Algorithm for generating irregular aggregate particles

```
; Filename: aggregate.dat
;
; Describe: Generate a aggregate.
;
; Written by Y.
;================================================================================
========
; Open log
set logfile aggregate.log
set log on overwrite
new
;set random 10000
; Setting size
def setSize
        aggregate_rad=14/2000.0 ;set aggregate size
end
setSize
;make temporary ball
ball id=1 x 0 y 0 z 0    rad aggregate_rad
;set some parameters
def setup
      cut_plane_num=8
      id_start=1
      xl=-0.0065
      yl=-0.0065
      zl=-0.0065
      n_xol=14
      n_yol=14
      n_zol=14
      rad_small=0.0005
end
setup
;find max ball id
def findMaxBId
      maxBId=0
      bp=ball_head
      loop while bp # null
            if b_id(bp)>maxBId then
                maxBId=b_id(bp)
            endif
            bp=b_next(bp)
      endloop
end
```

```
findMaxBId
; init some arrays
; Defining the required parameters, mainly used for saving ball information and Cut Plane information
def initArray
    ; ball params
    array bxx(maxBId)
    array byy(maxBId)
    array bzz(maxBId)
    array brad(maxBId)
    ;plane params
    array pxx(8)
    array pyy(8)
    array pzz(8)
    ;vector params
    array n1(8)
    array n2(8)
    array n3(8)
end
initArray
; get temporary ball info
; Getting the information of ball
def getTemp Ball Info
     bp=ball_head
    loop while bp # null
        _id=b_id(bp)
        bxx(_id)=b_x(bp)
        byy(_id)=b_y(bp)
        bzz(_id)=b_z(bp)
        brad(_id)=b_rad(bp)
        bp=b_next(bp)
    endloop
end
getTempBallInfo
; Delete temporary ball
delete ball    ;delete temporary ball
;=============================================================================
========
;make regulaer ball
; Generating regularly arranged balls within the range
def makeRegularBall
    xc = xl
    yc = yl
    zc = zl
    rc = rad_small
```

```
    idc = id_start
    loop zol(1,n_zol)
        loop yol(1,n_yol)
            loop xol(1,n_xol)
                command
                    ball id=idc x=xc y=yc z=zc rad=rc
                end_command
                idc = idc + 1
                xc=xc+2.0*rc        ;x
            end_loop
            yc=yc+2.0*rc        ;y
            xc=xl
        end_loop
        zc=zc+2.0*rc            ;z
        xc=xl
        yc=yl
    endloop
end
set echo off
makeRegularBall
set echo on
; Define a view for observing the model
def plotView
    command
        title 'Test by Y'
        plot create modelRegularBall
        plot add ball
        plot set background white
        plot set rotation 30 0 50
       ; plot add measure
        ; plot add axes black
        plot show
    endcommand
end
;getTotalBallCount
; Obtaining the total number of balls in the generation rule arrangement
def getTotalBallCount
     totalBallCount = int(max_bid)
     totalBallCount = float(totalBallCount)
end
getTotalBallCount
plotView
;pause key
plot close modelRegularBall             ; close view
; Setting all balls color parameters to 1
```

```
prop color 1 range id 1,totalBallCount   ; set color for grouping
;=============================================================================
========
; Used for setting the Cut Plane
def set Cut Plane8
	loop j(1, cut_plane_num)
		alpha = urand*pi/2.0 + (j-1.0)*pi/2.0
		beta = urand*pi/2.0 + (int((j-1.0)/4.0))*pi/2.0
		t1 = cos(alpha)
		t2 = sin(alpha)
		t3 = cos(beta)
		tt=sqrt(t1^2+t2^2+t3^2)
		n1(j)=t1/tt
		n2(j)=t2/tt
		n3(j)=t3/tt
		command
			print n1(j) n2(j) n3(j)
			;pause key
		endcommand
		pxx(j)=bxx(1) + brad(1)*(1-e0)*n1(j)
		pyy(j)=byy(1) + brad(1)*(1-e0)*n2(j)
		pzz(j)=bzz(1) + brad(1)*(1-e0)*n3(j)
	endloop
end
; Generate aggregate based on the cut plane
def generateAggregate
	e0 = 0.1; cut rate
	setCutPlane8
	bp = ball_head
	loop while bp # null
		_id=b_id(bp)
		_x=b_x(bp)
		_y=b_y(bp)
		_z=b_z(bp)
		_rad=b_rad(bp)
		if b_color(bp) = 1 then
			_p_1= _x *(n1(1)) + _y*(n2(1)) + _z*(n3(1)) - (pxx(1)*(n1(1)) + pyy(1)*(n2(1)) +
pzz(1)*(n3(1)))
			_p_2= _x *(n1(2)) + _y*(n2(2)) + _z*(n3(2)) - (pxx(2)*(n1(2)) + pyy(2)*(n2(2)) +
pzz(2)*(n3(2)))
			_p_3= _x *(n1(3)) + _y*(n2(3)) + _z*(n3(3)) - (pxx(3)*(n1(3)) + pyy(3)*(n2(3)) +
pzz(3)*(n3(3)))
			_p_4= _x *(n1(4)) + _y*(n2(4)) + _z*(n3(4)) - (pxx(4)*(n1(4)) + pyy(4)*(n2(4)) +
pzz(4)*(n3(4)))
			_p_5= _x *(n1(5)) + _y*(n2(5)) + _z*(n3(5)) - (pxx(5)*(n1(5)) + pyy(5)*(n2(5)) +
```

```
pzz(5)*(n3(5)))
                _p_6= _x *(n1(6)) + _y*(n2(6)) + _z*(n3(6)) - (pxx(6)*(n1(6)) + pyy(6)*(n2(6)) +
pzz(6)*(n3(6)))
                _p_7= _x *(n1(7)) + _y*(n2(7)) + _z*(n3(7)) - (pxx(7)*(n1(7)) + pyy(7)*(n2(7)) +
pzz(7)*(n3(7)))
                _p_8= _x *(n1(8)) + _y*(n2(8)) + _z*(n3(8)) - (pxx(8)*(n1(8)) + pyy(8)*(n2(8)) +
pzz(8)*(n3(8)))
                if _p_1 <= 0 then
                if _p_2 <= 0 then
                if _p_3 <= 0 then
                if _p_4 <= 0 then
                if _p_5 <= 0 then
                if _p_6 <= 0 then
                if _p_7 <= 0 then
                if _p_8 <= 0 then
                    if _x<=bxx(1)+brad(1) then
                    if _x>=bxx(1)-brad(1) then
                    if _y<=byy(1)+brad(1) then
                    if _y>=byy(1)-brad(1) then
                    if _z<=bzz(1)+brad(1) then
                    if _z>=bzz(1)-brad(1) then
                        command
                            group aggregate range id _id ; add ball into group aggregate
                        endcommand
                        b_color(bp) = 6    ;change color
                    end_if
                    end_if
                    end_if
                    end_if
                    end_if
                    end_if
                end_if
                end_if
                end_if
                end_if
                end_if
                end_if
                end_if
                end_if
            end_if
            bp=b_next(bp)
        endloop
end
generateAggregate
;=============================================================================
```

```
========
; Create asphalt group and delete it
;=========================================================================
========
; Setting the part with color 1 as the asphalt mortar section
group asphalt range color 1
; Delete the asphalt mortar section, then the aggregate can be obtained
delete ball range group asphalt    ; delete asphalt
plot show
save aggregate.sav
set log off
```

Generate 3D models and Torsional Shear Numerical Simulation Test

```
; Filename: make_asm.fis
;
; Describe: make_asm
;
; Written by Y.
;=========================================================================
========
; Initialization function，Defining model parameters，such as model size, etc.
def setup
    x_min   = 0.0
    x_max   = 0.0
    y_min   = 0.0
    y_max   = 0.0
    z_min   = 0.0
    z_max   = 0.0
    z_min0 =0.0
    z_max0 =0.0
    z_max1 =0.0
    wall1_s_stiff= 0.0
    wall1_n_stiff= 0.0
    wall2_s_stiff= 0.0
    wall2_n_stiff= 0.0
    rlo=0.0
    rhi=0.0
    dens_b_aggr=0.0
    psd_aggr=0.0
    id1=1
    poros= 0.0
    mult=0.0
    array ball_mult(20)
    array idd11(10)
    array idd22(10)
```

```
    array realVolFraction(10)
    array eachGradeCount(10)
end
setup
; ------------------------------------------------------------------
; Function filter and control the extent of the ball when it is assembled
def ff_cylinder
    ff_cylinder = 0
    _brad = fc_arg(0)
    _bx   = fc_arg(1)
    _by   = fc_arg(2)
    _bz   = fc_arg(3)
    _rad  = sqrt(_bx^2 + _by^2)
    if _rad + _brad > rl then
        ff_cylinder = 1
    end_if
end
;------------------------- generate the walls -------------------------
; Define functions that generate model walls
def make_wall
    extend = 0.5
    xl = x_max - x_min
    yl = y_max - y_min
    rl = 0.5 * xl
    zl = z_max1 - z_min
    v_total=pi*rl^2*(z_max-z_min)
    _xmin = x_min*(1+extend)
    _xmax = x_max*(1+extend)
    _ymin = y_min*(1+extend)
    _ymax = y_max*(1+extend)
    command
        wall type cylinder id=1 ks=wall1_s_stiff kn=wall1_n_stiff &
            end1 0.0 0.0 z_min0 end2 0.0 0.0 z_max0 rad rl rl
        wall id=5 ks=wall2_s_stiff kn=wall2_n_stiff face (_xmin,_ymin z_min) &
            (_xmax _ymin z_min) (_xmax _ymax z_min) (_xmin _ymax z_min)
        wall id=6 ks=wall2_s_stiff kn=wall2_n_stiff face (_xmin _ymax z_max1) &
            (_xmax _ymax z_max1) (_xmax _ymin z_max1) (_xmin _ymin z_max1)
    end_command
    command
        plot_view
    end_command
end
;------------------------- generate the particles I ----------------------
; Define the function for arranging the aggregate, according to the grading of each particle size, the start and end ID of each file record is recorded, which will be used later.
```

```
def assembly
    dens_b=dens_b_aggr
    group_vol_ini=(1-poros)*v_total*psd_aggr
    rar=(rlo+(rhi - rlo)*0.5)
    num=int(3.0*group_vol_ini/(4.0*pi*rar^3))
    id2=id1+num-1
    r_min=(rlo+(rhi - rlo)*0.5)/mult
    r_max=(rlo+(rhi - rlo)*0.9)/mult
    dens_b0=dens_b/(1-poros)
    count=0
    if num#0
        ball_group='ball_group'+string(group_num)
        command
            print ball_group
            ;pause key
        endcommand
        command
            gen no_shad id=id1,id2 x=x_min,x_max y=y_min,y_max z=z_min,z_max1 &
                    rad=r_min,r_max filter ff_cylinder tries 100000000000000
            prop dens=dens_b kn=n_stiff ks=s_stiff fric=fric_num range id=id1,id2
            group ball_group range id=id1,id2
        end_command
        group_vol=0.0
        bp=find_ball(id2)
        bp0=find_ball(id1)
        loop while bp#bp0
            b_color(bp)=group_num-1
            group_vol=group_vol+4.0/3.0*pi*b_rad(bp)^3
            bp=b_next(bp)
        end_loop
        b_color(bp0)=group_num-1
        group_vol=group_vol+4.0/3.0*pi*b_rad(bp0)^3
        _mult=(group_vol_ini/group_vol)^(1.0/3)
        ball_mult(group_num)=_mult
        idd11(group_num)=id1; Recording the start ID
        idd22(group_num)=id2; Recording the end ID
        eachGradeCount(group_num) = num ;
        command
            print _mult idd11(group_num) idd22(group_num)
            ;pause
        end_command
        id1=id2+1
    end_if
end
;make_extend
```

```
; Define radius expansion function, when generating a ball, in order to prevent the ball from exceeding the outside of the wall, a radius expansion method wall used to generate
; That was, when the generation starts, the particles with smaller radii were used, and after all the generation was completed, the radius was enlarged to expand to the actual required size
def make_extend
    loop group_i(1,group_tot)
        _mult = ball_mult(group_i)
        ball_mult(group_i)=_mult^(1.0/10)
        _mult = ball_mult(group_i)
        command
            print _mult
            ;pause key
        end_command
    end_loop
    mult=1.0
    loop _i(1,10)
        loop group_i(1,group_tot)
            mult = ball_mult(group_i)
            ball_group='ball_group' + string(group_i)
            command
                print mult
                ini rad mul=mult range group ball_group
                ;pause key
            end_command
        end_loop
        command
            cycle 100
        end_command
    end_loop
end
return
;=============================================================================
=================================================
; Filename: AC13.dat
;
; Describe: Make AC13 MODEL
;
; Written by Y.
;=============================================================================
=================================================
; Generate model
new
set random 10001    ; the random can change for diffrent model, such as 10001 10060 ...
;set pinterval 1000
; transfer make_asm.fis
```

```
call make_asm.fis
; set gradation information
; Define gradation information, mainly record the volume fraction and particle size
def setGradationInfo
	array volFraction(10)
		volFraction(1) = 0.021		;Calculated based on Gradation
		volFraction(2) = 0.161
		volFraction(3) = 0.250
		volFraction(4) = 0.136
	array sievePoreSize(10)
		sievePoreSize(1) = 16.0/2000.0 ; 'mm' --> 'm' && diameter --> radius
		sievePoreSize(2) = 13.2/2000.0
		sievePoreSize(3) = 9.50/2000.0
		sievePoreSize(4) = 4.75/2000.0
		sievePoreSize(5) = 2.36/2000.0
end
setGradationInfo
;=============================================================================
================================================
;MAKE ASSEMBLY
; Define a view for viewing the model
def plot_view
	command
		title 'Torsion Test by Y'
		; general view
		plot create GeneralView
		plot set title text 'Gradation Model'
		plot set caption size 20
		plot set mag 1.56
		plot set background white
		plot set rotation 20.0 0.0 50.0
		;plot set center 0.0 0.0 0.0
		;plot set mag 1.5
		;plot set dist 1
		plot add wall wireframe on
		plot add ball lblue lgreen yellow green orange cyan lmagenta &
				red magenta brown lgray lcyan lred lorange blue white
		;plot add disp black
		;plot measure
		plot show
	end_command
end
;=============================================================================
================================================
; Define additional views for viewing other relevant information about the model, such as wall forces,
```

```
etc.
def plot_view1
    command
        plot create Wall_History
        plot set title text 'Wall force id 1 2 3'
        plot set caption size 20
        plot set background white
        plot set mag 4
        plot set dist 1
        plot add hist 1 -2 3
        plot show
        plot create MeanUbal_History
        plot set title text 'Diagnostic muf'
        plot set caption size 20
        plot set background white
        plot set mag 4
        plot set dist 1
        plot add hist 4
        plot show
        plot create Poros_History
        plot set title text 'Hist Porosity'
        plot set caption size 20
        plot set background white
        plot set mag 4
        plot set dist 1
        plot add hist 5
        plot show
    end_command
end
;=============================================================================
==================================================
; Generate each records
def particle_make
    array id_record(10)
    i=1
    loop while i<=group_tot
        group_num=i
        particle_data ; obtain corresponding volume fraction and size information according to
each aggregate
        assembly    ; call the function to serve for the arrangement of aggregate
        id_record(group_num)=id2
        command
            print id1 id2 num id_record(group_num)
            ;pause key
        end_command
```

```
        i=i+1
    end_loop
end
;---------------------------------------------------------------
; Assignment values for model size parameters
set x_min=-0.05 x_max=0.05 y_min=-0.05 y_max=0.05 z_min=0.0 z_max=0.10 ;model size
set z_min0=-0.03 z_max0=0.13    ;cylinder wall height
set z_max1=0.10                 ;top wall z-coordinate
set poros=0.0 mult=2.0          ;poros
set wall1_n_stiff=1.0e6   wall1_s_stiff=0.25e6   ;wall stiff n s
set wall2_n_stiff=10.0e6 wall2_s_stiff=2.5e6
make_wall   ;create wall
set group_tot=4 ; total group ; there are 4 sieves in total
; Obtain size information for different particle sizes
def particle_data
    case_of i
    case 1
        rlo=sievePoreSize(2)
        rhi=sievePoreSize(1)
        psd_aggr= volFraction(1)
        dens_b_aggr=2500.0
        n_stiff=10.0e6
        s_stiff=2.5e6
        fric_num=0.01
    case 2
        rlo=sievePoreSize(3)
        rhi=sievePoreSize(2)
        psd_aggr=volFraction(2)
        dens_b_aggr=2500.0
        n_stiff=10.0e6
        s_stiff=2.5e6
        fric_num=0.01
    case 3
        rlo=sievePoreSize(4)
        rhi=sievePoreSize(3)
        psd_aggr=volFraction(3)
        dens_b_aggr=2500.0
        n_stiff=10.0e6
        s_stiff=2.5e6
        fric_num=0.01
    case 4
        rlo=sievePoreSize(5)
        rhi=sievePoreSize(4)
        psd_aggr=volFraction(4)
        dens_b_aggr=2500.0
```

```
            n_stiff=10.0e6
            s_stiff=2.5e6
            fric_num=0.01
        end_case
end
; Generate model
particle_make
set dt dscale
;set damp local 0.0
;set grav 0 0 0
prop fric 0.0
wall prop fric 0.0
measure id=1 x=0.0 y=0.0 z=0.065 rad=0.050      ; set measure
hist id 1 wall xforce id 1                      ; Monitor x-force on wall 1
hist id 2 wall zforce id 5                      ; Monitor z-force on top wall
hist id 3 wall zforce id 6                      ; Monitor z-force on bottom wall
hist id 4 diagnostic muf                        ; Monitor diagnostic
hist id 5 measure porosity id 1                 ; Monitor porosity
plot_view1
make_extend
cyc 1000
save ac13.sav
; Test gradation, obtain the number of aggregates
def checkGradation
        loop group_i(1,group_tot)
                sum = 0.0
                id001 = idd11(group_i)
                id002 = idd22(group_i)
                bp1 = find_ball(id001)
                bp = find_ball(id002)
                loop while bp # bp1
                        sum = sum + (4.0/3.0) * pi * b_rad(bp)^3
                        bp    = b_next(bp)
                end_loop
                sum = sum + (4.0/3.0) * pi * b_rad(bp1)^3
                _v_f = sum/v_total
                realVolFraction(group_i) = _v_f
                command
                        print realVolFraction(group_i) eachGradeCount(group_i)
                        pause 3
                endcommand
        end_loop
end
checkGradation
return
```

```
; Filename: regularBall_125.dat
;
; Describe: Make regularBall_125 Model
;
; Written by Y.
;================================================================================
================================================
; Generate regularly arranged balls within the scope of the model
new
restore ac13.sav
; Define individual parameters
def setup
	cut_plan_num=8
	id_start=1
	xl=-0.05
	yl=-0.05
	Zl=0.00125
	n_xol=40
	n_yol=40
	n_zol=40
	rsmall=0.00125		;In theory, the smaller the radius, the more accurate, but considering the computer performance, the value here is 0.00125.
end
setup
set cut_plan_num=8
; Find max particle Id
def getMaxBallId
	maxBallId=0
	bp=ball_head
	loop while bp # null
		if b_id(bp) > maxBallId then
			maxBallId=b_id(bp)
		endif
		bp=b_next(bp)
	endloop
end
getMaxBallId
print maxBallId
;pause key
; Initialize each array, similar to the irregular aggregate generation algorithm
def initArray
	;ball params
	array bxx(maxBallId)
	array byy(maxBallId)
	array bzz(maxBallId)
```

```
    array brad(maxBallId)
    ;plane params
    array pxx(maxBallId,8)
    array pyy(maxBallId,8)
    array pzz(maxBallId,8)
    array prad(maxBallId,8)
    ; vectors
    array n1(maxBallId,8)
    array n2(maxBallId,8)
    array n3(maxBallId,8)
    array pp(maxBallId,8)
    array xpa(maxBallId,8)
    ;eachGradeSmallBallTotalCount
    array eachGradeSmallBallTotalCount(10)
end
initArray
; Obtain the generated model ball parameters, mainly is coordinates and radius, etc.
def getTemporaryBallInfo
    bp=ball_head
    loop while bp # null
        idd=b_id(bp)
        bxx(idd)=b_x(bp)
        byy(idd)=b_y(bp)
        bzz(idd)=b_z(bp)
        brad(idd)=b_rad(bp)
        bp=b_next(bp)
    endloop
end
getTemporaryBallInfo
;pause key
delete ball; delete these particles after saving the information
;=============================================================================
=============================================================================
=======================
; Generate a cylindrical model of regular arrangement
def makeRegularCylinder
    xc = xl
    yc = yl
    zc = zl
    rc = rsmall
    idc = id_start
    loop zol(1,n_zol)
        loop yol(1,n_yol)
            loop xol(1,n_xol)
                radcc    = (xc^2 + yc^2)^(0.5)
```

```
                    radcc01 = radcc + rsmall*1.0
                    if    radcc01 <=0.05 then
                          command
                                ball id=idc x=xc y=yc z=zc rad=rc
                          end_command
                          idc = idc + 1
                    endif
                    xc=xc+2.0*rc
               end_loop
               yc=yc+2.0*rc
               xc=xl
          end_loop
          zc=zc+2.0*rc
          xc=xl
          yc=yl
     end_loop
end
;================================== Make regularBall_125
==============================================
set echo off
makeRegularCylinder
set echo on
; Obtain the total number of particles in regular arrangement
def getTotalBallCount
        totalBallCount = int(max_bid)
        totalBallCount = float(totalBallCount)
end
getTotalBallCount
save regularBall_125.sav
; Filename: modelIrregular.dat
;
; Describe: create irregular model.
;
; Written by Y.
; Loading irregular generation algorithm to generate irregular aggregate cylinder model
;=============================================================================
=================================================
new
set logfile modelIrregular.log
set log on overwrite
restore regularBall_125.sav
; Set the color to 1
prop color 1 range id 1,totalBallCount ; set all balls color for 1
;=============================================================================
=================================================
```

```
;def setCutPlane
; Define the cutting plane of the cutting algorithm
def setCutPlane
    loop k(idd1,idd2)
        loop kk(1,cut_plan_num)
            alpha = urand*pi/2.0 + (kk-1.0)*pi/2.0
            beta = urand*pi/2.0 + (int((kk-1.0)/4.0))*pi/2.0
            t1 = cos(alpha)
            t2 = sin(alpha)
            t3 = cos(beta)
            tt=sqrt(t1^2+t2^2+t3^2)
            n1(k,kk)=t1/tt
            n2(k,kk)=t2/tt
            n3(k,kk)=t3/tt
            pxx(k,kk)=bxx(k) + brad(k)*(1-e0)*n1(k,kk)          ;   e0 is cut_rate
            pyy(k,kk)=byy(k) + brad(k)*(1-e0)*n2(k,kk)
            pzz(k,kk)=bzz(k) + brad(k)*(1-e0)*n3(k,kk)
        endloop
    endloop
end
;=============================================================================
================================================
;def createAggregate
;Generate aggregate
def createAggregate
    array eachAggregateSmallBallNum(maxBallId)
    array eachAggregateGradeSmallBallZero(4,maxBallId)
    array eachGradeSmallBallZeroNum(4)
    loop j(1,4)                         ;The 4 grades
        e0 = 0.15                       ;cut_rate
        idd1=idd11(j)                   ;begin_id
        idd2=idd22(j)                   ;end_id
        cyc_ctrl = 1                    ;cycle_ctroller
        loop while cyc_ctrl # 0
            setCutPlane                 ;set cut plane base on e0
            eachGradeSmallBallTotalCount(j)=0   ;init to 0
            eachGradeSmallBallZeroNum(j)=0

            currentIndex = 0            ;Counter
            loop ii(idd1,idd2)          ;Loading Irregular Aggregate Generation Algorithms
Begin
                aggregate='aggregate' + string(ii)
                eachAggregateSmallBallNum(ii) = 0           ;init to 0
                eachAggregateGradeSmallBallZero(j,ii) = 0
                command
```

```
            print aggregate eachAggregateSmallBallNum(ii)
            ;pause key
            set echo off
        endcommand
        bp = ball_head
        loop while bp # null
            _id    =b_id(bp)
            _x     =b_x(bp)
            _y     =b_y(bp)
            _z     =b_z(bp)
            _rad =b_rad(bp)
            if b_color(bp) = 1 then
                _p_1   =   _x   *(n1(ii,1))   +   _y*(n2(ii,1))   +   _z*(n3(ii,1))   -
(pxx(ii,1)*(n1(ii,1)) + pyy(ii,1)*(n2(ii,1)) + pzz(ii,1)*(n3(ii,1)))
                _p_2   =   _x   *(n1(ii,2))   +   _y*(n2(ii,2))   +   _z*(n3(ii,2))   -
(pxx(ii,2)*(n1(ii,2)) + pyy(ii,2)*(n2(ii,2)) + pzz(ii,2)*(n3(ii,2)))
                _p_3   =   _x   *(n1(ii,3))   +   _y*(n2(ii,3))   +   _z*(n3(ii,3))   -
(pxx(ii,3)*(n1(ii,3)) + pyy(ii,3)*(n2(ii,3)) + pzz(ii,3)*(n3(ii,3)))
                _p_4   =   _x   *(n1(ii,4))   +   _y*(n2(ii,4))   +   _z*(n3(ii,4))   -
(pxx(ii,4)*(n1(ii,4)) + pyy(ii,4)*(n2(ii,4)) + pzz(ii,4)*(n3(ii,4)))
                _p_5   =   _x   *(n1(ii,5))   +   _y*(n2(ii,5))   +   _z*(n3(ii,5))   -
(pxx(ii,5)*(n1(ii,5)) + pyy(ii,5)*(n2(ii,5)) + pzz(ii,5)*(n3(ii,5)))
                _p_6   =   _x   *(n1(ii,6))   +   _y*(n2(ii,6))   +   _z*(n3(ii,6))   -
(pxx(ii,6)*(n1(ii,6)) + pyy(ii,6)*(n2(ii,6)) + pzz(ii,6)*(n3(ii,6)))
                _p_7   =   _x   *(n1(ii,7))   +   _y*(n2(ii,7))   +   _z*(n3(ii,7))   -
(pxx(ii,7)*(n1(ii,7)) + pyy(ii,7)*(n2(ii,7)) + pzz(ii,7)*(n3(ii,7)))
                _p_8   =   _x   *(n1(ii,8))   +   _y*(n2(ii,8))   +   _z*(n3(ii,8))   -
(pxx(ii,8)*(n1(ii,8)) + pyy(ii,8)*(n2(ii,8)) + pzz(ii,8)*(n3(ii,8)))
                if _p_1 <= 0 then
                if _p_2 <= 0 then
                if _p_3 <= 0 then
                if _p_4 <= 0 then
                if _p_5 <= 0 then
                if _p_6 <= 0 then
                if _p_7 <= 0 then
                if _p_8 <= 0 then
                    if _x <=bxx(ii)+brad(ii) then
                    if _x >=bxx(ii)-brad(ii) then
                    if _y <=byy(ii)+brad(ii) then
                    if _y >=byy(ii)-brad(ii) then
                    if _z <=bzz(ii)+brad(ii) then
                    if _z >=bzz(ii)-brad(ii) then
                        command
                            group aggregate range id _id          ;add the small ball
into aggregate group
```

```
                                    endcommand
                                    b_color(bp) = 6
                                    eachAggregateSmallBallNum(ii)                    =
eachAggregateSmallBallNum(ii) +1                                ;small ball add 1
                                    eachAggregateGradeSmallBallZero(j,ii)            =
eachAggregateGradeSmallBallZero(j,ii) +1            ;small ball add 1
                                    eachGradeSmallBallTotalCount(j)                  =
eachGradeSmallBallTotalCount(j) + 1                         ;add 1
                                else
                                    eachGradeSmallBallTotalCount(j)                  =
eachGradeSmallBallTotalCount(j)                             ;unchange
                                end_if
                                end_if
                                end_if
                                end_if
                                end_if
                                end_if
                            end_if
                            end_if
                            end_if
                            end_if
                            end_if
                            end_if
                            end_if
                            end_if
                        end_if
                        bp=b_next(bp)
                    endloop                     ;Loading Irregular Aggregate Generation
Algorithms End
                    command
                        set echo on
                    endcommand

                    if eachAggregateSmallBallNum(ii) = 0 then
                        eachGradeSmallBallZeroNum(j)= eachGradeSmallBallZeroNum(j) + 1
                    endif
                    currentIndex = currentIndex + 1
                    command
                        print           currentIndex            eachAggregateSmallBallNum(ii)
eachGradeSmallBallZeroNum(j)
                    endcommand
                endloop
                gradeSmallBallRate = eachGradeSmallBallTotalCount(j)/totalBallCount         ;The
rate of each grade aggregate
                command
```

```
                print e0 gradeSmallBallRate eachGradeSmallBallTotalCount(j) totalBallCount
                ;pause key
            endcommand
; Determine whether the particle gradation is within a certain range: for example, 95%~105%, otherwise regenerate the aggregate
            if eachGradeSmallBallTotalCount(j)/totalBallCount < volFraction(j)*(1-0.05) then ;Lower than 95%
                e0 = e0 - 0.01                                                  ;Reduce e0
                loop iii(idd1,idd2)
                if eachAggregateSmallBallNum(iii) # 0 then
                    aggregate='aggregate' + string(iii)
                    command
                        prop color=1 range group aggregate                      ;restore to asphalt with color 1
                        group delete aggregate                                  ;delete the group of aggregate
                    endcommand
                endif
                endloop
            else
                if eachGradeSmallBallTotalCount(j)/totalBallCount < volFraction(j)*(1+0.05) then
                    cyc_ctrl = 0                                                ; cyc_ctrl ==0 then Exit this aggregate making
                else                                                            ;greater than 105%
                    e0 = e0 + 0.01                                              ;Increase e0
                    loop iii(idd1,idd2)
                        if eachAggregateSmallBallNum(iii) # 0 then
                            aggregate='aggregate' + string(iii)
                            command
                                prop color=1 range group aggregate
    ;restore to asphalt with color 1
                                group delete aggregate
    ;delete the group of aggregate
                            endcommand
                        endif
                    endloop
                endif

        endloop
    endloop
end
```

```
createAggregate
;=============================================================================
================================================
;create asphalt group
;set color 1 to asphalt
group asphalt range color 1; Set the color to 1 for the asphalt mortar section
;=============================================================================
================================================
;plot model view
; Define a view for viewing the model
def groupModelPlotView
        command
                plot create groupModel
                plot add group
                plot add axes black
                plot set background white
                plot show
        endcommand
end
groupModelPlotView
;=============================================================================
================================================
set log off
save modelIrregular.sav
;end
; Filename: torsionTest_60.dat
;
; Describe: Torsion Test
;
; Written by Y.
; Torsion test
;=============================================================================
================================================
new
restore modelIrregular.sav
set echo off
; Crk.fis and fishcall.fis are third-party libraries, mainly used to record information such as cracks.
call crk.fis                    ; load crk.fis and fishcall.fis
call fishcall.fis
set echo on
set memory size 20        ; set memory and pinterval
set pinterval 100
delete wall 1                   ; delete all walls
delete wall 5 to 6
;=============================================================================
```

```
=================================================
;delete escaping particles
; define removal function
def remove_balls
    ;while_stepping
    rad_del_count = rad_del_count + 1
    if rad_del_count > 100
        rad_del_count = 0
        bp = ball_head
        loop while bp # null
            next = b_next(bp)
            if b_z(bp)<0.10*(1+0.0) then
            if b_z(bp)>0.0    then
                rad_del = ((abs(b_x(bp)))^2.0+(abs(b_y(bp)))^2.0)^0.5+rsmall
                if rad_del > 0.05*(1+0.005) then
                    ii = b_delete(bp)
                end_if
            end_if
            end_if
            bp = next
        end_loop
    end_if
end
;=================================================================================
=================================================
;create poros
; Setting the air voids, such as 0.04, and delete the corresponding amount of asphalt mortar particles
as the air voids.
def delPoros
     ;totalBallCount = 1000
     poros = ***                                ;
    ndel = int( poros * totalBallCount )          ; delete count
     aa = 1
     loop while aa <= ndel
            bb = int((totalBallCount)*urand + 1)
            bp = find_ball(bb)
            if bp # null then
            if b_color(bp)=1 then
                ii = b_delete(bp)
                aa = aa +1
             endif
             endif
     endloop
end
delPoros
```

```
;=============================================================================
=================================================
; print ndel totalBallCount poros
print ndel totalBallCount poros
pause 2
;=============================================================================
=================================================
;def makeDensityDistinguish
;Temporary setting density
; Density discrimination algorithm, mainly used to set related parameters later
; That is, the density parameter of the mortar is first set to 1, and the density parameters of each
aggregate particle are set to 2, 3, 4, ... in turn, correspondingly 1
def makeDensityDistinguish                ;Density discrimination method for every aggregate
        loop _n(1,maxBallId)
                if eachAggregateSmallBallNum(_n) # 0 then
                    aggregate = 'aggregate' + string(_n)
                    aa_dens = _n+1                  ; 2 3 4 5 6 ... the asphalt is 1
                    command
                        prop density =aa_dens range group aggregate
                    end_command
                endif
        end_loop
end
set echo off
makeDensityDistinguish
set echo on
; set the color to 1 for the asphalt mortar
group asphalt range color 1               ;create asphalt
;pause key
; set the density parameter of the asphalt mortar to 1
prop density 1 range group asphalt ;temporary setting asphalt density to 1
;=============================================================================
=========================================================
; load load burwrv.dll
;Loading burwrv.dll, the Burger's model
config cppudm
model load burwrv.dll ; 3d VC++ Release version
;=============================================================================
=========================================================
;makeParams
; define parameters
; That is:
def makeParams
    between_n_bond = asphalt_n_bond*1.0
    between_s_bond = asphalt_s_bond*1.0
```

```
cp = contact_head
loop while cp # null
    bp2 = c_ball2(cp)
    if pointer_type(bp2) # 101 then
        bp1 = c_ball1(cp)
        _density_1=b_dens(bp1)
        _density_2=b_dens(bp2)
        if _density_1 = _density_2 then
            ; Equal density and is1 asphalt mortar interior
            if _density_1 = 1 then                                  ;asphalt interior
                c_model(cp) = 'burger'
                c_prop(cp,'bur_cnm') = cnm_set
                c_prop(cp,'bur_knm') = knm_set
                c_prop(cp,'bur_cnk') = cnk_set
                c_prop(cp,'bur_knk') = knk_set
                c_prop(cp,'bur_csm') = csm_set
                c_prop(cp,'bur_ksm') = ksm_set
                c_prop(cp,'bur_csk') = csk_set
                c_prop(cp,'bur_ksk') = ksk_set
                c_prop(cp,'bur_fric') = fric_set
                b_fric(bp1)=asphalt_fric
                b_fric(bp2)=asphalt_fric
                c_nstrength(cp)=asphalt_n_bond
                c_sstrength(cp)=asphalt_s_bond
            else                                                    ;aggregate internal
                ; aggregate interior
                b_fric(bp1)=aggregate_fric
                b_fric(bp2)=aggregate_fric
                c_nstrength(cp)=aggregate_n_bond
                c_sstrength(cp)=aggregate_s_bond
            endif
        else
            if _density_1 = 1 then                                  ;between asphalt
and aggregate
                ; between aggregate and mortar
                c_model(cp) = 'burger'
                c_prop(cp,'bur_cnm') = cnm_set
                c_prop(cp,'bur_knm') = knm_set
                c_prop(cp,'bur_cnk') = cnk_set
                c_prop(cp,'bur_knk') = knk_set
                c_prop(cp,'bur_csm') = csm_set
                c_prop(cp,'bur_ksm') = ksm_set
                c_prop(cp,'bur_csk') = csk_set
                c_prop(cp,'bur_ksk') = ksk_set
                c_prop(cp,'bur_fric') = fric_set
```

```
                        b_fric(bp1)=asphalt_fric
                        b_fric(bp2)=aggregate_fric
                        c_nstrength(cp)=between_n_bond
                        c_sstrength(cp)=between_s_bond
                    else
                        if _density_2 = 1 then                                  ;between
asphalt and aggregate
                            ; between aggregate and mortar
                            c_model(cp) = 'burger'
                            c_prop(cp,'bur_cnm') = cnm_set
                            c_prop(cp,'bur_knm') = knm_set
                            c_prop(cp,'bur_cnk') = cnk_set
                            c_prop(cp,'bur_knk') = knk_set
                            c_prop(cp,'bur_csm') = csm_set
                            c_prop(cp,'bur_ksm') = ksm_set
                            c_prop(cp,'bur_csk') = csk_set
                            c_prop(cp,'bur_ksk') = ksk_set
                            c_prop(cp,'bur_fric') = fric_set
                            b_fric(bp1)=aggregate_fric
                            b_fric(bp2)=asphalt_fric
                            c_nstrength(cp)=between_n_bond
                            c_sstrength(cp)=between_s_bond
                        else                                                    ;Between  different
aggregates
                            b_fric(bp1)=aggregate_fric
                            b_fric(bp2)=aggregate_fric
                            c_nstrength(cp)=between_n_bond
                            c_sstrength(cp)=between_s_bond
                        endif
                    endif
                endif
            endif
            cp = c_next(cp)
        endloop
end
;=========================================================================
=========================================================
;Setting related parameters
; set aggregate params
set aggregate_n_bond=*** aggregate_s_bond=***
set aggregate_fric = ***
;=========================================================================
=========================================================
; set asphalt params
set asphalt_n_bond=***    asphalt_s_bond=***    ;
```

```
set asphalt_fric = ***
;=============================================================================
============================================================
; set Burger's params
set    cnm_set=***    knm_set=***    cnk_set=***    knk_set=***
set    csm_set=***    ksm_set=***    csk_set=***    ksk_set=***
set    fric_set=***
;=============================================================================
============================================================
;between
;set between_n_bond=*** between_s_bond=***
makeParams
;=============================================================================
============================================================
;set asphalt density kn ks
prop density=***    range group asphalt
prop kn=*** ks=*** range group asphalt
prop fric=*** range group asphalt
;pause key
;=============================================================================
============================================================
;set aggregate density kn ks
prop density=*** range group asphalt not            ;
prop kn=***    ks=*** range group asphalt    not
prop fric=*** range group asphalt not
;pause key
;=============================================================================
============================================================
;create createLoadingPlate
; Setting the loading plate, that is, set a layer of regularly arranged particles according to the size corresponding to the model at the top and bottom of the model, and set it to clump.
;Generate regular arranging particles as loaded plate
def createLoadingPlate
      xc = xl
      yc = yl
      zc = zl
      rc = rsmall
      idc = id_start
      loop zol(1,n_zol)
            loop yol(1,n_yol)
                  loop xol(1,n_xol)
                        radcc    = (xc^2 + yc^2)^(0.5)
                        radcc01 = radcc + rsmall*1.0
                        if    radcc01 <=0.05 then
                              command
```

```
                        ball id=idc x=xc y=yc Z=ZC rad=rc
                    end_command
                idc = idc + 1
                endif
                xc=xc+2.0*rc
            end_loop
            yc=yc+2.0*rc
            xc=xl
        end_loop
        zc=zc+2.0*rc
        xc=xl
        yc=yl
    end_loop
end
;=============================================================================
============================================================
;set top loading plate size
; Define the top loader size, start the ID, and set a different starting ID, here for subsequent search for the corresponding loaded plate.
set id_start=100001 xl=-0.055 yl=-0.055 zl=0.10125 n_xol=50 n_yol=50 n_zol=1
set rsmall=0.00125
set echo off
createLoadingPlate
set echo on
;=============================================================================
============================================================
;set bottom loading plate size
; Define the top loader size, start the ID, and set a different starting ID, here for subsequent search for the corresponding loaded plate.
set id_start=120001 xl=-0.055 yl=-0.055 zl=-0.00125 n_xol=50 n_yol=50 n_zol=1
set rsmall=0.00125
set echo off
createLoadingPlate
set echo on
;set the color to distinguish
; Set the loading plate color
prop color 8 range z 0.101 0.106
prop color 8 range z -0.0025 -0.001
;=============================================================================
============================================================
;set loading plate width clump
; Generated as clump as an loaded plate
clump id=10001 perm upd=50 range z 0.101 0.106
clump id=10002 perm upd=50 range z -0.0025 -0.001
;=============================================================================
```

```
;=============================================================
;bottomPlate
; Fix bottom loading plate
group bottomPlate range z -0.0025 -0.001
fix x y z xspin yspin zspin range group bottomPlate
ini x 0.0 y 0.0 z 0.0 xspin 0.0 yspin 0.0 zspin 0.0 range group bottomPlate ;first make clump ,then fix it
;=============================================================
;topPlate
;Top loading plate, mainly used for subsequent control of rotational speed as a source of load for torsional shear
group topPlate range z 0.101 0.106
fix z zspin range group topPlate
ini z 0.0 range group topPlate ;first make clump ,then fix it
;=============================================================
;set plate params
; Set the parameters of the contact between the parameters of the loading plate and the model, the strength, etc. need to be set larger, so as not to be disconnected here when loading.
prop density=*** kn=*** ks=*** fric=*** range z 0.099 0.106
prop density=*** kn=*** ks=*** fric=*** range z -0.0025 0.0001
prop pb_kn=*** pb_ks=*** pb_n=*** pb_s=*** pb_rad=*** range z 0.099 0.106
prop pb_kn=*** pb_ks=*** pb_n=*** pb_s=*** pb_rad=*** range z -0.0025 0.0001
;=============================================================
;load_n_stress
def load_n_stress
    bp_load_ns=find_ball(120001)
    b_zfap(bp_load_ns)=(0.0*pi*0.025^2.0)
end
load_n_stress;
;=============================================================
;make_clump or not
;if make_clump_control = 1
; whether to set the aggregate to clump form
def make_clump
  if make_clump_control=1 then
    loop _n(1,maxBallId)
      if eachAggregateSmallBallNum(_n) # 0 then
        aggregate = 'aggregate' + string(_n)
        command
          clump id=_n range group aggregate
```

```
                    end_command
                endif
            end_loop
        endif
end
set make_clump_control=0 ;make_clump not
set echo off
make_clump
set echo on
;=============================================================================
============================================================
;                                                    LOAD SETTING
;=============================================================================
============================================================
;load_vel
; loading, set the loading speed
def load_vel
        _vfinal=***
        rzvel_load=_vfinal/108.0
  end
load_vel
;startup Record old_time
; Record start time
def startup
        old_time=time
end
;torsional_loading
; Assign load speed to top load plate
def torsional_loading
        ;while_stepping
        real_time=time-old_time
        bp_load=find_ball(100001)
        clp_load=b_clump(bp_load)
        cl_rzvel(clp_load)=rzvel_load
end
set fishcall 0 torsional_loading
;collectData
; Collect relevant data, mainly including stress strain, etc.
def collectData
        real_time = time-old_time
        bp_load = find_ball(100001)
        clp_load = b_clump(bp_load)
        torque = abs(cl_zmom(clp_load))-torque0
        s_stress = 2*torque/(pi*0.05^3) ;
        t_strain = 0.05*rzvel_load/0.1*real_time ;
```

```
        linear_disp = real_time * _vfinal
end
;=============================================================================
============================================================
;Preloading
; Preloading
set dt auto
cyc 100
save torsionTest_60_00100_start.sav
cyc 900
save torsionTest_60_01000_start.sav
cyc 1000
save torsionTest_60_02000_start.sav
cyc 8000
save torsionTest_60_10000_start.sav
;=============================================================================
============================================================
;output_setup
; Define model ball particle information output function, which can be used for data processing, such as generating model destruction animation, etc.
def output_setup
    IO_READ  = 0
    IO_WRITE = 1
    IO_FISH  = 0
    IO_ASCII = 1
    array Ball_info(100000)
end
;output_ball_info
def output_ball_info
    cycle_cnt = cycle_cnt + 1
    if cycle_cnt < cycle_cnt_dist then
        exit
    end_if
    cycle_tot=cycle_tot+cycle_cnt
    cycle_cnt = 0
    i=1
    Ball_info(i)=' TITLE = "Information of all balls "'
    i=i+1
    Ball_info_name_str = ' VARIABLES= "Ball_Xpos"  "Ball_Ypos" "Ball_Zpos" "Ball_Radius" '
    Ball_info_name_str = Ball_info_name_str + '"Ball_Xdisp"  "Ball_Ydisp"  "Ball_Zdisp" '
    Ball_info_name_str = Ball_info_name_str + '"Ball_XYdisp" '
    Ball_info_name_str = Ball_info_name_str + '"Ball_Xvel" "Ball_Yvel" "Ball_Zvel" '
    Ball_info_name_str = Ball_info_name_str + '"Ball_Omega" '
    Ball_info_name_str = Ball_info_name_str + '"Ball_Color" '
    Ball_info(i)       = Ball_info_name_str + '"Ball_Dens" "Ball_Id" '
```

```
    bp=ball_head
    loop while bp # null
            i=i+1
            Ball_info_str = '  ' + string(b_x(bp)) + '  ' + string(b_y(bp)) + '  ' + string(b_z(bp)) + '  ' + string(b_rad(bp))
            Ball_info_str = Ball_info_str + '  ' + string(b_xdisp(bp)) + '  ' + string(b_ydisp(bp)) + '  ' + string(b_zdisp(bp))
            Ball_info_str = Ball_info_str + '  ' + string(((abs(b_xdisp(bp)))^2.0+(abs(b_ydisp(bp)))^2.0)^0.5)
            Ball_info_str = Ball_info_str + '  ' + string(b_xvel(bp)) + '  ' + string(b_yvel(bp)) + '  ' + string(b_zvel(bp))
            Ball_info_str = Ball_info_str + '  ' + string((((abs(b_xvel(bp)))^2.0+(abs(b_yvel(bp)))^2.0)^0.5)/(((abs(b_x(bp)))^2.0+(abs(b_y(bp)))^2.0)^0.5+0.00000000001))
            Ball_info_str = Ball_info_str + '  ' + string(b_color(bp))
            Ball_info(i)  = Ball_info_str + '  ' + string(b_dens(bp)) + ' ' + string(b_id(bp))
        bp = b_next(bp)
    end_loop
    Ball_info_file_name = 'torsionTest' + string(torsion_num) +'_Ball_info_' + string(cycle_tot) + '.dat'
    status = open(Ball_info_file_name, IO_WRITE, IO_ASCII)
    status = write(Ball_info,i)
    status = close
end
output_setup
set cycle_cnt=0 cycle_tot=0
set fishcall 3 output_ball_info
set torsion_num=60 cycle_cnt_dist=20000
;=============================================================================
============================================================
;loading by cycle
; Cyclic loading until the model destroyed, the criterion for model failure is when the loading stress is below a percentage of the maximum stress
def make_cyc
    mom_max = 0.0
    cyc_control = 1
    mimom = 0.0
    loop while cyc_control # 0
        mom_before = mimom
        command
            cycle 10000
        end_command
        mimom = int(torque/0.01)*0.01
        if mimom > mom_max then
            mom_max = mimom
```

```
            end_if
            if mimom < mom_before then
                cyff = mom_max * 0.6 ; 60% of the maximum stress is saved
            else
                cyff = mimom
            end_if
            if mimom < cyff then
                cyc_control = 0
            end_if
        end_loop
end
;=============================================================================
============================================================
;save_result
; Define the model after each running a certain number of steps, save it dynamically, and analyze and
observe the model test process.
def save_result
        save_n = save_n + 1
        if save_n < save_n_dist then
            exit
        end_if
        save_tot=save_tot+save_n
        save_n = 0
        save_name = 'torsionTest_60_' + string(save_tot) + '.sav'
        command
            save save_name
        endcommand
end
set save_n=0 save_tot=0 save_n_dist=100000
set fishcall 1 save_result
def collectData0
        bp_load=find_ball(100001)
        clp_load=b_clump(bp_load)
        torque0=abs(cl_zmom(clp_load))
end
collectData0
set time 0.0
print torque0
set fishcall 0 collectData
; Defining historical variables
hist id 101 rzvel_load
hist id 102 real_time
hist id 103 torque
hist id 104 s_stress        ; Shear stress
hist id 105 t_strain        ; Torsional strain
```

```
hist id 106 linear_disp
; Define a view for saving as a picture
plot create plot_model_01                                    ;6
plot set title text 'model_all'
plot set caption size 35
plot set background white
plot set rotation 20 0.0 180
;plot set mag 1.5
plot add ball white blue yellow
; Filename: makejpg.fis
;
; Describe: save the picture on the process of torsion test
;
; Written by Y.
;=============================================================================
================================================
; Define automatic hold picture function
def make_jpg
      cycle_n = cycle_n + 1
      if cycle_n < cycle_n_dist then
            exit
      end_if
      cycle_total=cycle_total+cycle_n
      cycle_n = 0
         fjpg = 'torsionTest_irregular_60_model_01_' + string(cycle_total) + '.jpg'
         command
              set out @fjpg
              set plot bitmap type jpg
              set plot bitmap size 1024 768
              plot current 6
              plot hard
           end_command
end
set cycle_n=0 cycle_total=0
set fishcall 1 make_jpg
set cycle_n_dist=10000
;call plotview.fis                ; plot view
;call makejpg.fis                 ; save jpg
; Open related views, etc.
plot create plot_torque              ;16
plot add hist 103 vs 102
plot set background white
;plot show
plot create plot_rzvel_load        ;17
plot add hist 101 vs 102
```

```
plot set background white
;plot show
plot create plot_s_stress          ;18
plot add hist 104 vs 102
plot set background white
;plot show
plot create plot_t_strain          ;19
plot add hist 105 vs 102
plot set background white
;plot show
plot create plot_sstress_strain          ;19
plot add hist 104 vs 105
plot set background white
;plot show
plot create plot_linear_disp          ;20
plot add hist 106 vs 102
plot set background white
;plot show
plot create plot_tstress_liner_dis          ;21
plot add hist 104 vs 106
plot set background white
;plot show
set hist_rep=10000
crk_init
startup
cyc 50000
save torsionTest_60_60000_start.sav
;=====================================
; Running model
make_cyc
save torsionTest_60_final.sav
;set log off
; Filename: setModelPoro.dat
;
; Describe: set porosity of model
;
; Written by Y.
; Generate void reduction, do no need to run in actual test
;=============================================================================
==================================================
new
restore modelIrregular.sav
set random 10000 ; the random can change for diffrent poro, such as 10001 10060 ...
; Define views for viewing gaps
def plotModelPoro
```

```
    command
        title 'Torsion Test by Y'
        ; general view
        plot create plot_model_poro
        plot set title text 'plot_model_poro'
        plot set caption size 20
        plot set mag 1.56
        plot set background white
        plot set rotation 20.0 0.0 50.0
        plot add wall wireframe on
        plot add ball    yellow blue
        plot show
    end_command
end
;
; Air voids generation function
def delPoros
        poros = 0.04
        ndel = int( poros * totalBallCount )
; Used to record the position information of the deleted ball particles
        array del_b_x(10000)
        array del_b_y(10000)
        array del_b_z(10000)
        aa = 1
        bbi = 0
        loop while aa <= ndel
                bb = int((totalBallCount)*urand + 1)
                bp = find_ball(bb)
                if bp # null then
                if b_color(bp)=1 then
                      bbi = bbi+1
                      del_b_x(bbi)=b_x(bp)
                      del_b_y(bbi)=b_y(bp)
                      del_b_z(bbi)=b_z(bp)
                      ii = b_delete(bp)
                      aa = aa +1
                  endif
                  endif
        endloop
end
delPoros
plotModelPoro
save modelPoro_04.sav
; press any key to makeBallRecurrencePoro
pause 3
```

```
delete ball
; Regenerate the ball according to the position information of the delete ball particles, and the gap distribution can be reproduced.
def makeBallRecurrencePoro
        loop ii(1,bbi)
          _xx = float(del_b_x(ii))
          _yy = float(del_b_y(ii))
          _zz = float(del_b_z(ii))
           command
               ball id ii x _xx    y _yy    z _zz rad 0.00125
               prop color 1 range id ii
           endcommand
        endloop
end
set echo off
makeBallRecurrencePoro
set echo on
```