Homeworkalgorithm

May 21, 2019

```
In [78]: %matplotlib inline
         import numpy as np
         import matplotlib.pyplot as plt


         #Gau elimination algortihm
         def Gau(A,b):
                 n=len(A)
                 print('Rang(A)=',n)

                 #check if rang(A)=dim(b)
                 if len(b)!=n:
                     raise ValueError ("Invalid argument: incompatible sizes between A & b.",

                 #bring matrice into upper triangular matrice
                 for r in range(n-1):
                     for i in range(1,n-r):
                         if A[r][r]==0: A=A
                         elif A[r][r]!=0:

                             m=A[r+i][r]/A[r][r]          #find multiplicator
                             if m==0: A=A
                             elif m!=0:
                                 #for A
                                 A[r]=A[r]*m
                                 A[r+i]=A[r+i]-A[r]
                                 A[r]=A[r]/m
                                 #for b
                                 b[r]=b[r]*m
                                 b[r+i]=b[r+i]-b[r]
                                 b[r]=b[r]/m

                 print("2. Upper Triangular Matrix: solved")

                 #substract rows to achieve diagonal matrice
                 for f in range(n-1):
                     for t in range(f+1,n):
```

```python
                if A[f][t]==0: A[f]=A[f]
                if A[t][t]==0: A[f]=A[f]
                elif A[f][t]!=0:
                    k=A[f][t]/A[t][t]

                    #for A
                    A[f]=A[f]-k*A[t]
                    #for b
                    b[f]=b[f]-k*b[t]

        print("3. Elimination to diagonal matrix: solved")

        #solve for b
        x=np.zeros(n)
        for t in range(n):
            x[t]=b[t]/A[t][t]
        print('4. Solution vetor x=',x)

        #Check if found x is correct
        print('5. Check Solution with original matrix A*x=b_check')
        print('Original solution b=', bOG)
        xc=np.zeros_like(bOG)
        for g in range(n):
            for h in range(n):
                xc[g]=xc[g]+AOG[g][h]*x[h]
        print('b_check=',xc)
```

In [80]:
```python
#initial values tridiaganol Matix

N=10


a=np.full(N,-1)
b=np.full(N,2)
c=np.full(N,-1)
c[N-1]=0

#create Matrix A from Input
A=np.zeros((N,N))
for k in range(N):
    A[k][k]=b[k]
for k in range(N-1):
    A[k][k+1]=a[k]
    A[k+1][k]=c[k]

#solution y for tridiaganol Matrix
```

```python
b=np.full(N, 0.1)

#preserve original matrix A as AOG and b as bOG
AOG=np.copy(A)
bOG=np.copy(b)

print('1. Inital Input for A*x=b')
print('A=', A)
print('b=', b)

print(Gau(A,b))
```

```
1. Inital Input for A*x=b
A= [[ 2. -1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [-1.  2. -1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. -1.  2. -1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. -1.  2. -1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  2. -1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. -1.  2. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. -1.  2. -1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0. -1.  2. -1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. -1.  2. -1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. -1.  2.]]
b= [0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1]
Rang(A)= 10
2. Upper Triangular Matrix: solved
3. Elimination to diagonal matrix: solved
4. Solution vetor x= [0.5 0.9 1.2 1.4 1.5 1.5 1.4 1.2 0.9 0.5]
5. Check Solution with original matrix A*x=b_check
Original solution b= [0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1]
b_check= [0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1]
None
```