# Homeowork 12

## Problem 1

### a)

In [1]:
```
import numpy as np
import scipy as scipy
import householder as hh
```

In [2]:
```
a = lambda x, y, z: 6*x+2*y+2*z
b= lambda x,y,z: 2*x +(2/3)*y+(1/3)*z
c= lambda x,y,z: x+2*y-z
```

In [3]:
```
a1=a(2.6,-3.8,-5)
b1=b(2.6,-3.8,-5)
c1=c(2.6,-3.8,-5)
print("Equation 1(x,y,z)=",a1)
print("Equation 2(x,y,z)=",b1)
print("Equation 3(x,y,z)=",c1)
```

```
Equation 1(x,y,z)= -1.9999999999999982
Equation 2(x,y,z)= 1.0000000000000004
Equation 3(x,y,z)= 0.0
```

### a-d)

1

a) $(x, y, z) = (2.6, -3.8, -5)$

$15.6 - 7.6 - 10 = -2$

$-2 = -2$

b) $\begin{bmatrix} 6 & 2 & 2 \\ 2 & 2/3 & 1/3 \\ 1 & 2 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \\ 0 \end{bmatrix}$

so: $\left[\begin{array}{ccc|c} 6 & 2 & 2 & -2 \\ 2 & 0.6667 & 0.3333 & 1 \\ 1 & 2 & -1 & 0 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 0 & 0 & 1 & -5 \\ 2 & 0.6667 & 0.3333 & 1 \\ 1 & 2 & -1 & 0 \end{array}\right]$

$\left[\begin{array}{ccc|c} 0 & 0 & 1 & -5 \\ 2 & 0.6667 & 0 & 2.6665 \\ 1 & 2 & 0 & -5 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 0 & 0 & 1 & -5 \\ 1.6667 & 0 & 0 & 1.3332 \\ 1 & 2 & 0 & -5 \end{array}\right]$

$\left[\begin{array}{ccc|c} 0 & 0 & 1 & -5 \\ 1 & 0 & 0 & 2.5998 \\ 1 & 2 & 0 & -5 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 0 & 0 & 1 & -5 \\ 1 & 0 & 0 & 2.5998 \\ 0 & 2 & 0 & 7.5998 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 0 & 0 & 1 & -5 \\ 1 & 0 & 0 & 2.5998 \\ 0 & 1 & 0 & -3.7999 \end{array}\right]$

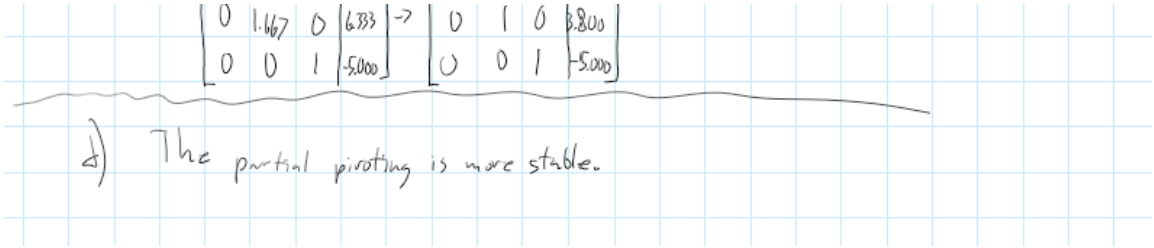
so $x = 2.5998$, $y = -3.7999$, $z = -5$

c) assuming you mean part b

$\left[\begin{array}{ccc|c} 6 & 2 & 2 & -2 \\ 2 & 0.6667 & 0.3333 & 1 \\ 1 & 2 & -1 & 0 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 6 & 2 & 2 & -2 \\ 0 & 0 & -0.3333 & 1.667 \\ 0 & 1.667 & -1.333 & 0.3333 \end{array}\right]$

$\left[\begin{array}{ccc|c} 6 & 2 & 2 & -2 \\ 0 & 1.667 & -1.333 & 0.3333 \\ 0 & 0 & -0.3333 & 1.667 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 6 & 0 & 3.600 & -2.4 \\ 0 & 1.667 & -1.333 & 0.3333 \\ 0 & 0 & -0.3333 & 1.667 \end{array}\right]$

$\left[\begin{array}{ccc|c} 6 & 0 & 0 & 15.60 \end{array}\right] \left[\begin{array}{ccc|c} 1 & 0 & 0 & 2.600 \end{array}\right]$




# Problem 2

In [4]:
```
A = np.array(([12,10,4],[10,8,-5],[4,-5,3]))
print(A)
```

```
[[12 10  4]
 [10  8 -5]
 [ 4 -5  3]]
```

In [5]:
```
print(np.linalg.eigvals(A))
```

```
[-5.1984251 20.1984251  8.       ]
```

In [6]:
```
D,C = hh.householder(A)
A3 = np.diag(D)+np.diag(C,1)+np.diag(C,-1)
print(A3)
print(np.linalg.eigvals(A3))
```

```
[10  4]
[[ 12 -10   0]
 [-10   3   5]
 [  0   5   7]]
[-4.9774314  19.11607184  7.86135957]
```

The above finding for the householder matrix supplies eigenvalues close to the original. I believe this is due to some numerical errors in the calculation.

# Problem 3

In [51]:
```
def power(N):
    A=np.zeros((N,N))
    for i in range(1,N):
        for j in range(1,i+1):
            A[i][j]=1/(i+j-1)

    x = np.ones(N).T
    tol = 1e-6
    max_iter = 100

    lam_store = 0

    for i in range(max_iter):
        x = A@x / np.linalg.norm(A@x)
```

```
        lam = (x.T@A@x)/(x.T@x)

        if np.abs(lam - lam_store) < tol:
            break

        lam_store = lam

    return lam, x, i

def powerNoN(A):
    x = np.ones(len(A)).T
    tol = 1e-6
    max_iter = 100

    lam_store = 0

    for i in range(max_iter):
        x = A@x / np.linalg.norm(A@x)

        lam = (x.T@A@x)/(x.T@x)

        if np.abs(lam - lam_store) < tol:
            break

        lam_store = lam

    return lam, x, i

def powersmall(N):
    B=np.zeros((N,N))
    for i in range(0,N):
        for j in range(0,N):
            B[i][j]=1/(i+j+1)

    lambig,xbig,iterbig = power(N)
    A = B-lambig*np.identity(N)

    x = np.ones(N).T
    tol = 1e-6
    max_iter = 100

    lam_store = 0

    for i in range(max_iter):
        x = A@x / np.linalg.norm(A@x)

        lam = (x.T@A@x)/(x.T@x)

        if np.abs(lam - lam_store) < tol:
            break

        lam_store = lam

    return lam+lambig, x, i
```

In [17]:
```
lam1,x1,iter1 = power(4)
print(lam1)
print(x1)
print(iter1)
```

```
0.9999996889870862
[0.         0.70953094 0.53214854 0.46193483]
11
```

In [18]:
```
lam2,x2,iter2 = power(20)
print(lam2)
print(x2)
print(iter2)
```

```
0.9999996302430362
[0.         0.43410165 0.32557626 0.28261831 0.25753438 0.24035505
 0.22752285 0.21740095 0.20911282 0.20213897 0.19614823 0.19091758
 0.18629022 0.18215206 0.1784177  0.17502169 0.17191282 0.16905038
 0.16640147 0.16393916]
13
```

b)

In [36]:
```
lamsmall,xsmall,itersmall = powersmall(16)
print(lamsmall)

N=16
B=np.zeros((N,N))
for i in range(0,N):
    for j in range(0,N):
        B[i][j]=1/(i+j+1)


lamList = np.linalg.eigvals(B)
print(min(lamList)-lamsmall)
```

```
0.004951309919023927
(-0.004951309919023933-9.227487176840956e-19j)
```

This method is accurate to 3 decimal places.

c)

In [45]:
```
E = 0.01*np.ones((N,N))

C = B+E


lamPertMin = (np.min(np.linalg.eig(C)[0]))

print("Difference in smallest eigenvalues:",lamsmall-lamPertMin)
print("2-Norm of E:",np.linalg.norm(E))
```

```
Difference in smallest eigenvalues: 0.004951309919023941
2-Norm of E: 0.16
```

Therefore this is consistent with the Bauer-Fike Theorem.

## d)

In [57]:
```
F = np.array([[1,0,0],[0,(np.sqrt(2)/2)*(1-1j),0],[0,0,(np.sqrt(2)/2)*(1+1j)]])

print(F)
```

```
[[1.         +0.j         0.         +0.j         0.         +0.j        ]
 [0.         +0.j         0.70710678-0.70710678j 0.         +0.j        ]
 [0.         +0.j         0.         +0.j         0.70710678+0.70710678j]]
```

In [58]:
```
print(powerNoN(F))
```

```
((0.804737854124365+5.204480025290235e-18j), array([ 0.57735027+0.00000000e+00j,
-0.57735027-6.00627088e-17j,
       -0.57735027+6.00627088e-17j]), 99)
```

Here $|\lambda| = 1$ for all eigenvalues, and is the maximum, therefore the power method will break down as $\frac{|\lambda_2|}{|\lambda_1|}$ is 1 and no eigenvalue is approached.