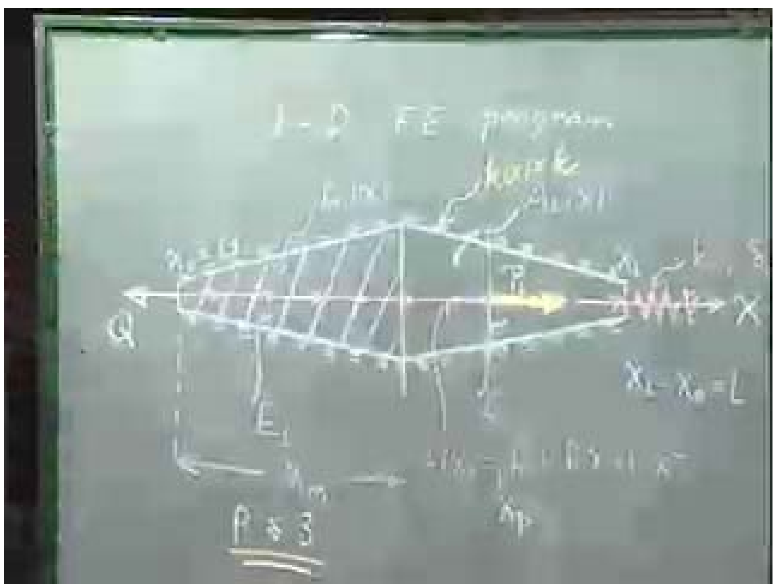
Finite Element Method

Prof. C.S. Upadhyay
Department of Mechanical Engineering
Indian Institute of Technology, Kanpur

Module - 5 Lecture - 1

Today in this lecture, we are going to talk about the one dimensional finite element program. Before we write the one dimensional finite element program, we have to fix what should be the features of the program; that is, what the code should be able to do, what kind of material data, loading data, and boundary condition it should be able to handle.

(Refer Slide Time: 00:09)



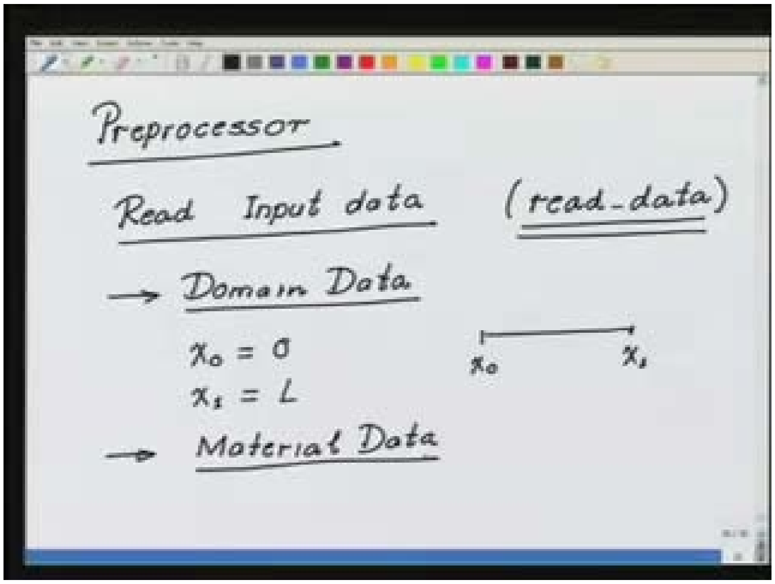
Let us take an example for the following case. This is our member (Refer Slide Time: 01:24) with this line being the line of symmetry or we call it as x-axis, in this member I can have a material in this region and some other material in this region. Here we would like to have material constant E_1 and the area, if we see in both the members, is tapered but with different tapersation. Here (Refer Slide Time: 02:14) I say area is $A_1(x)$. Here the material constant is or the Young modulus can be E_2. Taper ratio or the taper is given by $A_2(x)$. This is what we would like the code to handle. Where does this transition happen? This transition we are going to say is at the point x_m, this can be point x_0. This could be point x_1. We know that x_1 minus x_0 is equal to L. We are going to set our x_0 to 0.

Further, the other features are the following. I would like a distributed load along the centre line (Refer Slide Time: 03:51) this intensity f(x) given by f_0 plus f_{1x} plus f_{2x}(x square) that is we would like up to a quadratic distributed load. We would like to add some more features to our program. We like this program, if we desire on elastic support but what we would like is the elastic support should be distributed through the whole domain; that is, it should be applied to the full domain. We would like this elastic support to have an intensity k(x) is equal to k_0 so our elastic support has uniform spring constant we do not want the spring constant to be very different; it should not vary.

Another feature we would like to have is, let us say there is a point here (Refer Slide Time: 05:35) the location of which will be given by $x_{p.}$ At this point I apply a concentrated load of intensity P_1 or phi P_1. This is the concentrated load, I am applying here phi P_1 and we would like to handle only one concentrated load, not more than that. Further now, we would like our code to handle various sets of boundary conditions. What could be the boundary condition? I can have here an end load Q which is tensile. While, at the other end I could have the end force given in terms of the force applied by end spring. F is a force applied by end spring, for the spring I have to specify the stiffness coefficient of the spring and whether it is free compression or not delta L is the initial compression of the spring. This spring could be applied here this force could be applied here we could have any mix and match of various kinds of boundary conditions.

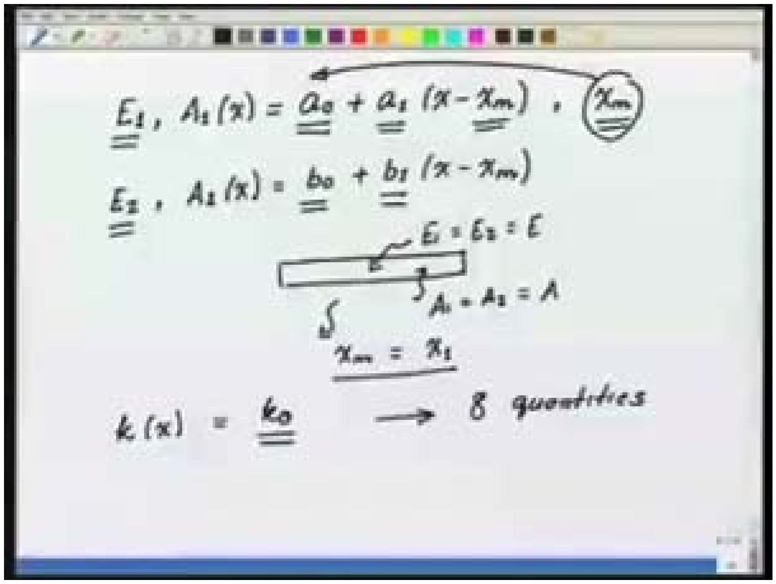
Further, we are also looking at boundary conditions which are symmetric in nature, we could fix this in. We would like to handle all these kinds of boundary conditions at each of these ends. I could choose anyone of the three results there; this is the problem we would like to solve. The other approximation that we want is P; let say is less than equal or to 3. If we look at this problem, as far as making the mesh is concerned, we have to honor certain boundaries. Which boundaries we have to honor? This boundary (Refer Slide Time: 07:45) where there is a transition of a cross sectional area or the material we want both the transition to be the same, we do not want separate. Another boundary is here (Refer Slide Time: 08:00); we have to put a node here because at this point a concentrated load is applied. If we see that the mesh is broken, the domain is broken into three parts; from this part from x_0 to x_m from x_m to x_p and from x_p to x_0. These are the three parts that we would like to handle. Let us now go and essentially look at, how to design the pre-processor to handle all these cases?

(Refer Slide Time: 08:41)



For the pre-processor design, we would like to first read the input data. This could be through a program. We can call it Read Input data; one can name it whatever he wants to but the name should reflect what that program is doing; so read data is our program. What would we like to read in? First, we would like to read in the Domain Data. From our figure that we have drawn there (Refer Slide Time: 09:45) domain data means I am interested in the extremities of the domain x_0 and x_1. So x_0 which for us is going to be 0 and x_1 which is equal to the length of the member or we could have said $x_{1\ minus}$ x_0 is equal to L but here we are setting x_0 is equal to 0 and x_1 is equal to L; this is what we have input first. We would like to input the material data, so how would we write the material data. In the material data we have to input the values of E, E_1.

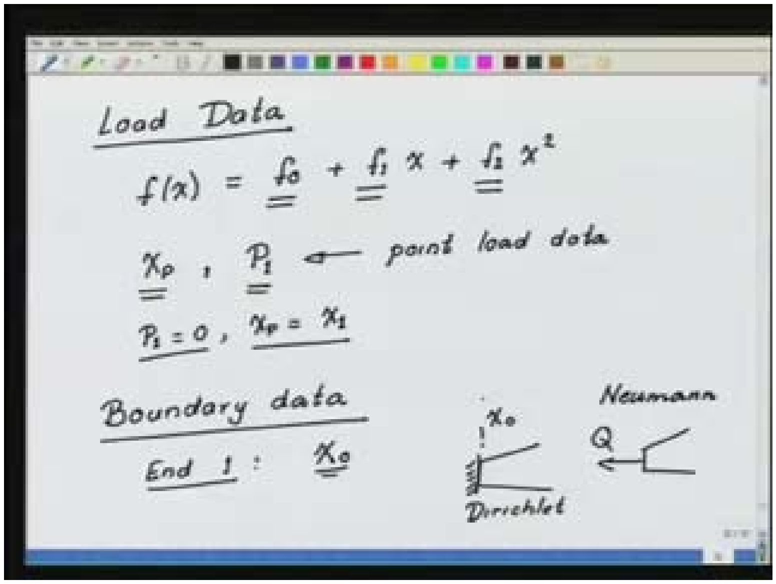
(Refer Slide Time: 10:41)



We want to have the values of E_1 as the function of x which we are going to given in terms of two parameter a_0 plus a_1(x minus x_m) so this is given in terms of a_0 plus a_1(x minus x_m). If I have to give this then I have to also input x_m. This E_1 a_0, a_1 x_m have to be input first, we have to input for the second part of the domain E_2, A_2 as the function of X is equal to b_0 plus b_1(x minus x_m). So the first part of the domain we input these two things, for the second part of the domain we input this quantity (Refer Slide Time: 12:00). It will be better to first input the transition point x_m as a separate entity and then for the two parts of the domain we give E_1A_1 and E_2A_2. In case I want only a single domain, material domain; for example, I take this bar for which E_1 is equal to E_2 is equal to E and A_1 is equal to A_2 is equal to A. It could be tapered but the taper is not changing. In that case, we will make x_m is equal to x_1. If we do that then what will happen is your program is going to take only E_1 and A_1 so as the domain because the second part is essentially at the end of the domain so this is will not come into play.

These data can be input all the time irrespective of whether I have a transition or not and according to the plot of x_m the appropriate data will be taken. If I see this is given, now if I have k(x) we know we wanted to be a constant k_0. This k_0 also has to be input as an input data. As far as the material data is concerned, we need to input the point of transition x_m, we need to input a_0, a_1, b_0, b_1, E_1 and E_2 and K_0. Totally we need 1, 4, 7, 8 data so 8 quantities has to be input. This quantity is a user supplied quantity because, this is something that the user is selling this corresponds to the domain the material data of interest to me. We have the material data we would like to give the load data.

(Refer Slide Time: 14:31)

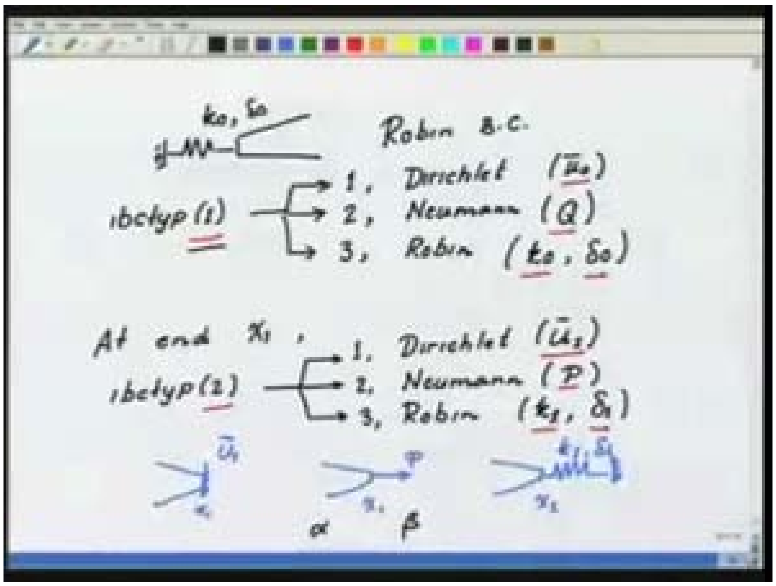


What is that we have to input as far the load data is concerned? We have said that our distributed load is defined over the whole domain if not defined piecewise. It could define in piecewise that it will only add little more to the complexity. Here we want to define it globally that is in the whole domain as f(x) as I have drawn in the figure is f_0 plus f_1x plus f_2x square so these quantities f_0, f_1, f_2 has to be input. If I want a constant load I make f_1 and f_2 is equal to 0. If I want a linear load, I will make f_2 equal to 0. If I want no load then I make f_0, f_1, f_2 equal to 0. This way we can upto a quadratic loading as a distributed load that we want.

What else do we want? We want to give whether point load or not, so if there is a point load we would like to know what this point is? Where the point load is applied and the value of the load P_1? Both these data has to be given. If I do not want the point load, then in case I do not want it then I will set P_1 equal to 0 and x_p equal to x_p plus 1. I am going to make this value 0 and I am moving this point where this 0 load is applied which is meaningless to the end of the load, just for our convenience. We have to remember that when P_1 equal to 0, I will have to set x_p is equal to x_1 we do not give x_p equal to some point interior of the domain; this data is available to us.

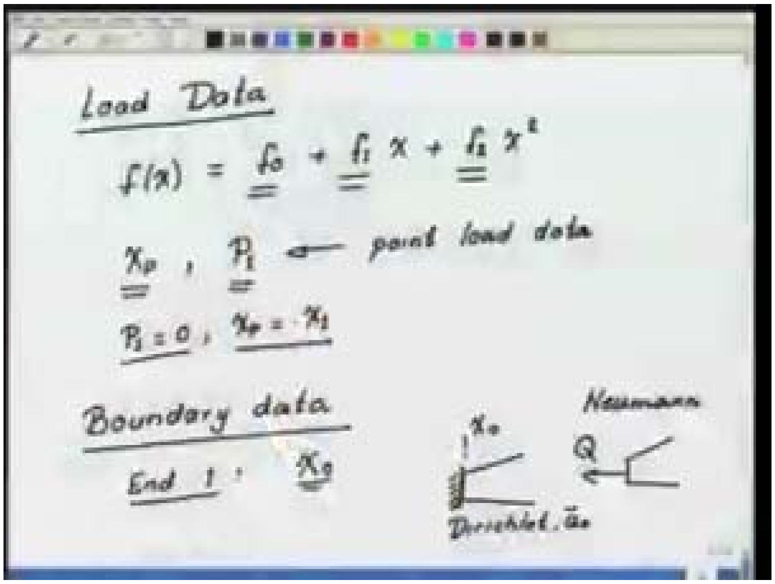
We have the domain data, the material data, and the load data. What else do we need? We need the boundary data because we need to specify the various boundary conditions. Let us now say End 1 (Refer Slide Time: 17:21) this corresponds to the point x_0. If I take member, what boundary conditions can I have at this point x_0? I could have (1) the displacement is given it need not be 0, it could be some value.

(Refer Slide Time: 17:46)



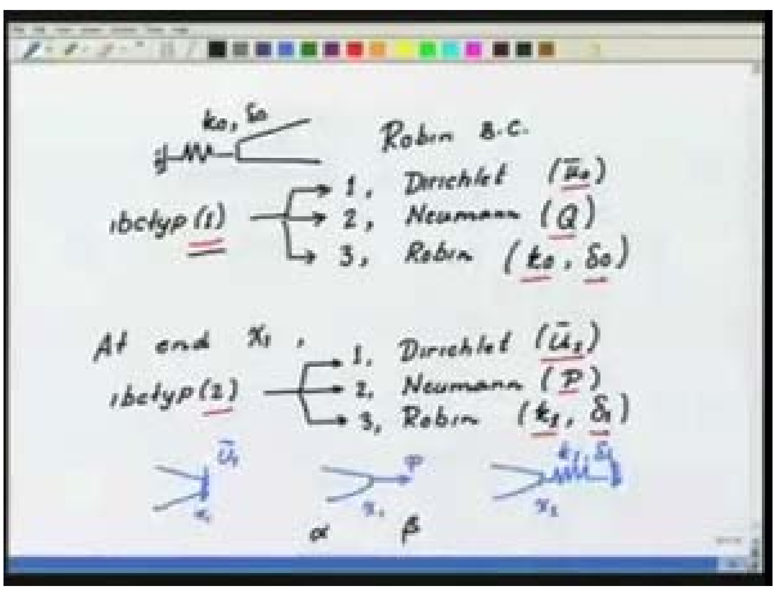
I could have a second candidate here; Q is given at the end, and the third candidate that the end has a spring with a constant k_0 and an initial compression $delta_0$ given. We are going to call them this (Refer Slide Time: 18:10) if we remember that we had introduced this as the Robin boundary condition and the previous one if I go up this is Dirichlet and this one is Neumann.

(Refer Slide Time: 18:13)



We have to have the feature that at this end I should be able to give any one of the boundary conditions and along the boundary condition type I should be able to input the data that goes with it. Here if I say Dirichlet means, u_0 is given as u_0 bar.

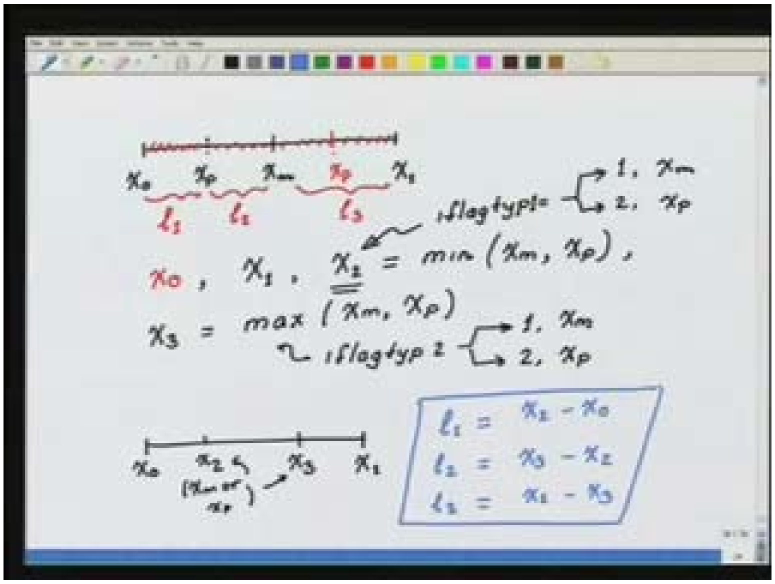
(Refer Slide Time: 18:46)



Let us say that we will specify the boundary condition type at End 1 by the following number: it is 1 when it is Dirichlet, it is 2 when it is Neumann, and it is 3 when it is Robin or mixed. When it is Dirichlet we have to input the value at the end - the displacement u_0 bar; if it is Neumann I have to specify what this end load is. It is Q, it is positive if it is tensile. If it is Robin then I have specify the spring constant k_0 and initial compression $delta_0$. These are the information that we need to have at the boundary. Similarly, at the other boundary at end x_1 we say is given by the boundary node 2, I will say of type 2. This would be again 1, 2, or 3 depending on whether it is Dirichlet Neumann or whatever or Robin. If it is Dirichlet then at this end (Refer Slide Time: 20:38) it is u_1 bar, if it is Neumann then it is the end tensile load P and if it is Robin then we have to give the end spring constant k_1 and the end compression $delta_1$. So all these data has to be input, so this has to be given, this has to be given, this has to be given, and this has to be given the type the boundary condition type has to be specified at each of the end. What is the type and accordingly these data has to be input?

For completeness sake specify what this means at the other end. Here (Refer Slide Time: 21:44) I mean that when it is 1 then I am specifying the displacement here. When it is 2, I am specifying the end load P, when it is 3, then I am specifying this spring constant and the initial compression delta at the end x_1. As far as the boundary condition is concerned this is the information we would like to have. The boundary condition site and the corresponding data, how do we input this data? We could have one input as the boundary condition type the other input as the two parameters alpha and beta. For the Dirichlet site, it will give alpha equal to u_0 so at each of the end alpha equal to u_0 bar. For the Neumann part, at n_1 it will give alpha equal to Q and for the Robin, it will be alpha equal to k_0 beta equal to $delta_0$. Similarly, I will do at the end x is equal to $x_{1;}$ at the two ends we have to give these two parameters. The second parameter for the first two types is irrelevant parameters but anyway we can specify anything and it will not be used. After I input all these data, I have to create some information which is needed. In the read data program itself we construct that information; what is that information? The information is as follows.

(Refer Slide Time: 23:11)

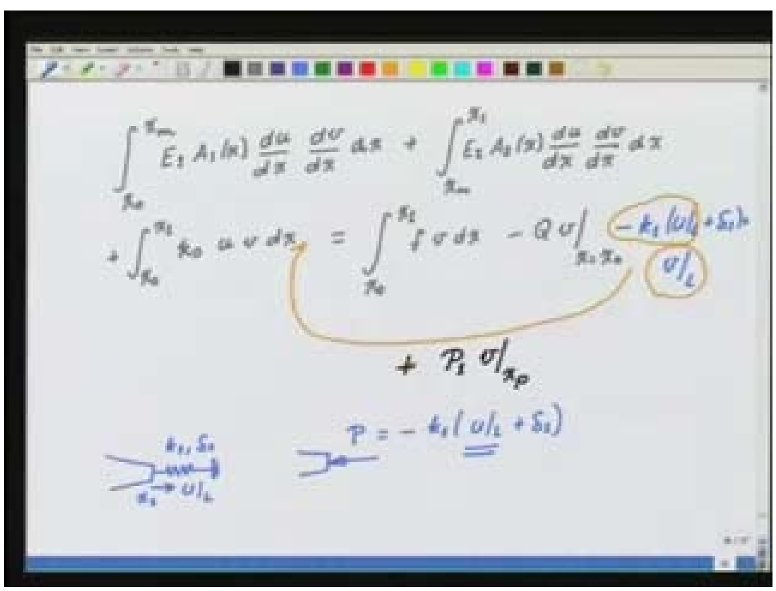


Let us say this is the point x_0, this is the point x_1, this is the point x_m, the x_p could be here (Refer Slide Time: 23; 39). The x_p could be here but I have to honor this transition as we have said. Transition means if the x_p is here then this is now a piece by itself, this is a piece by itself and this is a piece by itself. We have the 3 pieces which are given by in this case x_0 to x_p, x_p to x_m and x_m to x_1 and the length will be given by l_1, l_2, l_3; this information has to be available to us. The computer does not know whether the x_p is happening before x_m or it is coming after x_m. This information has to be created for the computer to understand that this partition has these points x_p, x_m and so on as the end points. Let us now find the three points; x_0 is fixed, we will say x_1 is fixed, x_2 is equal to minimum of x_m and x_p and x_3 is equal to maximum of x_m and x_p. When we say x_2 is the minimum of x_m and x_p along with it we will carry the flag, whether it is material point or it is the point of point load. Let us make the flag, let say that this will have an indicator x_2 will have an indicator iflag type it is equal to 1 if this is x_m or it is equal to 2 if it is x_p (Refer Slide Time: 26:15).

Similarly, here we will call it iflag type 1. Here for the max again, we will have the corresponding flag associated with it iflag type 2 this is equal to 1. If this is the point x_m and it is equal to two if it is x_p. This is very useful information that we are creating along with this point x_2 and x_3. It is going to be very useful when we are further constructing the mesh and deciding the element where they lie in, which material domain they lie in. Let us have this flag and along this point; what do we have now? I look at the domain I have the point x_0, the point x_1 then I have x_2 and I have the point x_3 and the flag tells me whether it is x_m or x_p here and here (Refer Slide Time: 27:38). This is the flag telling me this thing, but as far as the partitioning the domain is concerned we have this distinct partition. I will measure the length l_1 is equal to x_2 minus x_0, l_2 is equal to x_3 minus x_2 and l_3 is equal to $x_1 - x_3$. These lengths we are going to measure; note something that this l_1, l_2 and l_3 can each be either 0 or l.

For example, if I say that my material point and both the material point and the point load are at x_1 that is I do not have two material domain and I do not have point load applied in that case this x2 minus x0 is going to be l, x_3 minus x_2 is going to be 0, x_1 minus x_3 is going to 0 only l_1 going to be coming out to be l in that case and other two things are coming out to be 0. If I say that point load does not exist, I only have a material transition. If I have material transition then I will only get, in that case, x_2 is equal to x_3, x_3 is equal to x_1. So I get l_1 is equal to x_2 minus x_0, l_2 is equal to x_3 minus x_2 which is perfectly fine, l_3 is equal to 0. I will get only the two partitions. With this is data, now do we have all the information that we need? We now need to get one more data which is the mesh data.

(Refer Slide Time: 29:53)



We have defined the three partitions of the domain remember that these are not nodes, but these are the partitions of the domain honoring the material boundaries the point of application of the point load and the extremities of the domain so it is $x_{2,}$ x_3, x_1 with length $l_{1,}$ $l_{2,}$ l_3. We can ask the following question; in each of the partition, how many elements do we need to put? This is an input data which has to come from the user. User decide to put different number of elements in each of the partition based on his needs, based on certain apriory information that he has about the smoothness of the solution and so on.

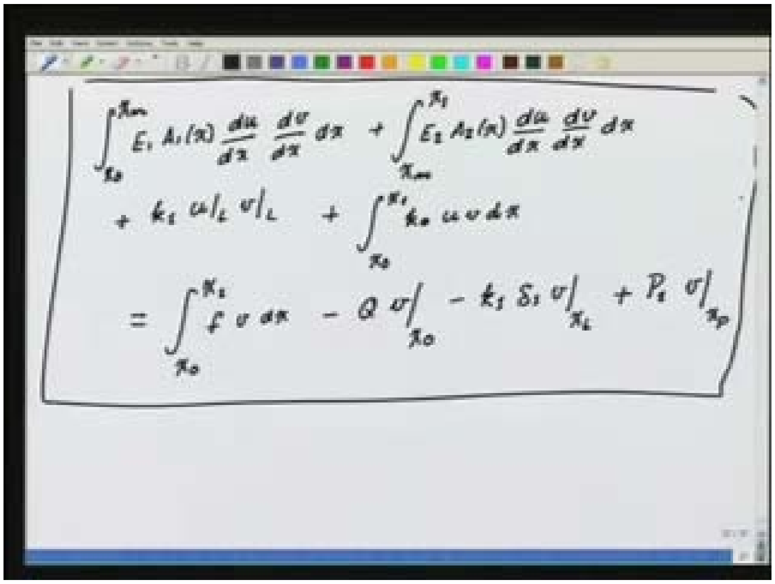
Let us say that in each of this region, we are specifying the number of elements we need n_1, n_2, n_3 and we are assuming piecewise uniform mesh. What we are saying is, scalar as an input data, what number of sub division we want in the first piece which is length l_1 in the second piece which is l_2 and the third piece which has length l_3; so these are given by n_1, n_2, n_3. Take in each piece a uniform sub division so that I get n_1 number of elements in the first piece, n_2 number of elements in the second, n_3 number of elements in the third. The total number of elements if we see will be equal to n_1 plus n_2 plus $n_{3,}$ if all these partitions are distinct.

We see that in case, one of these lengths or two of these lengths become 0, then this partition even though we specified it is going to be trivial information; we really not use that information. What more we need to specify? We have specified all the information that we needed to have with respect to the domain, with respect to the material, with respect to the loading data, with respect to the boundary condition, and here with respect to the number of elements we want in each of the pieces. The last thing that we need to specify is the order of approximation, (small) p this we are assuming is uniform in the whole domain (Refer Slide Time: 33:14). This is very important that we are going to certify the order of approximation to be uniform. We need not do that, we could specify different orders of approximations which can also be easily handled but, let us stick to that we are going to fix this order of approximation to be uniform.

For the problem that we have written, that is I have drawn on the black board for that problem what is the weak formulation? The weak formulation will come out to be integral from x_0 to x_m (Refer Slide Time: 34:00) according to the figure on the board. I will have $E_{1,}$ $A_1(x)$ du dx dv dx plus integral x_m to x_1 E_2 A_2 (x) du dx plus I will have integral of x_0 to x_1 k_0 u v dx. This is the part due to the spring, this is stiffness part due to the spring; this will be equal to integral x_0 to x_1 f v dx . From the figure on the black board, we have an end load at the point x_0 that will be the tensile load so it will be, Q into this will give work, Q into v at x equal to x_0. At the other end at x equal to x_1, if I look at the end there let me draw here end x_1, (Refer Slide Time: 36:05) here I have k_1 and initial compression $delta_1$. If there is an initial compression $delta_1$ then further displacement of this end due to the applied force as the reaction to the applied forces of an amount u at L. The spring actually compresses by an amount $delta_1$ plus u at l so the total compression is $delta_1$ plus u at L. The force applied by the spring is back on the member P equal to minus k_1 u at L plus $delta_1$. The u at l is an unknown, so this part minus k, let me first write it here, minus k_1 u at L plus $delta_1$ multiplied by v at l. This part (Refer Slide Time: 37:05) minus k into u of L because this is in terms of the unknown, I am going to carry it over to this and I am going to leave to this into v L obviously, I am going to leave minus k_1 $delta_1$ into v of L on the right hand side. This part goes to the left hand side because we are collecting all the unknowns on the left hand side.

Remember this part because this is going to be important as far as applying the boundary conditions is concerned. Similarly, at the end 0 we can do the same thing; where this end load Q which can be now given in terms of the compression of the end x_0 if I had a spring load there. Let us now see if this is the weak formulation. In the weak formulation which we could also derive using the variational principle something else is missing here (Refer Slide Time: 38:28). We have not added the part due to the point load at point x_0; so v at x_p so this part has to be added to the load vector site.

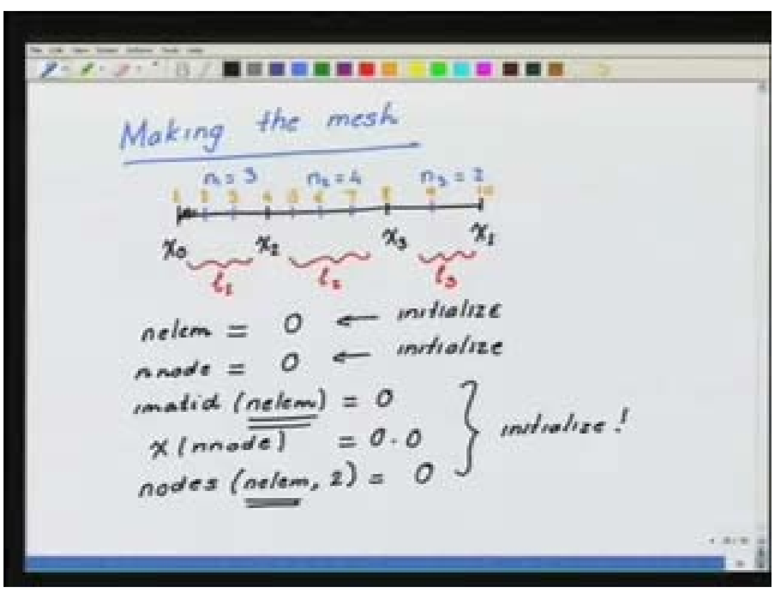
(Refer Slide Time: 38:51)



If I look at this weak form in its complete entirety, it will be x_0 to x_1 E_1 A_1 (x) du dx. dv dx , dx plus integral plus x_0 to x_m here x_m to x_1 E_2 A_2(x) du dx, dv dx plus k_1u at L, v at L this is equal to integral from plus, I have to add another part let me add, plus the part due to the spring x_0 to x_1 $k_0$uv dx. This will be equal to the load part integral x_0 to x_1 f v dx minus the right hand side will be integral x_0 to x_1 f v dx minus Q into v at x_0 minus k_1 delta one v at x L plus P_1 v at xp. This is going to be our complete weak formulation or variational formulation whichever way we decide to obtain.

We see that this thing should reflect in what we are doing in the program. In the program we are going to do the integration piecewise for this material constant because the material constant is changing so I have to go from x_0 to x_m from x_m to x_1 and this anyway will be added from the boundary conditions, this is not a problem. This part, (Refer Slide Time: 41:20) since, we are doing this piecewise we could do this from x_0 to x_m x_m to x_1; again it can be piecewise and this is the same thing. Here also in the load I am going from x_0 to x_1 which means I can do it x_0 to x_m and x_m to x_1 and this part is from the boundary condition, this part is from the boundary condition. This (Refer Slide Time: 41:44) I will have to add as an extra part due to the point load. With this in mind, what we have developed till now as far as leading the data is concerned; the data is going to be used. Once we have read the data then what is the first thing we have to do? First thing is given that the data that we have read we have to make the mesh.

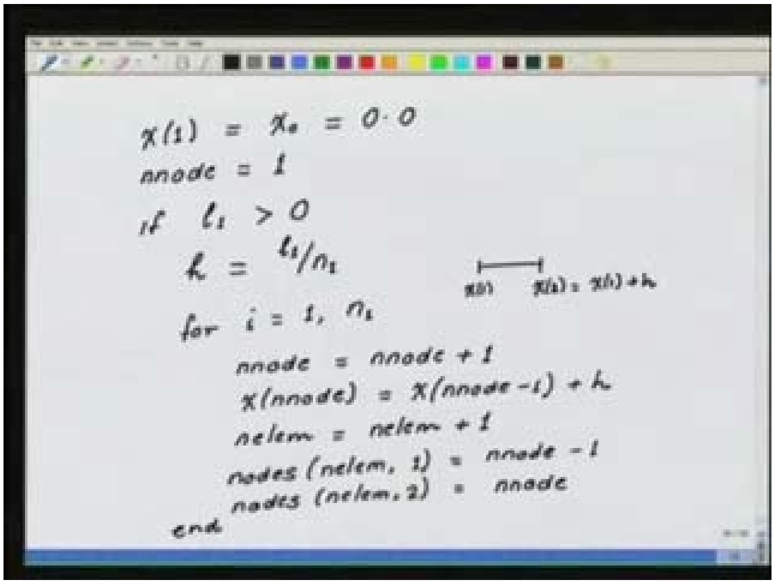
(Refer Slide Time: 42:22)



In order to make the mesh we have said that as far as the computer is concerned it has these points x_0, x_1, x_2, x_3; with length l_1, l_2, l_3. How do we go about constructing the mesh? We want a certain amount of partition in each piece. Let us say in the first one I am putting three elements so here I am going to put three equal elements so here n_1 is equal to 3. Let us say n_2 is equal to 4. I want to put elements here so let us say n_3 is equal to 2 so I put one element here. We should be able to construct these partitions using our program and along with this partition we should be able to construct the coordinates of these nodes we have created. How do we number the nodes? We are going to say this is going to be node 1, this is going to be node 2, 3, 4. We are going to number them sequentially 6, 7, 8, 9, 10 so this thing should be possible while we are making mesh as we are numbering the node we should be able to give the coordinate of the node. Let total amount of element in the mesh will be given by the parameter nelem that is the total number of elements. First we are going to initialize it.

Further, we are going to say that let the total number of nodes in the element will be given by n node; we are going to initialize this also. The initialization of the quantity is very important. We have written the program, we have to know what material properties to use in a particular element? If I am interested in this element, how does the computer know what E to put there? What A to put there? So somehow I have to inherit the information for each of the element from somewhere. We are going to have an ID or a pointer or an indicator of which material domain a particular element lies in through this vector imatid, this is the integral. To this we are going to set this is of size, so we fix some total number of elements. Let us say we say the maximum number of elements in the mesh cannot be more than hundred. We will initialize, we make this vector of size hundred and initialize all entries to 0. Further, we will have the nodes of each of the points also initialized to 0. Also, we will have another information which we need; we will say it will create these array nodes, this is also initialized to 0 (Refer Slide Time: 46:50); so all entries of this array which is if we see this element is based so this total number element is 100. This is 100, 2 array; all entries of this is set to 0. These are all the initializations that we will have to do.

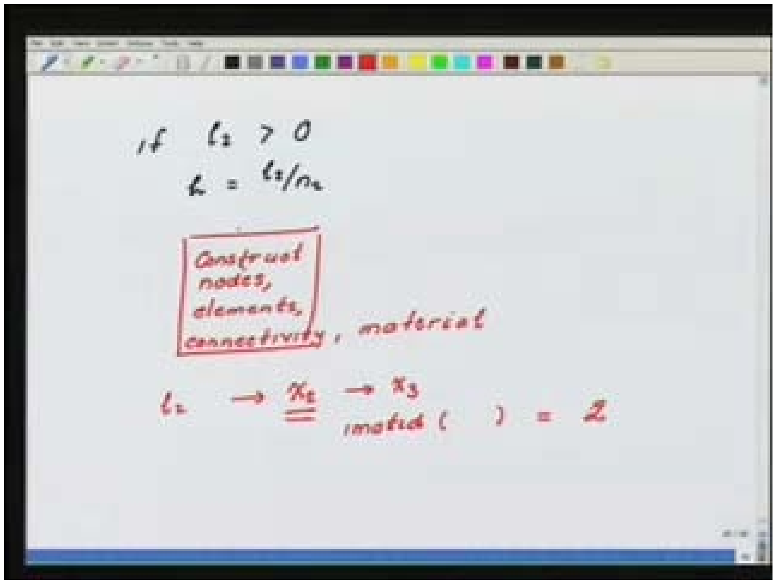
(Refer Slide Time: 47:37)



After this initialization, we are going to construct the nodes. The first node is given the value x_0 which is equal to 0.0. I am going to set, because I already created this node, n node is equal to 1 (Refer Slide Time: 47:47). I am incrementing the total number of nodes in the mesh as I am making the nodes.

If l_1 is greater than 0, if l_1 is equal to 0, I do not do anything. If l_1 is greater than 0, then I find the mesh type because this is uniform meshing in this case l_1 divided by n_1. For i equal to 1 to n_1 that is I take all these partitions, I am going to increment the number of nodes and I am going to say x n node is equal to x n node minus 1 plus h. Given the first node which is $x_{1,}$ I created the coordinate of x_2 is equal to x_1 plus h and so on. I construct all the n plus 1 nodes in this piece. As I am constructing these consecutive nodes I am also incrementing the number of elements. So I am going to say nelem is equal to nelem plus 1 that is two consecutive nodes have formed an element in between. I am going to say nodes nelem comma 1 is equal to n node minus 1 nodes nelem comma 2 is equal to n node. I am telling what are the two n nodes for this two particular element are. Then I am going to end this loop. I have found the mesh size in the piece, looped over to load in the piece, found what the loads are, and constructed it and so on.

(Refer Slide Time: 50:24)



I repeat the same process for l_2; we will say is l_2 is greater than 0 then I will find h is equal to l_2 by n_2 and continue. Similarly, I will do for l_3 for each of the piece. While I am constructing the element, I am also going to check whether the first point that is the x_2 if we remember x_2 and x_3 that is the first point it is due to a material interface or whether it is due to a point load. If it is due to a material interface then we know that l_1 is from x_0 to x_2 so here I am going to give imatid of element is equal to 1 or 2 depending on where it interfaces, at least here depending on what this flag was (Refer Slide Time: 51:38). This I am going to appropriately assign atleast in the first phase it is always going to be free element, the material one.

This way I continue here also (Refer Slide Time: 51:50) I assign while I construct the node, so construct nodes, elements and connectivity and then material information which material domain it lies in. In this case for l_2 we know that l_2 actually goes from x_2 to x_3. If x_2 was a material point then here my imatid of all the elements will be equal to 2. If x_2 was x_3 that is, it was a point load point, not a material point then it remains as 1 because the material is not changing because only the point load has been applied and so on we continue doing. This gives us all the information about the mesh, the elements, nodes, the coordinate, connectivity, number of elements, all these information comes out at the end of the program. This is a very important component of the pre-processor that is we have made the mesh and all the relevant information that has to go with the mesh as far as our computations are concerned. We will continue with this in the next lecture and we are going to now talk of how to construct the degrees of freedom information for the given mesh for the given order of approximation. From there we will continue further, talk about assembly solver and so on.

Thank you.