Homework#2

September 27, 2021

[1]:
```
import numpy as np
import pandas as pd

import bokeh.plotting as bp
from bokeh.models import tools as bmt, ColumnDataSource

from bokeh.io import output_notebook, export_png
output_notebook()
```

[2]:
```
def initialize_fig(title: str = 'your_title',
                   x_axis_label: str = 'x_axis_label',
                   x_axis_type: str = 'auto',
                   y_axis_label: str = 'y_axis_label',
                   y_axis_type: str = 'auto',
                   tools: str = 'pan,wheel_zoom,box_zoom,reset',
                   tooltips: list = [],
                   formatters: dict = {},
                   plot_height: int = 300,
                   fig_sizing_mode: str = 'scale_width',
                   ) -> bp.figure:
    # bokeh style
    TOOLS = tools
    hover_tool = bmt.HoverTool(tooltips=tooltips, formatters=formatters)

    fig = bp.figure(title=title,
                    x_axis_label=x_axis_label,
                    x_axis_type=x_axis_type,
                    y_axis_label=y_axis_label,
                    y_axis_type=y_axis_type,
                    plot_height=plot_height,
                    tools=TOOLS,
                   )
    fig.add_tools(hover_tool)
    fig.sizing_mode = fig_sizing_mode

    return fig
```

0.0.1 Question 1

Compare the Brooks and Corey, Campbell and van Genuchten k( ) – analytic relationship (i.e. moisture conductivity) for a silty clay loam. Plot the data and briefly discuss the differences. Use _r = 0.057 . For the van Genuchten parameters: Percent by Weight Finer Than Indicated Diameter (mm)

Soil	50	19	9.5	4.76	2.00	0.420	0.074	0.020	0.005	0.002
1.1	100	100	100	100	100	100	97	79	45	16
1.2	100	100	98	94	70	19	15	8	3	2
1.3	93	91	88	85	69	44	40	27	13	6
1.4	100	100	100	100	100	97	92	75	47	31

[3]:
```
# soil df setup
soil_mm_int = [50., 19., 9.5, 4.76, 2., 0.420, 0.074, 0.020, 0.005, 0.002]
# soil_mm = ['50.', '19.', '9.5', '4.76', '2.', '0.420', '0.074', '0.020', '0.
 ↪005', '0.002']
soil_1  = [ 100.,  100.,  100.,   100., 100.,    100.,     97.,     79.,     45.
 ↪,     16.]
soil_2  = [ 100.,  100.,   98.,    94.,  70.,     19.,     15.,      8.,      3.
 ↪,      2.]
soil_3  = [  93.,   91.,   88.,    85.,  69.,     44.,     40.,     27.,     13.
 ↪,      6.]
soil_4  = [ 100.,  100.,  100.,   100., 100.,     97.,     92.,     75.,     47.
 ↪,     31.]

df_1 = pd.DataFrame(data=soil_1, index=soil_mm_int, columns=['sample_percent'])
df_1.index.name = 'soil_mm'

df_2 = pd.DataFrame(data=soil_2, index=soil_mm_int, columns=['sample_percent'])
df_2.index.name = 'soil_mm'

df_3 = pd.DataFrame(data=soil_3, index=soil_mm_int, columns=['sample_percent'])
df_3.index.name = 'soil_mm'

df_4 = pd.DataFrame(data=soil_4, index=soil_mm_int, columns=['sample_percent'])
df_4.index.name = 'soil_mm'

# df4.index = df4.index.map(str)
```

[4]:
```
# < 2mm calculation

# soil 1
soil_1_2mm = df_1[df_1.index == 2.].sample_percent.values[0]
df_1['lt_2mm_percent'] = df_1[df_1.index <= 2.].apply(lambda x: round(x /␣
 ↪soil_1_2mm * 100, 2))
```

```
# soil 2
soil_2_2mm = df_2[df_2.index == 2.].sample_percent.values[0]
df_2['lt_2mm_percent'] = df_2[df_2.index <= 2.].apply(lambda x: round(x /␣
↪soil_2_2mm * 100, 2))

# soil 3
soil_3_2mm = df_3[df_3.index == 2.].sample_percent.values[0]
df_3['lt_2mm_percent'] = df_3[df_3.index <= 2.].apply(lambda x: round(x /␣
↪soil_3_2mm * 100, 2))

# soil 4
soil_4_2mm = df_4[df_4.index == 2.].sample_percent.values[0]
df_4['lt_2mm_percent'] = df_4[df_4.index <= 2.].apply(lambda x: round(x /␣
↪soil_4_2mm * 100, 2))
```

[5]:
```
fig = initialize_fig(title = 'Diameter (mm) Vs Percentage finer (%)',
                     x_axis_label = 'Diameter (mm) [Log-scale]',
                     y_axis_label = 'Percentage finer (%)',
                     x_axis_type = 'log',
                     tooltips = [
                         ('dimeter(mm)', '$x'),
                         ('% finer'    , '$y'),
                     ],
                     plot_height = 300,
                    )

fig.title.text_font_size = '15pt'
fig.xaxis.axis_label_text_font_size = '15pt'
fig.yaxis.axis_label_text_font_size = '15pt'

fig.circle(df_1.index, df_1.sample_percent, fill_color='red', size=10,␣
↪legend_label='soil 1 (sample percent)')
fig.line(df_1.index, df_1.sample_percent, line_width=3, line_color='red',␣
↪legend_label=f'soil 1 (sample percent)')
fig.circle(df_1.index, df_1.lt_2mm_percent, fill_color='maroon', size=10,␣
↪legend_label='soil 1 (finer percent)')
fig.line(df_1.index, df_1.lt_2mm_percent, line_width=3, line_color='maroon',␣
↪legend_label=f'soil 1 (finer percent)')

fig.circle(df_2.index, df_2.sample_percent, fill_color='green', size=10,␣
↪legend_label='soil 2 (sample percent)')
fig.line(df_2.index, df_2.sample_percent, line_width=3, line_color='green',␣
↪legend_label=f'soil 2 (sample percent)')
fig.circle(df_2.index, df_2.lt_2mm_percent, fill_color='lime', size=10,␣
↪legend_label='soil 2 (finer percent)')
```

```
fig.line(df_2.index, df_2.lt_2mm_percent, line_width=3, line_color='lime',␣
 ↪legend_label=f'soil 2 (finer percent)')

fig.circle(df_3.index, df_3.sample_percent, fill_color='blue', size=10,␣
 ↪legend_label='soil 3 (sample percent)')
fig.line(df_3.index, df_3.sample_percent, line_width=3, line_color='blue',␣
 ↪legend_label=f'soil 3 (sample percent)')
fig.circle(df_3.index, df_3.lt_2mm_percent, fill_color='navy', size=10,␣
 ↪legend_label='soil 3 (finer percent)')
fig.line(df_3.index, df_3.lt_2mm_percent, line_width=3, line_color='navy',␣
 ↪legend_label=f'soil 3 (finer percent)')

fig.circle(df_4.index, df_4.sample_percent, fill_color='purple', size=10,␣
 ↪legend_label='soil 4 (sample percent)')
fig.line(df_4.index, df_4.sample_percent, line_width=3, line_color='purple',␣
 ↪legend_label=f'soil 4 (sample percent)')
fig.circle(df_4.index, df_4.lt_2mm_percent, fill_color='fuchsia', size=10,␣
 ↪legend_label='soil 4 (finer percent)')
fig.line(df_4.index, df_4.lt_2mm_percent, line_width=3, line_color='fuchsia',␣
 ↪legend_label=f'soil 4 (finer percent)')


fig.legend.location = 'top_left'
fig.legend.click_policy='hide'
bp.show(fig)
```

[6]:
```
# Texture calculation

# soil 1
x1 = df_1.iloc[5:].index.values.tolist()
y1 = df_1.iloc[5:].sample_percent.values.tolist()
coefficients1, _ = np.polynomial.polynomial.polyfit(x1, y1, 1, full=True)
coefficients1 = coefficients1[::-1]
m1, c1 = coefficients1[0], coefficients1[1]
p1_05 = m1 * 0.05 + c1

# soil 2
x2 = df_2.iloc[1:].index.values.tolist()
y2 = df_2.iloc[1:].sample_percent.values.tolist()
coefficients2, _ = np.polynomial.polynomial.polyfit(x2, y2, 1, full=True)
coefficients2 = coefficients2[::-1]
m2, c2 = coefficients2[0], coefficients2[1]
p2_05 = m2 * 0.05 + c2

# soil 3
x3 = df_3.index.values.tolist()
```

```
y3 = df_3.sample_percent.values.tolist()
coefficients3, _ = np.polynomial.polynomial.polyfit(x3, y3, 1, full=True)
coefficients3 = coefficients3[::-1]
m3, c3 = coefficients3[0], coefficients3[1]
p3_05 = m3 * 0.05 + c3

# soil 3
x4 = df_4.iloc[4:].index.values.tolist()
y4 = df_4.iloc[4:].sample_percent.values.tolist()
coefficients4, _ = np.polynomial.polynomial.polyfit(x4, y4, 1, full=True)
coefficients4 = coefficients4[::-1]
m4, c4 = coefficients4[0], coefficients4[1]
p4_05 = m4 * 0.05 + c4

print(f'p1_05: {p1_05}')
print(f'p2_05: {p2_05}')
print(f'p3_05: {p3_05}')
print(f'p4_05: {p4_05}')

# since these values are significantly off just by looking at the table.
# Thus to avoid significant difference, we will just guess the value
s1_sample_05 = 89.
s2_sample_05 = 12.
s3_sample_05 = 34.
s4_sample_05 = 85.

s1_lt_05     = 89.
s2_lt_05     = 16.43
s3_lt_05     = 49.85
s4_lt_05     = 85.

df_1.loc[0.05] = [s1_sample_05, s1_lt_05]
df_2.loc[0.05] = [s2_sample_05, s2_lt_05]
df_3.loc[0.05] = [s3_sample_05, s3_lt_05]
df_4.loc[0.05] = [s4_sample_05, s4_lt_05]
```

```
p1_05: 60.66839664793997
p2_05: 24.384433872847595
p3_05: 44.15098690890004
p4_05: 66.12465271160718
```

[7]:
```
df_1 = df_1.sort_index(ascending=False)
df_2 = df_2.sort_index(ascending=False)
df_3 = df_3.sort_index(ascending=False)
df_4 = df_4.sort_index(ascending=False)
```

[8]:
```
df_2
```

[8]:

soil_mm	sample_percent	lt_2mm_percent
50.000	100.0	NaN
19.000	100.0	NaN
9.500	98.0	NaN
4.760	94.0	NaN
2.000	70.0	100.00
0.420	19.0	27.14
0.074	15.0	21.43
0.050	12.0	16.43
0.020	8.0	11.43
0.005	3.0	4.29
0.002	2.0	2.86

[9]:
```
soil_texture   =      ['gravel(d>2mm)', 'sand(0.05<=d<2mm)', 'silt(0.002<=d<0.
 ↪05mm)', 'clay(d<0.002mm)']
# total sample
soil_1_sample  =      [     0.,             11.,                73.,        ␣
 ↪           16. ]
soil_2_sample  =      [    30.,             58.,                10.,        ␣
 ↪            2. ]
soil_3_sample  =      [    31.,             35.,                28.,        ␣
 ↪            6. ]
soil_4_sample  =      [     0.,             15.,                54.,        ␣
 ↪           31. ]

# lt 2 mm sample
soil_1_lt_2    =      [  np.nan,             11.,                73.,        ␣
 ↪           16. ]
soil_2_lt_2    =      [  np.nan,          83.56,              13.58,         ␣
 ↪         2.86 ]
soil_3_lt_2    =      [  np.nan,          50.15,              41.15,         ␣
 ↪         8.70 ]
soil_4_lt_2    =      [  np.nan,             15.,                54.,        ␣
 ↪           31. ]
```

[10]:
```
# Looking at the figure 7.5 to classify the sand
print(f'The soil 1.1 is silt Loam.')
print(f'The soil 1.2 is Loamy Sand.')
print(f'The soil 1.3 is sandy Loam.')
print(f'The soil 1.4 is Silty Clay Loam.')
```

```
The soil 1.1 is silt Loam.
The soil 1.2 is Loamy Sand.
The soil 1.3 is sandy Loam.
The soil 1.4 is Silty Clay Loam.
```

[ ]:

[ ]:

[ ]:

0.0.2 Question 2

Compare the Brooks and Corey, Campbell and van Genuchten k( ) − analytic relationship (i.e. moisture conductivity) for a silty clay loam. Plot the data and briefly discuss the differences. Use _r = 0.057 . For the van Genuchten parameters: n = 1.24 (vg)* = *

[ ]:

[11]:
```
n = 1.24 # parameter that depends on pore-size distribution
porosity = 0.477 # porosity
 r = 0.057 # residual water content
Kh = 1.7e-4 # cm/s saturated hydraulic conductivity
b = 7.75 # parameter that depends upon pore size distribution
```

[12]:
```
# index is the
df = pd.DataFrame(index = np.arange(0., 1.05, .025), columns = ['BC', 'C',␣
 ↪'VG'])
```

[13]:
```
df[' *'] = (df.index - r) / (porosity - r)
```

[14]:
```
df
```

[14]:

	BC	C	VG	*
0.000	NaN	NaN	NaN	-0.135714
0.025	NaN	NaN	NaN	-0.076190
0.050	NaN	NaN	NaN	-0.016667
0.075	NaN	NaN	NaN	0.042857
0.100	NaN	NaN	NaN	0.102381
0.125	NaN	NaN	NaN	0.161905
0.150	NaN	NaN	NaN	0.221429
0.175	NaN	NaN	NaN	0.280952
0.200	NaN	NaN	NaN	0.340476
0.225	NaN	NaN	NaN	0.400000
0.250	NaN	NaN	NaN	0.459524
0.275	NaN	NaN	NaN	0.519048
0.300	NaN	NaN	NaN	0.578571
0.325	NaN	NaN	NaN	0.638095
0.350	NaN	NaN	NaN	0.697619
0.375	NaN	NaN	NaN	0.757143
0.400	NaN	NaN	NaN	0.816667

```
0.425  NaN  NaN  NaN  0.876190
0.450  NaN  NaN  NaN  0.935714
0.475  NaN  NaN  NaN  0.995238
0.500  NaN  NaN  NaN  1.054762
0.525  NaN  NaN  NaN  1.114286
0.550  NaN  NaN  NaN  1.173810
0.575  NaN  NaN  NaN  1.233333
0.600  NaN  NaN  NaN  1.292857
0.625  NaN  NaN  NaN  1.352381
0.650  NaN  NaN  NaN  1.411905
0.675  NaN  NaN  NaN  1.471429
0.700  NaN  NaN  NaN  1.530952
0.725  NaN  NaN  NaN  1.590476
0.750  NaN  NaN  NaN  1.650000
0.775  NaN  NaN  NaN  1.709524
0.800  NaN  NaN  NaN  1.769048
0.825  NaN  NaN  NaN  1.828571
0.850  NaN  NaN  NaN  1.888095
0.875  NaN  NaN  NaN  1.947619
0.900  NaN  NaN  NaN  2.007143
0.925  NaN  NaN  NaN  2.066667
0.950  NaN  NaN  NaN  2.126190
0.975  NaN  NaN  NaN  2.185714
1.000  NaN  NaN  NaN  2.245238
1.025  NaN  NaN  NaN  2.304762
```

[15]:
```
df['BC'] = (df[' *']**(2. * b + 3)) * Kh
```

[16]:
```
df
```

[16]:

	BC	C	VG	*
0.000	NaN	NaN	NaN	-0.135714
0.025	NaN	NaN	NaN	-0.076190
0.050	NaN	NaN	NaN	-0.016667
0.075	8.372948e-30	NaN	NaN	0.042857
0.100	8.308134e-23	NaN	NaN	0.102381
0.125	3.997164e-19	NaN	NaN	0.161905
0.150	1.310001e-16	NaN	NaN	0.221429
0.175	1.071900e-14	NaN	NaN	0.280952
0.200	3.750194e-13	NaN	NaN	0.340476
0.225	7.388542e-12	NaN	NaN	0.400000
0.250	9.619384e-11	NaN	NaN	0.459524
0.275	9.157611e-10	NaN	NaN	0.519048
0.300	6.824221e-09	NaN	NaN	0.578571
0.325	4.176676e-08	NaN	NaN	0.638095
0.350	2.174643e-07	NaN	NaN	0.697619
0.375	9.890967e-07	NaN	NaN	0.757143

```
0.400  4.011237e-06  NaN  NaN  0.816667
0.425  1.474082e-05  NaN  NaN  0.876190
0.450  4.972780e-05  NaN  NaN  0.935714
0.475  1.556318e-04  NaN  NaN  0.995238
0.500  4.558332e-04  NaN  NaN  1.054762
0.525  1.258593e-03  NaN  NaN  1.114286
0.550  3.296135e-03  NaN  NaN  1.173810
0.575  8.230716e-03  NaN  NaN  1.233333
0.600  1.968493e-02  NaN  NaN  1.292857
0.625  4.526695e-02  NaN  NaN  1.352381
0.650  1.004266e-01  NaN  NaN  1.411905
0.675  2.155876e-01  NaN  NaN  1.471429
0.700  4.489935e-01  NaN  NaN  1.530952
0.725  9.092883e-01  NaN  NaN  1.590476
0.750  1.794328e+00  NaN  NaN  1.650000
0.775  3.456522e+00  NaN  NaN  1.709524
0.800  6.510736e+00  NaN  NaN  1.769048
0.825  1.200935e+01  NaN  NaN  1.828571
0.850  2.172156e+01  NaN  NaN  1.888095
0.875  3.857209e+01  NaN  NaN  1.947619
0.900  6.732048e+01  NaN  NaN  2.007143
0.925  1.155984e+02  NaN  NaN  2.066667
0.950  1.954740e+02  NaN  NaN  2.126190
0.975  3.257817e+02  NaN  NaN  2.185714
1.000  5.355541e+02  NaN  NaN  2.245238
1.025  8.690228e+02  NaN  NaN  2.304762
```

[17]:
```
df['C'] = (df.index / porosity)**(2. * b + 3) * Kh
```

[18]:
```
df
```

[18]:
```
                BC             C   VG         *
0.000          NaN  0.000000e+00  NaN -0.135714
0.025          NaN  3.465366e-28  NaN -0.076190
0.050          NaN  1.284707e-22  NaN -0.016667
0.075  8.372948e-30  2.325371e-19  NaN  0.042857
0.100  8.308134e-23  4.762762e-17  NaN  0.102381
0.125  3.997164e-19  2.955930e-15  NaN  0.161905
0.150  1.310001e-16  8.620793e-14  NaN  0.221429
0.175  1.071900e-14  1.493010e-12  NaN  0.280952
0.200  3.750194e-13  1.765687e-11  NaN  0.340476
0.225  7.388542e-12  1.560398e-10  NaN  0.400000
0.250  9.619384e-11  1.095845e-09  NaN  0.459524
0.275  9.157611e-10  6.390190e-09  NaN  0.519048
0.300  6.824221e-09  3.195966e-08  NaN  0.578571
0.325  4.176676e-08  1.405080e-07  NaN  0.638095
0.350  2.174643e-07  5.535001e-07  NaN  0.697619
```

```
0.375  9.890967e-07  1.983524e-06  NaN  0.757143
0.400  4.011237e-06  6.545891e-06  NaN  0.816667
0.425  1.474082e-05  2.009343e-05  NaN  0.876190
0.450  4.972780e-05  5.784828e-05  NaN  0.935714
0.475  1.556318e-04  1.572862e-04  NaN  0.995238
0.500  4.558332e-04  4.062599e-04  NaN  1.054762
0.525  1.258593e-03  1.001858e-03  NaN  1.114286
0.550  3.296135e-03  2.369020e-03  NaN  1.173810
0.575  8.230716e-03  5.391555e-03  NaN  1.233333
0.600  1.968493e-02  1.184833e-02  NaN  1.292857
0.625  4.526695e-02  2.521385e-02  NaN  1.352381
0.650  1.004266e-01  5.209020e-02  NaN  1.411905
0.675  2.155876e-01  1.047077e-01  NaN  1.471429
0.700  4.489935e-01  2.051978e-01  NaN  1.530952
0.725  9.092883e-01  3.927462e-01  NaN  1.590476
0.750  1.794328e+00  7.353469e-01  NaN  1.650000
0.775  3.456522e+00  1.348778e+00  NaN  1.709524
0.800  6.510736e+00  2.426742e+00  NaN  1.769048
0.825  1.200935e+01  4.288022e+00  NaN  1.828571
0.850  2.172156e+01  7.449190e+00  NaN  1.888095
0.875  3.857209e+01  1.273526e+01  NaN  1.947619
0.900  6.732048e+01  2.144595e+01  NaN  2.007143
0.925  1.155984e+02  3.560254e+01  NaN  2.066667
0.950  1.954740e+02  5.831034e+01  NaN  2.126190
0.975  3.257817e+02  9.428537e+01  NaN  2.185714
1.000  5.355541e+02  1.506118e+02  NaN  2.245238
1.025  8.690228e+02  2.378211e+02  NaN  2.304762
```

[19]:
```
# Since  vg* =  *
df[' vg*'] = df[' *']
```

[20]:
```
df['VG_1'] = df[' vg*']**(1/2)
df['VG_2'] = 1 - (df[' vg*'])**(n/(n-1))
df['VG_3'] = (df['VG_2'])**((n - 1)/n)
df['VG_4'] = (1 - df['VG_3'])**2

# df['VG'] = df[' vg*']**0.5 * (1 - (1 - df[' vg*']**(n/(n-1)))**((n - 1)/n))**2␣
 ↪* Kh
df['VG'] = df['VG_1'] * df['VG_4'] * Kh
df = df.drop(columns=['VG_1', 'VG_2', 'VG_3', 'VG_4'], axis=1)
# df = df.dropna()
```

[21]:
```
df
```

[21]:
```
                BC             C            VG         *       vg*
0.000          NaN  0.000000e+00           NaN -0.135714 -0.135714
0.025          NaN  3.465366e-28           NaN -0.076190 -0.076190
```

0.050	NaN	1.284707e-22	NaN	-0.016667	-0.016667
0.075	8.372948e-30	2.325371e-19	9.644496e-21	0.042857	0.042857
0.100	8.308134e-23	4.762762e-17	1.206160e-16	0.102381	0.102381
0.125	3.997164e-19	2.955930e-15	1.728633e-14	0.161905	0.161905
0.150	1.310001e-16	8.620793e-14	5.138910e-13	0.221429	0.221429
0.175	1.071900e-14	1.493010e-12	6.782330e-12	0.280952	0.280952
0.200	3.750194e-13	1.765687e-11	5.448901e-11	0.340476	0.340476
0.225	7.388542e-12	1.560398e-10	3.134012e-10	0.400000	0.400000
0.250	9.619384e-11	1.095845e-09	1.419215e-09	0.459524	0.459524
0.275	9.157611e-10	6.390190e-09	5.379781e-09	0.519048	0.519048
0.300	6.824221e-09	3.195966e-08	1.781551e-08	0.578571	0.578571
0.325	4.176676e-08	1.405080e-07	5.322302e-08	0.638095	0.638095
0.350	2.174643e-07	5.535001e-07	1.472770e-07	0.697619	0.697619
0.375	9.890967e-07	1.983524e-06	3.868570e-07	0.757143	0.757143
0.400	4.011237e-06	6.545891e-06	9.912848e-07	0.816667	0.816667
0.425	1.474082e-05	2.009343e-05	2.578740e-06	0.876190	0.876190
0.450	4.972780e-05	5.784828e-05	7.442989e-06	0.935714	0.935714
0.475	1.556318e-04	1.572862e-04	4.458991e-05	0.995238	0.995238
0.500	4.558332e-04	4.062599e-04	NaN	1.054762	1.054762
0.525	1.258593e-03	1.001858e-03	NaN	1.114286	1.114286
0.550	3.296135e-03	2.369020e-03	NaN	1.173810	1.173810
0.575	8.230716e-03	5.391555e-03	NaN	1.233333	1.233333
0.600	1.968493e-02	1.184833e-02	NaN	1.292857	1.292857
0.625	4.526695e-02	2.521385e-02	NaN	1.352381	1.352381
0.650	1.004266e-01	5.209020e-02	NaN	1.411905	1.411905
0.675	2.155876e-01	1.047077e-01	NaN	1.471429	1.471429
0.700	4.489935e-01	2.051978e-01	NaN	1.530952	1.530952
0.725	9.092883e-01	3.927462e-01	NaN	1.590476	1.590476
0.750	1.794328e+00	7.353469e-01	NaN	1.650000	1.650000
0.775	3.456522e+00	1.348778e+00	NaN	1.709524	1.709524
0.800	6.510736e+00	2.426742e+00	NaN	1.769048	1.769048
0.825	1.200935e+01	4.288022e+00	NaN	1.828571	1.828571
0.850	2.172156e+01	7.449190e+00	NaN	1.888095	1.888095
0.875	3.857209e+01	1.273526e+01	NaN	1.947619	1.947619
0.900	6.732048e+01	2.144595e+01	NaN	2.007143	2.007143
0.925	1.155984e+02	3.560254e+01	NaN	2.066667	2.066667
0.950	1.954740e+02	5.831034e+01	NaN	2.126190	2.126190
0.975	3.257817e+02	9.428537e+01	NaN	2.185714	2.185714
1.000	5.355541e+02	1.506118e+02	NaN	2.245238	2.245238
1.025	8.690228e+02	2.378211e+02	NaN	2.304762	2.304762

[22]:
```
df = df.dropna()
```

[23]:
```
df
```

[23]:

	BC	C	VG	*	vg*
0.075	8.372948e-30	2.325371e-19	9.644496e-21	0.042857	0.042857

```
0.100  8.308134e-23  4.762762e-17  1.206160e-16  0.102381  0.102381
0.125  3.997164e-19  2.955930e-15  1.728633e-14  0.161905  0.161905
0.150  1.310001e-16  8.620793e-14  5.138910e-13  0.221429  0.221429
0.175  1.071900e-14  1.493010e-12  6.782330e-12  0.280952  0.280952
0.200  3.750194e-13  1.765687e-11  5.448901e-11  0.340476  0.340476
0.225  7.388542e-12  1.560398e-10  3.134012e-10  0.400000  0.400000
0.250  9.619384e-11  1.095845e-09  1.419215e-09  0.459524  0.459524
0.275  9.157611e-10  6.390190e-09  5.379781e-09  0.519048  0.519048
0.300  6.824221e-09  3.195966e-08  1.781551e-08  0.578571  0.578571
0.325  4.176676e-08  1.405080e-07  5.322302e-08  0.638095  0.638095
0.350  2.174643e-07  5.535001e-07  1.472770e-07  0.697619  0.697619
0.375  9.890967e-07  1.983524e-06  3.868570e-07  0.757143  0.757143
0.400  4.011237e-06  6.545891e-06  9.912848e-07  0.816667  0.816667
0.425  1.474082e-05  2.009343e-05  2.578740e-06  0.876190  0.876190
0.450  4.972780e-05  5.784828e-05  7.442989e-06  0.935714  0.935714
0.475  1.556318e-04  1.572862e-04  4.458991e-05  0.995238  0.995238
```

[24]:
```
fig = initialize_fig(title = 'Volumertic water content ( ) Vs Hydraulic␣
↪Conductivity Kh( )',
                     x_axis_label = 'Volumertic water content ( )',
                     y_axis_label = 'Hydraulic Conductivity Log scale Kh( )',
                     y_axis_type = 'log',
                     tooltips = [
                         (' ', '$x'),
                         ('Kh( )',     '$y'),
                     ],
                     plot_height = 300,
                    )

fig.title.text_font_size = '15pt'
fig.xaxis.axis_label_text_font_size = '15pt'
fig.yaxis.axis_label_text_font_size = '15pt'

fig.circle(df.index, df.BC, fill_color='red', size=10, legend_label=f'Brooks &␣
↪Corey')
fig.line(df.index, df.BC, line_width=3, line_color='red', legend_label=f'Brooks␣
↪& Corey')

fig.circle(df.index, df.C,  fill_color='blue', size=10,␣
↪legend_label=f'Campbell')
fig.line(df.index, df.C, line_width=3, line_color='blue',␣
↪legend_label=f'Campbell')

fig.circle(df.index, df.VG, fill_color='green', size=10, legend_label=f'Van␣
↪Genuchten')
```

```
fig.line(df.index, df.VG, line_width=3, line_color='green', legend_label=f'Van␣
↪Genuchten')

fig.legend.location = 'top_left'
fig.legend.click_policy='hide'
bp.show(fig)
```

The graph above shows the relation between the Volumetric Water Content ( ) plotted in the X-axis and the Hydraulic Conductivity Kh( ) plotted in the Y-axis. The Kh( ) is plotted in log scale. This shows the general relationship where for a soil, the unsaturated hydraulic conductivity is low, and it increases nonlinearly to its saturated value as water content increases to saturation. The Kh( ) is calcualted from 3 methods. Brooks and Corey shows significant difference (several orders of magnitude) with the other two at the lower volumetric water content, but the Kh( ) gets closer as we move towards the higher saturation.

[ ]:

[ ]:

0.0.3 Question 3

[25]:
```
# For various soils

_sand = 0.395 # porosity
_silt_loam = 0.485
_loam = 0.451
_clay = 0.482

_abs_sand = 12.1 # absolute pressure head
_abs_silt_loam = 78.6
_abs_loam = 47.8
_abs_clay = 40.5

Kh_sand = 1.76e-2 # saturated hydraulic conductivity
Kh_silt_loam = 7.20e-4
Kh_loam = 6.95e-4
Kh_clay = 1.28e-4

b_sand = 4.05 # parameter that depends upon pore size distribution
b_silt_loam = 5.30
b_loam = 5.39
b_clay = 11.40

# assume
r = 0.05 # residual water content

# HC: Hydraulic conductivity
```

```
# PH: Pressure head
# index is the
df = pd.DataFrame(index = np.arange(0.01, 0.5, .01), columns=[
    ' *_Sand', ' *_Silt_Loam', ' *_Loam', ' *_Clay',
    'HC_Sand', 'HC_Silt_Loam', 'HC_Loam', 'HC_Clay',
    'PH_Sand', 'PH_Silt_Loam', 'PH_Loam', 'PH_Clay',
])

df[' *_Sand'] = (df.index - r) / (_sand - r)
df[' *_Silt_Loam'] = (df.index - r) / (_silt_loam - r)
df[' *_Loam'] = (df.index - r) / (_loam - r)
df[' *_Clay'] = (df.index - r) / (_clay - r)

df['HC_Sand'] = (df[' *_Sand']**(2. * b_sand + 3.)) * Kh_sand
df['HC_Silt_Loam'] = (df[' *_Silt_Loam']**(2. * b_silt_loam + 3.)) *␣
↪Kh_silt_loam
df['HC_Loam'] = (df[' *_Loam']**(2. * b_loam + 3.)) * Kh_loam
df['HC_Clay'] = (df[' *_Clay']**(2. * b_clay + 3.)) * Kh_clay

df['PH_Sand'] = np.where(df[' *_Sand'] < 1, _abs_sand * (1 /␣
↪df[' *_Sand'])**b_sand, _abs_sand)
df['PH_Silt_Loam'] = np.where(df[' *_Silt_Loam'] < 1, _abs_silt_loam * (1 /␣
↪df[' *_Silt_Loam'])**b_silt_loam, _abs_silt_loam)
df['PH_Loam'] = np.where(df[' *_Loam'] < 1, _abs_loam * (1 /␣
↪df[' *_Loam'])**b_loam, _abs_loam)
df['PH_Clay'] = np.where(df[' *_Clay'] < 1, _abs_clay * (1 /␣
↪df[' *_Clay'])**b_clay, _abs_clay)
```

[26]:
```
df
```

[26]:

	*_Sand	*_Silt_Loam	*_Loam	*_Clay	HC_Sand	HC_Silt_Loam	\
0.01	-0.115942	-0.091954	-0.099751	-0.092593	NaN	NaN	
0.02	-0.086957	-0.068966	-0.074813	-0.069444	NaN	NaN	
0.03	-0.057971	-0.045977	-0.049875	-0.046296	NaN	NaN	
0.04	-0.028986	-0.022989	-0.024938	-0.023148	NaN	NaN	
0.05	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000e+00	
0.06	0.028986	0.022989	0.024938	0.023148	1.498721e-19	3.748682e-26	
0.07	0.057971	0.045977	0.049875	0.046296	3.289680e-16	4.654645e-22	
0.08	0.086957	0.068966	0.074813	0.069444	2.963239e-14	1.155388e-19	
0.09	0.115942	0.091954	0.099751	0.092593	7.220822e-13	5.779556e-18	
0.10	0.144928	0.114943	0.124688	0.115741	8.595829e-12	1.201906e-16	
0.11	0.173913	0.137931	0.149626	0.138889	6.504285e-11	1.434616e-15	
0.12	0.202899	0.160920	0.174564	0.162037	3.600054e-10	1.167379e-14	
0.13	0.231884	0.183908	0.199501	0.185185	1.584965e-09	7.176330e-14	
0.14	0.260870	0.206897	0.224439	0.208333	5.858853e-09	3.561039e-13	
0.15	0.289855	0.229885	0.249377	0.231481	1.886778e-08	1.492377e-12	
0.16	0.318841	0.252874	0.274314	0.254630	5.434749e-08	5.455306e-12	

0.17	0.347826	0.275862	0.299252	0.277778	1.427685e-07	1.781327e-11
0.18	0.376812	0.298851	0.324190	0.300926	3.471298e-07	5.290640e-11
0.19	0.405797	0.321839	0.349127	0.324074	7.902089e-07	1.449506e-10
0.20	0.434783	0.344828	0.374065	0.347222	1.699549e-06	3.704416e-10
0.21	0.463768	0.367816	0.399002	0.370370	3.478985e-06	8.910670e-10
0.22	0.492754	0.390805	0.423940	0.393519	6.818734e-06	2.032282e-09
0.23	0.521739	0.413793	0.448878	0.416667	1.286013e-05	4.421653e-09
0.24	0.550725	0.436782	0.473815	0.439815	2.343612e-05	9.224176e-09
0.25	0.579710	0.459770	0.498753	0.462963	4.141462e-05	1.853047e-08
0.26	0.608696	0.482759	0.523691	0.486111	7.117949e-05	3.597999e-08
0.27	0.637681	0.505747	0.548628	0.509259	1.192923e-04	6.773717e-08
0.28	0.666667	0.528736	0.573566	0.532407	1.953888e-04	1.239873e-07
0.29	0.695652	0.551724	0.598504	0.555556	3.133758e-04	2.211829e-07
0.30	0.724638	0.574713	0.623441	0.578704	4.930090e-04	3.853565e-07
0.31	0.753623	0.597701	0.648379	0.601852	7.619472e-04	6.569255e-07
0.32	0.782609	0.620690	0.673317	0.625000	1.158400e-03	1.097554e-06
0.33	0.811594	0.643678	0.698254	0.648148	1.734502e-03	1.799815e-06
0.34	0.840580	0.666667	0.723192	0.671296	2.560581e-03	2.900624e-06
0.35	0.869565	0.689655	0.748130	0.694444	3.730497e-03	4.599680e-06
0.36	0.898551	0.712644	0.773067	0.717593	5.368284e-03	7.184518e-06
0.37	0.927536	0.735632	0.798005	0.740741	7.636344e-03	1.106416e-05
0.38	0.956522	0.758621	0.822943	0.763889	1.074547e-02	1.681389e-05
0.39	0.985507	0.781609	0.847880	0.787037	1.496707e-02	2.523434e-05
0.40	1.014493	0.804598	0.872818	0.810185	2.064791e-02	3.742864e-05
0.41	1.043478	0.827586	0.897756	0.833333	2.822789e-02	5.490256e-05
0.42	1.072464	0.850575	0.922693	0.856481	3.826131e-02	7.969334e-05
0.43	1.101449	0.873563	0.947631	0.879630	5.144209e-02	1.145343e-04
0.44	1.130435	0.896552	0.972569	0.902778	6.863377e-02	1.630637e-04
0.45	1.159420	0.919540	0.997506	0.925926	9.090476e-02	2.300883e-04
0.46	1.188406	0.942529	1.022444	0.949074	1.195698e-01	3.219134e-04
0.47	1.217391	0.965517	1.047382	0.972222	1.562384e-01	4.467545e-04
0.48	1.246377	0.988506	1.072319	0.995370	2.028713e-01	6.152471e-04
0.49	1.275362	1.011494	1.097257	1.018519	2.618457e-01	8.410755e-04

	HC_Loam	HC_Clay	PH_Sand	PH_Silt_Loam	PH_Loam	\
0.01	NaN	NaN	NaN	NaN	NaN	
0.02	NaN	NaN	NaN	NaN	NaN	
0.03	NaN	NaN	NaN	NaN	NaN	
0.04	NaN	NaN	NaN	NaN	NaN	
0.05	0.000000e+00	0.000000e+00	inf	inf	inf	
0.06	5.631905e-26	8.160735e-47	2.046222e+07	3.796782e+10	2.091106e+10	
0.07	7.922253e-22	4.767637e-39	1.235325e+06	9.637328e+08	4.986828e+08	
0.08	2.115370e-19	1.665167e-34	2.391176e+05	1.123758e+08	5.606509e+07	
0.09	1.114402e-17	2.785333e-31	7.457783e+04	2.446232e+07	1.189249e+07	
0.10	2.412470e-16	8.813642e-29	3.020815e+04	7.496777e+06	3.572134e+06	
0.11	2.975634e-15	9.728179e-27	1.443578e+04	2.852423e+06	1.337029e+06	
0.12	2.489467e-14	5.191092e-25	7.732246e+03	1.260074e+06	5.825023e+05	

0.13	1.567600e-13	1.627238e-23	4.502340e+03	6.209243e+05	2.836097e+05
0.14	7.945425e-13	3.397708e-22	2.794284e+03	3.326062e+05	1.503173e+05
0.15	3.393559e-12	5.149076e-21	1.823697e+03	1.902898e+05	8.518754e+04
0.16	1.261963e-11	6.020888e-20	1.239688e+03	1.148245e+05	5.096470e+04
0.17	4.185746e-11	5.683364e-19	8.715025e+02	7.240274e+04	3.188520e+04
0.18	1.261231e-10	4.481958e-18	6.302054e+02	4.737136e+04	2.071197e+04
0.19	3.501867e-10	3.032722e-17	4.668036e+02	3.198431e+04	1.389140e+04
0.20	9.061350e-10	1.798390e-16	3.530068e+02	2.218870e+04	9.577324e+03
0.21	2.205102e-09	9.506592e-16	2.718108e+02	1.576085e+04	6.763467e+03
0.22	5.084421e-09	4.542673e-15	2.126349e+02	1.142974e+04	4.878164e+03
0.23	1.117662e-08	1.984997e-14	1.686938e+02	8.442505e+03	3.584737e+03
0.24	2.354398e-08	8.008888e-14	1.355191e+02	6.339011e+03	2.678515e+03
0.25	4.773629e-08	3.008176e-13	1.100985e+02	4.830106e+03	2.031535e+03
0.26	9.350552e-08	1.059222e-12	9.035757e+01	3.729523e+03	1.561759e+03
0.27	1.775170e-07	3.517503e-12	7.484122e+01	2.914576e+03	1.215396e+03
0.28	3.275404e-07	1.107408e-11	6.251079e+01	2.302810e+03	9.564517e+02
0.29	5.887979e-07	3.320315e-11	5.261350e+01	1.837791e+03	7.603919e+02
0.30	1.033401e-06	9.518785e-11	4.459598e+01	1.480245e+03	6.102101e+02
0.31	1.774141e-06	2.618434e-10	3.804614e+01	1.202422e+03	4.939349e+02
0.32	2.984338e-06	6.932892e-10	3.265348e+01	9.844364e+02	4.030189e+02
0.33	4.925984e-06	1.771767e-09	2.818141e+01	8.118542e+02	3.312793e+02
0.34	7.989135e-06	4.381246e-09	2.444786e+01	6.740719e+02	2.741895e+02
0.35	1.274636e-05	1.050649e-08	2.131138e+01	5.632134e+02	2.283981e+02
0.36	2.002717e-05	2.448280e-08	1.866113e+01	4.733677e+02	1.913976e+02
0.37	3.101859e-05	5.553909e-08	1.640950e+01	4.000559e+02	1.612938e+02
0.38	4.739990e-05	1.228539e-07	1.448676e+01	3.398535e+02	1.366425e+02
0.39	7.152121e-05	2.653905e-07	1.283699e+01	2.901198e+02	1.163335e+02
0.40	1.066382e-04	5.606410e-07	1.210000e+01	2.488026e+02	9.950601e+01
0.41	1.572186e-04	1.159668e-06	1.210000e+01	2.142952e+02	8.548808e+01
0.42	2.293375e-04	2.351439e-06	1.210000e+01	1.853303e+02	7.375115e+01
0.43	3.311870e-04	4.678925e-06	1.210000e+01	1.609024e+02	6.387670e+01
0.44	4.737243e-04	9.145262e-06	1.210000e+01	1.402081e+02	5.553130e+01
0.45	6.714938e-04	1.757426e-05	1.210000e+01	1.226020e+02	4.844765e+01
0.46	9.436629e-04	3.323172e-05	1.210000e+01	1.075626e+02	4.780000e+01
0.47	1.315317e-03	6.188152e-05	1.210000e+01	9.466607e+01	4.780000e+01
0.48	1.819075e-03	1.135574e-04	1.210000e+01	8.356661e+01	4.780000e+01
0.49	2.497084e-03	2.054984e-04	1.210000e+01	7.860000e+01	4.780000e+01

	PH_Clay
0.01	NaN
0.02	NaN
0.03	NaN
0.04	NaN
0.05	inf
0.06	1.786363e+20
0.07	6.610401e+16
0.08	6.498116e+14

```
0.09  2.446165e+13
0.10  1.921815e+12
0.11  2.404615e+11
0.12  4.148127e+10
0.13  9.051986e+09
0.14  2.363770e+09
0.15  7.111640e+08
0.16  2.399347e+08
0.17  8.898228e+07
0.18  3.572856e+07
0.19  1.535006e+07
0.20  6.990840e+06
0.21  3.349669e+06
0.22  1.678232e+06
0.23  8.747082e+05
0.24  4.722573e+05
0.25  2.631648e+05
0.26  1.508932e+05
0.27  8.878737e+04
0.28  5.348998e+04
0.29  3.292771e+04
0.30  2.067539e+04
0.31  1.322128e+04
0.32  8.598503e+03
0.33  5.680260e+03
0.34  3.807436e+03
0.35  2.586946e+03
0.36  1.780108e+03
0.37  1.239538e+03
0.38  8.727921e+02
0.39  6.210264e+02
0.40  4.462662e+02
0.41  3.236840e+02
0.42  2.368482e+02
0.43  1.747579e+02
0.44  1.299670e+02
0.45  9.738360e+01
0.46  7.349093e+01
0.47  5.583774e+01
0.48  4.270014e+01
0.49  4.050000e+01
```

[27]:
```
# | | vs
fig = initialize_fig(title = '| |- plot for various soil',
                     x_axis_label = 'Volumertic water content ( )',
                     y_axis_label = 'Pressure Head | |( )',
                     y_axis_type = 'log',
```

```
                      tooltips = [
                          (' ',      '$x'),
                          ('| |( )', '$y'),
                      ],
                      plot_height = 300,
                     )

fig.title.text_font_size = '15pt'
fig.xaxis.axis_label_text_font_size = '15pt'
fig.yaxis.axis_label_text_font_size = '15pt'
fig.circle(df.index, df.PH_Sand, fill_color='red', size=10,␣
 ↪legend_label=f'Sand')
fig.line(df.index, df.PH_Sand, line_width=3, line_color='red',␣
 ↪legend_label=f'Sand')

fig.circle(df.index, df.PH_Silt_Loam, fill_color='blue', size=10,␣
 ↪legend_label=f'Silt Loam')
fig.line(df.index, df.PH_Silt_Loam, line_width=3, line_color='blue',␣
 ↪legend_label=f'Silt Loam')

fig.circle(df.index, df.PH_Loam,  fill_color='green', size=10,␣
 ↪legend_label=f'Loam')
fig.line(df.index, df.PH_Loam, line_width=3, line_color='green',␣
 ↪legend_label=f'Loam')

fig.circle(df.index, df.PH_Clay, fill_color='fuchsia', size=10,␣
 ↪legend_label=f'Clay')
fig.line(df.index, df.PH_Clay, line_width=3, line_color='fuchsia',␣
 ↪legend_label=f'Clay')

fig.legend.location = 'top_right'
fig.legend.click_policy='hide'
bp.show(fig)
```

The Pressure Head $||( )$ and Volumertic water content $( )$ relation is highly non-linear as can be seen from above graph. Note that the Y-axis plots the absolute value. So a high absolute pressure head means more tension in the soil. For soils, as the porosity increases, the pressure head also increases. For example, the clay has very high pressure head compared to the sand. At very high tension, the curve is nearly vertical and reflects the residual water content held in the soil against all forces. As the water content in the soil increases, the tension also decreases and becomes zero (0) when the water content equals the porosity (saturation). Also can be noticed that as the water content increases the difference in the tension for various soil also decreases.

[28]:
```
# Kh vs
fig = initialize_fig(title = 'Kh-  plot for various soil',
                     x_axis_label = 'Volumertic water content ( )',
                     y_axis_label = 'Hydraulic Conductivity Kh( )',
```

```
                      y_axis_type = 'log',
                      tooltips = [
                          (' ',     '$x'),
                          ('Kh( )', '$y'),
                      ],
                      plot_height = 300,
                     )

fig.title.text_font_size = '15pt'
fig.xaxis.axis_label_text_font_size = '15pt'
fig.yaxis.axis_label_text_font_size = '15pt'
fig.circle(df.index, df.HC_Sand, fill_color='red', size=10,␣
 ↪legend_label=f'Sand')
fig.line(df.index, df.HC_Sand, line_width=3, line_color='red',␣
 ↪legend_label=f'Sand')

fig.circle(df.index, df.HC_Silt_Loam, fill_color='blue', size=10,␣
 ↪legend_label=f'Silt Loam')
fig.line(df.index, df.HC_Silt_Loam, line_width=3, line_color='blue',␣
 ↪legend_label=f'Silt Loam')

fig.circle(df.index, df.HC_Loam,  fill_color='green', size=10,␣
 ↪legend_label=f'Loam')
fig.line(df.index, df.HC_Loam, line_width=3, line_color='green',␣
 ↪legend_label=f'Loam')

fig.circle(df.index, df.HC_Clay, fill_color='fuchsia', size=10,␣
 ↪legend_label=f'Clay')
fig.line(df.index, df.HC_Clay, line_width=3, line_color='fuchsia',␣
 ↪legend_label=f'Clay')

fig.legend.location = 'top_left'
fig.legend.click_policy='hide'
bp.show(fig)
```

The graph above shows the relation between the Volumetric Water Content ( ) plotted in the X-axis and the Hydraulic Conductivity Kh( ) plotted in the Y-axis. The Kh( ) is plotted in log scale. This shows the general relationship where for a soil, the unsaturated hydraulic conductivity is low, and it increases nonlinearly to its saturated value as water content increases to saturation. For soil, as the porosity increases the hydraulic conductivity decreases as it becomes increasingly difficult for water to flow through highly porous soil. Thus the clay which is highly porous soil has much less conductivity than sand, and they differ by several orders of magnitude, but the Kh( ) gets closer as we move towards the higher saturation.

[ ]:

[ ]:

Question 4 The 50-year return period, one-hour design storm for a location in Alabama is, is 4.9 inches. For a clay-loam soil, apply the Green and Ampt model to estimate the infiltration and runoff resulting from this storm. Assume that the soil has a moisture content of 0.2 at the beginning of the storm. Compare your results with an initial moisture content of _i=0.10. Compare your results with an intensity of 2 in/hr. How would your results look if the soil was a sandy loam instead? Plot F(t) vs t and f(t) vs t for each scenario. Discuss your results in a paragraph. Ks = 0.1 cm/hr _s=0.44 =-28.4cm

[29]:
```
i = 4.9 * 2.54 # cm
 a = 0.20
 ns = 0.44
delta_ = ns - a

# for clay-loam soil
 _clay_loam = 0.44
 _clay_loam = 28.4 # cm # absolute pressure head
Ks_clay_loam = 0.1 # cm/h # saturated hydraulic conductivity
b_clay_loam = 8.52 # parameter that depends upon pore size distribution

F_tp = (delta_ * _clay_loam) / (1. + (i / Ks_clay_loam)) # cm total mass at␣
 ↪ponding
tp = F_tp / i # hour
dt = 1. - tp # 1-hour design storm
print(f'time of ponding: {tp} hr')
print(f'Mass at ponding: {F_tp} cm')

# rewriting green and ampt model eqn in simpler form
# _ + +  * ln(( _( + )+ )/(_ +  ))
# m +   * ln(c)
df_ga_clay_loam = pd.DataFrame(index = np.arange(0, 1.5, .01), columns = ['m',␣
 ↪'c', 'GA', 'delta', 'f_t', 't'])
df_ga_clay_loam.index.name = 'F_tp_dt'

df_ga_clay_loam['m'] = F_tp + (Ks_clay_loam * dt)
df_ga_clay_loam['c'] = (df_ga_clay_loam.index + (delta_ * _clay_loam)) / (F_tp␣
 ↪+ (delta_ * _clay_loam))

df_ga_clay_loam['GA'] = df_ga_clay_loam.m + delta_ * _clay_loam * np.
 ↪log(df_ga_clay_loam.c)
df_ga_clay_loam['delta'] = df_ga_clay_loam.GA - df_ga_clay_loam.index

df_ga_clay_loam['f_t'] = Ks_clay_loam * (1. + ( (delta_ * _clay_loam) /␣
 ↪df_ga_clay_loam.index ))
df_ga_clay_loam['f_t'] = np.where(df_ga_clay_loam.index <= F_tp, i,␣
 ↪df_ga_clay_loam['f_t'])
# cannot apply this as the infiltration rate is changing over time after time␣
 ↪of ponding
```

```
# df_ga_clay_loam['t'] = df_ga_clay_loam.index / df_ga_clay_loam.f_t

# instead use the formula in slide 16
df_ga_clay_loam['t1'] = (df_ga_clay_loam.index - F_tp) / Ks_clay_loam
df_ga_clay_loam['t2'] =  _clay_loam * delta_  / Ks_clay_loam
df_ga_clay_loam['t3'] = (df_ga_clay_loam.index + ( _clay_loam * delta_ )) /␣
 ↪(F_tp + ( _clay_loam * delta_ ))
df_ga_clay_loam['t'] = df_ga_clay_loam['t1'] - df_ga_clay_loam['t2'] * np.
 ↪log(df_ga_clay_loam['t3']) + tp
df_ga_clay_loam = df_ga_clay_loam.drop(columns=['t1', 't2', 't3'], axis=1)
```

```
time of ponding: 0.004365103060621213 hr
Mass at ponding: 0.054328072692491625 cm
```

[30]:
```
df_ga_clay_loam
```

[30]:

F_tp_dt	m	c	GA	delta	f_t	t
0.00	0.153892	0.992092	0.099779	0.099779	12.446000	0.002211
0.01	0.153892	0.993548	0.109772	0.099772	12.446000	0.002285
0.02	0.153892	0.995003	0.119750	0.099750	12.446000	0.002504
0.03	0.153892	0.996459	0.129713	0.099713	12.446000	0.002870
0.04	0.153892	0.997914	0.139662	0.099662	12.446000	0.003381
…	…	…	…	…	…	…
1.45	0.153892	1.203145	1.414435	-0.035565	0.570069	1.355652
1.46	0.153892	1.204600	1.422676	-0.037324	0.566849	1.373243
1.47	0.153892	1.206056	1.430907	-0.039093	0.563673	1.390935
1.48	0.153892	1.207511	1.439128	-0.040872	0.560541	1.408725
1.49	0.153892	1.208967	1.447339	-0.042661	0.557450	1.426614

[150 rows x 6 columns]

[31]:
```
# f(t) vs t
fig = initialize_fig(title = 'Clay Loam',
                     x_axis_label = 'Time (hr)',
                     y_axis_label = 'f(cm/hr) & F(cm)',
#                      tooltips = [
#                          ('time (hr)', '$x'),
#                          ('f(cm/hr)',  '$y'),
#                      ],
                     plot_height = 300,
                    )

fig.title.text_font_size = '15pt'
fig.xaxis.axis_label_text_font_size = '15pt'
fig.yaxis.axis_label_text_font_size = '15pt'
```

```
fig.circle(df_ga_clay_loam.t, df_ga_clay_loam.index, fill_color='red', size=7,␣
 ↪legend_label='F(t)-t plot')
fig.line(df_ga_clay_loam.t, df_ga_clay_loam.index, line_width=2,␣
 ↪line_color='red', legend_label='F(t)-t plot')

fig.circle(df_ga_clay_loam.t, df_ga_clay_loam.f_t, fill_color='blue', size=7,␣
 ↪legend_label='f(t)-t plot')
fig.line(df_ga_clay_loam.t, df_ga_clay_loam.f_t, line_width=2,␣
 ↪line_color='blue', legend_label='f(t)-t plot')

fig.legend.location = 'top_right'
fig.legend.click_policy='hide'
bp.show(fig)
```

[32]:
```
# if intensity of 2 in/hr is considered

i = 2. * 2.54 # cm

F_tp = (delta_  *  _clay_loam) / (1. + (i / Ks_clay_loam)) # total mass at␣
 ↪ponding
tp = F_tp / i
dt = 1. - tp # 1-hour design storm
print(f'time of ponding: {tp} hr')
print(f'Mass at ponding: {F_tp} cm')

# rewriting green and ampt model eqn in simpler form
# _ + +   * ln(( _( + )+ )/(_ +  ))
# m +    * ln(c)
df_ga_clay_loam_2 = pd.DataFrame(index = np.arange(0., 1.5, .01), columns =␣
 ↪['m', 'c', 'GA', 'delta', 'f_t', 't'])
df_ga_clay_loam_2.index.name = 'F_tp_dt'

df_ga_clay_loam_2['m'] = F_tp + (Ks_clay_loam * dt)
df_ga_clay_loam_2['c'] = (df_ga_clay_loam_2.index + (delta_  *  _clay_loam)) /␣
 ↪(F_tp + (delta_  *  _clay_loam))

df_ga_clay_loam_2['GA'] = df_ga_clay_loam_2.m + delta_  *  _clay_loam * np.
 ↪log(df_ga_clay_loam_2.c)
df_ga_clay_loam_2['delta'] = df_ga_clay_loam_2.GA - df_ga_clay_loam_2.index

df_ga_clay_loam_2['f_t'] = Ks_clay_loam * (1. + ( (delta_  *  _clay_loam) /␣
 ↪df_ga_clay_loam_2.index ))
df_ga_clay_loam_2['f_t'] = np.where(df_ga_clay_loam_2.index <= F_tp, i,␣
 ↪df_ga_clay_loam_2['f_t'])

df_ga_clay_loam_2['t1'] = (df_ga_clay_loam_2.index - F_tp) / Ks_clay_loam
```

```
df_ga_clay_loam_2['t2'] =  _clay_loam * delta_  / Ks_clay_loam
df_ga_clay_loam_2['t3'] = (df_ga_clay_loam_2.index + ( _clay_loam * delta_ )) /␣
 ↪(F_tp + ( _clay_loam * delta_ ))
df_ga_clay_loam_2['t'] = df_ga_clay_loam_2['t1'] - df_ga_clay_loam_2['t2'] * np.
 ↪log(df_ga_clay_loam_2['t3']) + tp
df_ga_clay_loam_2 = df_ga_clay_loam_2.drop(columns=['t1', 't2', 't3'], axis=1)
```

```
time of ponding: 0.0259021676344511 hr
Mass at ponding: 0.13158301158301158 cm
```

[33]:
```
df_ga_clay_loam_2
```

[33]:

F_tp_dt	m	c	GA	delta	f_t	t
0.00	0.228993	0.981061	0.098664	0.098664	5.080000	0.013362
0.01	0.228993	0.982500	0.108656	0.098656	5.080000	0.013436
0.02	0.228993	0.983939	0.118634	0.098634	5.080000	0.013655
0.03	0.228993	0.985379	0.128598	0.098598	5.080000	0.014021
0.04	0.228993	0.986818	0.138547	0.098547	5.080000	0.014531
…	…	…	…	…	…	…
1.45	0.228993	1.189766	1.413320	-0.036680	0.570069	1.366803
1.46	0.228993	1.191206	1.421561	-0.038439	0.566849	1.384394
1.47	0.228993	1.192645	1.429791	-0.040209	0.563673	1.402085
1.48	0.228993	1.194084	1.438012	-0.041988	0.560541	1.419876
1.49	0.228993	1.195524	1.446223	-0.043777	0.557450	1.437765

[150 rows x 6 columns]

[34]:
```
# f(t) vs t
fig = initialize_fig(title = 'Clay Loam with variying rainfall intensity',
                     x_axis_label = 'Time (hr)',
                     y_axis_label = 'f(cm/hr) or F(cm)',
#                      tooltips = [
#                          ('time (hr)', '$x'),
#                          ('f(cm/hr)',  '$y'),
#                      ],
                     plot_height = 300,
                    )

fig.title.text_font_size = '15pt'
fig.xaxis.axis_label_text_font_size = '15pt'
fig.yaxis.axis_label_text_font_size = '15pt'

fig.circle(df_ga_clay_loam_2.t, df_ga_clay_loam_2.index, fill_color='red',␣
 ↪size=7, legend_label='F(t)-t plot')
fig.line(df_ga_clay_loam_2.t, df_ga_clay_loam_2.index, line_width=2,␣
 ↪line_color='red', legend_label='F(t)-t plot')
```

```
fig.circle(df_ga_clay_loam_2.t, df_ga_clay_loam_2.f_t, fill_color='blue',␣
 ↪size=7, legend_label='f(t)-t plot')
fig.line(df_ga_clay_loam_2.t, df_ga_clay_loam_2.f_t, line_width=2,␣
 ↪line_color='blue', legend_label='f(t)-t plot')


fig.legend.location = 'top_right'
fig.legend.click_policy='hide'
bp.show(fig)
```

[35]:
```
# With both together

# f(t) vs t
fig = initialize_fig(title = 'Clay Loam Case 1 and 2',
                     x_axis_label = 'Time (hr)',
                     y_axis_label = 'f(cm/hr) & F(cm)',
#                      tooltips = [
#                          ('time (hr)', '$x'),
#                          ('f(cm/hr)',  '$y'),
#                      ],
                     plot_height = 300,
                    )

fig.title.text_font_size = '15pt'
fig.xaxis.axis_label_text_font_size = '15pt'
fig.yaxis.axis_label_text_font_size = '15pt'

fig.circle(df_ga_clay_loam.t, df_ga_clay_loam.index, fill_color='red', size=7,␣
 ↪legend_label='F(t)-t plot')
fig.line(df_ga_clay_loam.t, df_ga_clay_loam.index, line_width=2,␣
 ↪line_color='red', legend_label='F(t)-t plot')

fig.circle(df_ga_clay_loam.t, df_ga_clay_loam.f_t, fill_color='blue', size=7,␣
 ↪legend_label='f(t)-t plot')
fig.line(df_ga_clay_loam.t, df_ga_clay_loam.f_t, line_width=2,␣
 ↪line_color='blue', legend_label='f(t)-t plot')

fig.circle(df_ga_clay_loam_2.t, df_ga_clay_loam_2.index, fill_color='green',␣
 ↪size=7, legend_label='F(t)-t plot 2')
fig.line(df_ga_clay_loam_2.t, df_ga_clay_loam_2.index, line_width=2,␣
 ↪line_color='green', legend_label='F(t)-t plot 2')

fig.circle(df_ga_clay_loam_2.t, df_ga_clay_loam_2.f_t, fill_color='fuchsia',␣
 ↪size=7, legend_label='f(t)-t plot 2')
```

```
fig.line(df_ga_clay_loam_2.t, df_ga_clay_loam_2.f_t, line_width=2,␣
 ↪line_color='fuchsia', legend_label='f(t)-t plot 2')

fig.legend.location = 'top_right'
fig.legend.click_policy='hide'
bp.show(fig)
```

As can be seen, the time of ponding increases when the rainfall intensity decreased. As a result, the mass at ponding also increased. As a result, the initial infiltration rate until ponding which is the rainfall intensity is greater for first case.

If we replace the soil with the sandy loam soil compared to the clay loam, then since the porosity of the sandy loam is smaller than the clay loam, the tension or the pressure head increases. As a result, the sandy loam asymptote at higher infiltration rate than the clay loam. Similarly, the tension for the sandy loam is greater than the clay loam.

[ ]:

[ ]:

[ ]:

Question 5

5. Two rivers located 1000m apart from a fully penetrating an aquifer (not confined). The aquifer has a K value of 0. 5m/d. The region receives an average rainfall of 15 cm/yr and evapotranspiration loss is about 10 cm/yr. Assume that the water elevation in River 1 is 20m and the water elevation in River 2 is 18m.

a. Use the Dupuit equation with recharge, and determine the the location and height of the water divide.
b. What is the daily discharge per unit width into River 1 River 2?

[36]:
```
from sympy import symbols, Eq, solve

L = 1000 # length in meter
K = 0.5 # m/d hydraulic conductivity
P = 15. * (1 / 100) * (1 / 365.) # m/d
ET = 10. * (1 / 100.) * (1 / 365.) # m/d
W = P - ET # m/d
h0 = 20 # meter
hL = 18 # meter

q = 0 # for divide

# using dupuit eqn
location = symbols('location')
dupuit_eq = Eq( (K / (2 * L) * (h0**2 - hL**2)) + (W * (location - (L / 2))), q)
```

```
result = solve((dupuit_eq,), (location))
location, = result.values()
print(f'The water divide is {round(location, 3)} m from River 1')

depth = (h0**2 - ((location / L) * (h0**2 - hL**2)) + (W / K * (L -␣
↪location)))**(1/2)
print(f'The water divide has a depth of {round(depth, 3)} m')


# discharge in River 1
q1 = (K / (2 * L) * (h0**2 - hL**2)) + W * (0 - (L / 2))
print(f'discharge in river 1: {round(q1, 3)} m/d')

q2 = (K / (2 * L) * (h0**2 - hL**2)) + W * (L - (L / 2))
print(f'discharge in river 2: {round(q2, 3)} m/d')
```

```
The water divide is 361.300 m from River 1
The water divide has a depth of 19.306 m
discharge in river 1: -0.049 m/d
discharge in river 2: 0.087 m/d
```

[ ]: