Name:	Suhas Kamath	**Assignment No:**	Assignment 1
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS284
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Linear Algebra
Date:	August 31, 2025	**Term:**	AUG 2025

Question 5

Calculated Quantities

Since the matrix **A** and the vectors are random, the output of each code run will vary. However, the quantities calculated by one code run is as follows (in tabular format):

p-value	**Maximum value of** $\|\|A\hat{x}\|\|_p$	**Actual value of** $\|\|\mathbf{A}\|\|_p$	Order of Relative Error
1	83.6969	83.6971	10^{-6}
2	10.5423	10.5423	10^{-10}
∞	4.3310	4.3301	10^{-4}

Although the program has output numbers to up to 14 decimal places, I have rounded them down to 4 decimal places, for better readability. Furthermore, I have not added the other norms of $\mathbf{A}\widehat{x}$, because they do not have a corresponding norm of $||\mathbf{A}||$. The other norms, which were calculated, are as follows:

p-value	**Maximum value of** $\|\|A\hat{x}\|\|_p$
3	6.1872
4	5.0428
5	4.5759
6	4.3540

Code and Algorithm

The Python code for the calculation is as follows:

```
import numpy as np

#generate random matrix A
A = np.random.randn(100,2)

#this array keeps track of maximum 1-norm, 2-norm etc. of the vector Ax
#max_norm[0] represents maximum 1-norm of Ax,
#max_norm[1] represents maximum 2-norm of Ax, etc
max_norm = [-1,-1,-1,-1,-1,-1,-1]

for _ in range(1000):
    #Generate random x vector
    #The np.random.randn() method returns a matrix of 2x1 instead of a vector
    #the squeeze function is used to convert the 2x1 matrix to a vector
    x = np.squeeze(np.random.randn(2,1))

    #take 1-norm to 6-norm
    #x is converted to unit vector for the particular norm
    #i.e if 1-norm of Ax is to be calculated, x is normalised by its 1-norm, and so on
    for p in range(1,7):
        #Calculate the relevant norm of x and normalise x accordingly
        pnorm_x = np.linalg.norm(x,ord=p)
        unit_x = [element/pnorm_x for element in x]

        #Calculate Ax
        #Again the @ operator produces a matrix, which needs to be squeezed to a vector
        Ax = np.squeeze(A @ unit_x)

        #Update the max_norm
        max_norm[p-1]=max(max_norm[p-1], np.linalg.norm(Ax, ord=p))
```

```
        #take infinity norm
        #This section is outside the loop because p cannot
        #be suddenly assigned to infinity
        pnorm_x = np.linalg.norm(x,ord=np.inf)
        unit_x = [element/pnorm_x for element in x]
        Ax = np.squeeze(A @ unit_x)
        max_norm[6]=max(max_norm[6], np.linalg.norm(Ax, ord=np.inf))


print("1-norm of A =", np.linalg.norm(A, ord=1))
print("Max 1-norm of Ax =", max_norm[0], end="\n")

print("2-norm of A =", np.linalg.norm(A, ord=2))
print("Max 2-norm of Ax =", max_norm[1], end="\n")

print("infinity-norm of A =", np.linalg.norm(A, ord=np.inf))
print("Max infinity-norm of Ax =", max_norm[6])

print("Remaining norms of x: ", max_norm[2:6])
```

The program steps are as follows:

1. A random matrix with dimensions 100×2 is created. This is the **A** matrix.
2. A list called max_norm is created with 7 elements. This list shall store the maximum of the norms. The first element of the list, i.e. max_norm[0], contains the maximum 1-norm of **Ax**. The second element stores the maximum 2-norm, and so on. The last element shall store the ∞-norm of **Ax**.
3. A for loop is present, which shall generate 1000 random **x** vectors, such that $\mathbf{x} \in \mathbb{R}^{100}$. This vector is then normalised. Its norm is calculated and updated to the max_norm list (if it is the maximum).
4. Lastly, the 1-norm, 2-norm and ∞-norm of **A** is calculated, to compare with the maximum values obtained from the iterations above.

Sample Output

Since the matrix **A** and the vectors are random, the output of each code run will vary. However, one such output is as follows:

```
1-norm of A = 77.14076257113005
Max 1-norm of Ax = 77.12304412473856
2-norm of A = 9.568208854655976
Max 2-norm of Ax = 9.568208654815267
infinity-norm of A = 4.49453960570396
Max infinity-norm of Ax = 4.487772423831027
Remaining norms of x:  [np.float64(5.627964886251967), np.float64(4.679666918956469),
np.float64(4.341368483515785), np.float64(4.219939313675693)]
```

Observations

From the above table, we can devise the following:

$$||A\hat{x}||_p^m \leq ||A||_p^{(m,n)}, \forall x \in \mathbb{R}^n, ||\hat{x}||_p = 1$$

From this, we can also infer that there is one $\widehat{x_{max}}$ such that the above inequality is converted to equality, i.e.

$$||A\widehat{x_{max}}||_p^m = ||A||_p^{(m,n)}, ||\widehat{x_{max}}||_p = 1$$