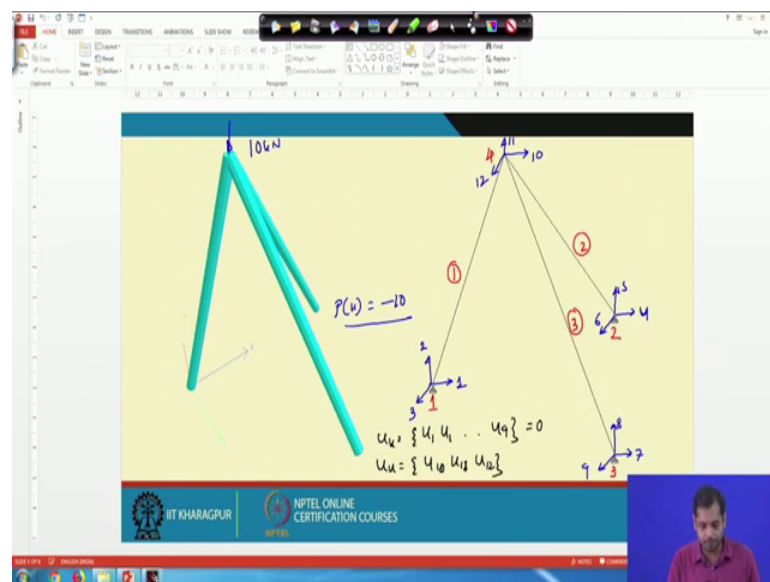


Matrix Method of Structural Analysis
Prof. Amit Shaw
Department of Civil Engineering
Indian Institute of Technology, Kharagpur

Lecture - 34
Analysis of 3D Truss (Contd.)

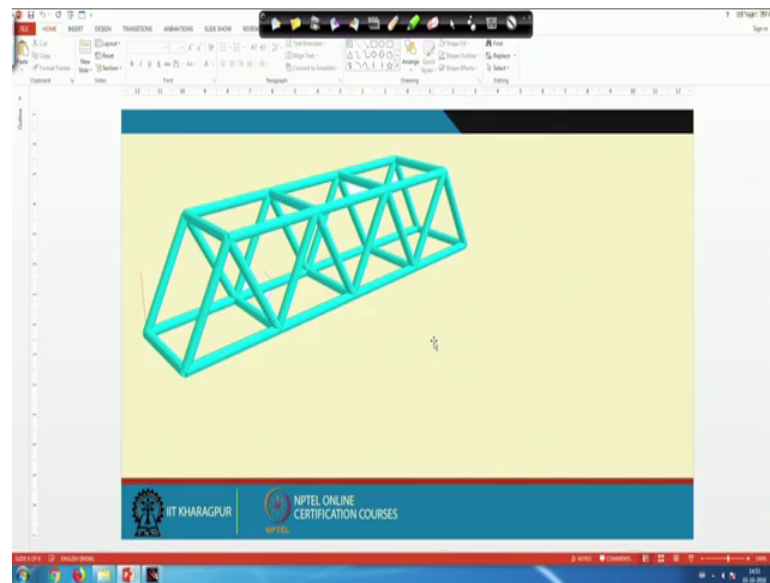
Hello everyone. This is the 4th lecture of this week. The last class we discussed the formulation of analysis for 3-D truss. Today we will translate that formulation into a code and demonstrate that code through 2 examples. The first example will consider is this one, which is relatively simpler in terms of number of members and number of nodes.

(Refer Slide Time: 00:37)



And the second one is this example, which has more number of nodes and members.

(Refer Slide Time: 00:39)



So, let us start with first this one. So, it has so, first thing when we have a problem, the first thing we have to do is we have to give the numbering of the nodes. Suppose this is node number 1, this is node number 1. This is node number 2, this is node number 3 and this is node number 4.

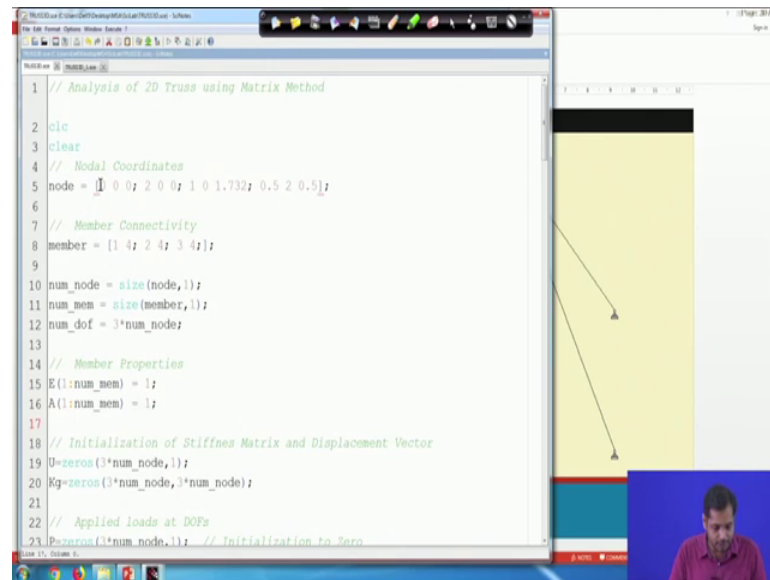
And say and members, say this is member number 1, member number 2 and member number 3. So, member one is connected between node 1 and 4, member 2 is between 2 and 4 member 3 is between 3 and 4. Now all these joints, joint 1, joint 2 and joint 3 their hinge joint so, essentially displacement at these joints are 0. Now the degrees of freedom at since, is a space truss at every point we have 3 degrees of freedom. And suppose these 3 degrees of freedoms are this is your 1 and this is this is 2 and this is 3.

So, this is u_1 , this is u_2 , this is u_3 . Similarly, we have here this is 4 this is 5 and this is 6, and this is 7, this is 8 and this is 9. And similarly we have this is 10 and this is 11, and then this one is 12. So, it has total 12 degrees of freedom. Out of these total 12 degrees of freedom, degrees of freedom 1 to 9 they all are hinge joint so, they are all constrained so, u_{known} is equal to u_{known} is equal to u_1 , u_2 to u_9 and this is equal to 0. Displacement at these degrees of freedom are 0, and u_{unknown} for which we have to solve this that is only u_{11} , u_{12} and u_{10} , u_{11} and u_{12} .

Now, nodal load vector, suppose we have a load of, here we have a load concentrated load of 10 kilo Newton. So, p which is in the direction of 11 degrees of freedom, then p

11 will be minus 10. And all other p will be 0 so, for node number 4. So now, let us give this information to the code and then see what is the result. And in the process will also discuss the various steps in the code.

(Refer Slide Time: 03:15)

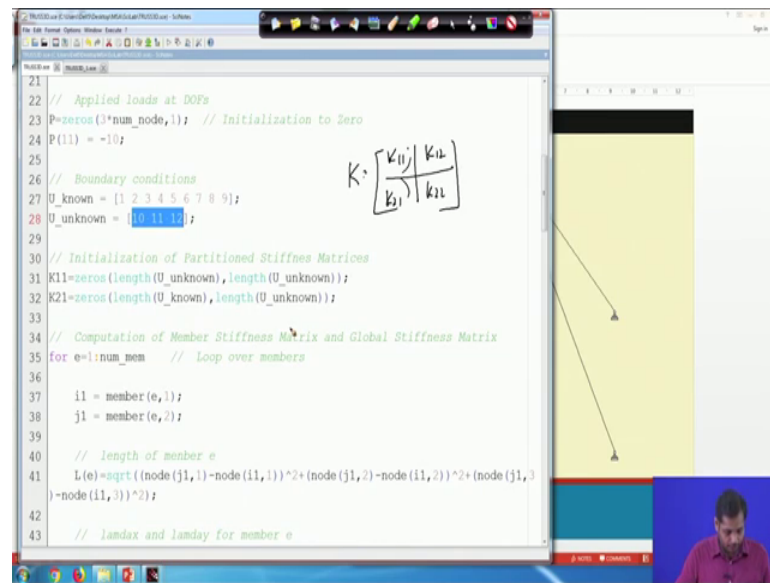


```
1 // Analysis of 2D Truss using Matrix Method
2 clc
3 clear
4 // Nodal Coordinates
5 node = [1 0 0; 2 0 0; 1 0 1.732; 0.5 2 0.5];
6
7 // Member Connectivity
8 member = [1 4; 2 4; 3 4];
9
10 num_node = size(node,1);
11 num_mem = size(member,1);
12 num_dof = 3*num_node;
13
14 // Member Properties
15 E(1:num_mem) = 1;
16 A(1:num_mem) = 1;
17
18 // Initialization of Stiffness Matrix and Displacement Vector
19 U=zeros(3*num_node,1);
20 Kg=zeros(3*num_node,3*num_node);
21
22 // Applied loads at DOFs
23 P=zeros(3*num_node,1); // Initialization to Zero
```

So, to start with this is the code. Now, these are the nodal coordinates of different nodes. 3 nodes and this is the member connectivity, all these members. Now total degrees of freedom for this is 3 into n, 3 degrees of freedom per node so, total 3 n degrees of freedom. This is member property, now you can have different these look at please note that e and they are they are essentially vector.

So, for demonstration purpose, let us assume there one, but you can you can have different values depending on the member property. And we have to change the corresponding element in that vector. So, this is the initialization of the displacement vector and the stiffness matrix; the stiffness matrix, size of the stiffness matrix will be 3 n by 3 n.

(Refer Slide Time: 04:03)



```
21
22 // Applied loads at DOFs
23 P=zeros(3*num_node,1); // Initialization to Zero
24 P(11) = -10;
25
26 // Boundary conditions
27 U_known = [1 2 3 4 5 6 7 8 9];
28 U_unknown = [10 11 12];
29
30 // Initialization of Partitioned Stiffness Matrices
31 K11=zeros(length(U_unknown),length(U_unknown));
32 K21=zeros(length(U_known),length(U_unknown));
33
34 // Computation of Member Stiffness Matrix and Global Stiffness Matrix
35 for e=1:num_mem // Loop over members
36
37     i1 = member(e,1);
38     j1 = member(e,2);
39
40     // length of member e
41     L(e)=sqrt((node(j1,1)-node(i1,1))^2+(node(j1,2)-node(i1,2))^2+(node(j1,3)-node(i1,3))^2);
42
43     // landax and landay for member e
```

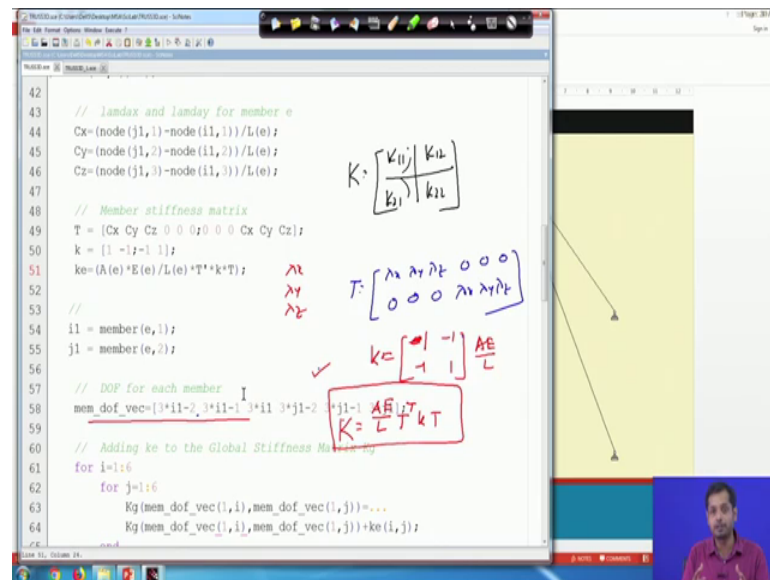
Handwritten diagram of a partitioned stiffness matrix K :

$$K = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}$$

Now, this is the applied load, this is initially, we substitute all the values are 0. And then only the force we have here is only in the 11th degrees of freedom we have minus 10. So, this will be minus 10 so, p 11 will be minus 10.

Now, just now as I said the all known displacements are 1 to 9, they all are constrained through says essentially those values are 0, only unknown displacements are 10 11 and 12 we have to solve for this. And K_{11} , K_{22} are the same either partition stiffness matrix, if you recall we discussed that in the case of 2-D truss as well, the total stiffness is now partition into K_{11} K_{12} K_{21} and k_{22} right now K . So, this is k_{11} and this is K_{21} , K_{11} and K_{21} . Now, this is the computation of stiffness matrix, it starts from here.

(Refer Slide Time: 05:09)



So, this loop is over the number of element. First determine the length of a given member, once we know the coordinate. These are the C x C y, C y or the lambda x lambda y lambda y here lambda x lambda y and lambda z if you recall, this is how it is related to this is lambda x, this is lambda y and this is lambda z. Now once we have lambda x lambda y lambda z, next thing is the transformation matrix. The transformation was again if you recall it was the transformation was if t was lambda x, lambda x, lambda y, lambda z then 0 0 0, then 0 0 0, lambda x, lambda y and lambda z, right.

This is the transformation matrix and this is that matrix; this is this matrix right, this one is this transformation matrix. This is the transformation matrix. And this small k is the local coordinate system, if you recall local coordinate system all local coordinate system is 1 minus 1 minus 1 1, then AE by L that is the stiffness matrix of a member and this is this stiffness matrix with respect to local coordinate system. AE by L is not multiply here it is multiplied at the end here.

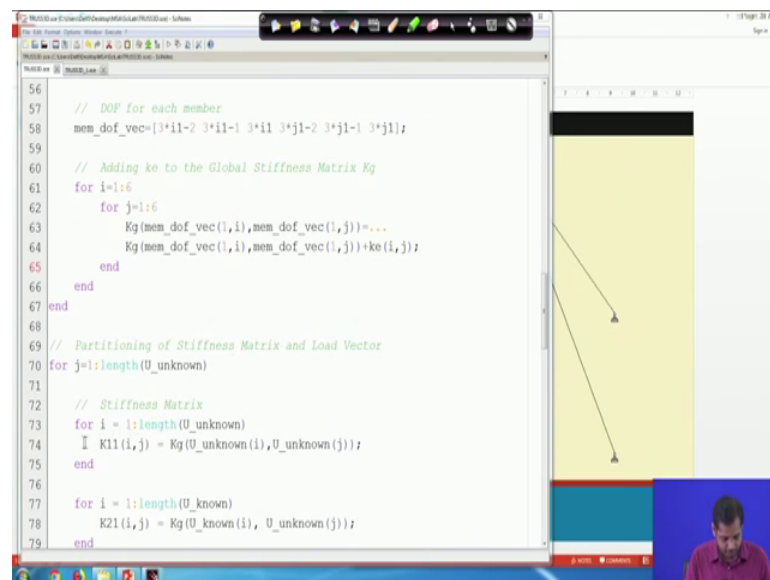
Now if you recall, then the global stiffness matrix, global not global stiffness matrix was means stiffness matrix with respect to global coordinate that was AE by L, AE by L and then T transpose k into T, right. This is the expression for stiffness matrix and these operation is performed here.

So, this is the stiffness matrix or member stiffness matrix with respect to global coordinate system. Now once we have member stiffness matrix, then we have to along

with the member stiffness matrix we know the connectivity of the member and also know the; therefore, we know and we also know the definition of different degrees of freedom. So, we know that a particular elements in the global; in the stiffness matrix member stiffness matrix corresponding to which degrees of freedom, right.

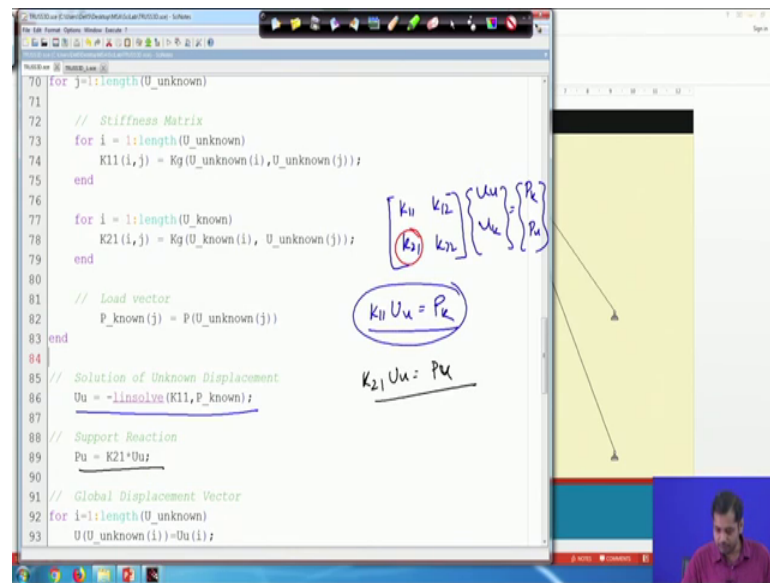
So, then based on that we have to populate the global stiffness matrix. Initialize, we initialize the global stiffness matrix at 0, and then element wise for every element we take the for every member will construct the stiffness matrix, and in that stiffness matrix we compare element by element by and then put that in the substitute in any subsequent places, in the stiffness matrix, global stiffness matrix.

(Refer Slide Time: 07:58)



And this is done here. Member stiffness matrix is 6 by 6 here if you recall. So, this is the assembling of the stiffness matrix. The resultant the kg will be the global stiffness matrix. Now once we have the global stiffness matrix, next is we have to partition the stiffness matrix, partitioning of the stiffness matrix is done here, K 1 1 is this and K 2 1 is this and then this is the load vector, correspondingly we have to partition the load vector as well.

(Refer Slide Time: 08:26)

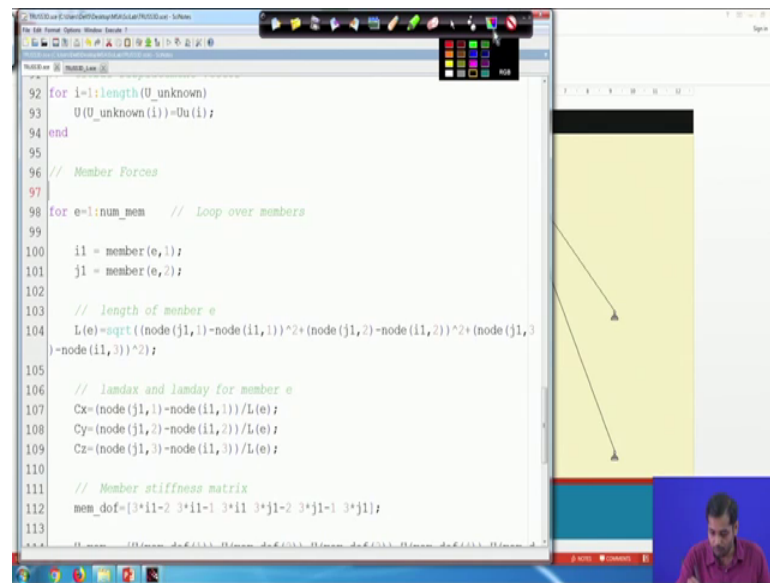


So, this is the corresponding load vector. And then what we have is, we have p if you recall, then we have K_{11} , K_{12} , K_{21} , K_{22} ; all K_{11} K_{22} they all are matrices, then this is equal to $U_{unknown}$, and U_{known} that is equal to P_{known} and $P_{unknown}$, right.

So, from this we will have this expression K_{11} into $U_{unknown}$ is equal to P_{known} because U_{known} is 0 that is why this contribution will be 0. So, if we solve this equation we get u_{11} . So, this is done in this expression. It is solved for U_u and then this u give you the solution of this unknown displacement. So, once we have the unknown displacement, next step is to we have to find out this support reaction. And the sub what will be the support reaction? Support reaction will be we have to consider the other part for the support reaction, for this part for the support reactions.

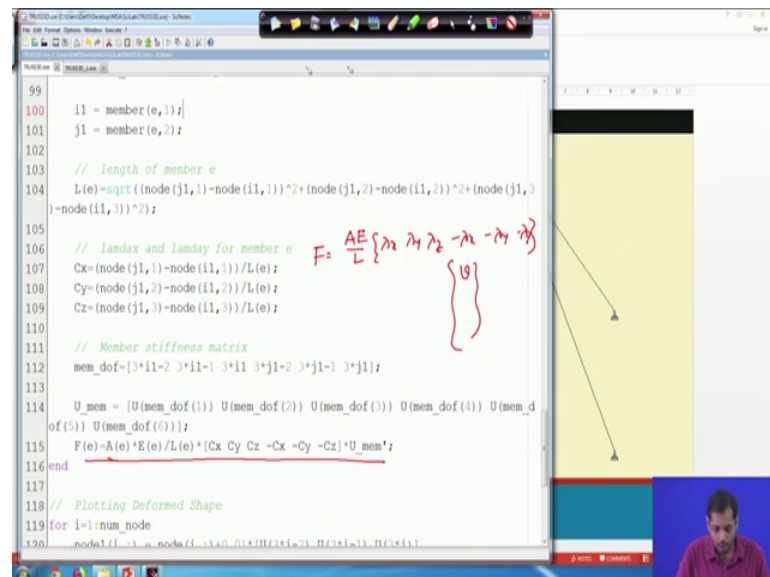
So, P support reaction will be if you recall then K_{21} into U_u is equal to P_u this gives you the support reaction right. And these support reaction this is been done here. So, once we have the support reactions and then the member forces, these is the member force.

(Refer Slide Time: 10:02)



This is the member force is calculated here, if you recall the member force the expression for member force for 3-D truss was; for 3-D truss was it was minus, in just 1 minute, let me come to that, yes. So, it was if you recall the member force for a given member.

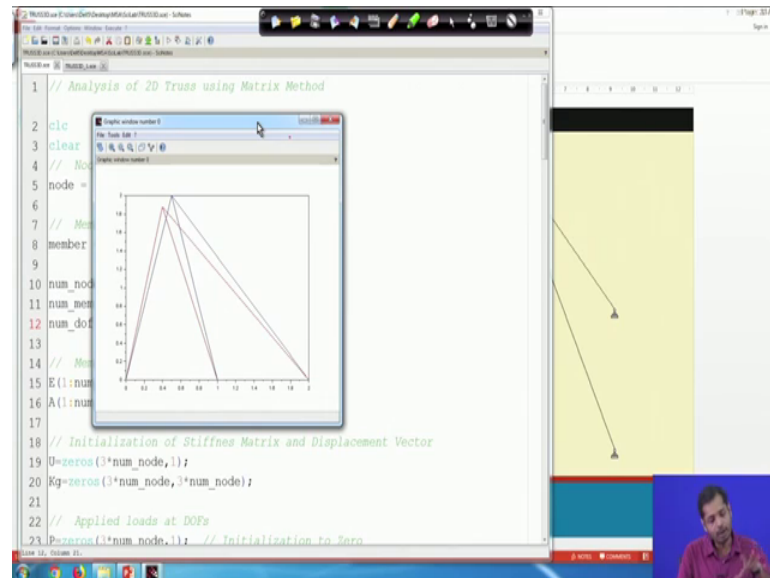
(Refer Slide Time: 10:27)



The member force is. F is equal to AE by L AE, AE by L, and then we have lambda x, lambda y, lambda z, then minus lambda x, minus lambda y, minus lambda z and then for that particular member, for that particular element what is the v.

So, if you from that we get the member force, and this is exactly performed here. So, this is this, now once we have the member stiffness member, member this then we all know we have all the information; which is required for the analysis. Support reactions displacement and the member forces. Now let us run it and then see what are the result coordinates and everything are given here.

(Refer Slide Time: 11:16)



So, run it; so these is the plot, this is not a 3-D plot, actually the structure is in 3-D, this is a 2-D projection of the plot. So, that is not important right now for us. So, let us see what are the values of displacement, first you see what is the displacement total U u.

(Refer Slide Time: 11:33)

The image shows a MATLAB script for the analysis of a 2D truss structure. The script is titled "Analysis of 2D Truss using Matrix Method". It defines the following variables and operations:

- Nodal Coordinates:** A 3x2 matrix `node` with values $\begin{bmatrix} 0 & 0 \\ 2 & 0 \\ 1 & 0 \end{bmatrix}$ and $\begin{bmatrix} 1.732 & 0.5 \\ 2 & 0.5 \end{bmatrix}$.
- Member Connectivity:** A 3x2 matrix `member` with values $\begin{bmatrix} 1 & 4 \\ 2 & 4 \\ 3 & 4 \end{bmatrix}$.
- Member Properties:** A 3x1 matrix `E` with values $\begin{bmatrix} 1.541493 \\ 1.048482 \\ 1.541493 \end{bmatrix}$ and a 3x1 matrix `A` with values $\begin{bmatrix} 1.782486 \\ 1.048482 \\ 1.048482 \end{bmatrix}$.
- Initialization:** Stiffness matrix `K` and displacement vector `U` are initialized as zeros of size $3 \times \text{num_node}$.
- Applied loads:** A vector `P` is initialized as zeros of size $3 \times \text{num_node}$.

The output window shows the calculated member forces for three members, with handwritten red numbers 1, 2, and 3 next to the force values:

- Member 1: -5.501493
- Member 2: -1.048482
- Member 3: -1.048482

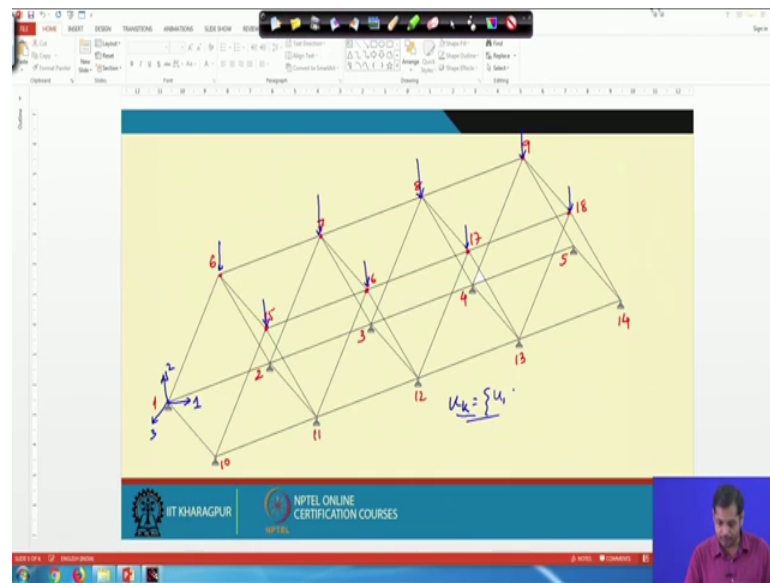
You see all the first u, first 9 U will be 0 because they are constrained, then we have rest of the rest of U u 10 11 and 12 these are the values of u 10 11 and 12. And then the support reactions, if you take the support reaction, these are the support reaction, the first 3 is for (Refer Time: 11:53) for join number 1, this is for join number 1, this is for join number 2 and this is for join number 3 in the corresponding reaction.

And similarly we have, we can calculate the member for member forces. Member forces will be this. So, this is 3 members, these are the members force. So, this is for member number 1, member number 1 the member number 2 and member number 3, right. Now you can, these truss since it is smaller comparatively you can do it manually. You do it manually and compare your results with these results. And then do other exercise like you change the Young's modulus, change the area of the area of particular member, and then do this exercise to get some idea about the structural behaviour.

Because you see if you in as in a truss, if some members are some members are having say, smaller dimensions; or some members are having smaller, lower Young's modulus, then what happens to the behaviour of the structure.

So, that exercise if you do you get that comprehension may come. So, this was for the first example. Let us do it the similar exercise for the second example. So, second example was this; this is the second example. And suppose and this is if I draw a line diagram line will be this.

(Refer Slide Time: 13:29)

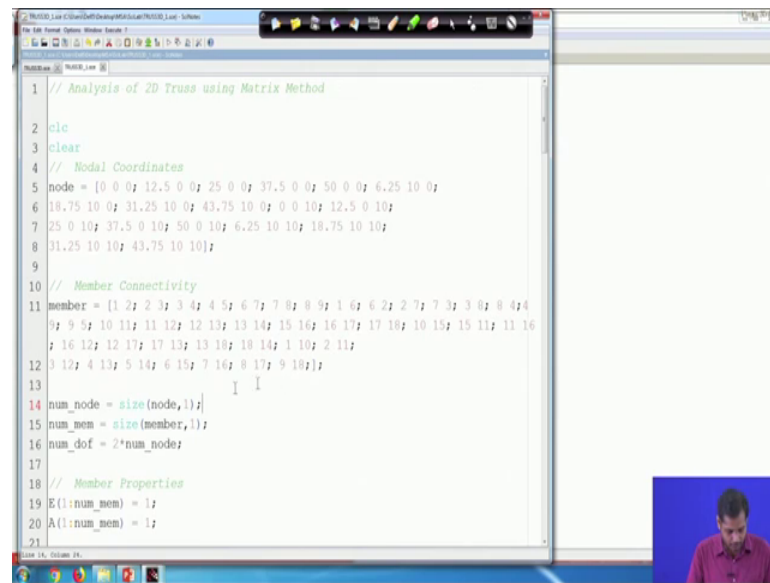


Now in this second example, let us first give the numbering of this a node numbering. This is node number 1, this is node number 2, this is node number 3, this is 4, this is 5, then this is 6, 7, 8 and 9. And similarly we have 10 11 12 13 and 14. And then this one is 15, then 16, 17 and 18 total 18 node these all are nodes.

Now, the coordinates of these nodes are written I will show you the coordinate. And suppose it is subjected to some load at all the places, we have some load. And these are all nodes, now for member number 1; for member number 1 the degrees of corresponding degrees of freedom will be; corresponding degrees of freedom will be this is 1, this is 2 and this is 3. Member number 2 similarly it is 3 4 5 6 and so on now. So, what another information from here is the since from node number 1 2 5 and 10 to 14 they are hinge joints.

So, corresponding degrees of freedom will be 0. So, u u known, u known will be it is member it is your degrees of freedom 1 2 all the degrees of freedom associated with these nodes, let us not write it here. So, see the code, now this is the code; this is we have another one. This is the code.

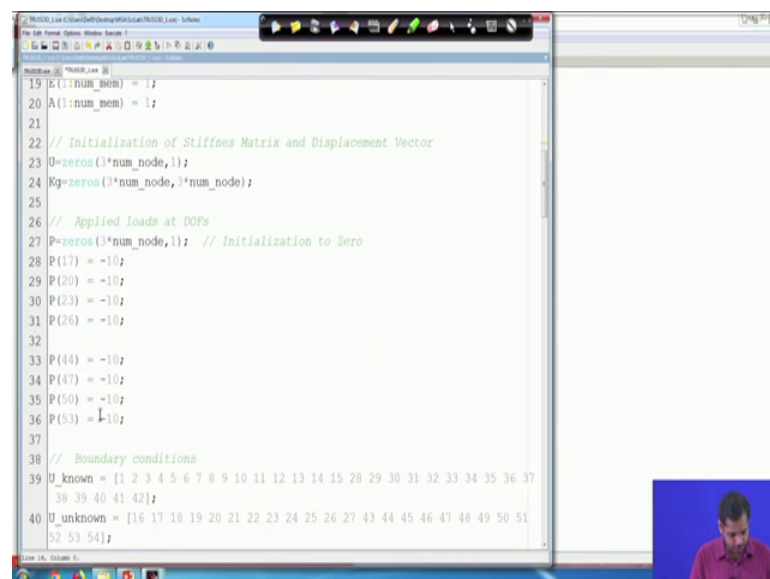
(Refer Slide Time: 15:20)



```
1 // Analysis of 2D Truss using Matrix Method
2
3 clear
4 // Nodal Coordinates
5 node = [0 0 0; 12.5 0 0; 25 0 0; 37.5 0 0; 50 0 0; 6.25 10 0;
6 18.75 10 0; 31.25 10 0; 43.75 10 0; 0 0 10; 12.5 0 10;
7 25 0 10; 37.5 0 10; 50 0 10; 6.25 10 10; 18.75 10 10;
8 31.25 10 10; 43.75 10 10];
9
10 // Member Connectivity
11 member = [1 2; 2 3; 3 4; 4 5; 6 7; 7 8; 8 9; 1 6; 6 2; 2 7; 7 3; 3 8; 8 4;
12 4 9; 9 5; 10 11; 11 12; 12 13; 13 14; 15 16; 16 17; 17 18; 10 15; 15 11; 11 16;
13 16 12; 12 17; 17 13; 13 18; 18 14; 1 10; 2 11;
14 3 12; 4 13; 5 14; 6 15; 7 16; 8 17; 9 18];
15
16 num_node = size(node,1);
17 num_mem = size(member,1);
18 num_dof = 2*num_node;
19
20 // Member Properties
21 E(1:num_mem) = 1;
22 A(1:num_mem) = 1;
```

This is the nodal coordinates given, and corresponding connectivity of these nodes are given, connectivity of the elements are given. This should be 3, 3 into number of nodes the degrees of freedoms as we discussed.

(Refer Slide Time: 15:38)

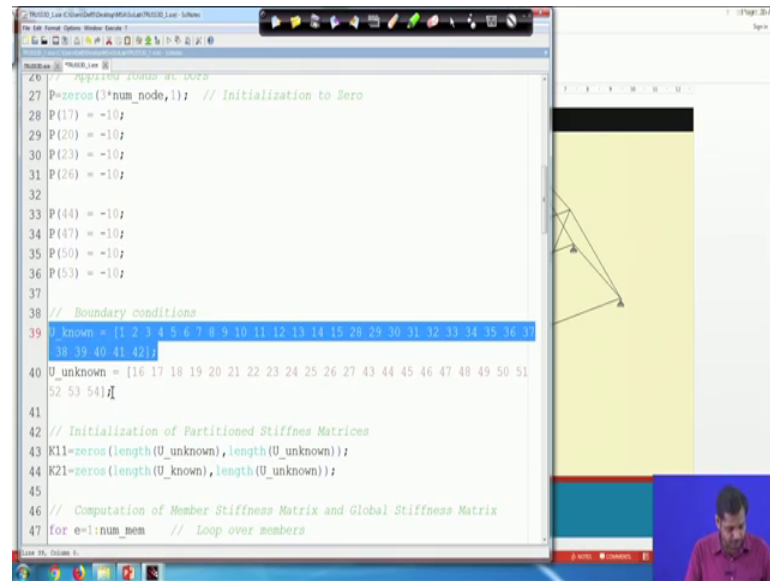


```
19 E(1:num_mem) = 1;
20 A(1:num_mem) = 1;
21
22 // Initialization of Stiffness Matrix and Displacement Vector
23 U=zeros(3*num_node,1);
24 Kg=zeros(3*num_node,3*num_node);
25
26 // Applied loads at DOFs
27 P=zeros(3*num_node,1); // Initialization to Zero
28 P(17) = -10;
29 P(20) = -10;
30 P(23) = -10;
31 P(26) = -10;
32
33 P(44) = -10;
34 P(47) = -10;
35 P(50) = -10;
36 P(53) = -10;
37
38 // Boundary conditions
39 U_known = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
40 38 39 40 41 42];
41 U_unknown = [16 17 18 19 20 21 22 23 24 25 26 27 43 44 45 46 47 48 49 50 51
42 52 53 54];
```

Now, this is you see, this is the load on the top on these nodes, these are the loads.

So, if we this is corresponding to the it is degrees of freedom along the y axis. So, we have to those values, those values will be only non-zero and rest of the values will be 0. So, this is the nodal vector.

(Refer Slide Time: 16:07)

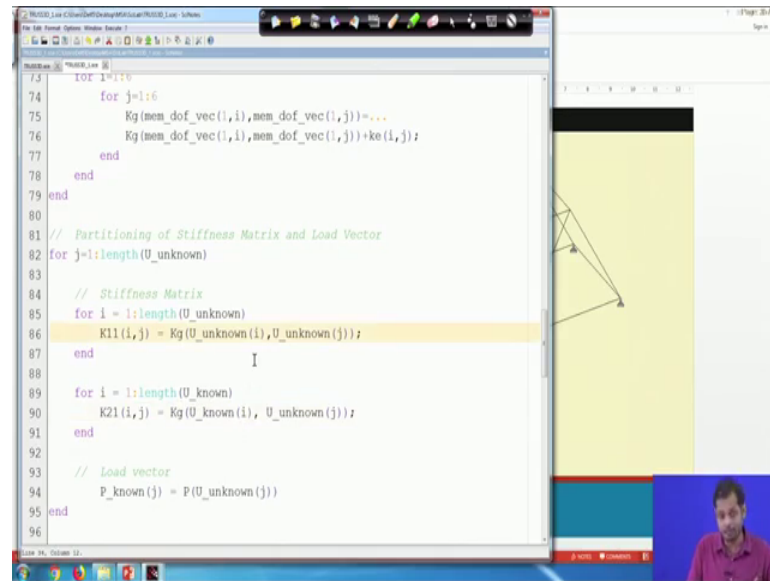


```
26 % Apply the loads at nodes
27 P=zeros(3*num_node,1); // Initialization to Zero
28 P(17) = -10;
29 P(20) = -10;
30 P(23) = -10;
31 P(26) = -10;
32
33 P(44) = -10;
34 P(47) = -10;
35 P(50) = -10;
36 P(53) = -10;
37
38 // Boundary conditions
39 U_known = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 20 29 30 31 32 33 34 35 36 37
38 39 40 41 42];
40 U_unknown = [16 17 18 19 20 21 22 23 24 25 26 27 43 44 45 46 47 48 49 50 51
52 53 54];
41
42 // Initialization of Partitioned Stiffness Matrices
43 K11=zeros(length(U_unknown),length(U_unknown));
44 K21=zeros(length(U_known),length(U_unknown));
45
46 // Computation of Member Stiffness Matrix and Global Stiffness Matrix
47 for e=1:num_mem // Loop over members
```

Now, the boundary condition, this is known degrees of freedom, again this is unknown degrees of freedom, we have discussed this. And finally, the same procedure, it is also the same procedure and then finally, we have the k_1 and k_2 . So, only thing the rest of these, all these steps are general only thing that that we have to give input is the information about the configuration of the structure, information about the boundary and information about the load.

Now, you see the purpose of this code for instance if you use any software. That software really you draw this and you apply a you apply the load using graphical interface; which is more, which is where you do not have to really give all this information in writing, you can you can do that through a graphical interface. But here the purpose is; here the purpose is to demonstrate the fact that these, how easy the method is to translate into getting translated into a composition code.

(Refer Slide Time: 17:08)

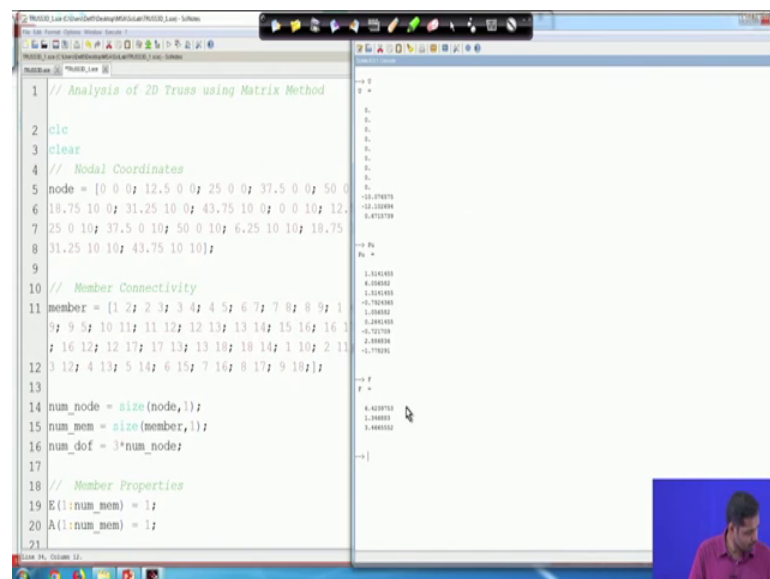


```
74 for j=1:6
75     Kg(mem_dof_vec(i,i),mem_dof_vec(i,j))=...
76     Kg(mem_dof_vec(i,i),mem_dof_vec(i,j))+ke(i,j);
77 end
78 end
79 end
80
81 // Partitioning of Stiffness Matrix and Load Vector
82 for j=1:length(U_unknown)
83     // Stiffness Matrix
84     for i = 1:length(U_unknown)
85         K11(i,j) = Kg(U_unknown(i),U_unknown(j));
86     end
87     I
88     for i = 1:length(U_known)
89         K21(i,j) = Kg(U_known(i), U_unknown(j));
90     end
91 end
92
93 // Load vector
94 P_known(j) = P(U_unknown(j))
95 end
96
```

The image shows a MATLAB script in a window titled 'MATLAB_1.m'. The script is part of a larger program for a 2D truss analysis. It includes a loop for assembling the global stiffness matrix K_{11} for unknown degrees of freedom. A comment indicates the partitioning of the stiffness matrix and load vector. The script also defines the load vector P_{known} . In the background, a diagram of a truss structure is visible, and a small video inset shows a person speaking.

Now, so if I solve it then if I solve it this.

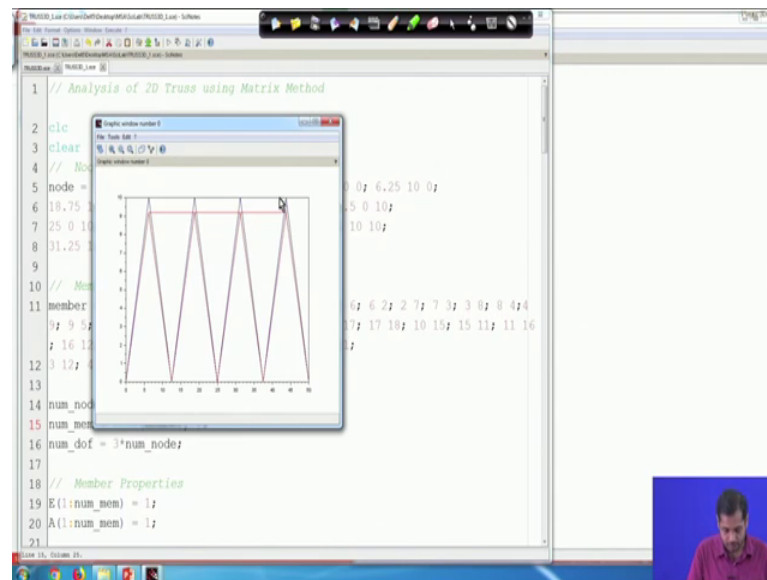
(Refer Slide Time: 17:17)



```
1 // Analysis of 2D Truss using Matrix Method
2
3 clear
4 // Nodal Coordinates
5 node = [0 0 0; 12.5 0 0; 25 0 0; 37.5 0 0; 50 0
6 18.75 10 0; 31.25 10 0; 43.75 10 0; 0 0 10; 12.
7 25 0 10; 37.5 0 10; 50 0 10; 6.25 10 10; 18.75
8 31.25 10 10; 43.75 10 10];
9
10 // Member Connectivity
11 member = [1 2; 2 3; 3 4; 4 5; 6 7; 7 8; 8 9; 1
12 9; 9 5; 10 11; 11 12; 12 13; 13 14; 15 16; 16 1
13 16 12; 12 17; 17 13; 13 18; 18 14; 1 10; 2 11;
14 3 12; 4 13; 5 14; 6 15; 7 16; 8 17; 9 18];
15
16 num_node = size(node,1);
17 num_mem = size(member,1);
18 num_dof = 3*num_node;
19
20 // Member Properties
21 E(1:num_mem) = 1;
22 A(1:num_mem) = 1;
23
```

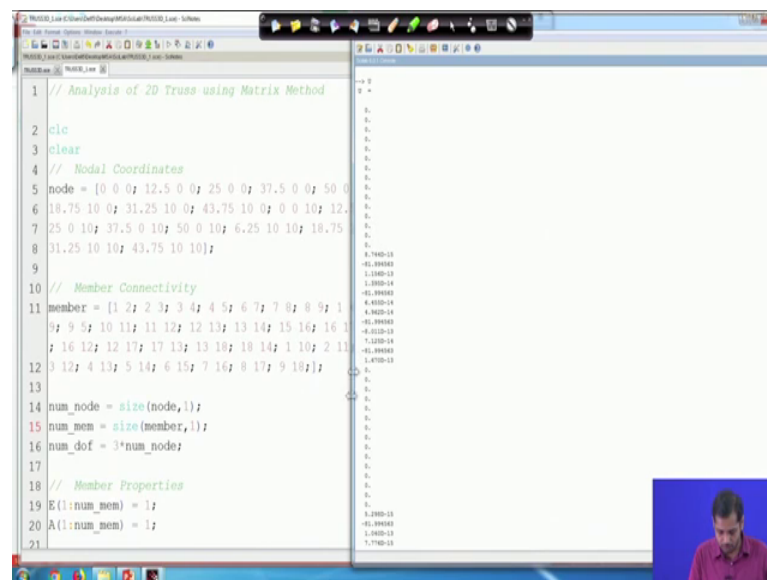
The image shows a MATLAB script in a window titled 'MATLAB_1.m'. The script is titled 'Analysis of 2D Truss using Matrix Method'. It defines the nodal coordinates for a truss structure with 18 nodes. It also defines the member connectivity, listing 18 members. The script calculates the number of nodes, members, and degrees of freedom. It also defines the member properties, setting the modulus of elasticity E and cross-sectional area A to 1 for all members. In the background, a diagram of a truss structure is visible, and a small video inset shows a person speaking.

(Refer Slide Time: 17:23)



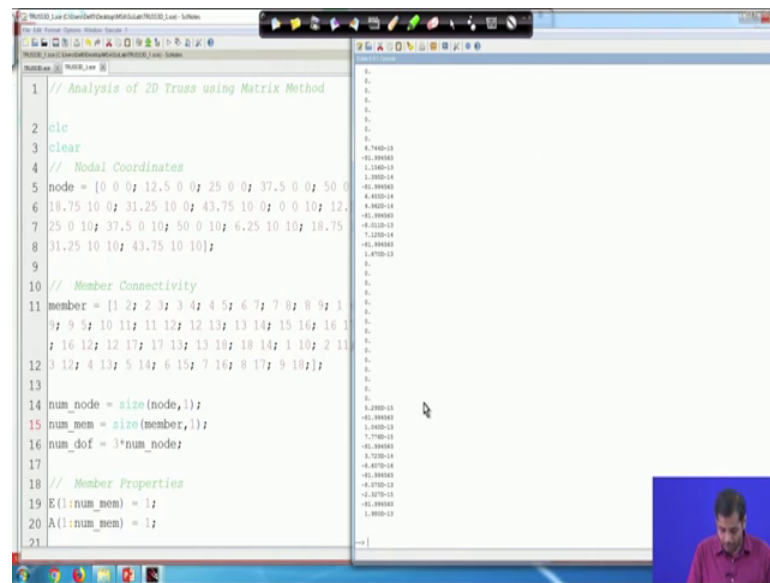
So, this is again this is a projected 2-D deform shape. Now if I see the say total displacement, total displacement will be this. These are all total displacement.

(Refer Slide Time: 17:34)



These 0's that you can see that is corresponding to the degrees of freedom; which are which are constrained.

(Refer Slide Time: 17:39)

A screenshot of a MATLAB script titled 'Analysis of 2D Truss using Matrix Method'. The script defines nodal coordinates, member connectivity, and member properties. The output window shows the calculated nodal displacements for 15 nodes.

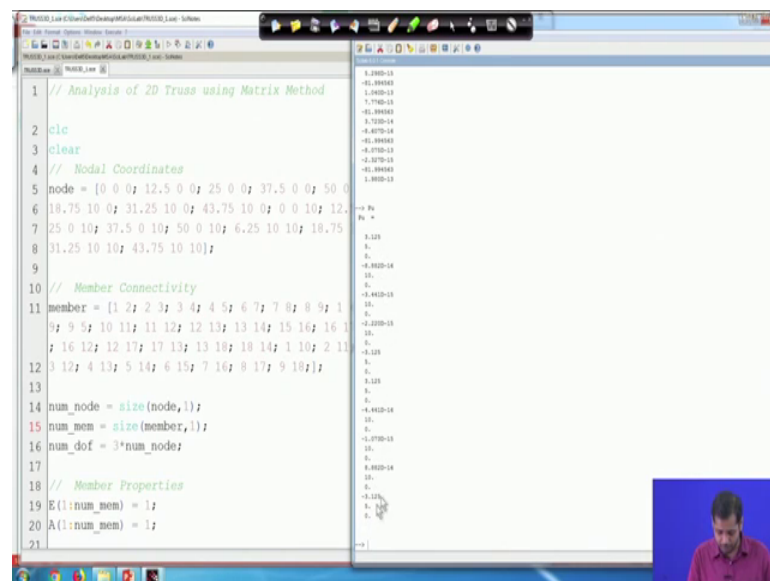
```
1 // Analysis of 2D Truss using Matrix Method
2
3 clear
4 // Nodal Coordinates
5 node = [0 0 0; 12.5 0 0; 25 0 0; 37.5 0 0; 50 0
6 18.75 10 0; 31.25 10 0; 43.75 10 0; 0 0 10; 12.5
7 25 0 10; 37.5 0 10; 50 0 10; 6.25 10 10; 18.75
8 31.25 10 10; 43.75 10 10];
9
10 // Member Connectivity
11 member = [1 2; 2 3; 3 4; 4 5; 6 7; 7 8; 8 9; 1
12 9; 9 5; 10 11; 11 12; 12 13; 13 14; 15 16; 16 1
13 16 12; 12 17; 17 13; 13 18; 18 14; 1 10; 2 11;
14 3 12; 4 13; 5 14; 6 15; 7 16; 8 17; 9 18];
15
16 num_node = size(node,1);
17 num_mem = size(member,1);
18 num_dof = 3*num_node;
19
20 // Member Properties
21 E(1:num_mem) = 1;
22 A(1:num_mem) = 1;
```

Output window showing nodal displacements (u, v, w) for 15 nodes:

Node	u	v	w
1	0	0	0
2	1.2500e-13	-1.0000e-13	1.0000e-14
3	1.2500e-13	-1.0000e-13	1.0000e-14
4	1.2500e-13	-1.0000e-13	1.0000e-14
5	1.2500e-13	-1.0000e-13	1.0000e-14
6	1.2500e-13	-1.0000e-13	1.0000e-14
7	1.2500e-13	-1.0000e-13	1.0000e-14
8	1.2500e-13	-1.0000e-13	1.0000e-14
9	1.2500e-13	-1.0000e-13	1.0000e-14
10	1.2500e-13	-1.0000e-13	1.0000e-14
11	1.2500e-13	-1.0000e-13	1.0000e-14
12	1.2500e-13	-1.0000e-13	1.0000e-14
13	1.2500e-13	-1.0000e-13	1.0000e-14
14	1.2500e-13	-1.0000e-13	1.0000e-14
15	1.2500e-13	-1.0000e-13	1.0000e-14

And then the support reactions, support reaction is P u.

(Refer Slide Time: 17:48)

A screenshot of a MATLAB script titled 'Analysis of 2D Truss using Matrix Method'. The script defines nodal coordinates, member connectivity, and member properties. The output window shows the calculated nodal displacements for 15 nodes.

```
1 // Analysis of 2D Truss using Matrix Method
2
3 clear
4 // Nodal Coordinates
5 node = [0 0 0; 12.5 0 0; 25 0 0; 37.5 0 0; 50 0
6 18.75 10 0; 31.25 10 0; 43.75 10 0; 0 0 10; 12.5
7 25 0 10; 37.5 0 10; 50 0 10; 6.25 10 10; 18.75
8 31.25 10 10; 43.75 10 10];
9
10 // Member Connectivity
11 member = [1 2; 2 3; 3 4; 4 5; 6 7; 7 8; 8 9; 1
12 9; 9 5; 10 11; 11 12; 12 13; 13 14; 15 16; 16 1
13 16 12; 12 17; 17 13; 13 18; 18 14; 1 10; 2 11;
14 3 12; 4 13; 5 14; 6 15; 7 16; 8 17; 9 18];
15
16 num_node = size(node,1);
17 num_mem = size(member,1);
18 num_dof = 3*num_node;
19
20 // Member Properties
21 E(1:num_mem) = 1;
22 A(1:num_mem) = 1;
```

Output window showing nodal displacements (u, v, w) for 15 nodes:

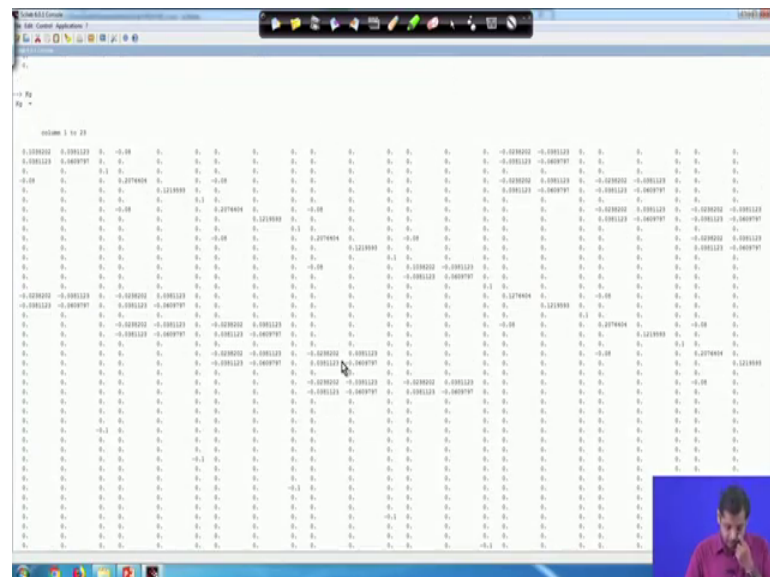
Node	u	v	w
1	0	0	0
2	1.2500e-13	-1.0000e-13	1.0000e-14
3	1.2500e-13	-1.0000e-13	1.0000e-14
4	1.2500e-13	-1.0000e-13	1.0000e-14
5	1.2500e-13	-1.0000e-13	1.0000e-14
6	1.2500e-13	-1.0000e-13	1.0000e-14
7	1.2500e-13	-1.0000e-13	1.0000e-14
8	1.2500e-13	-1.0000e-13	1.0000e-14
9	1.2500e-13	-1.0000e-13	1.0000e-14
10	1.2500e-13	-1.0000e-13	1.0000e-14
11	1.2500e-13	-1.0000e-13	1.0000e-14
12	1.2500e-13	-1.0000e-13	1.0000e-14
13	1.2500e-13	-1.0000e-13	1.0000e-14
14	1.2500e-13	-1.0000e-13	1.0000e-14
15	1.2500e-13	-1.0000e-13	1.0000e-14

The support reactions are these, if we join all the support reactions, let us then what will happen is, you should be getting the total load will be 0 if we join them. Whatever load we apply on this structure, in the corresponding direction if we join them that equilibrium should be satisfied; is a very important thing you see. All these we when we derive this in the formulation the equilibrium equation we use. So, necessarily the satisfy

equilibrium equation, but the problem comes when you translate those equations into a code.

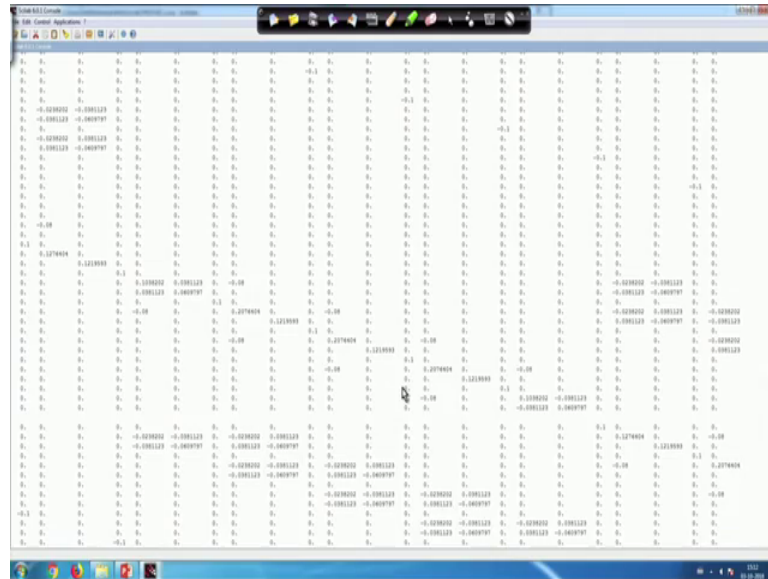
So, it is very important you evaluate your code. Validate your code just not the final results, you validate every steps in your code. For instance, whether one check I can make is check whether the whatever forces I am getting whether they satisfy the equilibrium or not right. So, these should be we should check that. Then k g if you check the total stiffness matrix, the global stiffness matrix, the global stiffness matrix is huge it is a global.

(Refer Slide Time: 18:50)



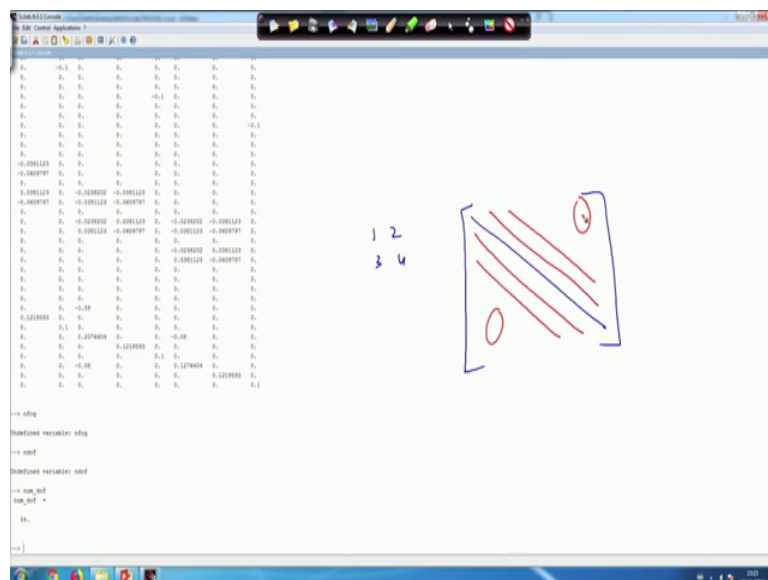
Now you see the maximum entity, it is a very important and we will discuss that towards the end. If the maximum entity in the global stiffness matrix is 0 some of the values are non-zero.

(Refer Slide Time: 19:01)



Now so, when you actually store a stiffness matrix in, this case the stiffness matrix is say here number of degrees of freedom is and number of.

(Refer Slide Time: 19:12)



Here number of degrees of freedom is how much; let us see it is, yes. So, it is number of degrees of freedom is 54, but you know, you can have a structure where number of degrees of freedom used in terms of 1000. So, and in that case your the stiffness matrix will be 1000 by 1000. If you see real structure, it is not even 1000 is much more than that your size of the stiffness matrix. But, if you look at the stiffness matrix,

the most of the entities in the stiffness matrix is 0. Some of the entities in the stiffness matrix depending on the connectivity of the member, some of the elements in the stiffness matrix will be non-zero.

So, in that case the storing of stiffness matrix, storing all that stiffness information is an important part in the code. One way is very crude ways you store stiffness as a matrix in a matrix form. But when you store in a matrix form, if the size of the matrix is say 1000 and 1000 by 1000 so, essentially he was storing 10 to the 4 elements 10 to 4 information. Even it is 0, but you are storing it right, when you when you store entire information in a matrix form. But since most of the elements in the stiffness matrix is 0 you can store in a different way. You can store only the nonzero elements in the stiffness matrix, and they are corresponding position in the global in the global stiffness matrix.

So, this is another aspect when you write a code so, you have to see how these matrices, all these better always better to avoid these matrix operations inversion, storing if we can do it in some other way, that you have to find out. Another important thing is we mentioned some time back during the discussion of discussion in some of the discussion earlier; that you can change the numbering pattern. See whatever way you number, you if your degrees of freedom numbering of the degrees of freedom and the numbering of nodes are consistent, you will get the results there is absolutely no problem in that.

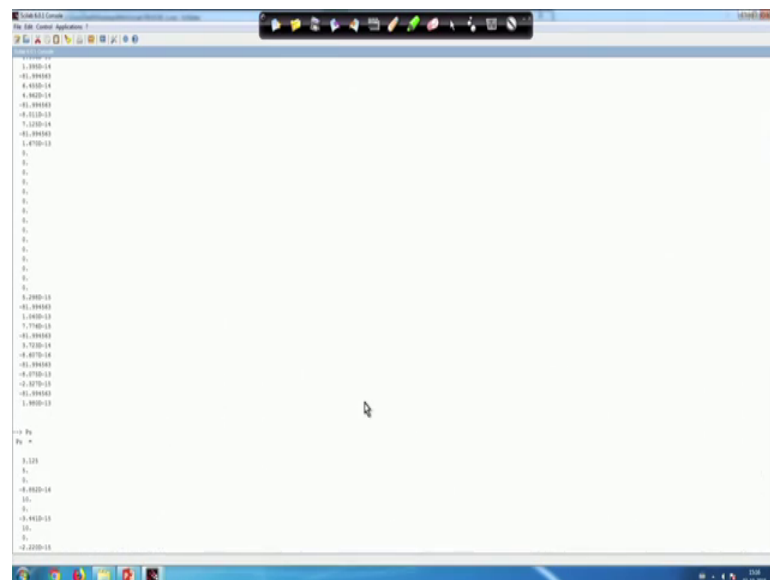
But you know if you change the numbering pattern, then what will happen? You make your stiffness matrix banded. Banded stiffness matrix means the diagonal term, and some of diagonal terms will be non-zero immediate of the diagonal terms will be non-zero, but rest of the thing is will be 0. For instance, one example of banded stiffness matrix is if I have to if you have to write some example suppose; it is 1, 2, 3, 4 and then so, you have a stiffness matrix, and then this is the diagonal entity, and then you have some off diagonal entity here. And then some off diagonal entity here, another of off diagonal entity, another off diagonally entity and this is 0, and this is 0. All the elements here is both triangles up and over lower triangle will be 0.

So, this is banded stiffness matrix. You can achieve this banded stiffness matrix by numbering, by changing the numbering pattern. If you if you number it oppositely, what you can have is you get a matrix. But those matrix is populated in this such a way that these this feature you will not get. You get you get something like this. For instance, this

matrix the numbering is done a very arbitrary way. This is not a, this may not be a banded stiffness matrix, you see. So, here it is 0, then non-zero then some of the elements 0 here, and then some of the elements 0 here. So, that will discuss later how to what would be the numbering pattern so, that you can get these stiffness matrix in this form.

So, this is important; that you keep in mind when you actually number the nodes. It is just not numbering oppositely. We check the determinant of k_g , just across check determinant of k_g is 0 it has to be because the boundary conditions were not satisfied. So, this is now you can explore as I said these codes will be uploaded on the forum. This are very basic code some of the information actually you have to do it you have to give manually.

(Refer Slide Time: 23:23)



But it gives you a flavor that of (Refer Time: 23:26) computer implementation of the method. So, you can further explore it, you can solve some other different problems using the code and check with other software and your solution and. And also you can do some exercise like changing the Young's modulus, make one suppose you have a truss where you have 100 members, and make one member very soft. Very soft in the sense, you reduce the Young's modulus of the one-member drastically. And rest of the members Young's modulus are same. And then check what is the behaviour of the track are you getting any localized behaviour or did not. Or are you getting a localized deformation at a place where your elements your members are weaker.

So, these are the important things these through this exercise, you can have some, you can develop some comprehension some idea about the behaviour of the structures. So, here I stop today. So, next class what we do is, we do this similar exercise for beam; again will take two problems in beam, and then see how using this code those beam problem can be solved. See you next class.

Thank you.