# The List of Annexes

## 1.1 Program designed in MATLAB® using Bernoulli's method: DP_Trubka.m

```
clear all
close all
clc
%Vektor uhlu vlaken
Alfa=[0  5  15  25  35  45  55  65  75  85  90];

%Vektor prumeru
D=(1:20)*1e-3;

%Pocet promennych
K=size(D);
K=K(2);

Q=size(Alfa);
Q=Q(2);

%Pole pro vysledky
V_vysledky=zeros(K,Q);
W_vysledky=zeros(K,Q);

for k=1:K
   for q=1:Q


%VSTUPY
%sila F [N]
F=100;

%GEOMETRIE
%delka l [m]
l=1;
%polomery -  vnitrni, vnejsi [m]
r1=D(k)/2;
%r2=(30e-3)/2;

%EL, Et, Glt, vlt, alfa, t[m] v tabulce/  matici - poradi je zavazne
V=[156.05e9  6.045e9  4.431e9  0.328  90  1e-3;
   156.05e9  6.045e9  4.431e9  0.328  Alfa(q)  1e-3;
   156.05e9  6.045e9  4.431e9  0.328  -Alfa(q)  1e-3;]; %- V jako vstupy
```

```
%N- pocet vrstev
N=size(V,1);
%soucinitel beta
beta= 1 ;

%pole pro ulozeni dat pro jdn. vrstvy - mozna nadbytecne
E=zeros(3,3,N);
T=zeros(3,3,N);
%J=zeros(N,1);
%Ex=zeros(N,1);
Cxy=zeros(3,3,N);
Gxy=zeros(N,1);
A=zeros(N,1);

%vsechny potrebne prumery
d=zeros(N+1,1);
d(1)=2*r1;

for i=1:N

    %sestaveni Ci
    e11=V(i,1)/(1-(V(i,2)/V(i,1))*V(i,4)^2);
    e22=e11*(V(i,2)/V(i,1));
    e12=V(i,4)*e22;
    e21=e12;
    e66=V(i,3);

    E(:,:,i)=[e11 e12 0;
              e21 e22 0;
              0     0    e66];

    %sestaveni Ti
    alfa=V(i,5)/180*pi;

    T(:,:,i)=[ (cos(alfa))^2              (sin(alfa))^2             -
2*sin(alfa)*cos(alfa);
          (sin(alfa))^2              (cos(alfa))^2             2*sin(alfa)*cos(alfa);
          sin(alfa)*cos(alfa)    -sin(alfa)*cos(alfa)   (cos(alfa))^2-(sin(alfa))^2];


    %sestaveni Ji
    d(i+1)=d(i)+2*V(i,6);
    J(i)=(pi*(d(i+1)^4/64))*(1-(d(i)/d(i+1))^4);
```

```
    %transformace Ei do systemu xy
    Exy(:,:,i)=T(:,:,i)*E(:,:,i)*T(:,:,i)';

    %Matice poddajnosti
    Cxy(:,:,i)=inv(Exy(:,:,i));

    %Vektor hodnot modulu pruznosti v tahu Exi =[Ex1 Ex2 ....ExN]
    exi=1/Cxy(1,1,i);
    Ex(i)=exi;

     %plocha pro Gxy*A
    A(i)=(pi()/4)*(d(i+1)^2-d(i)^2);

    %modul pruznosti ve smyku
    gxyi=1/Cxy(3,3,i);
    Gxy(i)=gxyi;

    end

      %Ohybova tuhost
      EJ=Ex*J' ;%vychazim z toho, ze fce zeros generuje radkove
vektory(?)

      %Soucin Gxy*A
      GxyA=Gxy'*A;

      %beta/Gxy*A
      betaGA=beta./GxyA;

      %PRUHYB
      x=0:l/10:l;

      %vypocet Mohr. integral
      v=(1/EJ)*((F*l^2*x)-(F*l*x.^2)-(F*x.^3)/3);

      %vypocet se zahrnutim smyku
      w=(1/EJ)*((F*l^2*x)-(F*l*x.^2)-(F*x.^3)/3)+betaGA*F*x;

      %Tabulky vysledku - matice

      V_vysledky(k,q)=v(1);
      W_vysledky(k,q)=w(1);
    end
end
%save ('DP_Trubka_uhly_prumery', 'V_vysledky', 'W_vysledky'  )
```

## 1.2 Program designed in MATLAB® using ABD matrices: DP_ABD_Trubka.m

```
clc
clear all
close all
%% Vstupy
 %Vektor uhlu vlaken
Alfa=[0  5  15  25  35  45  55  65  75  85  90];

%Vektor prumeru
P=(1:10)*1e-3;

%Pocet promennych
K=size(P);
K=K(2);

U=size(Alfa);
U=U(2);

%Pole pro vysledky
V_vysledky=zeros(K,U);
W_vysledky=zeros(K,U);

for s=1:K
    for t=1:U

%VSTUPZ
%Sila
F=100;

%GEOMETRIE
%delka l [m]
l=1;
%polomery -  vnitrni, vnejsi [m]
r1=P(s)/2;
%r2=(30e-3)/2;

% Matice vstupu pro jdntl. vrstvy
%    1     2      3       4         5            6
%   EL   ET   GLT   vLT   alfa(°)     t(m)
V=[156.05e9  6.045e9  4.431e9  0.328   90  1e-3;
     156.05e9  6.045e9  4.431e9  0.328  Alfa(t)   1e-3;
     156.05e9  6.045e9  4.431e9  0.328  -Alfa(t)  1e-3;]; %- V jako vstupy
```

```
%pocet vrstev k
xx=size(V);
k=xx(1);

%pole pro ukladani mezivypoctu
Tvx=zeros(6,6,k);
c=zeros(6,6,k);
C=zeros(3,3,k);
Cxy=zeros(3,3,k);
Q=zeros(3,3,k);
q=zeros(6,6,k);
A=zeros(3,3);
B=zeros(3,3);
D=zeros(3,3);

%% Matice tuhosti
for i=1:k

c(1,1,i)=V(i,1)/(1-((V(i,2)/V(i,1))*V(i,4)^2));
c(2,2,i)=V(i,2)/V(i,1)*c(1,1,i);
c(1,2,i)=V(i,4)*c(2,2,i);
c(2,1,i)=c(1,2,i);
c(6,6,i)=V(i,3);

C(:,:,i)=[c(1,1,i) c(1,2,i) 0;
          c(2,1,i) c(2,2,i) 0;
          0        0    c(6,6,i);];

%% Transformace matice tuhosti

% Transformacni matice ze systemu LTt do systemu xyz
   alfa=V(i,5)/180*pi;

   Ts(:,:,i)=[ (cos(alfa))^2              (sin(alfa))^2
2*sin(alfa)*cos(alfa);
        (sin(alfa))^2              (cos(alfa))^2              -2*sin(alfa)*cos(alfa);
        -sin(alfa)*cos(alfa)    sin(alfa)*cos(alfa)   (cos(alfa))^2-(sin(alfa))^2];

  Te(:,:,i)=[ (cos(alfa))^2              (sin(alfa))^2
sin(alfa)*cos(alfa);
        (sin(alfa))^2              (cos(alfa))^2              -
sin(alfa)*cos(alfa);
        -2*sin(alfa)*cos(alfa)    2*sin(alfa)*cos(alfa)   (cos(alfa))^2-
(sin(alfa))^2];
```

```
Cxy(:,:,i)=inv(Ts(:,:,i))*C(:,:,i)*Te(:,:,i);%(Tvx)^-1*C*Tvx' ????? \Tvx !!!!!=Q
pro rovinnou napjatost

end

%% Matice mimoose tuhosti
% for i=1:k
%     alfa=V(i,5)/180*pi;
%
q(1,1,i)=C(1,1,i)*(cos(alfa))^4+C(2,2,i)*(sin(alfa))^4+2*(C(1,2,i)+2*C(3,3,i)
)*(sin(alfa))^2*(cos(alfa))^2;
%           q(1,2,i)=(C(1,1,i)+C(2,2,i)-
4*C(3,3,i))*(sin(alfa))^2*(cos(alfa))^2+C(1,2,i)*((sin(alfa))^4+(cos(alfa))^4)
;
%           q(2,1,i)=q(1,2,i);
%           q(1,6,i)=(C(1,1,i)-C(1,2,i)-
2*C(3,3,i))*(sin(alfa))*(cos(alfa))^3+(C(1,2,i)-
C(2,2,i)+2*C(3,3,i))*(sin(alfa))^3*(cos(alfa));
%           q(6,1,i)=q(1,6,i);
%
q(2,2,i)=C(1,1,i)*(sin(alfa))^4+2*(C(1,2,i)+2*C(3,3,i))*(sin(alfa))^2*(cos(alf
a))^2+C(2,2,i)*(cos(alfa))^4;
%           q(2,6,i)=(C(1,1,i)-C(1,2,i)-
2*C(3,3,i))*(sin(alfa))^3*(cos(alfa))+(C(1,2,i)-
C(2,2,i)+2*C(3,3,i))*(sin(alfa))*(cos(alfa))^3;
%           q(6,2,i)=q(2,6,i);
%           q(6,6,i)=(C(1,1,i)+C(2,2,i)-
2*(C(1,2,i)+C(3,3,i)))*(sin(alfa))^2*(cos(alfa))^2+C(3,3,i)*((sin(alfa))^4+(c
os(alfa))^4);
%
%     Q(:,:,i)=[q(1,1,i)  q(1,2,i)  q(1,6,i);
%               q(2,1,i)  q(2,2,i)  q(2,6,i);
%               q(6,1,i)  q(6,2,i)  q(6,6,i);];
% end

%% Matice ABD
  h(1)=-sum(V(:,6))/2;%h0

for n=1:k;
       for m=1:k;
          for i=1:k;

              %Vsechna h
                h(i+1)=(h(1)+sum(V(1:i,6)));
```

```
            a(i)=(Cxy(n,m,i)*(h(i+1)-h(i)));
            b(i)=((1/2)*Cxy(n,m,i)*(h(i+1)^2-h(i)^2));
            d(i)=((1/3)*Cxy(n,m,i)*(h(i+1)^3-h(i)^3));
          end

A(n,m)=sum(a);
B(n,m)=sum(b);
D(n,m)=sum(d);
       end
end

% Aa=A^(-1);
% Bb=Aa*B;
% Cc=-Bb';
% Dd=D-B\A*B;
%
% AA=Aa+Bb\D*Bb';
% BB=Bb\Dd;
% DD=Dd^(-1);
%
% ABD_T=[AA  BB;
%                BB DD];
%
% deformace=ABD_T*[0 0 F 0 0 0]';

%Modul pruznosti v tahu
E=(A(1,1)-A(1,2:3)*(A(2:3,2:3)^(-1)*A(2:3,1)))/sum(V(:,6));

%Modul pruznosti ve smyku
G=(A(3,3)-A(3,1:2)*(A(1:2,1:2)^(-1)*A(1:2,3)))/sum(V(:,6));

%vsechny potrebne prumery
dd=zeros(k+1,1);
dd(1)=2*r1;

%sestaveni Ji a GA
  for i=1:k
    dd(i+1)=dd(i)+2*V(i,6);
    j(i)=(pi*(dd(i+1)^4/64))*(1-(dd(i)/dd(i+1))^4);

    %plocha pro G*A
        S(i)=(pi/4)*(dd(i+1)^2-dd(i)^2);

    %modul pruznosti ve smyku
```

```
%     gxyi=1/Cxy(6,6,i);
%     Gxy(i)=gxyi;

  end

  J=sum(j);
 S_celk=sum(S);

  %PRUHYB
     x=0:l/10:l;

     %vypocet Mohr. integral
     v=(1/(E*J))*((F*l^3)/3-(F*l^2*x)/2-(F*x.^3)/3+(F*x.^3)/2);

    %vypocet se zahrnutim smyku

     %soucinitel beta
      beta= 1 ;

       %Soucin Gxy*A

     GxyA=G*S_celk;
     %beta/Gxy*A
     betaGA=beta/GxyA;

  w=(1/(E*J))*((F*l^3)/3-(F*l^2*x)/2-(F*x.^3)/3+(F*x.^3)/2)+betaGA*F*(l-x);

     %Tabulky vysledku - matice
     V_vysledky_G(s,t)=v(1);
     W_vysledky_G(s,t)=w(1);

    end
end

save('DP_ABD_Trubka_uhly_prumery', 'V_vysledky_G', 'W_vysledky_G')
```

## 1.3 Script for FEM model using conventional shell

```
#prumer
d=0.008
x=d*0.1
r=d/2
#uhel
a=0

# -*- coding: mbcs -*-
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *

mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0)
mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(
   0.0, 0.0), point1=(-0.1, 0.0))
mdb.models['Model-1'].sketches['__profile__'].RadialDimension(curve=
   mdb.models['Model-1'].sketches['__profile__'].geometry[2], radius=0.005,
   textPoint=(0.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].delete(objectList=(
   mdb.models['Model-1'].sketches['__profile__'].dimensions[0], ))
mdb.models['Model-1'].sketches['__profile__'].RadialDimension(curve=
   mdb.models['Model-1'].sketches['__profile__'].geometry[2], radius=r,
   textPoint=(0.0, 0.0))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-1', type=
   DEFORMABLE_BODY)
mdb.models['Model-1'].parts['Part-1'].BaseShellExtrude(depth=1.0, sketch=
   mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']
mdb.models['Model-1'].parts['Part-1'].DatumCsysByThreePoints(coordSysType=
   CARTESIAN, name='Datum csys-1', origin=
   mdb.models['Model-1'].parts['Part-1'].InterestingPoint(
   mdb.models['Model-1'].parts['Part-1'].edges[0], CENTER), point1=
   mdb.models['Model-1'].parts['Part-1'].InterestingPoint(
   mdb.models['Model-1'].parts['Part-1'].edges[1], CENTER), point2=
   mdb.models['Model-1'].parts['Part-1'].InterestingPoint(
   mdb.models['Model-1'].parts['Part-1'].edges[0], MIDDLE))
mdb.models['Model-1'].Material(name='Material-1')
mdb.models['Model-1'].materials['Material-1'].Density(table=((1474.0, ), ))
```

```
mdb.models['Model-1'].materials['Material-1'].Elastic(table=((156050000000.0,
   6045000000.0, 0.328, 4431000000.0, 4431000000.0, 4431000000.0), ), type=
   LAMINA)
mdb.models['Model-1'].parts['Part-1'].DatumPlaneByThreePoints(point1=
   mdb.models['Model-1'].parts['Part-1'].vertices[0], point2=
   mdb.models['Model-1'].parts['Part-1'].InterestingPoint(
   mdb.models['Model-1'].parts['Part-1'].edges[0], MIDDLE), point3=
   mdb.models['Model-1'].parts['Part-1'].InterestingPoint(
   mdb.models['Model-1'].parts['Part-1'].edges[1], MIDDLE))
mdb.models['Model-1'].parts['Part-1'].PartitionFaceByDatumPlane(datumPlane=
   mdb.models['Model-1'].parts['Part-1'].datums[3], faces=
   mdb.models['Model-1'].parts['Part-1'].faces.getSequenceFromMask(('[#1 ]',     ), ))
mdb.models['Model-1'].parts['Part-1'].CompositeLayup(description='',
   elementType=SHELL, name='CompositeLayup-1', offsetType=BOTTOM_SURFACE,
   symmetric=False, thicknessAssignment=FROM_SECTION)
mdb.models['Model-1'].parts['Part-1'].compositeLayups['CompositeLayup-1'].Section(
   integrationRule=SIMPSON, poissonDefinition=DEFAULT, preIntegrate=OFF,
   temperature=GRADIENT, thicknessType=UNIFORM, useDensity=OFF)
mdb.models['Model-1'].parts['Part-1'].compositeLayups['CompositeLayup-1'].ReferenceOrientation(
   additionalRotationField='', additionalRotationType=ROTATION_NONE, angle=0.0
   , axis=AXIS_2, fieldName='', localCsys=
   mdb.models['Model-1'].parts['Part-1'].datums[2], orientationType=SYSTEM)
mdb.models['Model-1'].parts['Part-1'].compositeLayups['CompositeLayup-1'].CompositePly(
   additionalRotationField='', additionalRotationType=ROTATION_ANGLE, angle=
   90.0, axis=AXIS_2, material='Material-1', numIntPoints=3, orientation=
   mdb.models['Model-1'].parts['Part-1'].datums[2], orientationType=CSYS,
   plyName='Ply-1', region=Region(
   faces=mdb.models['Model-1'].parts['Part-1'].faces.getSequenceFromMask(
   mask=('[#3 ]', ), )), suppressed=False, thickness=0.001, thicknessType=
   SPECIFY_THICKNESS)
mdb.models['Model-1'].parts['Part-1'].compositeLayups['CompositeLayup-1'].CompositePly(
   additionalRotationField='', additionalRotationType=ROTATION_NONE, angle=a
   , axis=AXIS_2, material='Material-1', numIntPoints=3, orientation=
   mdb.models['Model-1'].parts['Part-1'].datums[2], orientationType=CSYS,
   plyName='Ply-2', region=Region(
   faces=mdb.models['Model-1'].parts['Part-1'].faces.getSequenceFromMask(
   mask=('[#3 ]', ), )), suppressed=False, thickness=0.001, thicknessType=
   SPECIFY_THICKNESS)
mdb.models['Model-1'].parts['Part-1'].compositeLayups['CompositeLayup-1'].CompositePly(
   additionalRotationField='', additionalRotationType=ROTATION_NONE, angle=-a
   , axis=AXIS_2, material='Material-1', numIntPoints=3, orientation=
   mdb.models['Model-1'].parts['Part-1'].datums[2], orientationType=CSYS,
   plyName='Ply-3', region=Region(
   faces=mdb.models['Model-1'].parts['Part-1'].faces.getSequenceFromMask(
   mask=('[#3 ]', ), )), suppressed=False, thickness=0.001, thicknessType=
   SPECIFY_THICKNESS)
mdb.models['Model-1'].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models['Model-1'].rootAssembly.Instance(dependent=OFF, name='Part-1-1',
   part=mdb.models['Model-1'].parts['Part-1'])
mdb.models['Model-1'].StaticStep(name='Step-1', previous='Initial')
mdb.models['Model-1'].rootAssembly.ReferencePoint(point=
   mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].InterestingPoint(
```

```
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].edges[3], CENTER))
mdb.models['Model-1'].rootAssembly.Set(name='m_Set-1', referencePoints=(
    mdb.models['Model-1'].rootAssembly.referencePoints[4], ))
mdb.models['Model-1'].rootAssembly.Set(edges=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].edges.getSequenceFromMask(
    ('[#18 ]', ), ), name='s_Set-1')
mdb.models['Model-1'].Coupling(controlPoint=
    mdb.models['Model-1'].rootAssembly.sets['m_Set-1'], couplingType=KINEMATIC,
    influenceRadius=WHOLE_SURFACE, localCsys=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].datums[2], name=
    'Constraint-1', surface=mdb.models['Model-1'].rootAssembly.sets['s_Set-1'],
    u1=ON, u2=ON, u3=ON, ur1=ON, ur2=ON, ur3=ON)
mdb.models['Model-1'].fieldOutputRequests['F-Output-1'].setValues(variables=(
    'U', 'UT', 'UR'))
mdb.models['Model-1'].rootAssembly.Set(name='Set-3', referencePoints=(
    mdb.models['Model-1'].rootAssembly.referencePoints[4], ))
mdb.models['Model-1'].ConcentratedForce(cf3=100.0, createStepName='Step-1',
    distributionType=UNIFORM, field='', localCsys=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].datums[2], name=
    'Load-1', region=mdb.models['Model-1'].rootAssembly.sets['Set-3'])
mdb.models['Model-1'].rootAssembly.Set(edges=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].edges.getSequenceFromMask(
    ('[#22 ]', ), ), name='Set-4')
mdb.models['Model-1'].EncastreBC(createStepName='Step-1', localCsys=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].datums[2], name=
    'BC-1', region=mdb.models['Model-1'].rootAssembly.sets['Set-4'])
mdb.models['Model-1'].rootAssembly.setElementType(elemTypes=(ElemType(
    elemCode=S4R, elemLibrary=STANDARD, secondOrderAccuracy=OFF,
    hourglassControl=DEFAULT), ElemType(elemCode=S3, elemLibrary=STANDARD)),
    regions=(
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(
    ('[#3 ]', ), ), ))
mdb.models['Model-1'].rootAssembly.setMeshControls(regions=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(
    ('[#3 ]', ), ), technique=STRUCTURED)
mdb.models['Model-1'].rootAssembly.seedPartInstance(deviationFactor=0.1,
    minSizeFactor=0.1, regions=(
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'], ), size=0.005)
mdb.models['Model-1'].rootAssembly.seedEdgeBySize(constraint=FINER,
    deviationFactor=0.1, edges=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].edges.getSequenceFromMask(
    ('[#3a ]', ), ), minSizeFactor=0.1, size=x)
mdb.models['Model-1'].rootAssembly.generateMesh(regions=(
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'], ))
mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF,
    explicitPrecision=SINGLE, getMemoryFromAnalysis=True, historyPrint=OFF,
    memory=50, memoryUnits=PERCENTAGE, model='Model-1', modelPrint=OFF,
    multiprocessingMode=DEFAULT, name='Job-0', nodalOutputPrecision=SINGLE,
    numCpus=1, numGPUs=0, queue=None, scratch='', type=ANALYSIS,
    userSubroutine='', waitHours=0, waitMinutes=0)
mdb.jobs['Job-0'].submit(consistencyChecking=OFF)
```

```
mdb.jobs['Job-0']._Message(STARTED, {'phase': BATCHPRE_PHASE,
    'clientHost': 'ntb-HPPB4310s', 'handle': 0, 'jobName': 'Job-0'})
mdb.jobs['Job-0']._Message(WARNING, {'phase': BATCHPRE_PHASE,
    'message': 'WHENEVER A TRANSLATION (ROTATION) DOF AT A NODE IS CONSTRAINED BY A
KINEMATIC COUPLING DEFINITION THE TRANSLATION (ROTATION) DOFS FOR THAT NODE CANNOT
BE INCLUDED IN ANY OTHER CONSTRAINT INCLUDING MPCS, RIGID BODIES, ETC.',
    'jobName': 'Job-0'})
mdb.jobs['Job-0']._Message(WARNING, {'phase': BATCHPRE_PHASE,
    'message': 'MPCS (EXTERNAL or INTERNAL, including those generated from rigid body definitions),
KINEMATIC COUPLINGS, AND/OR EQUATIONS WILL ACTIVATE ADDITIONAL DEGREES OF FREEDOM',
    'jobName': 'Job-0'})
mdb.jobs['Job-0']._Message(ODB_FILE, {'phase': BATCHPRE_PHASE,
    'file': 'C:\\Temp\\Job-0.odb', 'jobName': 'Job-0'})
mdb.jobs['Job-0']._Message(COMPLETED, {'phase': BATCHPRE_PHASE,
    'message': 'Analysis phase complete', 'jobName': 'Job-0'})
mdb.jobs['Job-0']._Message(STARTED, {'phase': STANDARD_PHASE,
    'clientHost': 'ntb-HPPB4310s', 'handle': 5316, 'jobName': 'Job-0'})
mdb.jobs['Job-0']._Message(STEP, {'phase': STANDARD_PHASE, 'stepId': 1,
    'jobName': 'Job-0'})
mdb.jobs['Job-0']._Message(WARNING, {'phase': STANDARD_PHASE,
    'message': 'The 3-direction at one or more points in one or more layers in 3200 elements as
defined in *ORIENTATION are in the opposite direction to the element normals. Either the 1 or 2 and
the 3-direction defined in *ORIENTATION will be reversed. The elements have been identified in
element set WarnElem3DirOppElemNormalStep1Inc1.',
    'jobName': 'Job-0'})
mdb.jobs['Job-0']._Message(ODB_FRAME, {'phase': STANDARD_PHASE, 'step': 0,
    'frame': 0, 'jobName': 'Job-0'})
mdb.jobs['Job-0']._Message(STATUS, {'totalTime': 0.0, 'attempts': 0,
    'timeIncrement': 1.0, 'increment': 0, 'stepTime': 0.0, 'step': 1,
    'jobName': 'Job-0', 'severe': 0, 'iterations': 0, 'phase': STANDARD_PHASE,
    'equilibrium': 0})
mdb.jobs['Job-0']._Message(MEMORY_ESTIMATE, {'phase': STANDARD_PHASE,
    'jobName': 'Job-0', 'memory': 132.121262550354})
mdb.jobs['Job-0']._Message(ODB_FRAME, {'phase': STANDARD_PHASE, 'step': 0,
    'frame': 1, 'jobName': 'Job-0'})
mdb.jobs['Job-0']._Message(STATUS, {'totalTime': 1.0, 'attempts': 1,
    'timeIncrement': 1.0, 'increment': 1, 'stepTime': 1.0, 'step': 1,
    'jobName': 'Job-0', 'severe': 0, 'iterations': 2, 'phase': STANDARD_PHASE,
    'equilibrium': 2})
mdb.jobs['Job-0']._Message(END_STEP, {'phase': STANDARD_PHASE, 'stepId': 1,
    'jobName': 'Job-0'})
mdb.jobs['Job-0']._Message(COMPLETED, {'phase': STANDARD_PHASE,
    'message': 'Analysis phase complete', 'jobName': 'Job-0'})
mdb.jobs['Job-0']._Message(JOB_COMPLETED, {'time': 'Thu Mar 12 16:21:06 2015',
    'jobName': 'Job-0'})
# Save by user on 2015_03_12-16.23.10; build 6.12-1 2012_03_13-20.44.39 119612
```

## 1.4 Script for FEM model using continuum shell

```
#prumer
d=0.002
r=d/2
k=r+0.003
#uhel
a=90

# Save by user on 2015_03_11-14.24.50; build 6.12-1 2012_03_13-20.44.39 119612
from part import *
from material import *
from section import *
from optimization import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *

mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0)
mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(
   0.0, 0.0), point1=(-2.5, 0.0))
mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(
   0.0, 0.0), point1=(-1.25, 0.0))
mdb.models['Model-1'].sketches['__profile__'].RadialDimension(curve=
   mdb.models['Model-1'].sketches['__profile__'].geometry[2], radius=r,
   textPoint=(0.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].RadialDimension(curve=
   mdb.models['Model-1'].sketches['__profile__'].geometry[3], radius=k,
   textPoint=(0.0, 0.0))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-1', type=
   DEFORMABLE_BODY)
mdb.models['Model-1'].parts['Part-1'].BaseSolidExtrude(depth=1.0, sketch=
   mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']
mdb.models['Model-1'].Material(name='Material-1')
mdb.models['Model-1'].materials['Material-1'].Density(table=((1474.0, ), ))
mdb.models['Model-1'].materials['Material-1'].Elastic(table=((156050000000.0,
   6045000000.0, 0.328, 4431000000.0, 4431000000.0, 4431000000.0), ), type=
   LAMINA)
mdb.models['Model-1'].parts['Part-1'].DatumPlaneByThreePoints(point1=
   mdb.models['Model-1'].parts['Part-1'].vertices[2], point2=
   mdb.models['Model-1'].parts['Part-1'].InterestingPoint(
   mdb.models['Model-1'].parts['Part-1'].edges[2], MIDDLE), point3=
   mdb.models['Model-1'].parts['Part-1'].InterestingPoint(
   mdb.models['Model-1'].parts['Part-1'].edges[3], MIDDLE))
```

```
mdb.models['Model-1'].parts['Part-1'].PartitionCellByDatumPlane(cells=
   mdb.models['Model-1'].parts['Part-1'].cells.getSequenceFromMask(('[#1 ]',
   ), ), datumPlane=mdb.models['Model-1'].parts['Part-1'].datums[2])
mdb.models['Model-1'].parts['Part-1'].DatumCsysByThreePoints(coordSysType=
   CARTESIAN, name='Datum csys-1', origin=
   mdb.models['Model-1'].parts['Part-1'].InterestingPoint(
   mdb.models['Model-1'].parts['Part-1'].edges[10], CENTER), point1=
   mdb.models['Model-1'].parts['Part-1'].InterestingPoint(
   mdb.models['Model-1'].parts['Part-1'].edges[8], CENTER), point2=
   mdb.models['Model-1'].parts['Part-1'].InterestingPoint(
   mdb.models['Model-1'].parts['Part-1'].edges[11], MIDDLE))
mdb.models['Model-1'].parts['Part-1'].CompositeLayup(description='',
   elementType=CONTINUUM_SHELL, name='CompositeLayup-1', symmetric=False)
mdb.models['Model-1'].parts['Part-1'].compositeLayups['CompositeLayup-1'].Section(
   integrationRule=SIMPSON, poissonDefinition=DEFAULT, preIntegrate=OFF,
   temperature=GRADIENT, thicknessModulus=None, useDensity=OFF)
mdb.models['Model-1'].parts['Part-1'].compositeLayups['CompositeLayup-1'].ReferenceOrientation(
   additionalRotationField='', additionalRotationType=ROTATION_NONE, angle=0.0
   , axis=AXIS_2, fieldName='', localCsys=
   mdb.models['Model-1'].parts['Part-1'].datums[4], orientationType=SYSTEM,
   stackDirection=STACK_3)
mdb.models['Model-1'].parts['Part-1'].compositeLayups['CompositeLayup-1'].CompositePly(
   additionalRotationField='', additionalRotationType=ROTATION_NONE, angle=0.0
   , axis=AXIS_2, material='Material-1', numIntPoints=3, orientation=
   mdb.models['Model-1'].parts['Part-1'].datums[4], orientationType=CSYS,
   plyName='Ply-1', region=Region(
   cells=mdb.models['Model-1'].parts['Part-1'].cells.getSequenceFromMask(
   mask=('[#3 ]', ), )), suppressed=False, thickness=0.001, thicknessType=
   SPECIFY_THICKNESS)
mdb.models['Model-1'].parts['Part-1'].compositeLayups['CompositeLayup-1'].CompositePly(
   additionalRotationField='', additionalRotationType=ROTATION_NONE, angle=0.0
   , axis=AXIS_2, material='Material-1', numIntPoints=3, orientation=
   mdb.models['Model-1'].parts['Part-1'].datums[4], orientationType=CSYS,
   plyName='Ply-2', region=Region(
   cells=mdb.models['Model-1'].parts['Part-1'].cells.getSequenceFromMask(
   mask=('[#3 ]', ), )), suppressed=False, thickness=0.001, thicknessType=
   SPECIFY_THICKNESS)
mdb.models['Model-1'].parts['Part-1'].compositeLayups['CompositeLayup-1'].CompositePly(
   additionalRotationField='', additionalRotationType=ROTATION_NONE, angle=0.0
   , axis=AXIS_2, material='Material-1', numIntPoints=3, orientation=
   mdb.models['Model-1'].parts['Part-1'].datums[4], orientationType=CSYS,
   plyName='Ply-3', region=Region(
   cells=mdb.models['Model-1'].parts['Part-1'].cells.getSequenceFromMask(
   mask=('[#3 ]', ), )), suppressed=False, thickness=0.001, thicknessType=
   SPECIFY_THICKNESS)
mdb.models['Model-1'].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models['Model-1'].rootAssembly.Instance(dependent=OFF, name='Part-1-1',
   part=mdb.models['Model-1'].parts['Part-1'])
mdb.models['Model-1'].StaticStep(name='Step-1', previous='Initial')
mdb.models['Model-1'].fieldOutputRequests['F-Output-1'].setValues(variables=(
   'U', 'UT', 'UR', 'RF', 'CF'))
```

```
mdb.models['Model-1'].rootAssembly.ReferencePoint(point=
   mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].InterestingPoint(
   mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].edges[8], CENTER))
mdb.models['Model-1'].rootAssembly.Set(name='m_Set-1', referencePoints=(
   mdb.models['Model-1'].rootAssembly.referencePoints[4], ))
mdb.models['Model-1'].rootAssembly.Surface(name='s_Surf-1', side1Faces=
   mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(
   ('[#204 ]', ), ))
mdb.models['Model-1'].Coupling(controlPoint=
   mdb.models['Model-1'].rootAssembly.sets['m_Set-1'], couplingType=KINEMATIC,
   influenceRadius=WHOLE_SURFACE, localCsys=
   mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].datums[4], name=
   'Constraint-1', surface=
   mdb.models['Model-1'].rootAssembly.surfaces['s_Surf-1'], u1=ON, u2=ON, u3=
   ON, ur1=ON, ur2=ON, ur3=ON)
mdb.models['Model-1'].rootAssembly.Set(faces=
   mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(
   ('[#110 ]', ), ), name='Set-2')
mdb.models['Model-1'].EncastreBC(createStepName='Step-1', localCsys=
   mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].datums[4], name=
   'BC-1', region=mdb.models['Model-1'].rootAssembly.sets['Set-2'])
mdb.models['Model-1'].rootAssembly.Set(name='Set-3', referencePoints=(
   mdb.models['Model-1'].rootAssembly.referencePoints[4], ))
mdb.models['Model-1'].ConcentratedForce(cf2=100.0, createStepName='Step-1',
   distributionType=UNIFORM, field='', localCsys=
   mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].datums[4], name=
   'Load-1', region=mdb.models['Model-1'].rootAssembly.sets['Set-3'])
mdb.models['Model-1'].rootAssembly.setElementType(elemTypes=(ElemType(
   elemCode=SC8R, elemLibrary=STANDARD, secondOrderAccuracy=OFF,
   hourglassControl=DEFAULT), ElemType(elemCode=SC6R, elemLibrary=STANDARD),
   ElemType(elemCode=UNKNOWN_TET, elemLibrary=STANDARD)), regions=(
   mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].cells.getSequenceFromMask(
   ('[#3 ]', ), ), ))
mdb.models['Model-1'].rootAssembly.seedPartInstance(deviationFactor=0.1,
   minSizeFactor=0.1, regions=(
   mdb.models['Model-1'].rootAssembly.instances['Part-1-1'], ), size=0.05)
mdb.models['Model-1'].rootAssembly.seedPartInstance(deviationFactor=0.1,
   minSizeFactor=0.1, regions=(
   mdb.models['Model-1'].rootAssembly.instances['Part-1-1'], ), size=0.005)
mdb.models['Model-1'].rootAssembly.seedEdgeBySize(constraint=FINER,
   deviationFactor=0.1, edges=
   mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].edges.getSequenceFromMask(
   ('[#3ff00 ]', ), ), minSizeFactor=0.1, size=0.001)
mdb.models['Model-1'].rootAssembly.seedEdgeBySize(constraint=FINER,
   deviationFactor=0.1, edges=
   mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].edges.getSequenceFromMask(
   ('[#3ff00 ]', ), ), minSizeFactor=0.1, size=0.01)
mdb.models['Model-1'].rootAssembly.seedEdgeBySize(constraint=FINER,
   deviationFactor=0.1, edges=
   mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].edges.getSequenceFromMask(
   ('[#3ff00 ]', ), ), minSizeFactor=0.1, size=0.005)
```

```
mdb.models['Model-1'].rootAssembly.seedEdgeBySize(constraint=FINER,
    deviationFactor=0.1, edges=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].edges.getSequenceFromMask(
    ('[#3ff00 ]', ), ), minSizeFactor=0.1, size=0.0005)
mdb.models['Model-1'].rootAssembly.seedEdgeBySize(constraint=FINER,
    deviationFactor=0.1, edges=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].edges.getSequenceFromMask(
    ('[#3ff00 ]', ), ), minSizeFactor=0.1, size=0.001)
mdb.models['Model-1'].rootAssembly.generateMesh(regions=(
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'], ))
mdb.models['Model-1'].parts['Part-1'].compositeLayups['CompositeLayup-1'].deletePlies(    )
mdb.models['Model-1'].parts['Part-1'].compositeLayups['CompositeLayup-1'].CompositePly(
    additionalRotationField='', additionalRotationType=ROTATION_ANGLE, angle=
    90.0, axis=AXIS_2, material='Material-1', numIntPoints=3, orientation=
    mdb.models['Model-1'].parts['Part-1'].datums[4], orientationType=CSYS,
    plyName='Ply-1', region=Region(
    cells=mdb.models['Model-1'].parts['Part-1'].cells.getSequenceFromMask(
    mask=('[#3 ]', ), )), suppressed=False, thickness=0.001, thicknessType=
    SPECIFY_THICKNESS)
mdb.models['Model-1'].parts['Part-1'].compositeLayups['CompositeLayup-1'].CompositePly(
    additionalRotationField='', additionalRotationType=ROTATION_NONE, angle=a
    , axis=AXIS_2, material='Material-1', numIntPoints=3, orientation=
    mdb.models['Model-1'].parts['Part-1'].datums[4], orientationType=CSYS,
    plyName='Ply-2', region=Region(
    cells=mdb.models['Model-1'].parts['Part-1'].cells.getSequenceFromMask(
    mask=('[#3 ]', ), )), suppressed=False, thickness=0.001, thicknessType=
    SPECIFY_THICKNESS)
mdb.models['Model-1'].parts['Part-1'].compositeLayups['CompositeLayup-1'].CompositePly(
    additionalRotationField='', additionalRotationType=ROTATION_NONE, angle=-a
    , axis=AXIS_2, material='Material-1', numIntPoints=3, orientation=
    mdb.models['Model-1'].parts['Part-1'].datums[4], orientationType=CSYS,
    plyName='Ply-3', region=Region(
    cells=mdb.models['Model-1'].parts['Part-1'].cells.getSequenceFromMask(
    mask=('[#3 ]', ), )), suppressed=False, thickness=0.001, thicknessType=
    SPECIFY_THICKNESS)
mdb.models['Model-1'].rootAssembly.regenerate()
mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF,
    explicitPrecision=SINGLE, getMemoryFromAnalysis=True, historyPrint=OFF,
    memory=50, memoryUnits=PERCENTAGE, model='Model-1', modelPrint=OFF,
    multiprocessingMode=DEFAULT, name='Job-e', nodalOutputPrecision=SINGLE,
    numCpus=1, numGPUs=0, queue=None, scratch='', type=ANALYSIS,
    userSubroutine='', waitHours=0, waitMinutes=0)
mdb.jobs['Job-e'].submit(consistencyChecking=OFF)
mdb.jobs['Job-e']._Message(STARTED, {'phase': BATCHPRE_PHASE,
    'clientHost': 'ntb-HPPB4310s', 'handle': 0, 'jobName': 'Job-e'})
mdb.jobs['Job-e']._Message(WARNING, {'phase': BATCHPRE_PHASE,
    'message': 'WHENEVER A TRANSLATION (ROTATION) DOF AT A NODE IS CONSTRAINED BY A
KINEMATIC COUPLING DEFINITION THE TRANSLATION (ROTATION) DOFS FOR THAT NODE CANNOT
BE INCLUDED IN ANY OTHER CONSTRAINT INCLUDING MPCS, RIGID BODIES, ETC.',
    'jobName': 'Job-e'})
```

```
mdb.jobs['Job-e']._Message(WARNING, {'phase': BATCHPRE_PHASE,
    'message': 'MPCS (EXTERNAL or INTERNAL, including those generated from rigid body definitions),
KINEMATIC COUPLINGS, AND/OR EQUATIONS WILL ACTIVATE ADDITIONAL DEGREES OF FREEDOM',
    'jobName': 'Job-e'})
mdb.jobs['Job-e']._Message(ODB_FILE, {'phase': BATCHPRE_PHASE,
    'file': 'C:\\Temp\\Job-e.odb', 'jobName': 'Job-e'})
mdb.jobs['Job-e']._Message(COMPLETED, {'phase': BATCHPRE_PHASE,
    'message': 'Analysis phase complete', 'jobName': 'Job-e'})
mdb.jobs['Job-e']._Message(STARTED, {'phase': STANDARD_PHASE,
    'clientHost': 'ntb-HPPB4310s', 'handle': 3840, 'jobName': 'Job-e'})
mdb.jobs['Job-e']._Message(STEP, {'phase': STANDARD_PHASE, 'stepId': 1,
    'jobName': 'Job-e'})
mdb.jobs['Job-e']._Message(WARNING, {'phase': STANDARD_PHASE,
    'message': 'The 3-direction at one or more points in one or more layers in 8200 elements as
defined in *ORIENTATION are in the opposite direction to the element normals. Either the 1 or 2 and
the 3-direction defined in *ORIENTATION will be reversed. The elements have been identified in
element set WarnElem3DirOppElemNormalStep1Inc1.',
    'jobName': 'Job-e'})
mdb.jobs['Job-e']._Message(ODB_FRAME, {'phase': STANDARD_PHASE, 'step': 0,
    'frame': 0, 'jobName': 'Job-e'})
mdb.jobs['Job-e']._Message(STATUS, {'totalTime': 0.0, 'attempts': 0,
    'timeIncrement': 1.0, 'increment': 0, 'stepTime': 0.0, 'step': 1,
    'jobName': 'Job-e', 'severe': 0, 'iterations': 0, 'phase': STANDARD_PHASE,
    'equilibrium': 0})
mdb.jobs['Job-e']._Message(MEMORY_ESTIMATE, {'phase': STANDARD_PHASE,
    'jobName': 'Job-e', 'memory': 348.168928146362})
mdb.jobs['Job-e']._Message(ODB_FRAME, {'phase': STANDARD_PHASE, 'step': 0,
    'frame': 1, 'jobName': 'Job-e'})
mdb.jobs['Job-e']._Message(STATUS, {'totalTime': 1.0, 'attempts': 1,
    'timeIncrement': 1.0, 'increment': 1, 'stepTime': 1.0, 'step': 1,
    'jobName': 'Job-e', 'severe': 0, 'iterations': 1, 'phase': STANDARD_PHASE,
    'equilibrium': 1})
mdb.jobs['Job-e']._Message(END_STEP, {'phase': STANDARD_PHASE, 'stepId': 1,
    'jobName': 'Job-e'})
mdb.jobs['Job-e']._Message(COMPLETED, {'phase': STANDARD_PHASE,
    'message': 'Analysis phase complete', 'jobName': 'Job-e'})
mdb.jobs['Job-e']._Message(JOB_COMPLETED, {'time': 'Wed Mar 11 14:48:04 2015',
    'jobName': 'Job-e'})
# Save by user on 2015_03_11-14.50.17; build 6.12-1 2012_03_13-20.44.39 119612
```

## 1.5 Script for FEM model using volume model

```
#prumer
d=0.01

#uhel
a=55

#polomery
r1=d/2
r2=r1+0.001
r3=r2+0.001
r4=r3+0.001

x1=d*0.1

# -*- coding: mbcs -*-
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *

mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=0.2)
mdb.models['Model-1'].sketches['__profile__'].sketchOptions.setValues(
    decimalPlaces=3)
mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(
    0.0, 0.0), point1=(-0.005, 0.0))
mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(
    0.0, 0.0), point1=(-0.01, 0.0))
mdb.models['Model-1'].sketches['__profile__'].RadialDimension(curve=
    mdb.models['Model-1'].sketches['__profile__'].geometry[2], radius=r1,
    textPoint=(0.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].RadialDimension(curve=
    mdb.models['Model-1'].sketches['__profile__'].geometry[3], radius=r2,
    textPoint=(0.0, 0.0))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-1', type=
    DEFORMABLE_BODY)
mdb.models['Model-1'].parts['Part-1'].BaseSolidExtrude(depth=1.0, sketch=
    mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']
mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=0.2)
mdb.models['Model-1'].sketches['__profile__'].sketchOptions.setValues(
    decimalPlaces=3)
```

```
mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(
   0.0, 0.0), point1=(-0.005, 0.0))
mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(
   0.0, 0.0), point1=(-0.01, 0.0))
mdb.models['Model-1'].sketches['__profile__'].RadialDimension(curve=
   mdb.models['Model-1'].sketches['__profile__'].geometry[2], radius=r2,
   textPoint=(0.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].RadialDimension(curve=
   mdb.models['Model-1'].sketches['__profile__'].geometry[3], radius=r3,
   textPoint=(0.0, 0.0))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-2', type=
   DEFORMABLE_BODY)
mdb.models['Model-1'].parts['Part-2'].BaseSolidExtrude(depth=1.0, sketch=
   mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']
mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=0.2)
mdb.models['Model-1'].sketches['__profile__'].sketchOptions.setValues(
   decimalPlaces=3)
mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(
   0.0, 0.0), point1=(-0.005, 0.0))
mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(
   0.0, 0.0), point1=(-0.01, 0.0))
mdb.models['Model-1'].sketches['__profile__'].RadialDimension(curve=
   mdb.models['Model-1'].sketches['__profile__'].geometry[2], radius=r3,
   textPoint=(0.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].RadialDimension(curve=
   mdb.models['Model-1'].sketches['__profile__'].geometry[3], radius=r4,
   textPoint=(0.0, 0.0))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-3', type=
   DEFORMABLE_BODY)
mdb.models['Model-1'].parts['Part-3'].BaseSolidExtrude(depth=1.0, sketch=
   mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']
mdb.models['Model-1'].Material(name='Material-1')
mdb.models['Model-1'].materials['Material-1'].Density(table=((1474.0, ), ))
mdb.models['Model-1'].materials['Material-1'].Elastic(table=((156050000000.0,
   6045000000.0, 6045000000.0, 0.328, 0.328, 0.328, 4431000000.0,
   4431000000.0, 4431000000.0), ), type=ENGINEERING_CONSTANTS)
mdb.models['Model-1'].CompositeSolidSection(layup=(SectionLayer(
   thickness=0.001, orientAngle=90.0, numIntPts=1, material='Material-1',
   plyName='Ply-1'), ), layupName='', name='Section-1', symmetric=False)
mdb.models['Model-1'].CompositeSolidSection(layup=(SectionLayer(
   thickness=0.001, orientAngle=a, numIntPts=1, material='Material-1',
   plyName='Ply-2'), ), layupName='', name='Section-2', symmetric=False)
mdb.models['Model-1'].CompositeSolidSection(layup=(SectionLayer(
   thickness=0.001, orientAngle=-a, numIntPts=1, material='Material-1',
   plyName='Ply-3'), ), layupName='', name='Section-3', symmetric=False)
mdb.models['Model-1'].parts['Part-1'].DatumCsysByThreePoints(coordSysType=
   CARTESIAN, name='Datum csys-1', origin=
   mdb.models['Model-1'].parts['Part-1'].InterestingPoint(
   mdb.models['Model-1'].parts['Part-1'].edges[0], CENTER), point1=
   mdb.models['Model-1'].parts['Part-1'].InterestingPoint(
```

```
    mdb.models['Model-1'].parts['Part-1'].edges[1], CENTER), point2=
    mdb.models['Model-1'].parts['Part-1'].InterestingPoint(
    mdb.models['Model-1'].parts['Part-1'].edges[0], MIDDLE))
mdb.models['Model-1'].parts['Part-2'].DatumCsysByThreePoints(coordSysType=
    CARTESIAN, name='Datum csys-1', origin=
    mdb.models['Model-1'].parts['Part-2'].InterestingPoint(
    mdb.models['Model-1'].parts['Part-2'].edges[0], CENTER), point1=
    mdb.models['Model-1'].parts['Part-2'].InterestingPoint(
    mdb.models['Model-1'].parts['Part-2'].edges[1], CENTER), point2=
    mdb.models['Model-1'].parts['Part-2'].InterestingPoint(
    mdb.models['Model-1'].parts['Part-2'].edges[0], MIDDLE))
mdb.models['Model-1'].parts['Part-3'].DatumCsysByThreePoints(coordSysType=
    CARTESIAN, name='Datum csys-1', origin=
    mdb.models['Model-1'].parts['Part-3'].InterestingPoint(
    mdb.models['Model-1'].parts['Part-3'].edges[0], CENTER), point1=
    mdb.models['Model-1'].parts['Part-3'].InterestingPoint(
    mdb.models['Model-1'].parts['Part-3'].edges[1], CENTER), point2=
    mdb.models['Model-1'].parts['Part-3'].InterestingPoint(
    mdb.models['Model-1'].parts['Part-3'].edges[0], MIDDLE))
mdb.models['Model-1'].parts['Part-1'].MaterialOrientation(
    additionalRotationField='', additionalRotationType=ROTATION_NONE, angle=0.0
    , axis=AXIS_3, fieldName='', localCsys=
    mdb.models['Model-1'].parts['Part-1'].datums[2], orientationType=SYSTEM,
    region=Region(
    cells=mdb.models['Model-1'].parts['Part-1'].cells.getSequenceFromMask(
    mask=('[#1 ]', ), )), stackDirection=STACK_3)
mdb.models['Model-1'].parts['Part-2'].MaterialOrientation(
    additionalRotationField='', additionalRotationType=ROTATION_NONE, angle=0.0
    , axis=AXIS_3, fieldName='', localCsys=
    mdb.models['Model-1'].parts['Part-2'].datums[2], orientationType=SYSTEM,
    region=Region(
    cells=mdb.models['Model-1'].parts['Part-2'].cells.getSequenceFromMask(
    mask=('[#1 ]', ), )), stackDirection=STACK_3)
mdb.models['Model-1'].parts['Part-3'].MaterialOrientation(
    additionalRotationField='', additionalRotationType=ROTATION_NONE, angle=0.0
    , axis=AXIS_3, fieldName='', localCsys=
    mdb.models['Model-1'].parts['Part-3'].datums[2], orientationType=SYSTEM,
    region=Region(
    cells=mdb.models['Model-1'].parts['Part-3'].cells.getSequenceFromMask(
    mask=('[#1 ]', ), )), stackDirection=STACK_3)
mdb.models['Model-1'].parts['Part-1'].Set(cells=
    mdb.models['Model-1'].parts['Part-1'].cells.getSequenceFromMask(('[#1 ]',
    ), ), name='Set-2')
mdb.models['Model-1'].parts['Part-1'].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=
    mdb.models['Model-1'].parts['Part-1'].sets['Set-2'], sectionName=
    'Section-1', thicknessAssignment=FROM_SECTION)
mdb.models['Model-1'].parts['Part-2'].Set(cells=
    mdb.models['Model-1'].parts['Part-2'].cells.getSequenceFromMask(('[#1 ]',
    ), ), name='Set-2')
```

```
mdb.models['Model-1'].parts['Part-2'].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=
    mdb.models['Model-1'].parts['Part-2'].sets['Set-2'], sectionName=
    'Section-2', thicknessAssignment=FROM_SECTION)
mdb.models['Model-1'].parts['Part-3'].Set(cells=
    mdb.models['Model-1'].parts['Part-3'].cells.getSequenceFromMask(('[#1 ]',
    ), ), name='Set-2')
mdb.models['Model-1'].parts['Part-3'].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=
    mdb.models['Model-1'].parts['Part-3'].sets['Set-2'], sectionName=
    'Section-3', thicknessAssignment=FROM_SECTION)
mdb.models['Model-1'].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models['Model-1'].rootAssembly.Instance(dependent=OFF, name='Part-1-1',
    part=mdb.models['Model-1'].parts['Part-1'])
mdb.models['Model-1'].rootAssembly.Instance(dependent=OFF, name='Part-2-1',
    part=mdb.models['Model-1'].parts['Part-2'])
mdb.models['Model-1'].rootAssembly.Instance(dependent=OFF, name='Part-3-1',
    part=mdb.models['Model-1'].parts['Part-3'])
mdb.models['Model-1'].rootAssembly.Coaxial(fixedAxis=
    mdb.models['Model-1'].rootAssembly.instances['Part-2-1'].faces[1], flip=OFF
    , movableAxis=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces[1])
mdb.models['Model-1'].rootAssembly.Coaxial(fixedAxis=
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].faces[1], flip=OFF
    , movableAxis=
    mdb.models['Model-1'].rootAssembly.instances['Part-2-1'].faces[1])
mdb.models['Model-1'].rootAssembly.Coaxial(fixedAxis=
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].faces[0], flip=OFF
    , movableAxis=
    mdb.models['Model-1'].rootAssembly.instances['Part-2-1'].faces[1])
mdb.models['Model-1'].rootAssembly.ParallelFace(fixedPlane=
    mdb.models['Model-1'].rootAssembly.instances['Part-2-1'].faces[2], flip=OFF
    , movablePlane=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces[2])
mdb.models['Model-1'].rootAssembly.ParallelFace(fixedPlane=
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].faces[2], flip=OFF
    , movablePlane=
    mdb.models['Model-1'].rootAssembly.instances['Part-2-1'].faces[2])
mdb.models['Model-1'].rootAssembly.DatumPlaneByThreePoints(point1=
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].vertices[0],
    point2=
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].InterestingPoint(
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].edges[0], MIDDLE),
    point3=
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].InterestingPoint(
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].edges[1], MIDDLE))
mdb.models['Model-1'].rootAssembly.PartitionCellByDatumPlane(cells=
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].cells.getSequenceFromMask(
    mask=('[#1 ]', ), )+\
    mdb.models['Model-1'].rootAssembly.instances['Part-2-1'].cells.getSequenceFromMask(
    mask=('[#1 ]', ), )+\
```

```
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].cells.getSequenceFromMask(
    mask=('[#1 ]', ), ), datumPlane=
    mdb.models['Model-1'].rootAssembly.datums[13])
mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coordSysType=
    CARTESIAN, name='Datum csys-2', origin=
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].InterestingPoint(
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].edges[10], CENTER)
    , point1=
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].InterestingPoint(
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].edges[8], CENTER),
    point2=
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].vertices[0])
mdb.models['Model-1'].StaticStep(name='Step-1', previous='Initial')
mdb.models['Model-1'].fieldOutputRequests['F-Output-1'].setValues(variables=(
    'U', ))
mdb.models['Model-1'].rootAssembly.ReferencePoint(point=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].InterestingPoint(
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].edges[8], CENTER))
mdb.models['Model-1'].rootAssembly.Set(name='m_Set-1', referencePoints=(
    mdb.models['Model-1'].rootAssembly.referencePoints[16], ))
mdb.models['Model-1'].rootAssembly.Surface(name='s_Surf-1', side1Faces=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(
    mask=('[#204 ]', ), )+\
    mdb.models['Model-1'].rootAssembly.instances['Part-2-1'].faces.getSequenceFromMask(
    mask=('[#204 ]', ), )+\
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].faces.getSequenceFromMask(
    mask=('[#204 ]', ), ))
mdb.models['Model-1'].Coupling(controlPoint=
    mdb.models['Model-1'].rootAssembly.sets['m_Set-1'], couplingType=KINEMATIC,
    influenceRadius=WHOLE_SURFACE, localCsys=
    mdb.models['Model-1'].rootAssembly.datums[15], name='Constraint-1',
    surface=mdb.models['Model-1'].rootAssembly.surfaces['s_Surf-1'], u1=ON, u2=
    ON, u3=ON, ur1=ON, ur2=ON, ur3=ON)
mdb.models['Model-1'].rootAssembly.Set(name='Set-2', referencePoints=(
    mdb.models['Model-1'].rootAssembly.referencePoints[16], ))
mdb.models['Model-1'].ConcentratedForce(cf3=100.0, createStepName='Step-1',
    distributionType=UNIFORM, field='', localCsys=
    mdb.models['Model-1'].rootAssembly.datums[15], name='Load-1', region=
    mdb.models['Model-1'].rootAssembly.sets['Set-2'])
mdb.models['Model-1'].rootAssembly.Set(faces=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(
    mask=('[#110 ]', ), )+\
    mdb.models['Model-1'].rootAssembly.instances['Part-2-1'].faces.getSequenceFromMask(
    mask=('[#110 ]', ), )+\
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].faces.getSequenceFromMask(
    mask=('[#110 ]', ), ), name='Set-3')
mdb.models['Model-1'].EncastreBC(createStepName='Initial', localCsys=
    mdb.models['Model-1'].rootAssembly.datums[15], name='BC-1', region=
    mdb.models['Model-1'].rootAssembly.sets['Set-3'])
mdb.models['Model-1'].rootAssembly.seedPartInstance(deviationFactor=0.1,
    minSizeFactor=0.1, regions=(
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'],
```

```
    mdb.models['Model-1'].rootAssembly.instances['Part-2-1'],
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1']), size=0.005)
mdb.models['Model-1'].rootAssembly.seedEdgeBySize(constraint=FINER,
    deviationFactor=0.1, edges=
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].edges.getSequenceFromMask(
    mask=('[#aa ]', ), )+\
    mdb.models['Model-1'].rootAssembly.instances['Part-2-1'].edges.getSequenceFromMask(
    mask=('[#aa ]', ), )+\
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].edges.getSequenceFromMask(
    mask=('[#aa ]', ), ), minSizeFactor=0.1, size=0.0002)
mdb.models['Model-1'].rootAssembly.seedEdgeByNumber(constraint=FINER, edges=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].edges.getSequenceFromMask(
    ('[#a900 ]', ), ), number=20)
mdb.models['Model-1'].rootAssembly.seedEdgeByNumber(constraint=FINER, edges=
    mdb.models['Model-1'].rootAssembly.instances['Part-2-1'].edges.getSequenceFromMask(
    ('[#a900 ]', ), ), number=20)
mdb.models['Model-1'].rootAssembly.seedEdgeByNumber(constraint=FINER, edges=
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].edges.getSequenceFromMask(
    ('[#a900 ]', ), ), number=20)
mdb.models['Model-1'].rootAssembly.seedEdgeByNumber(constraint=FINER, edges=
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'].edges.getSequenceFromMask(
    ('[#5600 ]', ), ), number=20)
mdb.models['Model-1'].rootAssembly.generateMesh(regions=(
    mdb.models['Model-1'].rootAssembly.instances['Part-3-1'],
    mdb.models['Model-1'].rootAssembly.instances['Part-2-1'],
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1']))
mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF,
    explicitPrecision=SINGLE, getMemoryFromAnalysis=True, historyPrint=OFF,
    memory=50, memoryUnits=PERCENTAGE, model='Model-1', modelPrint=OFF,
    multiprocessingMode=DEFAULT, name='Job-55', nodalOutputPrecision=SINGLE,
    numCpus=1, numGPUs=0, queue=None, scratch='', type=ANALYSIS,
    userSubroutine='', waitHours=0, waitMinutes=0)
mdb.jobs['Job-55'].submit(consistencyChecking=OFF)
```