Problems week 2

Magnus Chr. Hvidtfeldt

Technical University of Denmark, Lyngby, DK,

s255792@dtu.dk

```
from sympy import *
import numpy as np
import matplotlib.pyplot as plt
init_printing()
from scipy.linalg import lu
x = symbols('x')
```

1 Løsning af lineære ligningssystemer

1.1 Pivotering og faktorisering

1.1.1 Udfør gauss elimination uden pivotering, og forklar hvorfor denne proces ikke kan fuldføres.

$$A = \begin{bmatrix} 4 & 4 & 2 \\ 2 & 2 & 2 \\ 3 & 2 & 1 \end{bmatrix} \to \begin{bmatrix} 4 & 4 & 2 \\ 0 & 0 & -2 \\ 0 & -1 & -1/2 \end{bmatrix} \tag{1.1}$$

Now, by naive gaussian elimination, we cannot perform the next step in the process, since i cannot divide by 0 without switching the rows 2 and 3, which would be my next step.

1.1.2 Partial pivoting

$$\begin{bmatrix} 4 & 4 & 2 \\ 0 & 0 & -2 \\ 0 & -1 & -1/2 \end{bmatrix} \to \begin{bmatrix} 4 & 4 & 2 \\ 0 & 1 & 1/2 \\ 0 & 0 & -2 \end{bmatrix} \to \begin{bmatrix} 4 & 4 & 2 \\ 0 & 1 & 1/2 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

In step 2, i interchange R2 and R3, and switch the signs. In step 3, I switch R3 to be 1.

1.1.3 Calculate using scipy.

```
A = np.array([[4, 4, 2],
            [2, 2, 2],
            [3, 2, 1]])
P, L, U = lu(A)
display(L, U)
```

```
array([[ 1.  ,  0.  ,  0.  ],
       [ 0.75,  1.  ,  0.  ],
       [ 0.5 , -0.  ,  1.  ]])
```

```
array([[ 4. ,  4. ,  2. ],
       [ 0. , -1. , -0.5],
       [ 0. ,  0. ,  1. ]])
```

The factors relate to the Gauss-elimination as L is the product of all the elementary matrices and we see that U is the matrix we found by partial pivoting as expected.

1.2 Error and error judgment

```python
# 1
A = np.array([[3, 12, 10],
             [12, 0, 20],
             [0, 2, 30]])
b = np.array([72.3, 99.5, 56.5])
b_tilde = np.array([73.3, 98.4, 57.1])
sol1 = np.linalg.solve(A,b)
sol2 = np.linalg.solve(A,b_tilde)

nom = np.linalg.norm(b_tilde - b)
nomb = np.linalg.norm(b)
display("Relative norm b:", nom/nomb)

xnorm = np.linalg.norm(sol2 - sol1)
xnomb = np.linalg.norm(sol1)
display("Relative norm x:", xnorm/xnomb)

# 2
cond = np.linalg.cond(A)
condnum = cond * nom/nomb
# 3
cc = condnum/(xnorm/xnomb)*100
display("2. Konditionstal:", cond, "3. Den er: større %", cc)
```

```
'Relative norm b:'
```

0.0118442159246757

```
'Relative norm x:'
```

0.0230501153204702

```
'2. Konditionstal:'
```

3.90796509814076

```
'3. Den er: større %'
```

200.809331341478

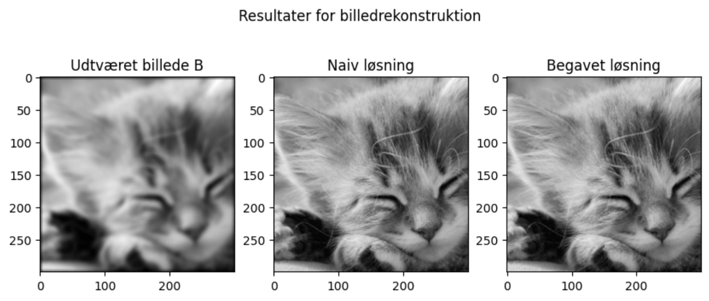
1.3 Brug af faktorisering til billedrestaurering

```python
from scipy import io
deblur = io.loadmat('deblur.mat')
A, B = deblur['A'], deblur['B']
A_inv = np.linalg.inv(A)
X_t = A_inv @ B @ A_inv

for i in range(1000):
    # Transponerer output da numpy giver en lower og vi ønsker en
    øvre som udgangspunkt. Ik spørg hvorfor.
    U = np.linalg.cholesky(A).T
    Ut = U.T
    X = np.linalg.solve(Ut.T,np.linalg.solve(U,np.linalg.solve(U.T,
    np.linalg.solve(Ut,B).T)).T)

condA = np.linalg.cond(A)**2
normB = np.linalg.norm(B)
E_max = 0.02 * normB/condA**2

E = np.random.rand(*B.shape)
E = E_max * E / np.linalg.norm(E)

Xt = A_inv @ (B+E) @ A_inv
rel_fejl = np.linalg.norm(X - X_t) / np.linalg.norm(X)
display(rel_fejl)
```

$1.46696900465391 \cdot 10^{-14}$

Da kan vi se at vores relative fejl er < 0.02.

```python
fig, axs = plt.subplots(1, 3, figsize=(10, 4))
fig.suptitle('Resultater for billedrekonstruktion')
axs[0].imshow(B, cmap='gray')
axs[0].set_title('Udtværet billede B')
axs[1].imshow(X_t, cmap='gray', vmin=0, vmax=256)
axs[1].set_title('Naiv løsning')
axs[2].imshow(Xt, cmap='gray', vmin=0, vmax=256)
axs[2].set_title('Begavet løsning')
plt.show()
```


Resultater for billedrekonstruktion
Udtværet billede B
Naiv løsning
Begavet løsning