

ISTANBUL TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE ENGINEERING AND
TECHNOLOGY



MKC525E
FINITE ELEMENT ANALYSIS IN ENGINEERING

Homework 5

Erdem Çalışkan
503191531
01/07/2020

1 Question 1

Domain in the problem: **A5+B3+C3+D2**

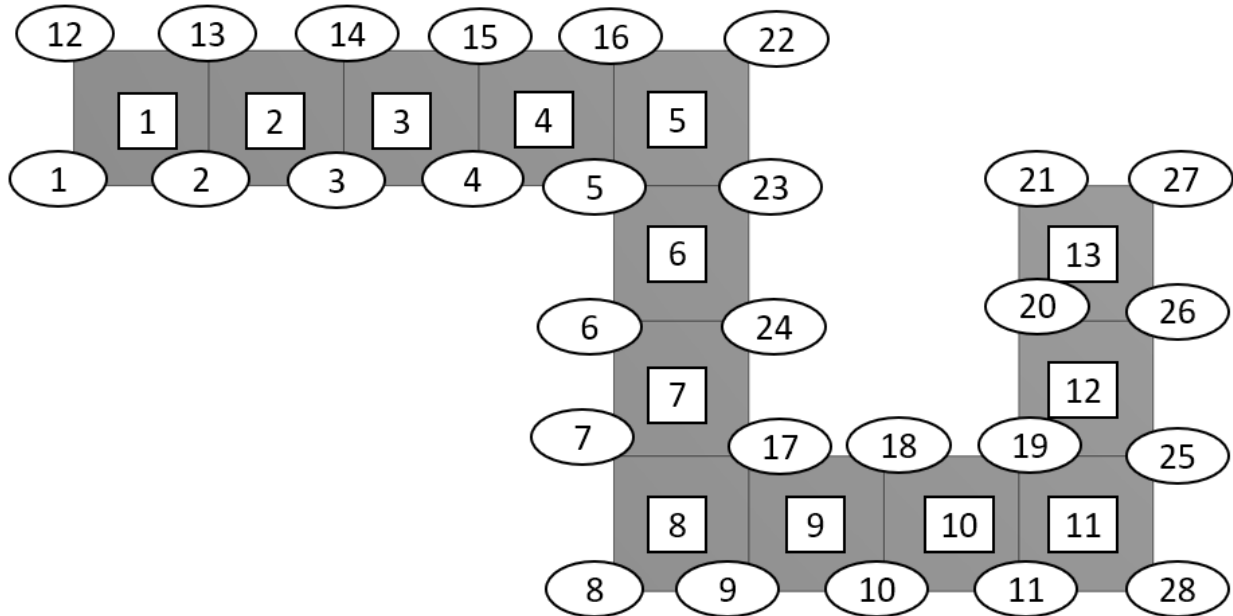


Figure 1: Domain; element numbers are inside square, node numbers are inside ellipse.

Topology matrix:

$$\begin{array}{cccccccc}
 1 & 2 & 3 & 4 & 25 & 26 & 23 & 24 \\
 3 & 4 & 5 & 6 & 27 & 28 & 25 & 26 \\
 5 & 6 & 7 & 8 & 29 & 30 & 27 & 28 \\
 7 & 8 & 9 & 10 & 31 & 32 & 29 & 30 \\
 9 & 10 & 45 & 46 & 43 & 44 & 31 & 32 \\
 11 & 12 & 47 & 48 & 45 & 46 & 9 & 10 \\
 13 & 14 & 33 & 34 & 47 & 48 & 11 & 12 \\
 15 & 16 & 17 & 18 & 33 & 34 & 13 & 14 \\
 17 & 18 & 19 & 20 & 35 & 36 & 33 & 34 \\
 19 & 20 & 21 & 22 & 37 & 38 & 35 & 36 \\
 21 & 22 & 55 & 56 & 49 & 50 & 37 & 38 \\
 37 & 38 & 49 & 50 & 51 & 52 & 39 & 40 \\
 39 & 40 & 51 & 52 & 53 & 54 & 41 & 42
 \end{array} \tag{1}$$

$$[ff] = \begin{Bmatrix} 31100 \\ 8333 \\ 2233 \\ 8333 \end{Bmatrix} \tag{2}$$

$$\text{Maximum deflection} = 0.46 \text{ mm} \tag{3}$$

$$\text{Maximum deflection (Ansys)} = 0.44 \text{ mm} \tag{4}$$

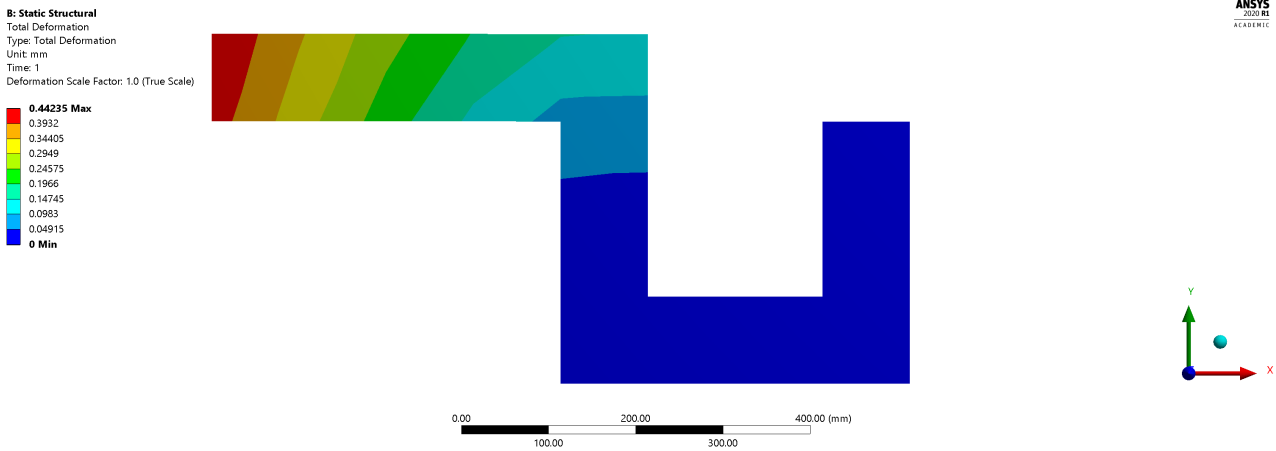


Figure 2: Total deformation in Ansys.

Mode	Natural frequency (Normal) [Hz]	Natural frequency (Constrained) [Hz]
1	3157.60 i	2522.95 i
2	2815.50 i	2452.48 i
3	2457.56 i	2210.45 i
4	1771.23 i	1736.88 i
5	1693.28 i	356.98
6	177.97	1419.05
7	399.60	1598.20
8	619.53	2326.74
9	1649.43	2705.62
10	2272.36	3198.49

Table 1: First 10 modes of the plate

Mode	Natural frequency (Normal) [Hz]	Natural frequency (Constrained) [Hz]
1	0.00	201.34
2	0.00	616.26
3	0.00	1319.40
4	386.46	2013.70
5	626.34	3185.60
6	1457.60	4324.90
7	2248.60	4633.20
8	3697.30	5291.00
9	4814.20	5912.20
10	5232.30	7182.50

Table 2: First 10 modes of the plate (Ansys)

1.1 Deflection at the upper right edge of the plate when a sudden in-plane distributed loading of 0.5 N/mm^2 is applied at $t = 0$

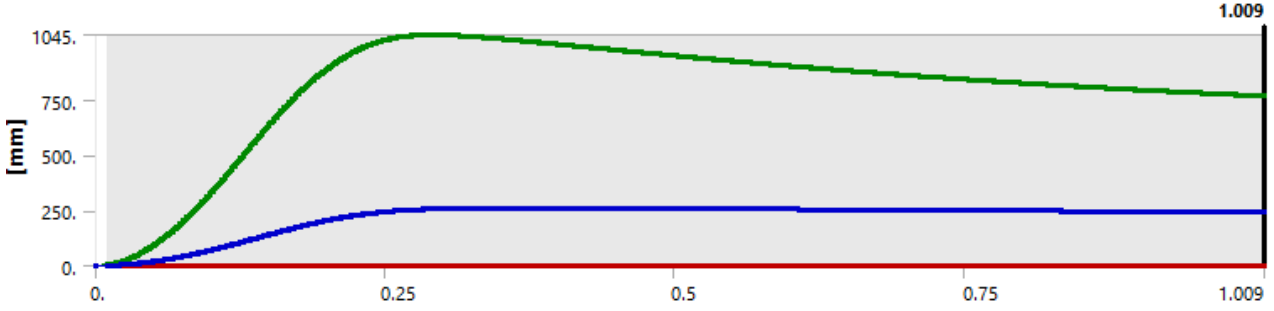


Figure 3: Total deformation in Ansys Explicit Dynamics (Plane strain).

In this section three different time steps are compared using Central Difference, Trapezoidal and Damped Newmark methods. And different number of time steps are examined using Trapezoidal method.

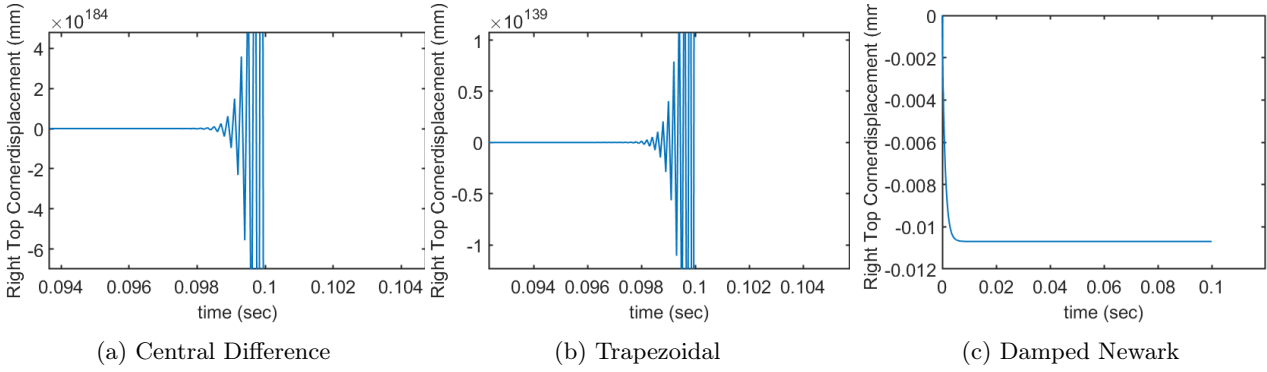


Figure 4: Deflection at the upper right edge of the plate ($\Delta t = 1e - 4$).

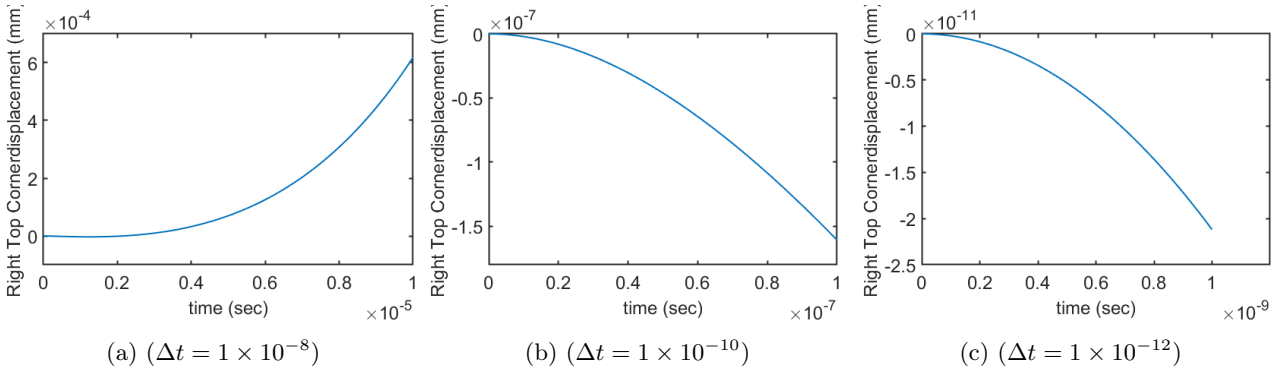


Figure 5: Deflection at the upper right edge of the plate for $(t = 1e4)$ time steps.

1.2 Reaction forces

DOF	Node 8	Node 9	Node 10	Node 11	Node 28
u	9	-96	-134	185	171
v	-158	-477	-1054	-1104	-534

Table 3: Reaction forces (Matlab)

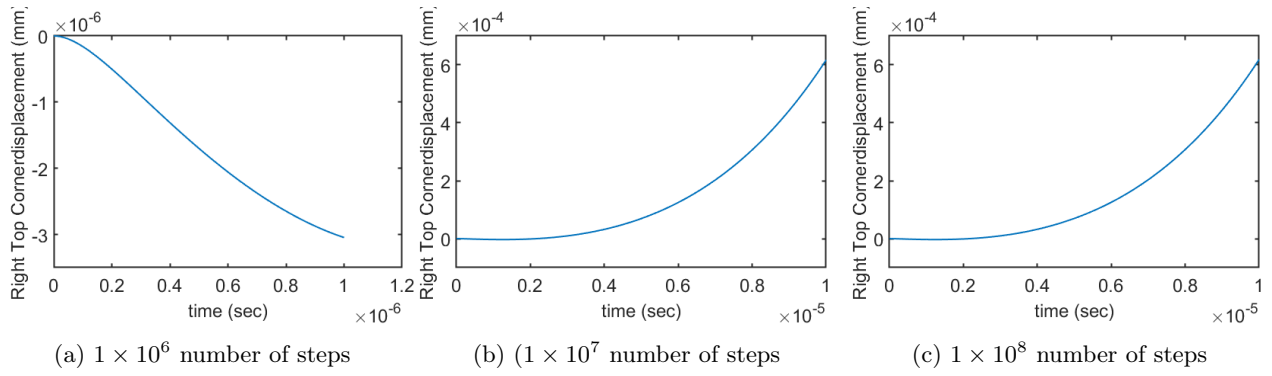


Figure 6: Deflection at the upper right edge of the plate for $(\delta t = 1e - 12)$ time steps.

DOF	Node 8	Node 9	Node 10	Node 11	Node 28
u	37	195	218	-11	-2
v	-393	-90	147	-50	-13

Table 4: Reaction forces (Ansys)

1.3 Element stresses

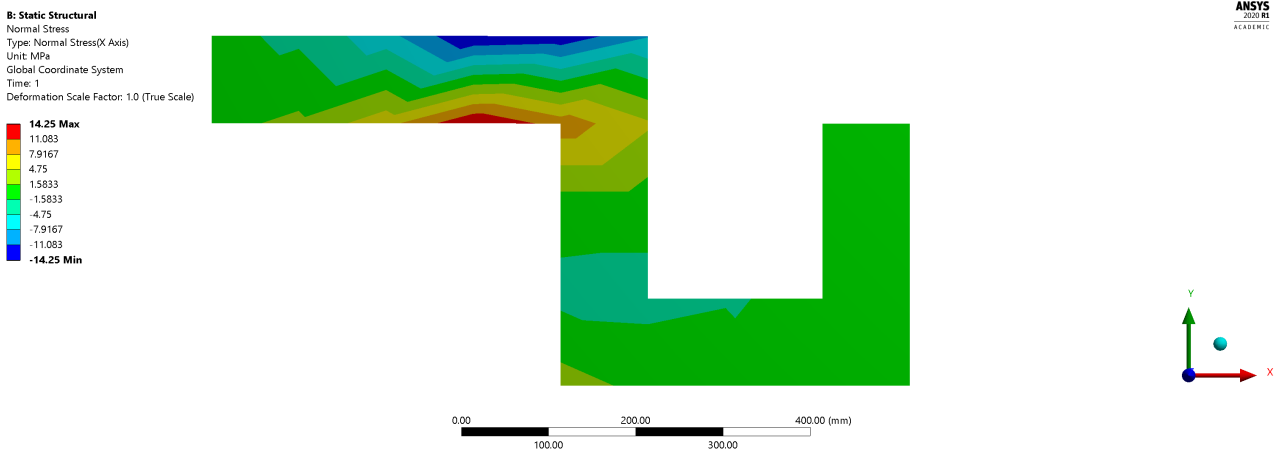


Figure 7: Normal stress (x-direction) in Ansys.

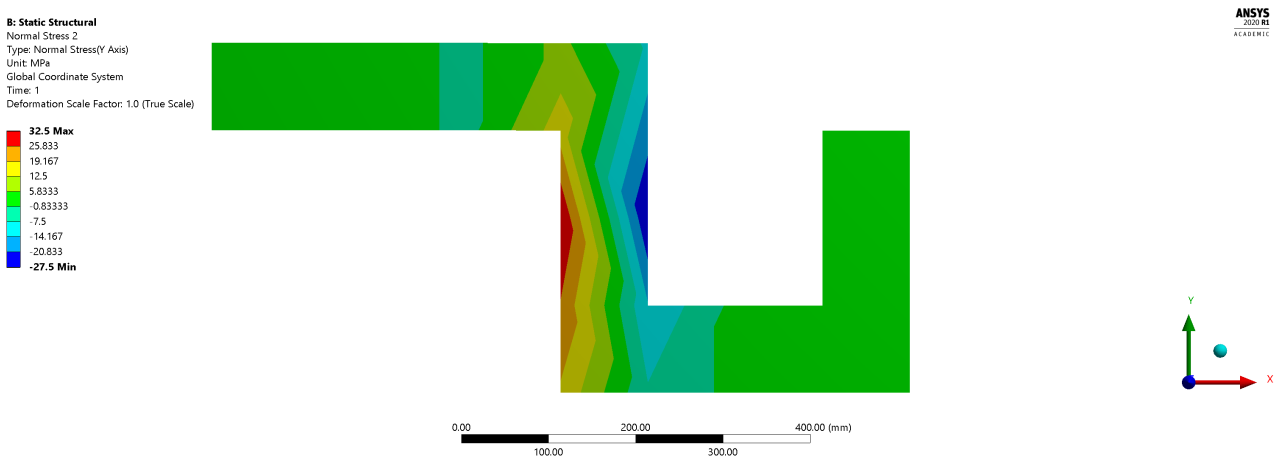


Figure 8: Normal stress (y-direction) in Ansys.

B: Static Structural
 Shear Stress
 Type: Shear Stress(XY Component)
 Unit: MPa
 Global Coordinate System
 Time: 1
 Deformation Scale Factor: 1.0 (True Scale)

ANSYS
 2020 R1
 ACADEMIC

4 Max
 3.2984
 2.5968
 1.8952
 1.1936
 0.492
 -0.20959
 -0.91119
 -1.6128
-2.3144 Min

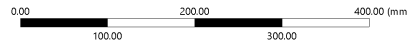


Figure 9: Shear stress in Ansys.

Element No	Element 1	Element 2	Element 3	Element 4	Element 5	Element 6	Element 7
σ_{11}	-2.20	0.90	3.47	-0.63	-2.58	-0.51	0.26
σ_{22}	3.46	3.20	0.91	-1.98	-0.10	2.25	1.22
τ	-0.48	-1.18	-0.58	0.95	0.39	-0.48	-0.42
Element No	Element 8	Element 9	Element 10	Element 11	Element 12	Element 13	
σ_{11}	0.05	0.00	0.09	0.08	-3.63	-2.63	
σ_{22}	0.16	0.00	0.28	0.25	2.98	2.04	
τ	-0.05	0.09	-0.01	-0.06	-0.48	-1.18	

Table 5: Element stresses

1	2398.07 i
2	2283.47 i
3	1910.13 i
4	1500.89 i
5	349.20
6	1206.37
7	1357.62
8	2057.53
9	2158.28
10	2402.52

Table 6: Natural frequencies, found using lumped mass matrices.

2 Question 2

Same domain is used for this problem but node numbering is different.

Boundary conditions:

1. Temperature at the left edge $T_L = 20\text{ }^\circ\text{C}$
2. Temperature at the right edge $T_L = 50\text{ }^\circ\text{C}$

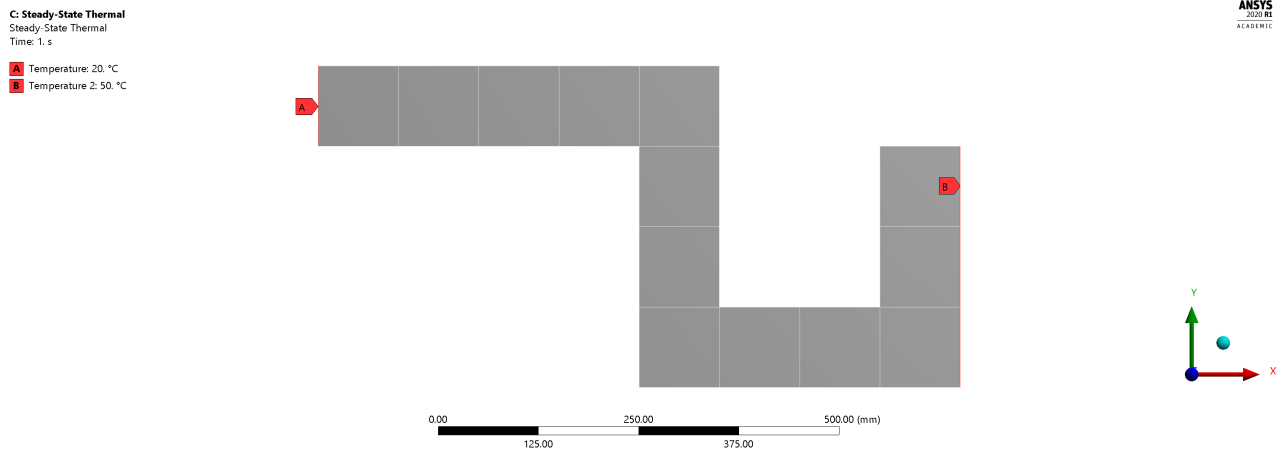


Figure 10: Boundary conditions.

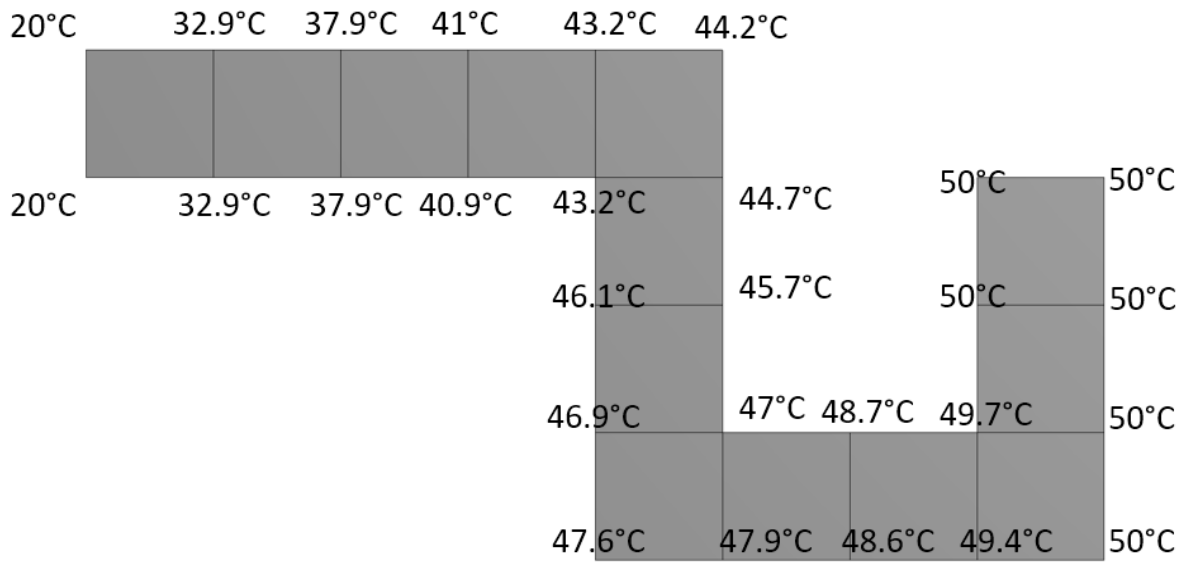


Figure 11: Temperature distribution.

C: Steady-State Thermal
 Temperature
 Type: Temperature
 Unit: °C
 Time: 1

ANSYS
 2020 R2
 ACADEMIC

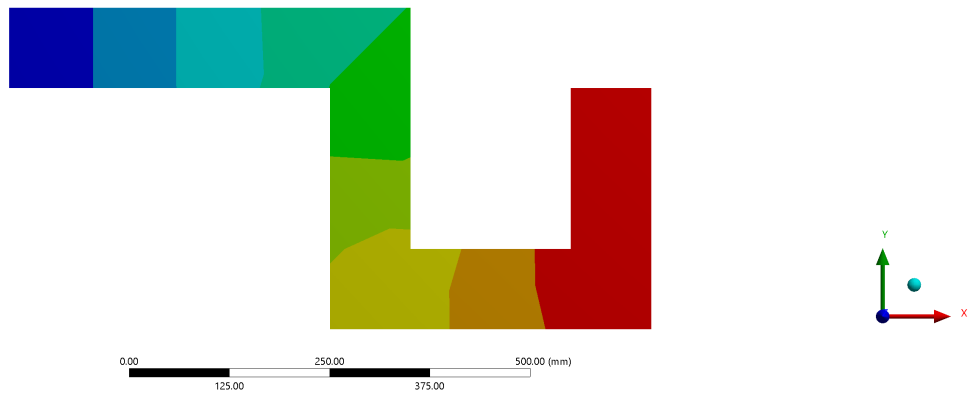


Figure 12: Temperature distribution.

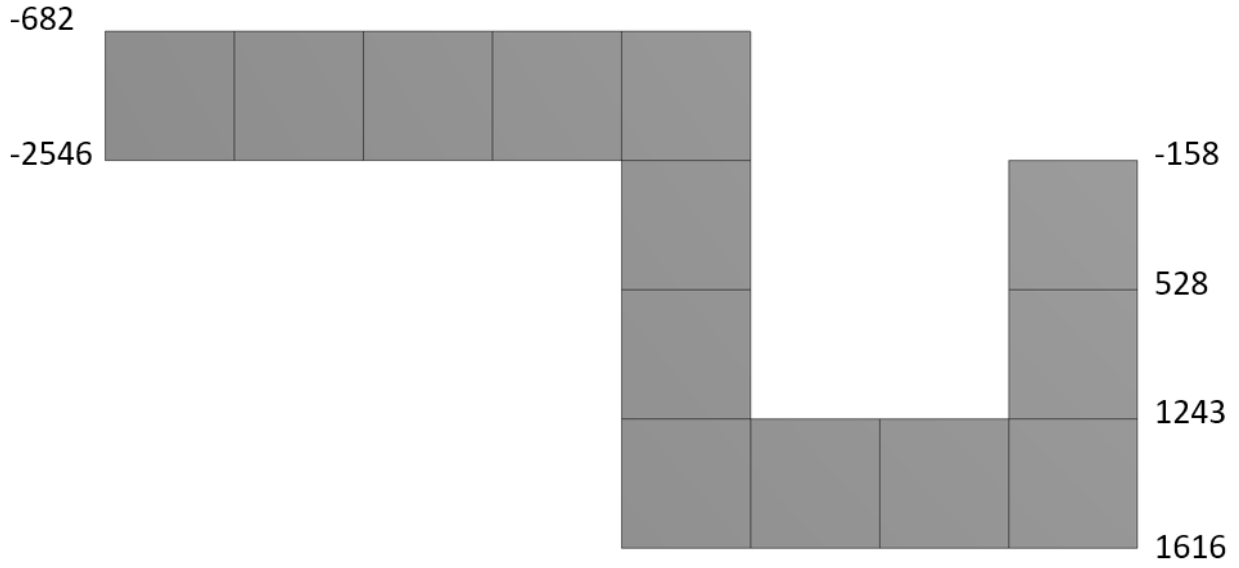
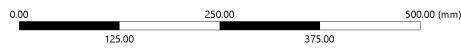
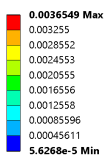


Figure 13: Heat flux.

C: Steady-State Thermal
 Total Heat Flux
 Type: Total Heat Flux
 Unit: W/mm²
 Time: 1



ANSYS
 2023 R2
 ACADEMIC

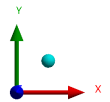


Figure 14: Heat flux.

Matlab Scripts

Listing 1: Matlab code for question 1

```

1 %HW5 Q1
2
3 clc
4 clear all
5 close all
6 format compact
7 format short e
8 tic
9
10 nel=13; %Total Number of Elements
11 nnode=28; %Total Number of Nodes
12
13 num_node=28; %number of total nodes
14 num_elem=13; %number of total elements
15
16 element_type=4; %number of nodes on each element
17 ndof=2; %degrees-of-freedom per node
18
19 num_eq=num_node*ndof; %number of equations
20
21 h=2; %Thickness
22 E=2.1e5; %Young Modulus
23 v=0.33; %Poisson Ratio
24
25 a=100; %Length of 1 element
26 b=100; %Width of 1 element
27 ro=7.85e-9; %Density
28
29 pz=5; %Distributed load
30
31
32
33 Kelstiff = zeros(12,12,nel);
34
35 k11 = [ 8+3*(1-v)    3*(1+v)    -8+3*(1-v)    3*(3*v-1)
36         3*(1+v)    8+3*(1-v)    -3*(3*v-1)    4-3*(1-v)
37        -8+3*(1-v)  -3*(3*v-1)    8+3*(1-v)    -3*(1+v)
38         3*(3*v-1)    4-3*(1-v)    -3*(1+v)    8+3*(1-v) ];
39
40 k21 = [ -4-3*(1-v)  -3*(1+v)    4-3*(1-v)    -3*(3*v-1)
41         -3*(1+v)   -4-3*(1-v)    3*(3*v-1)   -8-3*(1-v)
42         4-3*(1-v)    3*(3*v-1)   -8+3*(1-v)    3*(1+v)
43         -3*(3*v-1)  -8+3*(1-v)    3*(1+v)    -4-3*(1-v) ];
44 k22=k11;
45
46 kel=E*h/24/(1-v^2)*[ k11 transpose(k21);
47                     k21 k22 ];
48
49 me = ro*h*a*b/9*[ 4    0    2    0    1    0    2    0;
50                  0    4    0    2    0    1    0    2;
51                  2    0    4    0    2    0    1    0;
52                  0    2    0    4    0    2    0    1;
53                  1    0    2    0    4    0    2    0;
54                  0    1    0    2    0    4    0    2;
55                  2    0    1    0    2    0    4    0;

```

```

56         0  2  0  1  0  2  0  4];
57
58 topology=[ 1  2  3  4  25  26  23  24
59            3  4  5  6  27  28  25  26
60            5  6  7  8  29  30  27  28
61            7  8  9  10  31  32  29  30
62            9  10  45  46  43  44  31  32
63            11  12  47  48  45  46  9  10
64            13  14  33  34  47  48  11  12
65            15  16  17  18  33  34  13  14
66            17  18  19  20  35  36  33  34
67            19  20  21  22  37  38  35  36
68            21  22  55  56  49  50  37  38
69            37  38  49  50  51  52  39  40
70            39  40  51  52  53  54  41  42];
71
72 kglobal=zeros(num_eq);
73
74 for e=1:1:13
75     elem_top=[topology(e,:)];
76     kglobal(elem_top,elem_top)=kglobal(elem_top,elem_top)+kel;
77
78 end
79
80 gpoints = [ -0.577350269189626      -0.577350269189626;
81            0.577350269189626      -0.577350269189626;
82            0.577350269189626      0.577350269189626;
83           -0.577350269189626      0.577350269189626];
84
85 gweights = [1 1;
86             1 1;
87             1 1;
88             1 1];
89
90
91 E1 = E/(1-v^2);
92
93 ff = zeros(8,1);
94
95 N = zeros(1,8);
96 n = sym('n',[8 2]);
97 ns = sym('ns',[8 2]);
98 nt = sym('nt',[8 2]);
99 ns2 = sym('ns2',[8 2]);
100 nt2 = sym('nt2',[8 2]);
101 nsnt = sym('nsnt',[8 2]);
102
103 syms s t
104 for i=1:length(gweights)
105     for j=1:length(gweights)
106         for node=1:4
107             if node==1
108                 sc=-1;
109                 tc=-1;
110             elseif node==2
111                 sc=1;
112                 tc=-1;
113             elseif node==3

```

```

114         sc=1;
115         tc=1;
116         else
117         sc=-1;
118         tc=1;
119         end
120         n(node,:)=[(1/4)*(1+sc*s)*(1+tc*t),...
121                 (1/4)*(1+sc*s)*(1+tc*t)];
122     end
123
124     ns=diff(n,s);
125     nt=diff(n,t);
126     ns2=diff(n,s,2);
127     nt2=diff(n,t,2);
128     nsnt=diff(ns,t);
129
130     N=[n(1,:) n(2,:) n(3,:) n(4,:)];
131     Ns2=[ns(1,:) ns(2,:) ns(3,:) ns(4,:)];
132     Nt2=[nt(1,:) nt(2,:) nt(3,:) nt(4,:)];
133     Nsnt=[nsnt(1,:) nsnt(2,:) nsnt(3,:) nsnt(4,:)];
134
135     B=[Ns2/a^2; Nt2/b^2; 2/(a*b)*Nsnt];
136
137     D=[ E1 E1*v 0;...
138         E1*v E1 0;...
139         0 0 E/(2*(1+v))];
140
141     %Element consistent load vector
142     ff=a*b*N'*pz*gweights(i)*gweights(j);
143     ff=subs(ff,s,gpoints(i));
144     ff=subs(ff,t,gpoints(j));
145
146     end
147 end
148
149
150 %Global mass matrix
151 Mglob=zeros(nnode*ndof);
152
153 for nel=1:13
154     for i=1:8
155         for j=1:8
156             Mglob(topology(nel,i),topology(nel,j))= ...
157                 Mglob(topology(nel,i),topology(nel,j))+me(i,j);
158         end
159     end
160 end
161
162 %Lumped mass matrix using row sum method
163 MglobLum=zeros(nnode*ndof);
164 for i=1:nnode*ndof
165     MglobLum(i,i)=sum(Mglob(i,:));
166 end
167
168 %Global force vector
169 Fglob=zeros(nnode*ndof,1);
170 for nel=1:5
171     for i=[6 8]

```

```

172         Fglob(topology(nel,i),1)= ...
173         Fglob(topology(nel,i),1)+ff(i,1);
174     end
175 end
176
177 for nel=8
178     for i=6
179         Fglob(topology(nel,i),1)= ...
180         Fglob(topology(nel,i),1)+ff(i,1);
181     end
182 end
183
184 for nel=8
185     for i=6
186         Fglob(topology(nel,i),1)= ...
187         Fglob(topology(nel,i),1)+ff(i,1);
188     end
189 end
190
191 for nel=[9 10 13]
192     for i=[6 8]
193         Fglob(topology(nel,i),1)= ...
194         Fglob(topology(nel,i),1)+ff(i,1);
195     end
196 end
197
198
199 %Boundary conditions (Bottom edge cantilever)
200 BCs=[15;16;17;18;19;20;21;22;55;56];
201 activeDof=setdiff(1:nnode*ndof,BCs);
202 Kglobactive=kglobal(activeDof,activeDof);
203 Mglobactive=Mglob(activeDof,activeDof);
204 MglobLumactive=MglobLum(activeDof,activeDof);
205 Fglobactive=Fglob(activeDof,1);
206
207 %Solve
208 d=Kglobactive\Fglobactive;
209 do(activeDof,1)=[d];
210
211 do = [ do
212         0
213         0];
214 %Obtaining only vertical displacement DOFs
215
216
217 count=1;
218 for i=1:28
219     du(i)=do(count); %Nodal displacements
220     count=count+2;
221 end
222
223 count=2;
224 for i=2:27
225     dv(i)=do(count); %Nodal displacements
226     count=count+2;
227 end
228
229 maxdu=max(du) %Maximum deflection

```

```

230 maxdv=max(dv)
231
232 %Reaction forces and moments at nodes
233 Freaction=kglobal*do-Fglob;
234
235 %Strain at nodes (eps x, eps y, gama xy)
236 B1=subs(B,s,-1);
237 B1=subs(B1,t,-1);
238 B1=double(B1);
239 strain=zeros(3,1,13);
240
241 e=0;
242 for i=[1:11] %Nodes [1:11]
243     e=e+1;
244     strain(:,:,i)=B1*do(topology(e,:));
245 end
246
247 B2=subs(B,s,-1);
248 B2=subs(B2,t,1);
249 B2=double(B2);
250
251 e=0;
252 for i=[12:16] %Nodes [12 13 14 15 16]
253     e=e+1;
254     strain(:,:,i)=B2*do(topology(e,:));
255 end
256
257 e=8;
258 for i=[17:21] %Nodes [17 18 19 20 21]
259     e=e+1;
260     strain(:,:,i)=B2*do(topology(e,:));
261 end
262
263 B3=subs(B,s,1);
264 B3=subs(B3,t,1);
265 B3=double(B3);
266
267 e=4;
268 for i=[22:24] %Nodes [22 23 24]
269     e=e+1;
270     strain(:,:,i)=B3*do(topology(e,:));
271 end
272
273 e=10;
274 for i=[25:27] %Nodes [25 26 27]
275     e=e+1;
276     strain(:,:,i)=B3*do(topology(e,:));
277 end
278
279 B4=subs(B,s,1);
280 B4=subs(B4,t,-1);
281 B4=double(B4);
282 strain(:,:,28)=B4*do(topology(11,:)); %Node 25
283
284 %Stress at nodes (sigma x, sigma y,tau xy)
285 stress=zeros(3,1,13);
286 for i=1:13
287     stress(:,:,i)=D*strain(:,:,i)

```

```

288 end
289
290 %Normal modes and corresponding natural frequencies
291 [vecfreq , freq]=eig(kglobal ,Mglob);
292 freq=diag(freq);
293 [freq ,I1]=sort(freq , 'ascend'); %Sorted eigenvalues
294 vecfreq=vecfreq(I1 ,:); %Sorted eigenvalues
295 freq=sqrt(freq); %UNITS : rad per sec
296 freqHz=freq/(2*pi); %UNITS : Hertz
297
298 %Constrained modes and corresponding natural frequencies
299 [vecfreqc , freqc]=eig(Kglobactive ,Mglobactive);
300 freqc=diag(freqc);
301 [freqc ,I2]=sort(freqc , 'ascend'); %Sorted eigenvalues
302 vecfreqc=vecfreqc(I2 ,:); %Sorted eigenvalues
303 freqc=sqrt(freqc); %UNITS : rad per sec
304 freqcHz=freqc/(2*pi); %UNITS : Hertz
305
306 %Constrained modes and corresponding natural frequencies
307 %with lumped mass matrix
308 [vecfreqcLum , freqcLum]=eig(Kglobactive ,MglobLumactive);
309 freqcLum=diag(freqcLum);
310 [freqcLum ,I3]=sort(freqcLum , 'ascend'); %Sorted eigenvalues
311 vecfreqcLum=vecfreqcLum(I3 ,:); %Sorted eigenvalues
312 freqcLum=sqrt(freqcLum); %UNITS : rad per sec
313 freqcLumHz=freqcLum/(2*pi); % UNITS : Hertz
314
315 %Rayleigh damping model
316 cglob1=0.001*Kglobactive+0.02*Mglobactive;
317 em=Mglobactive;
318 ka=Kglobactive;
319 fe=Fglobactive;
320 ce=cglob1;
321 [mg1 ,junk1]=size(Mglobactive);
322
323 %INTEGRATION USING NEWMARK METHOD
324
325 %%Central difference formula
326 % gama=0.5;
327 % beta=0;
328
329 %%trapezoidal rule
330 % gama=0.5;
331 % beta=0.25;
332
333 % Damped Newmark Method
334 gama=0.6;
335 beta=0.3025;
336
337 %% Linear Acceleration
338 % gama=0.5;
339 % beta=1/6;
340
341 %% Fox-Goodwin
342 % gama=0.5;
343 % beta=1/12;
344
345 ksi=(0.001/max(freqc)+0.02/max(freqc))/2;

```

```

346 om=(ksi*(gama-0.5)+sqrt(gama/2-beta+ksi^2*(gama-0.5)^2))/(gama/2-beta);
347 deltacrit=om/max(freqc);
348 deltat=1e-4;
349 delta=deltat;
350
351 dold=zeros(mgl,1);
352 vold=zeros(mgl,1);
353 say=0;
354 say=say+1;
355
356 rubbish=inv(em+gama*delta*ce+beta*delta*delta*ka);
357 aold=em/(fe-ce*vold-ka*dold);
358 dtildanew=dold+delta*vold+0.5*delta*delta*(1-2*beta)*aold;
359 vtildanew=vold+(1-gama)*delta*aold;
360 anew=rubbish*(fe-ce*vtildanew-ka*dtildanew);
361 dnew=dtildanew+beta*delta*delta*anew;
362 vnew=vtildanew+gama*delta*anew;
363 aold=anew;
364 vold=vnew;
365 dold=dnew;
366 dtildaold=dtildanew;
367 vtildaold=vtildanew;
368 time1(say)=0;
369 RightTopCornerdisplacement(say)=0;
370 RightTopCornerdisplacement2(say)=0;
371 RightTopCornerdisplacement3(say)=0;
372 LeftTopCornerdisplacement(say)=0;
373 for i=1:1000
374     say=say+1;
375     time1(say)=time1(say-1)+delta;
376     dtildanew=dold+delta*vold+0.5*delta*delta*(1-2*beta)*aold;
377     vtildanew=vold+(1-gama)*delta*aold;
378     anew=rubbish*(fe-ce*vtildanew-ka*dtildanew);
379     dnew=dtildanew+beta*delta*delta*anew;
380     vnew=vtildanew+gama*delta*anew;
381     aold=anew;
382     vold=vnew;
383     dold=dnew;
384     dtildaold=dtildanew;
385     vtildaold=vtildanew;
386     df(activeDof)=dold;
387     RightTopCornerdisplacement(say)=df(54);
388     RightTopCornerdisplacement2(say)=df(52);
389     RightTopCornerdisplacement3(say)=df(50);
390     LeftTopCornerdisplacement(say)=df(24);
391 end
392
393 toc
394
395 figure, plot(time1, RightTopCornerdisplacement)
396 xlabel('time (sec)')
397 ylabel('Right Top Cornerdisplacement (mm)')
398 % figure, plot(time1, RightTopCornerdisplacement2)
399 % xlabel('time (sec)')
400 % ylabel('Right Top Cornerdisplacement 2 (mm)')
401 % figure, plot(time1, RightTopCornerdisplacement3)
402 % xlabel('time (sec)')
403 % ylabel('Right Top Cornerdisplacement 3 (mm)')

```



```

404 % figure , plot (time1 , LeftTopCornerdisplacement )
405 % xlabel ('time (sec)')
406 % ylabel ('Left Top Cornerdisplacement (mm)' )

```

Listing 2: Matlab code for question 2

```

1 %Final Q2
2
3 clear all; close all; clc; format short;
4
5 % width=5; left =1.5; right=2.5; %dimensions (m)
6 %
7 % seed_x=6;
8 % seed_y=3;
9
10 num_node=28; %number of total nodes
11 num_elem=13; %number of total elements
12
13 element_type=4; %number of nodes on each element
14 ndof=1; %degrees-of-freedom per node
15
16 num_eq=num_node*ndof; %number of equations
17
18 z=[1 7 13 19];
19 k=100;
20
21 %%-----Coordinates-----%%
22
23 x_coord=[0 10 20 30 40 50 0 10 20 31 40 50 40 50 40 50 40 50
24 60 70 80 60 70 80 70 80 70 80];
25
26 y_coord=[0 0 0 0 0 0 10 10 10 10 10 10 -10 -10 -20 -20 -30 -30
27 -30 -30 -30 -20 -20 -20 -10 -10 0 0];
28
29 Coords=[x_coord', y_coord'];
30
31 %%-----topology matrix-----%%
32
33 topology=[ 1 2 8 7
34 2 3 9 8
35 3 4 10 9
36 4 5 11 10
37 5 6 12 11
38 13 14 6 5
39 15 16 14 13
40 17 18 16 15
41 18 19 22 16
42 19 20 23 22
43 20 21 24 23
44 23 24 26 25
45 25 26 28 27];
46
47 %%-----Gauss Quadrature and Shape function -----%%
48
49 num_gauss_point=2;
50
51 if num_gauss_point==1
52 gp=0;
53 w=2;

```

```

52     elseif num_gauss_point==2
53         %if four gauss points are used (two in each direction)
54         gp=[-0.57735027, 0.57735027]; %[eta psi]
55         w= [1 ,1];
56     end
57
58     eta=gp(1);
59     psi=gp(2);
60
61     N=(1/4)*[(1-psi)*(1-eta) (1+psi)*(1-eta) (1+psi)*(1+eta) (1-psi)*(1+eta)];
62     %shape functions
63
64     dN=(1/4)*[eta-1 1-eta 1+eta -eta-1; % Gradient
65              psi-1 -psi-1 1+psi 1-psi];
66
67     %%——Obtaining Element Stiffness Matrix——%%
68
69     ke=zeros(element_type,element_type);
70     kglobal=zeros(num_eq);
71
72     for e=1:1:num_elem
73
74         elem_top=[topology(e,:)];
75         J=dN*Coords(elem_top',:); % compute Jacobian matrix
76         detJ=det(J); % Jacobian
77         B=J\dN; % compute the B matrix
78         D=k*eye(2);
79         for i=1:num_gauss_point
80             for j=1:num_gauss_point
81                 ke=ke+w(i)*w(j)*B'*D*B*detJ; % element conductance matrix
82                 kglobal(elem_top,elem_top)=kglobal(elem_top,elem_top)+ke;
83             end
84         end
85     end
86
87     % At node 1,2,3,4,5,6,7,12,13 and 18, the temperature is to be fixed
88     fix_temperature_nodes=[1 7 21 24 26 28];
89
90     % The temperature is to be fixed at a particular value ( C )
91     fix_temperature=[20 20 50 50 50 50];
92
93     % Nodal force(heat source) vector
94     nodal_force{1}=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
95
96     % 4) LINEAR 2-D HEAT CONDUCTION ANALYSIS
97     % 4.1) Matching vector for obtaining global matrix
98
99     % All DoF
100    all_DoF_number=(1:num_node)';
101
102    % Restrained DoF
103    restrained_DoF_number_for_matching=fix_temperature_nodes';
104
105    % Unrestrained DoF
106    unrestrained_DoF_number_for_matching=all_DoF_number;
107    unrestrained_DoF_number_for_matching(restrained_DoF_number_for_matching)...
108    =[];
109

```

```

110 matching_vector_first=[unrestrained_DoF_number_for_matching;
111 restrained_DoF_number_for_matching];           % Matching vector for reorganize
112
113 for i=1:length(matching_vector_first)
114     matching_vector_end(i,1)=find(matching_vector_first==i);
115     % Matching vector for orijinal position
116 end
117
118 % 4.2) Reorganize nodal force and stiffness matrix
119 nodal_force_reorganize=nodal_force{1}(matching_vector_first);
120
121 stiffness_of_system_reorganize = ...
122     kglobal(matching_vector_first, matching_vector_first);
123
124 % 4.3) Linear solution with finite element method
125 % a) Input
126
127 % a1) Forces
128 F_A=nodal_force_reorganize(1:length(unrestrained_DoF_number_for_matching));
129
130 % a2) Stiffness matrix partition
131 K_AA=stiffness_of_system_reorganize ...
132     ([1:length(unrestrained_DoF_number_for_matching)], ...
133     [1:length(unrestrained_DoF_number_for_matching)]);
134
135 K_AB=stiffness_of_system_reorganize ...
136     ([1:length(unrestrained_DoF_number_for_matching)], ...
137     [length(unrestrained_DoF_number_for_matching)+1:end]);
138
139 K_BA=stiffness_of_system_reorganize ...
140     ([length(unrestrained_DoF_number_for_matching)+1:end], ...
141     [1:length(unrestrained_DoF_number_for_matching)]);
142
143 K_BB=stiffness_of_system_reorganize ...
144     ([length(unrestrained_DoF_number_for_matching)+1:end], ...
145     [length(unrestrained_DoF_number_for_matching)+1:end]);
146
147 % a3) Fix temperture ( F )
148 D_B=fix_temperature';
149
150 % b) Solution
151 % Specify nodal temperture ( C )
152 D_A=K_AA\F_A-K_AB*D_B;
153 % Specify nodal in-bound heat fluxes(Watts)
154 F_B_heat_fluxes=K_BA*D_A+K_BB*D_B;
155
156 % 4.4) Orijinal position temperture and heat fluxes
157
158 temperature_of_system_reorganized=[D_A;D_B];
159
160 temperature_of_system_orijinal_position ...
161     =temperature_of_system_reorganized(matching_vector_end)
162 heat_fluxes_of_system_orijinal_position_{1}...
163     =nodal_force{1};
164
165 heat_fluxes_of_system_orijinal_position_{1}...
166     (restrained_DoF_number_for_matching)...
167     =heat_fluxes_of_system_orijinal_position_{1}...

```

```
168     (restrained_DoF_number_for_matching)+F_B_heat_fluxes;  
169  
170 heat_fluxes_of_system_orijinal_position...  
171     =heat_fluxes_of_system_orijinal_position_{1}
```