lp_programming_2

September 29, 2022

0.1 Programming Assignment #2, LP, Fall 2022

0.1.1 Aidan Pimentel (amp5496), Alexander Mills (adm5547), Matt Skiles (ms82657)

[14]:
```
import numpy as np
import pandas as pd
import random
from scipy.sparse import rand
from scipy.linalg import lu_factor, lu_solve, cho_factor, cho_solve
```

[15]:
```
def A_matrix (U, L, density, m, n):
    #define a matrix of random values between 0 and 1 of specific density and␣
 ↪size
    matrix=rand(m,n,density)
    #interpolate between upper and lower bounds with randomly generated number
    matrix = matrix*(U-L)+L
    #convert array to dataframe
    matrix_df=pd.DataFrame(matrix.toarray())
    #cycle through rows and check if all values in row are zero
    for row in matrix_df.index:
        if (matrix_df.loc[row,:]==0).all():
            #if all values in row are zero, then recurse
            return A_matrix(U,L,density,m,n)
    #cycle through columns and check if all values in column are zero
    for col in matrix_df.columns:
        if (matrix_df.loc[:,col]==0).all():
            #if all values in column are zero, then recurse
            return A_matrix(U,L,density,m,n)

    return matrix.toarray()

def b_matrix (U, L, density, m, n):
    #define a matrix of random values between 0 and 1 of specific density and␣
 ↪size
    matrix=rand(m,n,density)
    #interpolate between upper and lower bounds with randomly generated number
    matrix = matrix*(U-L)+L
    #convert array to dataframe
```

```python
    matrix_df=pd.DataFrame(matrix.toarray())
    #cycle through rows and check if all values in row are zero

    return matrix.toarray()
```

0.2 Part 1: LU factorization routines LFTRG / LFSRG

[7]:
```python
A=A_matrix(100,0,0.4,10,10)
b=b_matrix(50,0,0.8,10,1)
lu, piv = lu_factor(A)

x = lu_solve((lu, piv), b)
print(x)
```

```
[[-0.22309735]
 [ 0.32530725]
 [ 0.32022353]
 [-0.73688055]
 [ 0.30071514]
 [ 0.02832577]
 [ 0.00653248]
 [ 0.3054587 ]
 [ 0.17999383]
 [-0.30497714]]
```

0.3 Part 2: Bx = b directly (using LSLRG)

[8]:
```python
A=A_matrix(100,0,0.4,10,10)
b=b_matrix(50,0,0.8,10,1)
lu, piv = lu_factor(A)

x = lu_solve((lu, piv), b)
print(x)
```

```
[[ 0.99714519]
 [ 0.17390249]
 [-1.45324718]
 [ 1.48732842]
 [ 0.38180266]
 [-0.65679815]
 [-0.1894038 ]
 [-2.16785384]
 [ 0.33793284]
 [-1.48453791]]
```

0.4 Part 3: find an inverse of a square matrix (LINRG) and then use (MURRV)

[10]:
```
x = np.linalg.solve(A,b)
print(x)
```

```
[[ 0.99714519]
 [ 0.17390249]
 [-1.45324718]
 [ 1.48732842]
 [ 0.38180266]
 [-0.65679815]
 [-0.1894038 ]
 [-2.16785384]
 [ 0.33793284]
 [-1.48453791]]
```

[11]:
```
A_inv = np.linalg.inv(A)
print(A_inv.dot(b))
```

```
[[ 0.99714519]
 [ 0.17390249]
 [-1.45324718]
 [ 1.48732842]
 [ 0.38180266]
 [-0.65679815]
 [-0.1894038 ]
 [-2.16785384]
 [ 0.33793284]
 [-1.48453791]]
```

0.5 Part 4: Cholesky factors (CHFAC and LFSDS)

[12]:
```
def is_pos_def(x):
    return np.all(np.linalg.eigvals(x) > 0)

def cho_fun (B):
    D=(1/2*(B+np.transpose(B)))
    while False == is_pos_def(D):
        np.fill_diagonal(D, D.diagonal() + 20)        # we add a large number to␣
 ↪diagonal
    return cho_solve(cho_factor(D),b)

cho_fun(A)
```

[12]:
```
array([[-0.75206645],
       [ 3.4219417 ],
       [-1.16231783],
       [ 0.22123317],
```

```
 [ 2.61866061],
 [-4.11706996],
 [-0.90013178],
 [ 0.88876551],
 [-0.96993503],
 [ 2.67142054]])
```

end :)