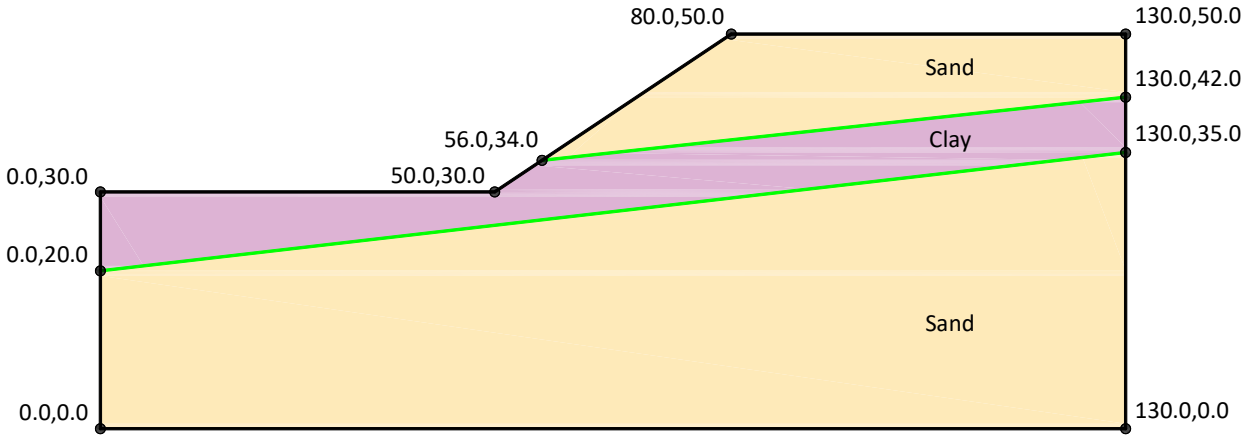
Using Python with HYRCAN

Created By: Roozbeh Geraili Mikola, PhD, PE
Email: hyrcan4geo@outlook.com
Web Page: www.geowizard.org



This tutorial will demonstrate how to use ***Python*** in ***HYRCAN***. Python is a general-purpose programming language with good support for scientific and numerical programming. The Python programming language is embedded inside ***HYRCAN*** and extended to allow models to be manipulated from Python programs. Currently Python environment contains the following extension modules:

- numpy 1.22.4 Array operations
- scipy 1.8.1 Collection of scientific libraries
- XlsxWriter 3.0.3 Writing files in the Excel 2007+ XLSX file format

Writing Python Code

There are two ways to write Python code in ***HYRCAN***. In the Python console or load the script text. In the console you write Python commands one by one, executing them by pressing ***Enter***, while script can contain more complex code made up of several lines, executed only when loaded. In this tutorial we use the latter. You can simply create a new file using any text editor (Notepad++ for instance) and type all the python commands in there and save the file with py extension (i.e. script_tut09.py). Now you will be able to load the script (Figure 1) into ***HYRCAN*** which gets executed by embedded Python interpreter. Please note that ***HYRCAN*** only recognizes the script files with extension ****.py*** and ****.hjs*** which corresponds to ***Python*** and ***JavaScript*** script files.

Select: *File* → 
Load Script

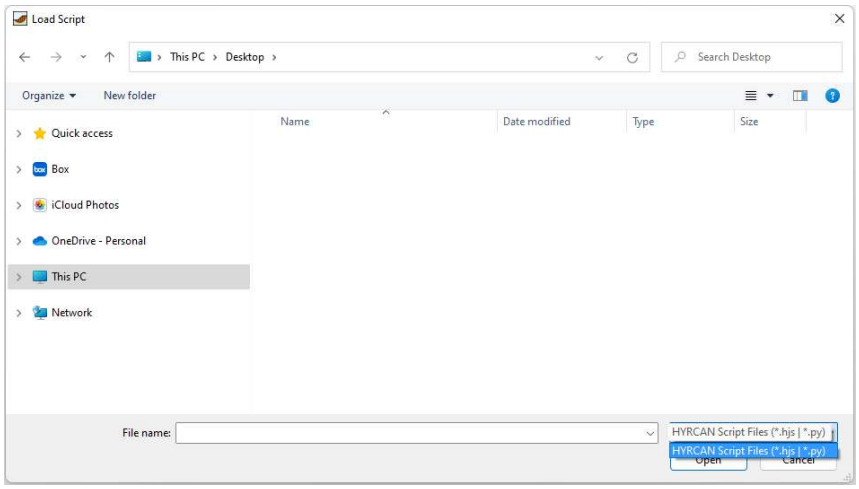


Figure 1- Load script dialog.

Working with Modules

First, we import the *hyrcan* module with the import statement. The *hyrcan* module defines the interaction between ***Python*** and ***HYRCAN***. You also will be able to import the *hyrcan* module as the shortened name like *hy*.

```
import hyrcan as hy
hy.command("""
newmodel()
set('failureDir','r2l')
extboundary(0,0,130,0,130,50,80,50,50,30,0,30,0,0)
matboundary(0,20,130,35)
matboundary(56,34,130,42)
definemat('ground','matID',1,'matName','Sand','uw',18,'cohesion',5,'friction',38)
definemat('ground','matID',2,'matName','Clay','uw',17,'cohesion',57,'friction',0)
assignsoilmat('matid',2,'atpoint',85,30)
definelimits('limit',20,65,'limit2',80,100)
set('Method','BishopSim','on')
""")
```

The above script creates a simple slope geometry using Python scripting. The *hy.command* (*hyrcan.command*) function is used to issue a series of ***HYRCAN*** commands. The triple quotes are used to define a multi-line string. Figure 1 shows the schematic view of the slope geometry generated by Python script.

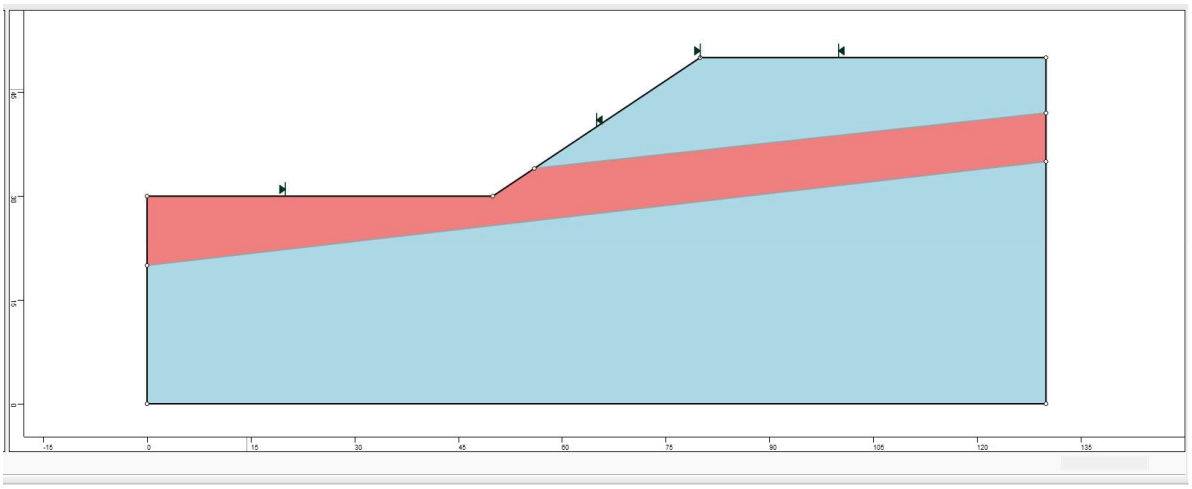


Figure 2- External boundary is created.

Now let's say we want to modify the property for the clay layer and perform the sensitivity analysis to estimate the factor of safety for multiple cohesion parameters. The following lines show how to modify the properties for the clay layer and perform the computation after each modification. *For* statement is used to repeat the process. Please note that 'silence' argument is used with *compute* command to prevent program from showing the information dialog at the end of each computation cycle. Otherwise, program would stop the process after first cycle and steps out of the for-loop.

```
fos_array = [-1,-1,-1,-1]
coh_array = [60,70,80,90]
cmd =
"definemat('ground','matID',2,'matName','Clay','uw',17,'cohesion',{coh},'friction'
,0)"
for i in range(0, 4):
    hy.command(cmd.format(coh=coh_array[i]))
    hy.command("compute('silence')")
    fos_array[i] = hy.min_fos('BishopSim')
```

hy.min_fos command is used to extract the minimum factor of safety for specified analysis method (i.e. Simplified Bishop). The method name could be a full name like: "Bishop Simplified", "Janbu Simplified", "Spencer", "GLE/Morgenstern-Price" or command name (short) like: "BishopSim", "JanbuSim", "Spencer", "GLE/M-P". List of all the functions in the *hyrcan* module is listed in the manual (Help -> Scripting Language -> Python)

Now you will be able to export the result into excel file using *XlsxWriter* module. XlsxWriter is a Python module that can be used to write text, numbers, formulas and hyperlinks to multiple worksheets in an Excel 2007+ XLSX file. XlsxWriter module is already install in the program's Python environment. More information about the XlsxWriter module can be found at https://xlsxwriter.readthedocs.io/. The following commands shows how to export the output from the program into the excel file. Figure 3 illustrates the output excel file created by XlsxWriter module.

```
import xlsxwriter

# Create a new XlsxWriter Workbook object
workbook = xlsxwriter.Workbook('coh_vs_fos_plot.xlsx')
# Add a new worksheet to a workbook
worksheet = workbook.add_worksheet()
# Create a new Format object to formats cells in worksheets
bold = workbook.add_format({'bold': 1})
# Add the worksheet data that the charts will refer to.
headings = ['Cohesion (kN/m2)', 'FOS']
# Write a row of data
worksheet.write_row('A1', headings, bold)
# Write a column of data
worksheet.write_column('A2', coh_array)
worksheet.write_column('B2', fos_array)
# Create a scatter chart sub-type with straight lines and markers.
chart = workbook.add_chart({'type': 'scatter',
                            'subtype': 'straight_with_markers'})
# Configure the first series.
chart.add_series({
    'name': '=Sheet1!$B$1',
    'categories': '=Sheet1!$A$2:$A$7',
    'values': '=Sheet1!$B$2:$B$7',
})
# Add a chart title and some axis labels.
chart.set_title({'name': 'Results of analysis'})
chart.set_x_axis({'name': 'Cohesion (kN/m2)', 'min': 50, 'max': 100})
chart.set_y_axis({'name': 'Factor of Safety'})
# Set an Excel chart style.
chart.set_style(11)
# Insert the chart into the worksheet (with an offset).
worksheet.insert_chart('D2', chart, {'x_offset': 25, 'y_offset': 10,
'x_scale': 1.5, 'y_scale': 1.5})
# Close the Workbook object and write the XLSX file
workbook.close()
```

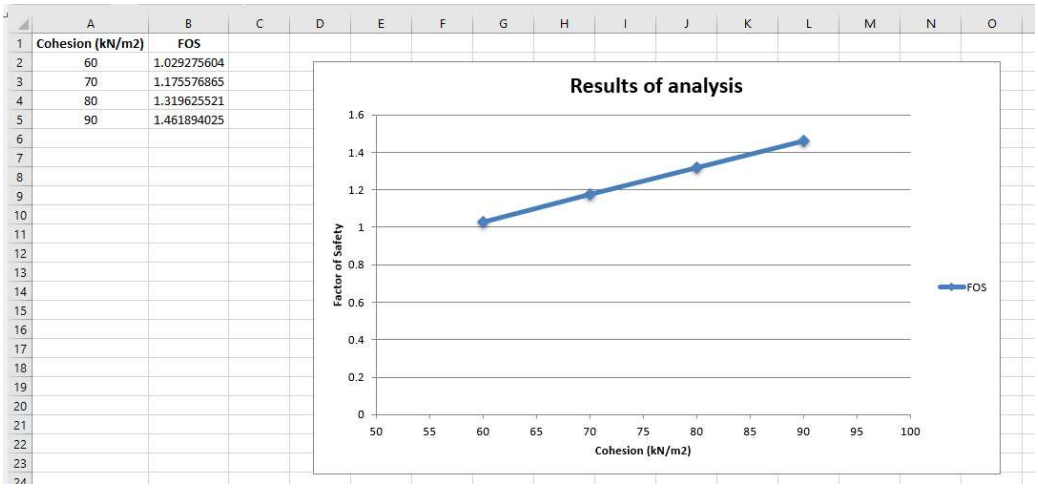


Figure 3- Exported excel file that summarizes cohesion versus fos values.

Script

The Python source codes for this example are listed below (script_tut09.py).

```
# Name: script_tut09.py
# Import hyrcan and xlsxwriter modules
import hyrcan as hy
import xlsxwriter

# Issue series of commands to create the initial slope geometry
hy.command("""
newmodel()
set('failureDir','r2l')
extboundary(0,0,130,0,130,50,80,50,50,30,0,30,0,0)
matboundary(0,20,130,35)
matboundary(56,34,130,42)
definemat('ground','matID',1,'matName','Sand','uw',18,'cohesion',5,'friction',38)
definemat('ground','matID',2,'matName','Clay','uw',17,'cohesion',57,'friction',0)
assignsoilmat('matid',2,'atpoint',85,30)
definelimits('limit',20,65,'limit2',80,100)
set('Method','BishopSim','on')
""")

# Initialize the fos and cohesion arrays
fos_array = [-1,-1,-1,-1]
coh_array = [60,70,80,90]
# Initialize the command text with the placeholder using curly brackets {}
cmd = "definemat('ground','matID',2,'matName','Clay','uw',17,'cohesion',{coh},'friction',0)"
# Loop through the cohesion values and perform the computation and store the fos value
for i in range(0, 4):
    hy.command(cmd.format(coh=coh_array[i]))
    hy.command("compute('silence')")
    fos_array[i] = hy.min_fos('BishopSim')

# Create a new XlsxWriter Workbook object
workbook = xlsxwriter.Workbook('coh_vs_fos_plot.xlsx')
# Add a new worksheet to a workbook
worksheet = workbook.add_worksheet()
# Create a new Format object to formats cells in worksheets
bold = workbook.add_format({'bold': 1})
# Add the worksheet data that the charts will refer to.
headings = ['Cohesion (kN/m2)', 'FOS']
# Write a row of data
worksheet.write_row('A1', headings, bold)
# Write a column of data
worksheet.write_column('A2', coh_array)
worksheet.write_column('B2', fos_array)
# Create a scatter chart sub-type with straight lines and markers.
chart = workbook.add_chart({'type': 'scatter',
                            'subtype': 'straight_with_markers'})
# Configure the first series.
chart.add_series({
    'name': '=Sheet1!$B$1',
    'categories': '=Sheet1!$A$2:$A$7',
    'values': '=Sheet1!$B$2:$B$7',
})
# Add a chart title and some axis labels.
chart.set_title({'name': 'Results of analysis'})
chart.set_x_axis({'name': 'Cohesion (kN/m2)', 'min': 50, 'max': 100})
chart.set_y_axis({'name': 'Factor of Safety'})
# Set an Excel chart style.
chart.set_style(11)
# Insert the chart into the worksheet (with an offset).
worksheet.insert_chart('D2', chart, {'x_offset': 25, 'y_offset': 10, 'x_scale': 1.5, 'y_scale':
1.5})
# Close the Workbook object and write the XLSX file
workbook.close()
```