INSTRUCTION MANUAL: EGS Analytical Model Sensitivity Studies

By: Benjamin Willis

1. Run Sensitivity Studies

To run the sensitivity studies download the respective study Jupyter Notebook from the GitHub repository:

https://github.com/BenjaminWillisINL/AnalyticalDataCSVs/tree/main/RunSensitivityStudies

1a. Run Varying Exit Perforations Study

Make sure that the "FrictionFactors.ipynb" Jupyter Notebook is in the same folder as the "With or Without Exit Perfs.ipynb" Jupyter Notebook.

Make Sure all storage variables are initialized to the correct dimensions as described in the screenshot below:

```
## NOTE: Only run this cell at the beginning, you will have to run the code below with WellsOrientation = 1,
## and then re-run the code below with WellsOrientation = 3.
## DO NOT re-run this code after initial initialization

## NOTE: The row number is the desired maximum number of perforations - in this case 100 perforations are used
## The column number is the number of fractures multiplied by 2 - in this case 3 fractures are used
storage5 = np.zeros((100,6))
storage10 = np.zeros((100,6))
storage15 = np.zeros((100,6))
storage20 = np.zeros((100,6))
storage25 = np.zeros((100,6))
storage30 = np.zeros((100,6))
storage35 = np.zeros((100,6))
storage40 = np.zeros((100,6))
storage45 = np.zeros((100,6))
storage50 = np.zeros((100,6))


## NOTE: The row number is the desired maximum number of perforations
## The column number is the length of the flowrates multiplied by 2- in this case 10 flowrates are tested
Pressuredrop = np.zeros((100,20))
```

Set the "WellsOrientation" variable to 1 to begin the parallel well design simulation in the screenshot below:

```
#ORIENTATION OF WELLS
# If 1, The wells are Parallel
# If 2, The wells are Anti-Parallel
# If 3, The wells are Non-Parallel  # Input the Difference between first and last fracture
# NOTE: Change to 3 after first run but DO NOT re-run initialization step above!!!!
WellsOrientation=3
```

Change the “ActivateProductionPerforations” variable to either 1 or 0 depending on if there are exit perforations or not as in the screenshot below:

```
#CONTROL VARIABLES

#Turn on pressure drop in injection well
InjectionWellActivate=1 #if 1, the pressure drop in the well is on
#Turn on pressure drop in production well
ProductionWellActivate=1 #if 1, the pressure drop in the well is on
#Activate Perforation
ActivateInjectionPerforations=1  #If 0, perforation pressure drop is zero.
#activate production well perforation
#NOTE: Change this to 1 in order to activate exit perforations
ActivateProductionPerforations=0
```

Run all code cells until Ln 49 is reached, once this line is reached run the code and wait, as it will take some time to store all the data. Use the screenshot below to identify this cell:

In [49]:

```
counter = 0
counter2 = 0
while counter2 <= (len(Flowrate_Initial)-1):
    while counter <= (len(NumberOfPerfs)-1):

        Pressure_Injection=np.zeros(NumberOfFractures+1)
        Pressure_Production=np.zeros(NumberOfFractures+1)
        Pressure_Fracture=np.zeros(NumberOfFractures)
        Flowrate_Fracture=np.zeros(NumberOfFractures)
        Pressure_Injection[0]=3e7
        MaxIterations=1000 #number of iterations
        #mmax=10000 #max number of iterations
        Tolerance=1e-6 #Tolerance
        FactorTest = 1000 #NumberOfFractures*FractureHeight
        Flowrate_Update=np.zeros(NumberOfFractures)
        Flowrate_Frac1=np.zeros(MaxIterations)
        Flowrate_Frac2=np.zeros(MaxIterations)
        Flowrate_Frac3=np.zeros(MaxIterations)

        for i in range(NumberOfFractures):
            Flowrate_Fracture[i]=Flowrate_Initial[counter2]/NumberOfFractures
            FractureLength[i]=abs(BaseFractureLength+(NumberOfFractures-i-1)*math.tan(WellAngle)*BaseInjectionWellSection+(NumberOfFr
        for i in range(NumberOfFractures+1):
            InjectionWellSection[i]=BaseInjectionWellSection/math.cos(WellAngle)
            ProductionWellSection[i]=BaseProductionWellSection/math.cos(WellAngle)


        for j in range(MaxIterations):
            Flowrate_Cumulative=Flowrate_Initial[counter2]/FactorTest
            #pressure drop in injection well
            for i in range(NumberOfFractures):
                if InjectionWellActivate==0:
                    Pressure_Injection[i+1]=Pressure_Injection[i]
                else:
                    Pressure_Injection[i+1]=Pressure_Injection[i]-Fhal(e,Diameter_InjectionWell,Rep(Flowrate_Cumulative,Diameter_Inje
                Flowrate_Cumulative=Flowrate_Cumulative-Flowrate_Fracture[i]/FactorTest
```

Once this code is done running, go and change the "WellsOrientation" variable to 3 and re-run the whole code, EXCEPT for the initialization of the storage variables. Do not reinitialize the storage variables. Wait for Ln 49 to finish running for the non-parallel well design. Once the code is done running, run the next line to save the data as CSV files as in the screenshot below:

In [16]:

```
np.savetxt("1storage5.csv", storage5, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Parallel'
np.savetxt("1storage10.csv", storage10, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Paralle
np.savetxt("1storage15.csv", storage15, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Paralle
np.savetxt("1storage20.csv", storage20, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Paralle
np.savetxt("1storage25.csv", storage25, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Paralle
np.savetxt("1storage30.csv", storage30, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Paralle
np.savetxt("1storage35.csv", storage35, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Paralle
np.savetxt("1storage40.csv", storage40, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Paralle
np.savetxt("1storage45.csv", storage45, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Paralle
np.savetxt("1storage50.csv", storage50, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Paralle



np.savetxt("1Pressuredrop.csv", Pressuredrop, delimiter=",", header = ','.join(['5 kg/s Parallel','10 kg/s Parallel','15 kg/s Parallel','
```

In [ ]:

All headers are included in the CSV files so the content of each column is known. The row numbers represent the number of perforations. With row 1 representing 1 perforation, and row 100 representing 100 perforations.

Now re-run the code again with exit perforations on or off depending on what was started with to get the other set of data.

1b. Running the Permeability Study

Make sure that the "FrictionFactors.ipynb" Jupyter Notebook is in the same folder as the "VaryPermeability.ipynb" Jupyter Notebook.

Make sure that all storage variables are initialized to the desired dimensions as shown in the screenshot below:

```
## NOTE: The row number is the desired maximum number of perforations - in this case 60 perforations are used
## The column number is the number of fractures multiplied by 2 - in this case 3 fractures are used
storage5 = np.zeros((60,6))
storage10 = np.zeros((60,6))
storage15 = np.zeros((60,6))
storage20 = np.zeros((60,6))
storage25 = np.zeros((60,6))
storage30 = np.zeros((60,6))

## NOTE: The row number is the desired maximum number of perforations
## The column number is the length of the flowrates multiplied by 2- in this case 6 flowrates are tested
Pressuredrop = np.zeros((60,12))
```

Set the "WellsOrientation" variable to 1 to begin the parallel well design simulation in the screenshot below:

```
#ORIENTATION OF WELLS
# If 1, The wells are Parallel
# If 2, The wells are Anti-Parallel
# If 3, The wells are Non-Parallel  # Input the Difference between first and last fracture
# NOTE: Change to 3 after first run but DO NOT re-run initialization step above!!!!
WellsOrientation=1
```

Now set the permeability value to the desired value (1e-12 m^2 or 1e-14 m^2) as shown in the screenshot below:

In [47]:

```
#For Fracture
NumberOfFractures=3
TotalWellLength=3227 #m, Length of the well
BasePermeability=1e-12 # or 1e-14 #m2, #Fracture base Permebility
FractureHeight=100 #m Fracture entrance length
FractureWidth=5.0 #m Fracture entrance pseudo width
BaseFractureLength=100 #m Height of fractures
Area_FractureEntrance=FractureHeight*FractureWidth #Area of fracture for the fluid to enter
```

Set control variables to desired parameters, in this scenario the exit perforations were turned off. Use the screenshot below to discern which control variables to use:

In [50]:

```
#CONTROL VARIABLES

#Turn on pressure drop in injection well
InjectionWellActivate=1 #if 1, the pressure drop in the well is on
#Turn on pressure drop in production well
ProductionWellActivate=1 #if 1, the pressure drop in the well is on
#Activate Perforation
ActivateInjectionPerforations=1  #If 0, perforation pressure drop is zero.
#activate production well perforation
ActivateProductionPerforations=0

#Variable Permeablity
# If 0, All fractures have same permeability
# If 1, The permeability values are assigned at random with respect to the base value
# If 2, Option to input custom permeability values.
VariablePermeability=0



if WellsOrientation==3:
    #dl=float(input("Enter the difference between the first and last fracture size: "))
    dl = 100
    BaseFractureLength=BaseFractureLength-100/2
    WellAngle=math.atan((100/2)/((NumberOfFractures-1)*BaseInjectionWellSection)) #Angle of Injection well
    print(math.degrees(WellAngle)*2)
#for i in range(NumberOfFractures):
#    Flowrate_Fracture[i]=Flowrate_Initial/NumberOfFractures
#    FractureLength[i]=abs(BaseFractureLength+(NumberOfFractures-i-1)*math.tan(WellAngle)*BaseInjectionWellSection+(NumberOfFractures-i-1
for i in range(NumberOfFractures+1):
    InjectionWellSection[i]=BaseInjectionWellSection/math.cos(WellAngle)
    ProductionWellSection[i]=BaseProductionWellSection/math.cos(WellAngle)

#Adaptive Perforation (Still in test phase)
# If 0, Adaptive Perforations are turned off
# If 1, Adaptive Perforations are turned on
# If 2, Custom input for Perforations
AdaptivePerf=0
```

Run all the code until the main flowrate cell is reached, then wait for this code to run, use the screenshot to determine which cell to run and wait:

In [ ]:

```
counter = 0
counter2 = 0
while counter2 <= (len(Flowrate_Initial)-1):
    while counter <= (len(NumberOfPerfs)-1):

        Pressure_Injection=np.zeros(NumberOfFractures+1)
        Pressure_Production=np.zeros(NumberOfFractures+1)
        Pressure_Fracture=np.zeros(NumberOfFractures)
        Flowrate_Fracture=np.zeros(NumberOfFractures)
        Pressure_Injection[0]=3e7
        MaxIterations=1000 #number of iterations
        #mmax=10000 #max number of iterations
        Tolerance=1e-6 #Tolerance
        FactorTest = 1000 #NumberOfFractures*FractureHeight
        Flowrate_Update=np.zeros(NumberOfFractures)
        Flowrate_Frac1=np.zeros(MaxIterations)
        Flowrate_Frac2=np.zeros(MaxIterations)
        Flowrate_Frac3=np.zeros(MaxIterations)

        for i in range(NumberOfFractures):
            Flowrate_Fracture[i]=Flowrate_Initial[counter2]/NumberOfFractures
            FractureLength[i]=abs(BaseFractureLength+(NumberOfFractures-i-1)*math.tan(WellAngle)*BaseInjectionWellSection+(NumberOfFractu
        for i in range(NumberOfFractures+1):
            InjectionWellSection[i]=BaseInjectionWellSection/math.cos(WellAngle)
            ProductionWellSection[i]=BaseProductionWellSection/math.cos(WellAngle)
```

Once the cell above is done running, reset the "WellsOrientation" variable to 3 and re-run the whole process again. Do not reinitialize the storage variables again, only run every cell block below the initialization for the storage variables. Once the flowrate cell is done running for the second time, run the next cell to convert all the stored data to a CSV file, as shown in the screenshot below:

In [31]:

```
np.savetxt("fstorage5.csv", storage5, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Parallel'
np.savetxt("fstorage10.csv", storage10, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Paralle
np.savetxt("fstorage15.csv", storage15, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Paralle
np.savetxt("fstorage20.csv", storage20, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Paralle
np.savetxt("fstorage25.csv", storage25, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Paralle
np.savetxt("fstorage30.csv", storage30, delimiter=",", header = ','.join(['Fracture 1 Parallel','Fracture 2 Parallel','Fracture 3 Paralle

np.savetxt("fPressuredrop.csv", Pressuredrop, delimiter=",", header = ','.join(['5 kg/s Parallel','10 kg/s Parallel','15 kg/s Parallel','
```

All headers are included in the CSV files so the content of each column is known. The row numbers represent the number of perforations. With row 1 representing 1 perforation, and row 60 representing 60 perforations.

1c. Running the Well Angle Study

Make sure that the "FrictionFactors.ipynb" Jupyter Notebook is in the same folder as the "Pheta.ipynb" Jupyter

Notebook. Also make sure that the “testpheta.CSV” file is in the same folder as the “Pheta.ipynb” Jupyter Notebook.

Make sure that all storage variables are initialized to the desired dimensions as shown in the screenshot below:

In [291…

```
#Initialization
Pressure_Injection=np.zeros(NumberOfFractures+1)
Pressure_Production=np.zeros(NumberOfFractures+1)
Pressure_Fracture=np.zeros(NumberOfFractures)
Flowrate_Fracture=np.zeros(NumberOfFractures)
FractureLength=np.zeros(NumberOfFractures)
InjectionWellSection=np.zeros(NumberOfFractures+1)
ProductionWellSection=np.zeros(NumberOfFractures+1)
Permeability_Fracture=np.zeros(NumberOfFractures)
Diameter_InjectionPerforation=np.zeros(NumberOfFractures)
Diameter_ProductionPerforation=np.zeros(NumberOfFractures)
WellAngle=0

#NOTE: 1800 data points were taken, 5 rows of data including well angle, fracture 1,2 and 3 flow distribution,
# and pressure drop(PSI)
storage = np.zeros((1800,5))
```

Set the flowrate to the desired flowrate (starting at 5 kg/s going to 50 kg/s in steps of 5). Also make sure to change the name of the storage CSV at the end of the file to make the desired flowrate. Use the screenshot below as an example of how to do this for 50 kg/s:

In [292…

```
#Flow Rate and Pressure
Flowrate_Initial=50 #kg/s
Pressure_Injection[0]=3e7
```

In [299…

```
np.savetxt("1storage50.csv", storage, delimiter=",", header = ','.join(['Well Angle','Fracture 1','Fracture 2','Fracture 3','Pressure Dro
```

Set the control variables to the desired parameters and make sure the “WellsOrientation” variable is set to 3, as the non-parallel is the only well design being tested in this study. Use the screenshot below to locate the control variables:

In [293…

```
#CONTROL VARIABLES

#Turn on pressure drop in injection well
InjectionWellActivate=1 #if 1, the pressure drop in the well is on
#Turn on pressure drop in production well
ProductionWellActivate=1 #if 1, the pressure drop in the well is on
#Activate Perforation
ActivateInjectionPerforations=1  #If 0, perforation pressure drop is zero.
#activate production well perforation
ActivateProductionPerforations=0

#Variable Permeablity
# If 0, All fractures have same permeability
# If 1, The permeability values are assigned at random with respect to the base value
# If 2, Option to input custom permeability values.
VariablePermeability=0

# NOTE: Import fracture spacing from testpheta CSV file or the code will not run
pwo = pd.read_csv('testpheta.csv')
pwo2 = pd.DataFrame(pwo)
dls = pwo2.to_numpy()
dl = dls[0:1800,0]
print(len(dl))
#dl = 0
#ORIENTATION OF WELLS
# If 1, The wells are Parallel
# If 2, The wells are Anti-Parallel
# If 3, The wells are Non-Parallel  # Input the Difference between first and last fracture
WellsOrientation=3

if WellsOrientation==3:
    #dl=float(input("Enter the difference between the first and last fracture size: "))
    #print(dl)
    #dl = 7.534760914
    BaseFractureLength=BaseFractureLength-dl[0]/2
    WellAngle=math.atan((dl[0]/2)/((NumberOfFractures-1)*BaseInjectionWellSection)) #Angle of Injection well
    print(math.degrees(WellAngle)*2)
```

Once all the control variables and the desired flowrate has been set, run the flowrate cell and wait for the code to be done. Then save the file as a CSV by running the final cell that was edited to match the flowrate earlier. Use the screenshot below to locate the flowrate cell.

In [ ]:

```
counter = 0

while counter <= (len(dl)-1):

    Pressure_Injection=np.zeros(NumberOfFractures+1)
    Pressure_Production=np.zeros(NumberOfFractures+1)
    Pressure_Fracture=np.zeros(NumberOfFractures)
    Flowrate_Fracture=np.zeros(NumberOfFractures)
    Pressure_Injection[0]=3e7
    MaxIterations=1000 #number of iterations
    #mmax=10000 #max number of iterations
    Tolerance=1e-6 #Tolerance
    FactorTest = 1000 #NumberOfFractures*FractureHeight
    Flowrate_Update=np.zeros(NumberOfFractures)
    Flowrate_Frac1=np.zeros(MaxIterations)
    Flowrate_Frac2=np.zeros(MaxIterations)
    Flowrate_Frac3=np.zeros(MaxIterations)
    BaseFractureLength = 100
    WellAngle = 0


    if WellsOrientation==3:
        #dl=float(input("Enter the difference between the first and last fracture size: "))
        #print(dl)
        #dl = 7.534760914
        BaseFractureLength=BaseFractureLength-dl[counter]/2
        WellAngle=math.atan((dl[counter]/2)/((NumberOfFractures-1)*BaseInjectionWellSection)) #Angle of Injection well
        print(math.degrees(WellAngle)*2)

    for i in range(NumberOfFractures):
        Flowrate_Fracture[i]=Flowrate_Initial/NumberOfFractures
        FractureLength[i]=abs(BaseFractureLength+(NumberOfFractures-i-1)*math.tan(WellAngle)*BaseInjectionWellSection+(NumberOfFractures-
    for i in range(NumberOfFractures+1):
        InjectionWellSection[i]=BaseInjectionWellSection/math.cos(WellAngle)
        ProductionWellSection[i]=BaseProductionWellSection/math.cos(WellAngle)
```

Reset the flowrate to the next desired flowrate interval and also change the name of the saved CSV file to match the next flowrate. Re-run the flowrate cell again and save to a CSV when the code is finished running.

All headers are included in the CSV files so the content of each column is known. The row numbers represent the various well angles, starting from 0.1° to 115° in steps of 0.1°.

1d. Running the Number of Fractures Study

Make sure that the "FrictionFactors.ipynb" Jupyter Notebook is in the same folder as the "FractureREAL.ipynb" and "VaryNumberofPerfsWith1FractureZone.ipynb" Jupyter Notebooks.

This study consists of two Jupyter Notebooks because of the two sets of data desired from this study. The first Jupyter Notebook "FractureREAL.ipynb" is used to test different numbers of fractures from 1 to 30 fracture zones with 1 inlet perforation. The second Jupyter Notebook "VaryNumberofPerfsWith1FractureZone.ipynb" is used to test different number of inlet perforations from 1 to 30 with 1 fracture zone. These plots are then mirrored on top of each other for comparison.

This study was only done for the parallel well design, therefore the "WellsOrientation" variable for both Ju[yter

Notebooks should always be set to 1, as shown in the screenshot below:

```
#dl = 0
#ORIENTATION OF WELLS
# If 1, The wells are Parallel
# If 2, The wells are Anti-Parallel
# If 3, The wells are Non-Parallel  # Input the Difference between first and last fracture
WellsOrientation=1
```

Using the "FractureREAL.ipynb" Notebook, set the flowrate to the desired flowrate (starting at 5 kg/s to 50 kg/s in steps of 5). Also make sure to initialize the storage variable to the correct flowrate name, an example for 50 kg/s is shown in the screenshot below:

In [90]:

```
#Flow Rate and Pressure

# NOTE: Change depending on which flowrate is desired to be captured
Flowrate_Initial=50 #kg/s

#storage = np.zeros((20,6))
```

```
# NOTE: Change the variable name to storage# with the # being the flowrate set in that run of the code
storage50 = np.zeros(len(NumberOfFractures))
print(len(NumberOfFractures))
```

Once this is done, the name of the storage variable in the flowrate cell should also match this new storage initialization variable name as shown in the screenshot below:

```
    # NOTE: Change the variable name to storage# with the # being the flowrate set in that run of the code
    storage50[counter] = ppp
    print(ppp)
    counter = counter + 1
print(Flowrate_Initial)
```

Also make sure to change the storage variables name in the final cell where it will be saved to a CSV, as shown in the screenshot below:

```
## NOTE: Must change the flowrate everytime to get a new value
# ALSO: change the name from storage5 to storage25 depending on the flowrate value

print(storage50)
np.savetxt("FRACTUREstorage50.csv", storage50, delimiter=",", header = ','.join(['Pressure Drop']))
```

Once all the storage variable names are changed, run the main flowrate cell and wait for the code to finish, once the code is finished running save the data to a CSV file by running the last code cell. Use the screenshot below to identify where the flowrate cell is:

In [ ]:

```
counter = 0
while counter <= (len(NumberOfFractures)-1):
    BaseFractureLength=100

    BaseInjectionWellSection=TotalWellLength/NumberOfFractures[counter] #m Length of Inj pipe section
    BaseProductionWellSection=TotalWellLength/NumberOfFractures[counter] #m Length of Prod pipe section



    #Initialization
    Pressure_Injection=np.zeros(NumberOfFractures[counter]+1)
    Pressure_Production=np.zeros(NumberOfFractures[counter]+1)
    Pressure_Fracture=np.zeros(NumberOfFractures[counter])
    Flowrate_Fracture=np.zeros(NumberOfFractures[counter])
    FractureLength=np.zeros(NumberOfFractures[counter])
    InjectionWellSection=np.zeros(NumberOfFractures[counter]+1)
    ProductionWellSection=np.zeros(NumberOfFractures[counter]+1)
    Permeability_Fracture=np.zeros(NumberOfFractures[counter])
    Diameter_InjectionPerforation=np.zeros(NumberOfFractures[counter])
    Diameter_ProductionPerforation=np.zeros(NumberOfFractures[counter])
    WellAngle=0
    Pressure_Injection[0]=3e7


    if WellsOrientation==3:
        #dl=float(input("Enter the difference between the first and last fracture size: "))
        #print(dl)
        dl = 100
        BaseFractureLength=BaseFractureLength-dl/2
        WellAngle=math.atan((dl/2)/((NumberOfFractures[counter]-1)*BaseInjectionWellSection)) #Angle of Injection well
        print(math.degrees(WellAngle)*2)
    for i in range(NumberOfFractures[counter]):
        Flowrate_Fracture[i]=Flowrate_Initial/NumberOfFractures[counter]
        FractureLength[i]=abs(BaseFractureLength+(NumberOfFractures[counter]-i-1)*math.tan(WellAngle)*BaseInjectionWellSection+(NumberOfF
    for i in range(NumberOfFractures[counter]+1):
        InjectionWellSection[i]=BaseInjectionWellSection/math.cos(WellAngle)
        ProductionWellSection[i]=BaseProductionWellSection/math.cos(WellAngle)


    #Calculate Fracture Permeability
    #Function is defined as:
    # var1*BasePermeability*(10^var2), where Var1 varies from 1 to 10 and Var2 varies from -1, 0 or 1, at random.
```

Re-run the code with the next desired flowrate and change all the storage variable names and the “Flowrate_Initial” variable.

All headers are included in the CSV files so the content of each column is known. The row numbers represent the number of fracture zones in the “FRACTUREstorage#.CSV” files. With row 1 representing 1 fracture zone and row 30 representing 30 fracture zones.

Next, open the "VaryNumberofPerfsWith1FractureZone.ipynb" Jupyter Notebook. Initialized the "Pressuredrop" variable to the correct dimensions as shown in the screenshot below:

```
## NOTE: The row number is the desired maximum number of perforations
## The column number is the length of the flowrates in this case 10 flowrates are tested
Pressuredrop = np.zeros((30,10))
```

Make sure all control variables are set to the desired parameters as shown in the screenshot below:

In [50]:

```
#CONTROL VARIABLES

#Turn on pressure drop in injection well
InjectionWellActivate=1 #if 1, the pressure drop in the well is on
#Turn on pressure drop in production well
ProductionWellActivate=1 #if 1, the pressure drop in the well is on
#Activate Perforation
ActivateInjectionPerforations=1  #If 0, perforation pressure drop is zero.
#activate production well perforation
#NOTE: Change this to 1 in order to activate exit perforations
ActivateProductionPerforations=0

#Variable Permeablity
# If 0, All fractures have same permeability
# If 1, The permeability values are assigned at random with respect to the base value
# If 2, Option to input custom permeability values.
VariablePermeability=0


if WellsOrientation==3:
    #d1=float(input("Enter the difference between the first and last fracture size: "))
    d1 = 100
    BaseFractureLength=BaseFractureLength-d1/2
    WellAngle=math.atan((d1/2)/((NumberOfFractures-1)*BaseInjectionWellSection)) #Angle of Injection well
    print(math.degrees(WellAngle)*2)
#for i in range(NumberOfFractures):
#    Flowrate_Fracture[i]=Flowrate_Initial/NumberOfFractures
#    FractureLength[i]=abs(BaseFractureLength+(NumberOfFractures-i-1)*math.tan(WellAngle)*BaseInjectionWellSection+(NumberOfFractures-i-1
for i in range(NumberOfFractures+1):
    InjectionWellSection[i]=BaseInjectionWellSection/math.cos(WellAngle)
    ProductionWellSection[i]=BaseProductionWellSection/math.cos(WellAngle)

#Adaptive Perforation (Still in test phase)
# If 0, Adaptive Perforations are turned off
# If 1, Adaptive Perforations are turned on
# If 2, Custom input for Perforations
AdaptivePerf=0
```

Run the main flowrate cell and wait for the code to finish, once the code is finished run the last cell of code and save it to a CSV. There is no need to change variable names with flowrates here, as all the flowrates are calculated one after the other automatically. Use the screenshot below to discern where the main flowrate cell is located:

In [54]:

```
counter = 0
counter2 = 0
while counter2 <= (len(Flowrate_Initial)-1):
    while counter <= (len(NumberOfPerfs)-1):

        Pressure_Injection=np.zeros(NumberOfFractures+1)
        Pressure_Production=np.zeros(NumberOfFractures+1)
        Pressure_Fracture=np.zeros(NumberOfFractures)
        Flowrate_Fracture=np.zeros(NumberOfFractures)
        Pressure_Injection[0]=3e7
        MaxIterations=1000 #number of iterations
        #mmax=10000 #max number of iterations
        Tolerance=1e-6 #Tolerance
        FactorTest = 1000 #NumberOfFractures*FractureHeight
        Flowrate_Update=np.zeros(NumberOfFractures)
        Flowrate_Frac1=np.zeros(MaxIterations)
        Flowrate_Frac2=np.zeros(MaxIterations)
        Flowrate_Frac3=np.zeros(MaxIterations)

        for i in range(NumberOfFractures):
            Flowrate_Fracture[i]=Flowrate_Initial[counter2]/NumberOfFractures
            FractureLength[i]=abs(BaseFractureLength+(NumberOfFractures-i-1)*math.tan(WellAngle)*BaseInjectionWellSection+(NumberOfFractu
        for i in range(NumberOfFractures+1):
            InjectionWellSection[i]=BaseInjectionWellSection/math.cos(WellAngle)
            ProductionWellSection[i]=BaseProductionWellSection/math.cos(WellAngle)


        for j in range(MaxIterations):
            Flowrate_Cumulative=Flowrate_Initial[counter2]/FactorTest
            #pressure drop in injection well
            for i in range(NumberOfFractures):
                if InjectionWellActivate==0:
                    Pressure_Injection[i+1]=Pressure_Injection[i]
                else:
                    Pressure_Injection[i+1]=Pressure_Injection[i]-Fhal(e,Diameter_InjectionWell,Rep(Flowrate_Cumulative,Diameter_Injectio
                Flowrate_Cumulative=Flowrate_Cumulative-Flowrate_Fracture[i]/FactorTest

            #Pressure drop in Production well
            Flowrate_Cumulative=Flowrate_Fracture[NumberOfFractures-1]/FactorTest
            if WellsOrientation!=2:
```

All headers are included in the CSV files so the content of each column is known. The row numbers represent the number of perforations in the “1Pressuredrop.CSV” file. With row 1 representing 1 inlet perforation and row 30 representing 30 inlet perforations.

2. Generate Plots

Depending on the desired plot select the appropriate “Compare” folder in the GitHub repository. Each “Compare” folder will have all the CSV files required to generate the plots as well as the Jupyter Notebooks used to generate those plots. Make sure that the respective Jupyter Notebook used has all the associated CSV files in the same folder in order to read the data. In each of the Jupyter Notebooks used to generate the plots a path is defined the save the plots to. Change this path to the desired path where

the plots will be saved. The screenshot below shows what this path variable looks like and where to find it:

```
is_windows = any(platform.win32_ver())
if is_windows==True:
    path=r'CustomPath'
else:
    path=r'/Users/willba/Desktop/pheta'
```

Once this is done, then all plots should be generated and saved to the desired paths.