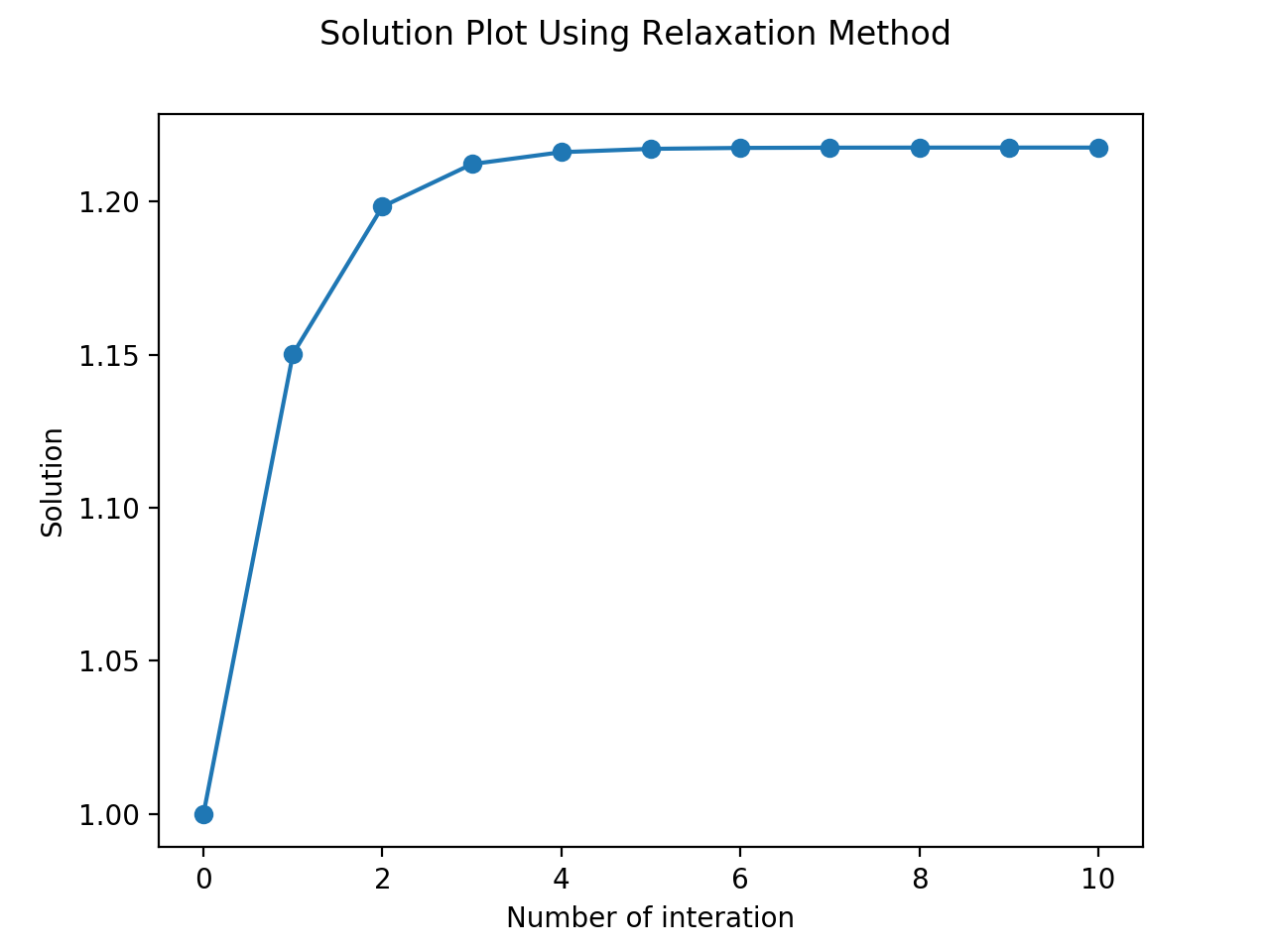
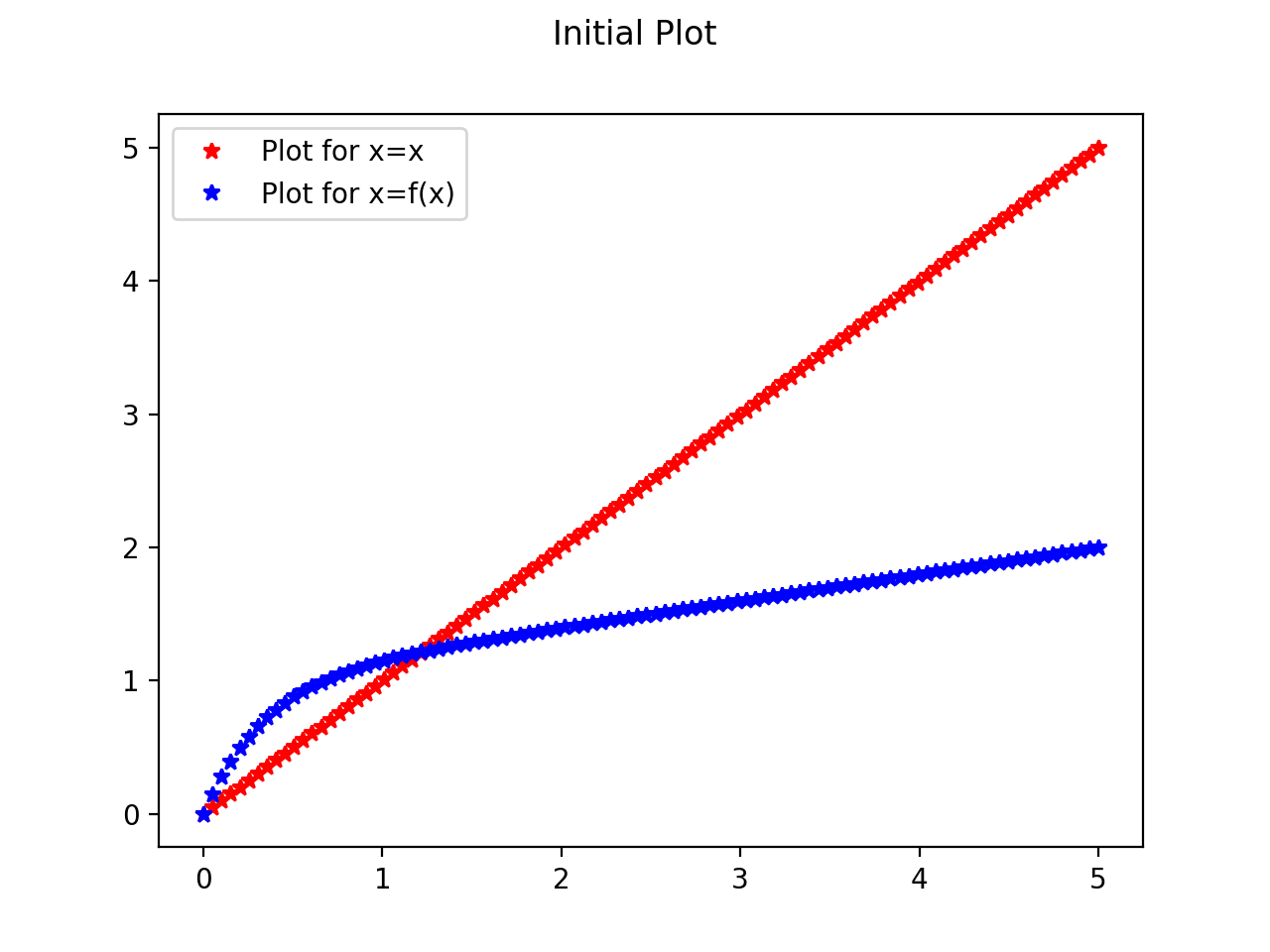
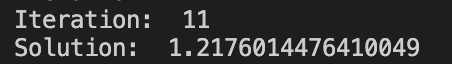
1. **Relaxation**

**Part a)**





The simple method using the definition of the relaxation method converged to a solution in 11 tries with an accuracy of 1e-6.

import numpy as np

import matplotlib.pyplot as plt

# Value of c

c = 3

# Function

def f(x):

y = 1-np.exp(-c\*x)+(0.2\*x)

return y

# Plot the graph

fig = plt.figure()

fig.suptitle('Initial Plot')

x = np.linspace(0,5,100)

plt.plot(x,x,'r\*',label = 'Plot for x=x')

plt.plot(x,f(x),'b\*',label = 'Plot for x=f(x)')

plt.legend()

plt.show()

def relaxation():

accuracy=10\*\*(-6)

Max\_Interation=10000

x0=1

All\_x=[]

All\_x.append(x0)

Have\_Solution = False

for i in range(Max\_Interation):

x1=f(x0)

if abs(x0-x1)<accuracy:

Have\_Solution = True

# i+1 because i starts at 0. Just a matter of definition of the iteration.

print('Iteration: ', i+1)

break

else:

x0=x1

All\_x.append(x0)

if Have\_Solution:

fig2 = plt.figure()

fig2.suptitle('Solution Plot Using Relaxation Method')

plt.plot(All\_x,'o-')

plt.xlabel("Number of interation")

plt.ylabel("Solution")

plt.show()

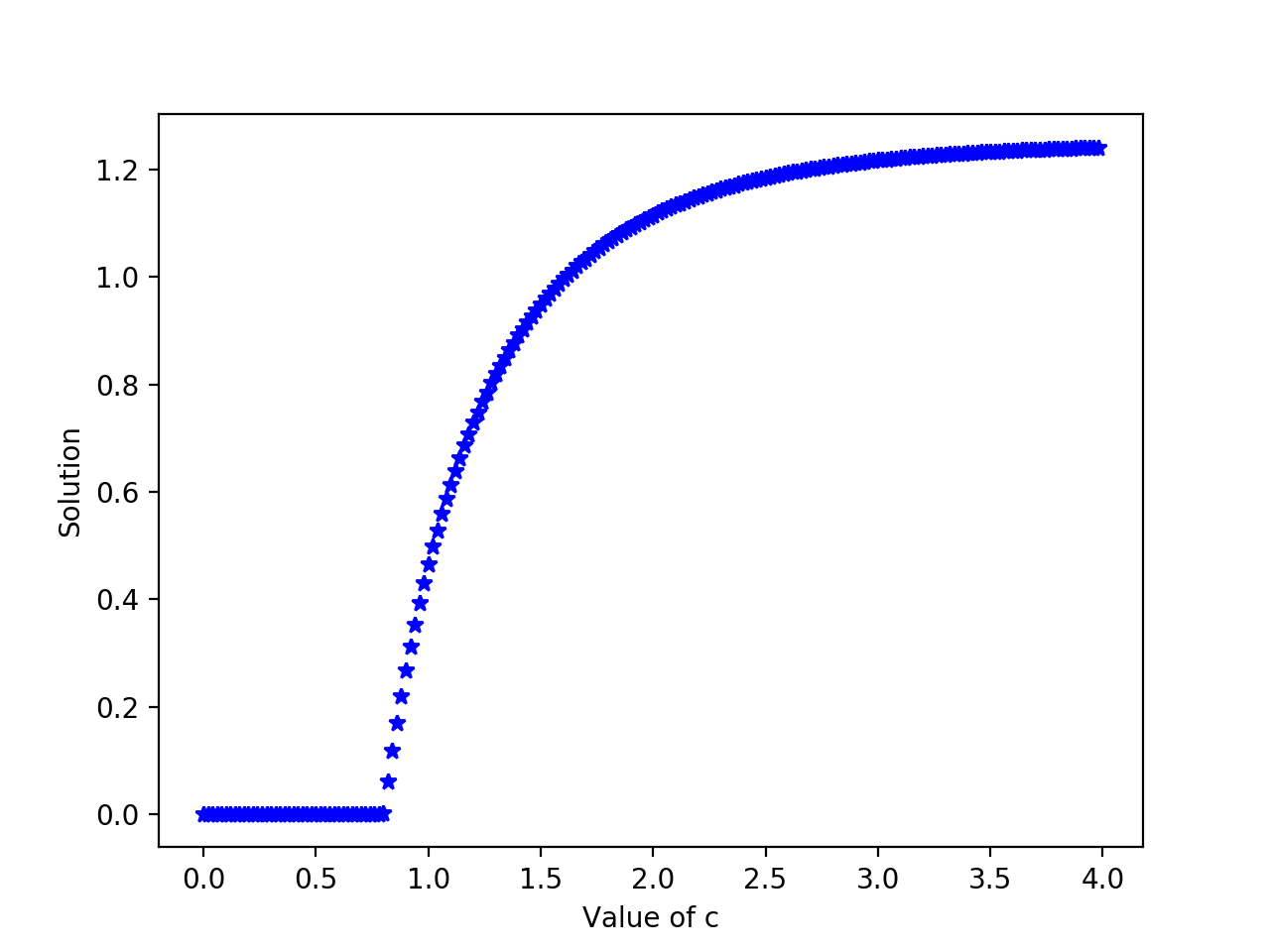
else:

print("We did not find a solution!")

return All\_x[-1]

print('Solution: ',relaxation())

**Part b)**



The code had been adjusted to calculate the solution using a range of values c. The transition from a region of all-zeros to a region of non-zero x is clearly seen on the plot.

import numpy as np

import matplotlib.pyplot as plt

# Value of c

c\_list = np.arange(0,4,0.02)

# Function

def f(x,c):

y = 1-np.exp(-c\*x)+(0.2\*x)

return y

# Plot the graph

# fig = plt.figure()

# fig.suptitle('Initial Plot')

# x = np.linspace(0,5,100)

# plt.plot(x,x,'r\*',label = 'Plot for x=x')

# plt.plot(x,f(x),'b\*',label = 'Plot for x=f(x)')

# plt.legend()

# plt.show()

def relaxation(c):

accuracy=10\*\*(-6)

Max\_Interation=10000

x0=1

All\_x=[]

All\_x.append(x0)

Have\_Solution = False

for i in range(Max\_Interation):

x1=f(x0,c)

if abs(x0-x1)<accuracy:

Have\_Solution = True

break

else:

x0=x1

All\_x.append(x0)

return x0

# if Have\_Solution:

# fig2 = plt.figure()

# fig2.suptitle('Solution Plot Using Relaxation Method')

# plt.plot(All\_x,'o-')

# plt.xlabel("Number of interation")

# plt.ylabel("Solution")

# plt.show()

# else:

# print("We did not find a solution!")

# relaxation()

x = []

for i in range(len(c\_list)):

x.append(relaxation(c\_list[i]))

plt.figure()

plt.plot(c\_list,x,'b\*')

plt.xlabel("Value of c")

plt.ylabel("Solution")

plt.show()

print('Solution: ', x[-1])

1. **Over-Relaxation**

**Part a)**

, ,

**Part b)**

The code had been modified to show the number of iterations to converge with an accuracy of 1e-6

****

import numpy as np

import matplotlib.pyplot as plt

# Value of c

c = 3

# Function

def f(x):

y = 1-np.exp(-c\*x)+(0.2\*x)

return y

# Plot the graph

# fig = plt.figure()

# fig.suptitle('Initial Plot')

# x = np.linspace(0,5,100)

# plt.plot(x,x,'r\*',label = 'Plot for x=x')

# plt.plot(x,f(x),'b\*',label = 'Plot for x=f(x)')

# plt.legend()

# plt.show()

def relaxation():

accuracy=10\*\*(-6)

Max\_Interation=10000

x0=1

All\_x=[]

All\_x.append(x0)

Have\_Solution = False

for i in range(Max\_Interation):

x1=f(x0)

if abs(x0-x1)<accuracy:

Have\_Solution = True

print('Number of Iteration: ', i, ', to Converge With Accuracy: ', accuracy)

break

else:

x0=x1

All\_x.append(x0)

# if Have\_Solution:

# fig2 = plt.figure()

# fig2.suptitle('Solution Plot Using Relaxation Method')

# plt.plot(All\_x,'o-')

# plt.xlabel("Number of interation")

# plt.ylabel("Solution")

# plt.show()

# else:

# print("We did not find a solution!")

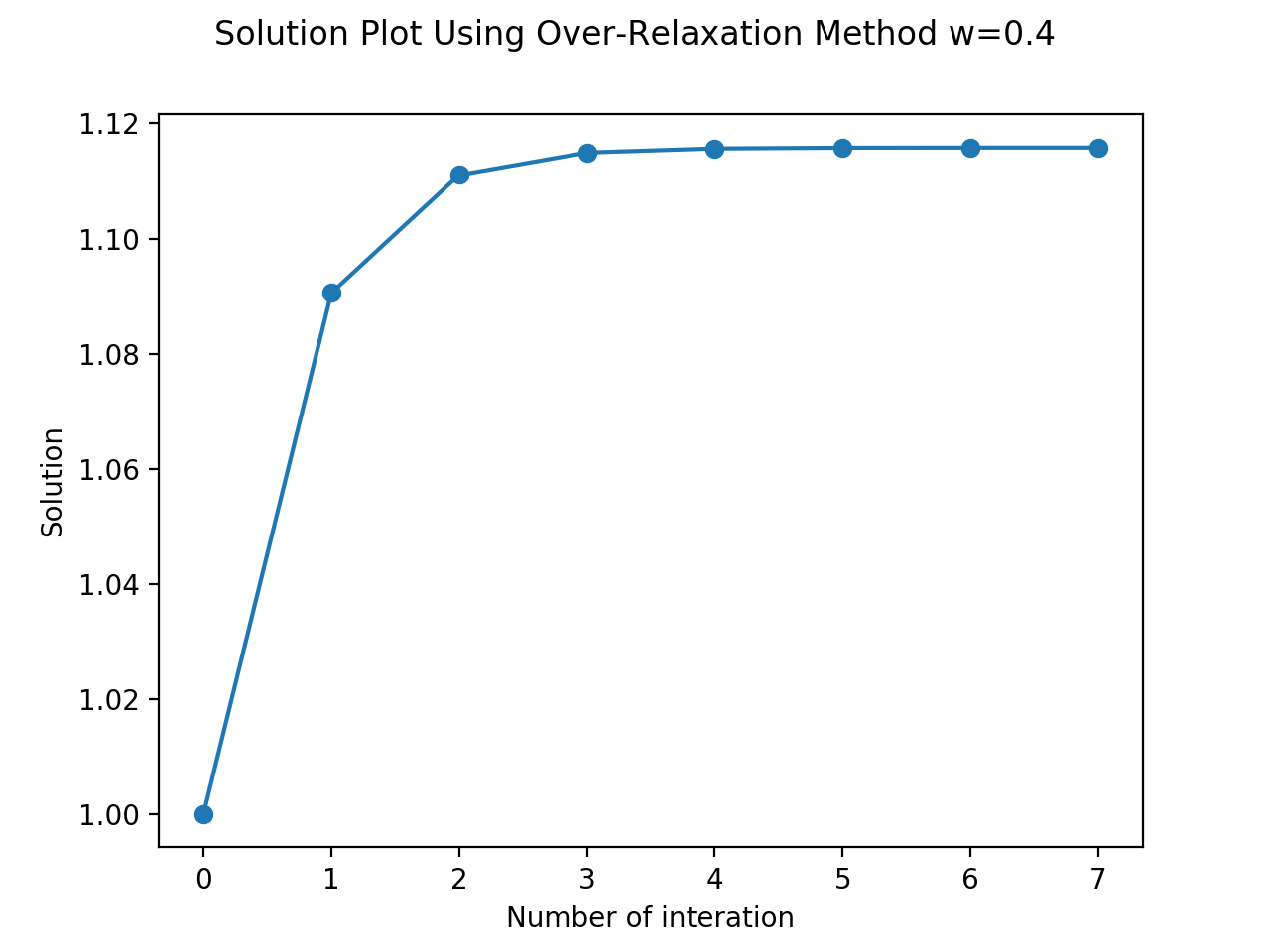
relaxation()

**Part c)**

At w=0.7 gave the fastest convergence with only 4 iterations.

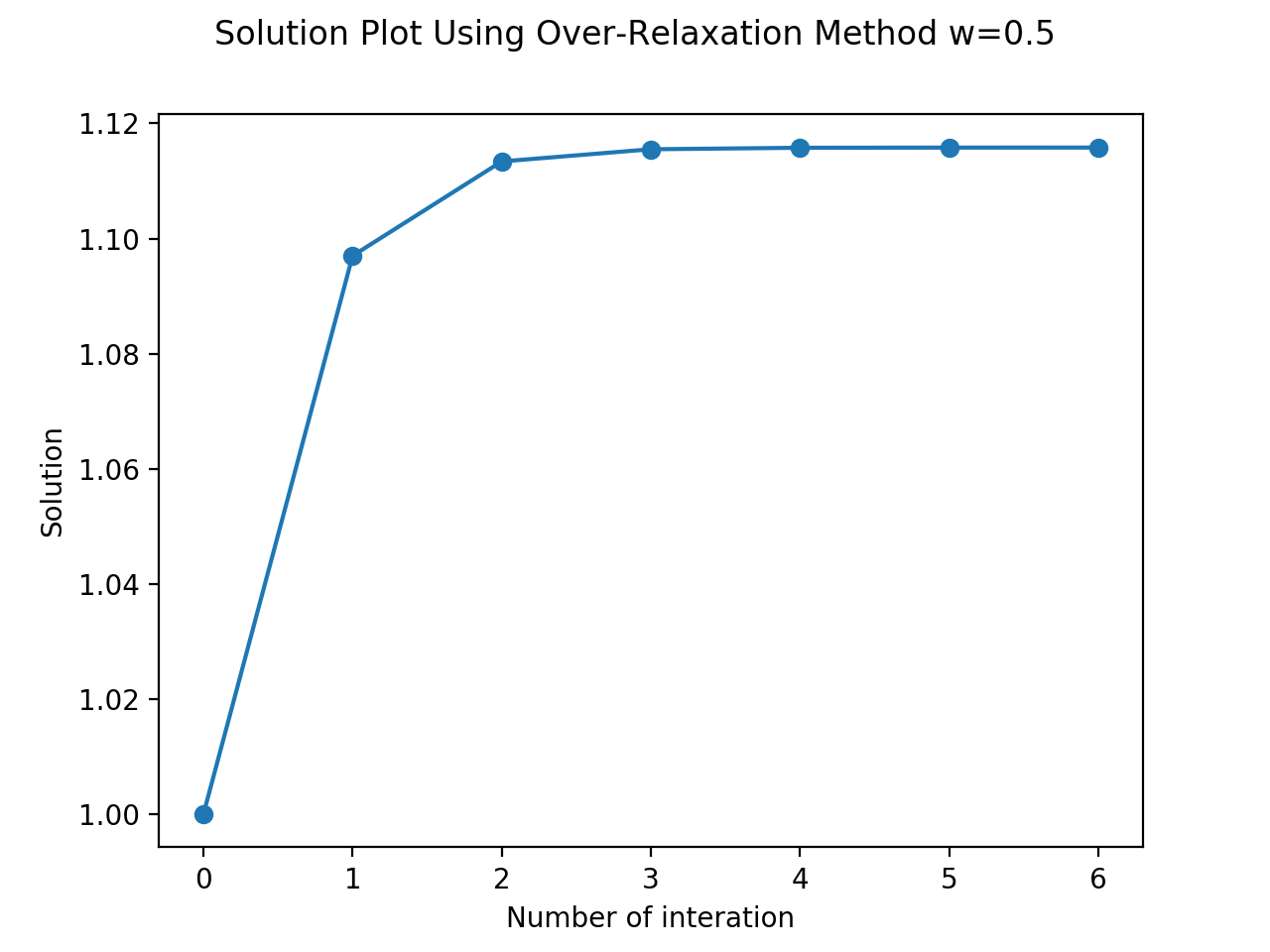
Using w=0.4





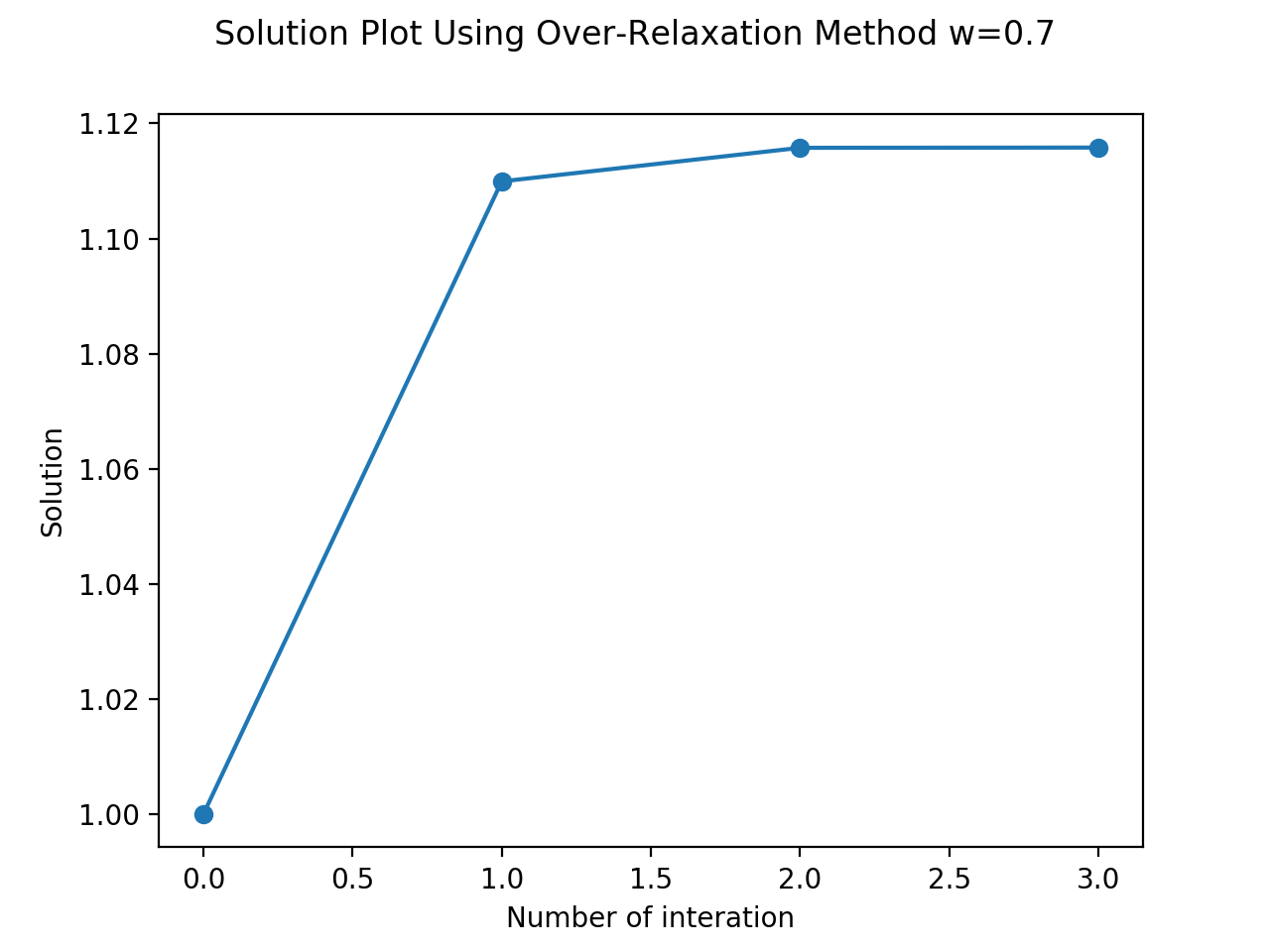
Using w=0.5





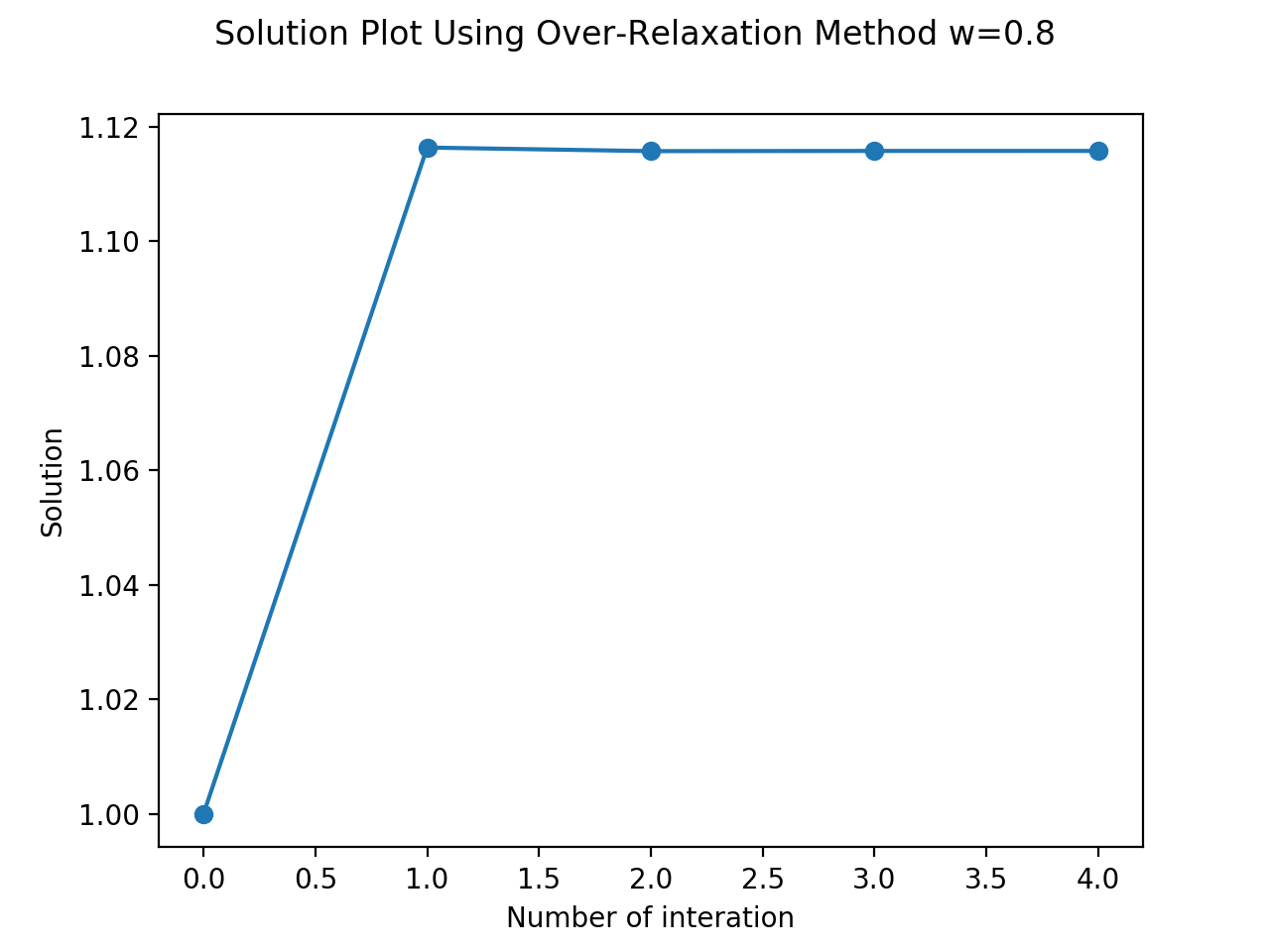
Using w=0.7





Using w=0.8





import numpy as np

import matplotlib.pyplot as plt

# Value of c

c = 2

# Function

def f(x):

y = 1-np.exp(-c\*x)+(0.2\*x)

return y

# def overrelaxation():

# accuracy = 10\*\*(-6)

# Max\_Interation=10000

# x0=1

# All\_x=[]

# All\_x.append(x0)

# Have\_Solution = False

# w=0.5

# for i in range(Max\_Interation):

# x1= (1 + w) \* f(All\_x[-1]) - w \* All\_x[-1]

# if abs(x0-x1)<accuracy:

# Have\_Solution = True

# print('Iteration: ', i+1)

# break

# else:

# x0=x1

# All\_x.append(x0)

# return x0

# overrelaxation()

def overrelax():

accuracy = 10\*\*(-6)

init = 1

x = [init]

# tested with different values of w, but -1 gave the smallest number = 2

# w = 0.3, iteration = 10

# w = 0.4, iteration = 8

# w = 0.5, iteration = 7

# w = 0.6, iteration = 6

# w = 0.7, iteration = 4

# w = 0.8, iteration = 5

# w = 0.9, iteration = 7

# at w = -1, iteration = 2 only. But the solution is wrong.

# any value more negative than -1 gives unstable and incorrect results.

w = (0.8)

x\_ = (1 + w) \* f(x[-1]) - w \* x[-1]

x.append(x\_)

if abs(x[-1] - x[-2]) == 0:

e = 0

elif ((f(x[-1]) - f(x[-2])) / (x[-1] - x[-2])) == 1:

e = abs(x[-1] - x[-2])

else:

f\_ = (f(x[-1]) - f(x[-2])) / (x[-1] - x[-2])

e = (x[-1] - x[-2]) / (1 - 1/((1 + w) \* f\_ - w))

while abs(e) > accuracy:

x\_ = (1 + w) \* f(x[-1]) - w \* x[-1]

x.append(x\_)

if abs(x[-1] - x[-2]) == 0:

e = 0

else:

f\_ = (f(x[-1]) - f(x[-2])) / (x[-1] - x[-2])

e = (x[-1] - x[-2]) / (1 - 1 / ((1 + w) \* f\_ - w))

return x

fig = plt.figure()

x\_over = overrelax()

print('Solution: ', x\_over[-1])

print('Number of Iteration: ', (len(x\_over)))

fig.suptitle('Solution Plot Using Over-Relaxation Method w=0.8')

plt.plot(x\_over,'o-')

plt.xlabel("Number of interation")

plt.ylabel("Solution")

plt.show()

**Part d)**

Using (-1) for the question gave the fastest number of iterations (2).

This may be because the equation in the over-relaxation method

x\_ = (1 + w) \* f(x[-1]) - w \* x[-1]

at w=(-1), naturally becomes

x\_ = x[-1]

this means that the solution with converge within two tries because the same value will be used. But this is incorrect and does not lead to a solution and is hence not very useful, but should be minded to avoid mistakes.

Furthermore, for values w<0 to find a solution faster than the ordinary relaxation method, it would mean the solution would converge faster, which means the error will reach the accuracy faster than the error for the ordinary relaxation method. Then, using the definition of the error for the over relaxation method, the error will converge faster when the initial difference of x and x’ is large with a carefully chosen w<0 value. For example, when w = -1, the error becomes:

(x-x’)/2

And if (x-x’) is very large, choosing w=-1 would converge the value faster.