Quinten Crum

Homework #2

Math 450

Fall 2022

Problem #7

A)

In [ ]:
```
import numpy as np
import math
import matplotlib.pyplot as plt
```

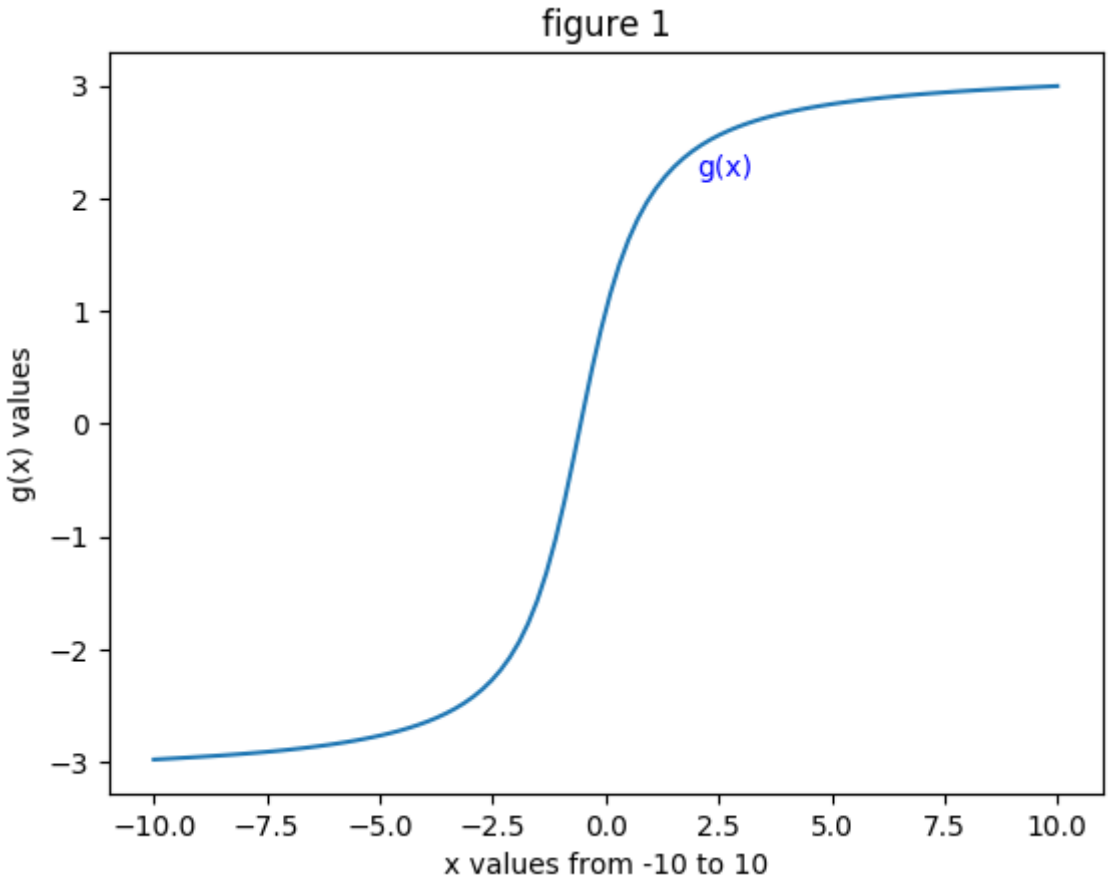
In [ ]:
```
def ai_func(ai, x, i):
    if(i == 0):
        return ai
    else:
        ai_1 = ai + ((x*math.cos(ai)+x)/10.0)
        return ai_func(ai_1,x,i-1)

def g_x_func(x):
    return ai_func(1,x,10) #a0 = 1, x=x, i=10
```

In [ ]:
```
x_vec = np.linspace(-10,10,100)
y_vec = np.zeros(len(x_vec))

for i in range(len(x_vec)):
    y_vec[i] = g_x_func(x_vec[i])


plt.plot(x_vec,y_vec)
plt.xlabel('x values from -10 to 10')
plt.ylabel('g(x) values')
plt.title('figure 1')
plt.text(2,2.2,'g(x)',color='blue')
plt.show()
```



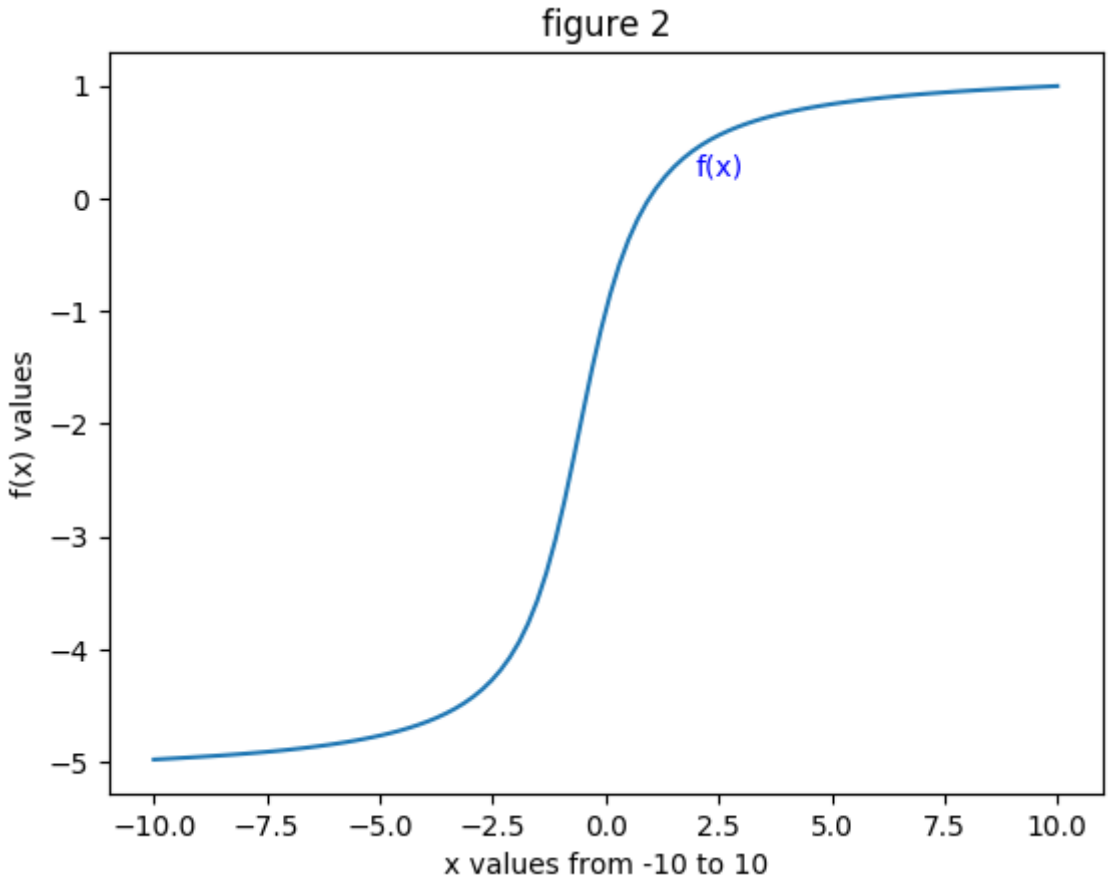
B)

In [ ]:

```
x_vec = np.linspace(-10,10,100)
y_vec = np.zeros(len(x_vec))

for i in range(len(x_vec)):
    y_vec[i] = g_x_func(x_vec[i]) - 2


plt.plot(x_vec,y_vec)
plt.xlabel('x values from -10 to 10')
plt.ylabel('f(x) values')
plt.title('figure 2')
plt.text(2,0.2,'f(x)',color='blue')
plt.show()
```



Newton's method may be ill suited for this new task of finding where f(x)=0 because Newton's method requires us to have a starting point within a ceratin neighborhood. As we can see in figure 2 the slopes around positive and negative 7.5-10.0 if we were do use these areas as our starting points we would actually get a value further away from the root of f.

C)

In [ ]:
```
def f_x_func(x):
    return g_x_func(x)-2

def secantMethod(x_vec,f_vec,i,digits):
    if (i >= len(x_vec)) or (abs(x_vec[i-1]-x_vec[i-2])< 10**((-1)*digits)): #ch
        #print('returning i: '+str(i))
        #print('Returning because of i: '+str(i >= len(x_vec)))
        return x_vec,f_vec,i,digits
    else:
        x_vec[i] = x_vec[i-1] - (((x_vec[i-1] - x_vec[i-2])/(f_vec[i-1] - f_vec[
        f_vec[i] = f_x_func(x_vec[i])
        return secantMethod(x_vec,f_vec,i+1,digits)
```

In [ ]:
```
iterations = 10 #number of iterations of x
x_vec = np.zeros(iterations,dtype=np.float128)
x_vec[1] = 1    #x0 already zero and this sets x1 to 1
f_vec = np.zeros(iterations,dtype=np.float128)
f_vec[0] = f_x_func(x_vec[0])   #f(x0)
f_vec[1] = f_x_func(x_vec[1])   #f(x1)

digits_9 = secantMethod(x_vec,f_vec,2,9)
digits_6 = secantMethod(x_vec,f_vec,2,6)
```

```
print('For 9 digits of accuracy it took '+str(digits_9[2])+' iterations of the s

print('For 6 digits of accuracy it took '+str(digits_6[2])+' iterations of the s

print()
print('These two values are the same even when we use the 9 digit tolerance.')
```

```
For 9 digits of accuracy it took 7 iterations of the secant method and we ended
up getting an x value of 0.96366361701790921005
For 6 digits of accuracy it took 7 iterations of the secant method and we ended
up getting an x value of 0.96366361701790921005

These two values are the same even when we use the 9 digit tolerance.
```

D)

In [ ]:
```
def bisectionMethod(a_vec,b_vec,i):
    if(f_x_func(a_vec[i-1])*f_x_func(b_vec[i-1]) > 0):
        print('Error in a,b values!')
        return a_vec,b_vec,i
    elif i >= len(a_vec):
        print('i too high for vector index')
        return a_vec,b_vec,i
    elif abs(a_vec[i-1]-b_vec[i-1]) < 10**((-1)*6):
        print('Accuracy desired hit!')
        return a_vec,b_vec,i-1
    else:
        c_i = (a_vec[i-1]+b_vec[i-1])/2.0
        f_c = f_x_func(c_i)
        f_a = f_x_func(a_vec[i-1])
        f_b = f_x_func(b_vec[i-1])

        if f_a*f_c < 0:
            a_vec[i] = a_vec[i-1]
            b_vec[i] = c_i
            return bisectionMethod(a_vec,b_vec,i+1)
        elif f_b*f_c < 0:
            b_vec[i] = b_vec[i-1]
            a_vec[i] = c_i
            return bisectionMethod(a_vec,b_vec,i+1)
        else:
            print('both inequalities are false')
            return a_vec,b_vec,i
```

In [ ]:
```
iterations = 100

a_vec = np.zeros(iterations)
b_vec = np.zeros(iterations)
a_vec[0] = 0
b_vec[0] = 1



ans = bisectionMethod(a_vec,b_vec,1)

print('For 6 digits of accuracy it took '+str(ans[2])+' iterations of the secant
```

```
print()
print()
print('Comparing the two answers:')
print()
print('The value we got from the secant method was:    '+str(digits_6[0][digits_
print('The value we got from the bisection method was: '+str(ans[0][ans[2]-1]))
print('The difference between the two is: '+str(abs(digits_6[0][digits_6[2]-1])
print()
print('The secant method took '+str(digits_6[2])+' iterations to get to the desi
print('The bisection method took '+str(ans[2])+' iterations to get the the desir
print('The secant method we expect to be faster than the bisection method as we
```

```
Accuracy desired hit!
For 6 digits of accuracy it took 20 iterations of the secant method and we ended
up getting an x value of 0.9636631011962891


Comparing the two answers:

The value we got from the secant method was:    0.96366361701790921005
The value we got from the bisection method was: 0.9636631011962891
The difference between the two is: 5.158216201475458035e-07 meaning that the two
answers are similar to the 6-digit accuracy we were looking for.

The secant method took 7 iterations to get to the desired level of accuracy.
The bisection method took 20 iterations to get the the desired level of accurac
y.
The secant method we expect to be faster than the bisection method as we can se
e. This is because as we saw in class the bisection method is only converges lin
early while the secant method converges super linearly.
```

Problem #8

A) $g(x) = 2x - ax^2 = x \iff f(x) = 0$

$\underbrace{x - ax^2}_{f(x)} = 0 \iff f(x) = 0$

Roots of $f(x)$: $x = \frac{-1 \pm \sqrt{1 - 4(-a)\cdot 0}}{2(-a)} = \frac{-1 \pm 1}{-2a}$

If plus 1 → $= \frac{-1+1}{-2a} = 0$ If minus 1 → $\frac{-1-1}{-2a} = \frac{1}{a}$

Therefore the limits of this fixed point iteration should be 0 or $\frac{1}{a}$.

B) $|g'(x)| < 1 \Rightarrow |2 - a2x| < 1 \Rightarrow 2 - a2x < 1$ or $2 - a2x > -1$

$-a2x < -1$ $\quad -a2x > -3$

$ax > \frac{1}{2}$ $\quad ax < \frac{3}{2}$

$x > \frac{1}{2}\cdot\frac{1}{a}$ $\quad x < \frac{3}{2}\cdot\frac{1}{a}$

g is continuous for all points therefore the interval that contains $\frac{1}{a}$ in which this converges to $\frac{1}{a}$ is:

$\frac{1}{2}\cdot\frac{1}{a} < x < \frac{3}{2}\cdot\frac{1}{a}$

Project / Projekt:

Participants / Teilnehmer:

No. / Nr.	Description / Beschreibung

Problem #9

$$\begin{array}{cccc|c} 1 & 1/2 & 1/3 & 1/4 & 1 \\ 1/2 & 1/3 & 1/4 & 1/5 & 2 \\ 1/3 & 1/4 & 1/5 & 1/6 & 3 \\ 1/4 & 1/5 & 1/6 & 1/7 & 4 \end{array}$$

r_1 r_2 r_3 r_4

① $r_1 \cdot (-1/2)$ add to r_2

$r_1 \cdot (-1/3)$ add to r_3

$r_2 \cdot (-1/4)$ add to r_4

$$\begin{array}{cccc|c} 1 & 1/2 & 1/3 & 1/4 & 1 \\ 0 & 1/12 & 1/12 & 3/40 & 3/2 \\ 0 & 1/12 & 4/45 & 1/12 & 8/3 \\ 0 & 3/40 & 1/12 & 9/112 & 15/4 \end{array}$$

② $r_2 \cdot (12)$

$$\begin{array}{cccc|c} 1 & 1/2 & 1/3 & 1/4 & 1 \\ 0 & 1 & 1 & 9/10 & 18 \\ 0 & 1/12 & 4/45 & 1/12 & 8/3 \\ 0 & 3/40 & 1/12 & 9/112 & 15/4 \end{array}$$

Project / Projekt:

Participants / Teilnehmer:

No. / Nr.	Description / Beschreibung

$r_2\ (-1/2)$ add to r_1

$r_2\ (-1/12)$ add to r_3

$r_2\ (-3/40)$ add to r_4

$$\left[\begin{array}{cccc|c} 1 & 0 & -1/6 & -1/5 & -8 \\ 0 & 1 & 1 & 9/10 & 18 \\ 0 & 0 & 1/180 & 1/120 & 7/6 \\ 0 & 0 & 1/120 & 9/700 & 12/5 \end{array}\right]$$

$r_3\ (180)$

$$\left[\begin{array}{cccc|c} 1 & 0 & -1/6 & -1/5 & -8 \\ 0 & 1 & 1 & 9/10 & 18 \\ 0 & 0 & 1 & 3/2 & 210 \\ 0 & 0 & 1/20 & 9/700 & 12/5 \end{array}\right]$$

$r_3\ (1/6)$ add to r_1

$r_3\ (-1)$ add to r_2

$r_3\ (-1/20)$ add to r_4

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 1/20 & 27 \\ 0 & 1 & 0 & -3/5 & -192 \\ 0 & 0 & 1 & 3/2 & 210 \\ 0 & 0 & 0 & 1/2600 & 13/20 \end{array}\right]$$

No. Description | Type

Nr. Beschreibung | Typ

r_4 (2800)

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 1/20 & 27 \\ 0 & 1 & 0 & -3/5 & -192 \\ 0 & 0 & 1 & 3/2 & 210 \\ 0 & 0 & 0 & 1 & 1820 \end{array}\right]$$

r_4 $(-1/20)$ add to r_1

r_4 $(3/5)$ add to r_2

r_4 $(-3/2)$ add to r_3

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & -64 \\ 0 & 1 & 0 & 0 & 900 \\ 0 & 0 & 1 & 0 & -2520 \\ 0 & 0 & 0 & 1 & 1820 \end{array}\right]$$

$x_1 = -64$

$x_2 = 900$

$x_3 = -2520$

$x_4 = 1820$

Problem 10

In [ ]:
```
import numpy as np
import math
import matplotlib.pyplot as plt
```

In [ ]:
```
def hilbert(n):
    x = np.arange(1, n+1) + np.arange(0, n)[:, np.newaxis]
    return 1.0/x

def swapRow(A,r1,r2):
    # put row r2 in r1 spot
    print('Swapping rows')
    A_new = np.copy(A)
    for i in range(len(A_new)):
        temp_var = A_new[r1][i]
        A_new[r1][i] = A_new[r2][i]
        A_new[r2][i] = temp_var

    return A_new




def gaussianElimination(A,b): #nxn matrices only
    A_temp = np.copy(A)
    A_inv = np.identity(len(A))
    b_temp = np.copy(b)

    #making zeros below diagnal
    for c in range(len(A)): #columns
        if(A_temp[c][c] == 0): # zero along the diagnal
                for i in range(c,len(A[0])):
                    if(A_temp[i][c] != 0):
                        A_temp = np.copy(swapRow(A_temp,c,i))
                        A_inv = np.copy(swapRow(A_inv,c,i))
        for r in range(c+1,len(A[0])): #rows
            #print('r: '+str(r)+'        c: '+str(c)+'        d: '+str(c)+','+st
            if(A_temp[r][c] != 0):
                temp = A_temp[r][c]*(-1.0)/A_temp[c][c] # value @ index trying t
                b_temp[r][0] = b_temp[r][0] + temp*b_temp[c][0]
                for i in range(c,len(A)):
                    A_temp[r][i] = A_temp[r][i] + temp*A_temp[c][i]
                for i in range(len(A)):
                    A_inv[r][i] = A_inv[r][i] + temp*A_inv[c][i]


    #print('Above time')
    #print()
    #making zeros above diagnal
    for c in range(1,len(A)): #columns
        if(A_temp[c][c] == 0): # zero along the diagnal
                for i in range(c,len(A[0])):
                    if(A_temp[i][c] != 0):
                        A_temp = np.copy(swapRow(A_temp,c,i))
                        A_inv = np.copy(swapRow(A_inv,c,i))
```

```
        for r in reversed(range(c)): #rows
            #print('r: '+str(r)+'        c: '+str(c)+'        d: '+str(c)+','+st
            if(A_temp[r][c] != 0):
                temp = A_temp[r][c]*(-1.0)/A_temp[c][c] # value @ index trying t
                b_temp[r][0] = b_temp[r][0] + temp*b_temp[c][0]
                for i in range(c,len(A)):
                    A_temp[r][i] = A_temp[r][i] + temp*A_temp[c][i]
                for i in range(len(A)):
                    A_inv[r][i] = A_inv[r][i] + temp*A_inv[c][i]


    #print('A: '+str(A_temp))
    #print('A_inv: '+str(A_inv))
    #print('b: '+str(b_temp))

    #print('Make diagnals 1')
    #print()
    #making diagnals 1
    for i in range(len(A)):
        temp = 1.0/A_temp[i][i]
        b_temp[i][0] *= temp
        A_temp[i][i] *= temp
        for j in range(len(A)):
            A_inv[i][j] *= temp




    #print('The solution is correct if: '+str(np.matmul(A_inv,b) == b_temp))
    #print('AA^-1 = I -->'+str(np.matmul(A,A_inv) == np.identity(len(A))))

    print('The solution is correct if A^-1b == b_')


    return A_inv,b_temp,A_temp
```

In [ ]:

```
m = 4
a = hilbert(m)
#b = np.array([[0.564348], [0.93798], [0.707275],[0.0248771]])
b = np.array([[1], [2], [3],[4]])

t = gaussianElimination(a,b)
print('My programs inverse to problem #9 was:')
print(t[0])
print('The solution to the system of linear equations in problem #9 is:')
print(t[1])
print()
```

```
AA^-1 = I -->[[False  True False False]
 [False False False False]
 [False False False False]
 [False False False False]]
My programs inverse to problem #9 was:
[[   16.  -120.   240.  -140.]
 [ -120.  1200. -2700.  1680.]
 [  240. -2700.  6480. -4200.]
 [ -140.  1680. -4200.  2800.]]
The solution to the system of linear equations in problem #9 is:
[[ -284]
 [ 3360]
```

```
[-8280]
[ 5599]]
```

In [ ]:
```
# 10x10 matrix
m = 10
a = hilbert(m)
b = np.array([[1]]*m)
for i in range(len(b)):
    b[i][0]=i+1


t = gaussianElimination(a,b)
print('If the following is an identity matrix then the inverse was found correct
print(np.matmul(t[2],t[0]))
```

```
If the following is an identity matrix then the inverse was found correctly:
[[ 9.99963466e+01 -4.94968266e+03  7.91932148e+04 -6.00538137e+05
   2.52222426e+06 -6.30548555e+06  9.60826178e+06 -8.75030521e+06
   4.37511957e+06 -9.23630235e+05]
 [-4.94968280e+03  3.26672459e+05 -5.88001130e+06  4.75621538e+07
  -2.08082251e+08  5.35063976e+08 -8.32315574e+08  7.70028551e+08
  -3.89824797e+08  8.31272187e+07]
 [ 7.91932198e+04 -5.88001146e+06  1.12894943e+08 -9.51235774e+08
   4.28052900e+09 -1.12363184e+10  1.77560632e+10 -1.66326446e+10
   8.50529589e+09 -1.82880726e+09]
 [-6.00538194e+05  4.75621563e+07 -9.51235795e+08  8.24399246e+09
  -3.78706474e+10  1.00987962e+11 -1.61580155e+11  1.52886126e+11
  -7.88316955e+10  1.70689305e+10]
 [ 2.52222457e+06 -2.08082267e+08  4.28052917e+09 -3.78706481e+10
   1.76729135e+11 -4.77167406e+11  7.71177917e+11 -7.35765267e+11
   3.82031324e+11 -8.32112857e+10]
 [-6.30548650e+06  5.35064026e+08 -1.12363190e+10  1.00987966e+11
  -4.77167413e+11  1.30136401e+12 -2.12073904e+12  2.03750560e+12
  -1.06423196e+12  2.32992180e+11]
 [ 9.60826347e+06 -8.32315665e+08  1.77560645e+10 -1.61580162e+11
   7.71177937e+11 -2.12073906e+12  3.48018678e+12 -3.36350379e+12
   1.76583932e+12 -3.88321115e+11]
 [-8.75030694e+06  7.70028647e+08 -1.66326459e+10  1.52886135e+11
  -7.35765292e+11  2.03750564e+12 -3.36350382e+12  3.26740572e+12
  -1.72304690e+12  3.80396878e+11]
 [ 4.37512051e+06 -3.89824850e+08  8.50529667e+09 -7.88317004e+10
   3.82031340e+11 -1.06423199e+12  1.76583935e+12 -1.72304691e+12
   9.12202286e+11 -2.02086155e+11]
 [-9.23630451e+05  8.31272312e+07 -1.82880745e+09  1.70689317e+10
  -8.32112897e+10  2.32992187e+11 -3.88321124e+11  3.80396883e+11
  -2.02086157e+11  4.49080955e+10]]
```

In [ ]:
```
# 100x100 matrix
m = 100
a = hilbert(m)
b = np.array([[1]]*m)
for i in range(len(b)):
    b[i][0]=i+1


t = gaussianElimination(a,b)
print('If the following is an identity matrix then the inverse was found correct
print(np.matmul(t[2],t[0]))
```

```
If the following is an identity matrix then the inverse was found correctly:
[[ 1.62033993e+02 -1.30494155e+04  3.39760303e+05 ...  6.14340221e+08
  -3.08299851e+08 -2.49853661e+08]
```

```
[-1.32398727e+04  1.44055489e+06 -4.27970508e+07 ... -1.13155238e+11
  6.33852485e+10  4.15408004e+10]
[ 3.55371413e+05 -4.40843750e+07  1.41746233e+09 ...  4.97914340e+12
 -3.13614934e+12 -1.64488457e+12]
...
[-3.84863803e+08  1.08391616e+11 -6.17441003e+12 ...  3.83516450e+17
 -2.47754423e+17  1.83774058e+16]
[ 6.54909412e+08 -1.49019397e+11  7.58032023e+12 ... -3.31222611e+17
  3.03310826e+17 -9.94954939e+16]
[-1.34480376e+08  3.11948117e+10 -1.59530647e+12 ...  5.57843597e+16
 -8.96065446e+16  5.83555373e+16]]
```

In [ ]:

```
# 200x200 matrix
m = 20
a = hilbert(m)
b = np.array([[1]]*m)
for i in range(len(b)):
    b[i][0]=i+1


t = gaussianElimination(a,b)
print('If the following is an identity matrix then the inverse was found correct
print(np.matmul(t[2],t[0]))
```

```
If the following is an identity matrix then the inverse was found correctly:
[[ 1.14433019e+02 -5.69344557e+03  6.19960855e+04  2.70706569e+05
  -8.93238975e+06  6.51095110e+07 -2.24514188e+08  3.52538616e+08
   7.12267519e+06 -8.33786976e+08  8.80084502e+08  4.06411236e+08
  -1.25447686e+09  8.28123788e+08 -5.45951746e+08 -2.01230645e+07
   1.91192220e+09 -2.95019087e+09  1.78032758e+09 -3.93991180e+08]
 [-5.74397759e+03  2.77059873e+05  1.36930008e+06 -1.41216400e+08
   1.89358432e+09 -1.16591310e+10  3.77585003e+10 -5.81848597e+10
   2.14051223e+09  1.26006344e+11 -1.32925962e+11 -6.99920421e+10
   2.12570260e+11 -1.51890081e+11  9.04794203e+10  1.78523267e+10
  -3.05987077e+11  4.51493559e+11 -2.68410626e+11  5.89948526e+10]
 [ 6.57567399e+04  1.09014109e+06 -3.07584481e+08  7.98195692e+09
  -8.68169247e+10  4.94090267e+11 -1.54651029e+12  2.37573099e+12
  -2.59238209e+11 -4.64116918e+12  4.93114055e+12  2.98258714e+12
  -8.94133968e+12  6.81591828e+12 -3.65954002e+12 -1.31540562e+12
   1.20290022e+13 -1.69276938e+13  9.90231091e+12 -2.16074233e+12]
 [ 1.78315179e+05 -1.32031979e+08  7.80901361e+09 -1.60929445e+11
   1.60205347e+12 -8.74763640e+12  2.69170412e+13 -4.17465286e+13
   8.23432936e+12  7.19726508e+13 -7.77589943e+13 -5.50335676e+13
   1.63061191e+14 -1.31271883e+14  6.28278347e+13  3.39505742e+13
  -2.02019257e+14  2.69337494e+14 -1.54643585e+14  3.34715380e+13]
 [-7.84237549e+06  1.77157688e+09 -8.37638926e+10  1.57719962e+12
  -1.50239378e+13  8.03092674e+13 -2.45977164e+14  3.89919755e+14
  -1.18751206e+14 -5.72053079e+14  6.37188420e+14  5.47270561e+14
  -1.60010627e+15  1.34967444e+15 -5.63570988e+14 -4.30987096e+14
   1.78653371e+15 -2.22701505e+15  1.24872073e+15 -2.67627319e+14]
 [ 5.80268009e+07 -1.08191636e+10  4.70762350e+11 -8.50041493e+12
   7.92947991e+13 -4.20942015e+14  1.29872571e+15 -2.13556892e+15
   9.37591155e+14  2.48547899e+15 -2.92418931e+15 -3.27737603e+15
   9.39219900e+15 -8.23875531e+15  2.88644670e+15  3.12120913e+15
  -9.13448006e+15  1.03656578e+16 -5.61405336e+15  1.18680225e+15]
 [-1.98803110e+08  3.46732379e+10 -1.45913226e+12  2.59491513e+13
  -2.41381163e+14  1.29223858e+15 -4.08213014e+15  7.12324913e+15
  -4.40918659e+15 -5.50739315e+15  7.27566579e+15  1.25825747e+16
  -3.47383534e+16  3.13895415e+16 -8.53577259e+15 -1.38473017e+16
   2.77331537e+16 -2.68942882e+16  1.36489391e+16 -2.81408039e+15]
 [ 3.12392874e+08 -5.40218020e+10  2.28685904e+12 -4.13487593e+13
   3.94969533e+14 -2.19781113e+15  7.36051837e+15 -1.42821819e+16
   1.27503006e+16  2.63684306e+15 -6.52652954e+15 -3.22222512e+16
   8.22930859e+16 -7.52634972e+16  1.29796944e+16  3.85627510e+16
```

```
 -4.77333720e+16  3.17729858e+16 -1.29030554e+16  2.41666564e+15]
[-6.09118320e+07  1.59078041e+10 -8.97076166e+11  2.03647873e+13
 -2.37106372e+14  1.59090052e+15 -6.48155357e+15  1.60629989e+16
 -2.27138155e+16  1.57082341e+16 -1.32781675e+16  5.73325840e+16
 -1.23390191e+17  1.09037335e+17 -2.93989881e+15 -6.61514682e+16
  3.72927672e+16  8.35775111e+15 -1.36913145e+16  3.48146055e+15]
[-3.05979103e+08  4.01909373e+10 -1.22035895e+12  1.38221688e+13
 -5.23946949e+13 -1.95134393e+14  2.68674615e+15 -1.14967170e+16
  2.64952458e+16 -3.77998730e+16  4.62353437e+16 -7.72898652e+16
  1.17217676e+17 -8.47081010e+16 -2.08726975e+16  6.53065085e+16
 -3.37228202e+15 -4.95708431e+16  3.52223343e+16 -7.81858862e+15]
[-5.22675532e+08  8.99454433e+10 -3.77254128e+12  6.74599034e+13
 -6.37960214e+14  3.53852530e+15 -1.20785013e+16  2.57295364e+16
 -3.46564948e+16  3.59958420e+16 -5.34803115e+16  9.36653399e+16
 -9.94456262e+16  3.78045976e+16  2.52870288e+16 -3.96995856e+16
  3.73787595e+16 -3.52127132e+16  2.04866834e+16 -4.73889758e+15]
[ 2.64336967e+09 -4.24812625e+11  1.68013760e+13 -2.84723513e+14
  2.55333394e+15 -1.33563862e+16  4.22278412e+16 -7.89604966e+16
  7.71656557e+16 -2.51102965e+16  1.97971247e+16 -1.12534029e+17
  1.59801967e+17 -8.00745964e+16  3.23111306e+14  5.79185259e+16
 -1.64597343e+17  2.09357926e+17 -1.20470295e+17  2.62263032e+16]
[-3.57221441e+09  5.80926968e+11 -2.32935617e+13  4.01325003e+14
 -3.67369819e+15  1.97305042e+16 -6.46107011e+16  1.26837979e+17
 -1.32404484e+17  3.92895657e+16  1.41462669e+16  1.13171489e+17
 -2.52337075e+17  1.80565654e+17 -3.20065186e+15 -1.27023301e+17
  2.21748599e+17 -2.30872885e+17  1.25154110e+17 -2.68999817e+16]
[ 2.31523200e+09 -3.88710926e+11  1.60551294e+13 -2.84709213e+14
  2.68370246e+15 -1.48682222e+16  5.03948083e+16 -1.02987891e+17
  1.12936638e+17 -3.45535375e+16 -2.57399298e+16 -7.46892952e+16
  2.12321176e+17 -1.53672896e+17 -2.59921631e+16  1.12685041e+17
 -1.08093432e+17  8.54566039e+16 -4.63062919e+16  1.06947300e+16]
[-8.97961801e+08  1.48479194e+11 -6.01011216e+12  1.03646170e+14
 -9.38878970e+14  4.90146731e+15 -1.50823804e+16  2.55515126e+16
 -1.53808117e+16 -1.74568795e+16  2.21797106e+16  2.61465971e+16
 -4.89620261e+16  6.09130735e+15  7.14620903e+15  3.81446499e+16
 -4.54465720e+16  4.69401131e+15  1.31527628e+16 -4.83846397e+15]
[-4.24147520e+08  7.63974572e+10 -3.34802993e+12  6.27315279e+13
 -6.24952780e+14  3.67376353e+15 -1.33146511e+16  2.94507940e+16
 -3.54659429e+16  1.05284343e+16  2.23334379e+16 -3.90219953e+15
 -5.25617832e+16  4.30643619e+16  6.20365004e+16 -1.53906872e+17
  1.47675330e+17 -8.10016664e+16  2.57007679e+16 -3.74478167e+15]
[ 2.48782666e+09 -3.89920607e+11  1.49729055e+13 -2.44520554e+14
  2.08511684e+15 -1.01018231e+16  2.77983474e+16 -3.66459331e+16
 -6.46285041e+15  8.85476475e+16 -9.54328518e+16 -1.12536434e+15
  3.76919494e+16  3.72054560e+16 -8.13294429e+16  1.14181789e+17
 -2.19970570e+17  2.54484911e+17 -1.40515193e+17  2.98187484e+16]
[-3.32791303e+09  5.06144193e+11 -1.88253303e+13  2.96360129e+14
 -2.41244097e+15  1.09239711e+16 -2.65250640e+16  2.25604764e+16
  4.50002061e+16 -1.35344909e+17  1.11341095e+17  5.03189792e+15
 -3.39177797e+15 -7.98966224e+16  3.75654990e+16 -2.40583748e+16
  2.10286634e+17 -3.20068860e+17  1.89848638e+17 -4.11384080e+16]
[ 1.91692182e+09 -2.88146644e+11  1.05845672e+13 -1.64285635e+14
  1.31386412e+15 -5.79832983e+15  1.33890271e+16 -8.82842425e+15
 -3.03063643e+16  7.73030913e+16 -5.82699228e+16 -4.75337368e+15
 -2.96060781e+15  4.37488943e+16 -3.45695216e+15 -8.41377394e+15
 -1.07550229e+17  1.79021060e+17 -1.07619371e+17  2.33353998e+16]
[-4.15559584e+08  6.21256307e+10 -2.26919592e+12  3.50051395e+13
 -2.77961548e+14  1.21523424e+15 -2.76046368e+15  1.67017457e+15
  6.62508689e+15 -1.62176527e+16  1.17790297e+16  1.36812236e+15
  5.15162131e+14 -8.27774754e+15 -1.39259865e+15  3.57853667e+15
  2.21867977e+16 -3.79154957e+16  2.28016279e+16 -4.93065015e+15]]
```

From what I can tell from the output above that as I increased the size of the Hilbert matrix the accuracy of my inverse matrix got worse and worse most likely due how small the values of this

matrix get when there are large lengthed matrices invovled.

Problem # 11

A) ① $\max(|x_2|, |x_3|, \ldots, |x_n|)$ will give a value greater than or equal to zero.

$\max(|x_2|, \ldots, |x_n|) \geq 0$

This can only equal to zero when $x_2 = x_3 = \ldots = x_n = 0$
Therefore the positivity condition is met.

② (linearity)

~~$\|\max(2|x_2|, 2|x_3|, \ldots, 2|x_n|)$~~

$\|\max(|\lambda x_2|, |\lambda x_3|, \ldots, |\lambda x_n|)\| = |\lambda| \, \|\max(|x_2|, |x_3|, \ldots, |x_n|)\|$

$\|(|\lambda x_{max}|)\| = |\lambda| \cdot \|(|x_{max}|)\|$

$|\lambda| \cdot \|(|x_{max}|)\| = |\lambda| \cdot \|(|x_{max}|)\|$ ✓ linearity condition met

③ (triangle inequality)

$\|\max(|x_2 + y|, \ldots, |x_n + y|)\| \leq \|\max(|x_2|, \ldots, |x_n|)\| + \|y\|$

$\|\max(|x_2|, \ldots, |x_n|) + \max(|y|, \ldots, |y|)\| \leq$ " " " "

$\|(\{x_{max}\}) + y\| \leq \|x_{max}\| + \|y\|$ x_{max}s are same

$\sqrt{(x_{max\,i} + y_i)^2 + \ldots} \leq \sqrt{(x_{max\,i})^2 + \ldots} + \sqrt{y_i^2 + \ldots}$

$\sqrt{(x_{max\,i} + y_i)^2 + \ldots} \leq$ ~~$(x_{max\,i}) + (y_i$~~ $x_{max\,i} + y_i + \ldots$

This inequality is true always for this function because x_{max} is either $\vec{0}$ or positive both of which satisfy the inequality therefore the triangle inequality is satisfied

B) ① Positivity

$$\left\| \sum_{i=1}^{n} |x_i|^3 \right\| = \left\| (|x_1|^3) \right\| + \left\| (|x_2|^3) \right\| + \dots + \left\| (|x_n|^3) \right\|$$

therefore $\left\| \sum_{i=1}^{n} |x_i|^3 \right\| = 0$ only when the individual norms equal zero.

$\left\| (|x_i|^3) \right\| = 0$ only when $x_i = 0$ therefor the positivity condition is not

② linearity

$$\left\| \sum_{i=1}^{n} |\lambda x_i|^3 \right\| \stackrel{?}{=} |\lambda| \left\| \sum_{i=1}^{n} |x_i|^3 \right\|$$

Let x_1 be 4 and x be 1-D so $n=1$

$$\left\| (\lambda 4)^3 \right\| \stackrel{?}{=} |\lambda| \left\| (4)^3 \right\|$$

$$\sqrt{(\lambda 4)^6} \stackrel{?}{=} |\lambda| \sqrt{4^6}$$

$$(\lambda 4)^3 \stackrel{?}{=} |\lambda| \, 4^3$$

$$\lambda^3 4^3 \neq |\lambda| \, 4^3 \quad \text{for all } \lambda \in \mathbb{R}^n$$

Therefore this is not a norm on $\mathbb{R}^n$

C) ① Positivity

$\left\| \left(\sum_{i=1}^{n} |x_i|^{1/2} \right)^2 \right\| = 0$ when $\sum_{i=1}^{n} |x_i|^{1/2} = 0$ which only occurs when $x_1 = x_2 = \dots = x_n = 0$ therefor linearity condition is satisfied

Beschreibung

② linearity

$$\| \left(\sum_{i=1}^{n} |\lambda x_i|^{1/2} \right)^2 \| = |\lambda| \, \| \left(\sum_{i=1}^{n} |x_i|^{1/2} \right)^2 \|$$

$$\| \left(\sum_{i=1}^{n} |\lambda|^{1/2} |x_i|^{1/2} \right)^2 \| = \text{" " " "}$$

$$\| \left(|\lambda|^{1/2} \sum_{i=1}^{n} |x_i|^{1/2} \right)^2 \| = \text{" " " "}$$

$$\| |\lambda| \left(\sum_{i=1}^{n} |x_i|^{1/2} \right)^2 \| = \text{" " " "}$$

$$\sqrt{\left(|\lambda| \left(\sum_{i=1}^{n} |x_i|^{1/2} \right)^2 \right)^2} = \text{" " " "}$$

$$|\lambda| \sqrt{\left(\left(\sum_{i=1}^{n} |x_i|^{1/2} \right)^2 \right)^2} = \text{" " " "}$$

$$|\lambda| \, \| \left(\sum_{i=1}^{n} |x_i|^{1/2} \right)^2 \| = |\lambda| \, \| \left(\sum_{i=1}^{n} |x_i|^{1/2} \right)^2 \| \quad \checkmark$$

Therefore linearity condition met

③ $$\| \left(\sum_{i=1}^{n} |x_i + y_i|^{1/2} \right)^2 \| \overset{?}{\leq} \|y\| + \|f\|$$

let $x = \begin{pmatrix} 1 \\ 8 \end{pmatrix}$ $y = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$

$$\| \left(\sqrt{1+3} + \sqrt{8+1} \right)^2 \| \overset{?}{\leq} \sqrt{10} + \sqrt{65}$$

$$25 \overset{?}{\leq} \sqrt{10} + \sqrt{65} \approx 11.22$$

The inequality doesn't hold therefore this function doesn't hold true in $\mathbb{R}^n$

D) ① Positivity

$$\| \sum_{i=1}^{n} \lambda^{-i} |x_i| \| = 0 \Rightarrow \| \lambda^{-1}|x_1| + \lambda^{-2}|x_2| + \ldots + \lambda^{-n}|x_n| \| = 0$$

This only equals zero when $\lambda^{-i}|x_i| = 0$ which can only be true when x_i equals zero for $i = 1 \ldots n$. Therefore positivity condition is met

(2) linearity

$$\left\| \sum_{i=1}^{n} 2^{-i} |\lambda x_i| \right\| = |\lambda| \left\| \sum_{i=1}^{n} 2^{-i} |x_i| \right\|$$

$$\sqrt{(2^{-1}|\lambda x_1|)^2 + (2^{-2}|\lambda x_2|)^2 + \dots} = |\lambda| \sqrt{(2^{-1}|x_1|)^2 + \dots}$$

$$\sqrt{\left(\frac{|\lambda||x_1|}{2}\right)^2 + \left(\frac{|\lambda||x_2|}{2^2}\right)^2 + \dots} = |\lambda| \sqrt{\left(\frac{|x_1|}{2}\right)^2 + \dots}$$

$$\sqrt{\frac{\lambda^2 x_1^2}{2^2} + \frac{\lambda^2 x_2^2}{2^4} + \dots} = |\lambda| \sqrt{\frac{x_1^2}{2^2} + \dots}$$

$$\sqrt{\lambda^2 \left(\frac{x_1^2}{2^2} + \frac{x_2^2}{2^4} + \dots\right)} = |\lambda| \sqrt{\frac{x_1^2}{2^2} + \dots}$$

$$\sqrt{\lambda^2} \sqrt{\frac{x_1^2}{2^2} + \frac{x_2^2}{2^4} + \dots} = |\lambda| \sqrt{\frac{x_1^2}{2^2} + \dots}$$

$$\lambda \sqrt{\frac{x_1^2}{2^2} + \frac{x_2^2}{2^4} + \dots} = |\lambda| \sqrt{\frac{x_1^2}{2^2} + \dots} \quad \checkmark$$

(3) Triangle Inequality

$$\left\| \sum_{i=1}^{n} 2^{-i} |x_i + y_i| \right\| \le \left\| \sum_{i=1}^{n} 2^{-i} |x_i| \right\| + \left\| \sum_{i=1}^{n} 2^{-i} |y_i| \right\|$$

~~Let $x = (\quad)$ $y = (\quad)$~~

~~$\sqrt{(2^{-1}|0+2|)^2 + (2^{-2}|0+e|)^2} \le \sqrt{(2^{-1}|0|)^2 + (2^{-2}|0|)^2} + \sqrt{(2^{-1}|2|)^2}$~~

$$\sqrt{(2^{-1}|x_1+y_1|)^2 + (2^{-2}|x_2+y_2|)^2 + \dots} \le \sqrt{(2^{-1}|x_1|)^2 + \dots} + \sqrt{(2^{-1}|y_1|)^2 + \dots}$$

$$\left[\sqrt{(2^{-1}|x_1+y_1|)^2 + \dots}\right]^2 \le \left[\sqrt{(2^{-1}|x_1|)^2 + \dots} + \sqrt{(2^{-1}|y_1|)^2 + \dots}\right]^2$$

$$(2^{-1}|x_1+y_1|)^2 + \dots \le \underbrace{((2^{-1}|x_1|)^2 + \dots)}_{\overline{X}_{1\dots n}} + \underbrace{((2^{-1}|y_1|)^2 + \dots)}_{\overline{Y}_{1\dots n}} + \underbrace{2((2^{-1}|x_1|)^2 + \dots)((2^{-1}|y_1|)^2 + \dots)}_{\overline{Z}_{1\dots n}}$$

$$\underbrace{\frac{x_1^2}{2}}_{\overline{x}_1} + \underbrace{\frac{y_1^2}{2}}_{\overline{y}_1} + \underbrace{\frac{2|x_1 y_1|}{2}}_{\overline{z}_1} + \dots \le \overline{X}_{1\dots n}$$

Therefor the condition holds true.