PySimple1Gen OpenSees Command

By: Scott Brandenberg
Date: June 28th, 2004

PySimple1Gen $File1 $File2 $File3 $File4 $File5 <$File6>

The PySimple1Gen command constructs PySimple1 materials (Boulanger, 2003) for pre-defined zeroLength elements. The command requires five arguments, and supports an optional sixth argument, all of which are file names. The first file contains soil and pile properties required to define the PySimple1 materials. The second file contains information about the nodes that define the mesh. The third file contains information about the zeroLength elements that are to be assigned PySimple1 materials (hereafter called p-y elements). The fourth file contains information about the beam column elements that are attached to p-y elements. The fifth file is the output file to which the PySimple1 materials are written. The sixth file is the output file to which the applied patterns are written (optional).

The command has been structured such that File2, File3, File4, File5 and File6 can be sourced directly by OpenSees from within a master tcl file. Hence File2, File3 and File4 serve two purposes:

1. They provide information to PySimple1Gen to create the PySimple1 materials.
2. They can be sourced directly in a master tcl file to define the nodes, zeroLength elements for p-y materials, and pile elements, respectively.

Furthermore, File5 and File 6 serve the following purpose:

1. They can be sourced by OpenSees from within a master tcl file to define the PySimple1 materials and the applied patterns, respectively.

The intended use of the files is demonstrated in an example problem in the Appendix.

File1

The first input file, File1, contains soil and pile properties that are required to calculate the material properties for the PySimple1 materials, and optional information about p-multipliers, and applied patterns (either loads on the pile nodes, or displacements on the free ends of the p-y elements). Optional information is placed inside angle brackets (i.e. < Optional Information >). The format of File1 is as follows:

matType(1) z_t(1) z_b(1) γ'_t(1) γ'_b(1) Additional Arguments(1)
.
.
.
matType(N) z_t(N) z_b(N) γ'_t(N) γ'_b(N) Additional Arguments(N)
<sp or load zPattern_t(1) zPattern_b(1) PatternVal_t(1) PatternVal_b(1)
.
.
.
sp or load zPattern_t(N) zPattern_b(N) PatternVal_t(N) PatternVal_b(N)
mp zMp_t(1) zMp_b(1) MpVal_t(1) MpVal_b(1)
.
.
.
mp zMp(N) zMp_b(N) MpVal_t(N) MpVal_b(N)>

where,

b = pile diameter (meters).
zground = z-coordinate of ground surface (meters).
NPile = number of piles out-of-plane (if the analysis is in the y-z plane, then NPile is the number of piles along the x-axis).

matType = py1	Approximates Matlock's (1970) soft clay p-y relation.
matType = py2	Approximates API (1993) sand p-y relation.
matType = py3	Liquefied sand with normalized mobilized liquefied strength ratio (approximates API sand shape).
matType = py4	Nonliquefied crust over liquefied sand relation (approximates either Matlock soft clay p-y relation or API sand relation).

z_t = z-coordinate of top of sub-layer (meters).
z_b = z-coordinate of bottom of sub-layer (meters).
γ'_t = buoyant unit weight at top of sub-layer (kN/m^3).
γ'_b = buoyant unit weight at bottom of sub-layer (kN/m^3).

Additional arguments are required for each different material (i.e. each different matType), as summarized below:

matType = py1 Matlock (1970) soft clay p-y relation.

AdditionalArguments = b_t b_b cu_t cu_b e50_t e50_b Cd_t Cd_b <c_t c_b>

b_t = pile diameter (m) at depth z_t.
b_b = pile diameter (m) at depth z_b.
cu_t = undrained shear strength (kPa) at depth z_t.
cu_b = undrained shear strength (kPa) at depth z_b.
e50_t = shear strain at 50% shear stress for clay at depth z_t.
e50_b = shear strain at 50% shear stress for clay at depth z_b.
Cd_t = drag coefficient at depth z_t.
Cd_b = drag coefficient at depth z_b.
c_t = viscous damping term (optional) on the far-field (elastic) component of the displacement rate (velocity) at depth z_t (0.0 default).
c_b = viscous damping term (optional) on the far-field (elastic) component of the displacement rate (velocity) at depth z_b (0.0 default).

matType = py2 API (1993) sand p-y relation.

AdditionalArguments = b_t b_b phi_t phi_b Cd_t Cd_b <c_t c_b>

b_t = pile diameter (m) at depth z_t.
b_b = pile diameter (m) at depth z_b.
phi_t = friction angle (degrees) at depth z_t.
phi_b = friction angle (degrees) at depth z_b.
Cd_t = drag coefficient at depth z_t.
Cd_b = drag coefficient at depth z_b.
c_t = viscous damping term (optional) on the far-field (elastic) component of the displacement rate (velocity) at depth z_t (0.0 default).
c_b = viscous damping term (optional) on the far-field (elastic) component of the displacement rate (velocity) at depth z_b (0.0 default).

matType = py3 Liquefied sand with normalized residual strength ratio p-y relation.

AdditionalArguments = b_t b_b phi_t phi_b S/σ_v'_t S/σ_v'_b ru_t ru_b Cd_t Cd_b <c_t c_b>

b_t = pile diameter (m) at depth z_t.
b_b = pile diameter (m) at depth z_b.
phi_t = friction angle (degrees) at depth z_t (used only to calculate y_{50}).
phi_b = friction angle (degrees) at depth z_b (used only to calculate y_{50}).
S/σ_v'_t = residual strength ratio at depth z_t at $r_u = 1.0$.
S/σ_v'_b = residual strength ratio at depth z_b at $r_u = 1.0$.
ru_t = peak excess pore pressure ratio at coordinate z_t.
ru_b = peak excess pore pressure ratio at coordinate z_b.

Cd_t = drag coefficient at coordinate z_t.
Cd_b = drag coefficient at coordinate z_b.
c_t = viscous damping term (optional) on the far-field (elastic) component of the displacement rate (velocity) at depth z_t (0.0 default).
c_b = viscous damping term (optional) on the far-field (elastic) component of the displacement rate (velocity) at depth z_b (0.0 default).

matType = py4 User-specified p_{ult} and y_{ult}.

AdditionalArguments = type p_{ult}_t p_{ult}_b y_{50}_t y_{50}_b Cd_t Cd_b <c_t c_b>

type = 1 for approximation of Matlock's (1970) soft clay p-y relation
type = 2 for approximation of API (1993) sand p-y relation
p_{ult}_t = ultimate capacity of p-y element (kN/m) at depth z_t.
p_{ult}_b = ultimate capacity of p-y element (kN/m) at depth z_b.
y_{50}_t = relative displacement (soil displacement minus pile cap displacement) at which 50% of ultimate resistance is reached in a monotonic virgin loading cycle at depth z_t (meters).
Y_{50}_b = relative displacement (soil displacement minus pile cap displacement) at which 50% of ultimate resistance is reached in a monotonic virgin loading cycle at depth z_b (meters).
Cd_t = drag coefficient at depth z_t.
Cd_b = drag coefficient at depth z_b.
c_t = viscous damping term (optional) on the far-field (elastic) component of the displacement rate (velocity) at depth z_t (0.0 default).
c_b = viscous damping term (optional) on the far-field (elastic) component of the displacement rate (velocity) at depth z_b (0.0 default).

Applied Patterns

sp is a character tag identifying the subsequent fields on the line as defining a displacement pattern assigned to the free ends of the p-y elements.
load is a character string identifying the subsequent fields on the line as defining a load pattern assigned to the pile nodes.
zPattern_t = z-coordinate of top of applied displacement or load (meters).
zPattern_b = z-coordinate of bottom of applied displacement or load (meters).
PatternVal_t = applied incremental displacement (meters) or load pattern (kN/m) at coordinate zPattern_t.
PatternVal_b = applied incremental displacement (meters) or load pattern (kN/m) at coordinate zPattern_b.

P-Multipliers

mp is a character string identifying the subsequent fields on the line as defining a p-multiplier.
zMp_t = z-coordinate of top of p-multiplier distribution (meters).
zMp_b = z-coordinate of bottom of p-multiplier distribution (meters).
MpVal_t = p-multiplier at coordinate zMp_t.
MpVal_b = p-multiplier at coordinate zMp_b.

File2

The second input file, File2, contains information about the nodes that define the pile elements and the zeroLength elements that are to be assigned PySimple1 materials. The format of File2 is as follows:

node nodenum(1) y(1) z(1)
.
.
.
node nodenum(N) y(N) z(N)

where,

nodenum = number of node
y = y-coordinate of node
z = z-coordinate of node

File3

The third input file, File3, contains information about the zeroLength elements that are to be assigned PySimple1 materials. The format of File3 is as follows:

element zeroLength elenum(1) node1(1) node2(1) –mat matTag (1) <ExtraInput.....>
.
.
.
element zeroLength elenum(N) node1(N) node2(N) –mat matTag (N) <ExtraInput.....>

where,

elenum = element number
node1 = a node defining the zeroLength element
node2 = a node defining the zeroLength element
matTag = material tag to be associated with a PySimple1 material
ExtraInput..... is an optional text string that comes after matTag. The reason for allowing "ExtraInput" is to facilitate dual use of File3 in both the PySimple1Gen command, and in a master tcl file as demonstrated in the Appendix.

File4

The fourth input file, File4, contains information about the pile elements to which the p-y elements connect. This file is required to calculate the tributary length required to define each PySimple1 material, and for applying load patterns to pile nodes. The format of File4 is as follows:

element elementType elenum(1) node1(1) node2(1) <ExtraInput.....>
.
.
.
element elementType elenum(N) node1(N) node2(N) <ExtraInput.....>

elenum = element number

node1 = a node defining the pile element

node2 = a node defining the pile element

ExtraInput..... is an optional text string that comes after matTag. The reason for allowing "ExtraInput" is to facilitate dual use of File4 in both the PySimple1Gen command, and in a master tcl file as demonstrated in the Appendix.

File5

The output file, File5, contains the PySimple1 materials. The format of File5 is as follows:

##
Start PySimple1 Materials

uniaxialMaterial PySimple1 matTag(1) pyType(1) P_{ult}(1) y_{50}(1) Cd(1) c(1)
.
.
.
uniaxialMaterial PySimple1 matTag(N) pyType(N) P_{ult}(N) y_{50}(N) Cd(N) c(N)

End PySimple1 Materials
##

where,

matTag = material tag associated with the corresponding zeroLength element.
soilType = 1 Backbone of p-y curve approximates Matlock (1970) soft clay relation.
soilType = 2 Backbone of p-y curve approximates API (1993) sand relation.
P_{ult} = capacity of PySimple1 material (kN).
y_{50} = relative displacement at $0.5p_{ult}$ (m).
Cd = drag coefficient.
c = viscous damping term (optional) on the far-field (elastic) component of the displacement rate (velocity).

File 6

File 6 contains the Pattern that applies loads to the pile nodes, and/or displacements to the free ends of the p-y elements. The format of File6 is as follows:

```
########################################################################
## Begin Pattern File
load nodenum ForceValue 0.0 0.0
.
.
load nodenum ForceValue 0.0 0.0
sp nodenum 1 DisplacementValue
.
.
sp nodenum 1 DisplacementValue
## End Pattern File
########################################################################
```

Note that p-y elements are assumed to be oriented in the 1-direction, so the patterns are applied in the 1-direction.

File Format

The format of File1 must be strictly followed to prevent I/O error. Metadata and comment lines are not permitted for File 1. For cases in which both patterns (i.e. “sp” or “load”) and p-multipliers (i.e. “mp”) are applied, they may be specified in any order (i.e. p-multipliers may be intermixed with applied loads and displacements).

For File2, data is read for each line that begins with “node” and other lines are ignored. For File3 and File4, data is read for each line that begins with “element” and other lines are ignored. Hence, extra data and blank rows are permitted for File2, File3, and File4 as long as rows that contain irrelevant data do not begin with the string “node” for File2 or “element” for File3 and File4. The output files, File5 and File6, contain a header that explains that the PySimple1Gen program was used to create the file.

The PySimple1Gen command does not contain the functionality of tcl. For example, the programming features of tcl allow nodes to be defined using a loop, as demonstrated below:

```
# Create pile nodes, with double nodes for adding zeroLength soil springs
for {set i 0}{$i<=30}{incr i 1}{
set yDim1 [expr $i*$dy1-660]
node [expr $i*2+1]          0.          $yDim1
node [expr $i*2+2]          0.          $yDim1
node [expr $i*2+63]         -$dx        $yDim1
node [expr $i*2+64]         -$dx        $yDim1
node [expr $i*2+125]        $dx         $yDim1
node [expr $i*2+126]        dx          $yDim1
}
```

However, PySimple1Gen cannot recognize such loops. The node number and nodal coordinates must be numbers for PySimple1Gen; expressions are not permitted.

Linear Interpolation

At a given node, soil properties, p-multipliers and displacement patterns are linearly interpolated based on the location of the node and the location and associated soil properties and pattern values defined in File1. Distributed loads along the pile are integrated over the tributary length to obtain nodal loads. A node will be assigned a load pattern if any part of its tributary area overlaps with any part of the applied distributed load, even if the node itself lies outside of the region in which distributed loads were defined. P-y elements lying outside of the region of defined p-multipliers will be assigned a p-multiplier of 1.0.

APPENDIX 1: EXAMPLE PROBLEM

An example problem has been constructed to show how the structure of the input and output files relates to a simple soil-pile model. The purpose of the example problem is to illustrate much of the functionality of the PySimple1Gen command, and not to accurately model the response the pile. The small number of p-y elements in the example problem is likely insufficient to result in an accurate solution, but permits a clear means of demonstrating the function of the PySimple1Gen command.

The extended pile shaft shown in Figure 1 is composed of six beam column elements, and penetrates a soil profile consisting of clay overlying sand. The clay layer deforms under uniform shear strain such that its surface displacement is 0.2 m, and the sand layer exhibits no displacement. Additionally, a distributed load is applied to the top portion of the shaft above the ground surface. Material properties are shown in the figure.

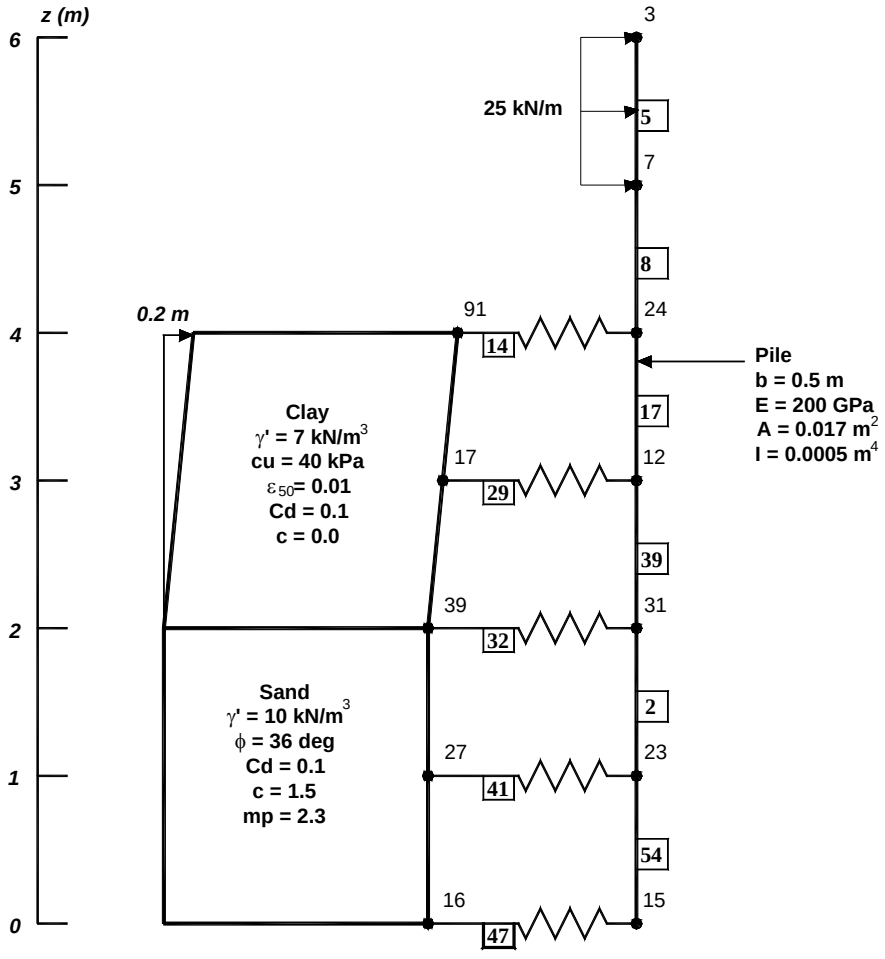

Figure 1: Example problem to illustrate PySimple1Gen command.

File1: Soil Properties Input File "SoilProp.tcl"

```
py1   4.0  2.0  7.0    7.0    0.5    0.5  40.0  40.0   0.01  0.01  0.1  0.1  0.0  0.0
py2   2.0  0.0  10.0  10.0  0.5    0.5  36.0  36.0   0.1    0.1    1.5  1.5
load  6.0   5.0     25.0   25.0
sp     4.0    2.0    0.2   0.0
sp     2.0    0.0    0.0   0.0
mp   2.0    0.0    2.3   2.3
```

File2: Nodes Input File "Nodes.tcl"

```
#####################################################################
## Begin Nodes

node 15  0  0
node 16  0  0
node  23  0  1
node  27  0  1
node  31  0  2
node  39  0  2
node  12  0  3
node  17  0  3
node  24  0  4
node  91  0  4
node  7    0   5
node  3    0   6

## End Nodes
#########################################################################
```

File3: Py Elements Input File "PyElements.tcl"

```
#####################################################################
## Begin p-y elements

element   zeroLength   47   15   16   –mat   5  –dir   1
element   zeroLength   41   23   27   –mat   17  –dir   1
element   zeroLength   32   31   39   –mat   19  –dir   1
element   zeroLength   29   12   17   –mat   23  –dir   1
element   zeroLength   14   24   91   –mat   7  –dir   1

## End p-y elements
#####################################################################
```

Note that the material tags (-mat ??) have been assigned arbitrarily. The PySimple1Gen command will use the arbitrary material tag to construct a PySimple1 material that is associated with the proper element.

File4: Pile Elements Input File "PileElements.tcl"

```
#####################################################################
## Begin beam-column elements

element   elasticBeamColumn   17   24  12  0.017   200000000.0   0.0005   1
element   elasticBeamColumn   39  12   31  0.017   200000000.0   0.0005   1
element   elasticBeamColumn   2    31   23   0.017  200000000.0   0.0005   1
element   elasticBeamColumn   54  23   15  0.017   200000000.0   0.0005   1
element   elasticBeamColumn   8    7   24   0.017   200000000.0   0.0005   1
element   elasticBeamColumn   5    3    7   0.017   200000000.0   0.0005   1

## End beam-column elements
#########################################################################
```

File5: Py Materials Output File "PyMaterials.tcl"

```
#############################################################################################
## Material Properties for py Elements

uniaxialMaterial PySimple1 5 2 419.773 0.00103901 0.1 1.5
uniaxialMaterial PySimple1 17 2 508.041 0.000652545 0.1 1.5
uniaxialMaterial PySimple1 19 1 187.165 0.0125 0.1 0
uniaxialMaterial PySimple1 23 1 83.5 0.0125 0.1 0
uniaxialMaterial PySimple1 7 1 30.4375 0.0125 0.1 0

## End Material Properties for py Elements
#############################################################################################
```

The material tags (i.e. the first number after the string Pysimple1) are the same tags that were previously assigned arbitrarily to the zeroLength elements.

File6: Pattern Output File "Pattern.tcl"

```
#############################################################################################
## Begin Pattern File

sp 16 1 0
sp 27 1 0
sp 39 1 0
sp 17 1 0.1
sp 91 1 0.2
load 7 12.5 0.0 0.0
load 3 12.5 0.0 0.0


## End Pattern File
#############################################################################################
```

Implementation in master tcl file

PySimple1Gen has be structured to receive certain arguments that are not required to define the p-y materials (i.e. "ExtraInput"), but are required to define the nodes and elements in OpenSees. So, as a matter of convenience, PySimple1Gen reads fields that are not essential to its function so that the files can serve two purposes:

1. They can be used arguments for the PySimple1Gen command.
2. They can be sourced directly from a master tcl script file to define nodes and elements.

For the example problem, assume that the input files (File1 through File4) required as arguments for PySimple1Gen are contained in the same folder as the master tcl file, and their file names are as follows:

File1: "SoilProp.txt"

File2: "Nodes.tcl"

File3: "PyElements.tcl"

File4: "PileElements.tcl"

Furthermore, assume that the output files (File5 and File6) are named:

File5: "PyMaterials.tcl"

File6: "Pattern.tcl"

Using the preceding information, the following master tcl file utilizes the files in the PySimple1Gen command, and also to define the nodes and elements in the domain.

```
#########################################################################
# Master file for py pushover analysis to illustrate PySimple1Gen
# command
#
# Created by Scott Brandenberg, April 30, 2004.
#########################################################################

wipe

set NumSteps 100
set StepSize [expr 1.0/$NumSteps]

#################################################################
# BUILD MODEL
#
model basic -ndm 2 -ndf 3

# define the nodes
source Nodes.tcl

# create geometric transformation for pile elements
geomTransf Linear 1
```

```
# define pile elements
source PileElements.tcl

# use PySimple1Gen command to generate PyMaterials.tcl and Pattern.tcl
PySimple1Gen "SoilProp.txt" "Nodes.tcl" "PyElements.tcl" "PileElements.tcl" "PyMaterials.tcl" "Pattern.tcl"

# define py elements and materials.  Always define materials before
# elements to prevent input error.  Note that PyMaterials.tcl was previously created using the
# PySimple1Gen command.
source PyMaterials.tcl
source PyElements.tcl


# Fix free ends of py elements against rotation and vertical displacement
fix 91 0 1 1
fix 17 0 1 1
fix 39 0 1 1
fix 27 0 1 1
fix 16 0 1 1

# Fix vertical deformation at pile tip
fix 15 0 1 0

#####################################################################
# NOW APPLY LOADING SEQUENCE AND ANALYZE (plastic)

# Create the pattern.  Note that Pattern.tcl was created using the PySimple1Gen command.
pattern Plain 1 Linear {
source Pattern.tcl
}

#create the recorder
recorder Node -file SoilDisplacement.dat -time -node 16 17 27 39 91 -dof 1 2 disp
recorder Node -file PileDisplacement.dat -time -node 3 7 12 15 23 24 31 -dof 1 2 disp
recorder Element -file PileElementRecorder.dat -time -ele 2 5 8 17 39 54  force
recorder Element -file PyElementRecorder.dat -time -ele 14 29 32 41 47 force

#####################################################################
# create the Analysis

constraints Penalty 1.e12 1.e12
test NormUnbalance 2e-3 100 0
numberer RCM
algorithm Newton
system ProfileSPD
integrator LoadControl $StepSize
analysis Static

#####################################################################
# analyze.

analyze $NumSteps

wipe  #flush output stream
```

APPENDIX 2: TECHNICAL INFORMATION

Calculating p_{ult} and y_{50}

The ultimate resistances of the p-y materials p_{ult}, were calculated in a manner similar to that described in LPile$^+$4.0m technical manual (Reese et al. 2000). The difference between the method used in PySimple1Gen and in LPile$^+$4.0m involves the treatment of layered soil systems. LPile$^+$4.0m utilizes the method developed by Georgiadis (1983) for calculating p_{ult} for layered soils. PySimple1Gen does not use the method of Georgiadis. Instead, p_{ult} is calculated based on the vertical effective stress at a given depth, with no consideration of the strength of overlying soil layers. The equations used in PySimple1Gen are included below:

<u>For soft clay (pyType = 1)</u>

$$pu1 = \left(3 + \frac{\sigma_v'}{c} + \frac{0.5}{z}\right) \cdot c \cdot b$$

$$pu2 = 9 \cdot c \cdot b$$

$$p_{ult} = \min(pu1, pu2)$$

where,

σ_v' = vertical effective stress
c = undrained shear strength
z = depth
b = pile diameter

<u>For sand (pyType=2)</u>

$$pu1 = \sigma_v' \cdot \frac{K_o \cdot z \cdot \tan(\phi) \cdot \sin(\beta)}{\tan(\beta - \phi) \cdot \cos(\alpha)} +$$

$$\sigma_v' \cdot \left(\frac{\tan(\beta)}{\tan(\beta - \phi)} \left(b + z \cdot \tan(\beta) \cdot \tan(\alpha)\right) + K_o \cdot z \cdot \tan(\beta) \cdot \left(\tan(\phi) \cdot \sin(\beta) - \tan(\alpha)\right) - K_a \cdot b \right)$$

$$pu2 = K_a \cdot b \cdot \sigma_v' \cdot \left(\tan^8(\beta) - 1\right) + K_o \cdot b \cdot \sigma_v' \cdot \tan(\phi) \cdot \tan^4(\beta)$$

$$p_{ult} = \min(pu1, pu2)$$

where,

σ_v' = vertical effective stress
Ko = coefficient of earth pressure at rest (default Ko = 0.4)
ϕ = friction angle of sand
$\beta = 45 + \phi/2$
$\alpha = \phi/2$
z = depth
b = pile diameter
K_a = coefficient of active earth pressure ($K_a = \tan^2(45 - \phi/2)$)

The relative displacement at $0.50p_{ult}$, y_{50}, is calculated in PySimple1Gen in the same way as discussed in the LPile+4.0m technical manual. For sand in LPile+4.0m, stiffness, k, is an input parameter. In PySimple1Gen, k is not an input parameter. Instead, it is calculated from a polynomial curve fit of the relationship between friction angle and stiffness for sand above the water table provided in Figure 3.29 in the LPile+4.0m technical manual.

<u>For liquefied sand (pyType = 3)</u>

for $r_u = 0$ (assuming undrained capacity with $r_u = 0$ is the same as drained capacity)

$$pu1_{ru0} = \sigma_v' \cdot \frac{K_o \cdot z \cdot \tan(\phi) \cdot \sin(\beta)}{\tan(\beta - \phi) \cdot \cos(\alpha)} +$$

$$\sigma_v' \cdot \left(\frac{\tan(\beta)}{\tan(\beta - \phi)} \left(b + z \cdot \tan(\beta) \cdot \tan(\alpha) \right) + K_o \cdot z \cdot \tan(\beta) \cdot \left(\tan(\phi) \cdot \sin(\beta) - \tan(\alpha) \right) - K_a \cdot b \right)$$

$$pu2_{ru0} = K_a \cdot b \cdot \sigma_v' \cdot \left(\tan^8(\beta) - 1 \right) + K_o \cdot b \cdot \sigma_v' \cdot \tan(\phi) \cdot \tan^4(\beta)$$

$$pu_{ru0} = \min(pu1_{ru0}, pu2_{ru0})$$

for $r_u = 1$

$$pu_{ru1} = 9 \cdot S \cdot b$$

for $0 < r_u < 1$

$$pu = pu_{ru0} + r_u \cdot \left[pu_{ru1} - pu_{ru0} \right]$$

where,

S = strength of sand mobilized against pile
b = pile diameter
r_u = peak excess pore pressure ratio
Ko = coefficient of earth pressure at rest (default Ko = 0.4)
ϕ = friction angle of sand

$\beta = 45 + \phi/2$
$\alpha = \phi/2$
z = depth
b = pile diameter
K_a = coefficient of active earth pressure ($K_a = \tan^2(45 - \phi/2)$)

For user-specified (pyType = 4)

p_{ult} and y_{50} are user-specified.

Tributary Length for p-y Elements

The p-y material properties must contain P_{ult} in units of force, not p_{ult} in units of force/length. P_{ult} represents the integral of p_{ult} over the tributary length. The integration was performed numerically in the following manner:

1. z_top (coordinate at the top of the tributary length) and z_bot (coordinate at the bottom of the tributary length) were calculated as the midpoints of the pile elements that share a node with the p-y element, provided that the pile elements had a p-y element attached to its other node as well. For example, pile elements that extend above the ground surface will not contribute any tributary length to a p-y element at or below the ground surface.
2. The tributary length was divided into 10 sublayers, each with thickness, dz.
3. P_{ult} was calculated as the sum of p_{ult}*dz over the ten sublayers from z_bot to z_top.

In the case when a p-y element lies near a boundary between two soil layers, such that its tributary length spans across the layer boundary, p_{ult} will be based on properties of both soil layers due to numerical integration over the tributary length. The type of p-y element (i.e. sand or clay) and y_{50} will be based on the material properties at the p-y element location. The numerical integration of p_{ult} over the tributary length will reduce errors associated with interface effects, however closely spaced p-y elements are recommended at layer interfaces. In the case when a p-y element lies at a boundary between two soil layers, the material type will be assigned the same type as the upper soil layer.

Tributary Length for Loads Applied to Pile Elements

Tributary lengths for loads applied to pile elements are calculated in the same manner as for p-y elements, except that beam column elements that do not attach to p-y elements can contribute tributary length to the calculation of nodal loads. For example, you may apply a distributed load to a beam column above the ground surface without any attached p-y elements (as was illustrated in the example). You may also apply a distributed load to a pile element with attached p-y elements.

Error Checking

1. The program checks that all five (or six) files can be opened. If not, the program will issue a warning and return without writing data to the output file(s).
2. The program checks that MatType = "py1", "py2", "py3" or "py4". If not, the program issues the following warning: "Invalid MatType in PySimple1Gen.", and then exits without writing data.
3. The program checks that the depth of each node lies within the depths specified in the soil properties file (File1). If not, a warning is issued and vertical stress is set to zero.

APPENDIX 3: REFERENCES

API(1993). *Recommended Practice for Planning, Design, and Constructing Fixed Offshore Platforms*. API RP 2A - WSD, 20th ed., American Petroleum Institute.

Boulanger, R. W. (2003). *The PySimple1 Material.* http://opensees.berkeley.edu.

Georgiadis, M. (1983). "Development of p-y curves for layered soils." *Proc., Geotechnical Practice in Offshore Engineering,* ASCE, pp. 536-545.

Matlock, H. (1970). "Correlations of design of laterally loaded piles in soft clay." *Proc. Offshore Technology Conference*, Houston, TX, Vol 1, No.1204, pp. 577-594.

Reese, L. C., Wang, S. T., Isenhower, W. M., Arrelaga, J.A., and Hendrix, J. A. (2000). *LPILE Plus Verion 4.0m,* Ensoft, Inc. Austin, TX.