Supplement

Included is a sample of all of the Lagrangian solid modeling code used within this study. The core of this code, which is written for a Fortran 90 compiler, is included along with several function files: a bar in tension, a bar in compression, a disk compressed into an elastic steel bar, and the pressure vessel simulation. The code is written so that the core file *spam.f* can be compiled along with the model-building function of choice, and the simulation will run accordingly. The files are:

- **spam.f**: this is the core of the code, and is entirely self contained except for the model function file, which defines the model and specifies the material and simulation properties.
- **tension_steel.f**: the model function for the bar in tension.
- **compress_steel.f**: the model function for the bar in compression. It is identical to the tension function except the top and bottom *FixXY* values are reversed so the direction is reversed.
- **ModelSmFixDisk.f**: The model function for a rigid 2D disk compressing into an elastic flat plat.
- **PressVessel.f**: The model function for the 2D pressure vessel, with both steel solid particles and liquid particles under pressure, to demonstrate Lagrangian Fluid Solid Interactions.

In addition, there are several MatLab files used to post-process the data:

- **Analyze_Data.m**: This file is first run, which reads all of the ASCII data files, and saves it as a single, clean binary MatLab data file *Data.mat*.
- **PlotTension.m**: The file processes the tension plot. It will prompt the model before and after the simulation (exaggerated deflection), as well as calculating the analytical tension stress and transverse deflections, and compares it to the numerical results.

- **PlotComp.m**: Effectively the same as the *PlotTension.m* script, but with small modifications for compression simulations rather than tensile.

- **Plot_Hertz.m**: The plotting function for the Hertz disk-flat simulation. Various exaggerated deflection plots are generated, as well as calculation of the Hertz analytical stresses to compare to the analytical results.

- **MakeFigs_PresVes.m**: Plots the 2D pressure vessel model, as well as the relative radial deflections.

To run the simulation, first the user compiles the *spam.f* function with the model function of interest. The machine-language program of the simulation is run, and the code will output the results as various ASCII text files of the simulation results. The user can then go into MatLab and run the *Analyze_Data.m* script, which reads all of the ASCII data files and outputs a single compressed binary MatLab file (*Data.mat*) of the simulation results. The user can then select the relevant MatLab script to load the MatLab data file and plot and analyze the results of the simulation.

The *Analyze_Data.m* script will consistently save the results as a binary *Data.mat* file, regardless of the study performed. The saved results are renamed, and the associated MatLab plotting scripts read these different file-names. The scripts are as follows:

- **Tension_steel.mat**

- **compress_steel.mat**

- **Hertz.mat**

- **PressVess.mat**

FORTRAN SOURCE CODE

spam.f

```
      program spam

c     ------------------------------------------------------------------
      implicit none

      integer ii,jj,kk,jj0,uu,vv,ctbar(3),ctZ,ctZmod
      integer ct,ct2,xx,yy,N,Nsphere,ts,fooint,StudyCt,CenterPt(2)
      integer Nts,LoopCT,LoopCT2,stop,start2,stop2,ctCyl,StopWeight
      integer xxx,yyy,uv(2),break,breakct,abreak,split,StartStop
      integer BSx,cubecyl,ctx(2)
      complex i
      double precision pi,dx3(3),third,Xloc,Yloc,Zloc,StartDefl,NewDefl
      double precision dx00, dxE, dTfactor,dTdxfactor,dTdxfactor2,TTxx
      double precision masX,rho_solid,foo3,Xr0,Yr0,Vydt,dt0x,dx0x
      double precision deflect, deflectWeight, Rred, Ered, bWeight,mass
      double precision h, c02, K_bulk, Vy0,foo,r,dwdx(3),t1,t2
      double precision g(3), foo2, foo3vv, dxJ, Arange, dxEj
      double precision dt,dt0,Dij(3,3),OMGij(3,3),Sij3(3,3)
      double precision dSdt(6),RijAB(3,3),Wij,rho_ij,Radius,RadiusFacto
      double precision Weight0,Weight,Pweight(3),Vtest(3),Tfract,Efrac
      double precision dx,dxd,dE,deltX,deltX2,dxi,a,angleFrac,angle
      double precision lambda,mu,Ey,poisson,TT,Time,Bulk_water
      double precision Csound,rho0min,dV,gamma,dVdx0
      double precision X1(3),X2(3),V1(3),V2(3)
      integer, allocatable, dimension(:) :: Mat, FixXY, IsSolid
      integer, allocatable, dimension(:) :: WeightXY
      integer, allocatable, dimension(:,:) :: Contact,ContactDir
```

```
      integer, allocatable, dimension(:,:) :: ContactDirCT
      integer, allocatable, dimension(:,:) :: ContactX,ContactDirX
      integer, allocatable, dimension(:,:) :: ContactDirCTX
      integer, allocatable, dimension(:,:) :: Links
      integer, allocatable, dimension(:) :: BreakContact, RadiusCrap
      double precision, allocatable, dimension(:,:) :: X,X0,X00
      double precision, allocatable, dimension(:,:) :: V,V0,dVdx
      double precision, allocatable, dimension(:,:) :: dVdxFct,dVdyFct
      double precision, allocatable, dimension(:,:) :: dVdzFct
      double precision, allocatable, dimension(:,:) :: V00
      double precision, allocatable, dimension(:,:) :: TcomboOut
      double precision, allocatable, dimension(:,:,:) :: DSDtN,DSDtN0
      double precision, allocatable, dimension(:,:,:) :: Tij,Eij,Eij0
      double precision, allocatable, dimension(:,:,:) :: TTij,Tij0
      double precision, allocatable, dimension(:,:,:) :: TijCombo
      double precision, allocatable, dimension(:) :: rho,rho0,P,P0
      double precision, allocatable, dimension(:) :: printVar,Tvm0
      double precision, allocatable, dimension(:) :: TimeFct
      double precision, allocatable, dimension(:) :: massFct
      double precision, allocatable, dimension(:,:) :: D0,Omg0
      double precision, allocatable, dimension(:,:) :: VonMiss
      double precision, allocatable, dimension(:,:) :: Xout,Yout,Zout
      double precision, allocatable, dimension(:,:) :: VoutX,VoutY,Vout
      double precision, allocatable, dimension(:,:) :: dVdy,Tvm
      double precision, allocatable, dimension(:,:) :: Pout,Rout
      double precision, allocatable, dimension(:,:) :: DPDT,DPDT0
      double precision, allocatable, dimension(:,:) :: T22o
      double precision, allocatable, dimension(:,:) :: T11,T12,T13
      double precision, allocatable, dimension(:,:) :: T21,T22,T23
      double precision, allocatable, dimension(:,:) :: T31,T32,T33
      double precision, allocatable, dimension(:,:) :: E22,E11
      double precision, allocatable, dimension(:) :: dx0
      double precision, allocatable, dimension(:) :: zOffset


c     ------------------------------------------------------------------
c     These are the universal constants
c     ------------------------------------------------------------------
      pi = 3.141592653589793 ! set the value of pi
      i = (0, 1) ! set the imaginary number i=sqrt(-1)
```

```
      g = (/ 0.0, 0.0, 0.0 /)
      third=1.0/3.0

c     ------------------------------------------------------------------
c     Declare array sizes
c     ------------------------------------------------------------------

      call ModelBlock(N)

      open (unit=1,file="SimSpec.dat",STATUS='OLD',ACTION='READ')
        read (1,*),Ey
        read (1,*),poisson
        read (1,*),rho_solid
        read (1,*),Vtest(1)
        read (1,*),Vtest(2)
        read (1,*),Vtest(3)
        read (1,*),gamma
        read (1,*),Nts
        read (1,*),LoopCT
        read (1,*),StudyCt
        read (1,*),StartDefl
        read (1,*),NewDefl
        read (1,*),dTfactor
        read (1,*),dTdxfactor
        read (1,*),stop
        read (1,*),Bulk_water
      close (1)

      Vy0 = Vtest(2)
      K_bulk = Ey/(3*(1-(2*poisson))) ! bulk modulus of iron
      mu=Ey/(2*(1+poisson))
      lambda=K_bulk-(2*mu/3)
      Ered=2/((1-(poisson**2))/Ey)

      Nts=Nts*StudyCt

      ALLOCATE (X(N,3))
      ALLOCATE (X0(N,3))
      ALLOCATE (X00(N,3))
      ALLOCATE (V(N,3))
```

```
ALLOCATE (V0(N,3))
ALLOCATE (V00(N,3))
ALLOCATE (dx0(N))
ALLOCATE (dVdx(N,3))
ALLOCATE (dVdxFct(N,Nts))
ALLOCATE (dVdyFct(N,Nts))
ALLOCATE (dVdzFct(N,Nts))
ALLOCATE (Mat(N))
ALLOCATE (FixXY(N))
ALLOCATE (IsSolid(N))
ALLOCATE (rho(N))
ALLOCATE (rho0(N))
ALLOCATE (massFct(N))
ALLOCATE (P(N))
ALLOCATE (P0(N))
ALLOCATE (Tij(N,3,3))
ALLOCATE (TTij(N,3,3))
ALLOCATE (Tij0(N,3,3))
ALLOCATE (TijCombo(N,3,3))
ALLOCATE (Eij(N,3,3))
ALLOCATE (Eij0(N,3,3))
ALLOCATE (DSDtN(N,3,3))
ALLOCATE (DSDtN0(N,3,3))
ALLOCATE (DPDT(N,3))
ALLOCATE (DPDT0(N,3))
ALLOCATE (D0(N,6))
ALLOCATE (Omg0(N,6))
ALLOCATE (Links(N,N+1))
ALLOCATE (Xout(N,Nts))
ALLOCATE (Yout(N,Nts))
ALLOCATE (Zout(N,Nts))
ALLOCATE (VoutX(N,Nts))
ALLOCATE (VoutY(N,Nts))
ALLOCATE (VoutZ(N,Nts))
ALLOCATE (dVdy(N,Nts))
ALLOCATE (Pout(N,Nts))
ALLOCATE (Rout(N,Nts))
ALLOCATE (TimeFct(Nts))
ALLOCATE (Tvm0(N))
ALLOCATE (Tvm(N,Nts))
```

```
      ALLOCATE (TcomboOut(N,Nts))
      ALLOCATE (T11(N,Nts))
      ALLOCATE (T12(N,Nts))
      ALLOCATE (T13(N,Nts))
      ALLOCATE (T21(N,Nts))
      ALLOCATE (T22(N,Nts))
      ALLOCATE (T23(N,Nts))
      ALLOCATE (T31(N,Nts))
      ALLOCATE (T32(N,Nts))
      ALLOCATE (T33(N,Nts))
      ALLOCATE (T22o(N,Nts))
      ALLOCATE (E22(N,Nts))
      ALLOCATE (E11(N,Nts))
      ALLOCATE (VonMiss(N,Nts))
      ALLOCATE (Contact(N,(N+1)))
      ALLOCATE (ContactDir(N,(N+1)))
      ALLOCATE (ContactDirCT(N,3))
      ALLOCATE (ContactX(N,(N+1)))
      ALLOCATE (ContactDirX(N,(N+1)))
      ALLOCATE (ContactDirCTX(N,3))
      ALLOCATE (BreakContact(N))

      ALLOCATE (printVar(N))

      ALLOCATE (RadiusCrap(N))

c     ------------------------------------------------------------------
c     Declare size and velocity of particles
c     ------------------------------------------------------------------

      open (unit=1,file="Fixed.dat",STATUS='OLD',ACTION='READ')
      do ii=1,N
        read (1,*),FixXY(ii)
      enddo
      close (1)

      open (unit=1,file="Contact.dat",STATUS='OLD',ACTION='READ')
      do ii=1,N
        read (1,*),Contact(ii,:)
      enddo
```

```
close (1)

open (unit=1,file="ContactDir.dat",STATUS='OLD',ACTION='READ')
do ii=1,N
  read (1,*),ContactDir(ii,:)
enddo
close (1)

open (unit=1,file="ContactDirCT.dat",STATUS='OLD',ACTION='READ')
do ii=1,N
  read (1,*),ContactDirCT(ii,:)
enddo
close (1)

open (unit=1,file="ContactX.dat",STATUS='OLD',ACTION='READ')
do ii=1,N
  read (1,*),ContactX(ii,:)
enddo
close (1)

open (unit=1,file="ContactDirX.dat",STATUS='OLD',ACTION='READ')
do ii=1,N
  read (1,*),ContactDirX(ii,:)
enddo
close (1)

open (unit=1,file="ContactDirCTX.dat",STATUS='OLD',ACTION='READ'
do ii=1,N
  read (1,*),ContactDirCTX(ii,:)
enddo
close (1)

open (unit=1,file="Mass.dat",STATUS='OLD',ACTION='READ')
do ii=1,N
  read (1,*),massFct(ii)
enddo
close (1)

open (unit=1,file="dX.dat",STATUS='OLD',ACTION='READ')
do ii=1,N
```

```
  read (1,*),dx0(ii)
enddo
close (1)

open (unit=1,file="X0.dat",STATUS='OLD',ACTION='READ')
do ii=1,N
  read (1,*),X(ii,:)
enddo
close (1)

open (unit=1,file="V0.dat",STATUS='OLD',ACTION='READ')
do ii=1,N
  read (1,*),V(ii,:)
enddo
close (1)

open (unit=1,file="dVdX.dat",STATUS='OLD',ACTION='READ')
do ii=1,N
  read (1,*),dVdx(ii,:)
enddo
close (1)

open (unit=1,file="rho0.dat",STATUS='OLD',ACTION='READ')
do jj=1,N
  read (1,*),rho0(jj)
enddo
close (1)

open (unit=1,file="rho_init.dat",STATUS='OLD',ACTION='READ')
do jj=1,N
  read (1,*),rho(jj)
enddo
close (1)

open (unit=1,file="IsSolid.dat",STATUS='OLD',ACTION='READ')
do jj=1,N
  read (1,*),IsSolid(jj)
enddo
close (1)
```

```
c ---------------------------------------------------------------
c ---------------------------------------------------------------

      call SpeedSound(Ey,poisson,rho_solid,Csound)
      dt0=dTfactor*0.5*(0.25*(2*(MINVAL(dx0)))/Csound)

      X00=X
      open (unit = 1, file = "Xoriginal.dat")
        do ii=1,N
          write (1,*),X00(ii,:)
        enddo
      close (1)
      Time=0

      do ii=1,N
        do jj=1,3
          do kk=1,3
            Tij(ii,jj,kk)=0
            TTij(ii,jj,kk)=0
            Eij(ii,jj,kk)=0
          enddo
          if (Vtest(jj)==0) then
            Tij0(:,jj,jj)=0
          else
            Tij0(:,jj,jj)=Vtest(jj)/(abs(Vtest(jj)))
          endif
        enddo
      enddo
      Tij0=Tij0*(1e-9)
      deflect=0

      call cpu_time ( t1 )

      do ts=1,Nts
        call cpu_time ( t2 )
        print *,ts,'/',Nts,', Elapsed CPU time = ', (t2 - t1),' seconds

        xxx=((ts-1)*StudyCt/Nts)
        if (xxx==0) then
```

```
  LoopCT2=3*LoopCT
else
  LoopCT2=LoopCT
endif
deflectWeight=MINVAL(dx0)*(StartDefl+(xxx*NewDefl/StudyCt))

do yyy=1,LoopCT2

dt=dTdxfactor*(MINVAL(dx0))/(MAXVAL(abs(V))+(1e-10))
if (abs(deflect)<deflectWeight) then
  dt0x=abs(dTdxfactor*(MINVAL(dx0))/Vtest(2))
else
  dt0x=dt0
endif
if (dt0x<dt) then
  dt=dt0x
endif
if (dt0<dt) then
  dt=dt0
endif
Time=Time+dt

if (ts<=stop) then
  if (abs(deflect)<deflectWeight) then
    deflect=deflect+(Vtest(2)*dt)
  endif
else
  deflect=deflect
endif


call LinkList(N,X,MAXVAL(dx0),Links)
call ContactStress(N,Tij,Contact,ContactDir,Tij0)

Contact=ContactX
ContactDir=ContactDirX
ContactDirCT=ContactDirCTX

do jj=1,N
  if (IsSolid(jj)==1) then
```

```
            ct=ContactX(jj,1)
            do jj0=1,Links(jj,1)
              ii=Links(jj,jj0+1)

              if (ii/=jj) then
                call kernel(X(ii,:),X(jj,:),2*dx0(jj),Wij,dwdx,r)
                do kk=1,3
                  dx3(kk)=abs(X(ii,kk)-X(jj,kk))
                enddo
                fooint=MAXLOC(dx3,1)
                dxE=dx0(jj)*(1+Eij(jj,fooint,fooint))
                if (dxE<(dx0(jj)*1.05)) then
                  dxE=(dx0(jj)*1.05)
                endif

                if (r<dxE) then
                  kk=1
                  do uu=2,(ContactX(jj,1)+1)
                    vv=ContactX(jj,uu)
                    if (ii==vv) then
                      kk=0
                    endif
                  enddo
                  if (kk==1) then
                    ct=ct+1
                    Contact(jj,ct+1)=ii
                    ContactDir(jj,ct+1)=fooint
                    ContactDirCT(jj,fooint)=ContactDirCT(jj,fooint)+1
                    Tij0(jj,fooint,fooint)=-1
                  endif
                endif
              endif
            enddo
            Contact(jj,1)=ct
            ContactDir(jj,1)=ct
          endif
        enddo

c _____________________________________________
```

```
c _____________________________________________

c_______Calculate the new velocity. __________

        X0=X
        V0=V
        do xx=1,3
          do ii=1,N
            dxE=dx0(ii)*(1+Eij(ii,xx,xx))
            foo = g(xx)
            mass=massFct(ii)
            if (IsSolid(ii)==1) then

            if (Contact(ii,1)>0) then
            do uu=2,((Contact(ii,1))+1)
              jj=Contact(ii,uu)
              if (ContactDir(ii,uu)==xx) then
                if (uu>(1+ContactX(ii,1))) then
                  call Compression(xx,ii,jj,N,X,Eij,Ey,dx0,mass,foo2,a
                else
                  call Tension(xx,ii,jj,N,X,Eij,Ey,dx0,mass,foo,foo2,T
                endif
                foo=foo+foo2
              else
                yy=ContactDir(ii,uu)
                call Shear(xx,yy,ii,jj,N,X,Eij,mu,dx0,mass,foo2,TT)
                foo=foo+foo2
              endif
              enddo
            endif
            endif

            if (IsSolid(ii)==1) then
             do jj=1,N
               if (IsSolid(jj)==0) then
                 X1=X(ii,:)
                 X2=X(jj,:)
                 call LJfctSolid(xx,X1,X2,dx0(ii),P(ii),a)
                 foo=foo+(a/mass)
```

```
      endif
    enddo
  else
    do jj=1,N
      if (IsSolid(jj)==1) then

        X1=X(ii,:)
        X2=X(jj,:)
        V1=V(ii,:)
        V2=V(jj,:)
        call LJfct(xx,X1,X2,V1,V2,dx0(ii),dt,a)

        foo=foo+a
      endif
    enddo
  endif


  ct=2
  do jj0=1,Links(ii,1)
    jj=Links(ii,jj0+1)
    call kernel(X(ii,:),X(jj,:),2*dx0(ii),Wij,dwdx,r)
    if (IsSolid(jj)==0) then
      foo2=(P(jj)-P(ii))*(dx0(ii)**2)/mass
      foo2=-foo2*((X(ii,xx)-X(jj,xx))/r)
      foo=foo2
    endif
  enddo


  dVdx0=dVdx(ii,xx)
  dVdx(ii,xx) = foo
  V(ii,xx)=V(ii,xx)+(dt*(dVdx0+dVdx(ii,xx))/2)
 enddo
enddo

X0=X
V00=V
do xx=1,3
 do ii=1,N
```

```
            if (IsSolid(ii)==1) then
              if ((Contact(ii,1))>0) then
                foo=0
                do uu=2,((Contact(ii,1))+1)
                  jj=Contact(ii,uu)
                  foo=foo+V00(jj,xx)
                enddo
                foo=foo/(Contact(ii,1))
                foo=(foo+(V00(ii,xx)))/2
                V(ii,xx)=foo
              endif
            endif
          enddo
        enddo


c_______Definition of FixXY_____________________

c       FixXY(ii)=0 --> Free Particle
c       FixXY(ii)=1 --> Fixed in space
c       FixXY(ii)=2 --> Fixed in X/1-direction
c       FixXY(ii)=3 --> Fixed in Y/2-direction
c       FixXY(ii)=4 --> Fixed in Z/3-direction
c       FixXY(ii)=5 --> Free in X/1-direction only
c       FixXY(ii)=6 --> Free in Y/2-direction only
c       FixXY(ii)=7 --> Free in Z/3-direction only
c       FixXY(ii)=8 --> Follows V_test
c       FixXY(ii)=9 --> Follows -V_test
c       FixXY(ii)=10 --> Follows V_test in the X/1-direction
c       FixXY(ii)=11 --> Follows -V_test in the X/1-direction
c       FixXY(ii)=12 --> Follows V_test in the Y/2-direction
c       FixXY(ii)=13 --> Follows -V_test in the Y/2-direction
c       FixXY(ii)=14 --> Follows V_test in the Z/3-direction
c       FixXY(ii)=15 --> Follows -V_test in the Z/3-direction

c_______End Definition of FixXY_________________


        foo2=0
c_______Calculate the new location. ___________
```

```
do uu=1,3
  do ii=1,N
    if (IsSolid(ii)==1) then
    dV=V(ii,uu)+V0(ii,uu)
    if (dV>(0.1*(dx0(ii))/dt)) then
      dV=0
      V(ii,uu)=0
    elseif (abs(dV)<(1e-6)) then
      dV=0
      V(ii,uu)=0
    endif

    if (FixXY(ii)==1) then
      X(ii,uu)=X00(ii,uu)
    elseif (FixXY(ii)==2) then
      if (uu==1) then
        X(ii,uu)=X00(ii,uu)
      else
        X(ii,uu)=X(ii,uu)+(dt*dV/2)
      endif
    elseif (FixXY(ii)==3) then
      if (uu==2) then
        X(ii,uu)=X00(ii,uu)
      else
        X(ii,uu)=X(ii,uu)+(dt*dV/2)
      endif
    elseif (FixXY(ii)==4) then
      if (uu==3) then
        X(ii,uu)=X00(ii,uu)
      else
        X(ii,uu)=X(ii,uu)+(dt*dV/2)
      endif
    elseif (FixXY(ii)==5) then
      if (uu==1) then
        X(ii,uu)=X(ii,uu)+(dt*dV/2)
      else
        X(ii,uu)=X00(ii,uu)
      endif
    elseif (FixXY(ii)==6) then
      if (uu==2) then
```

```
    X(ii,uu)=X(ii,uu)+(dt*dV/2)
  else
    X(ii,uu)=X00(ii,uu)
  endif
elseif (FixXY(ii)==7) then
  if (uu==3) then
    X(ii,uu)=X(ii,uu)+(dt*dV/2)
  else
    X(ii,uu)=X00(ii,uu)
  endif
elseif (FixXY(ii)==8) then
  if (uu==2) then
    X(ii,uu)=X00(ii,uu)+deflect
  else
    X(ii,uu)=X00(ii,uu)
  endif
elseif (FixXY(ii)==9) then
  if (uu==2) then
    X(ii,uu)=X00(ii,uu)-deflect
  else
    X(ii,uu)=X00(ii,uu)
  endif
elseif (FixXY(ii)==10) then
  if (uu==1) then
    X(ii,uu)=X00(ii,uu)+deflect
  else
    X(ii,uu)=X(ii,uu)+(dt*dV/2)
  endif
elseif (FixXY(ii)==11) then
  if (uu==1) then
    X(ii,uu)=X00(ii,uu)-deflect
  else
    X(ii,uu)=X(ii,uu)+(dt*dV/2)
  endif
elseif (FixXY(ii)==12) then
  if (uu==2) then
    X(ii,uu)=X00(ii,uu)+deflect
  else
    X(ii,uu)=X(ii,uu)+(dt*dV/2)
  endif
```

```
            elseif (FixXY(ii)==13) then
              if (uu==2) then
                X(ii,uu)=X00(ii,uu)-deflect
              else
                X(ii,uu)=X(ii,uu)+(dt*dV/2)
              endif
            elseif (FixXY(ii)==14) then
              if (uu==3) then
                X(ii,uu)=X00(ii,uu)+deflect
              else
                X(ii,uu)=X(ii,uu)+(dt*dV/2)
              endif
            elseif (FixXY(ii)==15) then
              if (uu==3) then
                X(ii,uu)=X00(ii,uu)-deflect
              else
                X(ii,uu)=X(ii,uu)+(dt*dV/2)
              endif
            else
              X(ii,uu)=X(ii,uu)+(dt*dV/2)
            endif
            else
              X(ii,uu)=X(ii,uu)+(dt*dV/2)
            endif
          enddo
        enddo


c_______Get new Stress. _____________________________
        call GetDensity(X,V,N,dx0,dt,massFct,rho,IsSolid,rho)

c Calculate the Tensial Stress
        Tij=Tij-TTij
        do xx=1,3
          do ii=1,N
            mass=massFct(ii)
            if (IsSolid(ii)==1) then

            if (Contact(ii,1)>0) then
              foo=0
```

```
                ct2=0
                do uu=2,((Contact(ii,1))+1)
                  TTxx=0
                  do vv=1,3
                    TTxx=TTxx+Tij0(ii,vv,vv)
                  enddo

                  jj=Contact(ii,uu)
                  if (ContactDir(ii,uu)==xx) then
                   if (uu>(1+ContactX(ii,1))) then
                     call Compression(xx,ii,jj,N,X,Eij,Ey,dx0,mass,a,TT)
                   else
                     call Tension(xx,ii,jj,N,X,Eij,Ey,dx0,mass,TTxx,a,TT
                   endif
                   foo=foo+TT
                   ct2=ct2+1
                  endif
                enddo

                if (FixXY(ii)>0) then
                  foo=foo/2
                endif
                if (ct2==0) then
                  TTij(ii,xx,xx)=0*TTij(ii,xx,xx)
                else
                  TTij(ii,xx,xx)=foo/ct2
                endif
              endif

              else
                TTij(ii,xx,xx)=(Bulk_water/3)*((rho(ii)/rho0(ii))-1)
              endif
            enddo
          enddo



c Calculate the Shear Stress
          do vv=1,3
            call GetUV(vv,uv)
```

```
          do ii=1,N
            if (IsSolid(ii)==1) then
              foo=0
              do uu=2,((Contact(ii,1))+1)
                jj=Contact(ii,uu)
                if (ContactDir(ii,uu)==uv(2)) then
                  call Shear(uv(1),uv(2),ii,jj,N,X,Eij,mu,dx0,mass,a,T
                  foo=foo+TT
                endif
              enddo

              if (FixXY(ii)==0) then
                if (ContactDirCT(ii,uv(2))==0) then
                  foo=0
                else
                  foo=foo/ContactDirCT(ii,uv(2))
                endif
              else
                foo=foo/2
              endif

              TTij(ii,uv(2),uv(1))=foo
              TTij(ii,uv(1),uv(2))=foo
            endif
          enddo
        enddo

        Tij=Tij+TTij


c _______________Get new strain_______________

        do kk=1,N
          do ii=1,3
            Eij(kk,ii,ii)=(1/Ey)*(Tij(kk,ii,ii))
            call GetUV(ii,uv)
            foo=(poisson/Ey)*(Tij(kk,uv(1),uv(1)))
            foo=foo+(poisson/Ey)*(Tij(kk,uv(2),uv(2)))
            Eij(kk,ii,ii)=Eij(kk,ii,ii)-foo
          enddo
```

```
          do ii=1,3
            do jj=1,3
              if (jj/=ii) then
                Eij(kk,ii,jj)=Tij(kk,ii,jj)/(2*mu)
              endif
            enddo
          enddo
        enddo




c _________________Save Data_____________________

        do ii=1,N
          foo=(Tij(ii,1,1)-Tij(ii,2,2))**2
          foo=foo+(Tij(ii,2,2)-Tij(ii,3,3))**2
          foo=foo+(Tij(ii,1,1)-Tij(ii,3,3))**2
          foo2=(Tij(ii,1,2)**2)+(Tij(ii,2,3)**2)+(Tij(ii,3,1)**2)
          foo=sqrt((foo+(6*foo2))/2)
          Tvm0(ii)=foo
          P(ii)=(Tij(ii,1,1)+Tij(ii,2,2)+Tij(ii,3,3))/3
        enddo

        TimeFct(ts)=Time
        Xout(:,ts)=X(:,1)
        Yout(:,ts)=X(:,2)
        Zout(:,ts)=X(:,3)
        VoutX(:,ts)=V(:,1)
        VoutY(:,ts)=V(:,2)
        VoutZ(:,ts)=V(:,3)
        Rout(:,ts)=rho
        Pout(:,ts)=P
        T11(:,ts)=Tij(:,1,1)
        T12(:,ts)=Tij(:,1,2)
        T13(:,ts)=Tij(:,1,3)
        T21(:,ts)=Tij(:,2,1)
        T22(:,ts)=Tij(:,2,2)
        T23(:,ts)=Tij(:,2,3)
        T31(:,ts)=Tij(:,3,1)
```

```
        T32(:,ts)=Tij(:,3,2)
        T33(:,ts)=Tij(:,3,3)
        Tvm(:,ts)=Tvm0
        E22(:,ts)=Eij(:,2,2)
        E11(:,ts)=Eij(:,1,1)
        dVdxFct(:,ts)=dVdx(:,1)
        dVdyFct(:,ts)=dVdx(:,2)
        dVdzFct(:,ts)=dVdx(:,3)

        enddo
      enddo

      call cpu_time ( t2 )
      print *,'Elapsed CPU time = ', (t2 - t1),' seconds'


c -------------------------------------------------------
c  Print data to dat-files
c -------------------------------------------------------

      open (unit = 1, file = "VarDat.dat")
        write (1,*),Ey
        write (1,*),mu
        write (1,*),StudyCt
      close (1)

      open (unit = 1, file = "X.dat")
      do jj=1,Nts
        printVar=Xout(:,jj)
        write (1,*),printVar
      enddo
      close(1)

      open (unit = 1, file = "Y.dat")
      do jj=1,Nts
        printVar=Yout(:,jj)
        write (1,*),printVar
      enddo
      close(1)
```

```
open (unit = 1, file = "Z.dat")
do jj=1,Nts
  printVar=Zout(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "P.dat")
do jj=1,Nts
  printVar=Pout(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "rho.dat")
do jj=1,Nts
  printVar=Rout(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "T11.dat")
do jj=1,Nts
  printVar=T11(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "T12.dat")
do jj=1,Nts
  printVar=T12(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "T13.dat")
do jj=1,Nts
  printVar=T13(:,jj)
  write (1,*),printVar
enddo
```

```
close(1)

open (unit = 1, file = "T21.dat")
do jj=1,Nts
  printVar=T21(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "T22.dat")
do jj=1,Nts
  printVar=T22(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "T23.dat")
do jj=1,Nts
  printVar=T23(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "T31.dat")
do jj=1,Nts
  printVar=T31(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "T32.dat")
do jj=1,Nts
  printVar=T32(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "T33.dat")
do jj=1,Nts
  printVar=T33(:,jj)
```

```
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "E22.dat")
do jj=1,Nts
  printVar=E22(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "E11.dat")
do jj=1,Nts
  printVar=E11(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "VonMises.dat")
do jj=1,Nts
  printVar=Tvm(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "Vx.dat")
do jj=1,Nts
  printVar=VoutX(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "Vy.dat")
do jj=1,Nts
  printVar=VoutY(:,jj)
  write (1,*),printVar
enddo
close(1)

open (unit = 1, file = "Vz.dat")
```

```
      do jj=1,Nts
        printVar=VoutZ(:,jj)
        write (1,*),printVar
      enddo
      close(1)

      open (unit = 1, file = "Time.dat")
      do jj=1,Nts
        printVar=TimeFct(jj)
        write (1,*),printVar
      enddo
      close(1)

c -----------------------------

      open (unit = 1, file = "Fixed.dat")
      do jj=1,N
        write (1,*),FixXY(jj)
      enddo
      close (1)

      open (unit = 1, file = "Contact.dat")
      do jj=1,N
        write (1,*),Contact(jj,:)
      enddo
      close (1)

      open (unit = 1, file = "ContactDir.dat")
      do jj=1,N
        write (1,*),ContactDir(jj,:)
      enddo
      close (1)

      open (unit = 1, file = "ContactDirCT.dat")
      do jj=1,N
        write (1,*),ContactDirCT(jj,:)
      enddo
      close (1)

      open (unit = 1, file = "ContactX.dat")
```

```
do jj=1,N
  write (1,*),ContactX(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactDirX.dat")
do jj=1,N
  write (1,*),ContactDirX(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactDirCTX.dat")
do jj=1,N
  write (1,*),ContactDirCTX(jj,:)
enddo
close (1)

open (unit = 1, file = "Links.dat")
do jj=1,N
  write (1,*),Links(jj,:)
enddo
close (1)

open (unit = 1, file = "Mass.dat")
do jj=1,N
  write (1,*),massFct(jj)
enddo
close (1)

open (unit = 1, file = "dX.dat")
do jj=1,N
  write (1,*),dx0(jj)
enddo
close (1)


open (unit = 1, file = "X0.dat")
do jj=1,N
  write (1,*),X(jj,:)
enddo
```

```
      close (1)

      open (unit = 1, file = "V0.dat")
      do jj=1,N
        write (1,*),V(jj,:)
      enddo
      close (1)

      open (unit = 1, file = "dVdX.dat")
      do jj=1,N
        write (1,*),dVdxFct(jj,:)
      enddo
      close (1)

      open (unit = 1, file = "dVdY.dat")
      do jj=1,N
        write (1,*),dVdyFct(jj,:)
      enddo
      close (1)

      open (unit = 1, file = "dVdZ.dat")
      do jj=1,N
        write (1,*),dVdzFct(jj,:)
      enddo
      close (1)


      end program



c _____________________________________________________________________

      subroutine GetUV(ii,uv)
c     ---------------------------------------------------------------
      implicit none

      integer, intent(in) :: ii
      integer, intent(out) :: uv(2)
      integer jj,ct
```

```
      ct=0
      do jj=1,3
        if (jj/=ii) then
          ct=ct+1
          uv(ct)=jj
        endif
      enddo

      END subroutine GetUV


c _____________________________________________________________________

      subroutine FindL(N,h,mu,X,V,P,Tij,DSDtN)
c     ----------------------------------------------------------------
      implicit none

      integer, intent(in) :: N
      double precision, intent(in) :: Tij(N,3,3)
      double precision, intent(in) :: X(N,3),V(N,3),h
      double precision, intent(in) :: mu
      double precision, intent(out) :: DSDtN(N,3,3),P(N)

      integer ii,jj, kk, uu, vv
      double precision L(3,3),D(3,3),Omg(3,3),r,W,Wtotal
      double precision dV,dx,D0(3,3),Omg0(3,3),DSDt(3,3)
      double precision foo,foo1,foo2,foo3,foo4,Sij(3,3)
      double precision dwdx(3)

      do jj=1,N
        Wtotal = 0
        do vv=1,3
          do uu=1,3
            L(uu,vv)=0
          enddo
        enddo
        do ii=1,N
          call kernel(X(jj,:),X(ii,:),h,W,dwdx,r)
          do vv = 1,3
```

```
    do uu = 1,3
      dV=(V(ii,uu)-V(jj,uu))
      dx=(X(ii,vv)-X(jj,vv))
      if (dx==0) then
        L(uu,vv)=L(uu,vv)
      else
        L(uu,vv)=L(uu,vv)+(dwdx(vv)*dV/dx)
        Wtotal=Wtotal+W
      endif
    enddo
  enddo
enddo
L=L*0/Wtotal

do vv=1,3
  do uu=1,3
    D(uu,vv)=(L(uu,vv)+L(vv,uu))/2
    Omg(uu,vv)=(L(uu,vv)-L(vv,uu))/2
  enddo
enddo


do vv=1,3
  do uu=1,3
    Sij(uu,vv)=Tij(jj,uu,vv)
    if (uu==vv) then
      Sij(uu,vv)=Sij(uu,vv)-P(jj)
    endif
  enddo
enddo

do vv=1,3
  do uu=1,3
    foo1=0
    foo2=0
    do kk=1,3
      foo1=foo1+(Sij(uu,kk)*Omg(vv,kk))
      foo2=foo2+(Omg(uu,kk)*Sij(kk,vv))
    enddo
    foo3=2*mu*D(uu,vv)
```

```
            if (uu==vv) then
              foo4=(-2*mu/3)*D(uu,vv)
            else
              foo4=0
            endif
            DSDtN(jj,uu,vv)=foo1+foo2+foo3+foo4
          enddo
        enddo



      enddo

      END subroutine FindL

c _____________________________________________________________________

      subroutine kernel(x1,x2,h,w,dwdx,r)

c-----------------------------------------------------------------------
c   Subroutine to calculate the smoothing kernel wij and its
c   derivatives dwdxij.

c     r    : Distance between particles i and j                    [in
c     dx   : x-, y- and z-distance between i and j                 [in
c     h : Smoothing length                                       [in]
c     w    : Kernel for all interaction pairs                      [out
c     dwdx : Derivative of kernel with respect to x, y and z       [out

      implicit none

      double precision, intent(in) :: x1(3),x2(3),h
      double precision, intent(out) :: w,dwdx(3),r
      integer i, j, d, skf
      double precision q, dw, factor,dx(3),pi,u

      do i=1,3
        dx(i)=abs(X1(i)-X2(i))
      enddo
      r=(dx(1)**2)+(dx(2)**2)+(dx(3)**2)
```

```
r=sqrt(r)

pi = 3.14159265358979
u=10/(7*pi)
q = r/h
w = 0.e0
do d=1,3
  dwdx(d) = 0.e0
enddo

if (r==0) then
  w=0
  dwdx(1)=0
  dwdx(2)=0
  dwdx(3)=0
else
  if (q<1) then
    w=1+(-1.5*(q**2))+(0.75*(q**3))
    do i=1,3
      dwdx(i)=((9*r/(4*h))-3)*(dx(i)/(h**2))
    enddo
  else
    if (q<2) then
      w=((2-q)**3)/4
      do i=1,3
        dwdx(i)=(-3/(4*h*r))*((2-q)**2)*dx(i)
      enddo
    else
      w=0
      dwdx(1)=0
      dwdx(2)=0
      dwdx(3)=0
    endif
  endif
endif
w=w*u
do i=1,3
  dwdx(i)=u*dwdx(i)
enddo
```

```
      END subroutine kernel

c _____________________________________________________________________

      subroutine GetDPDT(N,X,V,dx,K,massFct,rho,DPDT)
c     ---------------------------------------------------------------
      implicit none

      integer, intent(in) :: N
      double precision, intent(in) :: X(N,3),V(N,3)
      double precision, intent(in) :: dx(N),massFct(N)
      double precision, intent(in) :: K,rho(N)
      double precision, intent(out) :: DPDT(N,3)

      integer ii,jj, kk
      double precision W0, dwdx(3),foo,dV,r
      double precision h,mass

      do jj=1,N
        h=2*dx(jj)
        mass=massFct(jj)
        do ii=1,N
          call kernel(X(ii,:),X(jj,:),h,W0,dwdx,r)
          foo=0
          do kk=1,3
            dV=V(ii,kk)-V(jj,kk)
            foo=-K*mass*dV*dwdx(kk)/rho(ii)
            DPDT(jj,kk)=DPDT(jj,kk)+foo
          enddo
        enddo
      enddo

      END subroutine GetDPDT

c _____________________________________________________________________

      subroutine GetDensity0(X,N,dx,mass,rho0,IsSolid,rho)
c     ---------------------------------------------------------------
      implicit none
```

```
      integer, intent(in) :: N, IsSolid(N)
      double precision, intent(in) :: X(N,3),rho0(N)
      double precision, intent(in) :: dx(N),mass(N)
      double precision, intent(out) :: rho(N)

      integer ii,jj, kk
      double precision W0, r0, fooN, fooD, h, r, dwdx(3)
      double precision rhoIn(N)

      rhoIn=rho0
      do jj=1,N
        fooN=0
c        fooD=0
        h = 3*dx(jj)
        do ii=1,N
          if (IsSolid(ii)==IsSolid(jj)) then
            call kernel(X(ii,:),X(jj,:),h,W0,dwdx,r)
            fooN=fooN+(mass(ii)*W0)
          endif
        enddo
        rho(jj)=((3.14159*4/3)*fooN/(h**3))
      enddo

      END subroutine GetDensity0


c ____________________________________________________________

      subroutine GetDensity(X,V,N,dx,dt,massFct,rho0,IsSolid,rho)
c     ------------------------------------------------------------
      implicit none

      integer, intent(in) :: N, IsSolid(N)
      double precision, intent(in) :: X(N,3),V(N,3),rho0(N)
      double precision, intent(in) :: dt,dx(N),massFct(N)
      double precision, intent(out) :: rho(N)

      integer ii,jj, kk
      double precision W,r,dwdx(3),foo1,foo2
      double precision h,mass,rhoIn(N)
```

```
      rhoIn=rho0
      do jj=1,N
        h=dx(jj)
        mass=massFct(jj)
        foo1=0
        do ii=1,N
          if (IsSolid(ii)==IsSolid(jj)) then
            call kernel(X(ii,:),X(jj,:),h,W,dwdx,r)
            do kk=1,3
              foo1=foo1-((mass/rhoIn(ii))*V(ii,kk)*dwdx(kk))
            enddo
          else
            foo1=0
          endif
        enddo
        foo2=dt*foo1*rhoIn(jj)
        rho(jj)=rhoIn(jj)+foo2
      enddo


      END subroutine GetDensity

c _____________________________________________________________

      subroutine SpeedSound(YM,poisson,rho0min,Csound)

      implicit none

      double precision, intent(in) :: YM,poisson,rho0min
      double precision, intent(out) :: Csound
      double precision CsoundT,CsoundS,G

      CsoundT=(YM*(1-poisson))/rho0min
      CsoundT=CsoundT/((1+poisson)*(1-(2*poisson)))
      CsoundT=sqrt(CsoundT)

      G=(3*(1-poisson)/(1+poisson))-1
      G=G*YM/(4*(1-(2*poisson)))
      CsoundS=sqrt(G/rho0min)
```

```
      if (CsoundT>CsoundS) then
        Csound=CsoundT
      else
        Csound=CsoundS
      endif

      end subroutine SpeedSound

c _____________________________________________________________________

      subroutine LJfct(p,X1,X2,V1,V2,r0,dt,a)

      implicit none

      integer, intent(in) :: p
      double precision, intent(in) :: X1(3),X2(3)
      double precision, intent(in) :: V1(3),V2(3)
      double precision, intent(in) :: r0,dt
      double precision, intent(out) :: a

      integer ii
      double precision foo,r,r00,coeff,dir,foo2

      r=0
      do ii=1,3
        r=r+((X1(ii)-X2(ii))**2)
      enddo
      r=sqrt(r)
      if ((abs(X1(p)-X2(p)))==0) then
        dir=0
      else
        dir=(X1(p)-X2(p))/abs(X1(p)-X2(p))
      endif

      if (r>r0) then
        foo=0
      else
        coeff=(((r0/r)**12)-((r0/r)**4))
        foo=(V1(p)-V2(p))*coeff*dir/dt
```

```
      endif
      a=foo

      end subroutine LJfct

c _____________________________________________________________________

c _____________________________________________________________________

      subroutine LJfctSolid(p,X1,X2,r0,P2,f)

      implicit none

      integer, intent(in) :: p
      double precision, intent(in) :: X1(3),X2(3)
      double precision, intent(in) :: P2,r0
      double precision, intent(out) :: f

      integer ii
      double precision foo,r,del,trig

      r=0
      do ii=1,3
        r=r+((X1(ii)-X2(ii))**2)
      enddo
      del=(X1(p)-X2(p))
      r=sqrt(r)
      trig=del/r


      if (r>r0) then
        foo=0
      else
        foo=trig*P2*r0*r0
      endif
      f=foo

      end subroutine LJfctSolid
```

```
c _____________________________________________________________________

      subroutine Tension(xx,ii,jj,N,X,E,Ey,dx0,mass,TTxx,a,TT)

      implicit none

      integer, intent(in) :: xx,ii,jj,N
      double precision, intent(in) :: X(N,3),E(N,3,3)
      double precision, intent(in) :: Ey,dx0(N),mass,TTxx
      double precision, intent(out) :: a,TT


      integer ct,uv(2)
      double precision foo,dx,strain,Ai,A0,dir
      double precision dxE,strainE,a2,TT1,TT2,TTxxR

      A0=(dx0(ii))**2
      dx=X(jj,xx)-X(ii,xx)
      dxE=(1+((E(ii,xx,xx)+E(jj,xx,xx))/2))*dx0(ii)

      if (abs(dx)>0) then
        dir=dx/abs(dx)
      else
        dir=0
      endif

      strain=(abs(dx)-dx0(ii))/dx0(ii)
      strainE=(abs(dx)-dxE)/dxE

      a=0
      if (ii==jj) then
        a=0
        TT1=0
      else

        call GetUV(xx,uv)
        Ai=A0*(1+E(ii,uv(1),uv(1)))
        Ai=Ai*(1+E(ii,uv(2),uv(2)))
        a=dir*Ai*Ey*strainE/mass
```

```
      if (strain>0) then
        TT1=Ey*strain
      else
        TT1=0
      endif
    endif

    call Compression(xx,ii,jj,N,X,E,Ey,dx0,mass,a2,TT2)

    TTxxR=TTxx
    if (TTxxR>0) then
      TT=TT1
    elseif (TTxxR<0) then
      TT=TT2
    else
      if (strainE<0) then
        TT=TT2
      elseif (strainE>0) then
        TT=TT1
      else
        TT=(TT1+TT2)/2
      endif
    endif


    end subroutine Tension

c ___________________________________________________________________

    subroutine Compression(xx,ii,jj,N,X,E,Ey,dx0,mass,a,TT)

    implicit none

    integer, intent(in) :: xx,ii,jj,N
    double precision, intent(in) :: X(N,3),E(N,3,3)
    double precision, intent(in) :: Ey,dx0(N),mass
    double precision, intent(out) :: a,TT

    integer ct,uv(2)
    double precision foo,dx,strain,Ai,A0,dir
```

```
double precision dxE,strainE

A0=(dx0(ii))**2
dx=X(jj,xx)-X(ii,xx)
dxE=(1+((E(ii,xx,xx)+E(jj,xx,xx))/2))*dx0(ii)

if (abs(dx)>0) then
  dir=dx/abs(dx)
else
  dir=0
endif

strain=(abs(dx)-dx0(ii))/dx0(ii)
strainE=(abs(dx)-dxE)/dxE

a=0
if (ii==jj) then
  a=0
  TT=0
else

  call GetUV(xx,uv)
  Ai=A0*(1+E(ii,uv(1),uv(1)))
  Ai=Ai*(1+E(ii,uv(2),uv(2)))
  a=dir*Ai*Ey*strainE/mass

  if (strain<0) then
    TT=Ey*strain
  else
    TT=0
  endif
endif

if (TT>0) then
  TT=0
  a=0
endif

end subroutine Compression
```

```
c _____________________________________________________________________

      subroutine Shear(xx,yy,ii,jj,N,X,E,G,dx0,mass,a,TT)

      implicit none

      integer, intent(in) :: xx,yy,ii,jj,N
      double precision, intent(in) :: X(N,3),E(N,3,3)
      double precision, intent(in) :: G,dx0(N),mass
      double precision, intent(out) :: a,TT

      integer ct,uv(2)
      double precision delta,gamma,L
      double precision foo,Ai,A0,dir

      L=dx0(ii)
      A0=(dx0(ii))**2
      delta=(X(jj,xx)-X(ii,xx))
      if (abs(delta)>0) then
        dir=delta/abs(delta)
      else
        dir=0
      endif
      delta=abs(delta)
      gamma=delta/L


      if (ii==jj) then
        a=0
        TT=0
      else
        if (gamma>0) then
          TT=G*gamma
        else
          TT=0
        endif
```

```
  if (gamma>0) then
    call GetUV(yy,uv)
    Ai=A0*(1+E(ii,uv(1),uv(1)))
    Ai=Ai*(1+E(ii,uv(2),uv(2)))
    a=dir*Ai*G*gamma/mass
  else
    a=0
  endif

endif


end subroutine Shear


c _____________________________________________________________________

      subroutine ContactStress(N,TijIn,Contact,ContactDir,TijOut)

      implicit none

      integer, intent(in) :: N,Contact(N,N+1),ContactDir(N,N+1)
      double precision, intent(in) :: TijIn(N,3,3)
      double precision, intent(out) :: TijOut(N,3,3)

      integer ii,jj,xx,uu,cycle,ContactDirCTx
      double precision Tij0(N,3,3),Tij(N,3,3)
      double precision foo,Total(3),Avg(3)

      Tij0=TijIn

      do cycle=1,1000
      do ii=1,N
        do xx=1,3
          Avg(xx)=0
          ContactDirCTx=0
          do jj=2,(Contact(ii,1))
            if (ContactDir(ii,jj)==xx) then
              uu=Contact(ii,jj)
```

```
          Avg(xx)=Avg(xx)+Tij0(uu,xx,xx)
          ContactDirCTx=ContactDirCTx+1
        endif
      enddo
      if (ContactDirCTx==0) then
        Tij(ii,xx,xx)=Tij0(ii,xx,xx)
      else
        foo=Avg(xx)/ContactDirCTx
        Tij(ii,xx,xx)=(Tij0(ii,xx,xx)+foo)/2
      endif
    enddo
  enddo
  Tij0=Tij
  enddo
  TijOut=Tij


  end subroutine ContactStress


c _____________________________________________________________________

  subroutine LinkList(N,X,h,Links)

  implicit none

  integer, intent(in) :: N
  double precision, intent(in) :: X(N,3),h
  integer, intent(out) :: Links(N,(N+1))

  integer ii,jj,xx,uu,ct,fooInt,LinkCT(3),LLloc(N,3)
  double precision foo,DistCT(3),Wij,dwdx(3),r

  do ii=1,3
    DistCT(ii)=MAXVAL(X(:,ii))-MINVAL(X(:,ii))
    LinkCT(ii)=CEILING(DistCT(ii)/h)
  enddo
  Links(:,:)=0

  do ii=1,N
```

```
  do jj=1,3
    foo=(X(ii,jj)-MINVAL(X(:,jj)))/DistCT(jj)
    foo=ceiling(foo*LinkCT(jj))
    if (foo==0) then
      foo=foo+1
    endif
    LLloc(ii,jj)=foo
  enddo
enddo

do ii=1,N
  ct=1
  do jj=1,N
    xx=1
    do uu=1,3
      fooInt=abs(LLloc(ii,uu)-LLloc(jj,uu))
      if (fooInt>1) then
        xx=0
      endif
    enddo
    if (xx==1) then
      call kernel(X(ii,:),X(jj,:),h,Wij,dwdx,r)
      if (Wij>0) then
        ct=ct+1
        Links(ii,ct)=jj
      endif
    endif
  enddo
  Links(ii,1)=ct-1
enddo


end subroutine LinkList


c _____________________________________________________________________
```

tension_steel.f

```
subroutine ModelBlock(Nout)

implicit none

integer, intent(out) :: Nout
integer N,ctbar(3),Nsphere,ct,ii,jj,kk,uu
integer fooint,split
double precision RadiusFactor,Radius,masX,dx00
double precision angleFrac,angle,Yr0,dx3(3),third
double precision Zloc,Yloc,Xloc,r,pi,rho_solid
integer, allocatable, dimension(:) :: FixXY,IsSolid,BreakContact
integer, allocatable, dimension(:,:) :: Contact,ContactX
integer, allocatable, dimension(:,:) :: ContactDir,ContactDirX
integer, allocatable, dimension(:,:) :: ContactDirCT,ContactDirC
double precision, allocatable, dimension(:,:) :: X,V,dVdx
double precision, allocatable, dimension(:) :: massFct,dx0
double precision, allocatable, dimension(:) :: rho0,rho

open (unit = 1, file = "SimSpec.dat")
  write (1,*),207e9        ! Young's Modulus of Steel
  write (1,*),0.3          ! Poisson's Ratio of Steel
  write (1,*),7800         ! Density of Steel
  write (1,*),0            ! Forced Density in X direction (Vtest
  write (1,*),1e3         ! Forced Density in Y direction (Vtest
  write (1,*),0            ! Forced Density in Z direction (Vtest
  write (1,*),0.3          ! Gamma (blend stress)
  write (1,*),250          ! Number of Recorded Time Steps (Nts)
  write (1,*),6            ! Time Steps between recorded Time Ste
  write (1,*),1            ! Time Steps between recorded Time Ste
  write (1,*),0.18         ! (StartDefl)
  write (1,*),0.05         ! (NewDefl)
  write (1,*),0.25         ! (dTfactor)
  write (1,*),0.01         ! (dTdxfactor)
  write (1,*),250          ! (stop)
  write (1,*),2.15e9       ! (Bulk_water)
close (1)
```

```
      ctbar(1)=3     ! 3
      ctbar(2)=11    ! 11
      ctbar(3)=3     ! 3
      masX=0.1
      rho_solid=7800


      third=1.0/3.0
      dx00=(masX/rho_solid)**third
      N=(ctbar(1)*ctbar(2)*ctbar(3))


c _____________________________________________________
c                  ALLOCATION
c _____________________________________________________


      ALLOCATE (FixXY(N))
      ALLOCATE (IsSolid(N))
      ALLOCATE (BreakContact(N))
      ALLOCATE (X(N,3))
      ALLOCATE (V(N,3))
      ALLOCATE (dVdx(N,3))
      ALLOCATE (rho(N))
      ALLOCATE (rho0(N))
      ALLOCATE (massFct(N))
      ALLOCATE (dx0(N))
      ALLOCATE (Contact(N,(N+1)))
      ALLOCATE (ContactDir(N,(N+1)))
      ALLOCATE (ContactDirCT(N,3))
      ALLOCATE (ContactX(N,(N+1)))
      ALLOCATE (ContactDirX(N,(N+1)))
      ALLOCATE (ContactDirCTX(N,3))

c _____________________________________________________
```

```
c Set the parameters for the steel bar
      ct = 0
      Zloc=-dx00
      do uu=1,ctbar(3)
        Zloc=Zloc+dx00
        Yloc=-dx00
        do jj=1,ctbar(2)
          Yloc = Yloc + dx00
          Xloc = 0
          do ii=1,ctbar(1)
            ct = ct + 1
            if (jj==ctbar(2)) then
              FixXY(ct)=12
            elseif (jj==1) then
              FixXY(ct)=13
            else
              FixXY(ct)=0
            endif
            Xloc = Xloc + dx00
            X(ct,1) = Xloc
            X(ct,2) = Yloc
            X(ct,3) = Zloc
            V(ct,1) = 0
            V(ct,2) = 0
            V(ct,3) = 0
            dVdX(ct,1) = 0
            dVdX(ct,2) = 0
            dVdX(ct,3) = 0
            rho(ct) = rho_solid
            rho0(ct) = rho_solid
            dx0(ct) = dx00
            IsSolid(ct) = 1
            massFct(ct) = masX
            if (jj==(ctbar(2)/2)) then
              BreakContact(ct)=1
            elseif (jj==(1+(ctbar(2)/2))) then
              BreakContact(ct)=2
```

```
            else
              BreakContact(ct)=0
            endif
          enddo
        enddo
      enddo


c Set Contact links
      do jj=1,N
        if (IsSolid(jj)==1) then
          ct=0
          do ii=1,N
            if (IsSolid(ii)==1) then
              if (ii/=jj) then
                r=0
                do kk=1,3
                  dx3(kk)=abs(X(ii,kk)-X(jj,kk))
                  r=r+((X(ii,kk)-X(jj,kk))**2)
                enddo
                r=sqrt(r)
                if (r<(dx0(jj)*1.1)) then
                  split=1
                  if (BreakContact(ii)==1) then
                    if (BreakContact(jj)==2) then
                      split=0
                    endif
                  endif
                  if (BreakContact(ii)==2) then
                    if (BreakContact(jj)==1) then
                      split=0
                    endif
                  endif
                  split=1
                  if (split==1) then
                    ct=ct+1
                    Contact(jj,ct+1)=ii
                    fooint=MAXLOC(dx3,1)
                    ContactDir(jj,ct+1)=fooint
                    ContactDirCT(jj,fooint)=ContactDirCT(jj,fooint)+1
```

```
            endif
          endif
        endif
      endif
    enddo
    Contact(jj,1)=ct
    ContactDir(jj,1)=ct
  endif
enddo

ContactX=Contact
ContactDirX=ContactDir
ContactDirCTX=ContactDirCT



open (unit = 1, file = "Fixed.dat")
do jj=1,N
  write (1,*),FixXY(jj)
enddo
close (1)

open (unit = 1, file = "ContactDirX.dat")
do jj=1,N
  write (1,*),ContactDirX(jj,:)
enddo
close (1)

open (unit = 1, file = "Contact.dat")
do jj=1,N
  write (1,*),Contact(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactDir.dat")
do jj=1,N
  write (1,*),ContactDir(jj,:)
enddo
close (1)
```

```
open (unit = 1, file = "ContactDirCT.dat")
do jj=1,N
  write (1,*),ContactDirCT(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactX.dat")
do jj=1,N
  write (1,*),ContactX(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactDirCTX.dat")
do jj=1,N
  write (1,*),ContactDirCTX(jj,:)
enddo
close (1)

open (unit = 1, file = "Mass.dat")
do jj=1,N
  write (1,*),massFct(jj)
enddo
close (1)

open (unit = 1, file = "dX.dat")
do jj=1,N
  write (1,*),dx0(jj)
enddo
close (1)

open (unit = 1, file = "X0.dat")
do jj=1,N
  write (1,*),X(jj,:)
enddo
close (1)

open (unit = 1, file = "V0.dat")
do jj=1,N
  write (1,*),V(jj,:)
enddo
```

```
      close (1)

      open (unit = 1, file = "dVdX.dat")
      do jj=1,N
        write (1,*),dVdx(jj,:)
      enddo
      close (1)

      open (unit = 1, file = "rho0.dat")
      do jj=1,N
        write (1,*),rho0(jj)
      enddo
      close (1)

      open (unit = 1, file = "rho_init.dat")
      do jj=1,N
        write (1,*),rho(jj)
      enddo
      close (1)


      open (unit = 1, file = "IsSolid.dat")
      do jj=1,N
        write (1,*),IsSolid(jj)
      enddo
      close (1)


      Nout=N

      end subroutine ModelBlock

c _____________________________________________________________________
```

compress_steel.f

```
subroutine ModelBlock(Nout)

implicit none

integer, intent(out) :: Nout
integer N,ctbar(3),Nsphere,ct,ii,jj,kk,uu
integer fooint,split
double precision RadiusFactor,Radius,masX,dx00
double precision angleFrac,angle,Yr0,dx3(3),third
double precision Zloc,Yloc,Xloc,r,pi,rho_solid
integer, allocatable, dimension(:) :: FixXY,IsSolid,BreakContact
integer, allocatable, dimension(:,:) :: Contact,ContactX
integer, allocatable, dimension(:,:) :: ContactDir,ContactDirX
integer, allocatable, dimension(:,:) :: ContactDirCT,ContactDirC
double precision, allocatable, dimension(:,:) :: X,V,dVdx
double precision, allocatable, dimension(:) :: massFct,dx0
double precision, allocatable, dimension(:) :: rho0,rho

open (unit = 1, file = "SimSpec.dat")
  write (1,*),207e9        ! Young's Modulus of Steel
  write (1,*),0.3          ! Poisson's Ratio of Steel
  write (1,*),7800         ! Density of Steel
  write (1,*),0            ! Forced Density in X direction (Vtest
  write (1,*),-1e3         ! Forced Density in Y direction (Vtest
  write (1,*),0            ! Forced Density in Z direction (Vtest
  write (1,*),0.3          ! Gamma (blend stress)
  write (1,*),250          ! Number of Recorded Time Steps (Nts)
  write (1,*),6            ! Time Steps between recorded Time Ste
  write (1,*),1            ! Time Steps between recorded Time Ste
  write (1,*),0.18         ! (StartDefl)
  write (1,*),0.05         ! (NewDefl)
  write (1,*),0.25         ! (dTfactor)
  write (1,*),0.01         ! (dTdxfactor)
  write (1,*),250          ! (stop)
  write (1,*),2.15e9       ! (Bulk_water)
close (1)
```

```
      ctbar(1)=3     ! 3
      ctbar(2)=11    ! 11
      ctbar(3)=3     ! 3
      masX=0.1
      rho_solid=7800


      third=1.0/3.0
      dx00=(masX/rho_solid)**third
      N=(ctbar(1)*ctbar(2)*ctbar(3))


c _____________________________________________________
c                  ALLOCATION
c _____________________________________________________


      ALLOCATE (FixXY(N))
      ALLOCATE (IsSolid(N))
      ALLOCATE (BreakContact(N))
      ALLOCATE (X(N,3))
      ALLOCATE (V(N,3))
      ALLOCATE (dVdx(N,3))
      ALLOCATE (rho(N))
      ALLOCATE (rho0(N))
      ALLOCATE (massFct(N))
      ALLOCATE (dx0(N))
      ALLOCATE (Contact(N,(N+1)))
      ALLOCATE (ContactDir(N,(N+1)))
      ALLOCATE (ContactDirCT(N,3))
      ALLOCATE (ContactX(N,(N+1)))
      ALLOCATE (ContactDirX(N,(N+1)))
      ALLOCATE (ContactDirCTX(N,3))

c _____________________________________________________
```

```
c Set the parameters for the steel bar
      ct = 0
      Zloc=-dx00
      do uu=1,ctbar(3)
        Zloc=Zloc+dx00
        Yloc=-dx00
        do jj=1,ctbar(2)
          Yloc = Yloc + dx00
          Xloc = 0
          do ii=1,ctbar(1)
            ct = ct + 1
            if (jj==ctbar(2)) then
              FixXY(ct)=12
            elseif (jj==1) then
              FixXY(ct)=13
            else
              FixXY(ct)=0
            endif
            Xloc = Xloc + dx00
            X(ct,1) = Xloc
            X(ct,2) = Yloc
            X(ct,3) = Zloc
            V(ct,1) = 0
            V(ct,2) = 0
            V(ct,3) = 0
            dVdX(ct,1) = 0
            dVdX(ct,2) = 0
            dVdX(ct,3) = 0
            rho(ct) = rho_solid
            rho0(ct) = rho_solid
            dx0(ct) = dx00
            IsSolid(ct) = 1
            massFct(ct) = masX
            if (jj==(ctbar(2)/2)) then
              BreakContact(ct)=1
            elseif (jj==(1+(ctbar(2)/2))) then
              BreakContact(ct)=2
            else
              BreakContact(ct)=0
```

```
            endif
          enddo
        enddo
      enddo


c Set Contact links
      do jj=1,N
        if (IsSolid(jj)==1) then
          ct=0
          do ii=1,N
            if (IsSolid(ii)==1) then
              if (ii/=jj) then
                r=0
                do kk=1,3
                  dx3(kk)=abs(X(ii,kk)-X(jj,kk))
                  r=r+((X(ii,kk)-X(jj,kk))**2)
                enddo
                r=sqrt(r)
                if (r<(dx0(jj)*1.1)) then
                  split=1
                  if (BreakContact(ii)==1) then
                    if (BreakContact(jj)==2) then
                      split=0
                    endif
                  endif
                  if (BreakContact(ii)==2) then
                    if (BreakContact(jj)==1) then
                      split=0
                    endif
                  endif
                  split=1
                  if (split==1) then
                    ct=ct+1
                    Contact(jj,ct+1)=ii
                    fooint=MAXLOC(dx3,1)
                    ContactDir(jj,ct+1)=fooint
                    ContactDirCT(jj,fooint)=ContactDirCT(jj,fooint)+1
                  endif
                endif
```

```
        endif
      endif
    enddo
    Contact(jj,1)=ct
    ContactDir(jj,1)=ct
  endif
enddo

ContactX=Contact
ContactDirX=ContactDir
ContactDirCTX=ContactDirCT



open (unit = 1, file = "Fixed.dat")
do jj=1,N
  write (1,*),FixXY(jj)
enddo
close (1)

open (unit = 1, file = "ContactDirX.dat")
do jj=1,N
  write (1,*),ContactDirX(jj,:)
enddo
close (1)

open (unit = 1, file = "Contact.dat")
do jj=1,N
  write (1,*),Contact(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactDir.dat")
do jj=1,N
  write (1,*),ContactDir(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactDirCT.dat")
do jj=1,N
```

```
  write (1,*),ContactDirCT(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactX.dat")
do jj=1,N
  write (1,*),ContactX(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactDirCTX.dat")
do jj=1,N
  write (1,*),ContactDirCTX(jj,:)
enddo
close (1)

open (unit = 1, file = "Mass.dat")
do jj=1,N
  write (1,*),massFct(jj)
enddo
close (1)

open (unit = 1, file = "dX.dat")
do jj=1,N
  write (1,*),dx0(jj)
enddo
close (1)

open (unit = 1, file = "X0.dat")
do jj=1,N
  write (1,*),X(jj,:)
enddo
close (1)

open (unit = 1, file = "V0.dat")
do jj=1,N
  write (1,*),V(jj,:)
enddo
close (1)
```

```
      open (unit = 1, file = "dVdX.dat")
      do jj=1,N
        write (1,*),dVdx(jj,:)
      enddo
      close (1)

      open (unit = 1, file = "rho0.dat")
      do jj=1,N
        write (1,*),rho0(jj)
      enddo
      close (1)

      open (unit = 1, file = "rho_init.dat")
      do jj=1,N
        write (1,*),rho(jj)
      enddo
      close (1)


      open (unit = 1, file = "IsSolid.dat")
      do jj=1,N
        write (1,*),IsSolid(jj)
      enddo
      close (1)


      Nout=N

      end subroutine ModelBlock

c _____________________________________________________________________
```

ModelSmFixDisk.f

```
      subroutine ModelBlock(Nout)

      implicit none

      integer, intent(out) :: Nout
      integer N,ctbar(3),Nsphere,ct,ii,jj,kk,DiskRat
      integer fooint
      double precision RadiusFactor,Radius,masX,masXdisk,dx0x,dx0xDisk
      double precision angleFrac,angle,Yr0,dx3(3)
      double precision Zloc,Zloc2,Yloc,Xloc,r,mass,pi,rho_solid
      integer, allocatable, dimension(:) :: FixXY,IsSolid
      integer, allocatable, dimension(:,:) :: Contact,ContactX
      integer, allocatable, dimension(:,:) :: ContactDir,ContactDirX
      integer, allocatable, dimension(:,:) :: ContactDirCT,ContactDirC
      double precision, allocatable, dimension(:,:) :: X,V,dVdx
      double precision, allocatable, dimension(:) :: massFct,dx0,rho0


c     Start the Specification Parameters

      open (unit = 1, file = "SimSpec.dat")
        write (1,*),207e9        ! Young's Modulus of Steel
        write (1,*),0.3          ! Poisson's Ratio of Steel
        write (1,*),7800         ! Density of Steel
        write (1,*),0.0          ! Forced Density in X direction (Vtest
        write (1,*),100.0        ! Forced Density in Y direction (Vtest
        write (1,*),0.0          ! Forced Density in Z direction (Vtest
        write (1,*),0.3          ! Gamma (blend stress)
        write (1,*),100          ! Number of Recorded Time Steps (Nts)
        write (1,*),1            ! Time Steps between recorded Time Ste
        write (1,*),1            ! Time Steps between recorded Time Ste
        write (1,*),5*2.54e-6    ! (StartDefl)
        write (1,*),5*2.54e-6    ! (NewDefl)
        write (1,*),0.05         ! (dTfactor)
        write (1,*),0.005         ! (dTdxfactor)
        write (1,*),15           ! (stop)
```

```
        write (1,*),2.15e9       ! (Bulk_water)
      close (1)



      ctbar(1)=101
      ctbar(2)=10
      ctbar(3)=1
      Radius=20.0*(2.54e-4)
      DiskRat=3
      dx0x=0.25*(Radius/20.0)
      pi=ACOS(-1.0)
      rho_solid=7800

      if ((ctbar(1)*dx0x)>(2*Radius)) then
        angleFrac=pi
      else
        angleFrac=2*ASIN((ctbar(1)*dx0x)/(2*Radius))
        angleFrac=angleFrac/2
      endif

      dx0xDisk=dx0x/DiskRat
      masX=(dx0x**3)*rho_solid
      masXdisk=(dx0xDisk**3)*rho_solid

      Nsphere=(angleFrac*Radius/dx0xDisk)
      N=((Nsphere*DiskRat)+(ctbar(1)*ctbar(2)))*ctbar(3)

c _____________________________________________________
c                  ALLOCATION
c _____________________________________________________



      ALLOCATE (FixXY(N))
      ALLOCATE (IsSolid(N))
      ALLOCATE (X(N,3))
      ALLOCATE (V(N,3))
      ALLOCATE (dVdx(N,3))
      ALLOCATE (rho0(N))
```

```
      ALLOCATE (massFct(N))
      ALLOCATE (dx0(N))
      ALLOCATE (Contact(N,(N+1)))
      ALLOCATE (ContactDir(N,(N+1)))
      ALLOCATE (ContactDirCT(N,3))
      ALLOCATE (ContactX(N,(N+1)))
      ALLOCATE (ContactDirX(N,(N+1)))
      ALLOCATE (ContactDirCTX(N,3))

c _____________________________________________________


c Set the parameters for the steel bar


      Yr0=(ctbar(2)*dx0x)+Radius
      ct=0
      Zloc=-dx0x
      do kk=1,ctbar(3)
        Zloc=Zloc+dx0x
        Yloc=-dx0x
        do jj=1,ctbar(2)
          Yloc=Yloc+dx0x
          Xloc=-((dx0x*ctbar(1))/2)-(dx0x/2)
          do ii=1,ctbar(1)
            Xloc=Xloc+dx0x
            ct=ct+1
            if (jj==1) then
              FixXY(ct)=1
            else
              FixXY(ct)=0
            endif
            X(ct,1) = Xloc
            X(ct,2) = Yloc
            X(ct,3) = Zloc
            V(ct,1) = 0
            V(ct,2) = 0
            V(ct,3) = 0
            dVdX(ct,1) = 0
```

```
            dVdX(ct,2) = 0
            dVdX(ct,3) = 0
            rho0(ct) = rho_solid
            massFct(ct) = masX
            dx0(ct) = (massFct(ct)/rho_solid)**0.333333
            IsSolid(ct) = 1
          enddo
        enddo

        Zloc2=Zloc-(dx0xDisk*(DiskRat+1)/2)
        do ii=1,DiskRat
          Zloc2=Zloc2+dx0xDisk
          angle=(angleFrac/2)+(angleFrac/(2*Nsphere))
          do jj=1,Nsphere
            angle=angle-(angleFrac/Nsphere)
            Xloc=-(SIN(angle))*Radius
            Yloc=Yr0-((COS(angle))*Radius)
            ct=ct+1
            FixXY(ct)=8
            X(ct,1) = Xloc
            X(ct,2) = Yloc
            X(ct,3) = Zloc2
            V(ct,1) = 0
            V(ct,2) = 0
            V(ct,3) = 0
            dVdX(ct,1) = 0
            dVdX(ct,2) = 0
            dVdX(ct,3) = 0
            rho0(ct) = rho_solid
            massFct(ct) = masXdisk
            dx0(ct) = (massFct(ct)/rho_solid)**0.333333
            IsSolid(ct) = 1
          enddo
        enddo
      enddo
      print *,N,ct


c Set Contact links
      do jj=1,N
```

```
  if (IsSolid(jj)==1) then
    ct=0
    do ii=1,N
      if (IsSolid(ii)==1) then
        if (ii/=jj) then
          r=0
          do kk=1,3
            dx3(kk)=abs(X(ii,kk)-X(jj,kk))
            r=r+((X(ii,kk)-X(jj,kk))**2)
          enddo
          r=sqrt(r)
          if (r<(dx0(jj)*1.1)) then
            if (FixXY(jj)==FixXY(ii)) then
              ct=ct+1
              Contact(jj,ct+1)=ii
              fooint=MAXLOC(dx3,1)
              ContactDir(jj,ct+1)=fooint
              ContactDirCT(jj,fooint)=ContactDirCT(jj,fooint)+1
            endif
          endif
        endif
      endif
    enddo
    Contact(jj,1)=ct
    ContactDir(jj,1)=ct
  endif
enddo

ContactX=Contact
ContactDirX=ContactDir
ContactDirCTX=ContactDirCT


open (unit = 1, file = "Fixed.dat")
do jj=1,N
  write (1,*),FixXY(jj)
enddo
close (1)

open (unit = 1, file = "ContactDirX.dat")
```

```
do jj=1,N
  write (1,*),ContactDirX(jj,:)
enddo
close (1)

open (unit = 1, file = "Contact.dat")
do jj=1,N
  write (1,*),Contact(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactDir.dat")
do jj=1,N
  write (1,*),ContactDir(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactDirCT.dat")
do jj=1,N
  write (1,*),ContactDirCT(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactX.dat")
do jj=1,N
  write (1,*),ContactX(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactDirCTX.dat")
do jj=1,N
  write (1,*),ContactDirCTX(jj,:)
enddo
close (1)

open (unit = 1, file = "Mass.dat")
do jj=1,N
  write (1,*),massFct(jj)
enddo
close (1)
```

```
open (unit = 1, file = "dX.dat")
do jj=1,N
  write (1,*),dx0(jj)
enddo
close (1)

open (unit = 1, file = "X0.dat")
do jj=1,N
  write (1,*),X(jj,:)
enddo
close (1)

open (unit = 1, file = "V0.dat")
do jj=1,N
  write (1,*),V(jj,:)
enddo
close (1)

open (unit = 1, file = "dVdX.dat")
do jj=1,N
  write (1,*),dVdx(jj,:)
enddo
close (1)

open (unit = 1, file = "rho0.dat")
do jj=1,N
  write (1,*),rho0(jj)
enddo
close (1)

open (unit = 1, file = "rho_init.dat")
do jj=1,N
  write (1,*),rho0(jj)
enddo
close (1)

open (unit = 1, file = "IsSolid.dat")
do jj=1,N
  write (1,*),IsSolid(jj)
```

```
      enddo
      close (1)


      Nout=N

      end subroutine ModelBlock


c _____________________________________________________________________
```

PressVessel.f

```
subroutine ModelBlock(Nout)

implicit none

integer, intent(out) :: Nout
integer N,ctPV(3),Nsphere,ct,ii,jj,kk,uu
integer fooint,split
double precision RadiusFactor,Radius,masX,massL,dx00,Bulk_water
double precision angleFrac,angle,Yr0,dx3(3),third,RR,pi,P
double precision Zloc,Yloc,Xloc,rho_solid,rho_liquid,rho_liquidC
integer, allocatable, dimension(:) :: FixXY,IsSolid,BreakContact
integer, allocatable, dimension(:,:) :: Contact,ContactX
integer, allocatable, dimension(:,:) :: ContactDir,ContactDirX
integer, allocatable, dimension(:,:) :: ContactDirCT,ContactDirC
double precision, allocatable, dimension(:,:) :: X,V,dVdx
double precision, allocatable, dimension(:) :: dx0,massFct
double precision, allocatable, dimension(:) :: rho0,rho

open (unit = 1, file = "SimSpec.dat")
  write (1,*),207e9        ! Young's Modulus of Steel
  write (1,*),0.3          ! Poisson's Ratio of Steel
  write (1,*),7800         ! Density of Steel
  write (1,*),0.0          ! Forced Density in X direction (Vtest
  write (1,*),0.0          ! Forced Density in Y direction (Vtest
  write (1,*),0.0          ! Forced Density in Z direction (Vtest
  write (1,*),0.3          ! Gamma (blend stress)
  write (1,*),1000         ! Number of Recorded Time Steps (Nts)
  write (1,*),1            ! Time Steps between recorded Time Ste
  write (1,*),1            ! Time Steps between recorded Time Ste
  write (1,*),0.18         ! (StartDefl)
  write (1,*),0.05         ! (NewDefl)
  write (1,*),0.05         ! (dTfactor)
  write (1,*),0.01         ! (dTdxfactor)
  write (1,*),15           ! (stop)
  write (1,*),2.15e9       ! (Bulk_water)
close (1)
```

```
      ctPV(1)=25     ! 12
      ctPV(2)=20     ! 9
      ctPV(3)=1      ! 1

      masX=0.1
      P=101135*1
      rho_solid=7800
      rho_liquid=1000
      Bulk_water=2.15e9

      rho_liquidC=rho_liquid*((P/Bulk_water)+1)
      print *,rho_liquidC

      third=1.0/3.0
      dx00=((masX/rho_solid)**third)
      massL=rho_liquidC*(dx00**3)

      N=0
      Xloc=(-ctPV(1)*dx00)-(dx00/2)
      do ii=1,(2*ctPV(1))
      Xloc=Xloc+dx00
      Yloc=(-ctPV(1)*dx00)-(dx00/2)
      do jj=1,(2*ctPV(1))
        Yloc=Yloc+dx00
        RR=SQRT((Xloc**2)+(Yloc**2))
        if (RR<(ctPV(1)*dx00)) then
          N=N+1
        endif
      enddo
      enddo
      N=N*ctPV(3)
      print *,N


c _____________________________________________
c                 ALLOCATION
c _____________________________________________
```

```
      ALLOCATE (FixXY(N))
      ALLOCATE (IsSolid(N))
      ALLOCATE (BreakContact(N))
      ALLOCATE (X(N,3))
      ALLOCATE (V(N,3))
      ALLOCATE (dVdx(N,3))
      ALLOCATE (rho(N))
      ALLOCATE (rho0(N))
      ALLOCATE (massFct(N))
      ALLOCATE (dx0(N))
      ALLOCATE (Contact(N,(N+1)))
      ALLOCATE (ContactDir(N,(N+1)))
      ALLOCATE (ContactDirCT(N,3))
      ALLOCATE (ContactX(N,(N+1)))
      ALLOCATE (ContactDirX(N,(N+1)))
      ALLOCATE (ContactDirCTX(N,3))

c _____________________________________________


      print *,N

c Set the parameters for the steel bar

      ct=0
      Zloc=-dx00
      do uu=1,(ctPV(3))
        Zloc=Zloc+dx00
        Xloc=(-ctPV(1)*dx00)-(dx00/2)
        do ii=1,(2*ctPV(1))
          Xloc=Xloc+dx00
          Yloc=(-ctPV(1)*dx00)-(dx00/2)
          do jj=1,(2*ctPV(1))
            Yloc=Yloc+dx00
            RR=SQRT((Xloc**2)+(Yloc**2))
            if (RR<(ctPV(1)*dx00)) then
              ct = ct + 1
              if (RR<(ctPV(2)*dx00)) then
                IsSolid(ct) = 0
                massFct(ct) = massL
```

```
                rho0(ct)=rho_liquid
                rho(ct)=rho_liquidC
              else
                IsSolid(ct) = 1
                massFct(ct) = masX
                rho0(ct)=rho_solid
                rho(ct)=rho_solid
              endif
              FixXY(ct)=0
              X(ct,1) = Xloc
              X(ct,2) = Yloc
              X(ct,3) = Zloc

            endif
          enddo
        enddo
      enddo

      dx0 = dx0 + (dx00)


c Set Contact links
      do jj=1,N
        if (IsSolid(jj)==1) then
          ct=0
          do ii=1,N
            if (IsSolid(ii)==1) then
              if (ii/=jj) then
                RR=0
                do kk=1,3
                  dx3(kk)=abs(X(ii,kk)-X(jj,kk))
                  RR=RR+((X(ii,kk)-X(jj,kk))**2)
                enddo
                RR=sqrt(RR)
                if (RR<(dx0(jj)*1.1)) then
                  ct=ct+1
                  Contact(jj,ct+1)=ii
                  fooint=MAXLOC(dx3,1)
                  ContactDir(jj,ct+1)=fooint
                  ContactDirCT(jj,fooint)=ContactDirCT(jj,fooint)+1
```

```
          endif
        endif
      endif
    enddo
    Contact(jj,1)=ct
    ContactDir(jj,1)=ct
  endif
enddo

ContactX=Contact
ContactDirX=ContactDir
ContactDirCTX=ContactDirCT



open (unit = 1, file = "Fixed.dat")
do jj=1,N
  write (1,*),FixXY(jj)
enddo
close (1)

open (unit = 1, file = "ContactDirX.dat")
do jj=1,N
  write (1,*),ContactDirX(jj,:)
enddo
close (1)

open (unit = 1, file = "Contact.dat")
do jj=1,N
  write (1,*),Contact(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactDir.dat")
do jj=1,N
  write (1,*),ContactDir(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactDirCT.dat")
```

```
do jj=1,N
  write (1,*),ContactDirCT(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactX.dat")
do jj=1,N
  write (1,*),ContactX(jj,:)
enddo
close (1)

open (unit = 1, file = "ContactDirCTX.dat")
do jj=1,N
  write (1,*),ContactDirCTX(jj,:)
enddo
close (1)

open (unit = 1, file = "Mass.dat")
do jj=1,N
  write (1,*),massFct(jj)
enddo
close (1)

open (unit = 1, file = "dX.dat")
do jj=1,N
  write (1,*),dx0(jj)
enddo
close (1)

open (unit = 1, file = "X0.dat")
do jj=1,N
  write (1,*),X(jj,:)
enddo
close (1)

open (unit = 1, file = "V0.dat")
do jj=1,N
  write (1,*),V(jj,:)
enddo
close (1)
```

```
      open (unit = 1, file = "dVdX.dat")
      do jj=1,N
        write (1,*),dVdx(jj,:)
      enddo
      close (1)

      open (unit = 1, file = "rho0.dat")
      do jj=1,N
        write (1,*),rho0(jj)
      enddo
      close (1)

      open (unit = 1, file = "rho_init.dat")
      do jj=1,N
        write (1,*),rho(jj)
      enddo
      close (1)

      open (unit = 1, file = "IsSolid.dat")
      do jj=1,N
        write (1,*),IsSolid(jj)
      enddo
      close (1)


      Nout=N

      end subroutine ModelBlock

c _____________________________________________________________________
```

MATLAB SOURCE CODE

Analyze_Data.m

```
clear
close all
tic

Position=[200 200 900 800];
TitleSz=24; LblSz=20; AxSz=16;
NL=sprintf('\n');

raw=load('VarDat.dat');
Ey=raw(1);
G=raw(2);
StudyCt=raw(3);

V0=0;

X0=load('X0.dat');
Xdat=load('X.dat');
Ydat=load('Y.dat');
Zdat=load('Z.dat');
Vx=load('Vx.dat');
Vy=load('Vy.dat');
dVdx=load('dVdX.dat');
dVdy=load('dVdY.dat');
dVdz=load('dVdZ.dat');
P=load('P.dat');
rho=load('rho.dat');
Fixed=load('Fixed.dat');
```

```
E11=load('E11.dat');
E22=load('E22.dat');
dx=load('dX.dat');
T11=load('T11.dat');
T21=load('T21.dat');
T31=load('T31.dat');
T12=load('T12.dat');
T22=load('T22.dat');
T32=load('T32.dat');
T13=load('T13.dat');
T23=load('T23.dat');
T33=load('T13.dat');
IsSolid=load('IsSolid.dat');
Mass=load('Mass.dat');
rho0=load('rho0.dat');
rho_init=load('rho_init.dat');
TimeFct=load('Time.dat');

Cylrng=find(Fixed==8);
R1=max(X0(Cylrng,2))-min(X0(Cylrng,2))+(mean(dx)/2);

foo=size(Xdat);
ct=foo(1); stop=ct; a=ct;
N=foo(2); clear foo
if stop>ct
    stop=ct;
end
plotCt=5; Range=(1:plotCt)*floor(ct/plotCt);
Vy(:,1)=-V0; Vy(:,N)=V0;
Poisson=(Ey/(2*G))-1;

Ered=1/((1-(Poisson^2))/Ey);

Xdat2=Xdat;
Ydat2=Ydat;
Zdat2=Zdat;
for ii=1:N
    Xdat2(:,ii)=Xdat2(:,ii)-(X0(ii,1));
    Ydat2(:,ii)=Ydat2(:,ii)-(X0(ii,2));
    Zdat2(:,ii)=Zdat2(:,ii)-(X0(ii,3));
```

```
end

Xdat3=Xdat2;
Ydat3=Ydat2;
Zdat3=Zdat2;
for ii=1:ct
    for jj=1:N
        Xdat3(ii,jj)=Xdat3(ii,jj)-mean(Xdat2(ii,:));
        Ydat3(ii,jj)=Ydat3(ii,jj)-mean(Ydat2(ii,:));
        Zdat3(ii,jj)=Zdat3(ii,jj)-mean(Zdat2(ii,:));
    end
end

NL=(['\n']);
%dtplot=load('Time.dat');

Vr=sqrt((Vx.^2)+(Vy.^2));
Xr=sqrt((Xdat.^2)+(Ydat.^2));
Xr2=Xr;
for ii=1:ct
    Xr2=Xr(ii,:)-Xr(1,:);
end

save Data

toc
```

PlotTension.m

```
clear
close all
tic

Position=[200 200 900 800];
TitleSz=24; LblSz=20; AxSz=16;
NL=sprintf('\n');

% load Data
load Tension_steel

X=Xdat(end,:);
Y=Ydat(end,:);
Z=Zdat(end,:);

Xdat2=Xdat;
Ydat2=Ydat;
Zdat2=Zdat;
for ii=1:N
    Xdat2(:,ii)=Xdat2(:,ii)-((Xdat2(1,ii)));
    Ydat2(:,ii)=Ydat2(:,ii)-((Ydat2(1,ii)));
    Zdat2(:,ii)=Zdat2(:,ii)-((Zdat2(1,ii)));
end

Poisson=(Ey/(2*G))-1;

exagY=1;
exag=2e1;
exagstr='(20) ';
matstr='Steel';

Xe=Xdat(1,:)+(Xdat2(end,:)*exag);
Ye=Ydat(1,:)+(Ydat2(end,:)*exagY);
Ze=Zdat(1,:)+(Zdat2(end,:)*exag);

dtplot=(TimeFct(:,1)); dt=max(diff(TimeFct)); dx=median(dx);
height=max(X0(:,2))-min(X0(:,2))+dx;
```

```
width=max(X0(:,1))-min(X0(:,1))+dx;

ff=find(X0(:,2)==max(X0(:,2)));
Tcalc=Ey*max(Ydat2(:,:)')/height;
Tavg=max(abs(T22(:,ff)'));


YY=zeros(1,N);
for ii=2:(N-1)
    YY(ii)=(abs(Y(ii)-Y(ii-1))-dx)+(abs(Y(ii)-Y(ii+1))-dx);
    YY(ii)=YY(ii)*Ey/dx;
end
ratio=max(T22').*((min(T22').^-1)); ratio=ratio-1;



PoissonError=100*abs((Poisson/((max(abs(Xdat2(ct,:)))/width)/(max(abs(Y
TensionError = 100*abs((Tcalc(ct)/Tavg(ct))-1);

MaxStress=abs(Tavg(ct));

dxmat=(max(X0(:,2))-min(X0(:,2)))/20;
axismat=[(min(Xe)-dxmat) (max(Xe)+dxmat) (min(Ye)-dxmat) (max(Ye)+dxmat
TCstr='Tension';
MaxStressStr=[num2str(MaxStress*(1e-6)) ' MPa'];
TitleStr=[matstr ' - Max Stress = ' MaxStressStr 10];

figure(1)
set(gcf,'Position',Position)
set(0,'DefaultAxesFontSize',AxSz,'DefaultAxesFontWeight','b')
hold off
for ii=1:N
    if Fixed(ii)==0
        plot(Xe(ii),Ye(ii),'ko','LineWidth',4)
        hold on
    else
        plot(Xe(ii),Ye(ii),'kx','LineWidth',4)
        hold on

    end
```

```
end
axis(axismat)
title([TitleStr 'Exaggerated ' exagstr TCstr],'fontsize',TitleSz,'fontw
xlabel('X','fontsize',LblSz,'fontweight','b');
ylabel('Y','fontsize',LblSz,'fontweight','b');

figure(2)
set(gcf,'Position',Position)
set(0,'DefaultAxesFontSize',AxSz,'DefaultAxesFontWeight','b')
hold off
for ii=1:N
    if Fixed(ii)==0
        plot(Xdat(1,ii),Ydat(1,ii),'ko','LineWidth',4)
        hold on
    else
        plot(Xdat(1,ii),Ydat(1,ii),'kx','LineWidth',4)
        hold on

    end
end
axis(axismat)
title([TitleStr 'Before ' TCstr],'fontsize',TitleSz,'fontweight','b');
xlabel('X','fontsize',LblSz,'fontweight','b');
ylabel('Y','fontsize',LblSz,'fontweight','b');

display(' ');
display(['Poisson Error = ' num2str(PoissonError) '%']);
display(['Tension Error = ' num2str(TensionError) '%']);
display(' ');

toc
```

PlotComp.m

```
clear
close all
tic

Position=[200 200 900 800];
TitleSz=24; LblSz=20; AxSz=16;
NL=sprintf('\n');

% load Data
load compress_steel

X=Xdat(end,:);
Y=Ydat(end,:);
Z=Zdat(end,:);

Xdat2=Xdat;
Ydat2=Ydat;
Zdat2=Zdat;
for ii=1:N
    Xdat2(:,ii)=Xdat2(:,ii)-((Xdat2(1,ii)));
    Ydat2(:,ii)=Ydat2(:,ii)-((Ydat2(1,ii)));
    Zdat2(:,ii)=Zdat2(:,ii)-((Zdat2(1,ii)));
end

Poisson=(Ey/(2*G))-1;

exagY=1;
exag=2e1;
exagstr='(20) ';
matstr='Steel';

Xe=Xdat(1,:)+(Xdat2(end,:)*exag);
Ye=Ydat(1,:)+(Ydat2(end,:)*exagY);
Ze=Zdat(1,:)+(Zdat2(end,:)*exag);

dtplot=(TimeFct(:,1)); dt=max(diff(TimeFct)); dx=median(dx);
height=max(X0(:,2))-min(X0(:,2))+dx;
```

```
width=max(X0(:,1))-min(X0(:,1))+dx;

ff=find(X0(:,2)==max(X0(:,2)));
Tcalc=Ey*max(Ydat2(:,:)')/height;
Tavg=max(abs(T22(:,ff)'));


YY=zeros(1,N);
for ii=2:(N-1)
    YY(ii)=(abs(Y(ii)-Y(ii-1))-dx)+(abs(Y(ii)-Y(ii+1))-dx);
    YY(ii)=YY(ii)*Ey/dx;
end
ratio=max(T22').*((min(T22').^-1)); ratio=ratio-1;



PoissonError=100*abs((Poisson/((max(abs(Xdat2(ct,:)))/width)/(max(abs(Y
TensionError = 100*abs((Tcalc(ct)/Tavg(ct))-1);

MaxStress=abs(Tavg(ct));

dxmat=(max(X0(:,2))-min(X0(:,2)))/20;
axismat=[(min(Xe)-dxmat) (max(Xe)+dxmat) (min(Ye)-dxmat) (max(Ye)+dxmat

TCstr='Compression';
MaxStressStr=[num2str(MaxStress*(1e-6)) ' MPa'];

TitleStr=[matstr ' - Max Stress = ' MaxStressStr 10];

figure(1)
set(gcf,'Position',Position)
set(0,'DefaultAxesFontSize',AxSz,'DefaultAxesFontWeight','b')
hold off
for ii=1:N
    if Fixed(ii)==0
        plot(Xe(ii),Ye(ii),'ko','LineWidth',4)
        hold on
    else
        plot(Xe(ii),Ye(ii),'kx','LineWidth',4)
        hold on
```

```
    end
end
axis(axismat)
title([TitleStr 'Exaggerated ' exagstr TCstr],'fontsize',TitleSz,'fontw
xlabel('X','fontsize',LblSz,'fontweight','b');
ylabel('Y','fontsize',LblSz,'fontweight','b');

figure(2)
set(gcf,'Position',Position)
set(0,'DefaultAxesFontSize',AxSz,'DefaultAxesFontWeight','b')
hold off
for ii=1:N
    if Fixed(ii)==0
        plot(Xdat(1,ii),Ydat(1,ii),'ko','LineWidth',4)
        hold on
    else
        plot(Xdat(1,ii),Ydat(1,ii),'kx','LineWidth',4)
        hold on

    end
end
axis(axismat)
title([TitleStr 'Before ' TCstr],'fontsize',TitleSz,'fontweight','b');
xlabel('X','fontsize',LblSz,'fontweight','b');
ylabel('Y','fontsize',LblSz,'fontweight','b');

display(' ');
display(['Poisson Error = ' num2str(PoissonError) '%']);
display(['Tension Error = ' num2str(TensionError) '%']);
display(' ');

toc
```

Plot_Hertz.m

```
clear

load Hertz

X=Xdat; Y=Ydat; Z=Zdat;
[ct N]=size(X);

rng=(1:101)+(101*9);

for ii=[1 5:5:ct]
    figure(1)
    plot(X(ii,rng),Y(ii,rng),'*')
    title([num2str(ii) '/' num2str(ct)]);
    pause(0.1)
end

figure(2)
plot(Xdat(1,:),Ydat(1,:),'*')
```

MakeFigs_PresVes.m

```
clear all
close all

load PressVess

Position=[200 200 900 600];
TitleSz=24; LblSz=20; AxSz=16;
MkFig=0; % Set to 1 to save JPEG figures
Res='-r600';
NL=sprintf('\n');

aa=find(IsSolid==1); bb=find(IsSolid==0);
TimeFct=mean(TimeFct');

del=1.10;
axismat=[(del*min(X0(:,1))) (del*max(X0(:,1))) (del*min(X0(:,2))) (del

ha=figure(1);
set(gcf,'Position',Position)
set(0,'DefaultAxesFontSize',AxSz,'DefaultAxesFontWeight','b')
plot(X0(aa,1),X0(aa,2),'*',X0(bb,1),X0(bb,2),'o','Linewidth',2)
axis(axismat);
%legend('Hertz','SPAM','Location','Best');
title('Before Simulation','fontsize',LblSz,'fontweight','b');
ylabel('Y (m)','fontsize',LblSz,'fontweight','b')
xlabel('X (m)','fontsize',LblSz,'fontweight','b');
if MkFig==1
    print(ha,'-djpeg','Start.jpeg',Res)
    close all
end

ha=figure(2);
set(gcf,'Position',Position)
set(0,'DefaultAxesFontSize',AxSz,'DefaultAxesFontWeight','b')
plot(Xdat(end,aa),Ydat(end,aa),'*',Xdat(end,bb),Ydat(end,bb),'o','Linew
axis(axismat);
%legend('Hertz','SPAM','Location','Best');
```

```
title('After Simulation','fontsize',LblSz,'fontweight','b');
ylabel('Y (m)','fontsize',LblSz,'fontweight','b')
xlabel('X (m)','fontsize',LblSz,'fontweight','b');
if MkFig==1
    print(ha,'-djpeg','Final.jpeg',Res)
    close all
end

Xr2=Xr;
for ii=1:length(Xr2)
    Xr2(:,ii)=Xr(:,ii)-Xr(1,ii);
end

ha=figure(3);
set(gcf,'Position',Position)
set(0,'DefaultAxesFontSize',AxSz,'DefaultAxesFontWeight','b')
plot(TimeFct*(1e6),-Xr2(:,bb)*(1e9),'Linewidth',2)
title('Radial Displacement - Liquid','fontsize',LblSz,'fontweight','b')
ylabel(['\delta'  'R (nm)'],'fontsize',LblSz,'fontweight','b')
xlabel(['Time (' '{\mu}' 's)'],'fontsize',LblSz,'fontweight','b');
if MkFig==1
    print(ha,'-djpeg','Rfct_Liquid.jpeg',Res)
    close all
end

ha=figure(4);
set(gcf,'Position',Position)
set(0,'DefaultAxesFontSize',AxSz,'DefaultAxesFontWeight','b')
plot(TimeFct*(1e6),-Xr2(:,aa)*(1e9),'Linewidth',2)
title('Radial Displacement - Solid','fontsize',LblSz,'fontweight','b')
ylabel(['\delta'  'R (nm)'],'fontsize',LblSz,'fontweight','b')
xlabel(['Time (' '{\mu}' 's)'],'fontsize',LblSz,'fontweight','b');
if MkFig==1
    print(ha,'-djpeg','Rfct_Solid.jpeg',Res)
    close all
end
```

Hertz Study Results

This report discusses a detailed description of all of the simulation results when studying the Hertz Contact Stress of a Disk and Flat plat in compressive contact.

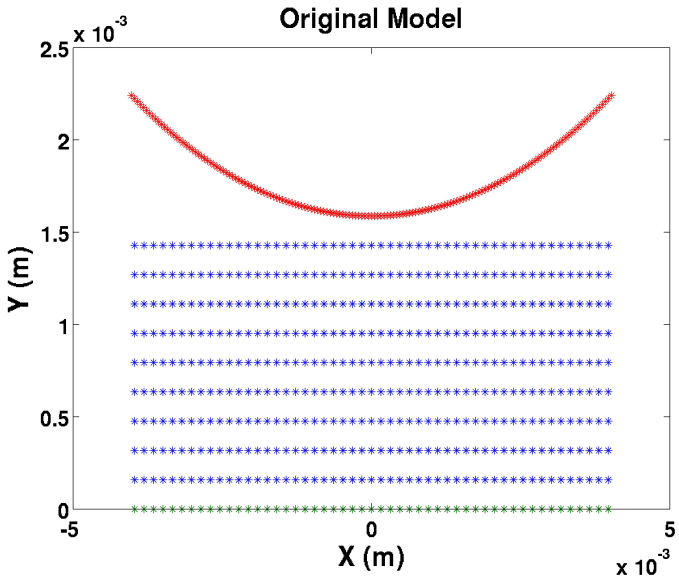
About the Study:

- The purpose of this study is to better verify and validate the Smoothed Particle Applied Mechanics (SPAM) model as it applies to Hertzian contact-mechanics.
- While the simulation is capable of 3D studies, only a 2D layer of particles are studied.
- The model will be represented by an inelastic disk being in contact with a elastic flat plate.
- The fixed 2D disk will be comprised of the same Lagrangian particles as the flat plate.
- The disk will be forced down at a user-specified velocity and then stay in place for a specified number of time steps.
- The plate will rest on a boundary of fixed solid particles beneath it.
- This effort will ensure that the deflection length is less than one tenth the disk radius.

Hertzian Equations for Comparison of SPAM:

- $P_{max} = |T_{22}|$ for the Top Center Particle
- $\delta = |Y(t) - Y(0)|$ for the Top Center Particle
- $a = R \sin\left(\cos^{-1}\left(\frac{R-\delta}{R}\right)\right)$
  - R = Radius of Disk
  - a = half length of (theoretical) contact area
- Reduced Modulus:
  - $\frac{1}{E} = \frac{1-\nu_1^2}{E_{y,1}} + \frac{1-\nu_2^2}{E_{y,2}}$
    - Ey = Young's Modulus of Elasticity (Pa)
    - ν = Poisson's Ration
    - 1 and 2 represent the parameters of the disk and plate respectively
  - The disk is assumed to be rigid and inelastic, and thus is assumed to have a Young's Modulus of infinity. Based on the assumptions taken for the disk radius, though it is safe to assume the disk is rigid, and thus the Young's Modulus is infinite, therefore:
    - $E = \frac{E_y}{1-\nu^2}$
- $Weight_N(N/m) = \frac{a^2 \pi E}{4R}$
  - This is the normalized weight, or the weight per unit length of the disk
- The calculated maximum stress is found via:
  - $Max\,Stress_{calculated} = \frac{2\,Weight_N}{\pi a}$
- Simulated Weight
  - $Weight = \Sigma_N(T_{22} * dx^2)$
  - Taken for all the fixed particles at the bottom of the flat disk
- The simulated weight is compared to a calculated total weight based on the simulated (with SPAM) maximum stress at the center of the top of the flat.
  - $Max\,Stress_{SPAM} = \frac{Weight}{\pi a\, dx} \quad \rightarrow \quad Weight = \pi a\, dx\,(Max\,Stress_{SPAM})$

Model:

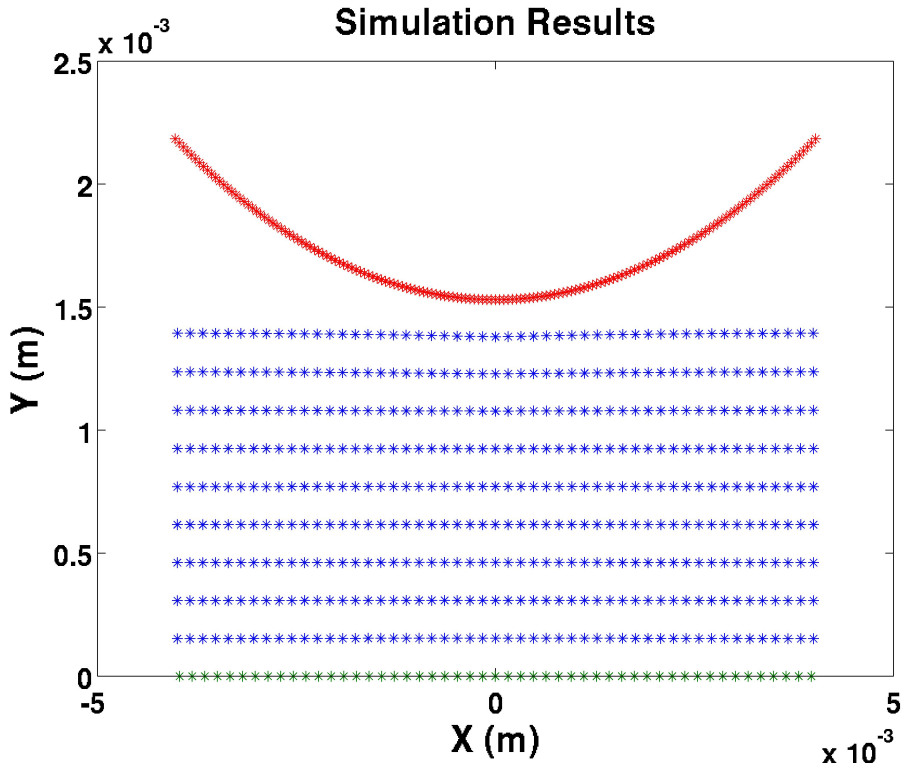
- Steel: Young's Modulus at 207 GPa, Poisson's Ratio at 0.3
- Color Code:
  - Red: Rigid Disk
  - Blue: elastic solid
  - Green = Fixed boundary
- 

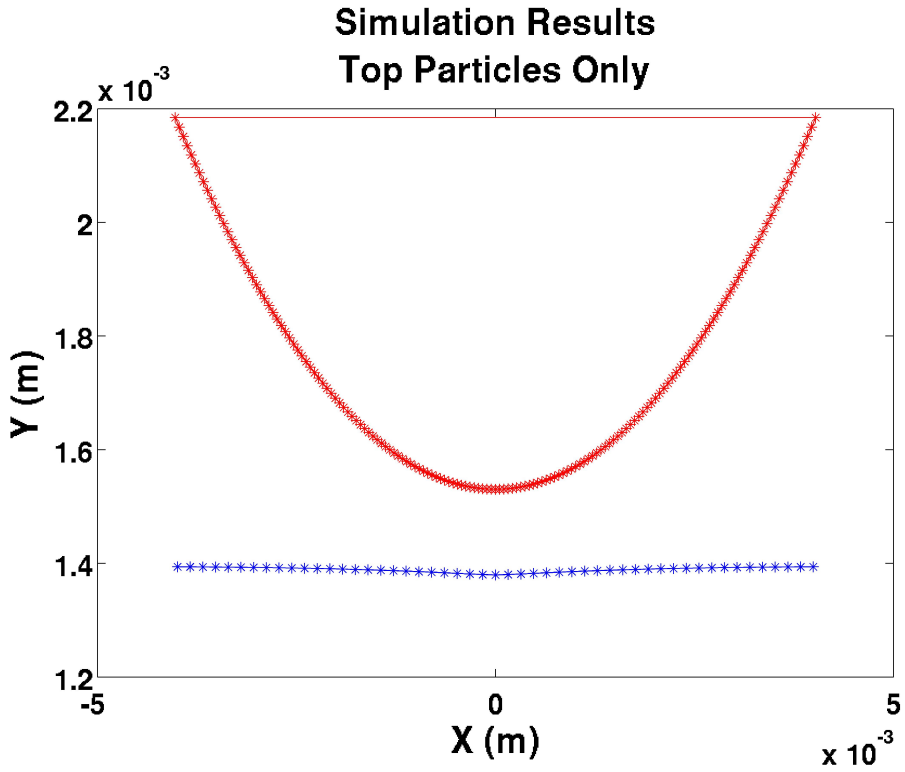
Results of Simulation:

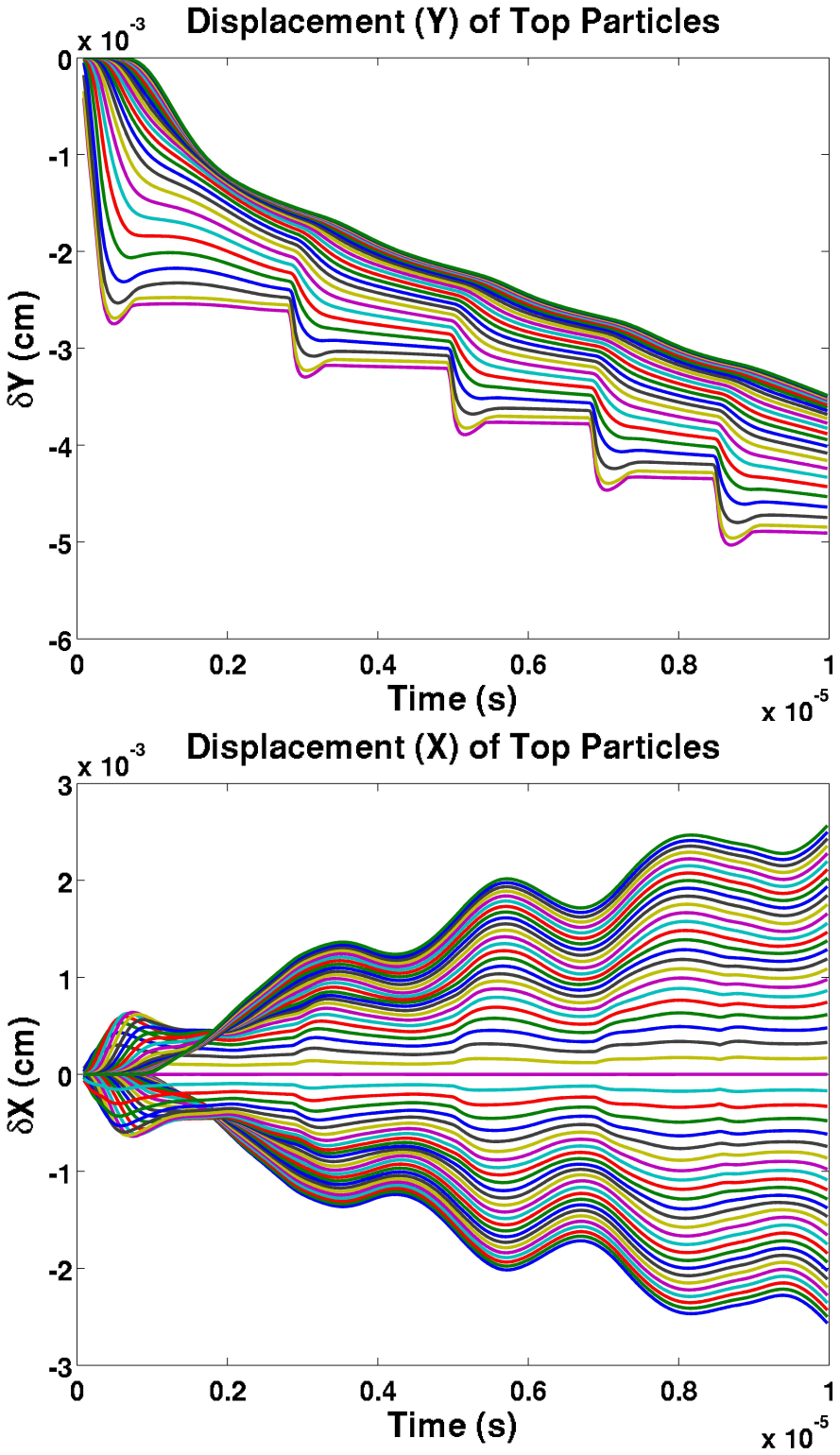
- Calculated Contact Area Half-Length (a) = 1.6501 meters
  - a / R = 8.78%
- Simulated Deflection: -0.0125 inch
- Stress (MPa) Tensor of Top Center Particle (Tij):

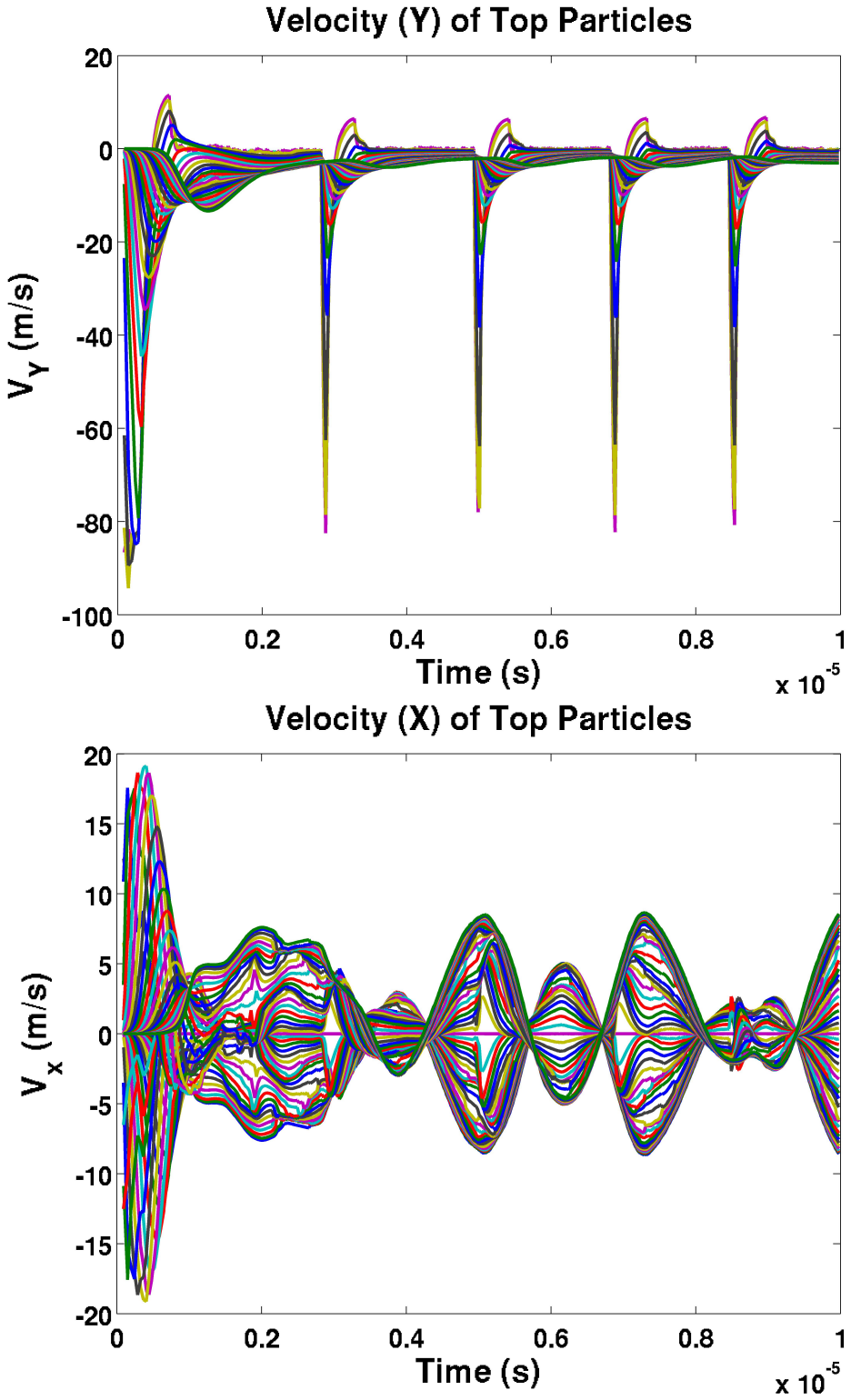
0	6.4147	0
6.4147	-4.0477	0
0	0	0

- Pressure – Error = 5.359%
  - Simulated (SPAM) Max Pressure, at Top Center Particle = 10.0525 GPa
  - Calculated (Hertz) Max Pressure, at Top Center Particle = 9.9896 GPa
- Weight – Error = 2.2053%
  - Simulated (SPAM) Max Pressure, at Top Center Particle = 5.9842 kN
  - Calculated (Hertz) Max Pressure, at Top Center Particle = 5.8552 kN


Simulation Results
x 10^-3
2.5
2
1.5
1
0.5
0
Y (m)
-5
0
5
X (m)
x 10^-3


Simulation Results
Top Particles Only
x 10^-3
2.2
2
1.8
1.6
1.4
1.2
Y (m)
-5
0
5
X (m)
x 10^-3


Displacement (Y) of Top Particles
x 10^-3
0
-1
-2
-3
-4
-5
-6
δY (cm)
0
0.2
0.4
0.6
0.8
1
Time (s)
x 10^-5
Displacement (X) of Top Particles
x 10^-3
3
2
1
0
-1
-2
-3
δX (cm)
0
0.2
0.4
0.6
0.8
1
Time (s)
x 10^-5


Velocity (Y) of Top Particles
20
0
-20
-40
-60
-80
-100
V_Y (m/s)
0
0.2
0.4
0.6
0.8
1
Time (s)
x 10^-5
Velocity (X) of Top Particles
20
15
10
5
0
-5
-10
-15
-20
V_x (m/s)
0
0.2
0.4
0.6
0.8
1
Time (s)
x 10^-5

R = 20 meters, L = 12.25 m, H = 2.5 m, dX = 0.25 m 51 * 10 Particles

Disk Deflection (m)	Contact Length (m)	Weight – SPAM (GN)	Weight – Hertz (GN)	Max T22 – SPAM (GPa)	Max T22 – Hertz (GPa)
0.0412	1.2828	6.5983	7.3621	7.3071	7.2952
0.0505	1.4206	8.9462	8.7679	7.8587	8.0784
0.0595	1.5420	11.1197	10.5022	8.6718	8.7690
0.0684	1.6532	13.0201	12.4286	9.5720	9.4015
0.0773	1.7566	14.8473	14.5197	10.5241	9.9897

Disk Deflection (m)	% Error (Weight)	% Error (Max T22)
0.0412	10.3750	0.1624
0.0505	2.0330	2.7202
0.0595	5.8792	1.1079
0.0684	4.7594	1.8128
0.0773	2.2565	5.3490

R = 30 meters, L = 18.375 m, H = 3.75 m, dX = 0.375 m 51 * 10 Particles

Disk Deflection (m)	Contact Length (m)	Weight – SPAM (GN)	Weight – Hertz (GN)	Max T22 – SPAM (GPa)	Max T22 – Hertz (GPa)
0.0618	1.9242	14.8477	16.5667	7.3080	7.2952
0.0758	2.1308	20.1331	19.7297	7.8595	8.0783
0.0893	2.3130	25.0249	23.6305	8.6721	8.7689
0.1027	2.4798	29.3003	27.9638	9.5721	9.4013
0.1159	2.6349	33.4118	32.6699	10.5244	9.9895

Disk Deflection (m)	% Error (Weight)	% Error (Max T22)
0.0618	10.3760	0.1758
0.0758	2.0448	2.7087
0.0893	5.9011	1.1036
0.1027	4.7793	1.8172
0.1159	2.2708	5.3543

R = 40 meters, L = 24.5 m, H = 5.0 m, dX = 0.50 m 51 * 10 Particles

Disk Deflection (m)	Contact Length (m)	Weight – SPAM (GN)	Weight – Hertz (GN)	Max T22 – SPAM (GPa)	Max T22 – Hertz (GPa)
0.0824	2.5657	26.3945	29.4497	7.3074	7.2952
0.1010	2.8411	35.7889	35.0722	7.8588	8.0784
0.1191	3.0840	44.4844	42.0097	8.6720	8.7689
0.1369	3.3063	52.0851	49.7116	9.5717	9.4013
0.1546	3.5132	59.3943	58.0780	10.5241	9.9896

Disk Deflection (m)	% Error (Weight)	% Error (Max T22)
0.0824	10.3741	0.1670
0.1010	2.0437	2.7179
0.1191	5.8907	1.1050
0.1369	4.7745	1.8127
0.1546	2.2665	5.3505

R = 50 meters, L = 30.625 m, H = 6.25 m, dX = 0.625 m 51 * 10 Particles

Disk Deflection (m)	Contact Length (m)	Weight – SPAM (GN)	Weight – Hertz (GN)	Max T22 – SPAM (GPa)	Max T22 – Hertz (GPa)
0.1030	3.2070	41.2442	46.0171	7.3078	7.2951
0.1263	3.5513	55.9236	54.8013	7.8590	8.0783
0.1488	3.8549	69.5117	65.6398	8.6720	8.7689
0.1711	4.1329	81.3883	77.6770	9.5720	9.4013
0.1932	4.3916	92.8097	90.7467	10.5240	9.9896

Disk Deflection (m)	% Error (Weight)	% Error (Max T22)
0.1030	10.3718	0.1730
0.1263	2.0480	2.7142
0.1488	5.8988	1.1050
0.1711	4.7779	1.8153
0.1932	2.2735	5.3501

R = 0.2 inches, L = 0.1275 inches, H = 0.025 inches, dX = 0.0025 inches 51 * 10 Particles

Disk Deflection (mm)	Contact Length (mm)	Weight – SPAM (N)	Weight – Hertz (N)	Max T22 – SPAM (GPa)	Max T22 – Hertz (GPa)
0.0105	0.3258	425.3947	475.1323	7.3094	7.2953
0.0128	0.3608	576.8719	565.7163	7.8591	8.0785
0.0151	0.3917	716.9652	677.5850	8.6720	8.7690
0.0174	0.4199	839.5070	801.8216	9.5718	9.4014
0.0196	0.4462	957.3682	936.7987	10.5245	9.9897

Disk Deflection (mm)	% Error (Weight)	% Error (Max T22)
0.0105	10.4682	0.1937
0.0128	1.9719	2.7160
0.0151	5.8118	1.1066
0.0174	4.7000	1.8124
0.0196	2.1957	5.3537

R = 0.3 inches, L = 0.1913 inches, H = 0.375 inches, dX = 0.0038 inches 51 * 10 Particles

Disk Deflection (mm)	Contact Length (mm)	Weight – SPAM (kN)	Weight – Hertz (kN)	Max T22 – SPAM (GPa)	Max T22 – Hertz (GPa)
0.0157	0.4888	0.9572	1.0689	7.3087	7.2952
0.0192	0.5412	1.2980	1.2728	7.8590	8.0785
0.0227	0.5875	1.6132	1.5246	8.6721	8.7691
0.0261	0.6299	1.8889	1.8042	9.5721	9.4014
0.0294	0.6693	2.1541	2.1079	10.5250	9.9898

Disk Deflection (mm)	% Error (Weight)	% Error (Max T22)
0.0157	10.4565	0.1853
0.0192	1.9760	2.7177
0.0227	5.8130	1.1058
0.0261	4.6973	1.8153
0.0294	2.1928	5.3577

R = 0.4 inches, L = 0.2550 inches, H = 0.5 inches, dX = 0.005 inches 51 * 10 Particles

Disk Deflection (mm)	Contact Length (mm)	Weight – SPAM (kN)	Weight – Hertz (kN)	Max T22 – SPAM (GPa)	Max T22 – Hertz (GPa)
0.0209	0.6517	1.7066	1.9001	7.3076	7.2955
0.0257	0.7216	2.3128	2.2632	7.8605	8.0784
0.0302	0.7833	2.8727	2.7115	8.6760	8.7687
0.0348	0.8398	3.3629	3.2089	9.5768	9.4012
0.0393	0.8923	3.8344	3.7489	10.5296	9.9894

Disk Deflection (mm)	% Error (Weight)	% Error (Max T22)
0.0209	10.1859	0.1663
0.0257	2.1896	2.6967
0.0302	5.9464	1.0570
0.0348	4.7990	1.8678
0.0393	2.2794	5.4081

R = 0.5 inches, L = 0.3188 inches, H = 0.625 inches, dX = 0.0063 inches 51 * 10 Particles

Disk Deflection (mm)	Contact Length (mm)	Weight – SPAM (kN)	Weight – Hertz (kN)	Max T22 – SPAM (GPa)	Max T22 – Hertz (GPa)
0.0262	0.8146	2.6590	2.9695	7.3092	7.2952
0.0321	0.9021	3.6060	3.5358	7.8593	8.0785
0.0378	0.9792	4.4818	4.2352	8.6726	8.7690
0.0435	1.0498	5.2477	5.0117	9.5723	9.4015
0.0491	1.1155	5.9843	5.8552	10.5250	9.9896

Disk Deflection (mm)	% Error (Weight)	% Error (Max T22)
0.0262	10.4552	0.1921
0.0321	1.9856	2.7127
0.0378	5.8221	1.1000
0.0435	4.7098	1.8169
0.0491	2.2053	5.3590

R = 0.2 inches, L = 0.25 inches, H = 0.025 inches, dX = 0.0025 inches 101 * 10 Particles

Disk Deflection (mm)	Contact Length (mm)	Weight – SPAM (kN)	Weight – Hertz (kN)	Max T22 – SPAM (GPa)	Max T22 – Hertz (GPa)
0.0102	0.3224	0.4040	0.5156	8.0164	7.2189
0.0124	0.3551	0.6047	0.6508	9.1878	7.9494
0.0146	0.3852	0.7897	0.7912	10.2975	8.6234
0.0168	0.4129	0.9484	0.9431	11.4494	9.2449
0.0190	0.4385	1.0611	1.1126	12.7192	9.8172

Disk Deflection (mm)	% Error (Weight)	% Error (Max T22)
0.0102	21.6538	11.0477
0.0124	7.0836	15.5788
0.0146	0.1923	19.4141
0.0168	0.5628	23.8462
0.0190	4.6256	29.5609

R = 0.3 inches, L = 0.375 inches, H = 0.0375 inches, dX = 0.00375 inches 101 * 10 Particles

Disk Deflection (mm)	Contact Length (mm)	Weight – SPAM (kN)	Weight – Hertz (kN)	Max T22 – SPAM (GPa)	Max T22 – Hertz (GPa)
0.0154	0.4836	0.9090	1.1601	8.0157	7.2188
0.0186	0.5326	1.3602	1.4642	9.1877	7.9493
0.0219	0.5777	1.7765	1.7801	10.2968	8.6233
0.0252	0.6194	2.1337	2.1220	11.4491	9.2449
0.0284	0.6577	2.3872	2.5033	12.7189	9.8172

Disk Deflection (mm)	% Error (Weight)	% Error (Max T22)
0.0154	21.6425	11.0386
0.0186	7.1023	15.5783
0.0219	0.2067	19.4061
0.0252	0.5482	23.8431
0.0284	4.6363	29.5575

R = 0.4 inches, L = 0.5 inches, H = 0.05 inches, dX = 0.005 inches 101 * 10 Particles

Disk Deflection (mm)	Contact Length (mm)	Weight – SPAM (kN)	Weight – Hertz (kN)	Max T22 – SPAM (GPa)	Max T22 – Hertz (GPa)
0.0205	0.6449	1.6161	2.0623	8.0156	7.2188
0.0248	0.7101	2.4182	2.6031	9.1875	7.9494
0.0292	0.7703	3.1581	3.1648	10.2971	8.6234
0.0336	0.8258	3.7930	3.7726	11.4494	9.2449
0.0379	0.8770	4.2438	4.4502	12.7185	9.8172

Disk Deflection (mm)	% Error (Weight)	% Error (Max T22)
0.0205	21.6366	11.0375
0.0248	7.0994	15.5758
0.0292	0.2121	19.4090
0.0336	0.5425	23.8448
0.0379	4.6367	29.5541

R = 0.5 inches, L = 0.625 inches, H = 0.0625 inches, dX = 0.00625 inches 101 * 10 Particles

Disk Deflection (mm)	Contact Length (mm)	Weight – SPAM (kN)	Weight – Hertz (kN)	Max T22 – SPAM (GPa)	Max T22 – Hertz (GPa)
0.0256	0.8061	2.5255	3.2234	8.0182	7.2187
0.0311	0.8877	3.8447	4.0651	9.1820	7.9499
0.0366	0.9629	4.9896	4.9451	10.2974	8.6234
0.0420	1.0322	5.9783	5.8977	11.4560	9.2443
0.0474	1.0961	6.6832	6.9584	12.7286	9.8165

Disk Deflection (mm)	% Error (Weight)	% Error (Max T22)
0.0256	21.6513	11.0749
0.0311	5.4199	15.4984
0.0366	0.8983	19.4129
0.0420	1.3672	23.9251
0.0474	3.9553	29.6661