PROFESSOR: Ladies and gentlemen, welcome to lecture number five. In the previous lectures we talked about the formulation of the finite element method, and we derived already some element matrices. I want to continue that discussion in the next lecture. However, now I would like to spend some time with you and discuss with you the implementation of the finite element method. I would like to present to you some important aspect regarding the implementation of the procedures that we talked about already.

We derived the equilibrium equations KU equals R in the earlier lectures, where R, of course, contains various contributions due to body load, surface load, and so on. And we particularly pointed out that that the total structural stiffness matrix is obtained by summing the element stiffness matrices, as schematically shown here. This we refer to as the direct stiffness procedure, and that direct stiffness procedure is also applicable to the load contributions, where we sum the element load contribution into a total nodal point load vector RB, and similarly for RS and so on, which then together make up the total load vector R.

We pointed out that the stiffness matrix of a typical element M is obtained via this relationship here. Notice in this integer, BM is the strain displacement transformation matrix, CM is the stress-strain law, and here we have Bm transposed. We're integrating this part over the total volume of the element. The RB vector for element M was written as shown here, where FB are the body loads per unit volume into the coordinate directions considered. Of course, these FB loads are a function of the coordinate is the element, and HM is the displacement interpolation matrix. We are once again integrating this product over the total volume of the element.

In both of these equations, we have been dealing with the HM matrix and the BM matrix. Of course, the BM matrix, as we discussed, is obtained from the HM matrix by appropriate differentiations and so on. Now we pointed out that if we had N number of degrees of freedom in the total structure, then our HM matrix is of order k by N. Capital N, the same N that we're talking about here. k is equal to 1, 2, or 3, depending on whether we are dealing with a 1, 2, or 3-dimensional analysis. The BM matrix is of order l by N, where l is equal to the number of strain components that we are including in the element. In a 3-dimensional element, for example, we would have l being equal to 6, because there are 6 strain components.

It's important that we talk here about capital N being equal to the total number of degrees of freedom. This way, this KM matrix here is a matrix of order capital N by capital N. In other words, this is an N-by-N matrix, and this RB vector here is a vector of length N. It's has N entries this way, and of course only one column.

Having here an N-by-N matrix and an N-by-1 vector here, we can directly assemble the element contributions into the global structural matrices and vectors. We can do this because this matrix here has the same order at that matrix here, or each of the element matrices has the same order as the total structure matrix.

Well, in practice, this is, of course, not efficient. We are dealing with a very large system. The capital N that we're talking about here might be 2,000, 3,000, 10,000, and it is not efficient to calculate here in a capital N-by-N matrix for each element, because we recognized already earlier that a large number of rows and columns are simply zeros in this matrix here. In fact, only those rows and columns contain non-zero contributions or non-zero elements which correspond to the element degrees of freedom.

In practice, therefore, we calculate compacted element matrices. We are calculating for element M or for typical element M matrix o order little n-by-n. Notice that I left out the superscript here. And our RV vector would be an n-by-1, lower case n, where n is now the number of element degrees of freedom.

The H matrix that we're using and the B matrix that we're using now is a k-by-little n,

l-by-little n, versus here, k-by-capital N, l-by-capital N. We talk here about the compacted matrices, and these compacted matrices contain really all the information of these, which we might call blown-up matrices. Because the non-zero entries in these blown-up matrices are all contained in the K matrix that I talked about here.

What we need, then, in practice is the K matrix, the compacted matrix, with connectivity arrays, in order to be able to go through this process here. In order to be able to assemble the compacted element matrices into a global structural stiffness matrix. That is done by a connectivity array, as I will be discussing just now. That is one of the important aspects that I would like to discuss with you in this lecture.

Well, I have prepared here some view graphs as before, and let us look at the first one here, in which I simply summarize one through three phases that we are going through in a finite element analysis. The first phase consists of the calculation of the structure matrices K, M, C, and R, whichever are applicable. Here I mean static dynamic analysis, et cetera. The second phase, then, once we have established these global structural stiffness matrices, consists of the solution of the equilibrium equation. Of course, this solution is carried out differently in static analysis and in dynamic analysis, and I will be discussing the solution procedures that we're using in later lectures.

Once we have solid state equilibrium equations for the displacements, velocities, accelerations, we can calculate the element stresses. The element stresses are then obtained from the strain displacement matrices and the stress-strain law, together with the displacement as a nodal point displacement that we have evaluated. In this lecture, I really want to discuss with you only phase one. In other words, how do we calculate the structure matrices whichever are applicable?

This phase one can be subdivided, again, into three different steps, and I summarize these on this view graph. The nodal point on element information are read and/or generated first. Notice that I also included generated here, because

there is much repetitiveness in many finite element analyses, and we want to take advantage of that repetitiveness, therefore generate information data whenever is possible.

Then once we had the nodal point in element information read into the computer program, we calculate the element stiffness matrices. The mass and damping matrices, if they are applicable, and equivalent node upon loads. Once we calculated the element stiffness matrices, we can assemble math and damping matrices, of course. Also, we can assemble all of these contributions here into the global structure matrices, K, M, C, and R. And I would like to discuss this process in detail with you.

Before going into the details, let me mention that the procedures which I will be discussing this you are really the procedures that are used in the computer program Sab and ADINA. But very similar procedures are also used in other computer programs.

The first important aspect that I like to mention to you is that in a finite element analysis, we have to define what degrees of freedom we want to admit at the nodal point. In other words, if we have a finite element mesh such as this one here, I will be talking about that mesh in more detail later. Then if this is a plane stress analysis, in other words, here we have a cantilever plate, say, subjected to a load there, if it's a plane stress analysis, I only want to admit certain degrees of freedom at each of these nodes. We have nine nodes here. For example, at these nodes, there shall be no degrees of freedom admitted, or else I want to knock out all of the degrees of freedom to simulate the boundary condition along here, and at these degrees of freedom here, away from the edge, I have two degrees of freedom to correspond to a plane stress condition.

Well, if I want to do that for this analysis and similarly, of course, I proceed in other analyses, then I want to deal with each nodal point in turn. And here I have a typical nodal point i, say, at which we can have altogether a certain number of degrees of freedom, the maximum that I might want to admit in a computer program. Now in

Sab and ADINA, we admit a maximum of 6 degrees of freedom, however, we might have more. In piping analysis, we might have generalized organization degrees of freedom, such as we are using in ADINA P, and then we would have more degrees of freedom at the nodal point i.

But let's talk now only about 6 degrees of freedom and the maximum. I denote the first degree of freedom as the U degree of freedom, the second degree of freedom as the V degree of freedom, the third degree of freedom as the W degree of freedom, the fourth one as the rotation about the x-axis, shown by this vector. Notice that here I'm talking about a vector such as this one here shown. And theta Y is the fifth degree of freedom, theta Z is the sixth degree of freedom.

What we are doing then is to define a matrix ID, or we call it the ID array, identification array, which has a certain number of rows and a certain number also columns. The number of rows is equal to the maximum number of degrees of freedom that we can have in the analysis. Now in this particular case, we have 6 as the maximum-- 1, 2, 3, 4, 5, 6. The number of nodal points, of course, vary from analysis to analysis. They might be 1,000, 5,000, or even more.

For each nodal point, we have 1 column, therefore 6 entries. And for each nodal point, we can use these 6 entries to define whether a degree of freedom is active or non-active. A degree of freedom is active if stiffness is put into that degree of freedom, if stiffness is defined in that degree of freedom.

Well, let us go through a specific example. Here we have a cantilever analysis, cantilever plate analysis, and here we have a 2-by-2 elements idealization of that plate, containing altogether 4 elements. It's a plane stress analysis. The loading on this plate is while at temperature 100 degrees Celsius top, 70 degrees Celsius bottom. And there might be a concentrated load, such as shown here. Et cetera.

But let us now focus our attention on, how do we generate the element matrices, and how do we assemble these element matrices into the global structure matrix? Well, this is a 2-by-2 element idealization . We notice the following. We notice that we have altogether 9 nodal points-- and let me use the pointer here. 1, 2, 3, 4, 5, 6,

7, 8, 9 nodal points-- that with these 9 nodal points, we have defined 4 elements-- element 1, 2, 3, 4-- that element 1 and 2 have the same material properties, and element 3 and 4 have the same material properties. Notice the Young's modulus is here 2 times 10 to the sixth, whereas it is 1 times 10 to the sixth for element 1 and 2. Therefore, we will have to define two sets of material properties-- one set for these elements, and one set for these elements.

We also notice that since we have 9 nodal points altogether, we could have a maximum of 18 degrees of freedom in a plane stress analysis, because there are two degrees of freedom in a plane stress analysis for each nodal point. However, these 18, of course, represent the absolute maximum, which in a static analysis, we could not use, because we would have to also constrain the element mesh sufficiently to obtain a solution, to have a stable structure, in other words, while the constraints here are that at these nodal points, that all displacement shall be equal to 0.

Now let us look at our ID array, the ID array which I mentioned right here. We will have, in other words, 9 columns here, because we have 9 nodal points, and we have our 6 rows here, because we have a maximum of 6 degrees of freedom. An active degree of freedom we denote by a 0 in this matrix. A non-active degree of freedom we denote by 1 in this ID matrix.

We notice the following. If I put here the number of nodal points along, we have 1, 2, 3, 4, 5, 6, 7, 8, 9. Of course, the degrees of freedom here are U, V, W, theta X, theta Y, theta Z.

Now, notice that what we are saying is that only for nodal points 4, 5, 6, 7, 8, and 9, we have these as active degrees of freedom. In other words, At nodal point 4, 5, 6, 7, 8, 9, we have active degrees of freedom. These will be active degrees of freedom. And let me finish these here too on this side. So we have these as active degrees of freedom.

These degrees of freedom and on this side are interactive, because their constraint. They are no displacement. Of course, stiffness is coming into these degrees of

freedom through, yes. So we could have made these active degrees of freedom. However, in addition, we also know that the displacements are 0 there, so we make them inactive on that basis.

The result, then, is that we have to read into the ID array as shown here. This is the typical reading that you would use in the ADINA computer program. Once in all of those entries here, and zeros there. Now the program knows that corresponding to these zeros here, we will have to set up element equations. In other words, actually, we have to set up for each zero one equation, one global degree of freedom. And the program that goes through this process as follows. It searches through-- let me go once more back-- through these elements of the ID matrix product that you have been feeding into the program. It searches through it, and replaces each one by a zero, going down column-wise, and each zero by a number. It starts with 1 and goes on consecutively to the maximum number of zeros that it encounters.

The final result, then, is this one. Maybe I can just put it on top, only to signify here that we had all of these now zeros which we are ones before, and some numbers in this corner here where we had zeros before.

Let me take the bottom view graph out, and so now we see what has happened. We have 0s everywhere here and equation numbers up here, which have been generated by going through the original columns of the ID array from top to bottom, and replacing the zeros by numbers-- 1, 2-- these were all ones, we put zeros here-- 3, 4-- put zeros here-- 5, 6-- put zeros here-- et cetera.

Now, these are the degrees of freedom that the structure will actually have, and they are defined because stiffness is coming into each of these degrees of freedom from the plane stress element, and we know them to be non-zero. In other words, if we look at our final element idealization again, we find that these vectors here are active degrees of freedom, which I pointed out earlier. In fact, this is here degree of freedom 1, this is degree of freedom 2, which is at nodal point 4, and here we have degree of freedom 3, and then 4, at nodal point 5. And those entries are just those here. At nodal point 4, remember nodal points along here. So this is nodal point 4.

We have the first and second degree of freedom of the structural model. At nodal point 5, we have the third and fourth degree of freedom of the total structural model. Et cetera.

Well, this is an important aspect. Let us now go on to some further aspects of the analyses. We will use this later on, once again, when we set up the connectivity arrays of the element.

Here is our finite element idealization, once again. And the next step now is to read in also the coordinates of all the elements and the temperatures at the nodal points. Now with this coordinate system, x, y, and z, as shown here, the coordinate of all of these nodal points can be read indirectly. You can read them up. if you know the length from here to there, being 60 centimeters and being from here to there 40 centimeters, surely all the coordinates can be defined. And knowing that the temperature is given here at 70 degrees Celsius and 100 degrees Celsius there, and there's a linear variation from top to bottom in temperature, we directly also define the temperature array, as shown here. These are the temperatures at nodal point 1 to 9, x, y, z coordinates at nodal point 1 to 9.

Of course, this is input. You have to input this to the computer program, just the same way as you have to input to the computer program the ID array. But then the computer program figures out what coordinates of these arrays pertain to the specific elements. And that is done in the following way. Once again, here is our element idealization. Now we have read in the fact that we want to have this as the first 2 degrees of freedom, then this degree of freedom. That we have already all established, that we have these degrees of freedom in the finite element mesh. The program knows that already. That it does know via the reading of the information in the ID array. We also have all the coordinates of the nodal points, and we also have the temperatures of all the nodal points. What we still have to now read in is how an element is connected to the various nodal points given in the structure. And that is the next important reading.

Here for element 1, we know that the node numbers are 5, 2, 1, and 4, and the

material property is number 1. Let's go back once more. For element 1, that is, this element here, I want to use this material property set, and I call that property set number 1. Property set number 1, also, for element 2. But property set number 2 for elements 3 and for element 4, because these element properties are different from these.

Well, so element one has nodal points-- I use the following convention-- 5, 2, 1, 4. I go counterclockwise around. So locally, I'm thinking of a coordinate system lying in the elements, such as shown here. And this one here being in the positive quadrant is my first nodal point that I assign to the element. This is, therefore, the local nodal point 1. Let's put another little picture here. This is a local nodal point 1. That's local nodal point 2. That's local nodal point 3. That's local nodal point 4 for any one of these elements. The local nodal point 1 corresponds to the global nodal point 5. 2 corresponds to 2. 3 here corresponds to 1 there, 4 here corresponds to 4 there. I say, therefore, that the node numbers of the element are 5, 2, 1, 4.

The element number 2, then, with this convention that I'm using, has nodal point 6, 3, 2, 5. Well, let's look here. We have here 6, 3, 2, 5. Also properties at 1.

Let's go on to element number 3. Here with this convention, once again, we have 8, 5, 4, 7. And now, however, the material property set number 2. Et cetera.

Now, once the program knows these nodal point numbers, it can figure out a connectivity array. And that is done using the ID array that the program has already established. It's being done in the following way. Remember that, once again, nodal point 1, 2, 3, 4, 5, 6, 7, 8, and 9 correspond to these columns? Now, we know from this information here that element 1 couples into nodal point 5, 2, 1, 4. Let's keep that in mind, now, 5, 2, 1, 4, and circle here 5, 2, 1, and 4, where this one here is the first local nodal point, this is the second local nodal point, this is the third local nodal point, and that is the fourth local nodal point. That's important.

Well, if we look at that information, then, recognizing that the local nodal point 1 corresponds to the global nodal point 5, we have to use these two equation numbers corresponding to the first nodal points. Let's put a little picture up here.

You see what I'm saying here is the following. If this is local nodal point 1, that's 2, that's 3, and that's 4, and if I know that this local nodal point with degrees of freedom, let's call them little u and little v, corresponds really to the global point 5 with degrees of freedom 3 and 4, then this u must correspond to 3 here, and that v must correspond to that 4 here. In other words, our connectivity array which we will be using is established as follows. For our compact matrix, we have 8 rows and columns, and for our actual matrix that we want to add into the structure matrix, we notice that what we want to do is take the first row and column here, and add it into the third row and column of the structure matrix.

The first one here, u here, corresponds to 3 here. The second one here, which is the v, corresponds to the 4 here. In other words, the first degree of freedom of the element corresponds to the third in the structure. The second degree of freedom of the element corresponds to the fourth in the structure. And that is shown right here. The first one corresponds to the third in the structure. The second one in the compact element stiffness matrix corresponds to the fourth degree of freedom in the structure.

Now if we go onto the second local nodal point, we see zeros here, and these zeros go directly into the connectivity array. The third one has also zeros here, and the fourth one has 1 and 2. So our connectivity array, then, proceeding in the same way, is shown as given here. These are the degrees of freedom at the first nodal point, this first local nodal point. These are the degrees of freedom at the second local nodal point. These are the degrees of freedom at the third local nodal point, and these are the degrees of freedom at the fourth local nodal point.

In other words, what I'm saying here really is the following. You see this here is the third degree of freedom in the structure. This is the fourth degree of freedom in the structure. There is no degree of freedom here because we have a support there. That is that zero and that zero. There is no degree of freedom here because we have a support here. 0, 0. At this node, we are talking about the first degree of freedom here and the second degree of freedom here of this structure, and that is given here. So if I have established the 8-by-8 stiffness matrix of the element, I can

directly use that 8-by-8 stiffness matrix with this connectivity array, and assemble the appropriate contributions from that 8-by-8 matrix into the global structure stiffness matrix.

The same process that is applied to all of the other elements-- let's look at one more element here. And element 2, as we pointed out earlier, has nodal point 6, 3, 2, 5. Well, what we then have to do is look at our ID array. 6, 3, 2, 5. And what we are seeing immediately is that we have a 5, 6 here and a 3, 4 here. So the first two entries in the element array should be 5, 6, and the last two entries should be 3, 4, because we have all these zeros there. And indeed, if we look at our LM vector here, that's what we obtain.

Similarly, we proceed for the other elements. This is the connectivity array for element 2, for element 3, and for element 4. We can use that now to assemble the element stiffness matrices into the global structural stiffness matrix. Of course, the program figures these out automatically from the ID array and from you having put into the program the nodal points of each element.

Let us look now at how do we actually deal with the stiffness matrix? Well, if we look at a typical stiffness matrix-- this might be a typical one here-- we have this pattern. Of course the matrix is symmetric, and what we have are some non-zero elements clustered to the diagonal, and some 0 elements out there.

It is convenient at this point to define a half bandwidth of the stiffness matrix. That half bandwidth is defined in the following way. We ignore, first of all, the diagonal element, and then we identify the furthest off diagonal element from that diagonal element. The furthest one from the diagonal element defines the half bandwidth of the matrix. mK is the half bandwidth of the matrix. In some literature, we also refer to the half bandwidth of the matrix as mK plus 1. But then remember that the total bandwidth off the matrix is simply 2 mK plus 1, because the diagonal element only occurs once.

Another way of looking at the definition of the half bandwidth is as follows. If we go from the diagonal up in each column, we will come to an element above which only

zeros are. Like in this case, for example, there are only zeros above K 4, 5. And we do the same for each all of the columns, and we define the last non-zero element, this one here, above which all elements are zero, as the skyline. So the skyline is defined as shown here. This is the skyline of the matrix.

The half bandwidth, then, is equal to the maximum column height minus 1. In this particular case, you see the maximum column height is 1, 2, 3, 4, which we have here also-- 1, 2, 3, 4. We subtract 1 and we get mK equal to 3. So this is the pattern that we observe in an actual finite element analysis. What we would like to achieve is that the half bandwidth is as small as possible, because then we know that our numerical operations is a solution of KU equals R are small.

Well, the actual storage that we're using, however, is a little different. Namely, it is effective to store the total information as a one-dimensional array. And the storage, then, is carried out as follows. Notice that in a one-dimensional array, I am using now the following convention. This here is the first element in the one-dimensional array. The second element is this k22, which is a2. The third element is k12. The fourth element is k33, and so on. In other words, the A vector here, being a one-dimensional vector, stores all of the information here in a one-dimensional order, where we simply go from the diagonal upwards to store all of the elements. So a6 is the diagonal element that is the fourth column here, a6 corresponds to k44, and a7 then stores this one, a8 stores that one, a9 stores that one.

Notice that we do carry along these zeros. And the reason for it is that when we do perform our solution of the equation, in general, but not always, in general, this zero becomes a non-zero element, and therefore it is effective to just carry it along, because we will have to later on store some non-zero information in it.

Notice that this is the total array, and we have altogether, in this particular case, 21 entries here, in addition to storing the stiffness matrix in this one-dimensional array. However, we also have to have an identification array that tells us which elements in this one-dimensional array are diagonal elements. Of course, here from this picture, we see immediately that these are the diagonal elements of the stiffness matrix.

However, imagine that we simply store A as a one-dimensional array [? alone, ?] then we would have to know that a2 corresponds to the second diagonal element, a6 corresponds to the fourth diagonal element, and so on.

And that is done by as a MAXA array. You see, MAXA here stores the addresses of the diagonal elements. Having MAXA, the array MAXA, and having the one-dimensional array that contains all the elements of the stiffness matrix strung out in one dimension, we can access any element in here during the solution of the equation as is required.

Let me mention one more point-- that the length of this array here is not equal to n, but it's n plus 1, because this last element here gives us a diagonal element that would occur out here. So 22 really is this diagonal element, which we really don't have, of course. But when we subtract 1 from it, we get this element here, or the address of this element here, and then we know how long this column is. We have to know how long this column is. And for that reason, we need this last entry here.

In practice, of course, what we might find is that we cannot store this total matrix in [? core, ?] because it's just too big. There are just too many elements in the matrix. If we talk about a 5,000 degree of freedom model with a bandwidth of 1,000, then we have 5 million elements in that matrix. And even on the very large-scale computers now, you have to somehow block that information in order to be able to deal with it.

And that we do in the following way. If a certain amount of high-speed storage is available, then based on that high-speed storage available, we simply block the total stiffness matrix. Once again, shown here. Here is the skyline, or the column heights are given by that skyline. We block it as shown here. This is here block 1, this is here block 2, this is here block 3, and that is block 4. In fact we will see, later on, when we talk about the solution of equations, it is necessary that we can keep two blocks at a time in the high-speed storage. In other words, if we have a total high-speed storage of 40,000 elements, then 20,000 elements would be this, and 20,000 elements would be this. Or in practice, of course, these might be slightly less than

20,000, and these might also be slightly less than 20,000. But we have to be able to keep two blocks in core, and that is our criteria to determine the block size.

Of course, the computer program, once again, does all that automatically. It just knows how much storage there is available, because you have a specified amount of storage that is available, and then it allocates the appropriate amount for each block and calculates the block size, the column heights in each block, and so on.

Once again, I'm showing here zero elements in a column. When we actually decompose the matrix as we do in Gauss elimination into an LDA transport form-- I will be discussing that later-- we fill up these zero elements in general, and that is the reason why we carry them along in the solution phase.

Let me make a few remarks on the bandwidth, the use of an effective bandwidth. Here we have a finite element model of a cantilever, a plane stress finite element model of a cantilever. At each node, we have two degrees of freedom, just as in this earlier model that we considered. Notice these are now 8-node elements, isoparametric elements that I will be discussing later. We have constraint, of course, at this end, all degrees of freedom, because the cantilever is 6 there. In this particular layout, notice we have used the following element or nodal point numbering. 1, 2, 3, 4, 5, 6 up to 13, then from 14 to 20, then from 21 to 33. Well, this then means that our bandwidth here, or half-bandwidth, I should rather say, and now let's simply include the diagonal. It doesn't make much difference. We are talking, in practical analysis, about a bandwidth, or half bandwidth, of 300. So having 300 or 301, really, in a practical analysis, makes very little difference. Our half bandwidth, including the diagonal element here, would be 46. How would we obtain that? Well, if we look at the coupling between nodal points that is due to the element stiffnesses, we recognize that for a typical element, let's look at this element, we have a maximum nodal 25 here, a minimum nodal point 3. Now, 25 minus 3 is equal to 22.

However, we now have to add 1 on because all of the diagonal part, and that together, then, gives 23. This 1 I'm adding on because of the diagonal element--

and if you where to look at the stiffness matrix in its assembled form-- of this element in its assembled form, you would see that this part here is to be added on, because you had also the diagonal contribution.

So the maximum difference between the nodal points plus 1 gives us 23. Now, that 23 is a maximum if 1 degree of freedom were at 1 nodal point. However, we take that 23 and we have to multiply it by 2, because we have 2 degrees of freedom per nodal point, and that then gives us 46. So the half bandwidth, including the diagonal, is 46.

Well, that is a very large bandwidth, in this particular case. And let us now try to rearrange the nodal point numbering to come up with a smaller bandwidth. And the better nodal point numbering here is shown in layout. 1, 2, 3, 4, 5, 6, 7, 8, and so on, down this way. If we now look at the same element again, the maximum difference in nodal points is 7. We add 1 again to get 8, and we have 2 degrees of freedom per nodal point. That gives us a half bandwidth of 16. And therefore, we have reduced the bandwidth by almost a factor of 3.

Now this means that the solution effort will go down, in the equation solution phase, will go down by a factor of 9. Because he would see that our solution effort is proportional to the half bandwidth squared. So if there's a factor of 3 here, the solution will actually be reduced by a factor of 9. Therefore, it is very important to use minimum bandwidth in the finite element mesh, or rather number the nodal point in such a way as to obtain a minimum half bandwidth.

If we actually deal with column solvers, we will see that in some cases we quite don't want to have the minimum bandwidth because we have a column solver, about but these are details that we will be addressing in a little bit. In general, we want to have really a minimum bandwidth.

Let us look now at the overall solution phase once more. This is a solution phase in the computer program STAP, which is in the textbook that you are using in this course. You might look at the description of this computer program in more detail, but the overall solution set is as shown here. We start the program, and we have to

read a nodal point data, which involves coordinates, boundary conditions, and we established the equation numbers in an ID array. I'll discuss that with you. We then would calculate and store load vectors for our load cases. We write these on tape. If we have a typical analysis, three, four load cases, we calculate all of these load vectors and store them on tape. Then we continue to read, generate, and store element data the way I've been discussing this with you, and in the actual analysis, we loop over element groups.

By that I mean that the total elements are subdivided into element groups. This is an effective concept because in an actual structural analysis, we deal with plane stress elements, beam elements, truss elements, three-dimensional shell elements, and so on. And it is effective to just group all of these elements together into specific groups. In other words, put the shell elements together in one group, the beam elements together in another group, and so on. In fact, this is a very effective way of proceeding also in non-linear analysis, in a more complicated analysis. If you're familiar a little bit with ADINA, what we do there is that we are grouping elements not just together according to the kinds of kinematics that they're representing, but also the material models they are containing, the kind of descriptions we want to use for non-linearities, et cetera.

Well, having then generated and stored the element data, we can read it and calculate the element stiffness matrices, and assemble these in a global structural stiffness matrix. Here we loop over all element groups once again. And this means generally we are storing this element data on tape, and then we are reading it and going over all of the element group data, as shown here.

Then next we can factorize the stiffness matrix. This is a basic step in the Gauss elimination procedure, as I will be discussing with you later. And now we loop over the load vectors for each load case, read the load vector, and calculate the nodal point displacement. We then read the element group data and calculate the element stresses. Print out the element stresses, and if there is another load case to be considered, we go back from here into there.

So the LDA transport factorization of the stiffness matrix which was outside this loop is only done once. A forward reduction and back substitution of the load vector is done for each load case, as shown here. I will be discussing those aspects in more detail later.

Let us now look at some typical effective elements. We have been discussing so far some simple elements, just to expose you to the basic concepts that are being used. And for that reason, we discussed some very simple elements in the earlier lectures. What I want to do now in the next lectures is to discuss this you modern, effective isoparametric elements. These are also the elements that we are using in the ADINA computer program.

Here we have a one-dimensional truss element which can be used as a cable element. As a simplest element, it would be 2-noded element. We would only assign this node and that node. And then it would be, of course, a simple 2-noded truss, a very common element, 2-node truss.

However, as shown here, we can also have a third note, which is this one. And we can have a fourth note. In other words, what we will be dealing with, really, are variable number node elements, which have two nodes, three nodes, or four nodes for the one-dimensional truss element. A ring element is obtained from the truss element in this very simple way. It's really an axis-symmetric truss, you might call it an axis-symmetric truss, with only one degree of freedom. It has only stiffness in its circumferential direction.

So this is a truss element. And the basic concept that we are using is that we can have a variable number of nodes. Similarly for this element, this is a plane stress, plane strain, or axis-symmetric element. Notice that the only displacement that we are talking about in this particular case are two displacements, u and v, or in ADINA, we call this the V and W degree of freedom, corresponding to the y- and z-axis. The z-axis is, in an axis-symmetric case, the axis of revolution. And with this degree of freedom, the kinematics of the element are defined when we talk about a specific number of nodes. And then the element can be used for plane stress, plane strain,

and axis-symmetric analysis, depending on which stress-strain law you're using, and which strain components and stress components you're dealing with.

Here, too, we can use the element simply with 4 nodes, so we would have this 4-noded element. We can also add another note, make it a 5-noded element. We can add this node, we have a 6-noded element, now a 7-noded element, and finally an 8-noded element. The variable number node concept is a very effective way of formulating elements, because the same basic sub-protein basic element can be used for all sorts of different applications.

The same holds for three-dimensional analysis. There's also a bit of a number node 3-dimensional element where we have a basic number of nodes being 8, and then we would have this brick element here-- let me just quickly sketch it out. This is the basic brick element. And now we can simply add additional nodes in to obtain higher-order elements. And the 21-noded element here is a very effective element for general three-dimensional analysis. However, in some other applications, even the 8-noded element is quite effective, effectiveness, of course, being always measured by the computational effort involved in dealing with such an element, and the accuracy that the element can give us in actual analysis. So this being one element, again, that can contain from 8 up to 21 notes, being a variable number node element, and being very effectively formulated using the isoparametric concept that I will be discussing in the next lecture. Notice that in that concept, we can have curved element sides, as shown here, for the higher order elements.

Finally there is, of course, our beam element. The beam element, which I'm sure you are quite familiar with, a simple 2-noded element which might not be referred to as a finite element, but on the other side, we might also call it a finite element. Originally, it was not called a finite element, but now if we look at the basic interpolation procedures that we are employing, we may very well refer to it at a finite element.

Then the shell element that I also will be discussing. Here we have a basic shell element that is shown here, having 16 nodes. At each nodal point, we have now 5

degrees of freedom. The 5 degrees of freedom being 3 translations, as shown here. And in addition, 2 rotations, these being the 2 rotations. The 16-node element is the highest order element that I can recommend for practical analysis. It's a quite expensive element, but very accurate, because it admits curvature into all directions. So for curvatures, it can be a very effective element to use.

Notice that at each node, we have 5 degrees of freedom where these translations are defined in the xyz global coordinate system, whereas the rotations are not necessarily aligned with the x- and y-axes. I will be discussing that later on in detail. But then with 16 notes and 5 degrees of freedom, we are talking about 80 degrees of freedom altogether, a very considerable number for a single element. In practice, therefore, we want to possibly use less nodes on the element, and an effective element that directly is obtained from this one is simply the 9-noded element. I will be discussing that element. And of course, this element and other elements from this one are directly obtained by simply assigning certain nodes to the element. We use here again the variable number node concept as we use it for the truss element, the 2-dimensional, and the 3-dimensional element.

Another important feature that I like to also just mention to you is the fact that this element can be used as a transition element. You might have a shell here, and you might have a solid here to be idealized, and there's a transition region. The solid element here would have, in general, 3 degrees of freedom at a node. The shell element here has 5 degrees of freedom at this node, 5 degrees of freedom at this node, and here we have a transition element that has shell degrees of freedom at these nodes, and solid degrees of freedom at these top and bottom nodes. Here altogether at this line, 6 degrees of freedom, 3 here, 3 there, whereas at this line, 5 degrees of freedom.

So this is an effective way of modeling a transition region between a shell and a solid in a compatible way. In other words, preserving full compatibility between the elements, and not using any constraint equations. I will be also talking further about this element later.

In this lecture, then, what I wanted to discuss with you were some basic concepts regarding the formulation of the finite element methods, in particular regarding the implementation of the finite element method. In other words, how do we actually implement what we formulated in the earlier lectures? Some of these concepts are very important concepts when it comes to actual practical implementation of the finite element method, particularly the one that I discussed regarding the connectivity arrays that are formulated, and so on.

This is all I wanted to say. Thank you for your attention.