lp_programming_2

September 22, 2022

[ ]:
```
## Programming Assignment #2, LP, Fall 2022
### Aidan Pimentel (amp5496), Alexander Mills (adm5547), Matt Skiles (ms82657)
```

[ ]:
```
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
```

[19]:
```
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
```

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

    return matrix.toarray()
```

[20]:
```
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)
```

```
[[ 0.108236  ]
 [ 0.24695984]
 [ 0.75035242]
 [-1.92928781]
 [-0.1544934 ]
 [ 0.02957011]
 [-0.27335293]
 [-0.01142524]
 [ 0.51052497]
 [ 0.38513341]]
```

[21]:
```
x = np.linalg.solve(A,b)
print(x)
```

```
[[ 0.108236  ]
 [ 0.24695984]
 [ 0.75035242]
 [-1.92928781]
 [-0.1544934 ]
 [ 0.02957011]
 [-0.27335293]
 [-0.01142524]
 [ 0.51052497]
 [ 0.38513341]]
```

[22]:
```
A_inv = np.linalg.inv(A)
print(A_inv.dot(b))
```

```
[[ 0.108236  ]
 [ 0.24695984]
 [ 0.75035242]
 [-1.92928781]
 [-0.1544934 ]
 [ 0.02957011]
 [-0.27335293]
 [-0.01142524]
 [ 0.51052497]
```

```
 [ 0.38513341]]
```

[23]:
```
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)
```

[23]:
```
array([[ 1.40466061],
       [ 1.48829233],
       [ 1.97740526],
       [-0.45639418],
       [ 0.34999757],
       [-0.71108676],
       [-1.01265374],
       [ 0.81237752],
       [-0.97552707],
       [-2.12799683]])
```

[23]:
```

```