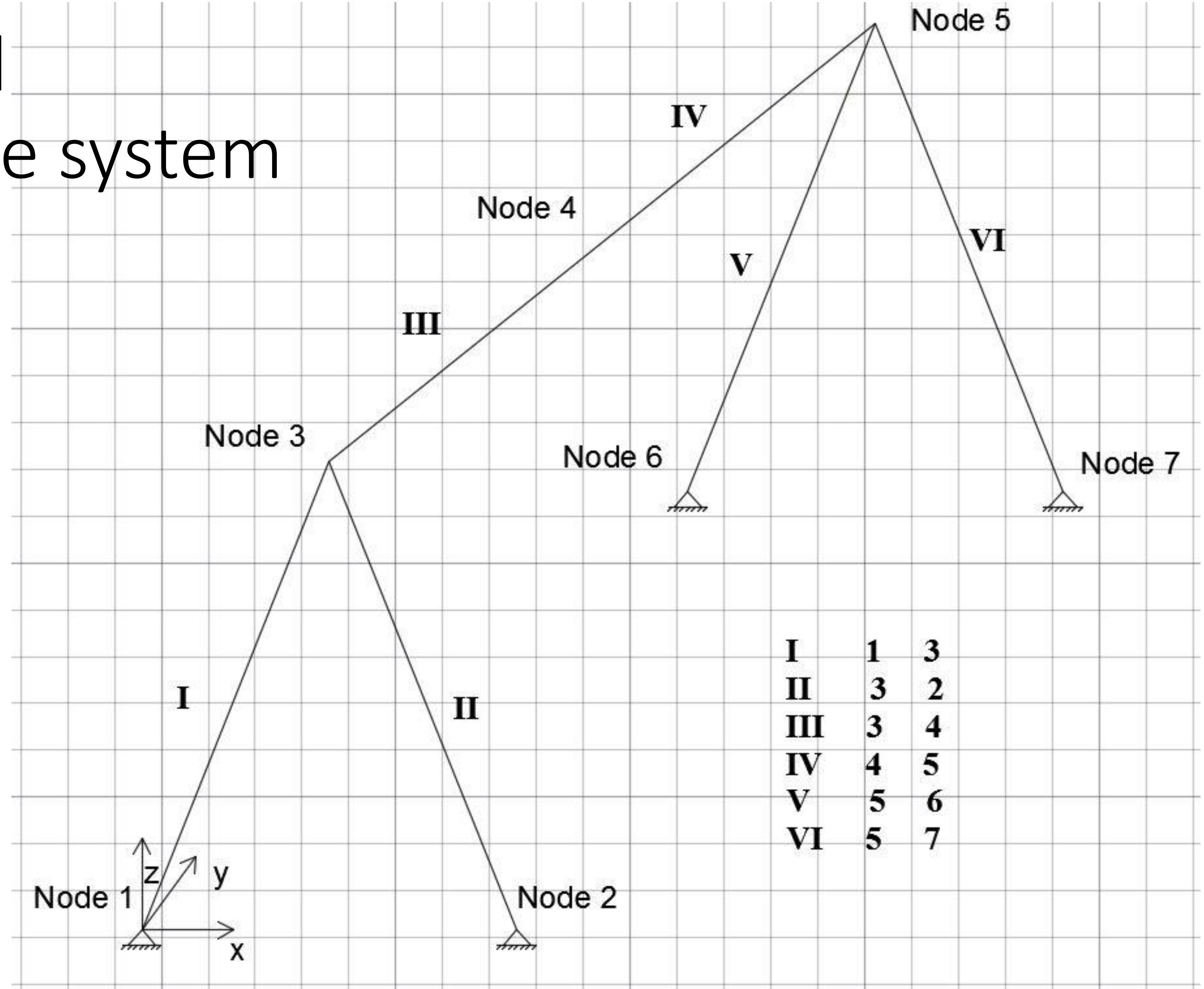
Final Project

ECI211 Fall 2015

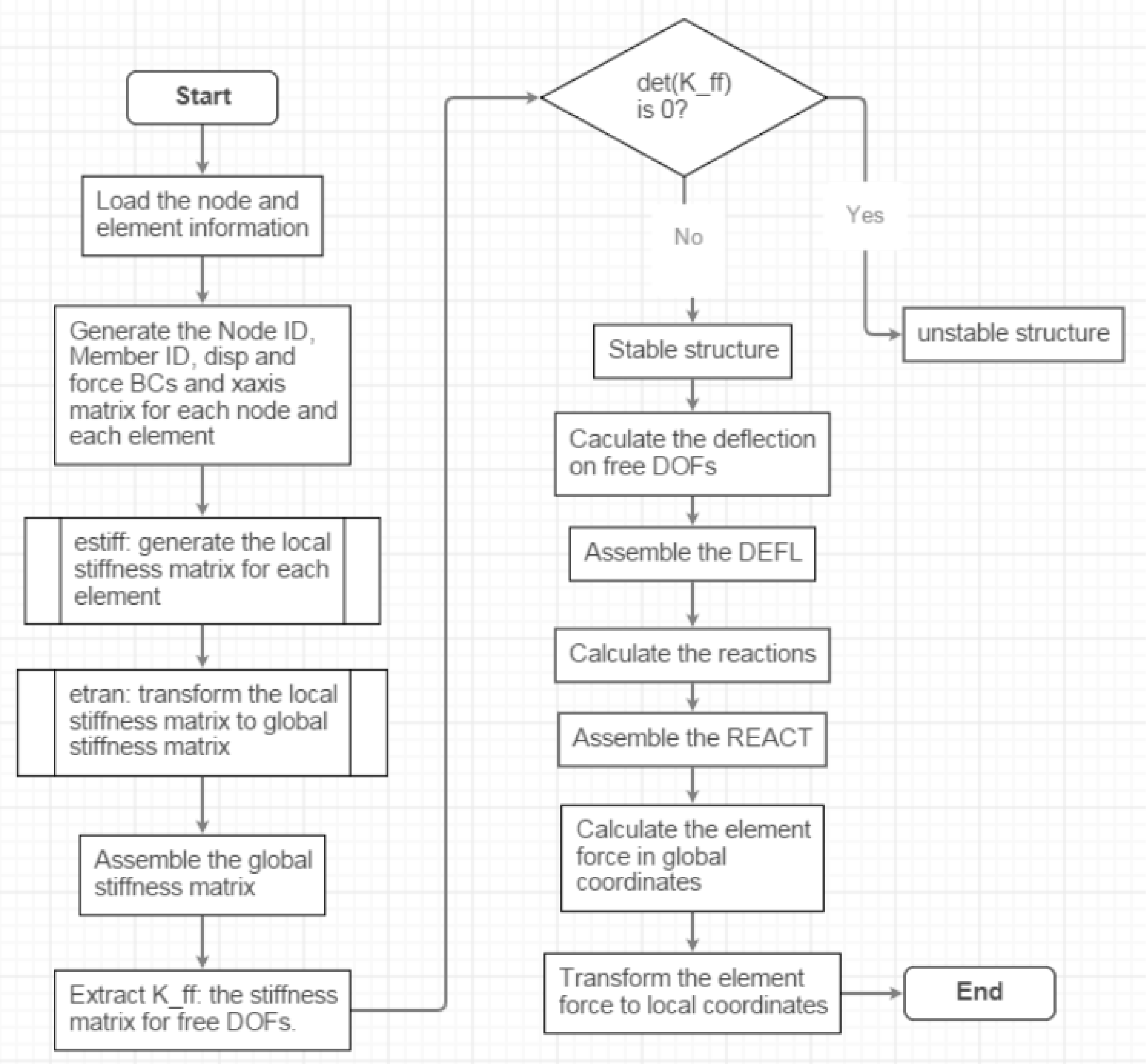
Yuan Feng

Dec 2nd, 2015

My global coordinate system



My flowchart



Load the node and element information

- % load the input information.
- load nnodes.txt;
- load coord.txt;
- load concen.txt;
- load fixity.txt;
- load nele.txt;
- load ends.txt;
- load Area.txt;
- load Izz.txt;
- load Iyy.txt;
- load J.txt;
- load E.txt;
- load v.txt;
- load beta_ang.txt;

Generate the node id.
Each line represents the 6 DOFs for one node.

- node_id=zeros(nnodes,6);
- for i=1:nnodes
-     for j=1:6
-         node_id(i,j)=(i-1)*6+j;
-     end
- end

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	38	39	40	41	42

Generate the member id.
Each line represents the 12 DOFs for one element.

- mem_id=zeros(nele,12);
- for i=1:nele
-     mem_id(i,1:6)=node_id(ends(i,1),1:6);
-     mem_id(i,7:12)=node_id(ends(i,2),1:6);
- end

1	2	3	4	5	6	13	14	15	16	17	18
13	14	15	16	17	18	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
19	20	21	22	23	24	25	26	27	28	29	30
25	26	27	28	29	30	31	32	33	34	35	36
25	26	27	28	29	30	37	38	39	40	41	42

Generate the displacement boundary conditions.
Each line in fixity represents the disp BC on 6 DOFs of one node.
0 represents that the DOF is fixed.
NaN represnts that the DOF is free.

- fixity_tran=fixity';
- D=fixity_tran(:);

- free_dof=find(isnan(D));
- fixed_dof=find(D==0);

0	0	0	NaN	NaN	NaN
0	0	0	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN
0	0	0	NaN	NaN	NaN
0	0	0	NaN	NaN	NaN

1 2 3 7 8 9 31 32 33 37 38 39

4 5 6 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 34 35 36 40 41 42

Generate the force boundary conditions.
Each line in concen represents the force BC on 6 DOFs of one node.
<u>0 represents that the DOF does not has external force.</u>
<u>The number represnts that the force values on the DOF.</u>

- concen_tran=concen';
- P_total=concen_tran(:);
-
- P_free=P_total(free_dof);

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	-4.5	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Generate the xaxis.
xaxis will be used in the rotation matrix.
The rotation matrix is to transform the stiffness matrix from global to local coordinate system.

```
• for i=1:nele
•     xaxis(i,:)=(coord(ends(i,2),:)-coord(ends(i,1),:));
•     Length(i) = norm(xaxis(i,:));
•     xaxisunit(i,:)=xaxis(i,:)/Length(i) ;
• end
```

0.3713	0	0.9284
0.3713	0	-0.9284
0.0000	1	0.0000
0.0000	1	0.0000
-0.3713	0	-0.9284
0.3713	0	-0.9284

Generate the local stiffness matrix for each element.

- k_stack_local=zeros(nele,12,12);
- gamma_stack_local=zeros(nele,12,12);
- for i=1:nele
- k_stack_local(i,:,:)=estiff(Area(i),Izz(i),Iyy(i),J(i),E(i),v(i),Length(i));
- gamma_stack_local(i,:,:)=etran(beta_ang(i),xaxisunit(i,1:3));
- end

Transform the local stiffness matrix to global stiffness matrix.

- k_stack_global=zeros(nele,12,12);
- for i=1:nele
-     k_stack_global(i,:,:)=(squeeze(gamma_stack_local(i,:,:))')...
-                             *squeeze(k_stack_local(i,:,:))...
-                             *squeeze(gamma_stack_local(i,:,:));
- end

```
size(squeeze(k_stack_local(i,:,:)))
12    12
size(k_stack_local(i,:,:))
1   12   12
```

Assemble the global stiffness matrix together.

- ndof=6*nnodes;
- k_total=zeros(ndof,ndof);
- for i=1:nele
-     k_total(mem_id(i,1:6),mem_id(i,1:6))=k_total(mem_id(i,1:6),mem_id(i,1:6))+squeeze(k_stack_global(i,1:6,1:6));
-     k_total(mem_id(i,1:6),mem_id(i,7:12))=k_total(mem_id(i,1:6),mem_id(i,7:12))+squeeze(k_stack_global(i,1:6,7:12));
-     k_total(mem_id(i,7:12),mem_id(i,1:6))=k_total(mem_id(i,7:12),mem_id(i,1:6))+squeeze(k_stack_global(i,7:12,1:6));
-     k_total(mem_id(i,7:12),mem_id(i,7:12))= k_total(mem_id(i,7:12),mem_id(i,7:12))+squeeze(k_stack_global(i,7:12,7:12));
- end

$$\begin{array}{c} \boldsymbol{K}^{(1)} \qquad\qquad\qquad \mathbf{0} \\ \begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} & K_{15} & K_{16} \\ K_{21} & K_{22} & K_{23} & K_{24} & K_{25} & K_{26} \\ K_{31} & K_{32} & K_{33} & K_{34} & K_{35} & K_{36} \\ K_{41} & K_{42} & K_{43} & K_{44} & K_{45} & K_{46} \\ K_{51} & K_{52} & K_{53} & K_{54} & K_{55} & K_{56} \\ K_{61} & K_{62} & K_{63} & K_{64} & K_{65} & K_{66} \end{bmatrix} \\ \mathbf{0} \qquad\qquad\qquad \boldsymbol{K}^{(2)} \end{array} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{Bmatrix}$$

Extract the K_ff.

- K_ff=k_total(free_dof,free_dof);
- K_sf=k_total(fixed_dof,free_dof);

C = det(A) returns the determinant for A.
If the structure is stable, det(A) is positive.
If the structure is unstable, det(A) is near 0.

```
• if abs(det(K_ff))<(10^(-15))
•     AFLAG=0;
•     display('Unstable Structure.');
•     display('Maybe you should fix more DOFs .');
• else
•     AFLAG=1;
•     display('The structure is stable.');
• end
```

C = rcond(A) returns the reciprocal condition number of A.
If A is well conditioned, rcond(A) is near 1.0.
If A is badly conditioned, rcond(A) is near 0.

```
if abs(rcond(K_ff))<(10^(-15))
    BFLAG=0;
    display('The matrix is badly conditioned. ');
    display('The results may have a large error. ');
    display('Some element may have a way bigger stiffness than others.');
    display('Maybe you need to check the units of element properties.');
else
    BFLAG=1;
    display('The matrix is well-conditioned.');
end
```

Put the deflection to its appropriate location. Include the fixed DOF.

- defl=K_ff\P_fr
- defl_vector_total=zeros(ndof,1);
- defl_vector_total(free_dof)=defl;

Each line in DEFL' represents the 6 DOFs at each node.

- DEFL=reshape(defl_vector_total,6,nnodes);
- DEFL=DEFL';
- display(DEFL)

Generate the reaction.

- react=K_sf*defl;

- react_vector_total=zeros(ndof,1);
- react_vector_total(fixed_dof)=react;
- REACT=reshape(react_vector_total,6,nnodes);
- REACT=REACT';
- display(REACT)

Calculate the element force. Transform the element from from global to local coordiante system.

- ELE_FOR=zeros(nele,12);
- for i=1:nele
-     d_global_tmp=zeros(1,12);
-     d_global_tmp(1:6)=DEFL(ends(i,1),1:6);
-     d_global_tmp(7:12)=DEFL(ends(i,2),1:6);
-     d_global_tmp=d_global_tmp';
-     f_local_tmp=squeeze(k_stack_local(i,:,:))*squeeze(gamma_stack_local(i,:,:))*d_global_tmp;
-     for j=1:12
-     ELE_FOR(i,j)=f_local_tmp(j);
-     end
- end
- display(ELE_FOR)

Results

- The structure is stable.
- The matrix is well-conditioned.

Results

- DEFL =

	1	2	3	4	5	6
1	0	0	0	7.5737e-04	2.5418e-06	-0.0013
2	0	0	0	7.5737e-04	-2.5418e-06	0.0013
3	-9.3650e-18	0.0026	-0.0123	-0.0026	-2.8740e-21	-3.4050e-19
4	3.7910e-16	-5.3331e-12	-4.4649	-5.2095e-16	-1.3301e-21	9.6745e-21
5	-2.1273e-18	-0.0026	-0.0123	0.0026	-5.3117e-23	2.8733e-19
6	0	0	0	-7.5737e-04	2.5418e-06	0.0013
7	0	0	0	-7.5737e-04	-2.5418e-06	-0.0013

Results

- REACT =

	1	2	3	4	5	6
1	0.4498	0.2505	1.1250	0	0	0
2	-0.4498	0.2505	1.1250	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0.4498	-0.2505	1.1250	0	0	0
7	-0.4498	-0.2505	1.1250	0	0	0

Results

- ELE_FOR =

	1	2	3	4	5	6	7	8	9	10	11	12
1	1.2116	0.2505	1.7670e-04	3.6971e-14	1.7152e-17	3.3173e-14	-1.2116	-0.2505	-1.7670e-04	-3.6971e-14	-0.4758	674.5522
2	1.2116	-0.2505	-1.7670e-04	-1.4211e-14	0.4758	-674.5522	-1.2116	0.2505	1.7670e-04	1.4211e-14	-1.1102e-16	2.8422e-14
3	0.5010	1.2575e-16	2.2500	-1.9951e-16	-1.2526e+03	3.5484e-14	-0.5010	-1.2575e-16	-2.2500	1.9951e-16	-2.1224e+03	1.5314e-13
4	0.5010	-1.4200e-16	-2.2500	-1.6503e-16	2.1224e+03	-1.5314e-13	-0.5010	1.4200e-16	2.2500	1.6503e-16	1.2526e+03	-5.9851e-14
5	1.2116	0.2505	1.7670e-04	0	-0.4758	674.5522	-1.2116	-0.2505	-1.7670e-04	0	3.3307e-16	-2.8422e-14
6	1.2116	0.2505	-1.7670e-04	-2.8422e-14	0.4758	674.5522	-1.2116	-0.2505	1.7670e-04	2.8422e-14	-8.8818e-16	-8.5265e-14

Thanks!

- Questions?