homework2

November 14, 2021

1 Homework Set 2

Please **submit this Jupyter notebook through Canvas** no later than **Mon Nov. 15, 9:00**.

Homework is in **groups of two**, and you are expected to hand in original work. Work that is copied from another group will not be accepted.

2 Exercise 0

Write down the names + student ID of the people in your group.

Florian Tiggeloven 11872802 Jacco van Wijk 11282479

Run the following cell to import the necessary packages.

[1]:
```
import numpy as np
import scipy.linalg as la
import matplotlib.pyplot as plt
import random
```

3 Exercise 1

The goal of this problem is to show that apparently harmless looking systems of linear equations may be very difficult to solve. Some functions that may be useful are `numpy.triu`, `numpy.tril`, `numpy.eye`, `random.randrange`. ## (a) Generate an $n \times n$ matrix B with random integer elements in the range $b_{ij} \in [-10, 10]$. Choose for instance $n = 20$.

[2]:
```
n = 20
b_min = -10
b_max = 10
B = np.random.randint(b_max - b_min, size=(n,n)) + b_min

print('\n'.join([''.join(['{:4}'.format(item) for item in row])
      for row in B]))
```

```
   0   5   0   5   5   6   8  -7  -6  -9   2  -4   1  -4   6   7  -8   8   5   6
   6  -2  -9   7   9 -10  -4  -4  -9  -6  -7 -10   6   4  -3  -3  -2   8   7  -8
   4   7   8  -2  -2   5  -5  -8  -4   2  -8   9  -3  -3   8   0   7  -9  -9   6
   7   2   7   0   4   5   7  -4   9  -2   8  -6 -10   2  -9   1   5   3   7   0
```

```
 -1   2   4 -10  -5  -4  -7   6  -3  -9   9   2  -4   1   1   6  -9  -1   5  -2
 -3  -3  -7   8  -2   8   4  -4  -2   4   8  -5   8   6  -1 -10   4   9   8  -2
  4   9 -10   0  -3  -8   1  -4   3   1   8  -2   8  -3  -2   9  -3  -3  -1 -10
  5 -10  -6  -8   9   2   8   2 -10  -3   2  -4   3   1   3  -4 -10  -6  -5  -5
 -6  -7  -4  -2   9  -7   1   1 -10  -7 -10   1   4  -4   5   4   5   2  -8 -10
  5   2  -7   0  -9  -6   0   9   6 -10   4  -9   8   5  -2   7   2  -6   5   6
 -7   8   5   1  -3   1   2  -9   3  -3  -7   6   8   6  -3   2   6  -7 -10  -2
  8   8   4  -4   5  -4  -9   2   1  -5  -9  -1  -1   9   2   1  -7  -1  -7  -6
  1  -4   0  -8  -9  -5  -8  -4   4   1  -2  -4  -2  -4  -9   6  -5   5   4  -1
 -7  -9  -1   9   8  -9  -3   9   9  -9  -6  -2  -7  -4  -5  -7  -8   6   0   1
 -9   6   4   7  -9  -4 -10  -6  -4   4  -2   2   7   2   0  -2  -6   4   5   9
  3  -2   5   8   1   1   9 -10  -8  -3  -6   7 -10  -7   0   3  -8  -4   2  -7
 -3   2  -3   7  -4  -6  -8  -4   3  -7   5   1   0   0 -10 -10  -2   0  -6  -5
  7  -7   4  -2   2   0   7  -8  -3   4  -3  -4  -3   1  -1   2   6   7  -6  -1
 -7   5   5   4  -4 -10  -7   2   3  -2   7   5   8 -10   7  -7  -9   0  -1   6
  4  -5  -4   5  -5  -2   7  -6 -10   7   6  -7   9  -8   4  -2   5  -3   6  -6
```

3.1 (b)

Remove the diagonal of B, save the upper triangular part in U and the lower triangular part in L, and put ones on the diagonals $l_{ii} = u_{ii} = 1$.

[3]:
```
# Remove diagonal
np.fill_diagonal(B, 0)

# Create U and L
U = np.triu(B) + np.eye(n)
L = np.tril(B) + np.eye(n)

# Print U and L
print("U = ")
print('\n'.join([''.join(['{:4}'.format(round(item)) for item in row])
      for row in U]), end="\n\n")
print("L = ")
print('\n'.join([''.join(['{:4}'.format(round(item)) for item in row])
      for row in L]))
```

```
U =
   1   5   0   5   5   6   8  -7  -6  -9   2  -4   1  -4   6   7  -8   8   5   6
   0   1  -9   7   9 -10  -4  -4  -9  -6  -7 -10   6   4  -3  -3  -2   8   7  -8
   0   0   1  -2  -2   5  -5  -8  -4   2  -8   9  -3  -3   8   0   7  -9  -9   6
   0   0   0   1   4   5   7  -4   9  -2   8  -6 -10   2  -9   1   5   3   7   0
   0   0   0   0   1  -4  -7   6  -3  -9   9   2  -4   1   1   6  -9  -1   5  -2
   0   0   0   0   0   1   4  -4  -2   4   8  -5   8   6  -1 -10   4   9   8  -2
   0   0   0   0   0   0   1  -4   3   1   8  -2   8  -3  -2   9  -3  -3  -1 -10
   0   0   0   0   0   0   0   1 -10  -3   2  -4   3   1   3  -4 -10  -6  -5  -5
   0   0   0   0   0   0   0   0   1  -7 -10   1   4  -4   5   4   5   2  -8 -10
   0   0   0   0   0   0   0   0   0   1   4  -9   8   5  -2   7   2  -6   5   6
```

```
   0   0   0   0   0   0   0   0   0   0   1   6   8   6  -3   2   6  -7 -10  -2
   0   0   0   0   0   0   0   0   0   0   0   1  -1   9   2   1  -7  -1  -7  -6
   0   0   0   0   0   0   0   0   0   0   0   0   1  -4  -9   6  -5   5   4  -1
   0   0   0   0   0   0   0   0   0   0   0   0   0   1  -5  -7  -8   6   0   1
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1  -2  -6   4   5   9
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1  -8  -4   2  -7
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0  -6  -5
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1  -6  -1
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   6
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1

L =
   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   6   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   4   7   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   7   2   7   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  -1   2   4 -10   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  -3  -3  -7   8  -2   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   4   9 -10   0  -3  -8   1   0   0   0   0   0   0   0   0   0   0   0   0   0
   5 -10  -6  -8   9   2   8   1   0   0   0   0   0   0   0   0   0   0   0   0
  -6  -7  -4  -2   9  -7   1   1   1   0   0   0   0   0   0   0   0   0   0   0
   5   2  -7   0  -9  -6   0   9   6   1   0   0   0   0   0   0   0   0   0   0
  -7   8   5   1  -3   1   2  -9   3  -3   1   0   0   0   0   0   0   0   0   0
   8   8   4  -4   5  -4  -9   2   1  -5  -9   1   0   0   0   0   0   0   0   0
   1  -4   0  -8  -9  -5  -8  -4   4   1  -2  -4   1   0   0   0   0   0   0   0
  -7  -9  -1   9   8  -9  -3   9   9  -9  -6  -2  -7   1   0   0   0   0   0   0
  -9   6   4   7  -9  -4 -10  -6  -4   4  -2   2   7   2   1   0   0   0   0   0
   3  -2   5   8   1   1   9 -10  -8  -3  -6   7 -10  -7   0   1   0   0   0   0
  -3   2  -3   7  -4  -6  -8  -4   3  -7   5   1   0   0 -10 -10   1   0   0   0
   7  -7   4  -2   2   0   7  -8  -3   4  -3  -4  -3   1  -1   2   6   1   0   0
  -7   5   5   4  -4 -10  -7   2   3  -2   7   5   8 -10   7  -7  -9   0   1   0
   4  -5  -4   5  -5  -2   7  -6 -10   7   6  -7   9  -8   4  -2   5  -3   6   1
```

3.2 (c)

Compute $A = L \cdot U$. What is the value of $\det(A)$ and why? Compute the determinant using the appropriate python command and confirm your prediction. In case that you have doubts about the result, compute separately $\det(L)$ and $\det(U)$.

[4]:
```
# Compute A = L * U
A = L.dot(U)

# Print A
print('A = ')
print('\n'.join([''.join(['{:4}'.format(round(item)) for item in row])
      for row in A]))

print(np.linalg.det(A))
```

```
print(np.linalg.det(L), np.linalg.det(U))
```

```
A =
   1   5   0   5   5   6   8  -7  -6  -9   2  -4   1  -4   6   7  -8   8   5   6
   6  31  -9  37  39  26  44 -46 -45 -60   5 -34  12 -20  33  39 -50  56  37  28
   4  27 -62  67  81 -41  -1 -64 -91 -76 -49 -77  43   9  11   7 -39  79  60 -26
   7  37 -11  36  43  62  20-117 -79 -63 -48   9 -12 -39  83  44  -6  12  -7  68
  -1  -3 -14  -9 -34 -60-113  13-121  16-119  82  95 -19 111 -17 -27 -59 -92   0
  -3 -18  20 -14   2  26  97  41 149  37 125 -78 -64  41-140 -26  43  50  81 -34
   4  29 -91 103 118-112  36  26 -37-114 -58-164  44  -4 -80  72-128 122  93 -96
   5  15  84 -41 -76  26   7 100  -5 -49 227  62  90 -60  78 159-209 -31   1 -33
  -6 -37  59 -73 -84 -39-104 189  78 -26  78 118 -93 -35  -7 110 -93-153 -82 -33
   5  27 -25  53  48   5 106  -8 -65 -82-115-130  85 -46  20  30 -94  26 -79 -97
  -7 -27 -67  12  28 -79 -79 -66  65  59-164  44  21  14 -52 -53 210  43 -30 -97
   8  48 -68  84  93 -56 -76 -20-225-192-198 -19-143 -84 169 -39-192 148  74  58
   1   1  36 -31 -72  37   3  35  15  69-265 102 -10-118  97 -28 118 -11-111 159
  -7 -44  80 -87 -70  47 -47 162 112-113  -9 138-338-121  81 -28 -45-139-171-101
  -9 -39 -50  -4  20 -27 -30 -25 108 144-187 -34-144  69-196-118 183  31  64 138
   3  13  23  -1  20 100  69-129 166  66 175  14-179  28  67  84  90 -31  88  48
  -3 -13 -21  12  33  -8  28  37 118 -17-149  74-165  15 -99 -56 275  50-157 -45
   7  28  67 -24 -42 114  43 -69  79  29 116  51  -6-108  76 160  23 -42  26 122
  -7 -30 -40  -6  12 -41 -92  19   7  -5-256 115 -91  23 -34  59 161-126-151  95
   4  15  41  -2  -2  97 141 -52 172 115 232 -66  58 -54-206 239 116 -34 188 234
5112322.980174853
0.999987291695321 1.0
```

The determinants calculated between A, U and L are vastly different where normally $det(XY) = det(X)det(Y)$ for any $n \times n$ matrices X and Y. Therefore, it can be concluded that the small errors of each element in matrix A creates a big error for the total multiplication of the determinant. This hints to the fact that the condition number for this matrix is very high, which will be further explained in section f).

3.3 (d)

Choose now an exact solution, for instance $x_e =$ `numpy.ones(n)`, and compute the corresponding right hand side $b = Ax_e$.

[5]:
```
# Create x_e and compute A * x_e
x_e = np.ones(n)
b = A.dot(x_e)

# Print b
print("b = ")
print('\n'.join([''.join(['{:4}'.format(round(item)) for item in [row]])
      for row in b]))
```

```
b =
  27
 129
```

```
-138
  39
-371
 335
-237
 350
-233
-336
-225
-630
  27
-700
-106
 714
 -91
 650
-388
1236
```

3.4 (e)

Solve $Ax = b$ using `scipy.linalg.lu_factor` and `scipy.linalg.lu_solve` and compare the solution with the exact x_e.

[6]:
```
# Compute L and U using lu_factor and compute x with lu_solve
lu, piv = la.lu_factor(A)
x = la.lu_solve((lu, piv), b)

print("x = ")
print('\n'.join([''.join(['{:4}'.format(round(item,4)) for item in [row]])
      for row in x]), end="\n\n")

print("x_e = ")
print('\n'.join([''.join(['{:4}'.format(round(item,4)) for item in [row]])
      for row in x_e]), end="\n\n")

print("x_e - x = ")
print('\n'.join([''.join(['{:4}'.format(round(item,4)) for item in [row]])
      for row in x_e - x]))
```

```
x =
-290.8079
56.5383
9.7206
4.7978
0.2911
0.6628
1.1175
```

```
1.032
1.0031
1.0002
1.0002
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0

x_e =
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0

x_e - x =
291.8079
-55.5383
-8.7206
-3.7978
0.7089
0.3372
-0.1175
-0.032
-0.0031
-0.0002
-0.0002
```

```
-0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
-0.0
```

The difference $x_e - x$ illustrates the difference between every element of the vector. The lower elements are very similar, with differences only being small. However, the higher up the vector, the higher the differences, with the highest difference reaching 2606. This is extremely significant.

3.5 (f)

Explain the bad results by computing the condition number of A.

[7]:
```
condition = [la.norm(A, ord=x) * la.norm(la.inv(A), ord=x) for x in [1,2,np.
 ↪inf]]
print(condition)
```

```
[8.083777374926507e+19, 2.3358311624204534e+19, 6.3514050354154095e+19]
```

The condition number describes the amount of difference between input error and output error. Therefore, if a function has a very high condition number, you can conclude that from small input errors, large output errors arise. This result is observed in section c) and d) with differences being very significant. This is no surprise however, since the condition number calculated above reaches e+20.

4 Exercise 2

(N.B. this is a theory exercise.) Suppose we write a $(p+q)\times(p+q)$ matrix M in block form where A, B, C, D are respectively $p\times p, p\times q, q\times p$ and $q\times q$ matrices

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}.$$

4.1 (a)

Verify that

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} I_p & 0 \\ CA^{-1} & I_q \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & D - CA^{-1}B \end{bmatrix} \begin{bmatrix} I_p & A^{-1}B \\ 0 & I_q \end{bmatrix}.$$

$$\begin{bmatrix} I_p & 0 \\ CA^{-1} & I_q \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & D - CA^{-1}B \end{bmatrix} \begin{bmatrix} I_p & A^{-1}B \\ 0 & I_q \end{bmatrix} \tag{1}$$

$$= \begin{bmatrix} I_pA + 0 \cdot 0 & I_p \cdot 0 + 0 \cdot (D - CA^{-1}B) \\ CA^{-1}A + I_q \cdot 0 & CA^{-1} \cdot 0 + I_q \cdot (D - CA^{-1}B) \end{bmatrix} \begin{bmatrix} I_p & A^{-1}B \\ 0 & I_q \end{bmatrix} \tag{2}$$

$$= \begin{bmatrix} A & 0 \\ C & D - CA^{-1}B \end{bmatrix} \begin{bmatrix} I_p & A^{-1}B \\ 0 & I_q \end{bmatrix} \tag{3}$$

$$= \begin{bmatrix} A \cdot I_p + 0 \cdot 0 & AA^{-1}B + 0 \cdot I_q \\ C \cdot I_p + 0 \cdot (D - CA^{-1}B) & CA^{-1}B + I_q(D - CA^{-1}B) \end{bmatrix} \tag{4}$$

$$= \begin{bmatrix} A & B \\ C & CA^{-1}B + D - CA^{-1}B \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{5}$$

4.2 (b)

Describe how a system $Mx = b$, with x and b in $\mathbb{R}^{p+q}$, can be solved by applying matrix-vector products with C and B and solves with A and $(D - CA^{-1}B)$.

Write your answer, using LaTeX, in this box.

Define M_1, M_2 and M_3 as

$$M_1 = \begin{bmatrix} I_p & 0 \\ CA^{-1} & I_q \end{bmatrix} \tag{6}$$

$$M_2 = \begin{bmatrix} A & 0 \\ 0 & D - CA^{-1}B \end{bmatrix} \tag{7}$$

$$M_3 = \begin{bmatrix} I_p & A^{-1}B \\ 0 & I_q \end{bmatrix} \tag{8}$$

Then for solving $Mx = b$ it is known that $x = M^{-1}b = (M_1M_2M_3)^{-1}b = M_3^{-1}M_2^{-1}M_1^{-1}b$. For the inverses of block matrices, it is equal to the matrix of the inverses of its blocks, i.e.:

$$M_1^{-1} = \begin{bmatrix} I_p & 0 \\ -CA^{-1} & I_q \end{bmatrix} \tag{9}$$

$$M_2^{-1} = \begin{bmatrix} A^{-1} & 0 \\ 0 & (D - CA^{-1}B)^{-1} \end{bmatrix} \tag{10}$$

$$M_3^{-1} = \begin{bmatrix} I_p & -A^{-1}B \\ 0 & I_q \end{bmatrix}. \tag{11}$$

This means that:

$$x = \begin{bmatrix} I_p & -A^{-1}B \\ 0 & I_q \end{bmatrix} \begin{bmatrix} A^{-1} & 0 \\ 0 & (D - CA^{-1}B)^{-1} \end{bmatrix} \begin{bmatrix} I_p & 0 \\ -CA^{-1} & I_q \end{bmatrix} b \tag{12}$$

$$= \begin{bmatrix} I_p & -A^{-1}B \\ 0 & I_q \end{bmatrix} \begin{bmatrix} A^{-1}I_p & 0 \\ (D - CA^{-1}B)^{-1}(-CA^{-1}) & (D - CA^{-1}B)^{-1}I_q \end{bmatrix} b \tag{13}$$

$$= \begin{bmatrix} A^{-1} & A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} \\ -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix} b \tag{14}$$

Also, b can be defined as a block vector with length p called b_p and length q called b_q, then this creates a formula that can be solved if you want to solve for x: (Willen we dit doen?)

$$x = \begin{bmatrix} A^{-1} & A^{-1}B(D-CA^{-1}B)^{-1}CA^{-1} \\ -(D-CA^{-1}B)^{-1}CA^{-1} & (D-CA^{-1}B)^{-1} \end{bmatrix} \begin{bmatrix} b_p \\ b_q \end{bmatrix} = \begin{bmatrix} A^{-1}b_p - A^{-1}B(D-CA^{-1}B)^{-1}CA^{-1}b \\ -(D-CA^{-1}B)^{-1}CA^{-1}b_p + (D-CA^{-1}B \end{bmatrix}$$

4.3 (c)

What is the cost, to highest order, of LU-factorizing A and of computing and LU-factorizing $D - CA^{-1}B$?

Because A is a $p \times p$ matrix, creating all zeros outside of the diagonal in the first column takes $p-1$ operations. For the second column this takes $p-2$ operations, the third column $p-3$ operations, etc. This creates the lower triangular matrix L and its cost is $O(p)$. Similarly for U, but in the opisite direction which does not change the cost, you get a cost of $O(p)$ again. To factorize these is also $O(p)$ which means that the cost of LU-factorizing A is $O(p * p * p) = O(P^3)$.

The matrix D has size $q \times q$, C has size $q \times p$, A^{-1} has size $p \times p$ and B has size $p \times q$. This means that computing A^{-1} has a cost of $O(p^3)$, computing CA^{-1}, q rows with p elements are multiplied, making its cost $O(p^2q)$. Multiplying that by B has cost $O(p^2q)$ and subtracting D by that has a cost of $O(q)$. This means that computing $D - CA^{-1}B$ has a cost of $O(p^3) + O(pq) + O(pq) + O(q) = O(p^3)$. Now with the same logic as for LU-factorizing A and the knowledge that $D - CA^{-1}B$ has a size of $q \times q$ the cost of LU-factorizing $D - CA^{-1}B$ is $O(q^3)$. The total cost of computing and LU-factorizing $D - CA^{-1}B$ is $O(q^3) + O(p^3)$ which depends on which of p and q is larger, i.e. if $p > q$ the cost is $O(p^3)$, if $p < q$ the cost is $O(q^3)$ (if p and q are equal it does not matter).

Remark: Although in this case no savings were obtained, the decomposition above is very useful for solving linear systems with many zero coefficients, in other words where M is a sparse matrix. After applying a permutation of the indices such a matrix is written in the above form, where q is as small as possible and A is blockdiagonal, i.e. $A = \begin{bmatrix} E & O \\ O & F \end{bmatrix}$. This blockdiagonal form then causes big savings in computational cost. Moreover, the procedure can be applied recursively to E and F.