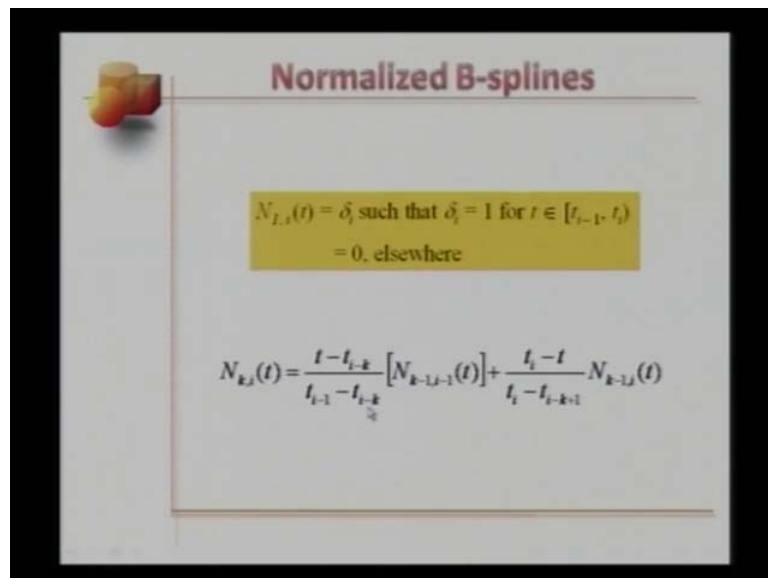
$$N_{k,j}(t) = \frac{t - t_{i-k}}{t_{i-1} - t_{i-k}} [N_{k-1,j-1}(t)] + \frac{t_i - t}{t_i - t_{i-k+1}} N_{k-1,j}(t)$$

So, essentially here I am going to be computing $N_{k,i}$ which has 4 components, this coefficient the basis function of one order less and in this coefficient and again the basis function of one order less with the last knot as at.

(Refer Slide Time: 06:56)



(Refer Slide Time: 08:27)

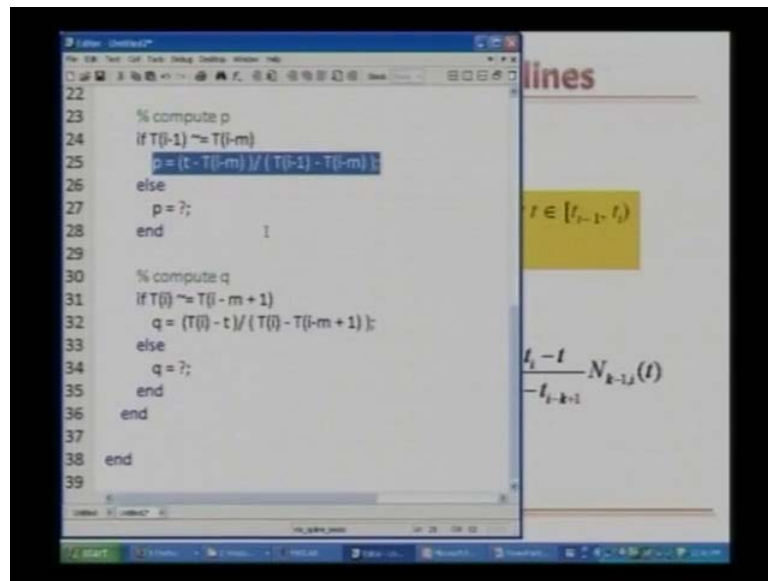


Let me call these 2 coefficient small p and small q, so I am going say p is equal to I am just commenting as it now, small t minus capital T i minus m over T i minus 1 minus capital T i minus m. Likewise, another coefficient q is equal to I just copy and paste this, here I will have t i minus t i want need this and in the denominator I have T i minus T i

minus T_{m+1} . So, once we have the expressions for the individual weights which are linear p and q let me code then, so we first compute p and then we compute q .

Let us go back to this expression, this is the coefficient p that we uninterested in computing, this is q that we want to compute. Now, notice that in case we are working with multiple knots t_{i-1} can be equal to t_i , likewise t_i can be equal to t_{i+1} will have to take care of these conditions expressively.

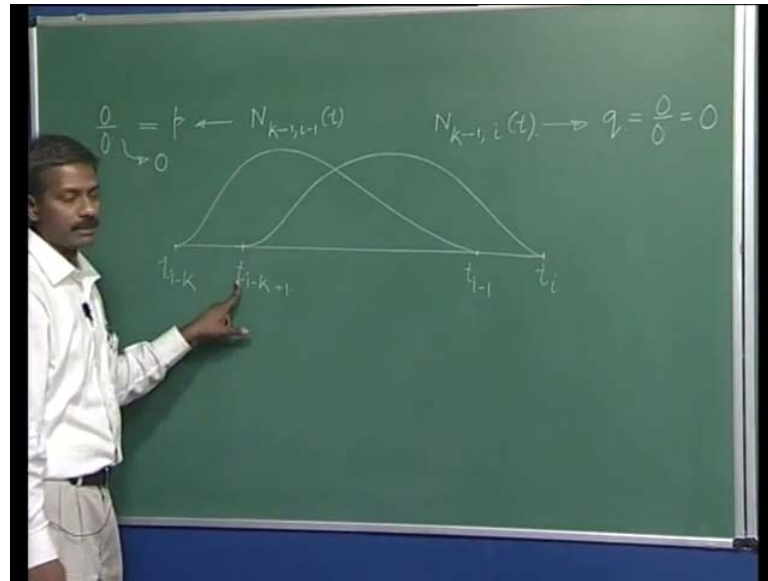
(Refer Slide Time: 08:59)



Let us work with those conditions if T_{i-1} is not equal to T_{i-m} , then I should be able to compute p as this expression here else and then end. What should p be in this case, we will come to that a little later. Likewise, I just copy this and pasted here, I will copy the expression for q and pasted here. This is the equal to sign and this is q , here I need to ensure that T_i is not equal to T_{i-m+1} .

So, in case we are not translating multiple knots here, q will have this expression capital T_{i-m+1} over capital $T_i - T_{i-m+1}$. In case these conditions are satisfied we have the expressions for p and q like, so in case of multiple knots what would the value p and q be.

(Refer Slide Time: 10:54)



Let us what thing out on board, so we have knots t_{i-k} t_{i-k+1} t_{i-1} and t_i , in the recursion relation we have 2 basis splines each about the k minus. One of which is standing with the last knot t_{i-1} and the other one is standing with last knot as t_i with $N_{k-1, i-1}$ of t coefficient p is associated and with $N_{k-1, i}$ of t coefficient q is appearer.

Now, the condition that t_{i-k} will be equal to t_{i-1} will be only physically possible, if all the intermediate knots including t_{i-k} and t_{i-1} are multiple knots. So, will have 2 cases here, case 1 that t is equal to this knot value which will be equal or t is greater than t_{i-1} or smaller than t_{i-k} , in case of multiple knots when t_{i-k} t_{i-k+1} and so on up till t_{i-1} or all equal.

And in case t is equal to this knot value if you look I will be expression for p , p will be taking, so $0/0$ form by convention we assume that this is 0. For all the other values of t $N_{k-1, i-1}$ of t will be 0. Likewise if we have these multiple knots where t_{i-k+1} t_{i-k+2} t_{i-k+3} or all equal to t_i . Again will have 2 cases, case 1 when t is equal to all these knot values in which case again the expression for q will assume the $0/0$ form which by convention, we treat as 0. Otherwise, for t not equal to these multiple knots $N_{k-1, i}$ will be 0 we keep this in mind and use p equals 0 here and q equals 0 here.

(Refer Slide Time: 13:55)

```
22
23 % compute p
24 if T(i-1) ~= T(i-m)
25     p = (t - T(i-m)) / (T(i-1) - T(i-m));
26 else
27     p = 0;
28 end
29
30 % compute q
31 if T(i) ~= T(i-m+1)
32     q = (T(i) - t) / (T(i) - T(i-m+1));
33 else
34     q = 0;
35 end
36 end
37
38 end
39
```

$t \in [t_{i-1}, t_i)$

$$\frac{t - t_{i-k+1}}{t_i - t_{i-k+1}} N_{k-1,j}(t)$$

(Refer Slide Time: 14:06)

```
1
2 function [val] = nb_spline_basis(m,i,T,t)
3
4 % m: order of the B-spline basis function
5 % i: INDEX of the last knot over which nb_spline_basis(m,i,T,t)
6 % T: the KNOT vector
7 % t: parameter value
8
9 % case 1: m = 1
10 if m == 1
11     if { t >= T(i-1) } & { t < T(i) }
12         val = 1;
13     else
14         val = 0;
15     end
16
17 % case 2: m > 1
18 else
```

$t \in [t_{i-1}, t_i)$

$$\frac{t}{t - t_{i-k+1}} N_{k-1,j}(t)$$

If you look at this expression here, we have computed p and q, we are yet to incorporate this recursion relation, so we will go back here and we will say that the recursion relation is to be incorporated here. And we will set the value v a 1 equals p times, now here is a trick I can use the concept of function within a function. In a sense I can use this same function to compute B-splines of order one less 2 less and so on. Once we understand this I will say n b underscore spline underscore basis, here we have order m minus 1.

Here the last knot over which this spline is standing is T_{i-1} the knot vector T and the parameter value plus few times n b underscore spline underscore basis. Once again order one less last knot here is t_i the knot vector T and the parameter value this should be adequate for us to compute a B-spline basis function of order m with the last knot as T_{i-1} . I will save this file as in B-spline basis functions it is time now to test the part, for that I am going to be using a new function.

(Refer Slide Time: 16:41)

```

1
2 clc, figure(1), cla
3 clear all
4
5 % order: m
6 m = 4;
7
8 % T: KNOT VECTOR
9 T = [0 1 2 3 4 5 6 7 8 9 10];
10
11 % plotting nb_spline_basis functions
12 for i = 1:200
13     t = (i-1)/199;
14
15     x(i) = (1-t)*T(0) + t*T(11);
16
17
18 % BSBF: B Spline basis functions

```

I am going to be using some commands which are typical format line, clear screen figure one clear axes, clear all variables, less test the previous function and B-spline basis for order 4 splines. And let us test for a very general knot vector with 10 knots spline or 11 knots, let me comment here order is given by m , T is the knot vector which is uniform in our case.

Will talk about knot vector generation in a little while and let us start of you B-spline basis functions, for index i 1 to let say 200, let us generate a parameter of t this is equal to index i minus 1 over 199. I am using t equals i minus 1 over 199, so that I can generate value of t between 0 and 1, if you see for i equals 1 t 0 and for i equals 200 t is 1.

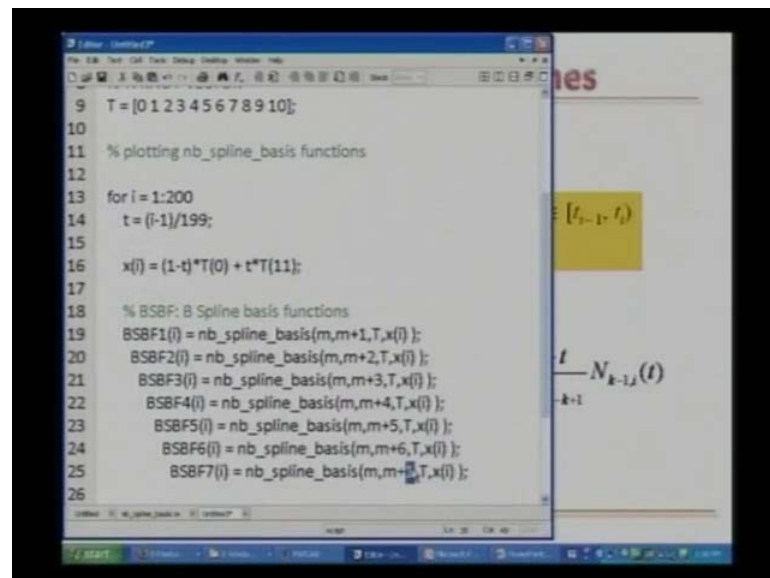
Let me first end this follow and continue further, I will this time let me not worry about the full support and let me allow my parameter values to go from 0 to 10. Let me store this parameter values in a vector x of i , this is $1 - t$ which is this variable here times $T(0)$ which is this value here plus t times $T(11)$ which is this value here. And now it is time

for us to compute B-spline basis functions, let me you write them chart B S B F standing from B-spline basis functions.

How many of these B-spline basis functions will be pass, you know that already from previous lecture if would 7 of them, so the first one will be given by n b spline basis the order is m will have last knot mention a little while the knot vector is capital T and the parameter value is small t. Let me be a little careful here, the first of B-spline basis functions will be standing over 4 knot span, this is the first knot spline, the second knot span, the third knot span and the fourth knot span.

This would be the last knot over which the first B-spline basis function will be standing, the index for that knot corresponds to 5. Since, m is equal to 4, I can write the index here as m plus 1. Is it for me to use small t for the parameter value here, if you realize I am storing the parameter value in this array x of i. So, I can replace this small t i x of i.

(Refer Slide Time: 21:39)



```

9 T = [0 1 2 3 4 5 6 7 8 9 10];
10
11 % plotting nb_spline_basis functions
12
13 for i = 1:200
14     t = (i-1)/199;
15
16     x(i) = (1-t)*T(0) + t*T(11);
17
18 % BSBF: B Spline basis functions
19 BSBF1(i) = nb_spline_basis(m,m+1,T,x(i) );
20 BSBF2(i) = nb_spline_basis(m,m+2,T,x(i) );
21 BSBF3(i) = nb_spline_basis(m,m+3,T,x(i) );
22 BSBF4(i) = nb_spline_basis(m,m+4,T,x(i) );
23 BSBF5(i) = nb_spline_basis(m,m+5,T,x(i) );
24 BSBF6(i) = nb_spline_basis(m,m+6,T,x(i) );
25 BSBF7(i) = nb_spline_basis(m,m+7,T,x(i) );
26

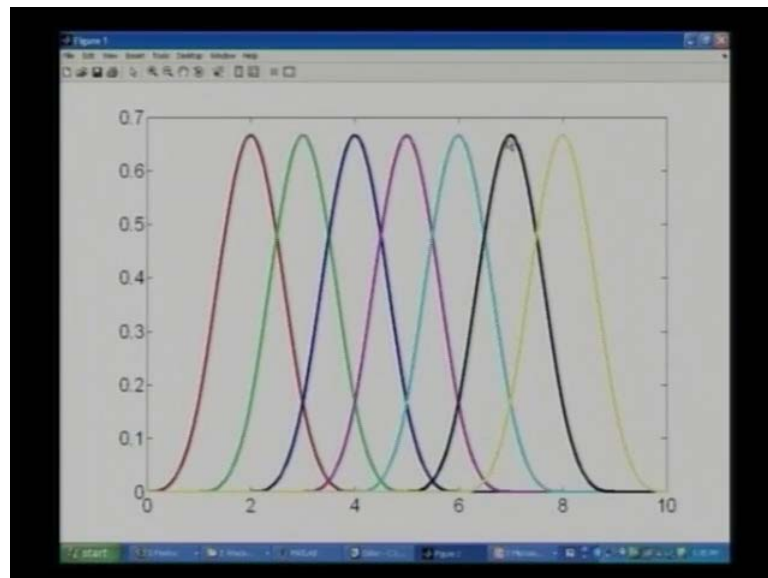
```

All I need to do is now copy and paste this statement a few times, since we have 7 B-spline basis functions I rename the variables on the left hand side accordingly. The order would remain the same, the knot vector T would remain the same the parameter values would remain the same this that the last knot over which the respective B-spline basis functions are standing will change. This would be m plus 2 m plus 3 m plus 4 m plus 5 m plus 6 m plus 7, I have already ended the followed.

Let me now select all text and make the code look a little beautiful, let us now plot with the B-spline basis functions. MATLAB has the command plot for that plot x B S B F 1 let me plot first one using the red color r, and let me specify the line width as 3, I can use the same command plot all the other functions. This is second B-spline function, third, fourth, fifth, sixth and seventh, let me choose different colors to plotted. This one let us say green, this one let us say blue, this one is magenta, sign, black and yellow, let me also specify the line style which is solid in all cases.

We are now ready to see the spots of these 7 B-spline basis functions, I will save the file as n b spline basis test, let me execute this file and see how the spots may reply. And whether I may have committed any mistake in coding, looks like I have apparently I have made a mistake in the coding here, MATLAB does not allow the in decease to be 0 or negative value. Maybe I should have chosen T 1 in case of T 0, let me go back and execute the code again. Now, I see only one of these B-spline basis functions, what I have done is I am not used the option hold on then uses let me save this file again and execute the code.

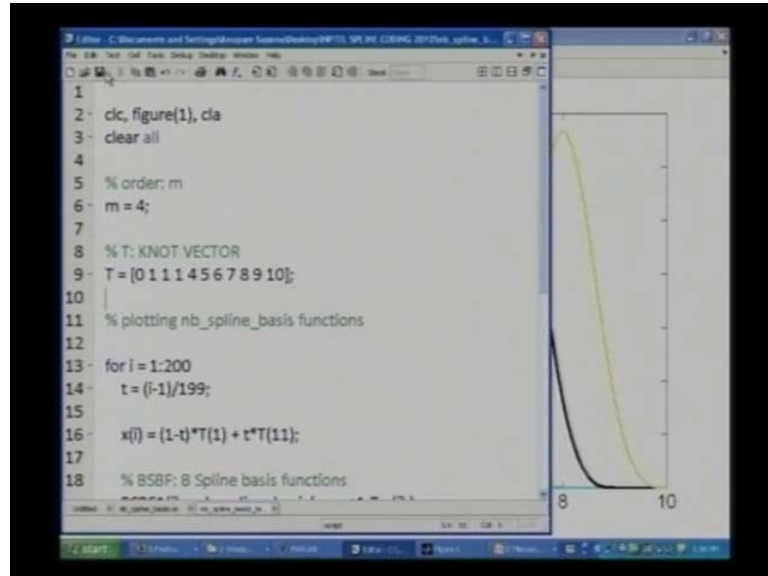
(Refer Slide Time: 26:35)



Now, you see all the B-spline basis functions, let me change the figure properties a little bit, so the background as white and set the font size as let us say 20. Now, this is a lot better can you name this B-spline basis functions as per a convention is could be n 4 4

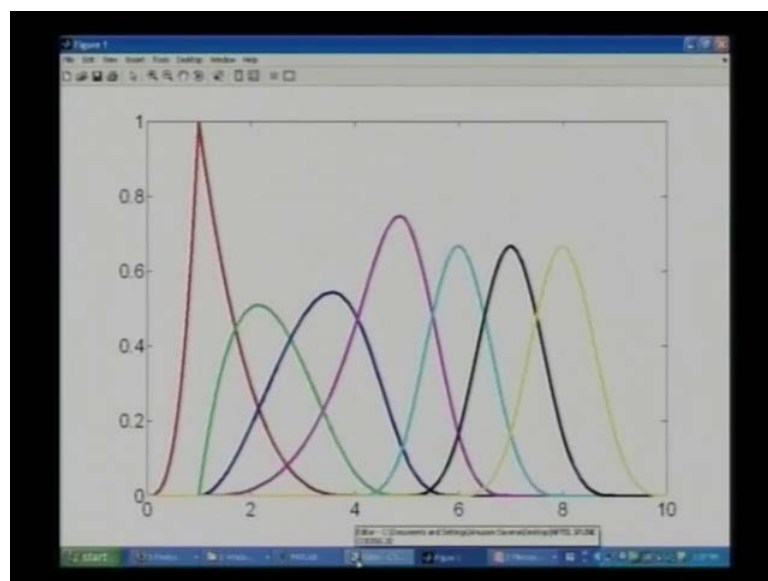
this n 4 5 n 4 6 n 4 7 n 4 8 n 4 9 and n 4 10. So, far we have been coding the B-spline basis functions and you have just tested them let us play a little more.

(Refer Slide Time: 27:43)



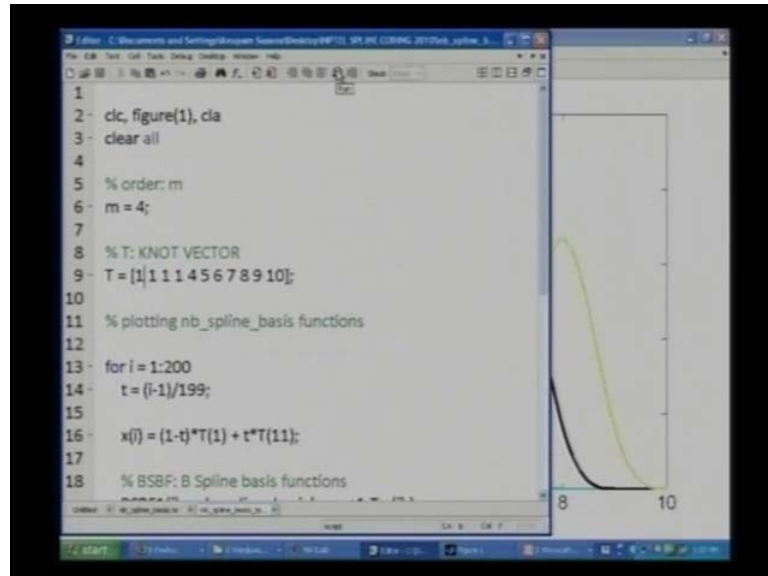
Let us go back to our editor and start modifying these knots, in the sense let us introduce multiple knots and see what happens. Let me increase the multiple (()) of the second knot by 1 or may be by 2, so that the second knot is now having multiplicity 3 what changes do you expect.

(Refer Slide Time: 28:22)



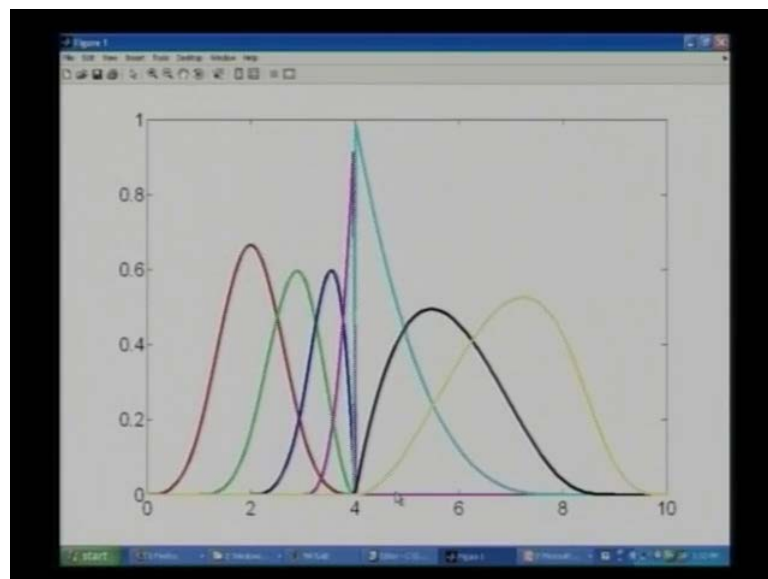
The B-spline basis functions will change in shape lowering, now to realize at this knot value 1, the first B-spline basis function as the value 1 and all the others have the value 0, this concourse with the local (()) property.

(Refer Slide Time: 29:06)



Let us go back to our editor and even make the first knot as 1 to be see any change let us find out.

(Refer Slide Time: 29:16)

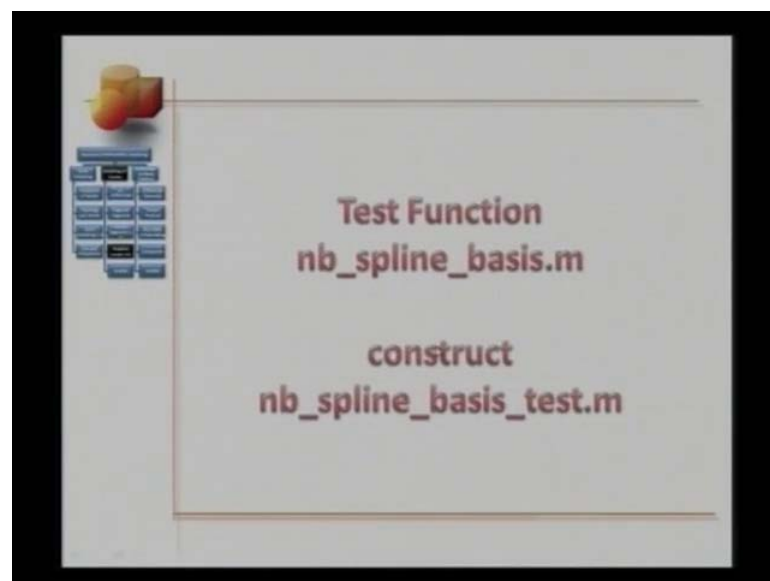


We do not see much change expect for the fact at this part of the first B-spline basis function, it is now has how about the if I increase the multiplicity of this knot by 2.

Making $T = 8$ here, and knot of multiplicity 3 by change this values to 7 and this value as well to 7 I execute this for again. Now, we have the last basis function with value 1 are the parameter value 7, all the other B-spline basis functions at this knot value are 0. We go ahead and change the last knot value to 7, to see what happens not much change just that the last knot value is not 7.

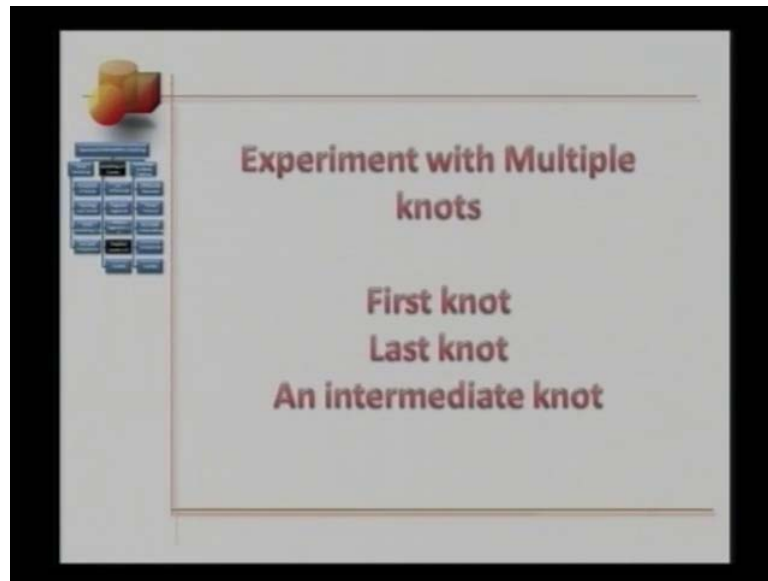
The last basis function still has the value 1 and all the other previous basis functions have the value 0 at $t = 7$. Let us go back to a editor modify this knots towards there why before and increase the multiplicity of any intermediate knot, let us say I make these 3 knots have the value 4 to expect any change. Yes, B-spline basis functions which are associated with this knot value change local in shape, you can experiments in whatever way you feel like to be comfortable with concept.

(Refer Slide Time: 32:19)

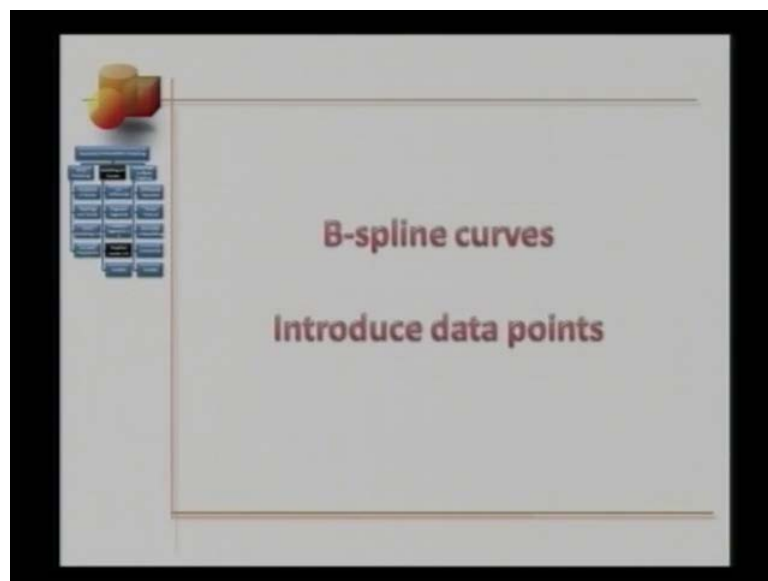


So, we have just about done testing the normalized basis spline function for which we constructed a test function here. We also experimented with multiple knots, we had first knot as a multiple knot of multiplicity 4 which was the order of the B-spline curve or the basis function, likewise we also tested for the last knot having multiplicity 4 and for any other intermediate knot we tested for it is multiplicity.

(Refer Slide Time: 32:30)

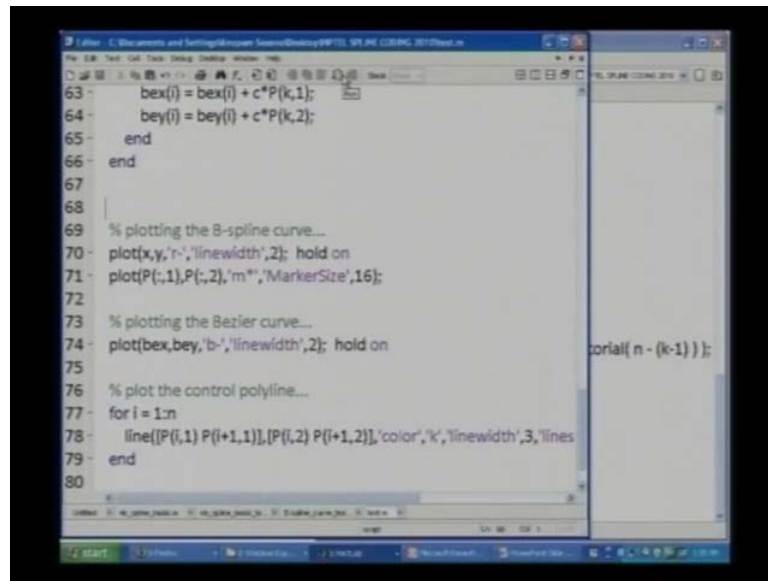


(Refer Slide Time: 33:05)



We are now ready to design B-spline curves for that you would need to introduce data points. Let us start with a new function some typical MATLAB commands clear all variables, clear string, post figure and here let me common that we are designing B-spline curves. Let me work with two dimensional data points for now, I am storing the x and y coordinates N and R A P. Let me arbitrarily specify these data points I have arbitrarily specify x and y coordinates in R A P. Let me copy this with the work space of MATLAB, so this is how my R A P looks like.

(Refer Slide Time: 33:20)



```
63-     bex(i) = bex(i) + c*P(k,1);
64-     bey(i) = bey(i) + c*P(k,2);
65- end
66- end
67-
68-
69- % plotting the B-spline curve...
70- plot(x,y,'r','linewidth',2); hold on
71- plot(P(:,1),P(:,2),'m','MarkerSize',16);
72-
73- % plotting the Bezier curve...
74- plot(bex,bey,'b','linewidth',2); hold on
75-
76- % plot the control polyline...
77- for i = 1:n
78-     line([P(i,1) P(i+1,1)], [P(i,2) P(i+1,2)], 'color','k','linewidth',3,'lines
79- end
80-
81- factorial(n - (k-1));
```

Let me compute the number of design points, these are $n + 1$, again n is equal to the size of p minus 1 going back to the work space size of p indicates the number of rows and the number of columns. Here we have 8 rows 1 2 3 4 5 6 7 8 and we have 2 columns 1 and 2, so we say that the number of design points are $n + 1$ n would be the number of rows and p minus 1.

Let me specify the order, let me denote the order by p and let me say for now that p is equal to 5 your working with order of 5 B-spline curves. The number of knots will be $m + 1$, where m is equal to $n + p$, we get the value of n from here and we have just specified the order p , for now let me specify arbitrarily a knot vector. For this case I would already know what m is, this is 7 plus 5 12, so the number of knots will be 12 plus 1 13 accordingly I specify T as 0 1 2 3 4 5 6 7 8 9 10 11 and 12.

Alternatively, I can also specify T using the following, for i going from 1 2 $m + 1$ T of i is equal to i minus 1. We will worry about plumping these B-spline curve, later let me just comment it (()) Let us now worry about the interval of full support and this would be given by the interval a b . Now, what is a ? And what is b ? Let me copy this piece of the code in the work space and let me see what T is for us, it is a set of integer values from 0 to 10. Let me double check what m is that look likes we have in working with order 4 B-spline curves.

So, the work space contains the values or the values of previous variables, let me go back to the editor, save it, as B spline curve text. This command here, is going to wipe out the values of all the previous pages, let me execute the curve to that error let me not worry about this error for now, I was able to reserve the error. Now, I am back to the values of the knot vector, so the transpose of the knot vector is given by this column from 0 to 12 in all 13 knots.

Now, we are working with order of 5 B-spline curves and we are looking for the knot values that form the full support for the B-spline curve. For that we need to count the first 5 knot splines, this is 1 2 3 4 and 5 and looks like this knot will be the knot that will be defining the lower bond of the full support. $T_0 T_1 T_2 T_3 T_4$, so if here should be T_p , now the index starts from 0 then this value should have been p minus 1, but since in MATLAB the index starting from 1 given though the values is 0.

This index here will correspond to p , let me uncomment this statement how about the right most value of the interval, we revisit the not vector again and start counting in knot splines from the bottom. First knot spline, second, third, fourth and fifth, looks like this value here T_9 would correspond to b what is T_9 for us. This is T_m plus 1 in number of knots minus p minus 1, let me uncomment this two, let me copy this statements to work space and verify if you are correct. So, a has the value 4 and b has the value 8, so we are in curve that is continue.

And generate a B-spline curve, lets also plot the corresponding Bezier curve to compare the two curve models, will worry about Bezier curves a little while. Now, let us say we have about 150 points, so we raise our index I from 1 to 150, let us have a parameter value T_1 between Z_1 one. For which we compute T_1 as I minus 1 over 149 clearly for i equals 1 T_1 is 0 for i equals 150 T_1 is 1. Now, let us have a parameter value t vary it mean the two values a and b , for that will have 1 minus t times a plus t times b , this should actually b times t .

Before, I mess up the code let me end disported and continue further, now let us store the x and y coordinates of the curve in 2 arrays x_i , we initialize in 0 and y_i for the y coordinates we initialize this also as 0. Now, these are the x and y coordinates of this spline curve to be mention this likewise, we store the x and y coordinates of the Bezier curve as well we store them in 2 arrays $b_e x_i$ and $b_e y_i$ we said both to 0.

I would want you at this time recall the definition of B-spline curves, that is a weighted linear combination of B-spline basic functions and the corresponding design points. Let me run a nested for loop to compute the two curve models for k equals 1 2 n plus 1 if you remember n plus 1 is a number of design points. I end this for loop here and within this for loop I compute the two curve models. For B-spline curve I have x of i equals x of i plus n B-spline basis, remember the construction let us look at the variable value, we have chosen to specify the order that is p .

We come back here we specify the order of the B-spline basis functions we specify the last knot we specify the knot vector and we specify the parameter value. The parameter value T is going from a to b , this is arrange of full support for the B-spline curve. Remember, that t is not normalize to have the values between 0 and 1 it is $T - 1$ that has the values between 0 and 1. So, we have just computed the x coordinates of the B-spline curve, but not yet I need to multiply by capital P $k - 1$, the x coordinates of all the data points.

Likewise, I compute the y coordinate for that I copy and paste go back and change this statement slightly, is a y 's the B-spline basis functions will remain the same. This start the data points here we will now have the y or the coordinate information, let me expand this a little bit. How about the Bezier curve? I would here want you to recall the construction of Bezier segments, I am going to be using a coefficient c which I will be computing as factorial something over factorial something times factorial something.

Now, this number here within the parenthesis should be the number of data points, should it be the number of data points or rather should it be the number of data points minus 1. So, we have the number of data points as $n + 1 - 1$ which would make it n , within the parenthesis here I should be actually computing the factorial $k - 1$. Why $k - 1$, because my index k here is starting from 1. While the relation that we have developed of Bezier segments had the index k starting from 0, so I need to make that little adjust here will be $n - k + 1$.

Let us see if this expression is the valid expression let us arbitrarily take the value n equal 6 k equals 3, copy the statement and get the value of c . So, it looks like a valid construction we move further and we say c equals c times $(1 - t)^{k - 1}$

times t^1 raised to $n - k - 1$. Now, this variable c will have the value from this statement, these are the other expressions pertaining to the Bernstein polynomials.

And of course, my parameter value here will be going from 0 to 1, let us continue further we say $b_{e x}$ of i is equal to $b_{e x}$ of i plus the Bernstein polynomial which will be stored in c times the x coordinate of the k th design point. Likewise, we will have a very simple expression for the y coordinate and this would have values of the coordinates on the design points.

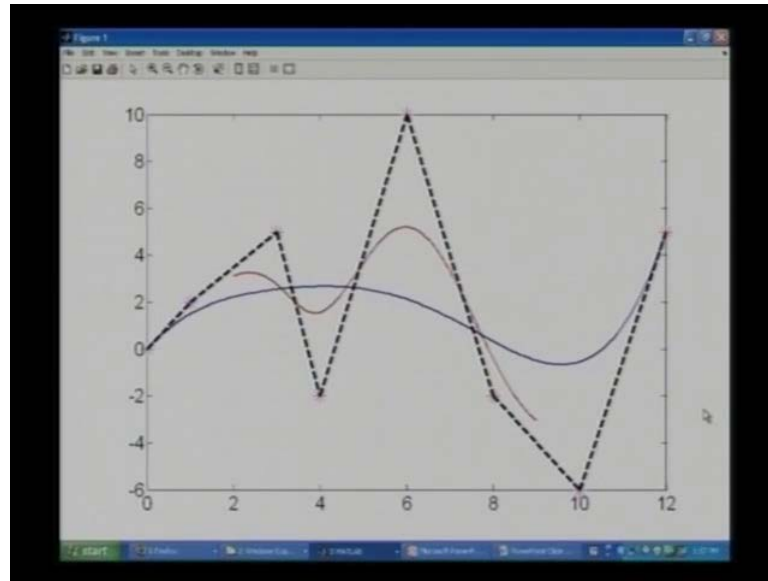
Looks like and I am just about done coding for Bezier curves and B-spline curves all I would need now is to right some statements to (()) plotting the B-spline curve, B-spline curve has informations stored in the x and y array. So, will have plot $x y$ with color red and line style solid line with as 2 we would further want to keep continuing plotting given data. So, we will use the command hold on will also plot the design points, the first column of the design points are given by P within parenthesis colon comma 1. And the coordinate values of the design points are stored in the second column for which this values 2.

Let me use magenta star took `mar` is data points and let me use a slightly larger marker sizes of let us say 16, I going to used this command hold on here. I will not be needing at any more, now let me plot the Bezier curve all I need to do is copy this statement, paste this statement here. And replace the x and y arrays by the $b_{e x}$ and $b_{e y}$ arrays. Let me use the deferent color to plot the Bezier curve, the color I am using is blue, let me also go ahead and plot the control polyline, just to help us regain the prospective.

I am going to be using the for loop for this and I am going to be using the line command, so this expression here will have information about the x co ordinance of two configurative points in a polyline $P_{i-1} P_{i+1}$. And likewise the array here will have the information pertaining to the coordinance of the design points, for change the column value here. I will use the black color, I will use line with as said 3 and the line style as may be said dashed and I have to end this for loop which I do right away.

So, looks like I am done for writing the code for plotting B-spline curves and segments, let us see in the code works I may or may not have made errors in writing code. Let us execute this file I get a result in a single shot this is interesting, let me change the figure properties here.

(Refer Slide Time: 01:00:34)



Let me increase the font size of the numbers, so the top point in magenta represent the design points, that the user has specified. The dashed line represents the control polyline, the curve in red shows B-spline curve and that in blue shows a Bezier curve. Now, this is a curve of order 5 as you would know, and a substitute figure out what the order of the degree of this Bezier curves. Notice, at the B-spline curve is started for values of t the parameter in the full support range. We are just about done writing the code to generate B-spline curves.

(Refer Slide Time: 01:02:00)

A presentation slide with a light gray background and a dark border. In the top left corner, there is a small graphic of three overlapping spheres (orange, yellow, red) and a vertical stack of blue rectangular buttons. The main text is centered and reads: **Experiment with end condition and data points**. Below this, there are two lines of text: "first end clamped, last end clamped, both ends clamped" and "move data points, add data points Comparison with Bézier Curve".

Let us experiment further with that code specifically that is experiment with the end conditions, and data points. We can think of clamping the first end of the B-spline curve clamping the last end of the curve or may be clamping both the ends. We can also think of moving data points or adding data points, since we already have written the code to generate a corresponding Bezier curve. We will be comparing or B-spline curve side by side with corresponding Bezier segment, so let us continue with the coding.

(Refer Slide Time: 01:02:50)

```

32 % clamping conditions...
33
34 % clamping the first end...
35 for i = 1:p
36     T(i) = T(p);
37 end
38
39
40 % clamping the last end
41 % for i = 1:p
42 % T(m+1 - (i-1)) = T(m+1 - (p-1));
43 % end
44
45
46 % interval of full support... [a b]
47 a = T(p);
48 b = T(m+1 - (p-1));
49

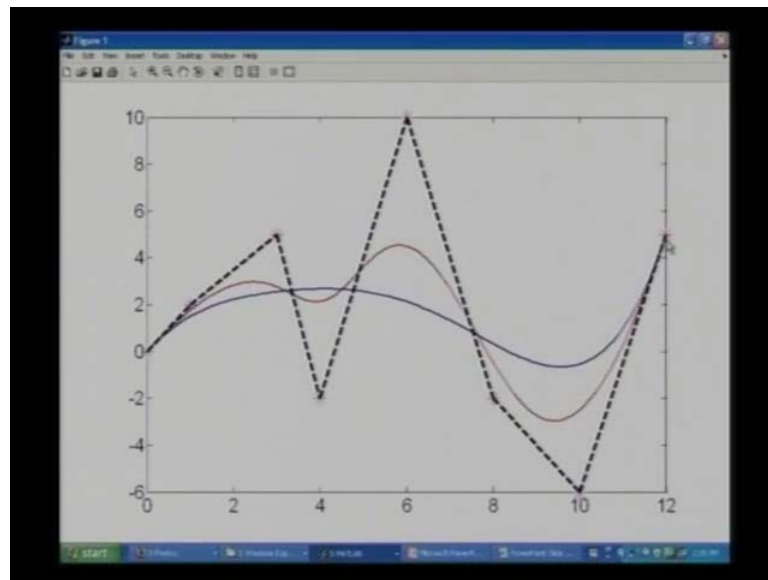
```

So, if you recall I had let some space for clamping conditions, let me use that space now so here let me first think of clamping the first end. For that I am going to be increasing the multiplicity of the first knot by p the order, so I will say for i equals 1 to p T_i is equals to T_p end. Let me also write the clamping condition, for the last point clamping the last end, here a similar piece of code can be used, but with slightly different larger. Here, we are going to be making the last knot a multiple knot of multiplicity p , how many knots to be have we have $m + 1$ of them, so will have to figure the entries in these parenthesis appropriately.

So, this entry here will be $m + 1 - p - 1$, rather this would be $i - 1$, this entry here will be $m + 1 - p - 1$. Let us go forward and see what this value is this value is the one that this variable b carries the last knot of full support, and all the other knots on the right of this will have the same value as says. Therefore, this index

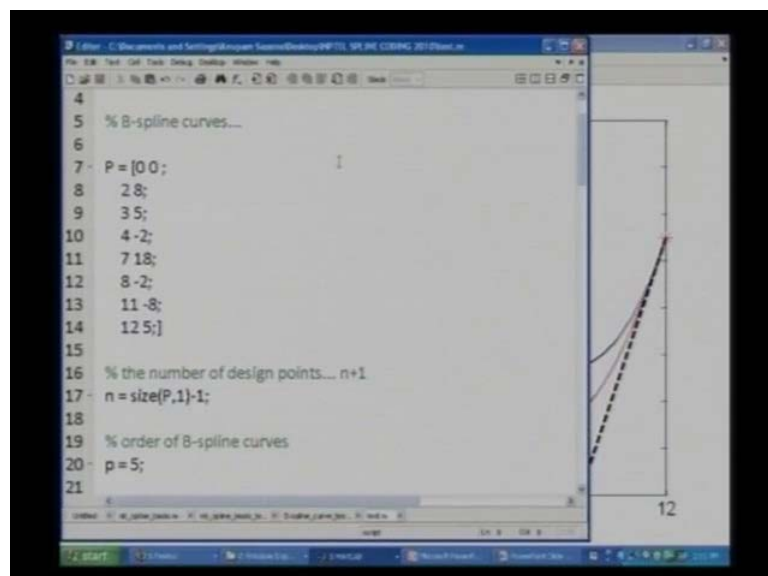
Let me go back to my editor and go back to where I am using the parameter values to generate my B-spline curves, may be it is here. What I would do is about subtract with very small value from t, this is quite small for our purpose let me execute the code now after making this modification.

(Refer Slide Time: 01:08:09)



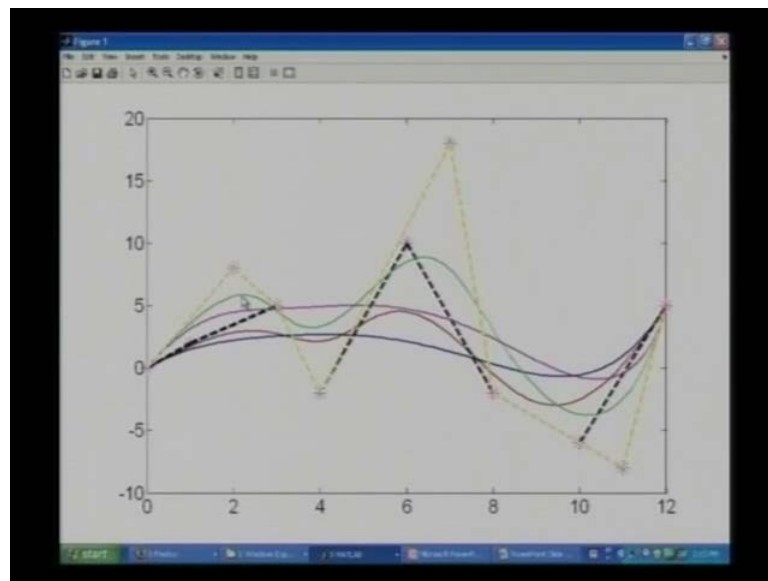
Now, I get a much better looking B-spline curve, which passes or can to pass or at least the impression that passes to the last data point, let us now experiment to the position of these data points.

(Refer Slide Time: 01:08:35)



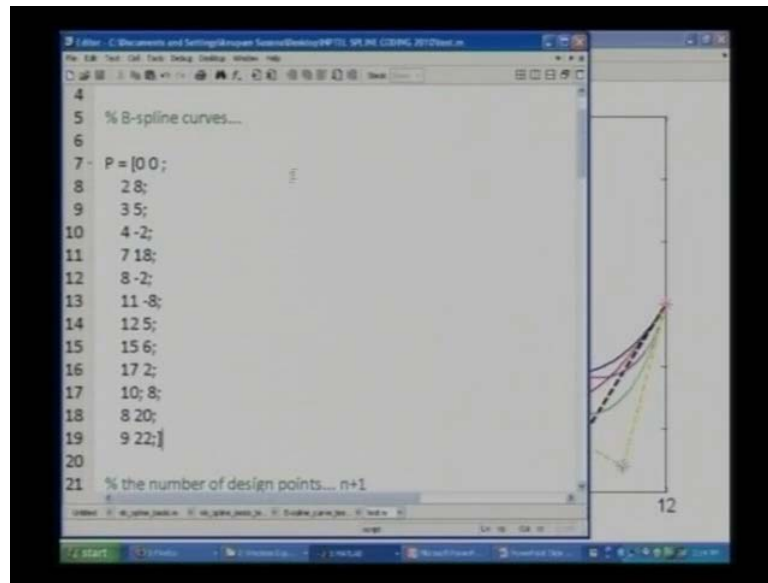
It is go up an arbitrarily change deferent values here, let this be 7 let this be 18 let me not clearly act of a this figure, so that we can compare the new results. So, the information in figure one says and let me plot the two curve models using deferent colors, let me use green for the B-spline curve and may be magenta of the corresponding Bezier curve. Let me plot the new control polyline using color yellow, let me go up and change the entries further. Let me make this 11 make this minus 8 let me work with 2 here make this 8 and execute the code now.

(Refer Slide Time: 01:10:08)



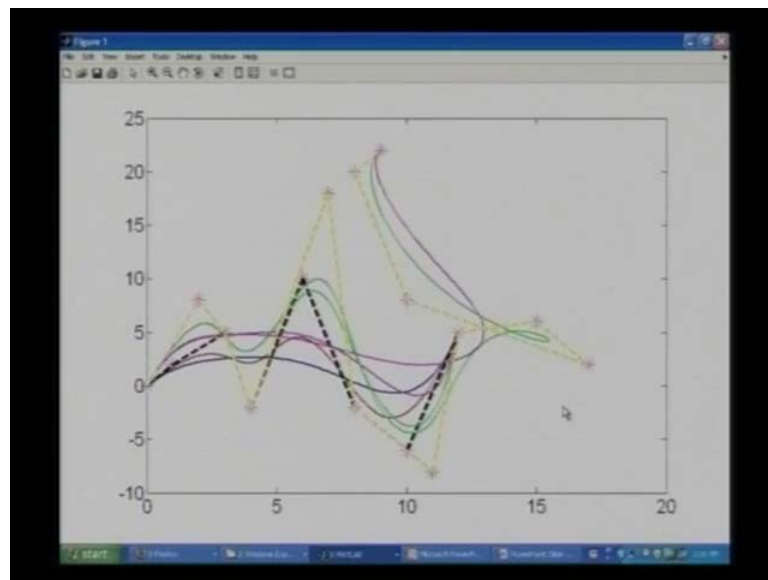
The red curves here is the original B-spline curve, this blue is the original Bezier curve, the green one here is the new B-spline curve and the magenta here is the new Bezier curve. Look at half closely the B-spline curves follow the angulations reflected by the control points or the control polyline, for that why do you think this is possible. It is this is because of this (()) property or this strong local bar is center city that the B-spline basis functions half to half. Let me add a few more points as the last experiment for this lecture.

(Refer Slide Time: 01:11:30)



I add point 15.6 17.2, let me go back to the little bit 10 8 8 20 and may be 9 22, I am arbitrarily specifying this data points. Let us now see what happens, looks like I have made an error here I should not be using this semi colon if you what happens now?

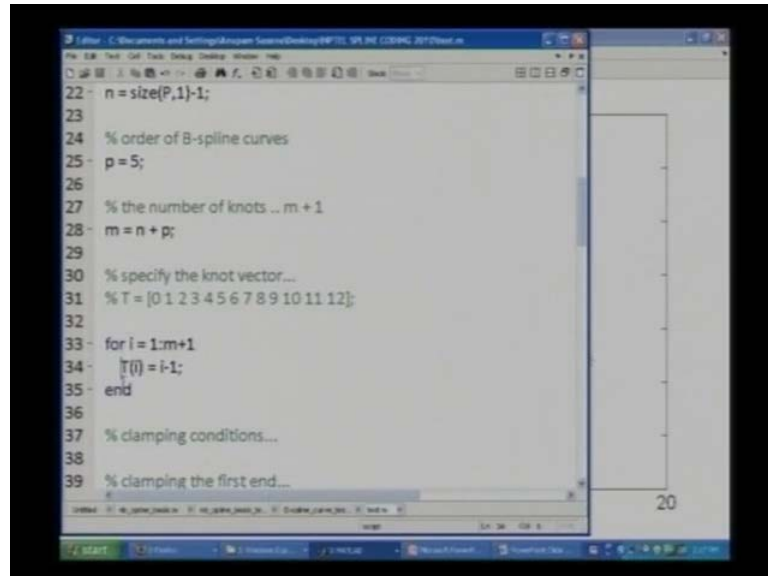
(Refer Slide Time: 01:12:38)



Try to concentrate on this green curve here and this magenta curve here, now if you notice a part of the B-spline curve remains the same only some part changes in shape. Again emphasizing local shape control where as this new Bezier curve changes global in shape, it may be fun and it may be lack easier for you if you learn the concept properly to

play or experiments with B-spline curves. You can think about how you can interactively move is data points, how you can automate the knot vector generation we have in talk about that.

(Refer Slide Time: 01:13:50)



```
22 - n = size(P,1)-1;
23
24 % order of B-spline curves
25 - p = 5;
26
27 % the number of knots .. m + 1
28 - m = n + p;
29
30 % specify the knot vector...
31 % T = [0 1 2 3 4 5 6 7 8 9 10 11 12];
32
33 - for i = 1:m+1
34 -     T(i) = i-1;
35 - end
36
37 % clamping conditions...
38
39 % clamping the first end...
```

But, you can think of computing the knot vectors like the way we discussed in the previous lecture, you may want to write another function to generate this array.