FLAC3D Gitbook

Kyeong Sun Kim

Civil and Environmental Engineering

Seoul National University

Sept, 2021

Contents

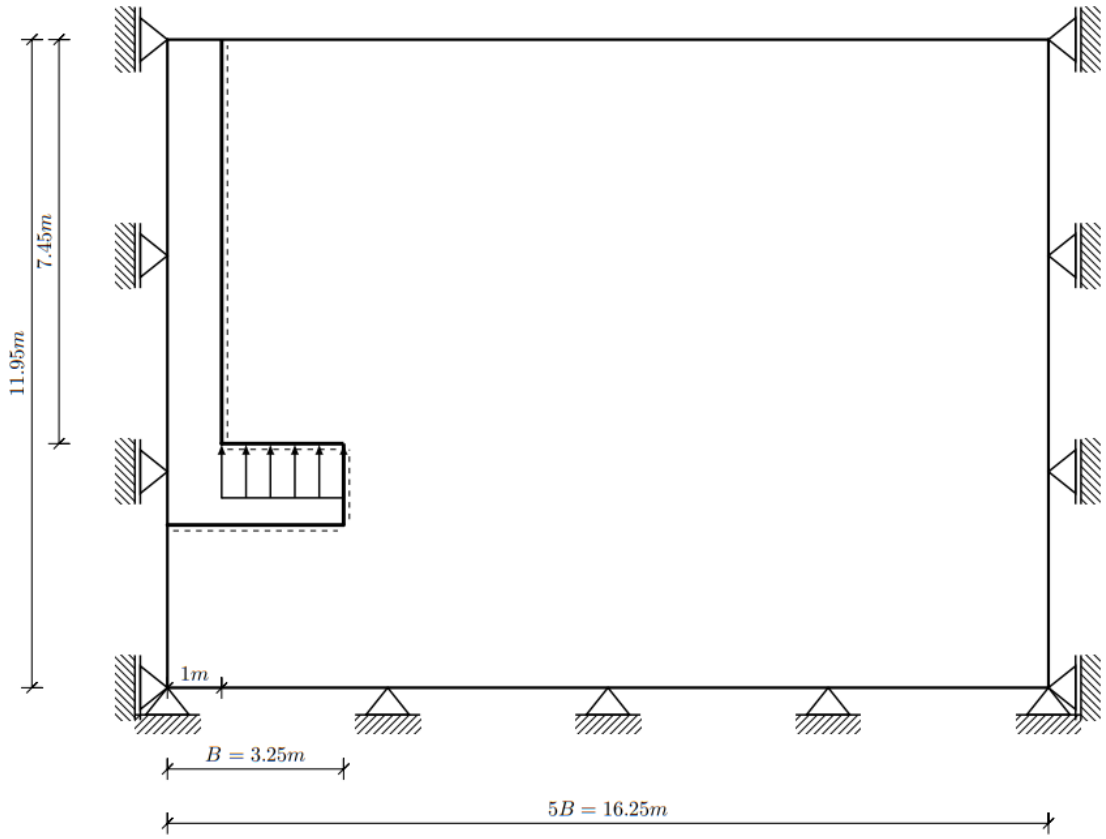
There is grandeur in this view of life, with its several powers, having been originally breathed into a few forms or into one; and that, whilst this planet has gone cycling on according to the fixed law of gravity, from so simple a beginning endless forms most beautiful and most wonderful have been, and are being, evolved.

Charles Darwin(1809-1882)

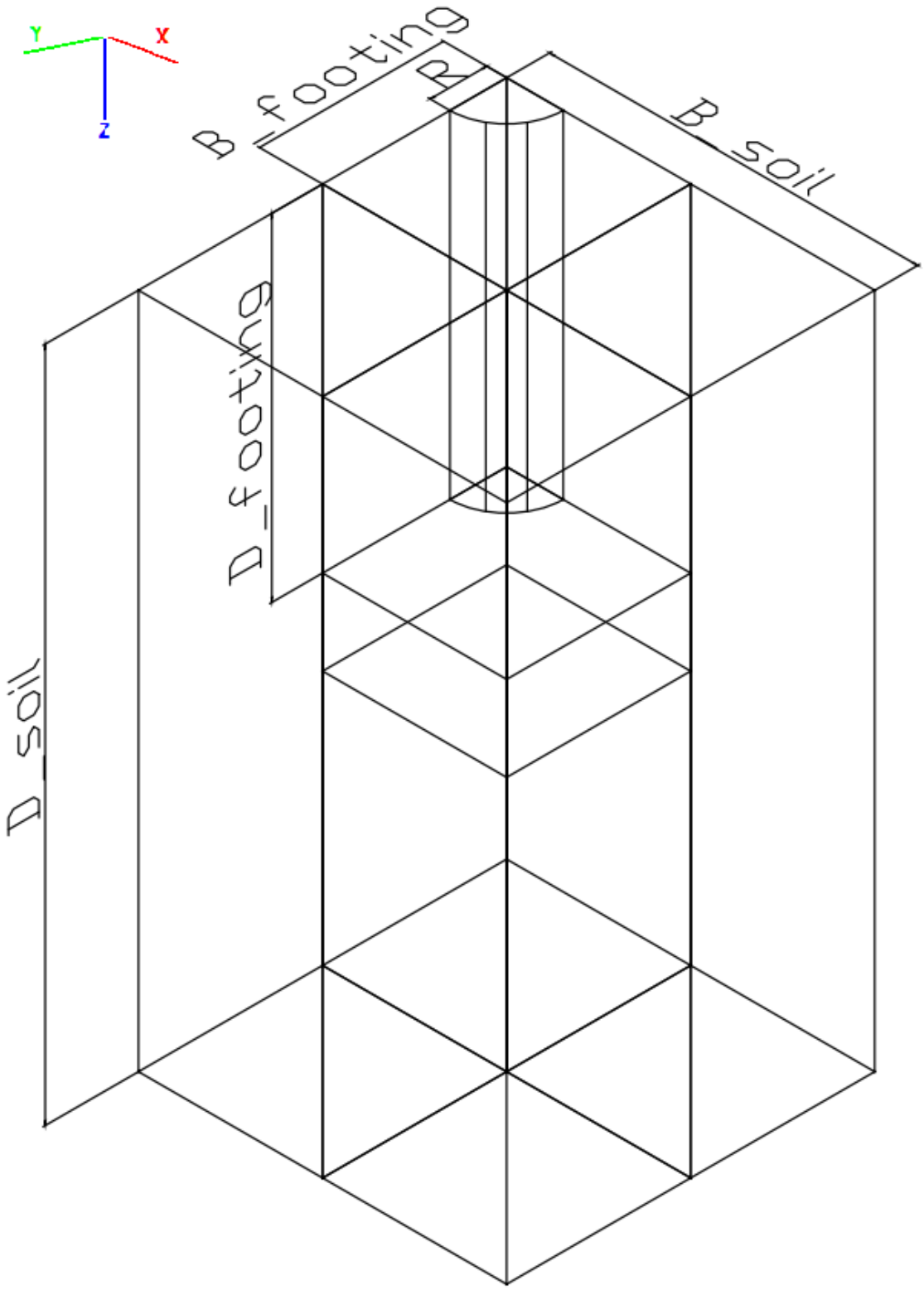
1

KAIST Model with Tran

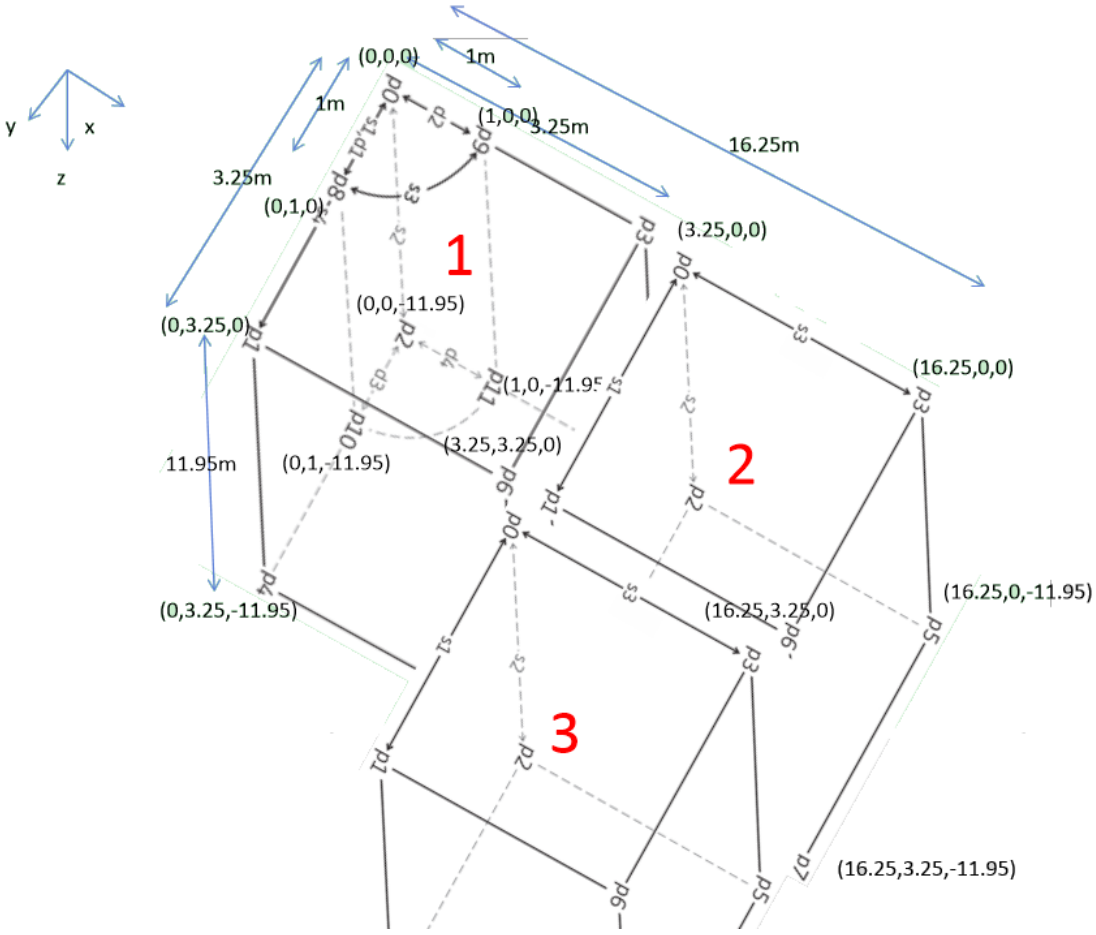
1.1 Initial Configuration



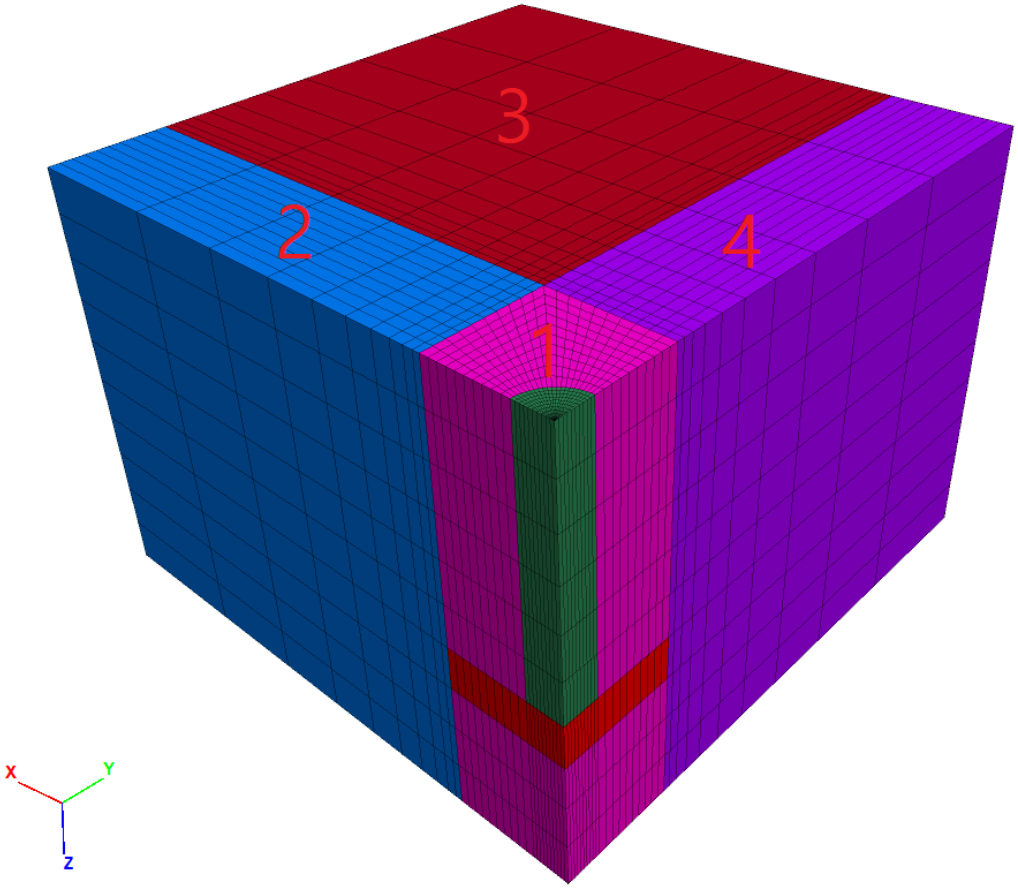
1. KAIST Model with Tran



1. KAIST Model with Tran



1. KAIST Model with Tran



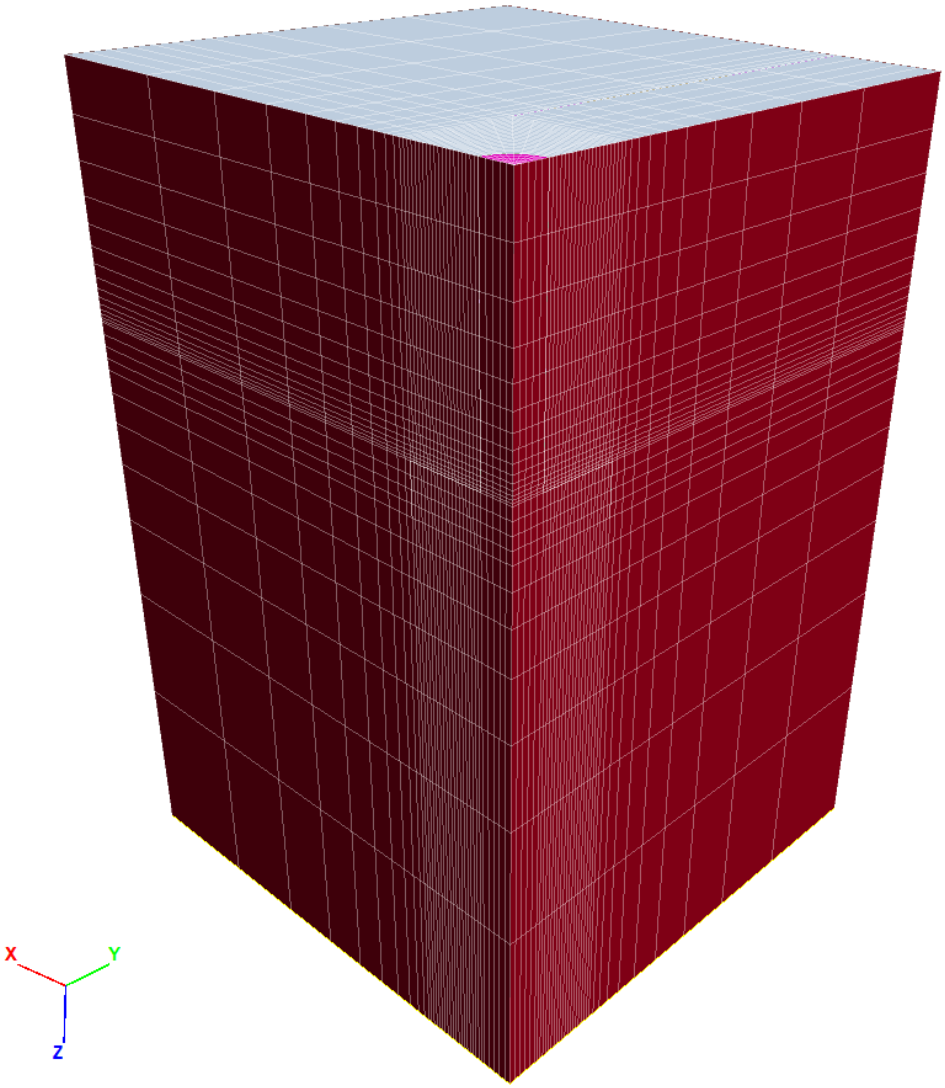


Figure 1.1: Coarse Mesh

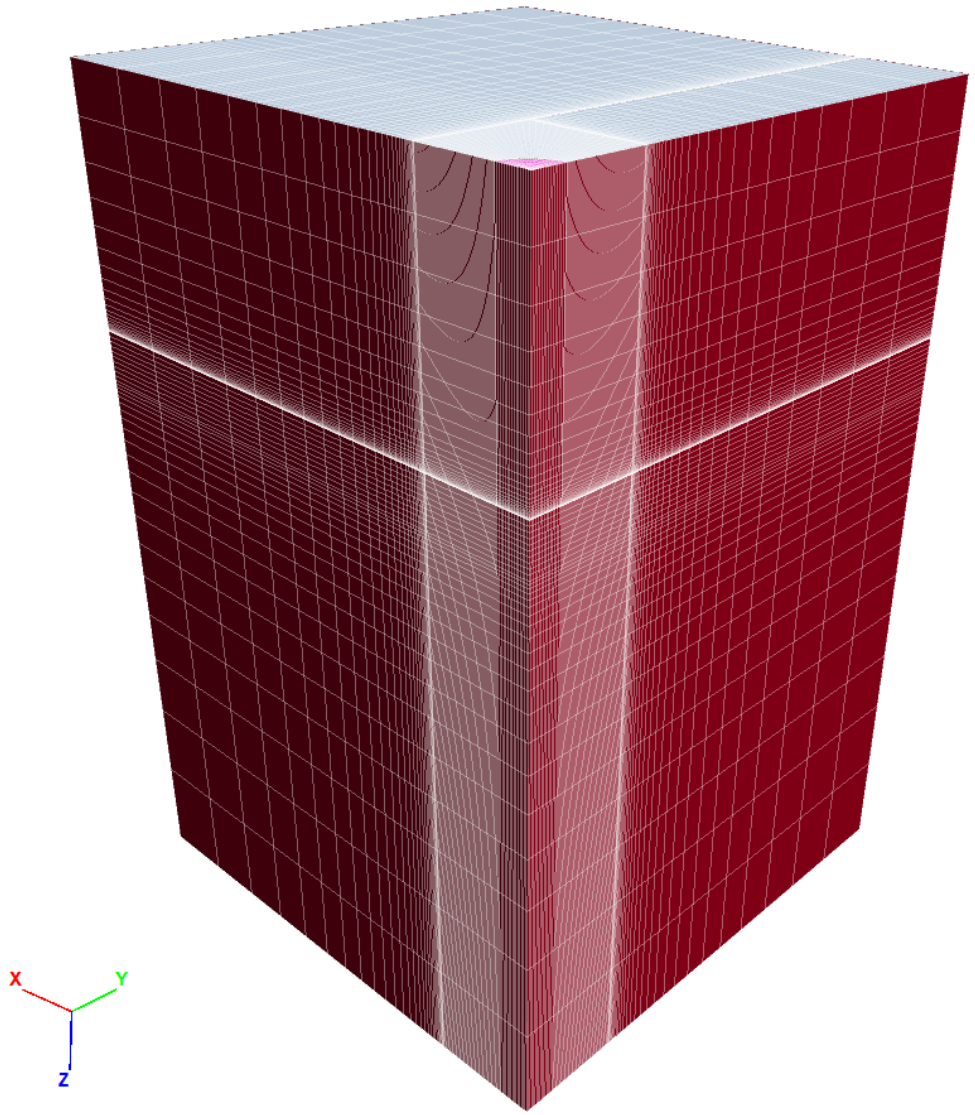


Figure 1.2: Fine Mesh

1.2 Code Block

```
############## SI UNIT ###################
####### m, Pa, kg/m3, m/s2,   ##########
#########################################

print("       ((((((((((((((((======== NEW RUN ========))))))))))))))))")
```

1. KAIST Model with Tran

```
import itasca as it
import numpy as np
np.set_printoptions(threshold=20)
it.command("python-reset-state false")
from itasca import zonearray as za
from itasca import gridpointarray as gpa

########################################
############ PARAMETERS ################
########################################

# Mesh Details
_radial = 5
_perimeter = 5
_axial = 10
_outer = 10

# Physical Constants
_gravity = 9.80665
_K = 0.318 # 1-sin(\phi) (Jacky)

# Dimensions
_D_shaft = 1.0
_H_shaft = 7.45
_T_plate = 1.5
_B_footing = 3.25 + 0.1
_D_footing = _H_shaft + _T_plate
_B_soil = _B_footing*5
_D_soil = _D_footing + _B_footing*5
```

```
# Concrete Properties
_bulk = 4.1667e10
_shear = 1.9231e10
_density_concrete = 2400

# Soil Properties
_E_o = 80e6
_const = 1e6
_poisson = 0.3
_density_soil = 1530
_cohesion = 10
_friction = 47

# Interface Properties
_stiff_norm = 1e10
_stiff_shear = 1e10

#########################################
############### ZONES ##################
#########################################


command_zone = """

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;   Upper Plate    ;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

model new
```

```
z crea r-c p 0 (0,0,0) ...
           p 1 (0,{B_footing},0) ...
           p 2 (0,0,{H_shaft}) ...
           p 3 ({B_footing},0,0) ...
           p 4 (0,{B_footing},{H_shaft}) ...
           p 5 ({B_footing},0,{H_shaft}) ...
           p 6 ({B_footing},{B_footing},0) ...
           p 7 ({B_footing},{B_footing},{H_shaft}) ...
           p 8 (0,{D_shaft},0) ...
           p 9 ({D_shaft},0,0) ...
           p 10 (0,{D_shaft},{H_shaft}) ...
           p 11 ({D_shaft},0,{H_shaft}) ...
           size {radial} 15 10 20 ...
           rat 1 0.8 1 1 ...
           fill

z crea b p 0 ({B_footing},0,0) ...
         p 1 ({B_footing},{B_footing},0) ...
         p 2 ({B_footing},0,{H_shaft}) ...
         p 3 ({B_soil},0,0) ...
         p 4 ({B_footing},{B_footing},{H_shaft}) ...
         p 5 ({B_soil},0,{H_shaft}) ...
         p 6 ({B_soil},{B_footing},0) ...
         p 7 ({B_soil},{B_footing},{H_shaft}) ...
         size {radial} 15 {outer} ...
         rat 1 0.8 1.4 ;blue

z crea b p 0 ({B_footing},{B_footing},0) ...
         p 1 ({B_footing},{B_soil},0) ...
         p 2 ({B_footing},{B_footing},{H_shaft}) ...
```

```
         p 3 ({B_soil},{B_footing},0) ...
         p 4 ({B_footing},{B_soil},{H_shaft}) ...
         p 5 ({B_soil},{B_footing},{H_shaft}) ...
         p 6 ({B_soil},{B_soil},0) ...
         p 7 ({B_soil},{B_soil},{H_shaft}) ...
         size {radial} 15 {outer} ...
         rat 1.4 0.8 1.4

z crea b p 0 (0,{B_footing},0) ...
         p 1 (0,{B_soil},0) ...
         p 2 (0,{B_footing},{H_shaft}) ...
         p 3 ({B_footing},{B_footing},0) ...
         p 4 (0,{B_soil},{H_shaft}) ...
         p 5 ({B_footing},{B_footing},{H_shaft}) ...
         p 6 ({B_footing},{B_soil},0) ...
         p 7 ({B_footing},{B_soil},{H_shaft}) ...
         size {outer} 15 {radial} ...
         rat 1.4 0.8 1 ;purple

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;   Plate    ;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

z crea r-c p 0 (0,0,{H_shaft}) ...
           p 1 (0,{B_footing},{H_shaft}) ...
           p 2 (0,0,{D_footing}) ...
           p 3 ({B_footing},0,{H_shaft}) ...
           p 4 (0,{B_footing},{D_footing}) ...
           p 5 ({B_footing},0,{D_footing}) ...
           p 6 ({B_footing},{B_footing},{H_shaft}) ...
```

```
            p 7 ({B_footing},{B_footing},{D_footing}) ...
            p 8 (0,{D_shaft},{H_shaft}) ...
            p 9 ({D_shaft},0,{H_shaft}) ...
            p 10 (0,{D_shaft},{D_footing}) ...
            p 11 ({D_shaft},0,{D_footing}) ...
            size {radial} 4 10 20 ...
            rat 1 1 1 1 ...
            fill

z crea b p 0 ({B_footing},0,{H_shaft}) ...
         p 1 ({B_footing},{B_footing},{H_shaft}) ...
         p 2 ({B_footing},0,{D_footing}) ...
         p 3 ({B_soil},0,{H_shaft}) ...
         p 4 ({B_footing},{B_footing},{D_footing}) ...
         p 5 ({B_soil},0,{D_footing}) ...
         p 6 ({B_soil},{B_footing},{H_shaft}) ...
         p 7 ({B_soil},{B_footing},{D_footing}) ...
         size {radial} 4 {outer} ...
         rat 1 1 1.4 ;blue

z crea b p 0 ({B_footing},{B_footing},{H_shaft}) ...
         p 1 ({B_footing},{B_soil},{H_shaft}) ...
         p 2 ({B_footing},{B_footing},{D_footing}) ...
         p 3 ({B_soil},{B_footing},{H_shaft}) ...
         p 4 ({B_footing},{B_soil},{D_footing}) ...
         p 5 ({B_soil},{B_footing},{D_footing}) ...
         p 6 ({B_soil},{B_soil},{H_shaft}) ...
         p 7 ({B_soil},{B_soil},{D_footing}) ...
         size {radial} 4 {outer} ...
         rat 1.4 1 1.4
```

```
z crea b p 0 (0,{B_footing},{H_shaft}) ...
         p 1 (0,{B_soil},{H_shaft}) ...
         p 2 (0,{B_footing},{D_footing}) ...
         p 3 ({B_footing},{B_footing},{H_shaft}) ...
         p 4 (0,{B_soil},{D_footing}) ...
         p 5 ({B_footing},{B_footing},{D_footing}) ...
         p 6 ({B_footing},{B_soil},{H_shaft}) ...
         p 7 ({B_footing},{B_soil},{D_footing}) ...
         size {outer} 4 {radial} ...
         rat 1.4 1 1 ;purple



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;   Down to Bottom    ;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

z crea r-c p 0 (0,0,{D_footing}) ...
           p 1 (0,{B_footing},{D_footing}) ...
           p 2 (0,0,{D_soil}) ...
           p 3 ({B_footing},0,{D_footing}) ...
           p 4 (0,{B_footing},{D_soil}) ...
           p 5 ({B_footing},0,{D_soil}) ...
           p 6 ({B_footing},{B_footing},{D_footing}) ...
           p 7 ({B_footing},{B_footing},{D_soil}) ...
           p 8 (0,{D_shaft},{D_footing}) ...
           p 9 ({D_shaft},0,{D_footing}) ...
           p 10 (0,{D_shaft},{D_soil}) ...
           p 11 ({D_shaft},0,{D_soil}) ...
```

```
            size {radial} 7 10 20 ...
            rat 1 1.4 1 1 ...
            fill

z crea b p 0 ({B_footing},0,{D_footing}) ...
         p 1 ({B_footing},{B_footing},{D_footing}) ...
         p 2 ({B_footing},0,{D_soil}) ...
         p 3 ({B_soil},0,{D_footing}) ...
         p 4 ({B_footing},{B_footing},{D_soil}) ...
         p 5 ({B_soil},0,{D_soil}) ...
         p 6 ({B_soil},{B_footing},{D_footing}) ...
         p 7 ({B_soil},{B_footing},{D_soil}) ...
         size {radial} 7 {outer} ...
         rat 1 1.4 1.4 ;blue

z crea b p 0 ({B_footing},{B_footing},{D_footing}) ...
         p 1 ({B_footing},{B_soil},{D_footing}) ...
         p 2 ({B_footing},{B_footing},{D_soil}) ...
         p 3 ({B_soil},{B_footing},{D_footing}) ...
         p 4 ({B_footing},{B_soil},{D_soil}) ...
         p 5 ({B_soil},{B_footing},{D_soil}) ...
         p 6 ({B_soil},{B_soil},{D_footing}) ...
         p 7 ({B_soil},{B_soil},{D_soil}) ...
         size {radial} 7 {outer} ...
         rat 1.4 1.4 1.4

z crea b p 0 (0,{B_footing},{D_footing}) ...
         p 1 (0,{B_soil},{D_footing}) ...
         p 2 (0,{B_footing},{D_soil}) ...
         p 3 ({B_footing},{B_footing},{D_footing}) ...
```

```
          p 4 (0,{B_soil},{D_soil}) ...
          p 5 ({B_footing},{B_footing},{D_soil}) ...
          p 6 ({B_footing},{B_soil},{D_footing}) ...
          p 7 ({B_footing},{B_soil},{D_soil}) ...
          size {outer} 7 {radial} ...
          rat 1.4 1.4 1 ;purple

"""
command = command_zone.format(
D_shaft = _D_shaft,
H_shaft = _H_shaft,
B_footing = _B_footing,
B_soil = _B_soil,
D_soil = _D_soil,
D_footing = _D_footing,
D_soil2 = _D_soil2,
radial=_radial,
axial = _axial,
perimeter = _perimeter,
outer = _outer)

it.command(command)

print("               ======== PASSED ZONES ========")

########################################
################ GROUPS ################
########################################

p = za.pos()
```

1. KAIST Model with Tran

```
x,y,z = p.T

shaft = reduce(np.logical_and, (np.sqrt(x**2+y**2)<_D_shaft, z<_H_shaft))
plate = reduce(np.logical_and, (x<_B_footing,y<_B_footing, z>_H_shaft,z<_D_footing)
za.set_group(shaft, "shaft")
za.set_group(plate, "plate")

print("radial mesh number is ", _radial)
print(it.zone.count(), "zones in whole model")
print(za.in_group("shaft").sum() + za.in_group("plate").sum(), "zones in shaft+plat

print("                ======== PASSED GROUPS ========")

########################################
######## CONSTITUTIVE MODEL ############
########################################

command_zone = """
zone cmodel assign mohr-coulomb range group ...
'Radial cylinder1' or ...
'Radial cylinder9' or ...
'plate' or ...
'shaft' or ...
'brick2' or ...
'brick3' or ...
'brick4' or ...
'brick6' or ...
'brick7' or ...
'brick8' or ...
'brick10' or ...
```

```
'brick11' or ...
'brick12'

; assign soil properties
zone property young {E_o_} ...
poisson {poisson_} ...
density {density_soil_} ...
cohesion {cohesion_} ...
friction {friction_} ...
flag-brittle true

; let modulus depends on depth
fish define fname(E_o,const)
loop foreach pnt zone.list
z_depth = zone.pos.z(pnt)
E = E_o+const*math.sqrt(z_depth)
zone.prop(pnt,'young')=E
end_loop
end
@fname({E_o_},{const_})
"""
command = command_zone.format(
E_o_=_E_o,
const_=_const,
poisson_=_poisson,
density_soil_=_density_soil,
cohesion_ = _cohesion,
friction_ = _friction,
bulk_=_bulk,
shear_=_shear,
```

```
density_=_density_concrete)

it.command(command)

print("               ======== PASSED CONSTITUTIVE MODEL ========")

########################################
############# INTERFACE ################
########################################

command_zone = """
zone interface 'interface 1' create by-face separate range group 'plate' ...
group 'Radial cylinder1' or ...
'brick2' or ...
'brick3' or ...
'brick4' or ...
'brick6' or ...
'brick7' or ...
'brick8' or ...
'brick10' or ...
'brick11' or ...
'brick12'

zone interface 'interface 2' create by-face separate range group 'shaft' ...
group 'Radial cylinder1' or ...
'brick2' or ...
'brick3' or ...
'brick4' or ...
'brick6' or ...
'brick7' or ...
```

```
'brick8' or ...
'brick10' or ...
'brick11' or ...
'brick12'

zone interface 'interface 1' node property ...
stiffness-normal {stiff_norm_} ...
stiffness-shear {stiff_shear_} ...
friction {friction_} ...
cohesion {cohesion_}

zone interface 'interface 2' node property ...
stiffness-normal {stiff_norm_} ...
stiffness-shear {stiff_shear_} ...
friction {friction_} ...
cohesion {cohesion_}
"""
command = command_zone.format(
stiff_norm_=_stiff_norm,
stiff_shear_=_stiff_shear,
friction_ = _friction,
cohesion_ = _cohesion)

it.command(command)

print("               ======== PASSED INTERFACE ========")

########################################
######## BOUNDARY CONDITIONS ###########
########################################
```

```
it.command("""
zone face skin
;zone gridpoint fix velocity-x 0 ...
zone face apply velocity-normal 0 ...
range group 'West2' or 'East3'

;zone gridpoint fix velocity-y 0 ...
zone face apply velocity-normal 0 ...
range group 'South3' or 'North3'

zone gridpoint fix velocity (0,0,0) range group 'Top2'
""")

print("               ======== PASSED BOUNDARY CONDITIONS ========")

########################################
############# INITILIZE ################
########################################

it.command("""
model gravity 0 0 {gravity_}
zone initialize-stress ratio {K_}
zone interface 'interface 1' node initialize-stresses
zone interface 'interface 2' node initialize-stresses
zone ratio local
model solve ratio 1e-5
model save 'initialization'
```

```
""".format(gravity_=_gravity, K_ = _K))

print("               ======== PASSED INITIAL EQUILIBRIUM ========")

########################################
########## VERTICAL LOADING ############
########################################

top = reduce(np.logical_and, (np.sqrt(x**2+y**2)<_D_shaft, z==np.amin(z)))
za.set_group(top, "top")
print(za.in_group("top").sum(), "zones in top group.")

command_zone ="""

; assign concrete properties
zone cmodel assign elastic range group 'shaft' or 'plate'
zone property bulk {bulk_} ...
shear {shear_} ...
density {density_} range group 'shaft' or 'plate'

zone initialize state 0
zone gridpoint initialize displacement (0,0,0)
zone gridpoint initialize velocity     (0,0,0)

; apply displacement to top
zone face apply velocity-normal 1e-5 range group 'top'

history interval 100

zone history name 'disp' displacement-y position (1,7.46,1)
```

1. KAIST Model with Tran

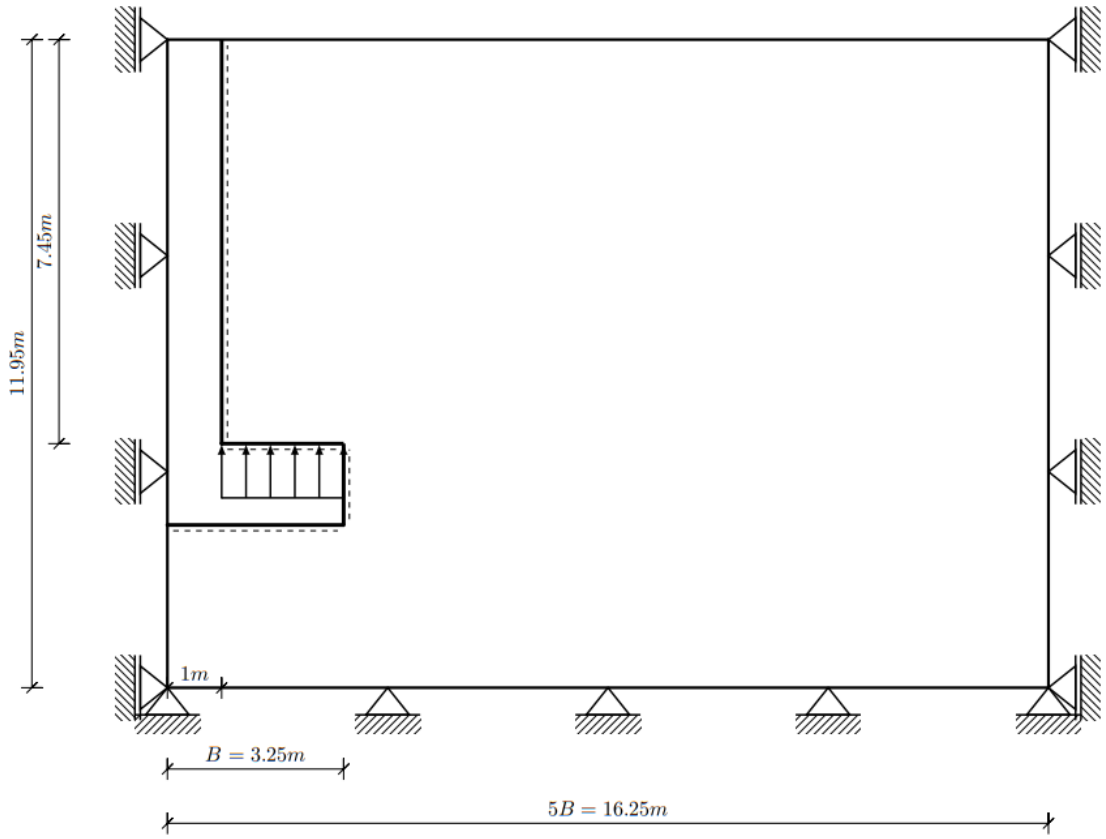
```
; find gridpoints of structure, store in map called cap

fish define find_cap
    global cap = map
    loop foreach local zone zone.list
        if zone.isgroup(zone,'topplate') then
            cap(zone.id(zone)) = zone
        endif
    endloop
end

fish history name 'cap' @find_cap

fish define vert_load
    local yftot = 0.0
    loop foreach zone cap
        yftot = yftot + zone.stress.int(zone)
    end_loop
    vert_load = -yftot
end

fish history name 'top_sum_stress' @vert_load

model largestrain true
model step 3000
model save 'vertical-loading'
"""
command = command_zone.format(
bulk_=_bulk,
```

```
shear_=_shear,
density_=_density_concrete)


it.command(command)


print("                ======== PASSED VERTICAL LOADING ========")
```

2

KAIST Model

2.1 Initial Configuration



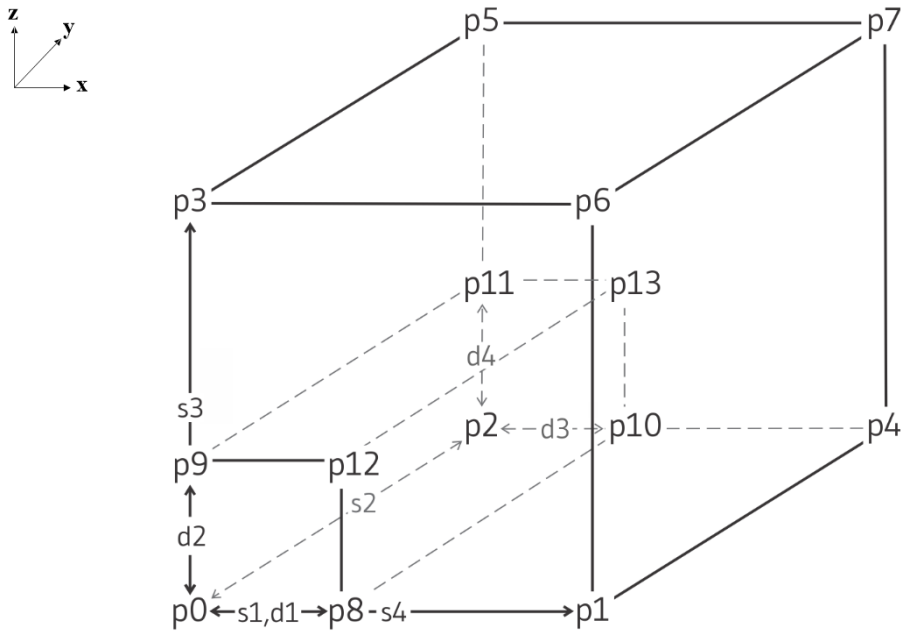

```
import itasca as it
import numpy as np
np.set_printoptions(threshold=20)
it.command("python-reset-state false")
from itasca import zonearray as za
from itasca import gridpointarray as gpa

########################################
############ PARAMETERS ################
########################################
```

2. KAIST Model

```
# Mesh Details
_radial = 20
_perimeter = _radial
_axial = 2*_radial
_outer = 2*_radial

# Physical Constants
_gravity = 9.80665
_K = 0.5

# Dimensions
_D_shaft = 1.0
_H_shaft = 7.45
_T_plate = 1.5
_B_footing = 3.25
_D_footing = _H_shaft + _T_plate
_B_soil = _B_footing*5
_D_soil = _D_footing+3

# Concrete Properties
_bulk = 4.1667e10
_shear = 1.9231e10 #E_star = 8.777e7
_density_concrete = 2400

# Soil Properties
_E_o = 80e6
_const = 1e6
_poisson = 0.3
_density_soil = 1530
```

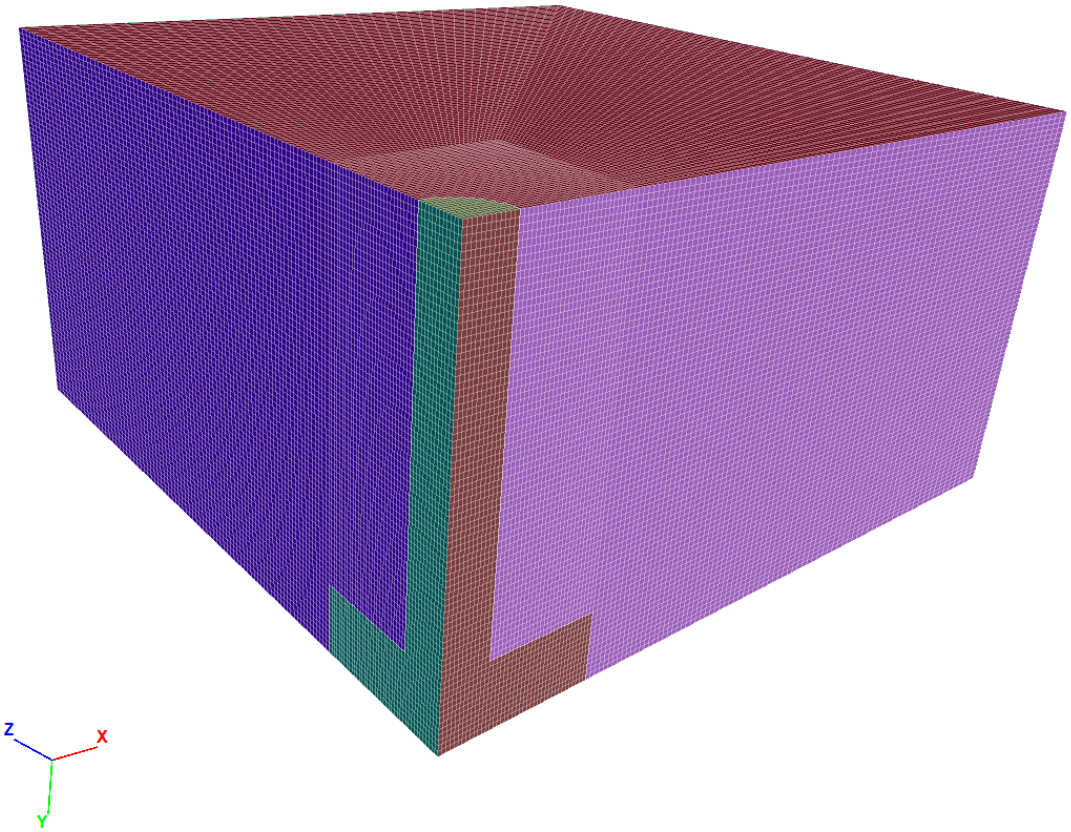
```
# Interface Properties
_stiff_norm = 8.777e8
_stiff_shear = 8.777e8
```

2.2 Zones




```
########################################
############### ZONES ##################
########################################

print("               ((((((((((((((((======== NEW RUN ========))))))))))))))))")

command_zone = """
model new
z crea r-t p 0 (0,0,0) ...
           p 1 ({B_soil},0,0) ...
           p 2 (0,{D_soil},0) ...
```

```
            p 3 (0,0,{B_soil}) ...
            p 4 ({B_soil},{D_soil},0) ...
            p 5 (0,{D_soil},{B_soil}) ...
            p 6 ({B_soil},0,{B_soil}) ...
            p 7 ({B_soil},{D_soil},{B_soil}) ...
            p 8 ({B_footing},0,0) ...
            p 9 (0,0,{B_footing}) ...
            p 10 ({B_footing},{D_soil},0) ...
            p 11 (0,{D_soil},{B_footing}) ...
            p 12 ({B_footing},0,{B_footing}) ...
            p 13 ({B_footing},{D_soil},{B_footing}) ...
            size {radial} {axial} {perimeter} {outer} ...
            rat 1 1 1 1.1 ...
            fill
"""
command = command_zone.format(
B_footing = _B_footing,
B_soil = _B_soil,
D_soil = _D_soil,
radial=_radial,
axial = _axial,
perimeter = _perimeter,
outer = _outer)

it.command(command)
print("radial mesh number is ", _radial)
```



2.3 Constitutive Model

It is easy to loop over sets of model objects (i.e., zones, gridpoints, structural element nodes, etc.) using the loop foreach construct. In this case, a container of objects must be given by a FISH intrinsic such as zone.list. A practical use of the loop foreach construct is to install a nonlinear initial distribution of elastic moduli in a FLAC3D grid. Suppose that the Young's modulus at a site is given by this equation:

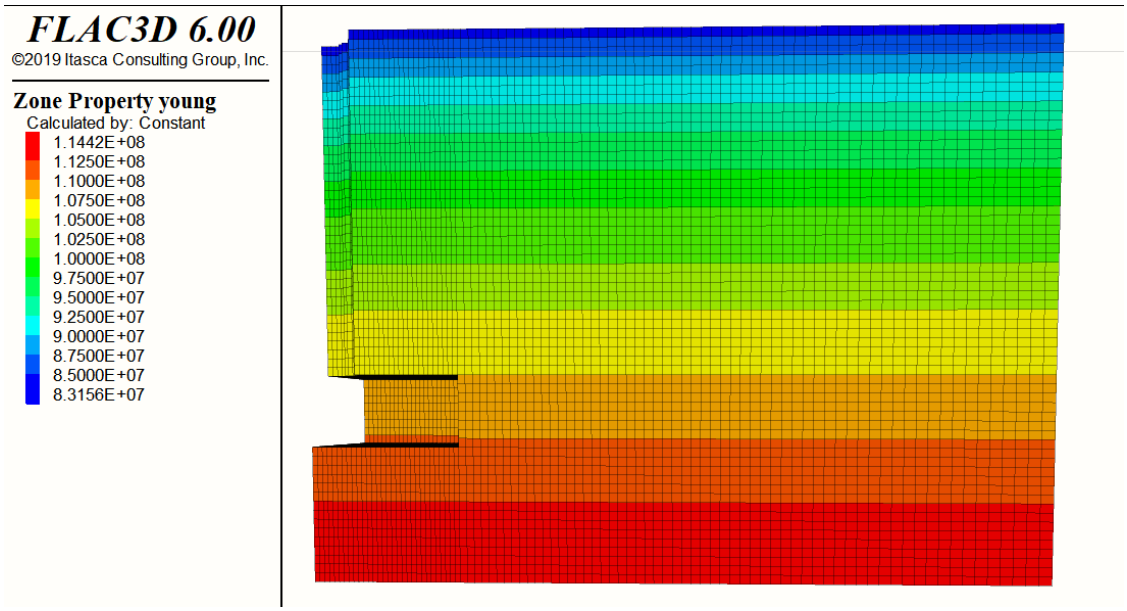
$$E = E_0 + c\sqrt{z}$$

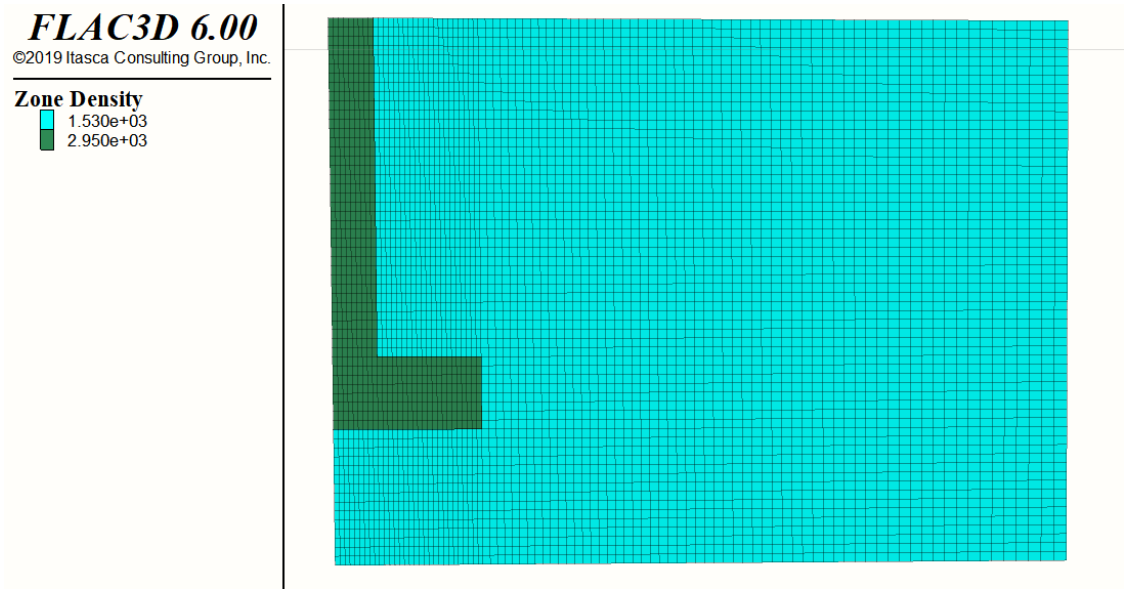
where z is the depth below surface, and c and E_0 are constants. We write a FISH function to install appropriate values of bulk and shear modulus in the grid:

```
########################################
######## CONSTITUTIVE MODEL ############
########################################
```

```
it.command("""
zone cmodel assign elastic range group "Radial Tunnel1"

fish define fname(E_o,const)
loop foreach pnt zone.list
z_depth = zone.pos.y(pnt)
E = E_o+const*math.sqrt(z_depth)
zone.prop(pnt,'young')=E
end_loop
end
@fname({E_o_},{const_})
zone property poisson {poisson_} density {density_soil}
plot item create zone contour property name 'young'
plot item create zone contour property name 'density'
""".format(E_o_=_E_o,const_=_const,poisson_=_poisson,density_soil=_density_soil))

# GROUP #
p = za.pos()
x,y,z = p.T
print(it.zone.count(), "zones in whole model")

shaft = reduce(np.logical_and, (np.sqrt(x**2+z**2)<_D_shaft, y<_H_shaft))
za.set_group(shaft, "shaft")
print(za.in_group("shaft").sum(), "zones in shaft group.")

plate = reduce(np.logical_and, (x<_B_footing,z<_B_footing, y>_H_shaft,y<_D_footing)
za.set_group(plate, "plate")
print(za.in_group("plate").sum(), "zones in plate group.")
```

```
it.command("""
zone cmodel assign elastic range group 'shaft'
zone cmodel assign elastic range group 'plate'
zone property bulk {bulk_} shear {shear_} density {density_} range group 'shaft'
zone property bulk {bulk_} shear {shear_} density {density_} range group 'plate'
""".format(bulk_=_bulk,shear_=_shear,density_=_density_concrete))
```





2.3.1 Elastic (Isotropic) model

2.3.1.1 Hooke's Law

$$\Delta\sigma_{ij} = 2G\Delta\epsilon_{ij} + \alpha_2\Delta\epsilon_{kk}\delta_{ij}$$

, where α_2 is material constant related to the bulk modulus K, and shear modulus G as

$$\alpha_2 = K - \frac{2}{3}G$$

, and new stress values are obtained from the relation

$$\sigma_{ij}^N = \sigma_{ij} + \Delta\sigma_{ij}$$

- Note that bulk modulus K, and shear modulus G, are related to Young's modulus E and Poisson's ratio ν by the following equations:

$$K = \frac{E}{3(1-2\nu)}$$

$$G = \frac{E}{2(1+\nu)}$$

or

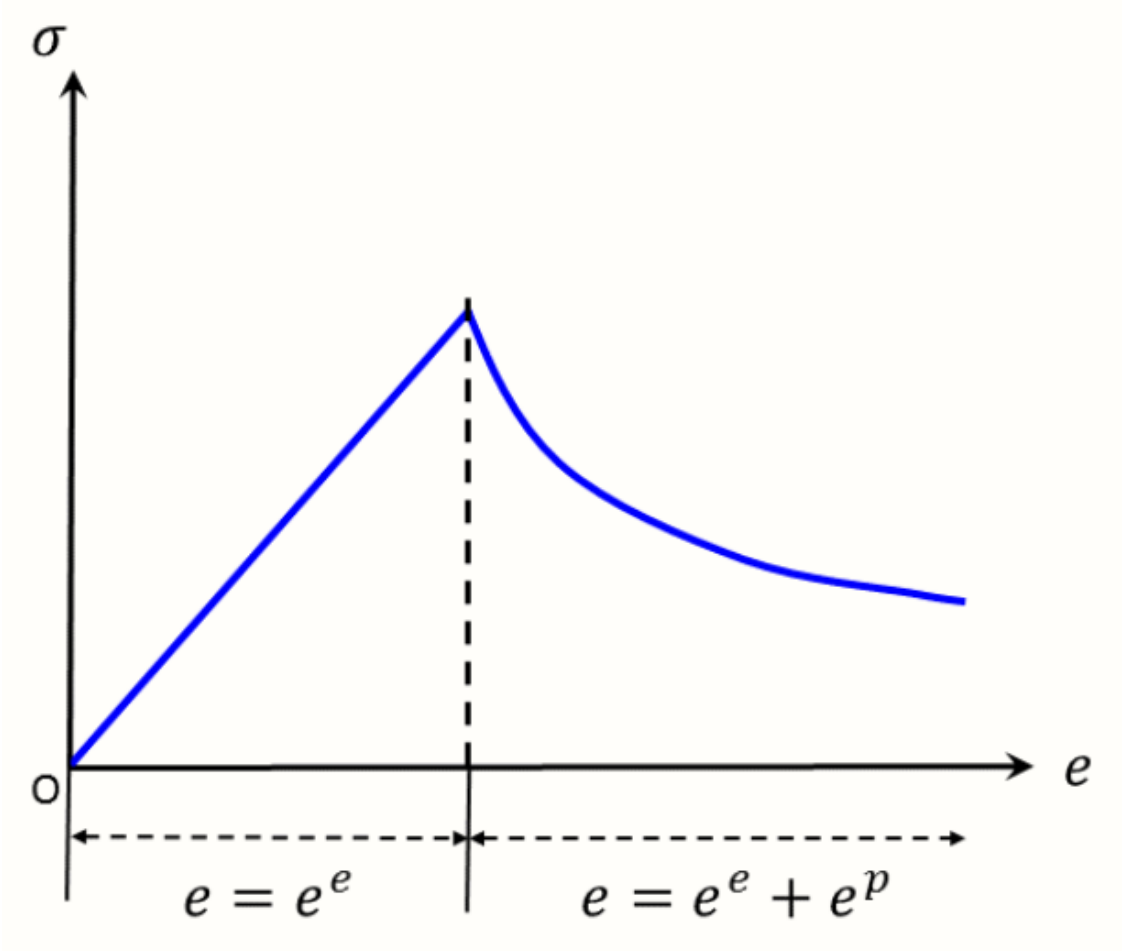
$$E = \frac{9KG}{3K+G}$$

$$\nu = \frac{3K-2G}{2(3K+G)}$$

Strain-softening Model

Strain softening Model is also available, but not implemented:

2. KAIST Model




```
#it.command("""
#zone cmodel assign strain-softening range group "Radial Tunnel1"
#zone property density 2500 bulk 2e8 shear 1e8 range group "Radial Tunnel1"
#zone property cohesion 2e6 friction 45 tension 2e5 dilation 10 range group "Radi
#zone property table-friction 'fri' table-cohesion 'coh' table-dilation 'dil' ran
#table 'fri' add (0, 45) (.05, 42) (.1, 40) (1, 40)
#table 'coh' add (0,2e6) (.05,1e6) (.1,5e5) (1,5e5)
#table 'dil' add (0, 10) (.05,  3) (.1,  0)
#""")
```

The strain-softening/hardening model allows representation of nonlinear material softening and hardening behavior based on prescribed variations of the Mohr-Coulomb model properties (i.e., cohesion, friction, dilation and tensile strength) as functions of the deviatoric plastic strain.

2.3.1.2 Plastic-Strain Increments

The plastic-strain increments involved in the preceding formula may be derived from the definition $\Delta\epsilon_i^p = \lambda\frac{\partial g}{\partial\sigma_i}$ of the flow rule. It has the form:

$$\Delta\epsilon_1^{ps} = \lambda^s$$

$$\Delta\epsilon_3^{ps} = -\lambda^s N_\psi$$

$$\Delta\epsilon_3^{pt} = \lambda^t$$

After determination of the new stresses for the step, the hardening parameters for the zone are updated following the procedure described above. If appropriate, these parameters are then used to determine new values for the zone cohesion, friction, dilation, and tensile strength.

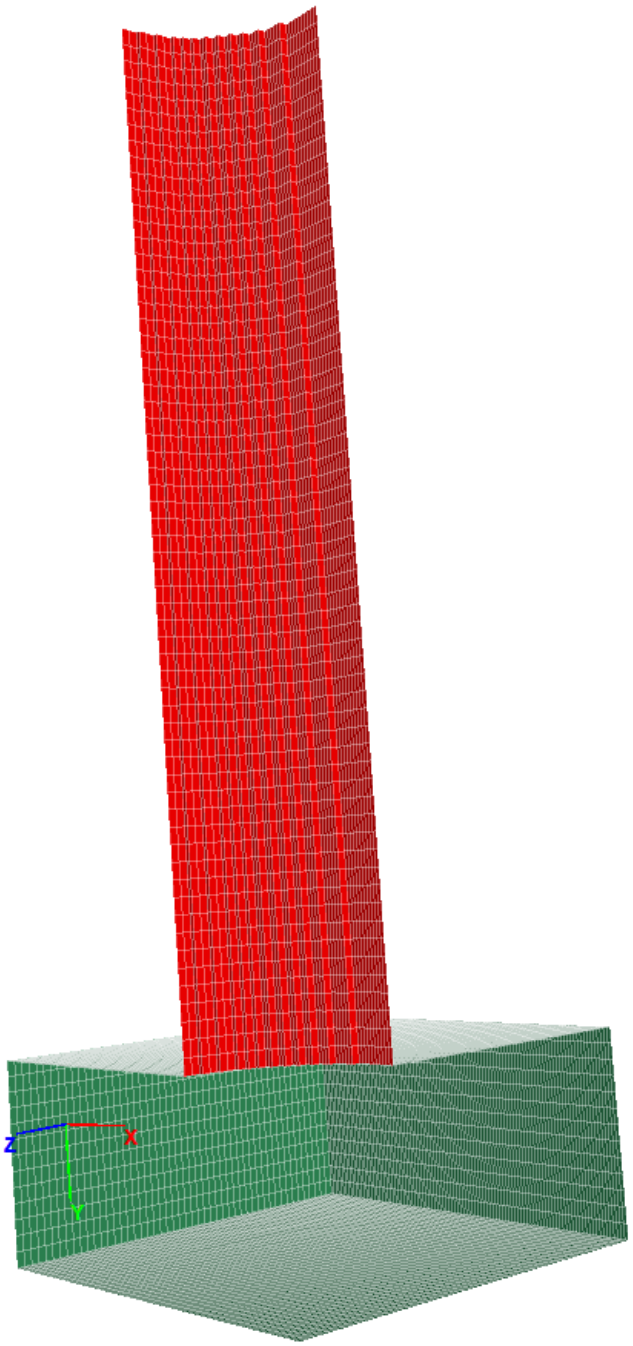
2.3.1.3 strain-softening model properties

- bulk modulus, K
- cohesion, c
- dilation, ψ
- friction, ϕ
- poisson, ν
- shear, G
- tension, σ^t
- young, E
- strain-shear-plastic, accumulated plastic shear strain
- strain-tension-plastic, accumulated plastic tensile strain
- table-cohesion, dilation, friction, tension

Note tension cut-off is $\sigma^t = min(\sigma^t, \frac{c}{tan\phi})$

2.4 Interface

```
########################################
############# INTERFACE ################
########################################

it.command("""
zone interface 'interface 1' create by-face separate range group 'plate' group 'Rac
zone interface 'interface 1' node property stiffness-normal {stiff_norm_} stiffness
zone interface 'interface 2' create by-face separate range group 'shaft' group 'Rac
zone interface 'interface 2' node property stiffness-normal {stiff_norm_} stiffness

""".format(stiff_norm_=_stiff_norm, stiff_shear_=_stiff_shear))

print("               ======== PASSED INTERFACE ========")
```



During each timestep, the absolute normal penetration and the relative shear velocity are calculated for each interface node and its contacting target face. Both of these values are then used by the interface constitutive model to calculate a normal force and a shear-force vector. The constitutive model is defined by a linear Coulomb shear-strength criterion that limits the shear force acting at an interface node, normal and shear stiffnesses, tensile and shear bond strengths, and a dilation

angle that causes an increase in effective normal force on the target face after the shear-strength limit is reached. By default, pore pressure is used in the interface effective stress calculation. This option can be activated/deactivated using the command zone interface effective command by setting effective = on/off. Figure 2 illustrates the components of the constitutive model acting at interface node (P):

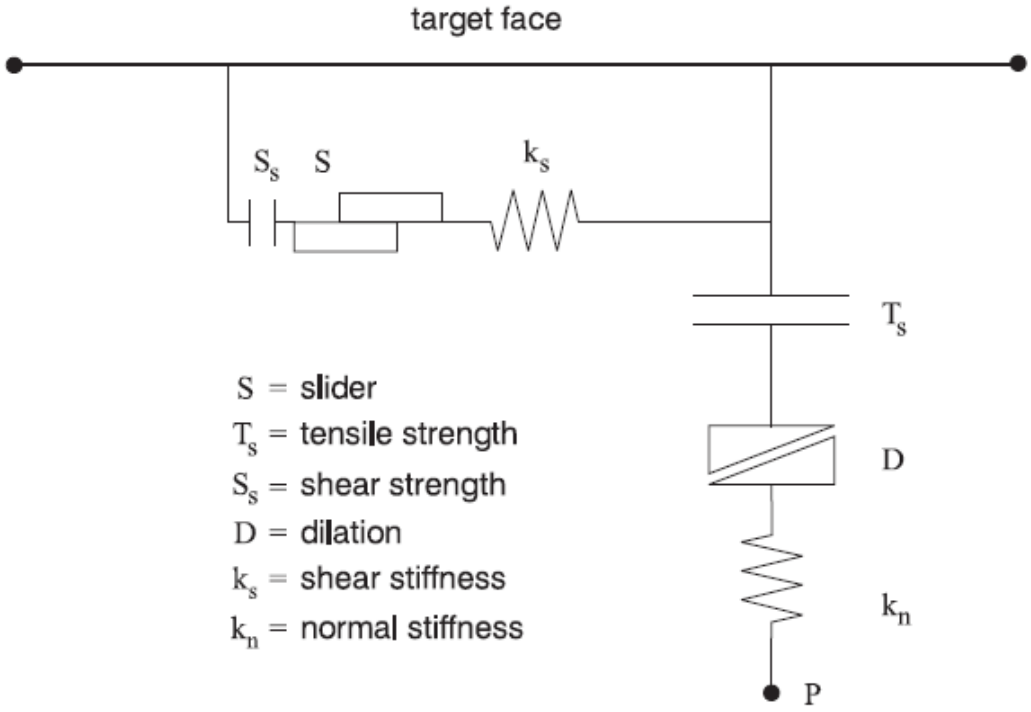


Figure 2: Components of the bonded interface constitutive model.

The normal and shear forces that describe the elastic interface response are determined at calculation time $(t + \Delta t)$ using the relations:

$$F_n^{(t+\Delta t)} = k_n u_n A + \sigma_n A$$

$$F_{si}^{(t+\Delta t)} = F_{si}^{(t)} + k_s \Delta u_{si}^{(t+0.5\Delta t)} A + \sigma_{si} A$$

The inelastic interface logic works in the following way:

1. Bonded interface — The interface remains elastic if stresses remain below the bond strengths; there is a shear bond strength, as well as a tensile bond strength. The normal bond strength is set using the tension interface property

keyword. The command zone interface node property shear-bond-ratio = sbr sets the shear bond strength to sbr times the normal bond strength. The default value of property shear-bond-ratio (if not given) is 100.0. The bond breaks if either the shear stress exceeds the shear strength, or the tensile effective normal stress exceeds the normal strength. Note that giving property shear-bond-ratio alone does not cause a bond to be established—the tensile bond strength must also be set.

2. Slip while bonded — An intact bond, by default, prevents all yield behavior (slip and separation). There is an optional property switch (bonded-slip) that causes only separation to be prevented if the bond is intact (but allows shear yield, under the control of the friction and cohesion parameters, using (F_n) as the normal force). The command to allow/disallow slip for a bonded interface segment is zone interface node and by setting bonded-slip on or off.

The default state of bonded-slip (if not given) is off.

3. Coulomb sliding — A bond is either intact or broken. If it is broken, then the behavior of the interface segment is determined by the friction and cohesion (and of course the stiffnesses). This is the default behavior, if bond strengths are not set (zero). A broken bond segment cannot take effective tension (which may occur under compressive normal force, if the pore pressure is greater). The shear force is zero (for a nonbonded segment) if the effective normal force is tensile or zero.

The Coulomb shear-strength criterion limits the shear force by the relation:

$$F_{smax} = cA + tan\phi(F_n - pA)$$

During sliding, shear displacement may cause an increase in the effective normal stress on the joint, according to the relation:

$$\sigma_n := \sigma_n + \frac{|F_s|_o - F_{smax}}{Ak_s} tan\psi k_n$$

On printout (see the zone interface node list command) the value of tension denotes whether a bond is intact or broken (or not set) — nonzero or zero, respectively.

The normal and shear forces calculated at the interface nodes are distributed in equal and opposite directions to both the target face and the face to which the interface node is connected (the host face). Weighting functions are used to distribute the forces to the gridpoints on each face. The interface stiffnesses are added to the accumulated stiffnesses at gridpoints on both sides of the interface in order to maintain numerical stability.

Interface contacts are detected only at interface nodes, and contact forces are transferred only at interface nodes. The stress state associated with a node is assumed to be uniformly distributed over the entire representative area of the node. Interface properties are associated with each node; properties may vary from node to node.

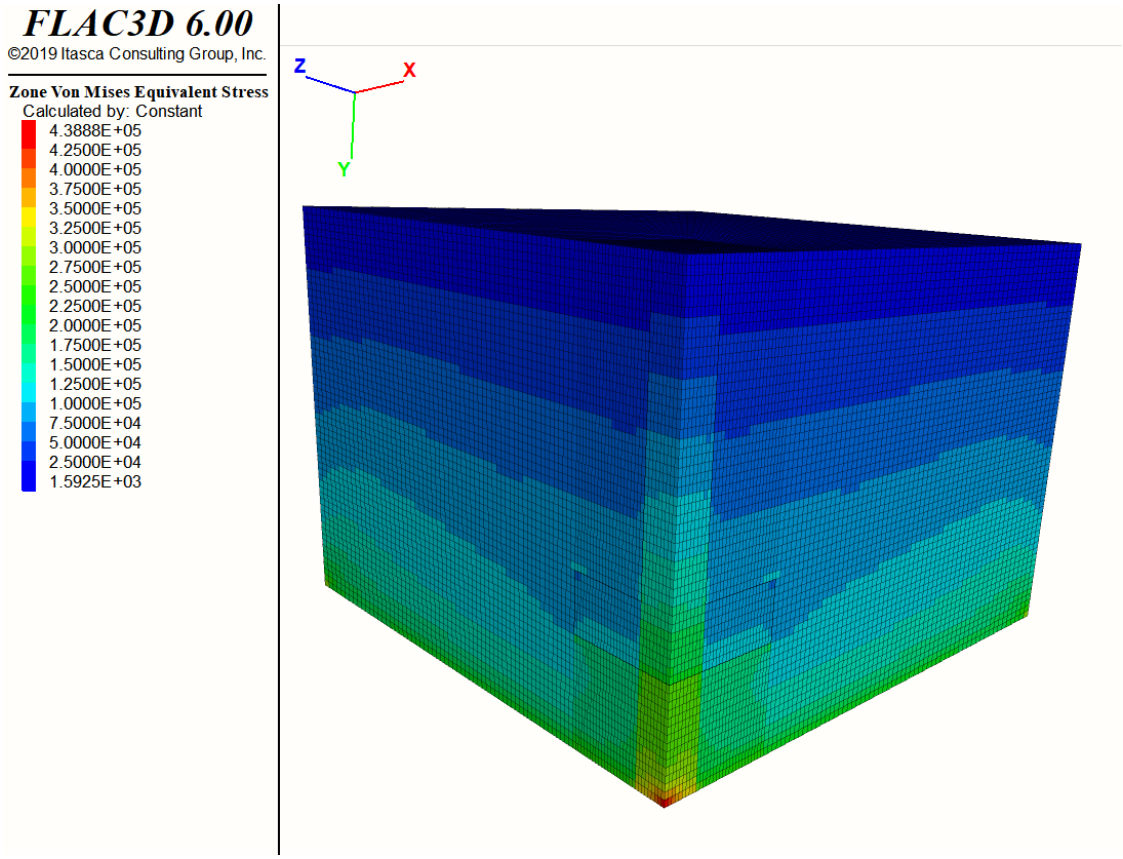
2.5 Boundary Conditions

```
########################################
######## BOUNDARY CONDITIONS ###########
########################################

it.command("""
zone face skin
zone face apply velocity-normal 0 ...
```

```
range group 'West7' or 'West6' or 'Bottom7' or 'Bottom6' or 'East6' or 'Top6'

zone face apply velocity (0,0,0) range group 'North3'
""")
```

2.6 Initial Equilibrium

```
########################################
############# INITILIZE ################
########################################

it.command("""
model gravity 0 {gravity_} 0
zone initialize-stress ratio {K_}
zone interface 'interface 1' node initialize-stresses
zone interface 'interface 2' node initialize-stresses
""".format(gravity_=_gravity, K_ = _K))

it.command("""
zone ratio local
model solve ratio 1e-4
model save 'initial'
""")

print("               ======== PASSED INITIAL EQUILIBRIUM ========")
```



2.7 Vertical Loading

```
########################################
########## VERTICAL LOADING ############
########################################


#top = reduce(np.logical_and, (np.sqrt(x**2+z**2)<_D_shaft, y<=np.amin(y)))
#za.set_group(top, "top")
#print(za.in_group("top").sum(), "zones in top group.")


it.command("""
zone initialize state 0
zone gridpoint initialize displacement (0,0,0)
zone gridpoint initialize velocity     (0,0,0)
```

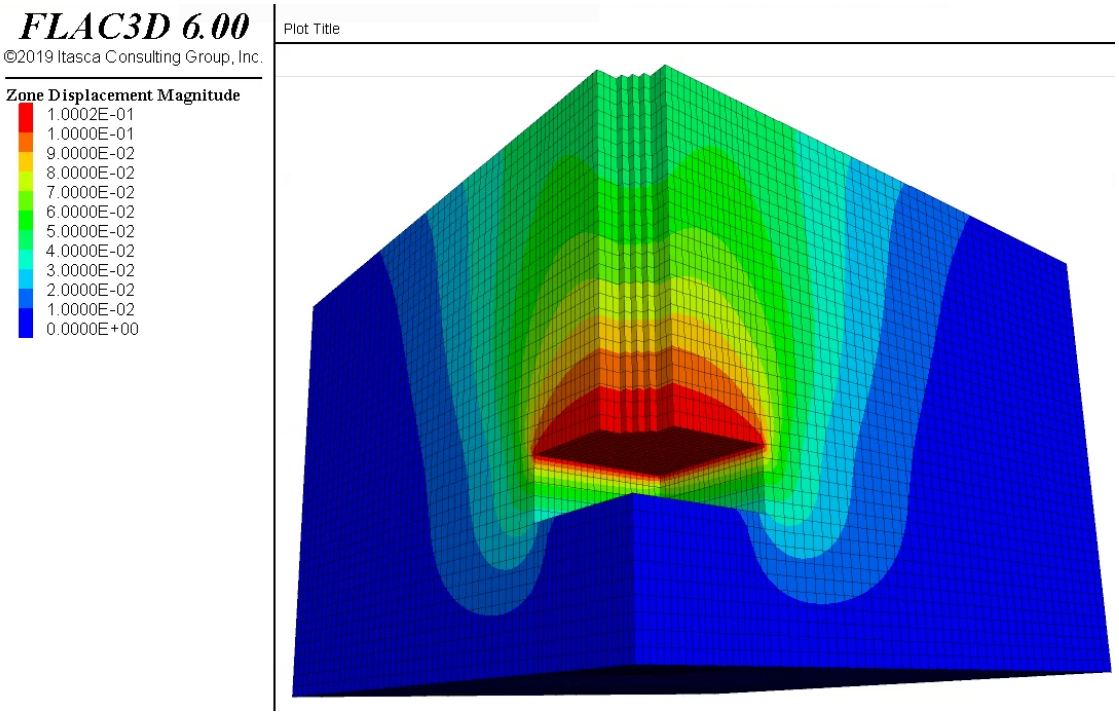
```
table 'ramp' add ([global.step],0) ([global.step+5000],1e-6) ...
      ([global.step+10000],1e-6) ; Increase velocity applied to pile
                                 ; over 5,000 steps

zone face apply velocity-normal 1e-5 range group 'North1' or 'South2'

; Set up histories for monitoring model behavior
history interval 10
zone history name 'disp' displacement-y position (3,7,3)

; find gridpoints at pile cap, store in map called cap
;fish define find_cap
;    global cap = map
;    loop foreach local gp gp.list
;        if gp.isgroup(gp,'top') then
;            cap(gp.id(gp)) = gp
;        endif
;    endloop
;end
;fish history name 'cap' @find_cap
;
;fish define vert_load
;    local yftot = 0.0
;    loop foreach gp cap
;        yftot = yftot + gp.force.unbal.y(gp)
;    end_loop
;    vert_load = yftot / (0.25*{D_shaft_}*{D_shaft_}*math.pi)
;end
;
```

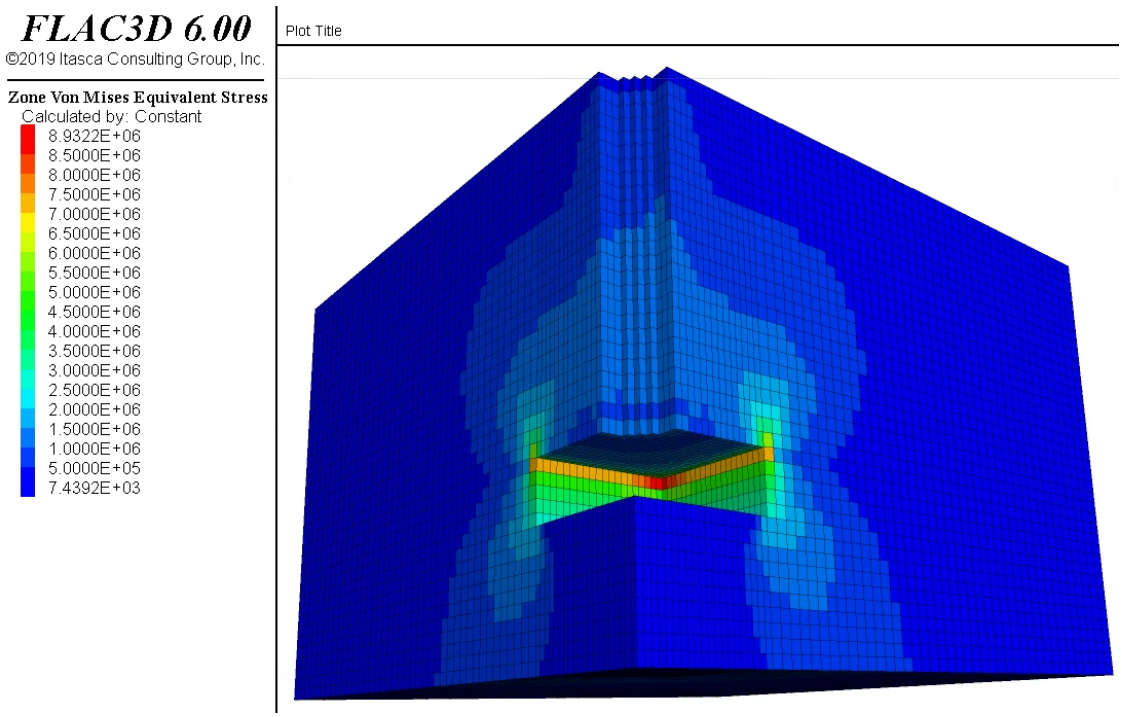
```
;fish history name 'load' @vert_load
;zone mechanical damping combined
model largestrain true
model step 10000
model save 'vertical-loading'
""")

print("               ======== PASSED VERTICAL LOADING ========")
```

2.8 Plot

```
########################################
################ PLOT ##################
########################################

it.command("""
plot create
plot item create zone label group
plot create
plot item create zone contour property name 'young'
plot create
plot item create zone label density
plot create
plot item create zone active on ...
    contour Displacement
""")

print("               ======== PASSED PLOT ========")
```

2.9 Results


FLAC3D 6.00
©2019 Itasca Consulting Group, Inc.
Zone Displacement Magnitude
1.0002E-01
1.0000E-01
9.0000E-02
8.0000E-02
7.0000E-02
6.0000E-02
5.0000E-02
4.0000E-02
3.0000E-02
2.0000E-02
1.0000E-02
0.0000E+00
Plot Title

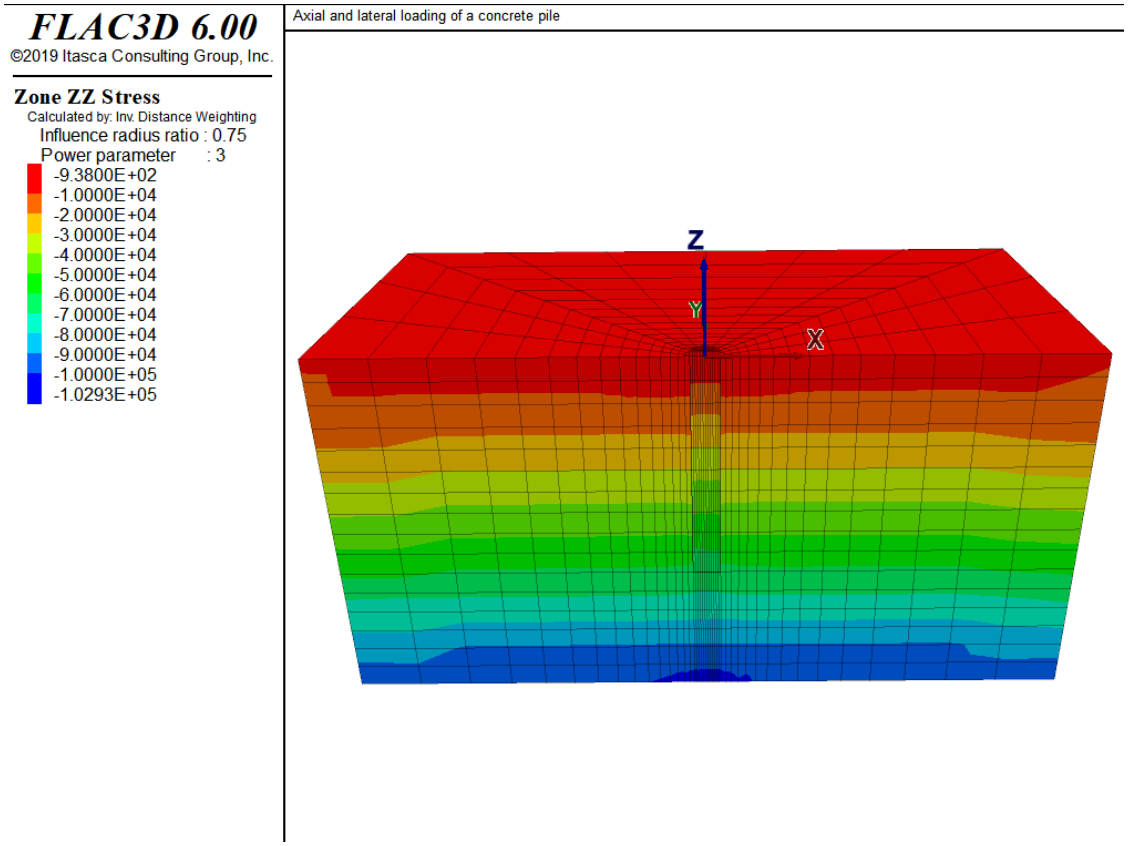

FLAC3D 6.00
©2019 Itasca Consulting Group, Inc.
Zone Von Mises Equivalent Stress
Calculated by: Constant
8.9322E+06
8.5000E+06
8.0000E+06
7.5000E+06
7.0000E+06
6.5000E+06
6.0000E+06
5.5000E+06
5.0000E+06
4.5000E+06
4.0000E+06
3.5000E+06
3.0000E+06
2.5000E+06
2.0000E+06
1.5000E+06
1.0000E+06
5.0000E+05
7.4392E+03
Plot Title

My modelling choices
Are founded in logic.
Can I convince you?
If you look down,
Are you surprised to see
Only your feet?
The train keeps to the tracks
But the country around
Remains unexplored.
The excellent map
misses no detail, but it is
as large as the world.

Muir Wood(2004)

3

Axial Concrete Pile



3.1 Problem Description

3.1.1 Problem Statement

The pile is subjected to an axial load of 100 kN, and then the top of the pile is moved horizontally for a displacement of 4 cm.

The goal is to determine relation of axial loading to the ultimate bearing capacity. And, lateral load-deflection curve is calculated.

1) origin at the top of the pile, z upward.
2) z=0: free surface
3) z=-8: fixed in z-eirection
4) x=+8, -8, y = 8: roller
5) skin friction is modeled by placing an interface between pile concrete wall and clay. In it, fric angle of 20 and c=30kPa are assumed.
6) toe interface is placed between pile tip and clay *note: Zone faces are separated in a previous command so that the gridpoints common to both will be separated as well.*
   note: include Figure of grid (geometry)

3.1.2 Main Parameters

Diameter = 0.6 m

Length = 5

Clay

GWT = 5.5m

3.2 Modeling Procedure

1) equil. stress state under gravity load before install.
   1-1) water table is created at z=5.5

1-2) wet density of clay is assigned below this water table.

2) equil. stress state after installation.

2-1) change properties of pile zones from those

representing clay to those representing concrete.

2-2) vertical equil. stress distribution at this equil. state is shown in

note: include Figure of contours of vertical stress at ini state incld. pile weight

3) apply vertical velocity at top of pile

"ramp" = boundary condition is increased linearly

note: critical timestep is controlled by high stiffness of concrete

If velocity is sudden, inertial effects will dominate and renders difficulty to identification of steady state response of system

table "ramp" is used to apply velocity to pile top gridpoints.

note: FISH FUNCTION vert_load calculates axial stress at the top of pile and stores value as a history

For efficiency, gridpoints on cap surface are stored in symbol "cap" as a map

note: include plot of axial stress vs axial displ. at pile toe. ramp = (0,5e-8), step number = 30000

note: combined damping is used to remove kinetic energy for prescribed loading condition. This is because mass-adjustment process depends on velocity sign-changes..

note: FISH FUNCTION tot_reac monitors soil reaction along pile as a func of lateral displ. tot_reac creates tables of soil reaction (p) vs. lateral displ (y) at diff. locations along pile to generate p-y curve.

note: include Figure of p-y curve at 11 equidistant points along pile

3.3 Zones

```
model new
model title 'Axial and lateral loading of a concrete pile'
; create grid interactively from the extruder tool,
; exported to geometry.f3dat from State Record pane.
call 'geometry' suppress
zone generate from-extruder
; Reflect the grid to get a 1/2 space instead of a 1/4 space
zone reflect dip-direction 270 dip 90
```

3.4 Groups

```
; Name intersections of things named in the two extruder views
zone group 'clay' range group 'clay-c' or 'clay-s' or 'wetclay-s'
zone group 'pile' range group 'pile-c' group 'pile-s' or 'remove-s'
zone group 'remove' range group 'remove-s' group 'pile-c' not ;
zone face group 'wall' internal range group 'wall-c' group 'pile'
zone face group 'base' internal range group 'base-s' group 'pile'
zone face skin ; Name far field boundaries
; Delete the area marked for removal
zone delete range group 'remove'
;
; setup interfaces
; separate using zone separate
```

```
; all at once so common nodes are separated
zone separate by-face new-side group 'iwall' slot 'int' ...
     range group 'wall' or 'base'
; Want two different interfaces for proper normal direction at corner
zone interface 'side' create by-face range group 'wall' and 'iwall'
zone interface 'base' create by-face range group 'base' and 'iwall'
; Save initial geometric state
model save 'geometry'
```

3.5 Properties

```
; Initialize gravity, pore-pressures, density, and stres state
model gravity 10
; water table information
zone water density 1000
zone water plane origin (0,0,-5.5) normal (0,0,-1)
zone initialize density 1230
zone initialize density 1550 range group 'wetclay-s' ; Wet density
; assign properties to the soil and interfaces - temporarily remove pile cap
zone cmodel assign mohr-coulomb ...
     range group 'clay'
zone property bulk 8.333e7  shear 3.846e7  cohesion 30000 fric 0 ...
     range group 'clay'
zone cmodel assign elastic                 range group 'pile'
zone property bulk 8.333e7  shear 3.846e7  range group 'pile'
zone cmodel assign null                    range group 'remove-s'
zone interface 'side' node property stiffness-normal 1e8 ...
                           stiffness-shear 1e8 friction 20 cohesion 30000
```

```
zone interface 'base' node property stiffness-normal 1e8 ...
                              stiffness-shear 1e8 friction 20 cohesion 30000
```

3.6 B.C. and I.C.

```
; boundary and initial stress conditions
zone face apply velocity-normal 0 range group 'Bottom'
zone face apply velocity-normal 0 range group 'East' or 'West'
zone face apply velocity-normal 0 range group 'North' or 'South'
zone initialize-stress ratio 0.4286
zone interface 'side' node initialize-stresses
zone interface 'base' node initialize-stresses
```

3.7 Initial Equilibrium

```
; Solve to initial equilibrium
zone ratio local
model solve ratio 1e-4
model save 'initial'
```

3.8 Alterations

3.8.1 install the pile

```
; install the pile
model restore 'initial'
zone cmodel assign elastic                              range group 'pile'
zone property bulk 13.9e9 shear 10.4e9 density 2500 range group 'pile'
```

```
model solve ratio 1e-4
model save 'install'
```

3.8.2 vertical loading

```
; vertical loading
zone initialize state 0
zone gridpoint initialize displacement (0,0,0)
zone gridpoint initialize velocity     (0,0,0)
table 'ramp' add ([global.step],0) ([global.step+30000],-5e-8) ...
      ([global.step+58000],-5e-8) ; Increase velocity applied to pile
                                  ; over 30,000 steps
zone face apply velocity-normal 1 table 'ramp' range group 'Top'
history interval 250
zone history name 'disp' displacement-z position (0,0,0)
call 'load'
fish history name 'load' @vert_load
zone mechanical damping combined
model step 58000
model save 'vertical-loading'
```

3.8.3 vertical then lateral loading

```
; vertical loading then lateral loading
model restore 'install'
zone initialize state 0
zone gridpoint initialize displacement (0,0,0)
zone gridpoint initialize velocity     (0,0,0)
zone face apply stress-zz [-1.0e5/(math.pi*0.3*0.3)] range group 'Top'
model solve ratio 1e-4
```

```
model save 'lateral-load-start'


; apply lateral loading as x-velocity on cap
zone initialize state 0
zone gridpoint initialize displacement (0,0,0)
zone gridpoint initialize velocity     (0,0,0)
zone face apply velocity-x 1e-7 range group 'Top'
zone history name 'disp' displacement-x position 0,0,0
call 'p-y' suppress ; Calculates p-y curve for pile, when tot_reac is called
@make_pydata        ; Generate p-y curve calculation data
@output_structure   ; Sanity check of p-y curve data
fish history name 'load' @tot_reac
model step 416500
model save 'lateral-load'
```

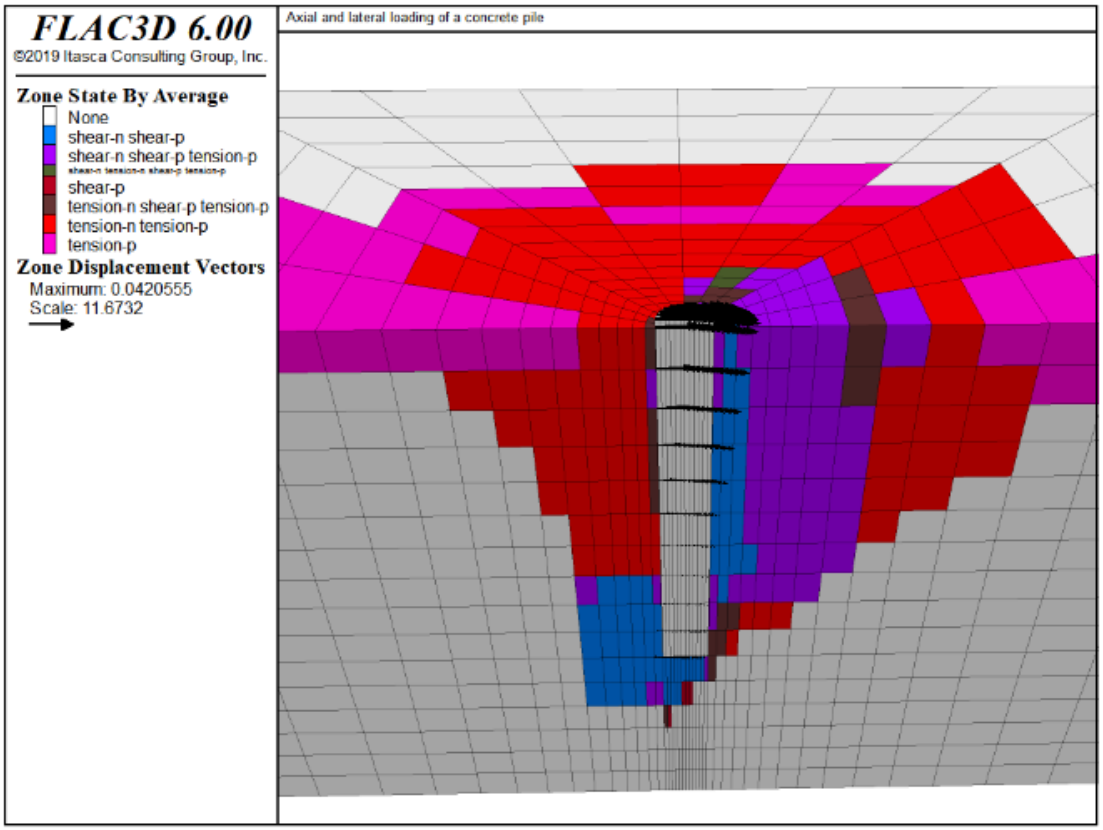


Figure 5: Pile displacement and plasticity state of soil at 4 cm lateral deflection.

In spite of their obvious deficiencies and unreliability, pile driving formulas still enjoy great popularity among practicing engineers, because the use of these formulas reduces the design of pile foundations to a very simple procedure. The price one pays for this artificial simplification is very high.

Karl Terzaghi(1943)

4

Pull-Tests

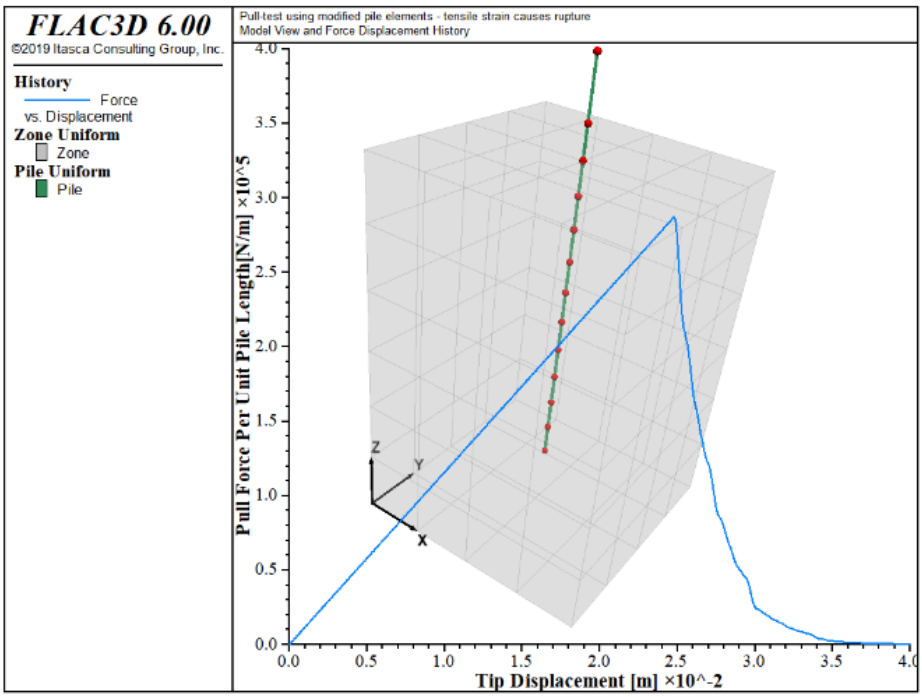


Figure 14: Rockbolt pull force in N/m versus rockbolt axial displacement in meters for the case of a single 25 mm grouted bolt—with tensile rupture.

4.1 Problem Description

note: FISH function force is used to sum the reaction forces and monitor nodal displacement generated by the pull-test

note: free length of bolt that extends out of block + larger diameter
Perfectly plastic behavior of grout = max cohesion is exceeded +post-peak weakening of shear bond strength

note: bond strength softening of the grout is defined with keyword coupling-cohesion-table (see Rockbolt Behavior)

The relation btw shear disp. and cohesion weakening is prescribed thru table cct. softening of friction of grout canbe defined using keyword coupling-friction-table.

4.2 Zones

```
; ================================================================
;   Simulation of pull-test for grouted reinforcement
;   using modified pile elements - Softening of cohesion
; ================================================================
model new
fish automatic-create off
model title 'Pull-test using modified pile elements - cohesion softening'
; Create a single rock block and set its material properties.
zone create brick size 4 4 6 point 1 (0.4,0,0) point 2 (0,0.4,0) ...
                             point 3 (0,0,0.6)
```

4.3 Properties

```
zone cmodel assign elastic
zone property bulk 5e9 shear 3e9
zone face apply velocity-normal 0.0 range position-z 0.6
; Create a pile element and assign properties
struct pile create by-line (0.2,0.2,0.1) (0.2,0.2,0.7) segments 12
struct pile property rockbolt-flag on
struct pile property young 200e9 poisson 0.25 cross-sectional-area 5e-4 ...
                     perimeter 0.08
struct pile property tensile-yield   2.25e5  ; ultimate tensile strength
struct pile property moi-y 2.0e-8 moi-z 2.0e-8 moi-polar 4.0e-8 ; 0.25*pi*r^4
struct pile property coupling-cohesion-shear 1.75e5 ...
                     coupling-stiffness-shear 1.12e7
struct pile property coupling-cohesion-normal 1.75e5 ...
                     coupling-stiffness-normal 1.12e7
; rel btw shear displ and coh weakening is prescribed thru table cct
struct pile property coupling-cohesion-table 'cct'
; change in cohesion with relative shear displacement
table 'cct' add (0,1.75e5) (0.025,1.75e4)
```

4.4 Initial Equilibrium

```
struct node fix velocity-x range position-z 0.7
struct node initialize velocity-x 1e-6 local range position-z 0.7
call 'pileforce' suppress ; FISH function calculates reaction force on zones
```

4.5 Alterations

```
; Set up histories for monitoring model behavior
history interval 10
fish history name 'force' @force
struct node history name 'disp' displacement-z position (0.2,0.2,0.7)
; Achieve a total displacement of 4.0 cm
model cycle 40000
;
model save 'pull-5'
```

4.6 Some other notes

2.3. pull test with confinement “Pulltest06.f3dat” +modified pile logic.(see Behavior of Shear Coupling Springs) linear law is implemented.whereby reinforcement shear strength is defined as constant

Confining stress of 4 MN/m2 = 4MPa

As described in Behavior of Shear Coupling Springs a linear law is implemented in the modified pile logic, whereby the reinforcement shear strength is defined as a constant (coupling-cohesion-shear) plus the effective pressure on the reinforcement multiplied by the reinforcement perimeter (perimeter) times a friction angle (coupling-friction-shear). This pressure dependence is activated automatically in FLAC3D by issuing the reinforcement properties (perimeter) and (coupling-friction-shear).

*(coupling-cohesion-shear)+ effective pressure*perimeter*fric angle(coupling-friction-shear)*

This pressure dependence is activated automatically by issuing reinforcement

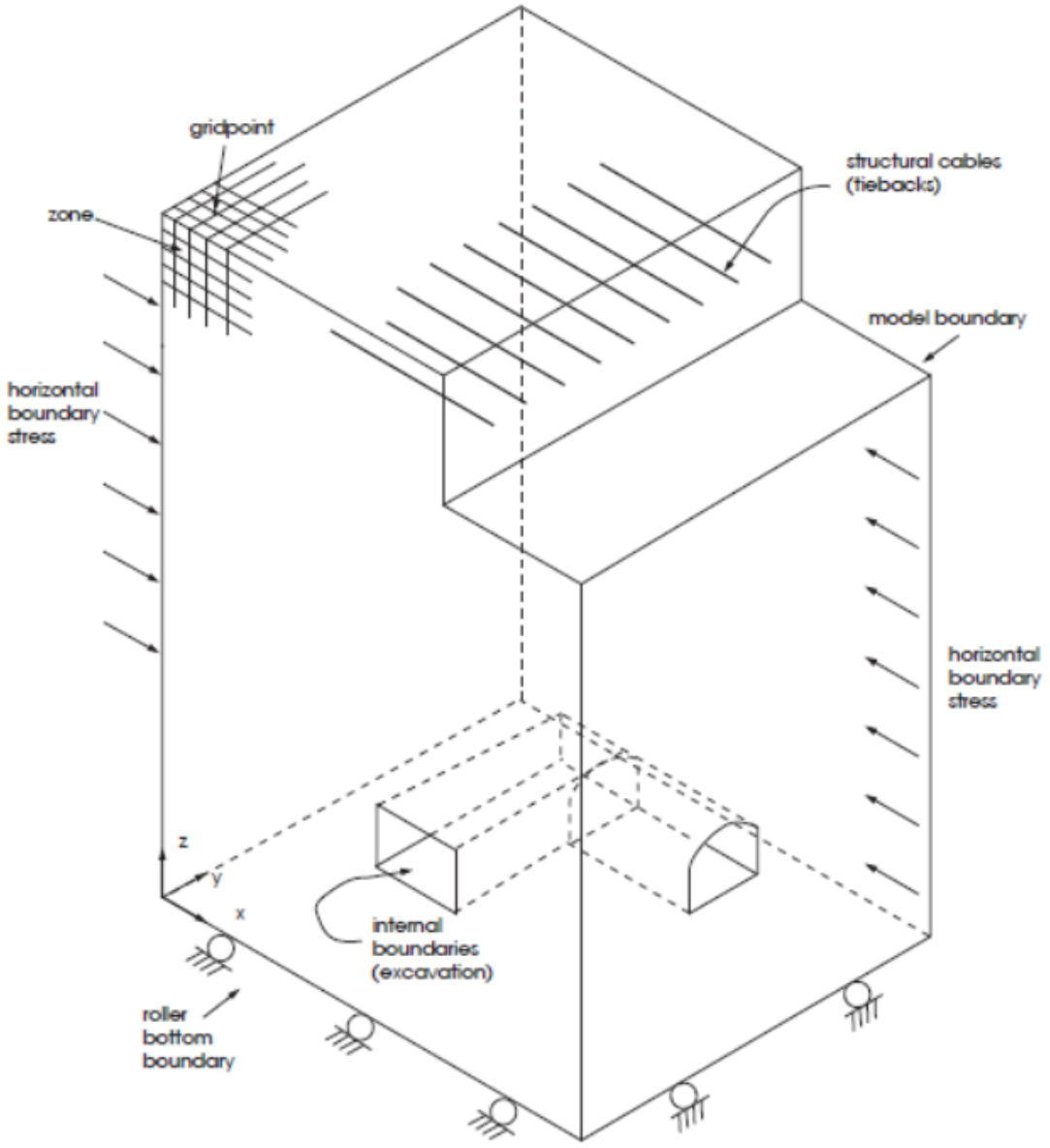
properties(perimeter) and

2.5. pull test with tensile rupture “Pulltest08.f3dat” *note: tensile-yield, tensile-failure-strain: for limiting axial yield force and limiting axial strain for rockbolt*

5

Grid

5.1 Primitive Shapes



note: zone create generates primitive grid note: zone gridpoint create puts gridpoints at specific locations note: zone gridpoint merge ensures separate primitives are connected properly note: zone attach connects primitive meshes of different zone sizes.

```
zone create radial-cylinder size 5 10 6 12 fill
zone create radial-cylinder size 5 10 6 12 ratio 1 1 1 5
each size is controlled by a ratio (geometric ratio of 1.2 times preceding zone)
```

ex) 5 along inner radius of cylindrical tunnel,
10 along axis
6 along circumference of tunnel
12 between periphery of tunnel and outer boundary of model
note: size keyword defines the number of zones in the grid.

keywords for zone create:

- dimension - edge - fill - point (boundary dimensions) - ratio (coarser toward edge) - size

5.2 several primitive shapes connected:

```
zone create radial-cylinder size 5 10 6 12 rat 1 1 1 1.2 ...
                            point 0 (0,0,0) point 1 (100,0,0) ...
                            point 2 (0,200,0) point 3 (0,0,100)
zone create radial-tunnel size 5 10 5 12 rat 1 1 1 1.2 ...
                           point 0 (0,0,0) point 1 (0,0,-100) ...
                           point 2 (0,200,0) point 3 (100,0,0)
; here, model boundary dimensions are 100, 200, 100
; boundary coord are defined using point keyword

zone reflect dip 90 dip-direction 270 origin (0,0,0)
```

This adds symmetric part.

note: The symmetry plane is a vertical plane (located by the dip,dip-direction, and origin keywords) coincident with the $x = 0$ plane. Note that dip angle (dip) and dip direction (dip-direction) assume thatx corresponds to "East," y to "North", and z to "Up."

third option, the zone gridpoint create command, is available to position single points in the model region.

> *note: zone gridpoint create is used for positioning reference points ofprimitives*

During execution of a zone create command, a check is made for each boundary gridpoint against the boundary gridpoints of zones that already exist.

If two boundary gridpoints fall within a tolerance of 1 10-7 (relative to the magnitude of the gridpoints position vector) of each other, they are assumed to be the same point,

If it is discovered that some gridpoints don't match, the zone gridpoint merge command can be used to merge these gridpoints after the zone create command has been applied.

Example: (zone attach) - Two unequal sub-grids

```
zone create brick size 4 4 2 point 0 (0,0,0) point 1 (4,0,0) ...
                             point 2 (0,4,0) point 3 (0,0,2)
zone create brick size 8 8 4 point 0 (0,0,2) point 1 (4,0,2) ...
                             point 2 (0,4,2) point 3 (0,0,4)
zone attach by-face range position-z 2
```

Example: (zone densify)

```
zone create brick size 4 4 4
zone densify segments 2 range position-x 2 4
```

the first two command lines can be changed to where zone densify segments 2 refines the upper zones (between the z-coordinate of 2 and 4) with the segment number of 2 on each edge.

5.3 Structural Element Operation

Creating a liner in the service tunnel

```
; liner
structure shell create by-face range cylinder ...
                                        end-1 (0,0,-1) end-2 (0,50,-1) ...
                                        radius 3
```

The liner contains 240 structural shell elements and is connected to the FLAC3D grid at 143 structural-node links. The grid with the liner is shown below.

5.4 Densifying grid by specifying max size length

```
model new
zone create brick size 4 4 4
plot 'Brick' export bitmap filename 'densify3.png'
;
zone densify local maximum-length (0.5,0.5,0.4) range position-z 2 4
zone attach by-face
;
plot 'Brick' export bitmap filename 'densify4.png'
```

> *note that in the local z-direction, the maximum size length is 0.4. FLAC3D densifies the maximum length in this direction to be 1/3 (= 0.4)*

The zone attach by-face command in this example is used to attach faces of sub-grids together rigidly to form a single grid.

Always use the zone attach by-face command after the zone densify command if there are different numbers of gridpoints along faces of different zones.

5.4.1 Densify a grid using geometric information

```
model new
zone create brick size 10 10 10
;
geometry set "setA" polygon create ...
                    by-positions (0,0,1) ( 5,0, 1) ( 5,10, 1) (0,10,1)
geometry set "setA" polygon create ...
                    by-positions (5,0,1) (10,0, 5) (10,10, 5) (5,10,1)
geometry set "setB" polygon create ...
                    by-positions (0,0,5) ( 5,0, 5) ( 5,10, 5) (0,10,5)
geometry set "setB" polygon create ...
                    by-positions (5,0,5) (10,0,10) (10,10,10) (5,10,5)
plot 'Brick2' export bitmap filename 'densify5.png'

zone densify segments 2 range geometry-space "setA" set "setB" count 1
zone attach by-face
;
plot 'Brick2' export bitmap filename 'densify6.png'
```

6

Syntax and Grammar

6.1 Introduction

```
import itasca as it
it.command("python-reset-state false")
it.command("""
model new
zone create brick size 10 10 10
zone cmodel assign elastic
zone property density 2950 young 12e9 poisson 0.25
cycle 1
""")
it.zone.count()
z=it.zone.find(1)
print z
z.pos()
```

```
volume_sum = 0.0
for z in it.zone.list():
  volume_sum += z.vol()

print volume_sum
print z.vol() * it.zone.count()
assert volume_sum == z.vol() * it.zone.count()

z = it.zone.near ((5,5,5))
z.pos()
```

6.2 Zones

```
it.zone.count() # 1000
z = it.zone.find(1)
for z in it.zone.list():
  z = it.zone.near((5,5,5))
z.pos()
z.vol()
```

6.3 Properties

```
z.props() or z.props()['bulk']
z.prop('shear')
z.set_prop('bulk', 8.5e9)
```

6.4 Gridpoints

```
gp = it.gridpoint.near((2,2,2))
for gp in it.gridpoint.list():
  total_mass = gp.mass_gravity()
z.vol()*z.density()*1000
```

6.5 Structural Elements

```
it.structure.list()
it.structure.find(1)
it.structure.near((0,2,2))
it.structure.node.find(1)
s_node.links()[0]
```

6.6 Extra Variables

```
z.set_extra(1, 1.23)
z.set_extra(2, "a test string")
z.set_extra(1, gp.pos())
```

6.7 Groups and B.C.

```
if z.group("default") == "lower":
  gp.set_fix(0, True)
  gp.set_fix(1, True)
  gp.set_force_load((1e6,2e6,1e6))
it.zone.near((5,5,5)).stress()
```

```
it.zone.near((5,5,5)).strain()
"""
```

6.8 Parameteric Studies

```
"*note: for modulus in [6e9, 8e9, 10e9, 12e9]:"
it.command("""
model restore 'before_cycling'
zone prop young {}
model solve
""".format(modulus))
vertical_disp = it.gridpoint.near((5,5,10)).disp_z()
print "~~~".format(modulus,vertical_disp)
```

6.9 Results

As discussed in Reaching Equilibrium, we recommend you use the convergence 1.0 limit criteria when using the model solve command

Initially solving to a convergence of 100 or more and then examining the response is good practice. Making isosurface plots of convergence to see where the model is having trouble can give insight into where model adjustments in the interest of efficiency are best placed.

In order to determine a collapse load, it is often better to use "strain-controlled" rather than "stress-controlled" boundary conditions (i.e., apply a constant velocity and measure the reaction forces, rather than apply forces and measure displacements).

```
; Solve
model solve convergence 1
model factor-of-safety file 'slope3dfos' associated convergence 1
```

6.9.1 Measuring time of calculation

```
[global start = time.clock]
model step 100
[global rate = 10 * zone.num / (time.clock-start) ]
[io.out('Calculation rate = '+string(rate)+' kilo-zones/sec')]
```

6.10 Setting FISH variables

```
import itasca as it
it.command('python-reset-state false')
it.fish.set('x', 10)
x = it.fish.get('x') yields 10
```

6.10.1 Issuing Command

```
import itasca as it
import numpy as np
data = np.loadtext('brick-data.txt')
command_template = ;;;
zone create brick
zone cmodel assign elastic
zone property density {density} young {young} poisson {poisson}
;;;
density = data[0]
young = data[1]
poisson = data[2]

command = command_template.format(density=density, young=young, poisson=poisson)
it.command(command)
```

6.11 String

```
"The value of x is {:.2f}".format(0.3872)
"The value of x is {:.2e}".format(0.3872)
"My name is Sasha"
"My name is {}".format("Sasha")
"My name is {name}".format(name="Sasha")
```

6.12 Fish Syntax

6.12.1 Use of (...)

A complete FISH statement occupies one line. However, a line may be typed across two or more lines as long as each line but the ultimate is terminated with the continuation character (...).

Use of temporary variables as hinge points to concatenate lengthy formulas can also be handy. The following example shows how this can be done:

```
fish define long_sum  ;example of a sum of many things
    local temp = v1 + v2 + v3 + v4 + v5 + v6 + v7 + v8 + v9 + v10
    long_sum = temp + v11 + v12 + v13 + v14 + v15
end
```

6.12.2 Variable Types

```
model new
fish define types
    v1 = 2
    v2 = 3.4
    v3 = 'Have a nice day'
    v4 = v1 * v2
```

```
    v5 = v3 + ', old chap'
    v6 = vector(1,2,3)
    v7 = matrix(vector(1,1,1))
    v8 = true
end
@types
fish list
```

The resulting screen display looks like this:

```
           Name    Value
           ------- -------------------
(function) types 0 (integer)
           v1    2 (integer)
           v2    3.400000000000000e+00 (real)
           v3    'Have a nice day' (string)
           v4    6.800000000000000e+00 (real)
           v5    'Have a nice day, old chap' (string)
           v6    (1.000000000000000e+00,2.000000000000000e+00,
                  3.000000000000000e+00) (vector3)                [**]
           v7    3 x 1 (matrix)
           v8    true (boolean)
```

6.12.3 Traditional for loop in FISH

Standard `for loop` is also available in FISH to provide for additional loop control.

```
fish define xxx
    sum  = 0
    prod = 1
    loop for (n = 1, n <= 10, n = n + 1)
        sum  = sum  + n
        prod = prod * n
```

```
    end_loop
    io.out('The sum is ' + string(sum) + ...
           ' and the product is ' + string(prod))
end
@xxx
```

6.12.3.1 Controlled loop

```
model new
fish define xxx
    sum  = 0
    prod = 1
    loop n (1,10)
        sum  = sum  + n
        prod = prod * n
    end_loop
    io.out('The sum is ' + string(sum) +  ...
           ' and the product is ' + string(prod))
end
@xxx
```

In this case, the loop variable n is given successive values from 1 to 10, and the statements inside the loop (between the loop and endloop statements) are executed for each value. As mentioned, variable names or an arithmetic expression could be substituted for the numbers 1 or 10. Note that the exit statement can be used to break out of a FISH loop and the continue statement can be used to skip the remaining instructions in the loop, moving to the next sequence of the loop.

It is important to note that this formulation of looping is different from a for loop in most high-level programming languages. For instance, one cannot easily control the ending condition (i.e., loop from 1 to 10 excluding 10) or the incrementing mechanism (i.e., loop from 1 to 10 by twos or loop backward). A standard for loop

is also available in FISH to provide for additional loop control.

6.12.4 if else endif construct

These statements allow conditional execution of FISH function segments; else and then are optional. The item test consists of one of the following symbols or symbol pairs:

= # > < >= <=

The displayed value of abc in this example depends on the argument provided to abc when it is executed. You should experiment with different test symbols (e.g., replace > with <).

Until now, our FISH functions have been invoked from FLAC3D, either by using the sqaure brackets [] of inline FISH, by giving the function name prepended with the the @ character, or by using the fish list command. It is also possible to do the reverse, to give FLAC3D commands from within FISH functions. Most valid FLAC3D commands can be embedded between the following FISH statements:

```
model new
fish define abc(xx)
    if xx > 0 then
        abc = 33
    else
        abc = 11
    end_if
end
[abc(1)]
[abc(-1)]
```

6.12.5 Arrays and Maps

It is often the case that one would like to store a list of objects that they will loop over in the future. These may be computed values from zones, for instance, or specific gridpoint pointers themselves. FISH has two containers to use in these circumstances, termed arrays and maps.

An array holds a list of FISH variables of any type that can be looped over or accessed by the integer index of the element of the array. Arrays can be multidimensional and do not resize dynamically. The simple example below shows how one can create an array of integers and then sum the values.

6.12.5.1 Array example

```
model new
fish define array_operation
    ;create and populate an array with products of 2
    arr = array.create(10)
    loop local n(1,10)
        arr[n] = 2*n
    end_loop

    ;compute the sum and product of elements in the array
    sum = 0
    prod = 1
    local i = 1
    loop while (i <= array.size(arr,1))
        sum = sum + arr[i]
        prod = prod * arr[i]
        i = i + 1
    end_loop
    io.out('The sum is ' + string(sum) + ...
```

```
            ' and the product is ' + string(prod))
end
@array_operation
```

In this example, an array is created and filled with numbers. The loop while construct is used to loop over the array entries and the sum and product are computed and output.

A map, on the other hand, is an associative container, meaning that one can access the members of a map by an integer or string used to insert a value in the map. Maps can dynamically be resized and added to one another (appending maps together), and are the preferred constructs for storing lists of FISH variables for later access.

6.12.5.2 Map example

```
model new
fish define map_operation
    ;create and populate a map with products of 2
    my_map = map(1,2)
    loop local n(2,10)
        map.add(my_map,n,2*n)
    end_loop

    ;compute the sum and product of elements in the map
    sum = 0
    prod = 1
    loop foreach n my_map
        sum = sum + n
        prod = prod * n
    end_loop
    io.out('The sum is ' + string(sum) + ...
```

```
            ' and the product is ' + string(prod))
end
@map_operation
```

Unlike with arrays, maps can be looped through using the loop foreach construct. In this case, n is the value held in each map entry, not the integer name of the object in the map. Likewise, instead of using integers to insert objects into the map, one could use strings such as first, second, etc. This allows one to easily and efficiently store and access FISH variables by a user-defined name.

6.12.6 Fish Function

FISH functions to calculate bulk and shear moduli

```
model new
fish define derive(y_mod,p_ratio)
    s_mod = y_mod / (2.0 * (1.0 + p_ratio))
    b_mod = y_mod / (3.0 * (1.0 - 2.0 * p_ratio))
end
[derive(5e8,0.25)]
[b_mod]
[s_mod]
```

Plasticity theory was developed by people who thought in terms of metals, and for about twenty years workers in soil mechanics have been looking at the theory, rather as outsiders, and asking whether it had anything to offer them

(Peter Wroth 1973))

7 Theory

7.1 Structural Elements

Structural elements include beams, cables, piles, shells, geogrids, and liners. These can be either independent of, or coupled to, the grid. Full dynamic equations of motion are solved (Lagrangian solution procedure as opposed to implicit matrix-inversion procedure).

Section organization: 1. means by which SEL are created and joined 2. Boundary and Initial Conditions, Local Coord Sys. 3. Damping, Thermal, Material Properties 4. Mechanical behavior

Further, a detailed discussion of General Formulation of Structural-Element Logic provides more complex interaction btw SEL and grid.

7.1.1 Types of SEL

1. Beam : 2-noded, straight finite element with 6 dof per node, 3 transl, 3 rot, which behaves as LE matl w no failure limit. Beam elements may be rigidly

connected to grid st forces and bending moments develop within the beam as grid deforms. Beam Structural Elements — Beam structural elements are two-noded, straight, finite elements with six degrees of freedom per node: three translational components, and three rotational components. A physical beam (i.e., an arbitrarily curved, beam structure of isotropic material and bisymmetrical cross-section) can be modeled as a collection of beam elements. Each element behaves as a linearly elastic material with no failure limit; however, it is possible to introduce a limiting plastic moment, or even a plastic hinge (across which discontinuity in rotation may develop), between elements. Beam elements may be rigidly connected to the grid such that forces and bending moments develop within the beam as the grid deforms, and they may be loaded by point or distributed loads. Beam elements are used to model structural-support members in which bending resistance and limited bending moments occur, including support struts in an open-cut excavation and general framed structures loaded by point or distributed loads.

2. Cable Structural Elements — Cable structural elements are two-noded, straight, finite elements with one axially oriented translational degree-of-freedom per node. A physical cable (i.e., an arbitrarily curved, cable structure of isotropic material) can be modeled as a collection of cable elements. Each element can yield in tension or compression, but cannot resist a bending moment. A shear-directed (parallel with the cable axis) frictional interaction occurs between the cable and the grid. A cable may be anchored at a specific point in the grid, or grouted so that force develops along its length in response to relative motion between the cable and the grid. Cables may also be point-loaded or pretensioned. Cable elements are used to model a wide variety of structural-support members for which tensile capacity is important, including cable bolts and tiebacks.
3. Pile Structural Elements — Pile structural elements are two-noded, straight, finite elements with six degrees of freedom per node. A physical pile can be modeled as a collection of pile elements. The stiffness matrix of a pile element

is identical to that of a beam element; however, in addition to providing the structural behavior of a beam, both a normal-directed (perpendicular to the pile axis) and a shear-directed (parallel with the pile axis) frictional interaction occurs between the pile and the grid. In this sense, piles offer the combined features of beams and cables. In addition to skin-friction effects, end-bearing effects can also be modeled (see Axially Loaded Pile). Piles may be loaded by point or distributed loads. Pile elements are used to model structural-support members, such as foundation piles, for which both normal- and shear-directed frictional interaction with the rock or soil mass occurs. A special material model is also available as an extension to the pile element to simulate the behavior of rockbolt reinforcement. This model includes the ability to account for changes in confining stress around the reinforcement, strain-softening behavior of the material between the structural element and the grid, and tensile rupture of the element.

4. Shell Structural Elements — Shell structural elements are three-noded, flat finite elements. Five finite-element types (2 membrane elements, 1 plate-bending element and 2 shell elements) are available. A physical shell (i.e., an arbitrarily curved, shell structure of either isotropic or orthotropic material) can be modeled as a faceted surface composed of a collection of shell elements. The structural response of the shell is controlled by the finite-element type (to resist membrane loading only, bending loading only, or both membrane and bending loading). Each shell element behaves as an isotropic or orthotropic, linearly elastic material with no failure limit; however, one can introduce a plastic-hinge line (across which a discontinuity in rotation may develop) along the edges between elements, using the same double-node procedure as is applied to beams. Shell elements may be rigidly connected to the grid such that stresses develop within the shell as the grid deforms, and they may be loaded by point loads or surface pressures. Shell elements are used to model the structural support provided by any thin-shell structure in which the displacements caused by transverse-shearing deformations can be neglected.

5. Geogrid Structural Elements — Geogrid structural elements are three-noded, flat, finite elements that are assigned a finite-element type that resists membrane but does not resist bending loading. A physical membrane can be modeled as a collection of geogrid elements. The geogrid elements behaves as an isotropic or orthotropic, linearly elastic material with no failure limit. A shear-directed (in the tangent plane to the geogrid surface) frictional interaction occurs between the geogrid and the FLAC3D grid, and the geogrid is slaved to the grid motion in the normal direction. A geogrid can be anchored at a specific point in the FLAC3D grid, or attached so that stress develops along its surface in response to relative motion between the geogrid and the FLAC3D grid. The geogrid can be thought of as the two-dimensional analog of a one-dimensional cable. Geogrid elements are used to model flexible membranes whose shear interaction with the soil are important, such as geotextiles and geogrids.
6. Liner Structural Elements — Liner structural elements are three-noded, flat finite elements that can be assigned any of the five finite-element types available for shell elements. A physical liner can be modeled as a collection of liner elements that are attached to the surface of the FLAC3D grid. In addition to providing the structural behavior of a shell, a shear-directed (in the tangent plane to the liner surface) frictional interaction occurs between the liner and the FLAC3D grid. Also, in the normal direction, both compressive and tensile forces can be carried, and the liner may break free from (and subsequently come back into contact with) the grid. Liner elements are used to model thin liners for which both normal-directed compressive/tensile interaction and shear-directed frictional interaction with the host medium occurs, such as shotcrete-lined tunnels or retaining walls.

An option that allows interaction with the FLAC3D grid on both sides of the liner is available with the liner element.

7.1.1.1 Joining

When using struct cable create command, if any of the nodes lie within zones, these nodes will be linked to these zones, and the link properties will be set consistent with the corresponding element behavior.

Specifying more than one segment improves accuracy, especially with piles and cables interacting with host medium

In this case, distribution of shear forces along pile or cable is a function of number of nodes. - Try to provide approx. 1 node in each zone, since zones are constant-stress regions. it is not necessary to have more than one interaction point within zone. - Try to provide 2~3 cable elements within development length of cable. This development length is determined by dividing the specified yield strength F by the grout cohesive strength c. By following this procedure, failure by "pull-out" can occur if such conditions arise. If cable elements are too long, then only the yield failure mode of each element is possible.

7.1.2 Joining SEL to one another

SEL can be joined by sharing a node or by having one of their nodes linked to another or to a zone. If two or more elements share a node, all forces and moments are transferred btw elements at the node. If its necessary to limit forces, then two separate nodes may be created and connected by a node-to-node link and set appropriate attachment conditions set.

Two structure beam create commands will create 2 nodes in 1 geometric location and specify two separate IDs. Forces and moments will not be transferred btw adjoining elements; instead, only forces wil be transmitted into surrounding zone at the common location.

7.2 Interface

There are several instances in geomechanics in which it is desirable to represent planes on which sliding or separation can occur. For [example://](example:/){.uri}

1. joint, fault, or bedding planes in a geologic medium;
2. an interface between a foundation and the soil;
3. a contact plane between a bin or chute and the material that it contains;
4. a contact between two colliding objects; and
5. a planar "barrier" in space, which represents a fixed, non-deformable boundary at an arbitrary position and orientation.

FLAC3D provides interfaces that are characterized by Coulomb sliding and/or tensile and shear bonding. Interfaces have the properties of friction, cohesion, dilation, normal and shear stiffnesses, and tensile and shear bond strength. Although there is no restriction on the number of interfaces or the complexity of their intersections, it is generally not reasonable to model more than a few simple interfaces with FLAC3D because it is awkward to specify complicated interface geometry. The program 3DEC (Itasca 2007) is specifically designed to model many interacting bodies in three dimensions; it should be used instead of FLAC3D for the more complicated interface problems.

Interfaces may also be used to join regions that have different zone sizes. In general, the zone attach command should be used to join grids together. However, in some circumstances, it may be more convenient to use an interface for this purpose. In this case, the interface is prevented from sliding or opening because it does not correspond to any physical entity.

7.3 Formulation

FLAC3D represents interfaces as collections of triangular elements (interface elements), each of which is defined by three nodes (interface nodes). Interface elements can be created at any location in space. Generally, interface elements are attached to a zone surface face; two triangular interface elements are defined for every quadrilateral zone face. Interface nodes are then created automatically at every interface element vertex. When another grid surface comes into contact with an interface element, the contact is detected at the interface node and is characterized by normal and shear stiffnesses, and sliding properties.

Each interface element distributes its area to its nodes in a weighted fashion. Each interface node has an associated representative area. The entire interface is thus divided into active interface nodes representing the total area of the interface. Figure 1 illustrates the relation between interface elements and interface nodes, and the representative area associated with an individual node.

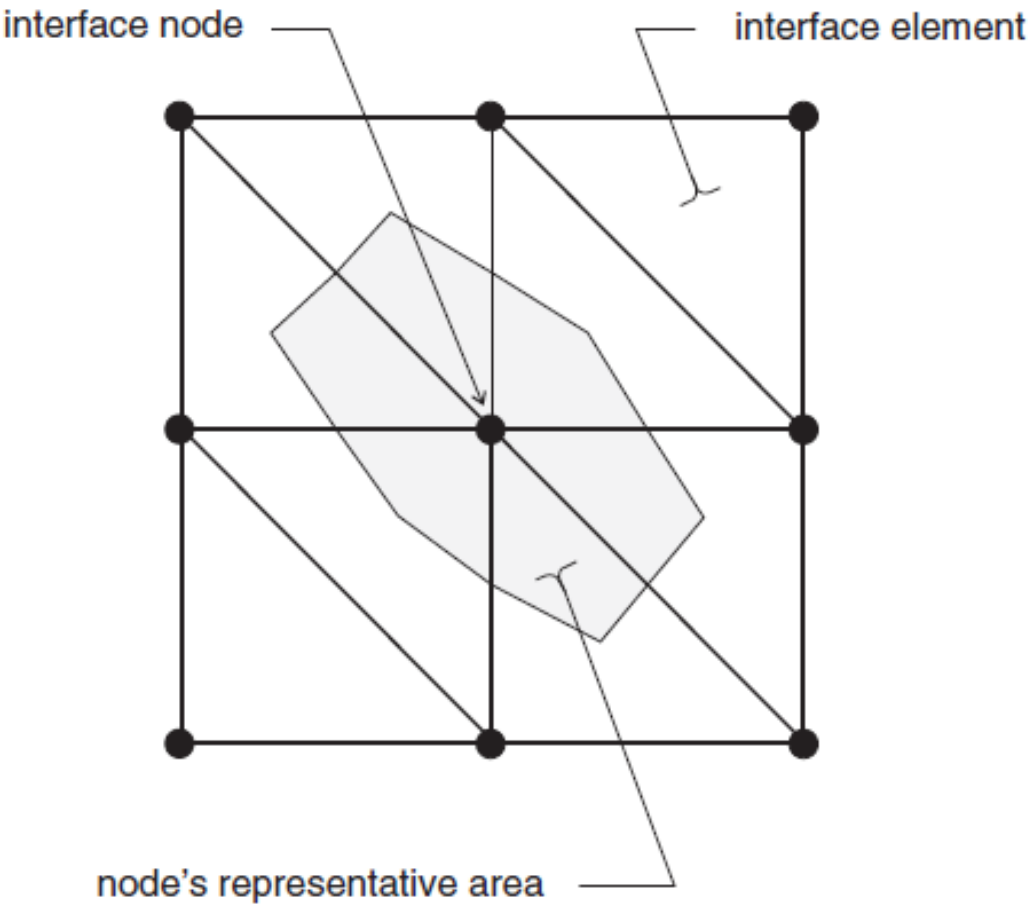


Figure 1: Distribution of representative areas to interface nodes.

It is important to note that interfaces are one-sided in FLAC3D. (This differs from the formulation of two-sided interfaces in two-dimensional FLAC (Itasca 2011).) It may be helpful to think of FLAC3D interfaces as "shrink-wrap" that is stretched over the desired surface, causing the surface to become sensitive to interpenetration with any other face with which it may come into contact.

The fundamental contact relation is defined between the interface node and a zone surface face, also known as the target face. The normal direction of the interface force is determined by the orientation of the target face.

During each timestep, the absolute normal penetration and the relative shear

velocity are calculated for each interface node and its contacting target face. Both of these values are then used by the interface constitutive model to calculate a normal force and a shear-force vector. The constitutive model is defined by a linear Coulomb shear-strength criterion that limits the shear force acting at an interface node, normal and shear stiffnesses, tensile and shear bond strengths, and a dilation angle that causes an increase in effective normal force on the target face after the shear-strength limit is reached. By default, pore pressure is used in the interface effective stress calculation. This option can be activated/deactivated using the command zone interface effective command by setting effective = on/off. Figure 2 illustrates the components of the constitutive model acting at interface node (P):

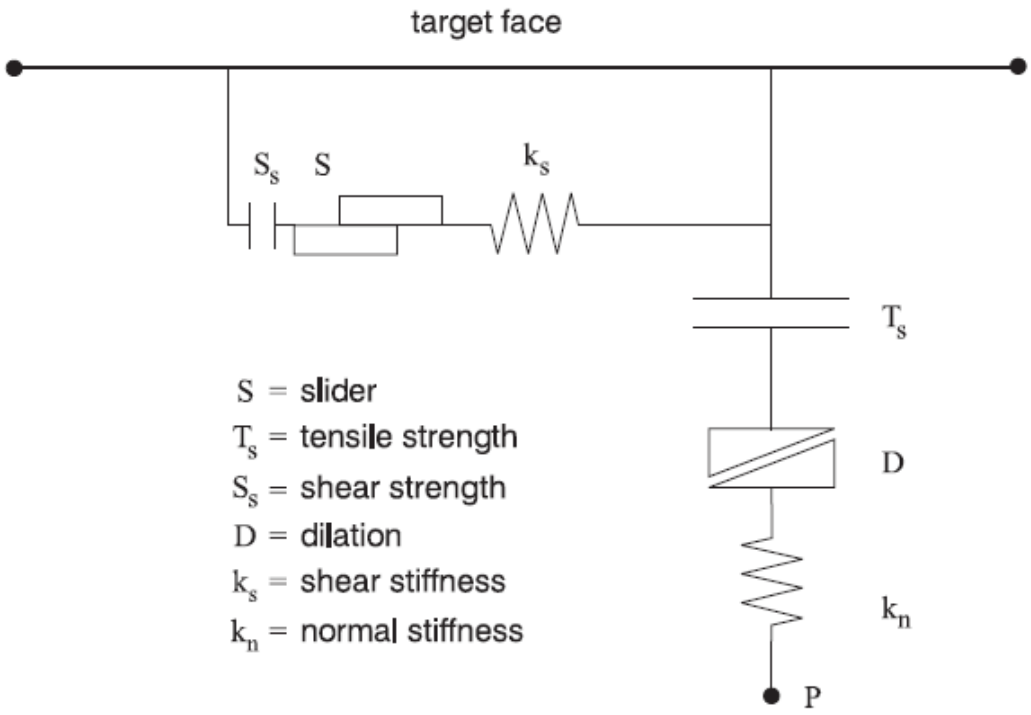


Figure 2: Components of the bonded interface constitutive model.

The normal and shear forces that describe the elastic interface response are determined at calculation time $(t + \Delta t)$ using the relations:

$$F_n^{(t+\Delta t)} = k_n u_n A + \sigma_n A$$

$$F_{si}^{(t+\Delta t)} = F_{si}^{(t)} + k_s \Delta u_{si}^{(t+0.5\Delta t)} A + \sigma_{si} A$$

The inelastic interface logic works in the following way:

1. Bonded interface — The interface remains elastic if stresses remain below the bond strengths; there is a shear bond strength, as well as a tensile bond strength. The normal bond strength is set using the tension interface property keyword. The command zone interface node property shear-bond-ratio = sbr sets the shear bond strength to sbr times the normal bond strength. The default value of property shear-bond-ratio (if not given) is 100.0. The bond breaks if either the shear stress exceeds the shear strength, or the tensile effective normal stress exceeds the normal strength. Note that giving property shear-bond-ratio alone does not cause a bond to be established—the tensile bond strength must also be set.

2. Slip while bonded — An intact bond, by default, prevents all yield behavior (slip and separation). There is an optional property switch (bonded-slip) that causes only separation to be prevented if the bond is intact (but allows shear yield, under the control of the friction and cohesion parameters, using (F_n) as the normal force). The command to allow/disallow slip for a bonded interface segment is zone interface node and by setting bonded-slip on or off.

The default state of bonded-slip (if not given) is off.

3. Coulomb sliding — A bond is either intact or broken. If it is broken, then the behavior of the interface segment is determined by the friction and cohesion (and of course the stiffnesses). This is the default behavior, if bond strengths are not set (zero). A broken bond segment cannot take effective tension (which may occur under compressive normal force, if the pore pressure is greater). The shear force is zero (for a nonbonded segment) if the effective normal force is tensile or zero.

The Coulomb shear-strength criterion limits the shear force by the relation:

$$F_{smax} = cA + tan\phi(F_n - pA)$$

During sliding, shear displacement may cause an increase in the effective normal stress on the joint, according to the relation:

$$\sigma_n := \sigma_n + \frac{|F_s|_o - F_{smax}}{Ak_s} tan\psi k_n$$

On printout (see the zone interface node list command) the value of tension denotes whether a bond is intact or broken (or not set) — nonzero or zero, respectively.

The normal and shear forces calculated at the interface nodes are distributed in equal and opposite directions to both the target face and the face to which the interface node is connected (the host face). Weighting functions are used to distribute the forces to the gridpoints on each face. The interface stiffnesses are added to the accumulated stiffnesses at gridpoints on both sides of the interface in order to maintain numerical stability.

Interface contacts are detected only at interface nodes, and contact forces are transferred only at interface nodes. The stress state associated with a node is assumed to be uniformly distributed over the entire representative area of the node. Interface properties are associated with each node; properties may vary from node to node.

By default, the effect of pore pressure is included in the interface calculation by using effective stress as the basis for the slip condition. (The interface pore pressure is interpolated from the target face.) This applies in model configure fluid mode, or if pore pressures are assigned with the zone water or zone gridpoint initialize pore-pressure command without specifying model configure fluid. The user

can switch options for interface s by using the zone interface effective command and by setting effective on or off. By default in the FLAC3D logic, fluid flow (saturated or unsaturated) is carried across an interface, provided the interface keyword maximum-edge is not used for that particular interface. The permeable interface option can be deactivated/reactivated for interface s by using the zone interface permeability command and by setting effective on or off. Note that if the keyword maximum-edge is used after the zone interface element command, and permeability is on for a particular interface, a warning is issued to inform the user that this interface will be considered as impermeable to fluid flow. (Note that for fluid flow calculation only, a mechanical model must be present. Also, the model cycle 0 command with model mechanical active on should be used to initialize the weighting factors used to transfer fluid flow information across the interface.) No pressure drop normal to the joint and no influence of normal displacement on pore pressure is calculated.

Also, flow of fluid along the interface is not modeled.

7.4 Creation of Interface Geometry

Interfaces are created with the zone interface create command. For cases in which an interface between two separate grids in the model is required, the zone interface create by-face command should be used to attach an interface to one of the grid surfaces. This command generates interface elements for interface s along all surface zone faces with a center point that fall within a specified range. Any surfaces on which an interface is to be created must be generated initially; it must be possible to specify an existing surface in order to create the interface elements. A gap must be specified between two adjacent surfaces, unless the zone interface create by-face command and the separate keyword are given. In this case, the separate sub-grids

may have surface gridpoints at the same location in space.

By default, two interface elements are created for each zone face. The number of interface elements can be increased by using the zone interface s element maximum-edge v command. [1] This causes all interface elements with edge lengths larger than v to subdivide into smaller elements until their lengths are smaller than v. This command can be used to increase the resolution and decrease arching of forces in portions of a model that have large contrasts in zone size across an interface.

Several rules should be followed when using interface elements in FLAC3D:

1. If a smaller surface area contacts a larger surface area (e.g., a small block resting on a large block), the interface should be attached to the smaller region.

2. If there is a difference in zone density between two adjacent grids, the interface should be attached to the grid with the greater zone density (i.e., the greater number of zones within the same area).
3. The size of interface elements should always be equal to or smaller than the target faces with which they will come into contact. If this is not the case, the interface elements should be subdivided into smaller elements.

4. Interface elements should be limited to grid surfaces that will actually come into contact with another grid.

   A simple example illustrating the procedure for interface creation is provided in “DippingJoint.f3dat”. The corresponding project file, “DippingJoint.f3prj”, is located in the folder “datafiles InterfaceDippingJoint.” The example is a block specimen containing a single joint dipping at an angle of 45ř

```
; Create interface elements on the top surface of the base
zone interface 'joint' create by-face separate range group 'Top' group 'Base'
model save 'int1'
return
```

7.5 Typical Properties

Table 8: Typical Values for Dilation Angle (Vermeer and de Borst 1984)

dense sand	15°
loose sand	< 10°
normally consolidated clay	0°
granulated and intact marble	12°–20°
concrete	12°

Post-Failure Properties In many instances, particularly in mining engineering, the response of a material after failure has initiated is an important factor in the engineering design. Consequently, the post-failure behavior must be simulated in the material model. In FLAC3D, this is accomplished with properties that define four types of post failure response: 1. shear dilatancy; 2. shear hardening/softening; 3. volumetric hardening/softening; and 4. tensile softening. These properties are only activated after failure is initiated, as defined by the Mohr-Coulomb relation or the tensile-failure criterion. Shear dilatancy is simulated with the Mohr-Coulomb, ubiquitous-joint and strain-softening MohrCoulomb and ubiquitous-joint models. Shear hardening/softening is simulated with the strain-softening Mohr-Coulomb and

ubiquitous-joint models, and volumetric hardening/softening is simulated with the modified Cam-clay model. Tensile softening is simulated with the strain-softening Mohr-Coulomb and ubiquitous-joint models

7.5.1 Shear Dilatancy

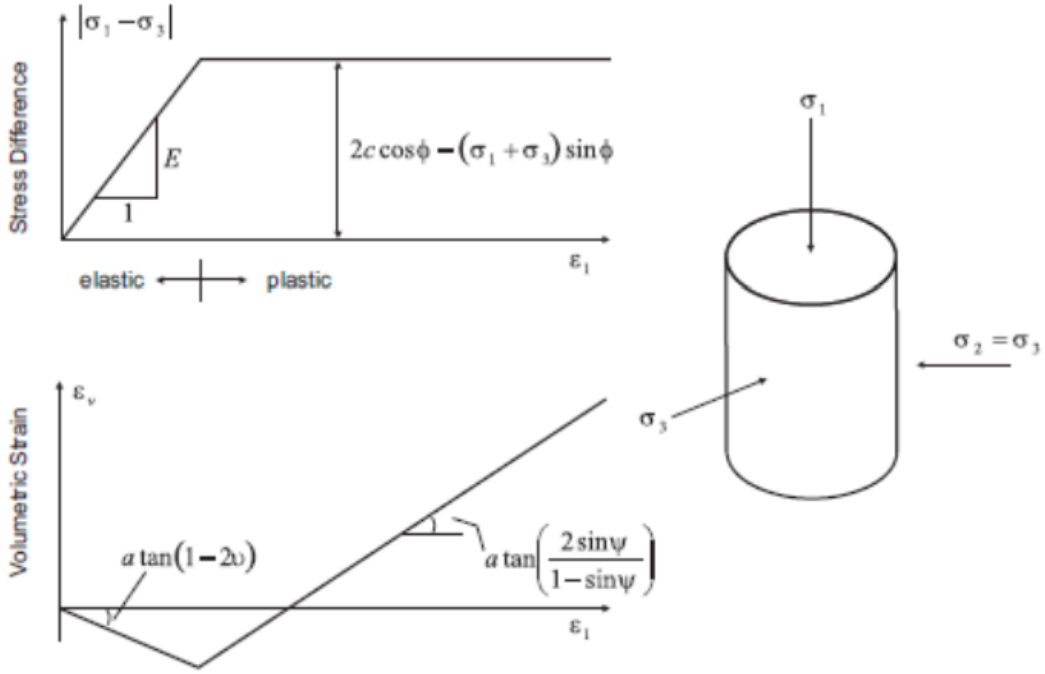


Figure 1: Idealized relation for dilation angle, ψ, from triaxial test results (Vermeer and de Borst 1984).

Shear dilatancy, or dilatancy, is the change in volume that occurs with shear distortion of a material. Dilatancy is characterized by a dilation angle, , which is related to the ratio of plastic volume change to plastic shear strain. This angle can be specified in the Mohr-Coulomb ubiquitous-joint and strain-hardening/ softening models in FLAC3D. Dilation angle is typically determined from triaxial tests or shear-box tests. For example, the idealized relation for dilatancy, based upon the Mohr-Coulomb failure surface, is depicted for a triaxial test in the figure below. The dilation angle is found from the plot of volumetric strain versus axial strain. Note that the initial slope for this plot corresponds to the elastic regime, while the slope used to measure the dilation angle corresponds to the plastic regime

	Cohesion (kPa)	Friction Angle Peak (degrees)	Residual (degrees)
fines			
sandy gravel with silty or clayey fines	1.0	35	32
mixture of gravel and sand with fines	3.0	28	22
uniform sand — fine	—	32	30
uniform sand — coarse	—	34	30
well-graded sand	—	33	32
low-plasticity silt	2.0	28	25
medium- to high-plasticity silt	3.0	25	22
low-plasticity clay	6.0	24	20
medium-plasticity clay	8.0	20	10
high-plasticity clay	10.0	17	6
organic silt or clay	7.0	20	15

Selected Strength Properties (drained, laboratory-scale) for Soils (Ortiz et al. 1980)

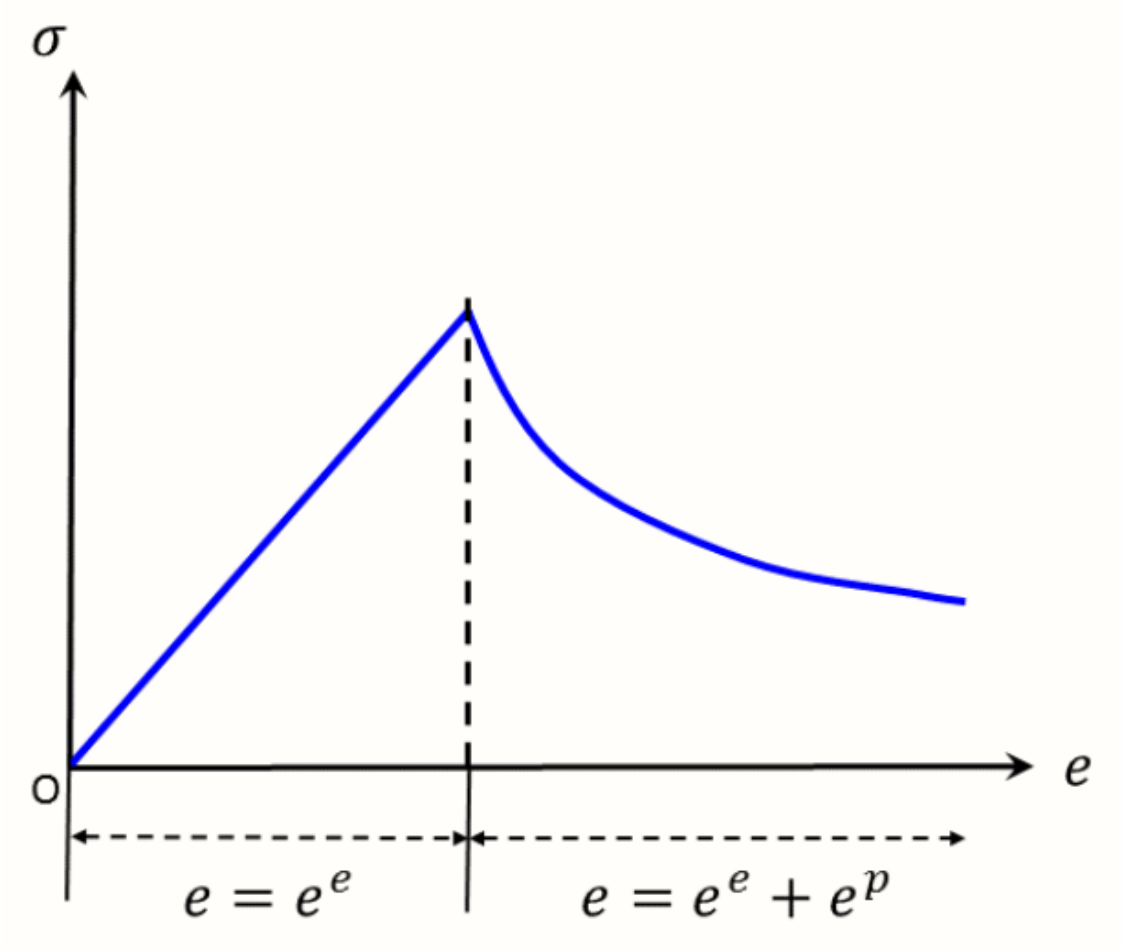
	Dry Density (kg/m ^{3})	Elastic Modulus E (MPa)	Poisson's Ratio
loose uniform sand	1470	10 - 26	0.2 - 0.4
dense uniform sand	1840	34 - 69	0.3 - 0.45

7.5.2 Compression Test on strain-softening material

```
model new
; Create zones
zone create cylinder point 0 (0,0,0) point 1 (1,0,0) point 2 (0,2,0) ...
point 3 (0,0,1) size 4 5 4
zone reflect normal (1,0,0)
```

```
zone reflect normal (0,0,1)
; Constitutive Model and properties
zone cmodel assign strain-softening
zone property bulk 1.19e10 shear 1.1e10
zone property cohesion 2.72e5 friction 44 tension 2e5
zone property table-cohesion 'coh' table-friction 'fri'
table 'coh' add (0,2.72e5) (1e-4,2e5) (2e-4,1.5e5) (3e-4,1.03e5) (1,1.03e5)
table 'fri' add (0,44) (1e-4,42) (2e-4,40) (3e-4,38) (1,38)
; Boundary Conditions
zone face apply velocity (0, 1e-7,0) range position-y 0
zone face apply velocity (0,-1e-7,0) range position-y 2
; Histories
zone history displacement-y position (0,0,0)
zone history stress-yy position (0,1,0)
zone history stress-yy position (1,1,0)
model step 3000
model save 'Softening'
```

The horizontal stress-displacement response is monitored again, as shown in the image below. This test produces distinct peak and residual failure stress levels



The strain-softening model assumes both a brittle softening (due to reduction in cohesion) and a gradual softening (due to a reduction in friction angle). The selection of the properties is discussed further in Material Properties. Comparison of the two images above illustrates the different responses of the two models. The initial response up to the onset of failure is identical, but postfailure behavior is quite different. Clearly, more data are required to use the strain-softening model and, typically, the softening model must be calibrated for each specific problem.

8 Command

8.1 Interface

8.1.1 zone face group

```
zone face group s keyword ... <range>
```

Example:

```
zone face group 'wall' internal range group 'wall-c' group 'pile'
```

Adds a group name 'wall' to zone faces included in the optional range. Also can be used with zone face group "slotname = groupname" Zone face may belong to many groups, up to 128. keywords: internal, or, remove.
- internal: to select all faces.
If specified, then the range looks at internal faces as well as surface faces.

- or: considered inside the surface if any of those conditions has a non-null model.

- remove: group is removed from the zone face.

8.1.2 zone face skin

```
zone face skin
```

Generates contiguous face groups by skinning.

"skinning" refers to automatic face group generation, which occurs as follows: the program looks for contiguous faces and puts them into automatically named groups within the slot named skin. Two faces are not considered contiguous if the angle between two faces exceeds the break angle, or if the two faces belong to different groups according to the command's slot specification. Unless internal is specified, the operation of zone face skin is restricted to external faces. Only zone faces that have no zone on the other side, a hidden zone on the other side are considered surface faces. When use-hidden-zones is used, within specified range will be included for consideration.

group names are generated as follows: sEast, sWest, sNorth, sSouth, sTop, sBottom

Internal faces are assigned names of the group on either side that changed in order to create them.

8.1.3 zone separate

```
zone separete by-face new-side group 'iwall' slot 'int' range group 'wall' or 'base
```

separates internal faces specified by the range. The gridpoints of the face are duplicated, and new face is created. New faces and gridpoints get copies of all group and extra variable assignments belonging to the original face and gridpoint.

Example: If Fred and George are group names assigned to zones, then

```
range group 'Fred' group 'George'
```

will select faces that are connected to zone of both group Fred and George.

keyword: - by-face: attempts to separate all internal faces in the range - new-side group: newly created faces will be assigned the group name in the specified slot. The default slot, is named Default.

8.1.4 zone interface create

```
zone interface 'side' create by-face range group 'wall' and 'iwall'


zone interface 'side' node property stiffness-normal 1e8 stiffness-shear 1e8 fricti
```

creates nodes or elements on interface side

There are two techniques for creating an interface. 1. To derive an interface from a range of zone faces using by-face keyword. 2. To specify a triangular interface element from 3 points. 3. Command may also be used to construct an interface node that may be used with the element keyword.

keyword: - by-face: interface elements are created on all surface zone faces that are within the specified range. An error will occur if interface elements from that interface already exist on the selected faces. If the optional separate keyword is used, then internal zone faces are selected by the range. The list of selected faces is then automatically separated (as with zone separate) and interface elements placed on one side. The following are for the separate keyword: new-side-origin, clear-attach.

- element point i keyword . . . : create a triangular interface element by specifying 3 points. Three verticies must be specified by the following keyword. Each point can be created and located in space using the position keyword, or an existing interface node with ID i may be specified. The interface element that is created is not attached to a grid face even if the location corresponds to that of the face. the element is fixed in space. The active side of the element is defined by walking around the edge of the element, from point 1 to point 2 to point 3; the active side is up when walking in a clockwise direction.

For each point of the element, one of the following two keywords must be supplied: node i, position v.

- node v : this creates an interface node at position v. If a node already exists at the selected location, an error is reported. The created node is fixed in space.

8.1.5 range

Logical Operations in a Range Phrase A range phrase targets a set of objects by specifying one or more range elements (the keywords that follow the range keyword in the command). If multiple range elements are specified, the final set of objects returned by the range phrase will be, by default, the intersection of the separate range elements.

The union keyword may be specified anywhere within a range phrase to return a range that is a union of the present range elements.

The not keyword may be used at the end of a particular range element to return the inverse of the objects specified by the element. The not keyword must appear at the end of the range element specification (e.g., range position-x 0 100 not position-y 0 100 not).

When the keyword extent is used, the extent of the object (rather than the object centroid) is used to verify whether the object is within the range. This keyword is only applied to geometric range elements (e.g., sphere).

Logical Operations & Named Ranges

Using the named-range keyword, an existing named range may be inserted into a range phrase and be treated as a range element within that range phrase. Consequently the logical operations described above will work on a named range exactly as they do on range elements. See the next topic, Named Ranges, for further information.

8.1.5.1 Named Ranges

Creating a named range provides an easy-access method for subsequent use of the range. Range creation is performed using the model range command as follows:

model range create ‘fred’ position-x 5 500 id 100 200 group ‘external’ not geometry group edge ‘mainedges’ range named-range ‘fred’ This is equivalent to:

geometry group edge ‘mainedges’ range position-x 5 500 . . . id 100 200 group ‘external’ not Note that, in the former case, the range fred could be reused without retyping all of the range elements.

Named Ranges vs. Groups

Named ranges and groups are close in effect. Either are used as range elements within a range. However, a named range is still a range, which means with each use it is possible it will return different objects, depending the range elements that define it and the state of the model. By contrast, repeated uses of a group within a range will return the same objects each time. The group itself must be altered to return different objects.

8.2 Define Fish functions

```
# define make_pydata, output_structure, tot_reac #
# ================================================
#;call 'p-y' suppress ; Calculates p-y curve for pile, when tot_reac is called
#; Calculate PY Curves at equidistant locations along pile
#; Stored in map called pile, key = 1.0*y positions rounded to integer
#;    value = map, key 'y' = y-position
#;                     'length' = length of ownership of this segment
#;                     'gp' = Gridpoint nearest (0,y,0)
#;                     'nodes' = map of interface nodes at that position, key ord
it.command("""
fish define make_pydata
```

```
global pile = map
; Build list of nodes
loop foreach local ipnt inter.list
    loop foreach node inter.node.list(ipnt)
        local y = inter.node.pos.y(node)
        local key = math.round(inter.node.pos.y(node) * 1.0)
        if map.has(pile,key) == false then
            local level = map
            local nodes = map
            level('y') = y
            level('gp') = gp.near(0,y,0,'pile')
            level('nodes') = nodes
            level('length') = 0.0
            pile(key) = level
        endif
        level = pile(key)
        nodes = level('nodes')
        nodes(map.size(nodes)) = node
        level('nodes') = nodes
        pile(key) = level
    end_loop
end_loop
; Calculate length of ownership of each segment
local prev_key = '0'
loop foreach key map.keys(pile)
    if type.name(prev_key) == 'integer' then
        local levelp = pile(prev_key)
        local leveln = pile(key)
        local dist = math.abs(gp.pos.y(levelp('gp')) - gp.pos.y(leveln('gp')))
        levelp('length') = levelp('length') + dist
```

```
            leveln('length') = leveln('length') + dist
            pile(prev_key) = levelp
            pile(key) = leveln
        endif
        prev_key = key
    end_loop
end

; Output resulting data structure...
fish define output_structure
    loop foreach local level pile
        io.out('Y-Position ' + string(level('y')))
        io.out('Length ' + string(level('length')))
        io.out('GP ' + string(gp.id(level('gp'))))
        local nodelist = ''
        loop foreach local node level('nodes')
            nodelist = nodelist + ' ' + string(interface.node.id(node))
        end_loop
        io.out('Nodes'+nodelist)
    end_loop
end

fish define tot_reac
    local total = 0.0
    loop foreach local key map.keys(pile)
        local level = pile(key)
        local accum = 0.0
        loop foreach local node level('nodes')
            local area = 2.0 * interface.node.area(node)
            local norm = zone.face.normal(inter.node.target.zone(node),inter.node.t
```

```
            local xnstress = inter.node.stress.normal(node) * norm->x * -1.0
            local xsstress = inter.node.stress.shear.x(node)
            accum = accum + (xnstress + xsstress) *  area
        end_loop
        local xdis = gp.disp.x(level('gp'))
        total = total + accum
        table(string(key),xdis) = accum/level('length')
    end_loop
    tot_reac = total
end
""")
print("               ======== PASSED DEFINING ALL FISH FUNCTIONS ========        =
```

8.3 Some Code

Properties may be specified for each type of physical item, and will be inherited automatically by the associated component objects. For example, the command

```
struct cable property grout-friction=30.0 range id=3
```

will assign a grout friction angle of 30 degrees to all cable elements thar are part of the cable with an ID number of 3, whereas the command

```
struct cable property grout-friction=30.0 range component-id=3
```

will assign a grout friction angle of 30 degrees to the single cable element with a component-ID number of 3.

```
" GROUPS AND MASK ARRAYS "
it.command("zone group \"lower\" range position-z 0 5")
za.in_group("lower")
za.in_group("lower").sum(), "zones in lower group."
corner_mask = reduce(np.logical_and, (x<3, y<3, z<3))
```

```
za.set_group(corner_mask, "corner", "geometry")
print za.in_group("corner", "geometry").sum(), "zones in corner group."
" GRIDPOINTS ARRAY FUNCTIONS "
gpos = gpa.pos()
gx, gy, gz = gpos.T
print gz
f = gpa.fixity()
print f
f[:,][gz==0] = True, True, True
print f
gpa.set_fixity(f)
top_gridpoints = gz==10
radial_distance = np.sqrt((gx-5)**2+(gy-5)**2)
central_gridpoints = radial_distance < 5
mask = np.logical_and(top_gridpoints, central_gridpoints)
print "boundary load applied to {} gridpoints".format(mask.sum())
fapp = gpa.force_app()
print fapp
fapp[:,2] = mask*1e6*(5.0-radial_distance)/5.0
gpa.set_force_app(fapp)
print "zone centroids: "
print za.pos()
za.gridpoints()
za.faces()
za.ids()
print za.neighbors()
" ================RESULTS================== "
it.command("model solve")
print "gridpoint displacements:"
print gpa.disp()
```

```
print "gridpoint displacement magnitudes: "
mag = np.linalg.norm(gpa.disp(), axis=1)
print mag
max_index = np.argmax(mag)
print "Maximum displacement: {} at location {}".format(gpa.disp()[max_index],
                                                       gpa.pos()[max_index])
print "Vertical displacement along the vertical line x=5, y=5: from z=0 to z=10"
print gpa.disp()[np.logical_and(gx==5, gy==5)][:,2]
za.stress()
za.stress_flat()
" =========REFERENCE EXAMPLES================ "
""" Some Numpy Operation Examples
np.array([1,2,3,4,5])
np.linspace(0,1,15)
np.zeros((4,4))
a = np.linspace(0,1,15)
b = np.ones_like(a)
np.sin(a)
print a[0]
a[0] = 20.2
print a
c = np.array(((1,2,3),(4,5,6),(7,8,9),(10,11,12)))
print c
c[0][0]
c[:,0]
"""
""" SOME GRIDPOINTS EXAMPLES
z = it.zone.near((5,5,5))
print "central zone id: {}, position: {}".format(z.id(), z.pos())
```

```
for gp in z.gridpoints():
    print "gridpoint with id: {} at {}".format(gp.id(), gp.pos())
"""
```

8.4 Interface

Interfaces are most often created with the zone interface create by-face command:

```
zone interface 'flt' create by-face separate range position-z 5
zone interface 'flt' node property stiffness-normal 1e8 stiffness-shear 1e8
```

This command creates an interface named flt by separating all zone faces found at z=5 and placing interface elements and nodes on one side. Contacts are found (and forces generated) by interface nodes coming into contact with zone surface faces that are not part of the interface itself.

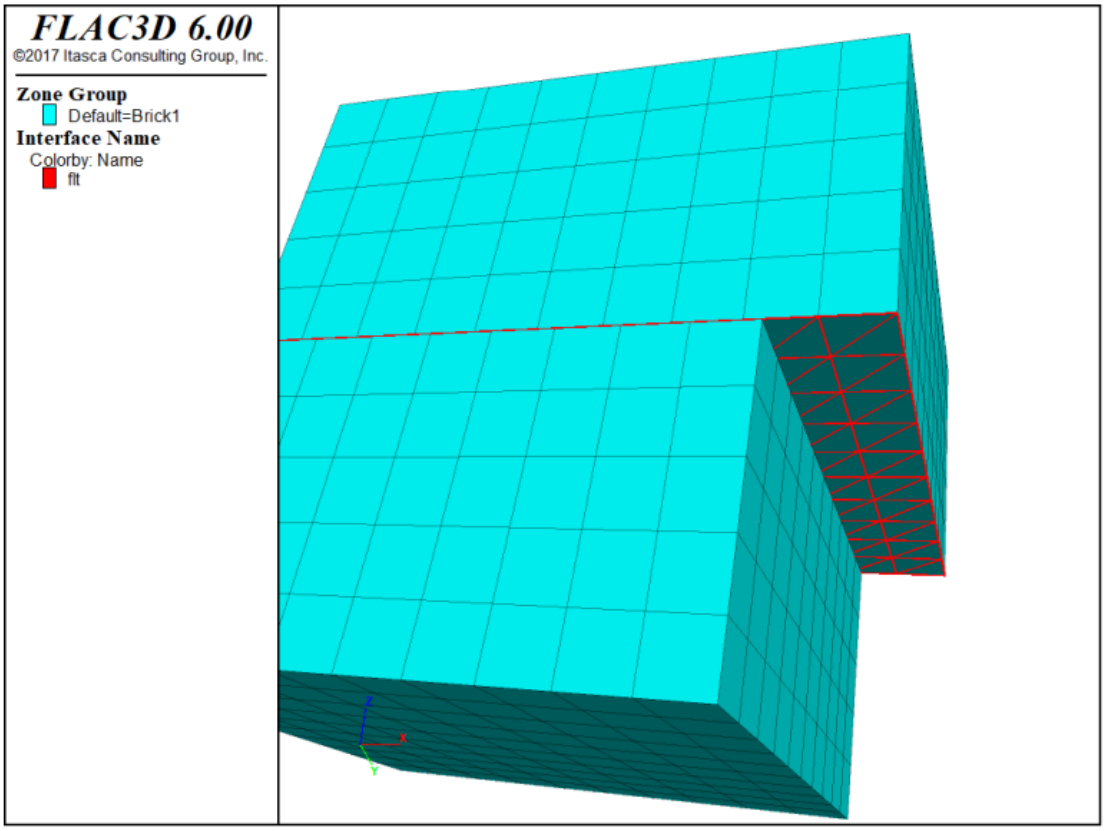


Figure 1: Discontinuity resulting from interface slip.

Figure 8.1: Code chunk syntax

Every reference to an interface uses a name (like the table command) to indicate which interface is being referred to

Interfaces in FLAC3D are one-sided objects. Interface elements are (in general) created attached to zone surface faces, which in turn create interface nodes. All contact detection is between interface nodes and zone surface faces.

Most commonly, interfaces need to be created starting with a mesh that is continuous. In that case, it is easiest to separate the mesh and create the interfaces all in one command using the separate keyword (as shown in the example above).

The new-side-origin keyword can be used to determine which side of the separation interface elements are applied to. The side opposite the location of the origin will be selected. In the example above, to select the other side, change the command to

```
zone interface 'flt' create by-face separate new-side-origin (0,0,10) ...
range position-z 5
```

8.4.1 Penetration

the calculation of normal stress due to interface penetration is absolute instead of incremental. This means that moving a gridpoint relative to an interface in the normal direction will have an immediate effect on the normal stress calculated. This is not true of accumulated shear stress, which remains incremental.

In small-strain mode, gridpoints do not actually move their location in response to accumulated velocities.

small-displacement" field. is a vector field that is accumulated from the velocity field every step in small-strain mode only. For the purposes of calculating interface penetration, a gridpoint location is based on the current location plus the small-strain displacement. The smallstrain displacement may be initialized using the zone gridpoint initialize displacement-small command, and this is sometimes useful to reset the effective penetration of an interface after some grid motion. To initialize interface stresses without actually moving gridpoints, we introduce a normal stress increment that is added to the normal stress calculated from absolute penetration. It is this value that is set when the zone interface node initialize-stresses command is given. It can also be set explicitly using zone interface node stress-normal-increment.

Spatial Variation and Randomness of Property Distribution

Material properties can be specified to adjust or to vary as a function of grid position. In fact, a different property can be assigned to every zone in a FLAC3D model, regardless of model size. There are two commands avialable in FLAC3D to specfiy material properties. The first, zone property, allows multiple properties to be specified in a single command. This is convenient to completely specify properties in different regions, where that property does not vary within that region. The second can only specify one property at a time, but has optional keywords that allow the property value to be automatically varied in space. This is the zone

property-distribution command. Among the available keywords are add, multiply, gradient, and vary, which can be used to provide fixed or linear variations of properties with position. For example, the following command provides a linear variation of cohesion in the x-direction:

```
zone property-distribution cohesion 1e6 gradient (-1e5,0,0)
```

An initial profile of a property can also be assigned via FISH. With FLAC3D, it is also possible to study the influence of nonhomogeneity in a material. Any type of statistical property distribution can be introduced, as each element may have a unique property value. Two optional keywords are available to apply a random distribution of a selected property with the zone propertydistribution command. The keyword deviation-gaussian assumes a normal (Gaussian) distribution for the property, with a mean value, v, and standard deviation, s. The keyword deviation-uniform assumes a uniform distribution with mean value, v, and standard deviation, s. Be careful to ensure that properties do not acquire negative values if s is large. As an example, the following command would give a mean friction angle of 40° with a standard deviation of ±5%: zone property-distribution friction 45 deviation-gaussian 2 Use of the variation keywords in any command (not just for property variation) is defined in the topic Value Modifiers (Add, Multiply, Gradient, Vary).

9

KAIST Model Code for Recovery

9.1 version 16

```
import itasca as it
import numpy as np
np.set_printoptions(threshold=20)
it.command("python-reset-state false")
from itasca import zonearray as za
from itasca import gridpointarray as gpa

it.command("""
program threads 128
""")

########################################
############ PARAMETERS ################
```

```
########################################

# Mesh Details
_radial = 20
_perimeter = _radial
_axial = 2*_radial
_outer = 2*_radial

# Physical Constants
_gravity = 9.80665
_K = 0.318

# Dimensions
_D_shaft = 1.0
_H_shaft = 7.45
_T_plate = 1.5
_B_footing = 3.25
_D_footing = _H_shaft + _T_plate
_B_soil = _B_footing*5
_D_soil = _D_footing+3

# Concrete Properties
_bulk = 13.9e9
_shear = 10.4e9
#_bulk = 13.9e9
#_shear = 10.4e9
_density_concrete = 2400

# Soil Properties
_E_o = 80e6
```

```
_const = 10e6
_poisson = 0.25
_density_soil = 1530

# Interface Properties
_stiff_norm = 1e10
_stiff_shear = 1e10

########################################
############### ZONES ##################
########################################

print("       ((((((((((((((((======== NEW RUN ========))))))))))))))))")

command_zone = """
model new
z crea r-t p 0 (0,0,0) ...
           p 1 ({B_soil},0,0) ...
           p 2 (0,{D_soil},0) ...
           p 3 (0,0,{B_soil}) ...
           p 4 ({B_soil},{D_soil},0) ...
           p 5 (0,{D_soil},{B_soil}) ...
           p 6 ({B_soil},0,{B_soil}) ...
           p 7 ({B_soil},{D_soil},{B_soil}) ...
           p 8 ({B_footing},0,0) ...
           p 9 (0,0,{B_footing}) ...
           p 10 ({B_footing},{D_soil},0) ...
           p 11 (0,{D_soil},{B_footing}) ...
           p 12 ({B_footing},0,{B_footing}) ...
           p 13 ({B_footing},{D_soil},{B_footing}) ...
```

```
            size {radial} {axial} {perimeter} {outer} ...
            rat 1 1 1 1.01 ...
            fill
"""
print("radial mesh number is ", _radial)
command = command_zone.format(
B_footing = _B_footing,
B_soil = _B_soil,
D_soil = _D_soil,
radial=_radial,
axial = _axial,
perimeter = _perimeter,
outer = _outer)

it.command(command)



########################################
######## CONSTITUTIVE MODEL ############
########################################

it.command("""
zone cmodel assign elastic range group "Radial Tunnel1"

fish define fname(E_o,const)
loop foreach pnt zone.list
z_depth = zone.pos.y(pnt)
E = E_o+const*math.sqrt(z_depth)
zone.prop(pnt,'young')=E
end_loop
```

```
end
@fname({E_o_},{const_})
zone property poisson {poisson_} density {density_soil}
""".format(E_o_=_E_o,const_=_const,poisson_=_poisson,density_soil=_density_soil))



# GROUP #
p = za.pos()
x,y,z = p.T
print(it.zone.count(), "zones in whole model")

shaft = reduce(np.logical_and, (np.sqrt(x**2+z**2)<_D_shaft, y<_H_shaft))
za.set_group(shaft, "shaft")
print(za.in_group("shaft").sum(), "zones in shaft group.")

plate = reduce(np.logical_and, (x<_B_footing,z<_B_footing, y>_H_shaft,y<_D_footing)
za.set_group(plate, "plate")
print(za.in_group("plate").sum(), "zones in plate group.")

it.command("""
zone cmodel assign elastic range group 'shaft'
zone cmodel assign elastic range group 'plate'
zone property bulk {bulk_} shear {shear_} density {density_} range group 'shaft'
zone property bulk {bulk_} shear {shear_} density {density_} range group 'plate'
""".format(bulk_=_bulk,shear_=_shear,density_=_density_concrete))

########################################
############# INTERFACE ################
########################################
```

```
it.command("""
zone interface 'interface 1' create by-face separate range group 'plate' group 'Rac
zone interface 'interface 1' node property stiffness-normal {stiff_norm_} stiffness
zone interface 'interface 2' create by-face separate range group 'shaft' group 'Rac
zone interface 'interface 2' node property stiffness-normal {stiff_norm_} stiffness

""".format(stiff_norm_=_stiff_norm, stiff_shear_=_stiff_shear))

print("               ======== PASSED INTERFACE ========")

########################################
######## BOUNDARY CONDITIONS ###########
########################################

it.command("""
zone face skin
zone face apply velocity-normal 0 range group 'West6' or 'West7' or 'Bottom6' or 'B

zone face apply velocity (0,0,0) range group 'North3'
""")

########################################
############# INITILIZE ################
########################################

it.command("""
model gravity 0 {gravity_} 0
zone initialize-stress ratio {K_}
zone interface 'interface 1' node initialize-stresses
zone interface 'interface 2' node initialize-stresses
```

```
""".format(gravity_=_gravity, K_ = _K))

it.command("""
zone ratio local
model solve ratio 1e-4
model save 'initial'
""")

print("               ======== PASSED INITIAL EQUILIBRIUM ========")

########################################
########## VERTICAL LOADING ############
########################################

it.command("""
zone initialize state 0
zone gridpoint initialize displacement (0,0,0)
zone gridpoint initialize velocity     (0,0,0)

table 'ramp' add ([global.step],0) ([global.step+5000],1e-6) ...
      ([global.step+10000],1e-6) ; Increase velocity applied to pile
                                 ; over 5,000 steps



zone face apply velocity-normal 1e-5 range group 'North1' or 'South2'


history interval 10
zone history name 'disp' displacement-y position (3,7,3)
```

```
;=====================================================
; find gridpoints at pile cap, store in map called cap
fish define find_cap
    global cap = map
    loop foreach local gp gp.list
        if gp.isgroup(gp,'top') then
            cap(gp.id(gp)) = gp
        endif
    endloop
end

fish history name 'cap' @find_cap

fish define vert_load
    local yftot = 0.0
    loop foreach gp cap
        yftot = yftot + gp.force.unbal.y(gp)
    end_loop
    vert_load = yftot / (0.25*0.25*{D_shaft_}*{D_shaft_}*math.pi)
end
fish history name 'load' @vert_load
;=====================================================

zone mechanical damping combined
model largestrain true
model step 10000
model save 'vertical-loading'
""".format(D_shaft_=_D_shaft))
```

```
print("               ======== PASSED VERTICAL LOADING ========")

########################################
################ PLOT ##################
########################################

it.command("""
plot create
plot item create zone label group
plot create
plot item create zone contour property name 'young'
plot create
plot item create zone label density
""")

print("               ======== PASSED PLOT ========")
```

```
#; ===========================================================
```

```
#
#; find gridpoints at pile cap, store in map called cap
#fish define find_cap
#    global cap = map
#    loop foreach local gp gp.list
#        if gp.isgroup(gp,'top') then
#            cap(gp.id(gp)) = gp
#        endif
#    endloop
#end
#
#;fish history name 'cap' @find_cap


#fish define vert_load
#    local yftot = 0.0
#    loop foreach gp cap
#        yftot = yftot + gp.force.unbal.y(gp)
#    end_loop
#    vert_load = yftot / (0.25*{D_shaft_}*{D_shaft_}*math.pi)
#end
#

#fish define tot_reac
```

```
#    local total = 0.0
#    loop foreach local key map.keys(shaft)
#        local level = shaft(key)
#        local accum = 0.0
#        loop foreach local node level('nodes')
#            local area = interface.node.area(node)
#            local norm = zone.face.normal(inter.node.target.zone(node),inter.nod
#            local ynstress = inter.node.stress.normal(node) * norm->y * -1.0
#            local ysstress = inter.node.stress.shear.y(node)
#            accum = accum + (ynstress + ysstress) *  area
#        end_loop
#        local ydis = gp.disp.y(level('gp'))
#        total = total + accum
#        table(string(key),ydis) = accum/level('length')
#    end_loop
#    tot_reac = total
#end
#
#fish history name 'load' @vert_load
#fish history name 'tot_reac' @tot_reac
```

```
#; ===========================================================


#plot create
#plot clear
#plot active on
#plot background 'white'
#plot outline active on width 2 color 'black'
#plot legend active on heading color 'black' copyright color 'black' ...
#    placement left size 25,50 ...
#    step active off ...
#    time-real active off ...
#    time-model active off ...
#    title-customer active off ...
#    view-info active off
#plot title-job active off
#plot title active off
#
#plot item create axes active on ...
#    axis-x color 'black' draw-positive on draw-negative off label-positive 'X' .
#    axis-y color 'black' draw-positive on draw-negative off label-positive 'Y' .
#    axis-z color 'black' draw-positive on draw-negative off label-positive 'Z' .
```

```
#    position (0,0,0) size 1 ...
#    font size 10 family 'Arial' style bold ...
#    transparency 0 ...
#    legend active off
#
#plot item create zone active on ...
#    contour Displacement ...

#; ========================================================


#zone.field.name = 'stress-yy'
#zone.field.method.name = 'poly' ; Use polynomial fit extrapolation
#global value1 = zone.field.get(0,_H_shaft,0) ; Return stress-xx
#zone.field.name = 'stress-von-mises' ; Reset the filed name directly to 'stress-
#global value2 = zone.field.get(0,_H_shaft,0) ; Return stress-von-mises
#


#; ========================================================


# VERTICAL LOADING 1#

#top = reduce(np.logical_and, (np.sqrt(x**2+z**2)<_D_shaft, y==np.amin(y)))
#za.set_group(top, "top") # set the zones with shaft = true have "shaft" and "geo

#it.command("""
#zone initialize state 0
#zone gridpoint initialize displacement (0,0,0)
```

```
#zone gridpoint initialize velocity     (0,0,0)
#table 'ramp' add ([global.step],0) ([global.step+30000],-5e-8) ...
#      ([global.step+58000],-5e-8) ; Increase velocity applied to pile
#                                  ; over 30,000 steps
#zone face apply velocity-normal 1 table 'ramp' range group 'top'
#
#history interval 250
#zone history name 'disp' displacement-y position (0,0,0)
#
#; ========================================================
#;  find gridpoints at pile cap, store in map called cap
#
#fish define find_cap
#    global cap = map
#    loop foreach local gp gp.list
#        if gp.isgroup(gp,'top') then
#            cap(gp.id(gp)) = gp
#        endif
#    endloop
#end
#@find_cap
#
#; ========================================================
#; monitor vertical loading at pile cap
#
#fish define vert_load
#    local yftot = 0.0
#    loop foreach gp cap
#        yftot = yftot + gp.force.unbal.y(gp)
#    end_loop
```

```
#    vert_load = yftot / (0.25*{1}*{1}*math.pi)
#end
#;call 'load'
#; ============================================================
#
#fish history name 'load' @vert_load
#zone mechanical damping combined
#model step 58000
#model save 'vertical-loading'
#""")
#print("               ========       ======== Passed Vertical Loading 1 ========
#
## VERTICAL LOADING 2#
## ============================================================
#it.command("""
#; vertical loading
#model restore 'initial'
#zone initialize state 0
#zone gridpoint initialize displacement (0,0,0)
#zone gridpoint initialize velocity     (0,0,0)
#zone face apply stress-yy [1.0e5/(math.pi*1*1)] range group 'top'
#model solve ratio 1e-4
#model save 'vertical-load-end'
#""")
#print("               =======        ======== Passed Vertical Loading 2 ========
#
## define make_pydata, output_structure, tot_reac #
## ============================================================
##;call 'p-y' suppress ; Calculates p-y curve for pile, when tot_reac is called
##; Calculate PY Curves at equidistant locations along pile
```

```
##; Stored in map called pile, key = 1.0*y positions rounded to integer
##;    value = map, key 'y' = y-position
##;                     'length' = length of ownership of this segment
##;                     'gp' = Gridpoint nearest (0,y,0)
##;                     'nodes' = map of interface nodes at that position, key or
#
#it.command("""
#fish define make_pydata
#    global pile = map
#    ; Build list of nodes
#    loop foreach local ipnt inter.list
#        loop foreach node inter.node.list(ipnt)
#            local y = inter.node.pos.y(node)
#            local key = math.round(inter.node.pos.y(node) * 1.0)
#            if map.has(pile,key) == false then
#                local level = map
#                local nodes = map
#                level('y') = y
#                level('gp') = gp.near(0,y,0,'pile')
#                level('nodes') = nodes
#                level('length') = 0.0
#                pile(key) = level
#            endif
#            level = pile(key)
#            nodes = level('nodes')
#            nodes(map.size(nodes)) = node
#            level('nodes') = nodes
#            pile(key) = level
#        end_loop
#    end_loop
```

```
#    ; Calculate length of ownership of each segment
#    local prev_key = '0'
#    loop foreach key map.keys(pile)
#        if type.name(prev_key) == 'integer' then
#            local levelp = pile(prev_key)
#            local leveln = pile(key)
#            local dist = math.abs(gp.pos.y(levelp('gp')) - gp.pos.y(leveln('gp')
#            levelp('length') = levelp('length') + dist
#            leveln('length') = leveln('length') + dist
#            pile(prev_key) = levelp
#            pile(key) = leveln
#        endif
#        prev_key = key
#    end_loop
#end
#
#; Output resulting data structure...
#fish define output_structure
#    loop foreach local level pile
#        io.out('Y-Position ' + string(level('y')))
#        io.out('Length ' + string(level('length')))
#        io.out('GP ' + string(gp.id(level('gp'))))
#        local nodelist = ''
#        loop foreach local node level('nodes')
#            nodelist = nodelist + ' ' + string(interface.node.id(node))
#        end_loop
#        io.out('Nodes'+nodelist)
#    end_loop
#end
#
```

```
#fish define tot_reac
#    local total = 0.0
#    loop foreach local key map.keys(pile)
#        local level = pile(key)
#        local accum = 0.0
#        loop foreach local node level('nodes')
#            local area = 2.0 * interface.node.area(node)
#            local norm = zone.face.normal(inter.node.target.zone(node),inter.nod
#            local xnstress = inter.node.stress.normal(node) * norm->x * -1.0
#            local xsstress = inter.node.stress.shear.x(node)
#            accum = accum + (xnstress + xsstress) *  area
#        end_loop
#        local xdis = gp.disp.x(level('gp'))
#        total = total + accum
#        table(string(key),xdis) = accum/level('length')
#    end_loop
#    tot_reac = total
#end
#""")
#print("               ======== PASSED DEFINING ALL FISH FUNCTIONS ========
## GENERATE p-y data #
## =======================================================
#it.command("""
#@make_pydata        ; Generate p-y curve calculation data
#@output_structure   ; Sanity check of p-y curve data
#fish history name 'load' @tot_reac
#model step 416500
#model save 'lateral-load'
#""")
#print("               ======== PASSED ALL SCRIPTS SUCCESSFULLY ========")
```

```
#it.command("""
#zone cmodel assign strain-softening range group "Radial Tunnel1"
#zone property density 2500 bulk 2e8 shear 1e8 range group "Radial Tunnel1"
#zone property cohesion 2e6 friction 45 tension 2e5 dilation 10 range group "Radi
#zone property table-friction 'fri' table-cohesion 'coh' table-dilation 'dil' ran
#table 'fri' add (0, 45) (.05, 42) (.1, 40) (1, 40)
#table 'coh' add (0,2e6) (.05,1e6) (.1,5e5) (1,5e5)
#table 'dil' add (0, 10) (.05,  3) (.1,  0)
#""")

#top = reduce(np.logical_and, (np.sqrt(x**2+z**2)<_D_shaft, y<=np.amin(y)))
#za.set_group(top, "top")
#print(za.in_group("top").sum(), "zones in top group.")
```

10

Simulation of Pull-Tests for Fully Bonded Rock Reinforcement

Problem Statement

The most common method to determine properties for fully bonded rock reinforcement (such as grouted cable bolts, resin-grouted steel rebar, or rockbolts) is to perform pull-tests on small segments of grouted reinforcement in the field. Typically, segments 50 cm in length or longer are grouted into boreholes. The ends of these segments are pulled with a jack mounted to the surface of the tunnel and connected to cable via a barrel-and-wedge-type anchor. The force applied to the reinforcement and the deformation of the reinforcement are plotted to produce an axial force-deflection curve. From this curve, the peak shear strength of the grout bond is determined. The results for pull-tests on one-half meter segments of several types of cables are illustrated in Fig 1. These plots are expressed in terms of tons/m versus deformation in mm.

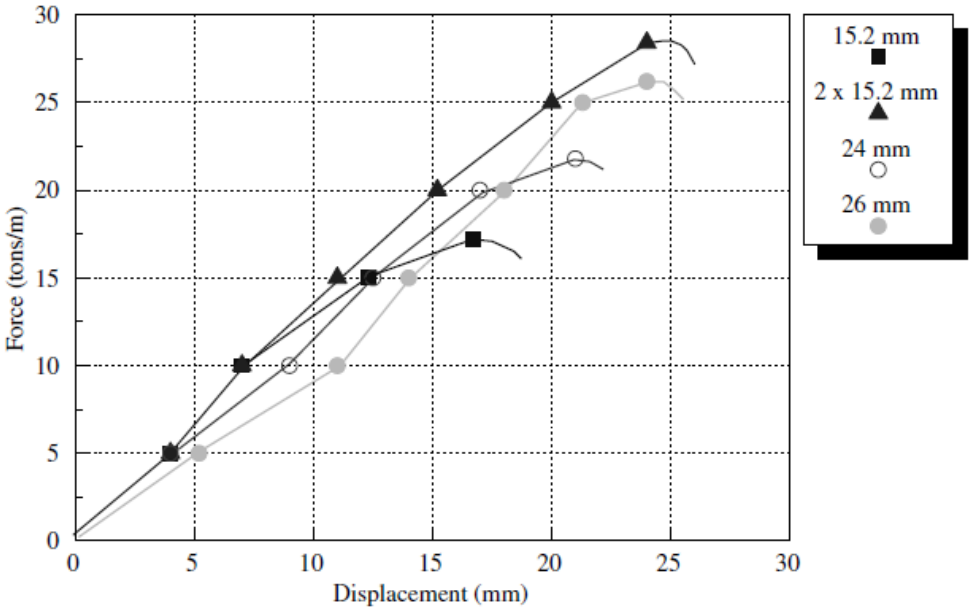


Figure 1: Field results for pull-tests on various types of cables for a bond length of 0.5 m and a water/cement ratio of 1/3.

Figure 1: Field results for pull-tests on various types of cables for a bond length of 0.5 m and a water/cement ratio of 1/3.

In this example application, FLAC3D is used to model pull-tests of this type. Two different approaches are available to simulate pull-tests using the structural elements in FLAC3D:

1. cable elements, which assume the grout behaves as an elastic-perfectly plastic material with confining stress dependence but no loss of strength after failure; and
2. modified pile elements, which account for changes in confining stress, strain-softening behavior of the grout, and rupture of the reinforcement.

One additional advantage of using the modified pile elements to model reinforcement behavior is that bending moment resistance can be included. This behavior is needed when the reinforcement is subjected to shearing, as described later in Pull-Test with Tensile Rupture and Shear Test on Rockbolts.

The purpose of the following examples is to demonstrate the use and capabilities of both approaches.

Approach 1: Modeling Rock Reinforcement Using Cable Elements

In the following sections, the pull-tests are simulated using the cable-element logic. The commands needed to create and view variables associated with cable elements are described in Cable Commands.

Pull-out Strength without Confinement

First, we consider the case where the confining stress dependence on the reinforcement shear-bond strength may be neglected. The cable properties required by FLAC3D's cable-bolt model must be extracted from the field pull-test curve. This is easily done when the field test data are presented in terms of force/unit length versus deformation, as shown in Figure 1. Assuming no yielding in the cable, the value of the grout shear stiffness, grout-stiffness, is simply the slope of the curve, with the ultimate bond strength, grout-cohesion, being the peak-pull strength value per unit length. This assumption is justified by the fact that the steel normally yields at stresses that are higher than those corresponding to the pull-out forces, as shown in Figure 1. For example, for a representative cable of 15 mm diameter and 0.5 m length and an (admissible) yield strength of 1200 MPa (assumed in the FLAC3D models in the following sections), the maximum yield force is 212 kN. This value is approximately 50% greater than the highest pull-out force of 28 tons/m $\times$ 0.5 = 140 kN, at which the grout fails in Figure 1. In addition, in these examples, the compliance of the reinforcement element and the rock are neglected relative to the compliance of the grout.

For example, all of the pull-test results shown here have roughly the same loading slope, so an average value of grout-stiffness is chosen for all:

This value of grout-stiffness is very low, indicating a rather poor grouting job for the cable. Typical grout-stiffness values would be approximately one order of magnitude or more higher than this.

The value of grout-cohesion for the single 15.2 mm wire is simply the peak shear resistance in tons/m. In this case, grout-cohesion

17.5 tons/m, or 17.5×10^4 N/m. To check whether this value of grout-cohesion is reasonable, it can be converted to grout shear strength by dividing by the approximate surface area of the wire (assuming the bond fails at the grout/cable interface). We find that the peak shear strength is 3.66 MPa. This value should roughly equal half the uniaxial compressive strength of the grout, indicating either a very poor grout, or that the cable was allowed to rotate during the pull-test, yielding artificially low grout shear-strength values.

The data file "Pull01.f3dat" is a simple representation of a pull-test using FLAC3D. The cable end-node is pulled at a small, constant y-oriented velocity—see Figure 2. A FISH function is used to sum the reaction forces and monitor nodal displacement generated by the pull-tests for comparison to field test results.

Figure 2: Sketch of geometry of FLAC3D model for pull-test.

A plot of the relation between pull force and cable displacement (histories force and disp, respectively) for the case of a single 15.2 mm cable is shown in Figure 3. This figure illustrates the general force-displacement behavior given in Figure 1. The peak force is reached at a displacement of approximately 17 mm. After this point, the cable is simply pulled out of the borehole in much the same fashion as a block sliding on a plane.

Figure 3: Pull force in N/m versus cable displacement in meters for the case of a single 15.2 mm grouted cable.

Figure 4 to 6 show the axial force distribution on the cable for displacements of 10 mm, 16.5 mm, and 17.5 mm, respectively. Note that cable-bond slip progresses rapidly after peak strength is reached at the first cable element. Superimposed on the axial forces are locations at which the grout bond is yielding. At 10 mm (Figure 4), the grout bond has not failed. At 16.5 mm (Figure 5), bond failure is initiated and rapidly propagates (Figure 6) down the entire cable length. At that stage, the force on the cable end is simply the sum of grout_cohesion

is the length of cable segments) for all n-slipping segments. If the embedded length were long enough, the cable axial force would eventually reach the yield force of the cable itself. The cable should then break when the extension strain

equals the ultimate breaking strain of the cable (generally around 3%). An example illustrating tensile rupture of the element is given in Pull-Test with Tensile Rupture.

break

Figure 6: At 17.5 mm deformation: Plot of axial force and cable grout yield points for pull-test simulation on a 15.2 mm cable bolt.

Pull-out Strength with Confinement

The cable shear bond strength will, in general, increase with increasing effective pressure (p') acting on the cable. A linear law is implemented in FLAC3D whereby the cable shear bond strength is defined as a constant (grout-cohesion) plus the effective pressure on the cable multiplied by the cable perimeter (grout-permeter) times the tangent of the friction angle (grout-friction). This pressure dependence is activated automatically in FLAC3D by issuing the cable properties grout-permeter and grout-friction. Note that in this case, the input data for grout-cohesion must correspond to the shear bond strength in a cable pull-test carried out without confining pressure. Numerical results of pull-tests on the 15.2 mm cable are presented in Figure 7 and 8 for a friction angle of 30° and two levels of initial confining pressure, namely p' = 2 and 4 MN/m^2. Compare these figures to Figure 3. These figures indicate an increasing failure level with increasing initial confining pressure, illustrating the frictional character of the cable-rock interface. Results for the pull-test with confined pressure on the 15.2 mm cable were obtained using the data files "Pull02.f3dat" and "Pull03.f3dat".

break

Figure 7: Pull-test on 15.2 mm cable, p' = 2 MN/m^2.

Figure 8: Pull-test on 15.2 mm cable, p' = 4 MN/m^2.

Approach 2: Modeling Rock Reinforcement Using Modified Pile Elements

In the following sections, the modified pile-element logic in FLAC3D is used to model rock reinforcement. The purpose of the following examples is to illustrate modeling procedures with the modified pile elements. In addition to a demonstration of the features that simulate pull-tests, this section also presents examples related

to the simulation of shear tests on rockbolts (see Shear Test on Rockbolts and Shear Test on Rockbolts with Rupture). The tests in this section assume that conditions are representative of a 25 mm diameter grouted bolt with grout properties the same as those defined in Pull-out Strength without Confinement.

Commands needed to create and view variables associated with the modified elements are described in Pile Commands. Note that the modified pile-element formulation is activated by giving the structure pile property rockbolt-flag on command.

Pull-out Strength without Confinement

The data file "Pulltest04.f3dat" creates a pull-test without confinement. The problem is equivalent to that described in Pull-out Strength without Confinement. Compared with the model in file "Pulltest01.f3dat", the current model considers a larger diameter bolt and a free length of bolt that extends out of the block. The load is applied at the tip of the reinforcement by prescribing a constant velocity, as was done before. A FISH function, force, is used to sum the reaction forces and monitor nodal displacement generated by the pull-test.

Figure 9 represents the relation between axial force (per unit length of cable) and displacement at the tip of the bolt, for a total displacement of 20 mm. The figure shows that the maximum pull-out load is comparable to the one obtained with Approach 1 in Pull-out Strength without Confinement (the slight difference may be attributed to the different bolt size and the free length of bolt included in this model). Figure 9 also shows the default perfectly plastic behavior of the grout once the maximum cohesion is exceeded.

Figure 9: Pull force in N/m versus cable displacement in meters for the case of single 25 mm grouted rockbolt.

Pull-Test with Displacement Weakening

The data file "Pulltest05.f3dat" defines a pull-test similar to the one described in the previous section, Pull-out Strength without Confinement, but with post-peak weakening of shear bond strength. The bond strength softening of the grout is

defined with the keyword coupling-cohesion-table, as described in [The relation between shear displacement and cohesion weakening is prescribed through table

(Note that softening of the friction of the grout could also be defined using the keyword coupling-friction-table).

Figure 10 represents the relation between axial force (per unit length of the bonded reinforcement) and displacement at the tip of the reinforcement for a total displacement of 40 mm. The figure clearly shows the weakening of cohesion after the maximum pull-out load is reached.

Figure 10: Pull force in N/m versus reinforcement displacement in meters for the case of a single 15.2 mm grouted reinforcement—with displacement weakening.

Pull-Test with Confinement

The data file "Pulltest06.f3dat" shows how a pull-test with confinement is simulated using the modified pile logic. The model is equivalent to the one discussed in Pull-out Strength with Confinement, this time for a confining stress, 4 MN/m^2. As described in Behavior of Shear Coupling Springs a linear law is implemented in the modified pile logic, whereby the reinforcement shear strength is defined as a constant (coupling-cohesion-shear) plus the effective pressure on the reinforcement multiplied by the reinforcement perimeter (perimeter) times a friction angle (coupling-friction-shear). This pressure dependence is activated automatically in FLAC3D by issuing the reinforcement properties (perimeter) and (coupling-friction-shear).

Figure 11 represents the relation between axial force (per unit length of the grouted reinforcement) and displacement at the tip of the reinforcement for a total displacement of 40 mm. The results are comparable to those shown in Figure 8.

Figure 11: Pull force in N/m versus reinforcement axial displacement in meters for the case of a single 25 mm grouted bolt—uniform 4 MN/m^2 confinement.

Pull-Test with Confinement — User-Defined Behavior

This example shows the use of the option coupling-confining-table to define a "mean" value σc of confining stress on the rockbolt for cases in which the principal stresses σx and σy perpendicular to the axis of the rockbolt are not the same.

Figure 12 shows the results obtained numerically for the relation between confining and deviatoric stresses in the case of a rockbolt with a diameter of 25 mm. In the figure, the horizontal axis represents the *normalized deviatoric stress*, defined as

while the vertical axis represents the *normalized confining stress*,

The relation was obtained from pull-test models in FLAC3D, considering different values of deviatoric stresses in the plane perpendicular to the axis bolt. The mean confining stress

using the expression

The data file "Pulltest07.f3dat" creates a model that illustrates the use of the coupling-confining-table option and the relation represented in Figure 12. The results obtained with this data file are shown in Figure 13. In this example, one of the principal stresses on the plane perpendicular to the axis of the rockbolt is zero. Note that the pull-out resistance in Figure 13 is greatly reduced compared to Figure 11 as a result of the zero stress in the lateral direction.

Figure 12: Example of relation between normalized confining stress and normalized deviator stress obtained numerically for a 25 mm rockbolt.

Figure 13: Rockbolt pull force in N/m versus rockbolt axial displacement in meters for the case of a single 25 mm grouted bolt—lateral confinement defined by the relationship represented in Figure 12.

Pull-Test with Tensile Rupture

This example shows the definition of limiting axial yield force and limiting axial strain for the rockbolt, using the options tensile-yield and tensile-failure-strain, respectively.

The data file "Pulltest08.f3dat" sets up a model that considers a limiting tensile force of 1.5×10^5 N and a limiting axial strain of 1×10^{-4}. The results from this model are shown in FigureFigure #pulltest-pile5. The diagram shows that the limiting tensile force is reached after a pull-out displacement of approximately

2.6 cm. Note that after this limiting force is reached, the pull-out force rapidly decreases to zero, indicating the rupture of the rockbolt.

Figure 14: Rockbolt pull force in N/m versus rockbolt axial displacement in meters for the case of a single 25 mm grouted bolt—with tensile rupture.

Shear Test on Rockbolts

This section and the next present two examples of shear tests. The tests are similar to the pull-tests described above, except that a velocity acting on a plane perpendicular to the axial direction of the bolt is applied to the top of the bolt.

The data file "Shear01.f3dat" shows how the model is created. Note that in the case of a shear test on rockbolts, values of stiffness and strength for the normal coupling springs (that were not needed in the pull-tests) need to be defined. Figure 15 shows a plot of shear force versus shear displacement. We select a stiffness value of 10^{10} N/m/m and cohesive strength of 10^{8} MPa/m in order to illustrate the shear behavior within a shear displacement range of 6 mm. These values do not represent a specific material and should be adjusted to fit experimental data.

The figure also includes a view of the model after the test. The large displacement of the rockbolt near the rock surface is the result of the yielding of the normal coupling springs, which simulates crushing of the rock.

Figure 15: Rockbolt shear force in N versus rockbolt shear displacement in meters for the case of a single 25 mm grouted bolt.

Shear Test on Rockbolts with Rupture

The data file "Shear02.f3dat" presents a model similar to the one described in Shear Test on Rockbolts, but with the added option of bending rupture. The failure is controlled by specified values of limiting (yield) bending moment and limiting tensile strain (these values are assigned using the keywords plastic-moment and tensile-failure-strain, as described in Pile Commands).

The results obtained with the data file "Shear02.f3dat" are shown in Figure 16. Note that the relation between applied force and lateral displacement shows

some oscillation after rupture. This oscillation is the normal response following the sudden rupture of the bolt.

```
Data Files


**Pull01.f3dat**


    ; ===================================================================
    ;   Simulation of pull-test for grouted reinforcement
    ; ===================================================================
    model new
    fish automatic-create off
    [global t =  'Pull Test for Grouted Cable Anchor ']
    [t += '(no external confinement, fric=0)']
    model title [t]
    ; Create a single rock block and set its material properties.
    zone create brick size 4 7 4 point 1 (0.4,0,0) point 2 (0,0.7,0) ...
                                 point 3 (0,0,0.4)
    zone cmodel assign elastic
    zone property bulk 5e9 shear 3e9
    ; Create a single cable and set its associated properties
    struct cable create by-line (0.2,0.0,0.2) (0.2,0.5,0.2) segments 10
    struct cable property cross-sectional-area 181e-6 young 98.6e9 ...
                          yield-tension 0.232e6 grout-stiffness 1.12e7 ...
                          grout-cohesion 1.75e5 grout-friction 0.0 ...
                          grout-perimeter 7.85e-2
    ; Fix free end of rock block and apply velocity to cable end
    zone face apply velocity-normal 0 range position-y 0
    struct node fix velocity-x                    range position-y 0
    struct node initialize velocity-x -1e-6 local range position-y 0
    call 'force' suppress ; FISH function to calculate reaction force on zones
```

```
    ; Set up histories for monitoring model behavior
    history interval 10
    fish history name 'force' @force
    struct node history name 'disp' displacement-y component-id 1
    ; Apply velocity to achieve total displacement of 2.0 cm
    model cycle 10000
    model save 'pull-1-1'
    model cycle 6500
    model save 'pull-1-2'
    model cycle 1000
    model save 'pull-1-3'
    model cycle 2500
    model save 'pull-1-4'
```

Pull02.f3dat

```
    ; ================================================================
    ;   Simulation of pull-test for grouted reinforcement
    ;     confinement 2 MPa
    ; ================================================================
    model new
    fish automatic-create off
    model title 'Pull Test for Grouted Cable Anchor - Confinement 2 MPa, fric=30'
    ; Create a single rock block and set its material properties.
    zone create brick size 4 7 4 point 1 (0.4,0,0) point 2 (0,0.7,0) ...
                                 point 3 (0,0,0.4)
    zone cmodel assign elastic
    zone property bulk 5e9 shear 3e9
    zone initialize stress xx -2e6 zz -2e6
    zone face apply stress-normal -2e6 range union position-x 0 ...
```

```
                                                position-x 0.4 position-z 0 ...
                                                position-z 0.4
    ; Create a single cable and set its associated properties
    struct cable create by-line (0.2,0.0,0.2) (0.2,0.5,0.2) segments 10
    struct cable property cross-sectional-area 181e-6 young 98.6e9 ...
                          yield-tension 0.232e6 grout-stiffness 1.12e7 ...
                          grout-cohesion 1.75e5 grout-friction 30.0 ...
                          grout-perimeter 7.85e-2
    ; Fix free end of rock block and apply velocity to cable end
    zone face apply velocity-normal 0 range position-y 0
    struct node fix velocity-x range position-y 0
    struct node initialize velocity-x -1e-6 local range position-y 0
    call 'force' suppress ; FISH function to calculate reaction force on zones
    ; Set up histories for monitoring model behavior
    history interval 10
    fish history name 'force' @force
    struct node history name 'disp' displacement-y component-id 1
    ; Apply velocity to achieve total displacement of 4.525 cm
    model cycle 45000
    ;
    model save 'pull-2'
```

Pull03.f3dat

```
    ; ================================================================
    ;   Simulation of pull-test for grouted reinforcement
    ;     confinement 4 MPa
    ; ================================================================
    model new
    fish automatic-create off
```

```
    model title 'Pull Test for Grouted Cable Anchor - Confinement 4 MPa, fric=30'
    ; Create a single rock block and set its material properties.
    zone create brick size 4 7 4 point 1 (0.4,0,0) point 2 (0,0.7,0) ...
                                point 3 (0,0,0.4)
    zone cmodel assign elastic
    zone property bulk 5e9 shear 3e9
    zone initialize stress xx -4e6 zz -4e6
    zone face apply stress-normal -4e6 range union position-x 0 position-x 0.4 ...
                                                position-z 0 position-z 0.4
    ; Create a single cable and set its associated properties
    struct cable create by-line (0.2,0.0,0.2) (0.2,0.5,0.2) segments 10
    struct cable property cross-sectional-area 181e-6 young 98.6e9 ...
                          yield-tension 0.232e6 grout-stiffness 1.12e7 ...
                          grout-cohesion 1.75e5 grout-friction 30.0 ...
                          grout-perimeter 7.85e-2
    ; Fix free end of rock block and apply velocity to cable end
    zone face apply velocity-normal 0 range position-y 0
    struct node fix velocity-x range position-y 0
    struct node initialize velocity-x -1e-6 local range position-y 0
    call 'force' suppress ; FISH function to calculate reaction force on zones
    ; Set up histories for monitoring model behavior
    history interval 10
    fish history name 'force' @force
    struct node history name 'disp' displacement-y component-id 1
    ; Apply velocity to achieve total displacement of 4.525 cm
    model cycle 45000
    ;
    model save 'pull-3'
```

Pull04.f3dat

```
; ================================================================
;   Simulation of pull-test for grouted reinforcement
;   using modified pile elements
; ================================================================
model new
fish automatic-create off
model title 'Pull-test using modified pile elements'
; Create a single rock block and set its material properties.
zone create brick size 4 4 6 point 1 (0.4,0,0) point 2 (0,0.4,0) ...
                             point 3 (0,0,0.6)
zone cmodel assign elastic
zone property bulk 5e9 shear 3e9
zone face apply velocity-normal 0.0 range position-z 0.6
; Create a pile element and assign properties
struct pile create by-line (0.2,0.2,0.1) (0.2,0.2,0.7) segments 12
struct pile property rockbolt-flag on
struct pile property young 200e9 poisson 0.25 cross-sectional-area 5e-4 ...
                     perimeter 0.08
struct pile property tensile-yield 2.25e5  ; ultimate tensile strength
struct pile property moi-y 2.0e-8 moi-z 2.0e-8 moi-polar 4.0e-8 ; 0.25*pi*r^4
struct pile property coupling-cohesion-shear 1.75e5 ...
                     coupling-stiffness-shear 1.12e7
struct pile property coupling-cohesion-normal 1.75e5 ...
                     coupling-stiffness-normal 1.12e7
; Set up pull out test
struct node fix  velocity-x range position-z 0.7
struct node init velocity-x 1e-6 local range position-z 0.7
call 'pileforce' suppress ; FISH function calculates reaction force on zones
; Set up histories for monitoring model behavior
```

```
    history interval 10
    fish history name 'force' @force
    struct node history name 'disp' displacement-z position (0.2,0.2,0.7)
    ; Achieve a total displacement of 2.0 cm
    model cycle 20000
    ;
    model save 'pull-4'
```

Pull05.f3dat

```
    ; ================================================================
    ;   Simulation of pull-test for grouted reinforcement
    ;   using modified pile elements - Softening of cohesion
    ; ================================================================
    model new
    fish automatic-create off
    model title 'Pull-test using modified pile elements - cohesion softening'
    ; Create a single rock block and set its material properties.
    zone create brick size 4 4 6 point 1 (0.4,0,0) point 2 (0,0.4,0) ...
                                 point 3 (0,0,0.6)
    zone cmodel assign elastic
    zone property bulk 5e9 shear 3e9
    zone face apply velocity-normal 0.0 range position-z 0.6
    ; Create a pile element and assign properties
    struct pile create by-line (0.2,0.2,0.1) (0.2,0.2,0.7) segments 12
    struct pile property rockbolt-flag on
    struct pile property young 200e9 poisson 0.25 cross-sectional-area 5e-4 ...
                         perimeter 0.08
    struct pile property tensile-yield   2.25e5  ; ultimate tensile strength
    struct pile property moi-y 2.0e-8 moi-z 2.0e-8 moi-polar 4.0e-8 ; 0.25*pi*r^4
```

```
    struct pile property coupling-cohesion-shear 1.75e5 ...
                         coupling-stiffness-shear 1.12e7
    struct pile property coupling-cohesion-normal 1.75e5 ...
                         coupling-stiffness-normal 1.12e7
    struct pile property coupling-cohesion-table 'cct'
    ; change in cohesion with relative shear displacement
    table 'cct' add (0,1.75e5) (0.025,1.75e4)
    ; Set up pull out test
    struct node fix velocity-x range position-z 0.7
    struct node initialize velocity-x 1e-6 local range position-z 0.7
    call 'pileforce' suppress ; FISH function calculates reaction force on zones
    ; Set up histories for monitoring model behavior
    history interval 10
    fish history name 'force' @force
    struct node history name 'disp' displacement-z position (0.2,0.2,0.7)
    ; Achieve a total displacement of 4.0 cm
    model cycle 40000
    ;
    model save 'pull-5'
```

Pull06.f3dat

```
    ; ================================================================
    ;   Simulation of pull-test for grouted reinforcement
    ;   using modified pile elements - Confinement 4 MPa
    ; ================================================================
    model new
    fish automatic-create off
    [global t =  'Pull-test using modified pile elements - ']
    [t += 'Confinement 4 MPa, fric=30']
```

```
model title [t]
; Create a single rock block and set its material properties.
zone create brick size 4 4 6 point 1 (0.4,0,0) point 2 (0,0.4,0) ...
                             point 3 (0,0,0.6)
zone cmodel assign elastic
zone property bulk 5e9 shear 3e9
zone face apply velocity-normal 0.0 range position-z 0.6
model largestrain on
; Create a pile element and assign properties
struct pile create by-line (0.2,0.2,0.1) (0.2,0.2,0.7) segments 12
struct pile property rockbolt-flag on
struct pile property young 200e9 poisson 0.25 cross-sectional-area 5e-4 ...
                     perimeter 0.08
struct pile property tensile-yield   2.25e5  ; ultimate tensile strength
struct pile property moi-y 2.0e-8 moi-z 2.0e-8 moi-polar 4.0e-8 ; 0.25*pi*r^4
struct pile property coupling-cohesion-shear 1.75e5 ...
                     coupling-stiffness-shear 1.12e7
struct pile property coupling-cohesion-normal 1.75e5 ...
                     coupling-stiffness-normal 1.12e7
struct pile property coupling-friction-shear 30.0
; Install in situ stresses
zone initialize stress xx -4e6 yy -4e6
zone face apply stress-normal -4e6 range union position-x 0 position-x 0.4 ...
                                            position-y 0 position-y 0.4
; Set up pull out test
struct node fix velocity-x range position-z 0.7
struct node init velocity-x 1e-6 local range position-z 0.7
call 'pileforce' suppress ; FISH function calculates reaction force on zones
; Set up histories for monitoring model behavior
history interval 10
```

```
    fish history name 'force' @force
    struct node history name 'disp' displacement-z position (0.2,0.2,0.7)
    ; Achieve a total displacement of 4.0 cm
    model cycle 40000
    ;
    model save 'pull-6'
```

Pull07.f3dat

```
    ; ===================================================================
    ;   Simulation of pull-test for grouted reinforcement
    ;   using modified pile elements - Confinement 4 MPa (w/table)
    ; ===================================================================
    model new
    fish automatic-create off
    [global t =  'Pull-test using modified pile elements - ']
    [t += 'Confinement 4 MPa (w/table)']
    model title [t]
    ; Create a single rock block and set its material properties.
    zone create brick size 4 4 6 point 1 (0.4,0,0) point 2 (0,0.4,0) ...
                                 point 3 (0,0,0.6)
    zone cmodel assign elastic
    zone property bulk 5e9 shear 3e9
    zone face apply velocity-normal 0.0 range position-z 0.6
    model largestrain on
    ; Create a pile element and assign properties
    struct pile create by-line (0.2,0.2,0.1) (0.2,0.2,0.7) segments 12
    struct pile property rockbolt-flag on
    struct pile property young 200e9 poisson 0.25 cross-sectional-area 5e-4 ...
                          perimeter 0.08
```

```
    struct pile property tensile-yield   2.25e5  ; ultimate tensile strength
    struct pile property moi-y 2.0e-8 moi-z 2.0e-8 moi-polar 4.0e-8 ; 0.25*pi*r^4
    struct pile property coupling-cohesion-shear 1.75e5 ...
                         coupling-stiffness-shear 1.12e7
    struct pile property coupling-cohesion-normal 1.75e5 ...
                         coupling-stiffness-normal 1.12e7
    struct pile property coupling-friction-shear 30.0
    ; define table for confining stress correction factor
    table 'cct' add (0,0.5) (0.3,0.48) (0.5,0.45) (0.6,0.39) (0.68,0.36)
    struct pile property coupling-confining-table 'cct'
    ; note : (snn-szz)/(snn+szz) is 1 , so cfac=0.36
    ; Install in situ stresses
    zone initialize stress xx -4e6
    zone face apply stress-normal -4e6 range union position-x 0 position-x 0.4
    ; Set up pull out test
    struct node fix velocity-x range position-z 0.7
    struct node initialize velocity-x 1e-6 local range position-z 0.7
    call 'pileforce' suppress ; FISH function calculates reaction force on zones
    ; Set up histories for monitoring model behavior
    history interval 10
    fish history name 'force' @force
    struct node history name 'disp' displacement-z position (0.2,0.2,0.7)
    ; Achieve a total displacement of 4.0 cm
    model cycle 40000
    ;
    model save 'pull-7'
```

Pull08.f3dat

```
    ; ================================================================
```

```
;   Simulation of pull-test for grouted reinforcement
;   using modified pile elements
;     Definition of failure due to maximum tensile strain or stress
; ===================================================================
model new
fish automatic-create off
[global t =  'Pull-test using modified pile elements - ']
[t += 'tensile strain causes rupture']
model title [t]
; Create a single rock block and set its material properties
zone create brick size 4 4 6 point 1 (0.4,0,0) point 2 (0,0.4,0) ...
                             point 3 (0,0,0.6)
zone cmodel assign elastic
zone property bulk 5e9 shear 3e9
zone face apply velocity-normal 0.0 range position-z 0.6
model largestrain on
zone mech damping combined
; Create a pile element and assign properties
struct pile create by-line (0.2,0.2,0.1) (0.2,0.2,0.7) segments 12
struct pile property rockbolt-flag on
struct pile property young 200e9 poisson 0.25 cross-sectional-area 5e-4 ...
                     perimeter 0.08
struct pile property moi-y 2.0e-8 moi-z 2.0e-8 moi-polar 4.0e-8 ; 0.25*pi*r^4
struct pile property coupling-cohesion-shear 1.75e5 ...
                     coupling-stiffness-shear 1.12e7
struct pile property coupling-cohesion-normal 1.75e5 ...
                     coupling-stiffness-normal 1.12e7
struct pile property coupling-friction-shear 30.0
struct pile property tensile-yield 1.5e5  ; ultimate tensile strength
struct pile property tensile-failure-strain 1.0e-4 ; ultimate tensile strain
```

```
    ; Install in situ stresses
    zone initialize stress xx -4e6 yy -4e6
    zone face apply stress-normal -4e6 range union position-x 0 position-x 0.4 ...
                                                position-y 0 position-y 0.4
    ; Set up pull out test
    struct node fix velocity-x range position-z 0.7
    struct node initialize velocity-x 1e-6 local range position-z 0.7
    call 'pileforce' suppress ; FISH function calculates reaction force on zones
    ; Set up histories for monitoring model behavior
    history interval 10
    fish history name 'force' @force
    struct node history name 'disp' displacement-z position (0.2,0.2,0.7)
    ; Achieve a total displacement of 2.0 cm
    model cycle 40000
    ;
    model save 'pull-8'
```

Shear01.f3dat

```
    ; ================================================================
    ;   Simulation of shear test for a grouted bolt
    ;   using modified pile elements
    ; ================================================================
    model new
    fish automatic-create off
    model title 'Shear test using pile elements'
    ; Create a single rock block and set its material properties
    zone create brick size 3 3 6 point 1 (0.3,0,0) point 2 (0,0.3,0) ...
                                 point 3 (0,0,0.6)
    zone face skin ; Label the model boundaries
```

```
zone cmodel assign elastic
zone property bulk 5e10 shear 3e10 density 2000
zone face apply velocity-normal 0 range group 'Bottom'
zone face apply velocity-normal 0 range group 'East' or 'West'
zone face apply velocity-normal 0 range group 'North' or 'South'
model largestrain on
zone mechanical damping combined
; Create a pile element and assign properties
struct pile create by-line (0.15,0.15,0.1) (0.15,0.15,0.628) segments 22
struct node group 'Top' range position-z 0.628 ; Tag the node at the top
                                                ; with a name
struct pile property rockbolt-flag on
struct pile property young 200e9 poisson 0.25 cross-sectional-area 5e-4 ...
                     perimeter 0.08
; ultimate tensile strength
struct pile property tensile-yield   2.25e5
struct pile property moi-y 2.0e-8 moi-z 2.0e-8 moi-polar 4.0e-8
;                          0.25*pi*r^4
struct pile property coupling-cohesion-shear 1.75e5 ...
                     coupling-stiffness-shear 1.12e7
struct pile property coupling-cohesion-normal 1.75e8 ...
                     coupling-stiffness-normal 1.0e9
struct pile property coupling-friction-shear 30.0
; Set up shear test (need to fix local nodal axes at the top node,
; to avoid axes rotating with element)
struct node fix system-local range group 'Top'
struct node initialize velocity-x 1e-6 local range group 'Top'
struct node fix velocity-x range group 'Top'
call 'shearforce' ; FISH function to calculate force on the grid
; Set up histories for monitoring model behavior
```

```
    history interval 100
    fish history name 'force' @force
    struct node history name 'disp' displacement-x position (0.15,0.15,0.628)
    ; Achieve a total displacement of 5.0 cm
    model cycle 50000
    ;
    model save 'shear-1'
```

Shear02.f3dat

```
    ; ================================================================
    ;   Simulation of shear test for a grouted bolt
    ;   using modified pile elements - Bending moment rupture
    ; ================================================================
    model new
    fish automatic-create off
    model title 'Shear test using pile elements - Bending moment rupture'
    ; Create a single rock block and set its material properties
    zone create brick size 3 3 6 point 1 (0.3,0,0) point 2 (0,0.3,0) ...
                                 point 3 (0,0,0.6)
    zone face skin ; Label the model boundaries
    zone cmodel assign elastic
    zone property bulk 5e10 shear 3e10 density 2000
    zone face apply velocity-normal 0 range group 'Bottom'
    zone face apply velocity-normal 0 range group 'East' or 'West'
    zone face apply velocity-normal 0 range group 'North' or 'South'
    model largestrain on
    zone mechanical damping combined
    ; Create a pile element and assign properties
    struct pile create by-line (0.15,0.15,0.1) (0.15,0.15,0.628) segments 22
```

```
struct node group 'Top' range position-z 0.628 ; Tag the node at the top
                                                ; with a name
struct pile property rockbolt-flag on
struct pile property young 200e9 poisson 0.25 cross-sectional-area 5e-4 ...
                     perimeter 0.08
; ultimate tensile strength
struct pile property tensile-yield   2.25e5
; ultimate moment and tensile strength
struct pile property plastic-moment   5e3 tensile-failure-strain   1e-4
struct pile property moi-y 2.0e-8 moi-z 2.0e-8 moi-polar 4.0e-8
;                         0.25*pi*r^4
struct pile property coupling-cohesion-shear 1.75e5 ...
                     coupling-stiffness-shear 1.12e7
struct pile property coupling-cohesion-normal 1.75e8 ...
                     coupling-stiffness-normal 1.0e10
struct pile property coupling-friction-shear 30.0
; Set up shear test (need to fix local nodal axes at the top node,
; to avoid axes rotating with element)
struct node fix system-local range group 'Top'
struct node initialize velocity-x 1e-6 local range group 'Top'
struct node fix velocity-x range group 'Top'
call 'shearforce' ; FISH function to calculate force on the grid
; Set up histories for monitoring model behavior
history interval 100
fish history name 'force' @force
struct node history name 'disp' displacement-x position (0.15,0.15,0.628)
; Achieve a total displacement of 2.0 cm
warning off
model cycle 20000
```

```
    ;
    model save 'shear-2'
```