APPENDIX

A1. Visual Basic Analysis code to generate multiple FEMAP neutral files containing different concrete and soil material properties:

```
Sub Main()
    Dim App As Object
    Set App = GetObject(, "femap.model")
    Dim Mat As Object
    Set Mat = App.feMatl

#Set some constant or starter variables here
#Worksheets(NUMBER).Cells(Row,Column).Value
        Directory = "C:\MASTERS\MODELS\"
        Folder = "SSI_Pile_Diameter_20cm_6m_0.5%"
        Soil_ID = 3
        Conc_ID = 1
        Sheet = 1
        Row = 2
        Name = Worksheets(Sheet).Cells(Row, 9).Value
        While Row < 83
            Name = Worksheets(Sheet).Cells(Row, 9).Value
            Mat.Get (Soil_ID)
                Mat.mval(0) = Worksheets(Sheet).Cells(Row, 5).Value #E_val
                Mat.mval(54) = Worksheets(Sheet).Cells(Row, 6).Value #CS
            Mat.Put (Soil_ID)
            Mat.Get (Conc_ID)
                Mat.mval(0) = Worksheets(Sheet).Cells(Row, 7).Value #E_val
                Mat.mval(54) = Worksheets(Sheet).Cells(Row, 8).Value #CS
            Mat.Put (Conc_ID)

rc = App.feFileWriteNeutral2(0, Directory & Folder & "\" & Folder & Name, False,
True, False, True, True, False, False, False, True, True, 8, 9, 0)
            Row = Row + 1
        Wend
End Sub
```

A2. Python code for analysing multiple Reconan FEA models (Multi-run):

```
import os,shutil
from subprocess import Popen, PIPE
FOLDER = "SSI_Pile_Diameter_20cm_6m_0.5%"
PATH_mod = "C:\\MASTERS\\MODELS\\"+FOLDER+"\\"
Names = ["50M20k20G20M.neu",...,"150M150k60G60M.neu"]
Reconan = "C:\\MASTERS\\MODELS\\"+FOLDER+"\\ReconanFEA_v3.04_NoEYE.exe"
Reconan = Reconan.replace("\\","/")
Reconan_Origin = "C:\\MASTERS\\ReconanFEA_v3.04_NoEYE.exe"
if os.path.exists(Reconan) == False:
    shutil.copy(Reconan_Origin,Reconan)

for I in range (0,len(Names)):
    name    = PATH_mod + Names[I]
    NEU_inp = PATH_mod + "Input.NEU"
    neu = "\""+NEU_inp +"\"\n"
    os.rename(name,NEU_inp)

    print("doing:",Names[I])
    process = Popen(Reconan,stdin=PIPE,text=True)
    print(process.communicate(input=neu))

    os.rename(PATH_mod + "Input.NEU",PATH_mod +
Names[I].replace(".neu","")+".neu")
    os.rename(PATH_mod + "Input_OUT.neu",PATH_mod +
Names[I].replace(".neu","")+"_OUT.neu")
    os.rename(PATH_mod + "Input_ERROR_FILE.dat",PATH_mod +
Names[I].replace(".neu","")+"_ERR.dat")
    os.rename(PATH_mod + "Input_FORCES.dat",PATH_mod +
Names[I].replace(".neu","")+"_FORCES.dat")
    os.rename(PATH_mod + "Input_FORCESDofs.dat",PATH_mod +
Names[I].replace(".neu","")+"_Dofs.dat")
    os.rename(PATH_mod + "Input_TIME.dat",PATH_mod +
Names[I].replace(".neu","")+"_TIME.dat")

print("MULTI RUN IS DONE WITH FOLDER: ",FOLDER)
```

A3. Python code for extracting output displacements and forces:

```
import xlwings, numpy
FOLDER = "C:\\MASTERS\\MODELS\\SSI_Pile_Diameter_20cm_6m_0.5%\\"

neu_files = ["50M20k20G20M.neu"..."150M150k60G60M.neu"]

class backend:

    Alphabet = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'AA',
'AB', 'AC', 'AD', 'AE', 'AF', 'AG', 'AH', 'AI', 'AJ', 'AK', 'AL', 'AM',
'AN', 'AO', 'AP', 'AQ', 'AR', 'AS', 'AT', 'AU', 'AV', 'AW', 'AX', 'AY',
'AZ']
    def getdisplacement(F,Forces:list,Disp:list):
        if not F in Forces:
            print("COULD NOT FIND FORCE ",F," in ",Forces)
```

```
            return "OOPS " + str(F)
        if not len(Forces) == len(Disp):
            raise KeyError(f"Lengths do not match {len(Forces) = } and
{len(Disp) = }")
        else:
            for I in range(len(Forces)):
                if Forces[I] == F:
                    return Disp[I]
    def GetNode(line_str):
                return line_str.split(sep=",")[0]
    def GetDispArray(line):
        Arr     = line.split(" ")
        Val_Arr = []
        for I in Arr:
            if not I == "":
                if not I == "\n":
                    Val_Arr.append(I.replace("\n",""))
        return Val_Arr
    def GetXYZArray(DispArr:list,Direction:str):
        I = 0
        Disp = []
        for direction in DispArr[0]:
            if direction == Direction:
                break
            I = I + 1
        for line in range(1,len(DispArr)):
            Disp.append(float(DispArr[line][I]))
        return Disp
    def avg(Arr):
        return sum(Arr)/len(Arr)
    def interpolate(X_val,X_Arr,Y_Arr):
        X1 = X2 = Y1 = Y2 = 0
        if X_val <=0:
            return "ERROR"
        elif not len(X_Arr) == len(Y_Arr):
            print("Arrays not compatible lengths")
            return 9999
        else:
            for I in range(0,len(X_Arr)-1):
                X1 = X_Arr[I]
                X2 = X_Arr [I+1]
                if X1 < X_val and X_val <= X2:
                    Y1 = Y_Arr[I]
                    Y2 = Y_Arr[I+1]
                    return Y1 + (Y2-Y1)/(X2-X1)*(X_val-X1)
    def cell(Letter,Number):
        return str(Letter+str(Number))
class Neutral:
    def Get_Loaded_Nodes(Neutral):
        I = 0
        LoadedNodes = []
        with open(Neutral) as NEU:
            STATE = False
            for line in NEU:
                I = I + 1
                if line == "   507\n":
                    Line507   = I
                    Line_node = I + 22
                    STATE = True
                if STATE == True:
```

```
                    node = (backend.GetNode(line))
                    if I == Line_node:
                        if node == str(-1):
                            break
                        else:
                            LoadedNodes.append(node)
                            Line_node = Line_node + 7
        NEU.close()
        return LoadedNodes
    def NumberOfCases(Output):
        CaseNr = 0
        with open(Output) as OUT:
            for line in OUT:
                if line[0:4] == "Case":
                    CaseNr = CaseNr + 1
                if line == "T1 Translation\n":
                    break
        return CaseNr
    def T1_Traslation(Output,Nodes,CaseNr):
        if not type(Nodes) == list:
            raise TypeError("Nodes must be an array, please provide an
array.")
        Case = 0
        Displacement = []
        Nodes_displaced = []
        STATE = False
        with open(Output) as OUT:
            LINE = 0
            for line in OUT:
                LINE = LINE+1
                if line == "T1 Translation\n":
                    Case = Case + 1
                    if Case == CaseNr:
                        STATE = True
                if STATE == True:
                    try:
                        node = str(line.split(",")[0])
                        if node == str(-1):
                            break
                        disp = float(line.split(",")[1])
                        if str(int(node)) in Nodes:
                            Nodes_displaced.append(float(node))
                            Displacement.append(float(disp))
                    except:
                        continue
        OUT.close()
        return Displacement, Nodes_displaced
    def FX(Data):
        DISP = []
        with open(Data) as DAT:
            STATE = False
            LINE = 0
            for line in DAT:
                LINE = LINE + 1
                if LINE == 10:
                    STATE = True
                if STATE == True:
                    DISP.append(backend.GetDispArray(line))
        DAT.close()
        FX = backend.GetXYZArray(DISP,"Fx")
```

```
        return FX
Alphabet = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'AA',
'AB', 'AC', 'AD', 'AE', 'AF', 'AG', 'AH', 'AI', 'AJ', 'AK', 'AL', 'AM',
'AN', 'AO', 'AP', 'AQ', 'AR', 'AS', 'AT', 'AU', 'AV', 'AW', 'AX', 'AY',
'AZ']

RESULTS_xlsx = xlwings.Book("C:/MASTERS/Results.xlsx")
RESULTS_xlsx.sheets.add(FOLDER.replace("C:\\MASTERS\\MODELS\\SSI_Pile_Diame
ter_","").replace("\\",""))
RESULTS_xlsx.sheets.add(str(FOLDER.replace("C:\\MASTERS\\MODELS\\SSI_Pile_D
iameter_","").replace("\\","")+"_Curve"))
RESULT_Sheet =
RESULTS_xlsx.sheets[FOLDER.replace("C:\\MASTERS\\MODELS\\SSI_Pile_Diameter_
","").replace("\\","")]
Curve_Sheet =
RESULTS_xlsx.sheets[FOLDER.replace("C:\\MASTERS\\MODELS\\SSI_Pile_Diameter_
","").replace("\\","")+"_Curve"]
RESULT_Sheet[backend.cell("A",1)].value = "Model Name"
RESULT_Sheet[backend.cell("B",1)].value = "Increment"
RESULT_Sheet[backend.cell("C",1)].value = "Fx Max"
RESULT_Sheet[backend.cell("D",1)].value = "Disp Max"
RESULT_Sheet[backend.cell("E",1)].value = "Fx half"
RESULT_Sheet[backend.cell("F",1)].value = "Disp half"

for I in range(len(neu_files)):

    NEU = FOLDER + neu_files[I].replace(".neu",".neu")
    OUT = FOLDER + neu_files[I].replace(".neu","_OUT.neu"    )
    DAT = FOLDER + neu_files[I].replace(".neu","_FORCES.dat" )

    NODES = Neutral.Get_Loaded_Nodes(NEU)
    CASES = Neutral.NumberOfCases(OUT)-0 # DEPENDS
    DISP, NODE = Neutral.T1_Traslation(OUT,NODES,CASES)
    print(DISP)
    if max(DISP) == 0:
        print("Case ",CASES,"'s DISP array is empty using",f"{CASES-1 = }")
        CASES = CASES - 1
        DISP, NODE = Neutral.T1_Traslation(OUT,NODES,CASES)
        print(DISP)

    print("Number of CASES: ",CASES)
    FX    = Neutral.FX(DAT)
    if FX[len(FX)-1] == 0:
        print("Removed zero entry at last location of the FX arr.")
        del(FX[len(FX)-1])

    FX_MAX, Increments = max(FX),len(FX)
    FX_HALF = FX_MAX/2
    dDISP = round(backend.avg(DISP),6)/Increments
    DISPLACEMENT = [round(dDISP*(I+1),6) for I in range(Increments)]
    DISPLACEMENT.insert(0,0)
    FX.insert(0,0)

    Displacement_MAX = backend.getdisplacement(FX_MAX,FX,DISPLACEMENT)
    Displacement_HALF = backend.interpolate(FX_HALF,FX,DISPLACEMENT)

    if FX_MAX == FX[len(FX)-1]:
        print("GOOD, maximum displacement is last displacement in disp
array.")
```

```
        str_displacement = neu_files[I].replace(".neu"," Displacement")
        str_forces       = neu_files[I].replace(".neu"," Forces")
        str_neutral      = neu_files[I]
    else:
        print("CAUTION*, maximum displacement is NOT last displacement in
disp array. an * will be added to excel names")
        str_displacement ="*"+neu_files[I].replace(".neu"," Displacement")
        str_forces       ="*"+neu_files[I].replace(".neu"," Forces")
        str_neutral      ="*"+neu_files[I]
    if min(FX[0:len(FX)-1]) < 0:
        str_displacement = "!"+str_displacement
        str_forces       = "!"+str_forces
        str_neutral      = "!"+str_neutral

    print()
    Curve_Sheet[backend.cell("A",1+I*2)].value = str_displacement
    Curve_Sheet[backend.cell("A",2+I*2)].value = str_forces

    RESULT_Sheet[backend.cell("A",I+2)].value = str_neutral
    RESULT_Sheet[backend.cell("B",I+2)].value = Increments
    RESULT_Sheet[backend.cell("C",I+2)].value = FX_MAX
    RESULT_Sheet[backend.cell("D",I+2)].value = Displacement_MAX
    RESULT_Sheet[backend.cell("E",I+2)].value = FX_HALF
    RESULT_Sheet[backend.cell("F",I+2)].value = Displacement_HALF

    for k in range(0,len(FX)):
        Curve_Sheet[backend.cell(Alphabet[k+1],1+I*2)].value =
DISPLACEMENT[k]
        Curve_Sheet[backend.cell(Alphabet[k+1],2+I*2)].value = FX[k]
    RESULTS_xlsx.save()

    print("DOING FILE ",NEU)
    print("FX = ",FX)
    print("DISPLACEMENT = ",DISPLACEMENT,"\n")
    print(f"{FX_MAX = }, {FX_HALF = }")
    print(f"{Displacement_MAX = }, {Displacement_HALF = }")
    print("- "*40,"\n")
```

A4.1. Horizontal failure force proposed formula using POLYREG-HYT-3:

$$
\begin{aligned}
Fx\ Max = {} & +6.75566 \times 10^{-2} * D * CS_{Soil} - 3.65967 \times 10^{-4} * {E_{Con}}^{3} - 1.38088 \times 10^{1} * RR^{2} \\
& + 1.19355 \times 10^{-2} * D * RR * CS_{Soil} - 4.36401 \times 10^{-2} * E_{Soil} + 4.52060 \times 10^{-2} * D * H \\
& + 1.23276 \times 10^{-2} * RR * {CS_{Con}}^{2} + 4.24305 * RR^{3} + 3.42707 \times 10^{-4} * H * {CS_{Soil}}^{2} \\
& + 1.45597 \times 10^{-3} * D^{3} - 5.11971 \times 10^{-2} * {CS_{Con}}^{2} + 1.95308 * D * RR \\
& + 2.54949 \times 10^{-2} * {E_{Con}}^{2} - 2.67408 \times 10^{-4} * D * CS_{Soil} * CS_{Con} - 6.28916 \times 10^{-2} \\
& * RR^{2} * CS_{Soil} - 3.92401 \times 10^{-2} * H^{2} * RR - 1.68938 \times 10^{-1} * D^{2} + 1.37599 \times 10^{-1} * D \\
& * CS_{Con} + 5.63646 \times 10^{-5} * {E_{Soil}}^{2} * E_{Con} + 7.28580 \times 10^{-5} * D^{2} * E_{Soil} \\
& + 3.69146 \times 10^{-4} * D^{2} * CS_{Soil} - 4.85409 \times 10^{-2} * H * CS_{Soil} - 3.92286 \times 10^{-4} * H \\
& * CS_{Soil} * CS_{Con} + 4.76295 \times 10^{-3} * RR * {E_{Con}}^{2} - 3.89024 \times 10^{-1} * D * RR^{2} \\
& - 2.28042 \times 10^{-1} * RR * E_{Con} - 7.56234 \times 10^{-4} * D^{2} * CS_{Con} - 2.53875 \times 10^{-4} * D \\
& * {CS_{Soil}}^{2} - 6.33564 \times 10^{-1} * CS_{Soil} + 3.42777 \times 10^{-2} * D * H * RR + 1.57659 \times 10^{-4} \\
& * CS_{Soil} * {CS_{Con}}^{2} + 2.72961 \times 10^{-4} * RR * {CS_{Soil}}^{2} - 6.27248 \times 10^{-5} * E_{Soil} * CS_{Soil} \\
& * E_{Con} - 3.54573 \times 10^{-3} * D * RR * E_{Con} - 8.74253 \times 10^{-3} * D * RR * CS_{Con} \\
& + 9.73444 \times 10^{-5} * E_{Soil} * {E_{Con}}^{2} - 6.03255 \times 10^{-1} * RR * CS_{Con} + 3.58691 \times 10^{-3} \\
& * CS_{Soil} * E_{Con} + 4.36762 \times 10^{-5} * D^{2} * E_{Con} - 3.62583 \times 10^{-3} * H * RR * E_{Con} \\
& + 6.06874 \times 10^{-1} * CS_{Con} + 2.01790 \times 10^{-5} * E_{Soil} * CS_{Soil} * CS_{Con} - 1.29024 \times 10^{-2} \\
& * E_{Soil} * E_{Con}
\end{aligned}
$$

A4.2. Horizontal failure displacement proposed formula using POLYREG-HYT-3:

$$Disp\ Max = -3.37933 \times 10^{-4} * E_{Soil} + 4.34521 \times 10^{-4} * CS_{Soil} + 3.86938 \times 10^{-7} * D * RR * CS_{Soil} + 1.69665 \times 10^{-8} * D * E_{Soil} * CS_{Con} - 1.76014 \times 10^{-8} * {CS_{Soil}}^{3} + 1.74355 \times 10^{-7} * D^{3} + 1.08035 \times 10^{-6} * CS_{Soil} * CS_{Con} - 7.94735 \times 10^{-9} * E_{Soil} * CS_{Soil} * E_{Con} + 2.18456 \times 10^{-3} * RR - 2.07166 \times 10^{-7} * D^{2} * CS_{Soil} + 1.84817 \times 10^{-2} + 1.17571 \times 10^{-8} * {E_{Soil}}^{2} * CS_{Soil} + 1.09358 \times 10^{-7} * D * {CS_{Soil}}^{2} - 5.95088 \times 10^{-8} * D * CS_{Soil} * CS_{Con} + 1.85516 \times 10^{-5} * D * CS_{Con} - 7.78108 \times 10^{-8} * {CS_{Con}}^{3} - 6.79943 \times 10^{-8} * D * E_{Soil} * CS_{Soil} - 2.87435 \times 10^{-8} * D^{2} * E_{Con} - 5.17600 \times 10^{-8} * E_{Soil} * E_{Con} * CS_{Con} + 2.78766 \times 10^{-8} * D * CS_{Soil} * E_{Con} + 8.6054710^{-9} * {E_{Soil}}^{2} * E_{Con} - 3.64023 \times 10^{-8} * H * {E_{Con}}^{2} + 6.25615 \times 10^{-8} * E_{Con} * {CS_{Con}}^{2} + 4.10003 \times 10^{-8} * H * CS_{Soil} * E_{Con} - 1.22456 \times 10^{-7} * D^{2} * CS_{Con} + 3.03558 \times 10^{-7} * H * {CS_{Con}}^{2} + 1.80813 \times 10^{-7} * H * E_{Soil} * CS_{Con} - 7.25575 \times 10^{-7} * D^{2} * RR - 4.47225 \times 10^{-5} * H * CS_{Con} + 1.19670 \times 10^{-5} * H * E_{Soil} - 3.22576 \times 10^{-7} * RR * CS_{Soil} * E_{Con} + 8.88846 \times 10^{-8} * D^{2} * E_{Soil} + 2.92132 \times 10^{-7} * RR * E_{Soil} * E_{Con} + 1.57940 \times 10^{-7} * D * H * CS_{Soil} + 6.25684 \times 10^{-8} * RR * CS_{Soil} * CS_{Con} - 2.31004 \times 10^{-6} * RR^{2} * CS_{Soil} - 3.45315 \times 10^{-7} * H^{2} * E_{Soil} - 6.44718 \times 10^{-6} * D^{2} + 5.58161 \times 10^{-7} * E_{Soil} * CS_{Con} - 1.30977 \times 10^{-5} * RR * E_{Soil} - 9.13977 \times 10^{-8} * H * E_{Soil} * CS_{Soil} + 9.33435 \times 10^{-7} * H * RR * E_{Soil} - 1.18726 \times 10^{-7} * D * RR * E_{Soil}$$

A4.3. Horizontal displacement at Fx Max/2 proposed formula using POLYREG-HYT-3:

$$Disp\ Fx2 = -3.21915 \times 10^{-8} * D^{2} * CS_{Soil} + 4.91719 \times 10^{-4} * D + 1.94523 \times 10^{-4} * RR + 1.70059 \times 10^{-7} * RR * {E_{Con}}^{2} - 1.84572 \times 10^{-8} * D * E_{Soil} * CS_{Soil} - 2.79567 \times 10^{-8} * D * CS_{Soil} * CS_{Con} + 2.61228 \times 10^{-7} * D * RR * CS_{Soil} - 1.68066 \times 10^{-8} * E_{Soil} * E_{Con} * CS_{Con} - 3.28292 \times 10^{-7} * D^{2} * RR - 2.92916 \times 10^{-6} * {CS_{Con}}^{2} - 4.57150 \times 10^{-7} * H * RR * E_{Soil} - 1.30731 \times 10^{-9} * CS_{Soil} * {E_{Con}}^{2} + 6.06473 \times 10^{-8} * H * E_{Soil} * CS_{Con} - 2.71120 \times 10^{-7} * H^{2} * CS_{Con} - 5.74925 \times 10^{-7} * D * H * RR - 4.44624 \times 10^{-9} * CS_{Soil} * E_{Con} * CS_{Con} + 8.26352 \times 10^{-9} * D * E_{Soil} * CS_{Con} + 1.58602 \times 10^{-6} * CS_{Soil} * CS_{Con} - 1.56988 \times 10^{-4} * RR^{3} - 5.46978 \times 10^{-8} * RR * {CS_{Soil}}^{2} + 1.44902 \times 10^{-9} * {E_{Soil}}^{3} - 2.77091 \times 10^{-7} * D * RR * E_{Con} + 3.13361 \times 10^{-8} * D^{2} * E_{Soil} - 7.34435 \times 10^{-8} * D * H * E_{Con} + 2.89187 \times 10^{-8} * E_{Con} * {CS_{Con}}^{2} + 4.82819 \times 10^{-7} * D * CS_{Soil} + 5.66467 \times 10^{-6} * H^{2} * RR + 2.48879 \times 10^{-9} * {E_{Soil}}^{2} * E_{Con} + 1.92611 \times 10^{-7} * RR * CS_{Soil} * E_{Con} + 7.80113 \times 10^{-8} * D^{3} + 8.85533 \times 10^{-4} * RR^{2} - 5.79554 \times 10^{-5} * E_{Soil} - 1.15444 \times 10^{-5} * D^{2} + 2.73096 \times 10^{-9} * E_{Soil} * {CS_{Soil}}^{2} + 1.21853 \times 10^{-6} * D * E_{Con} - 2.68169 \times 10^{-5} * RR * E_{Con} + 2.15545 \times 10^{-8} * D * {CS_{Soil}}^{2} + 3.65461 \times 10^{-6} * D * CS_{Con} + 6.84043 \times 10^{-8} * D^{2} * H - 4.44656 \times 10^{-9} * {CS_{Soil}}^{3} - 5.46503 \times 10^{-10} * {E_{Soil}}^{2} * CS_{Soil} + 7.24596 \times 10^{-9} * {CS_{Con}}^{3} - 1.35831 \times 10^{-8} * D^{2} * CS_{Con} - 6.05752 \times 10^{-7} * RR^{2} * CS_{Soil} - 1.25931 \times 10^{-8} * H * E_{Soil} * CS_{Soil} - 2.01421 \times 10^{-8} * H * {CS_{Con}}^{2} + 1.69627 \times 10^{-8} * H * CS_{Soil} * E_{Con} - 9.43300 \times 10^{-4} - 9.81210 \times 10^{-7} * D * E_{Soil} - 1.90335 \times 10^{-9} * {CS_{Soil}}^{2} * E_{Con} + 8.29442 \times 10^{-5} * CS_{Soil}$$