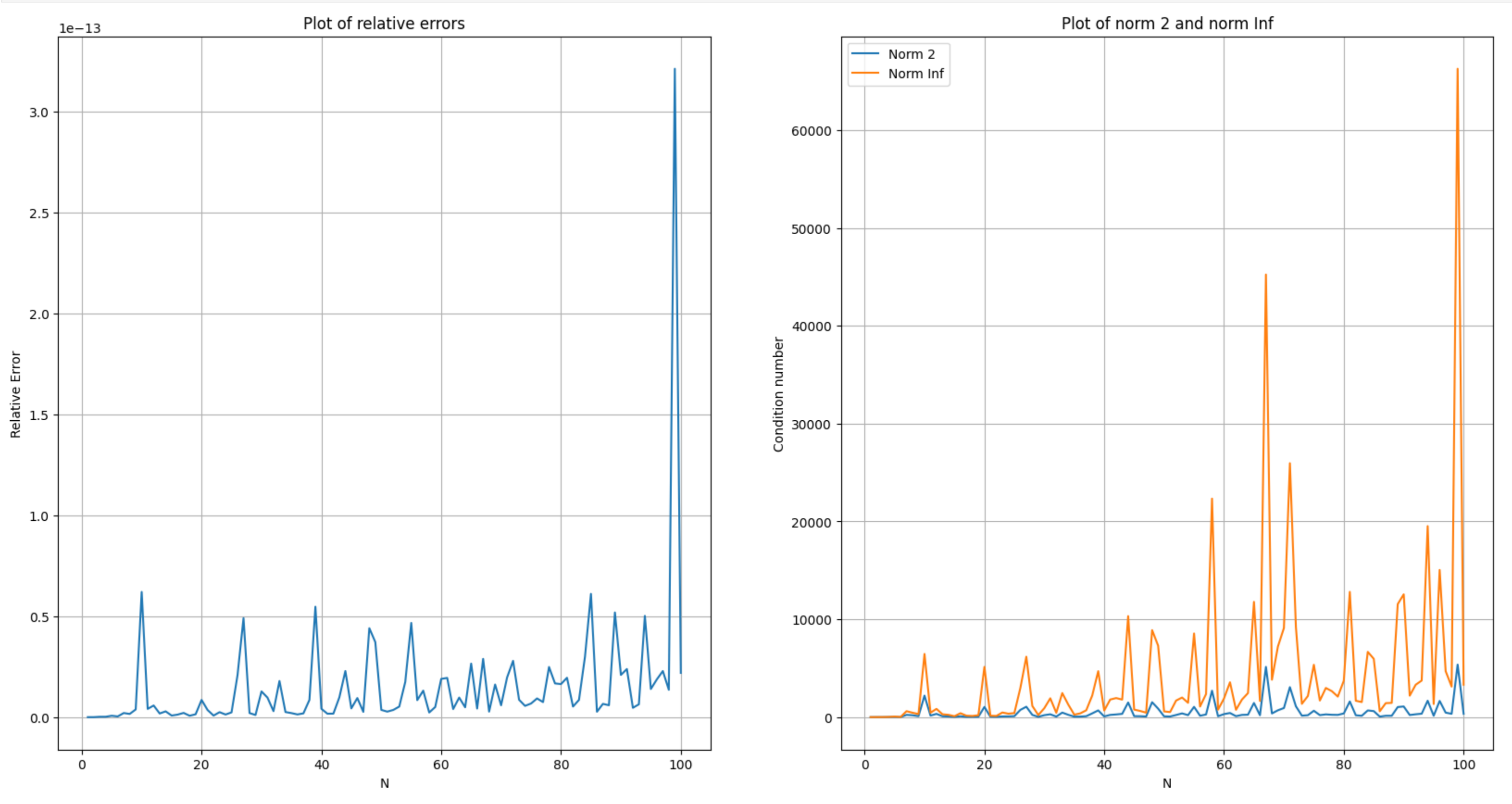
```python
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
```

Esercizio 1

```python
n = 10

A = np.random.randn(n,n)
xtrue = np.ones(n)

b = np.dot(A, xtrue)

cond2 = np.linalg.cond(A, 2)
print("Conditioning number using norm 2: ", cond2)
if cond2 < 10e5:
    print("A is not ill conditionated with norm 2 because K(A) is less than 10^5")
else:
    print("A is ill conditionated with norm 2")

condInf = np.linalg.cond(A, np.Inf)
print("Conditioning number using norm Inf: ", condInf)
if condInf < 10e5:
    print("A is not ill conditionated with norm Inf because K(A) is less than 10^5")
else:
    print("A is ill conditionated with norm Inf")

x = np.linalg.solve(A,b)
print("Solution of the linear system Ax=b: ", x)

absolute_err = np.linalg.norm(xtrue - x, 2)
relative_err = absolute_err / np.linalg.norm(xtrue, 2)

print("Absolute Error: ", absolute_err)
print("Relative Error: ", relative_err)
```

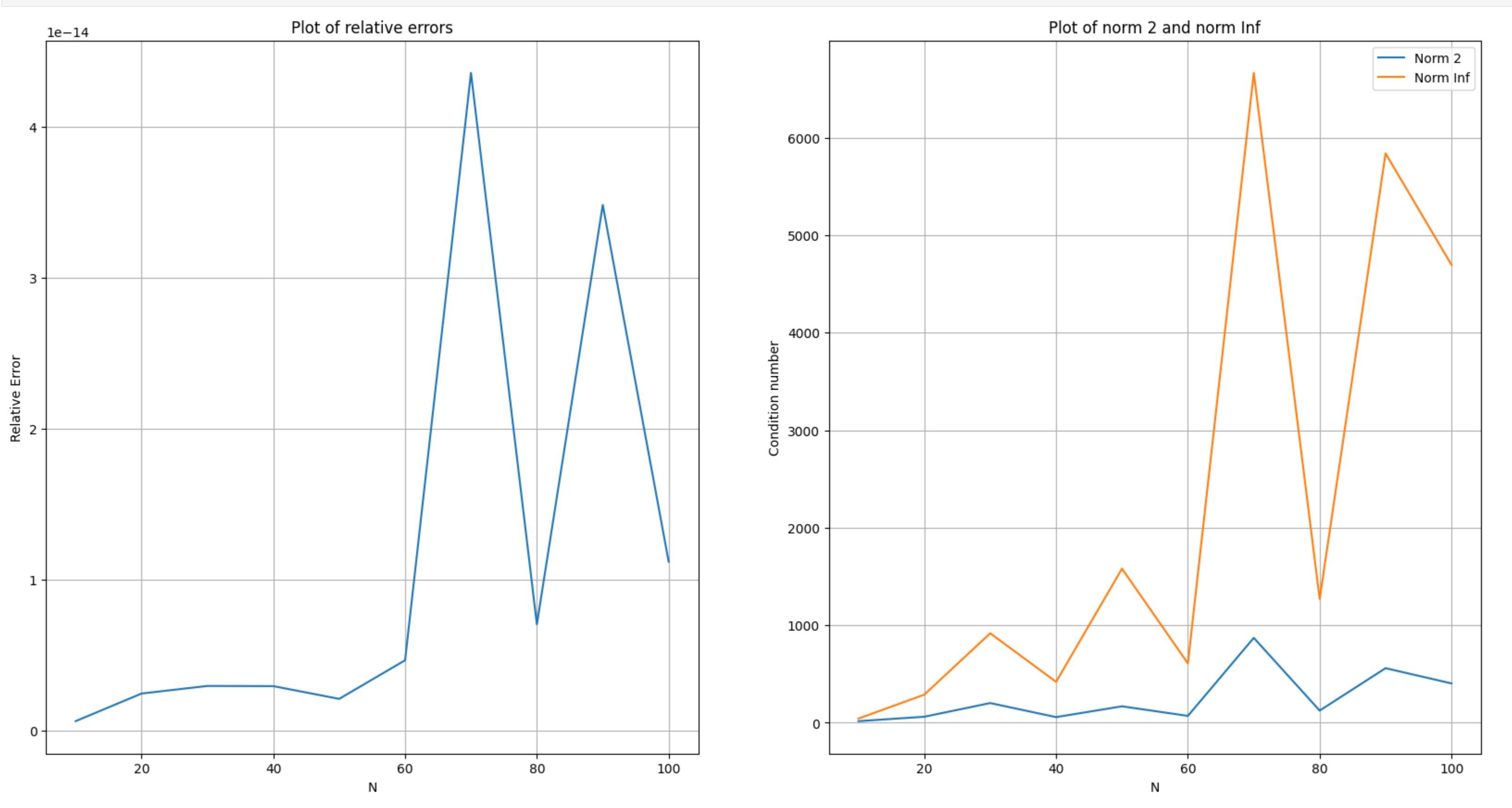
```
Conditioning number using norm 2:  98.12963022216229
A is not ill conditionated with norm 2 because K(A) is less than 10^5
Conditioning number using norm Inf:  345.08591011330316
A is not ill conditionated with norm Inf because K(A) is less than 10^5
Solution of the linear system Ax=b:  [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Absolute Error:  3.1185214200286946e-15
Relative Error:  9.861630619313312e-16
```

```python
N = 100
array_relErr = []
array_norm2 = []
array_normInf = []

x_plot = np.arange(1, N+1, 1)

for n in range (1, N+1):
    A = np.random.randn(n,n)
    xtrue = np.ones(n)
    b = np.dot(A, xtrue)
    x = np.linalg.solve(A,b)
    cond2 = np.linalg.cond(A, 2)
    condInf = np.linalg.cond(A, np.Inf)
    absolute_err = np.linalg.norm(xtrue - x, 2)
    relative_err = absolute_err / np.linalg.norm(xtrue, 2)
    array_relErr.append(relative_err)
    array_norm2.append(cond2)
    array_normInf.append(condInf)

plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
plt.plot(x_plot, array_relErr)
plt.title("Plot of relative errors")
plt.xlabel("N")
plt.ylabel("Relative Error")
plt.grid()

plt.subplot(1,2,2)
plt.plot(x_plot, array_norm2)
plt.plot(x_plot, array_normInf)
plt.legend(["Norm 2", "Norm Inf"])
plt.title("Plot of norm 2 and norm Inf")
plt.xlabel("N")
plt.ylabel("Condition number")
plt.grid()

plt.show()
```

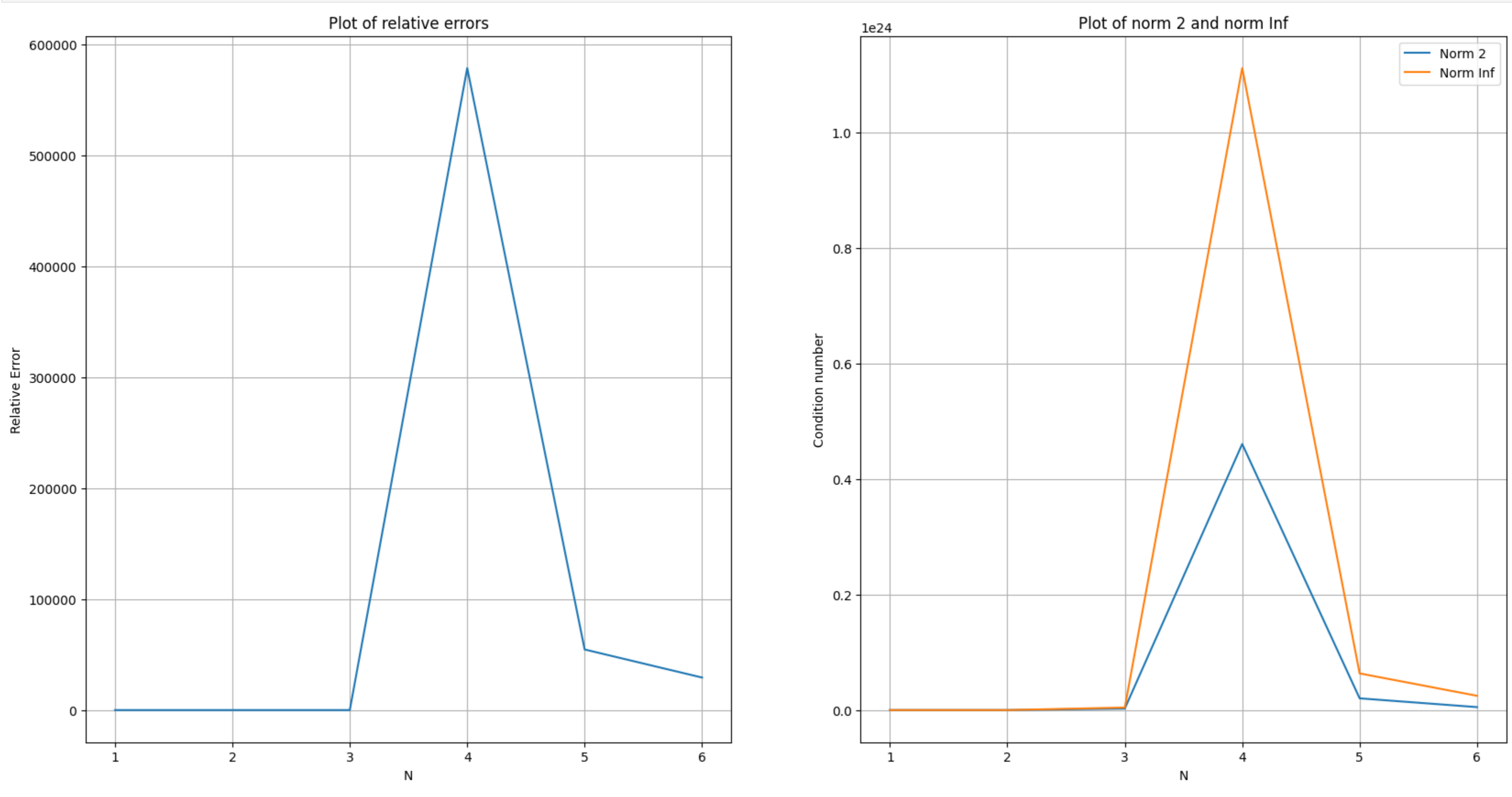


Esercizio 2

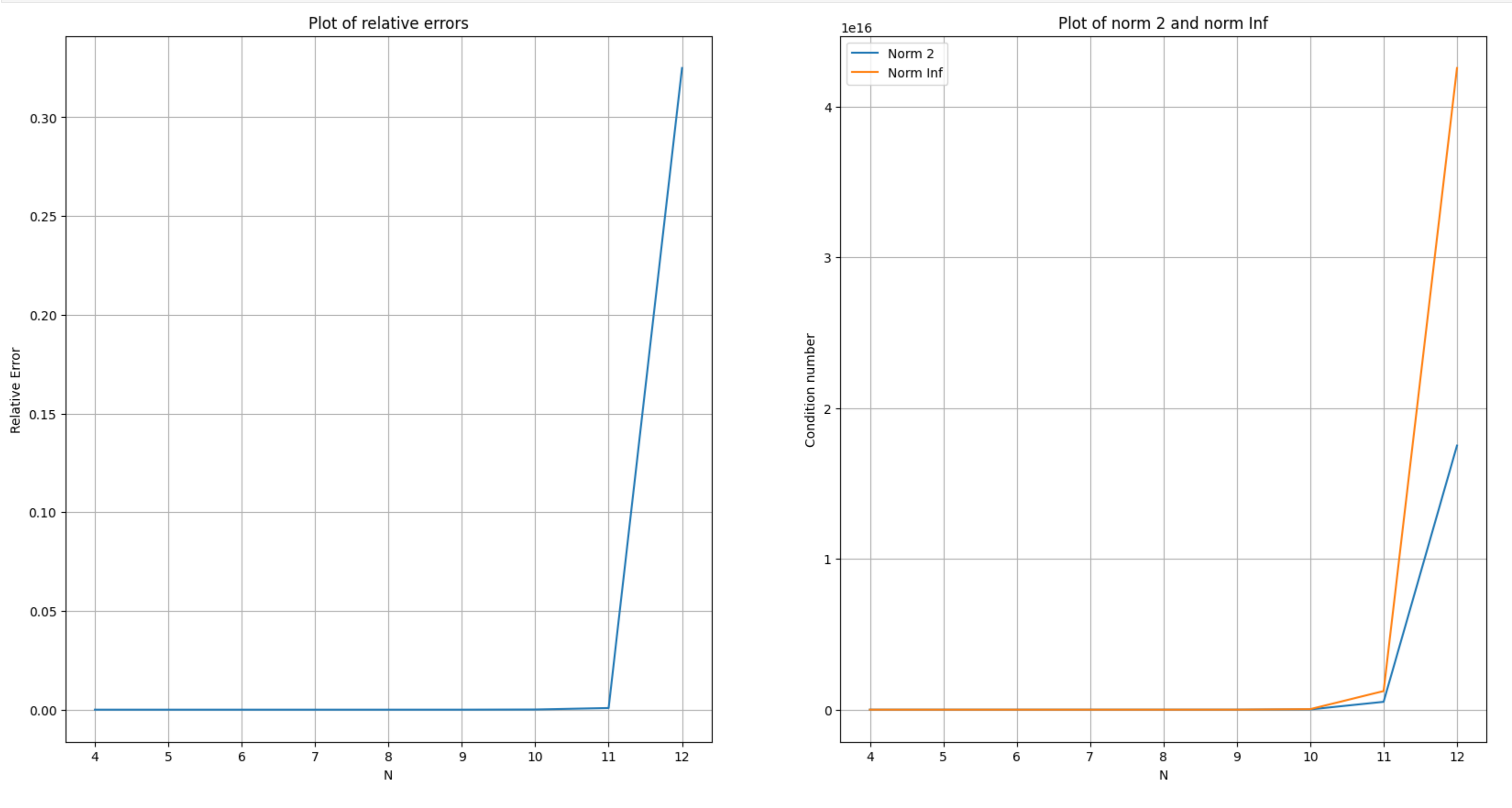
```python
########################## RANDOM BY 10 ##########################
array_relErr = []
array_norm2 = []
array_normInf = []

for n in range (10, 101, 10):
    A = np.random.randn(n,n)
    xtrue = np.ones(n)
    b = np.dot(A, xtrue)
    cond2 = np.linalg.cond(A, 2)
    condInf = np.linalg.cond(A, np.Inf)
    x = np.linalg.solve(A,b)
    absolute_err = np.linalg.norm(xtrue - x, 2)
    relative_err = absolute_err / np.linalg.norm(xtrue, 2)
    array_relErr.append(relative_err)
    array_norm2.append(cond2)
    array_normInf.append(condInf)

x_plot = np.arange(10, 101, 10)

plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
plt.plot(x_plot, array_relErr)
plt.title("Plot of relative errors")
plt.xlabel("N")
plt.ylabel("Relative Error")
plt.grid()

plt.subplot(1,2,2)
plt.plot(x_plot, array_norm2)
plt.plot(x_plot, array_normInf)
plt.legend(["Norm 2", "Norm Inf"])
plt.title("Plot of norm 2 and norm Inf")
plt.xlabel("N")
plt.ylabel("Condition number")
plt.grid()

plt.show()
```




```python
########################## VANDER ##########################
array_relErr = []
array_norm2 = []
array_normInf = []

for n in range (5, 31, 5):
    x = np.arange(1, n+1, 1)
    A = np.vander(x, n)
    xtrue = np.ones(n)
    b = np.dot(A, xtrue)
    cond2 = np.linalg.cond(A, 2)
    condInf = np.linalg.cond(A, np.Inf)
    x = np.linalg.solve(A,b)
    absolute_err = np.linalg.norm(xtrue - x, 2)
    relative_err = absolute_err / np.linalg.norm(xtrue, 2)
    array_relErr.append(relative_err)
    array_norm2.append(cond2)
    array_normInf.append(condInf)

x_plot = np.arange(1, 7, 1)

plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
plt.plot(x_plot, array_relErr)
plt.title("Plot of relative errors")
plt.xlabel("N")
plt.ylabel("Relative Error")
plt.grid()

plt.subplot(1,2,2)
plt.plot(x_plot, array_norm2)
plt.plot(x_plot, array_normInf)
plt.legend(["Norm 2", "Norm Inf"])
plt.title("Plot of norm 2 and norm Inf")
plt.xlabel("N")
plt.ylabel("Condition number")
plt.grid()

plt.show()
```




```python
########################## HILBERT ##########################
array_relErr = []
array_norm2 = []
array_normInf = []

for n in range (4, 13, 1):
    A = sp.linalg.hilbert(n)
    xtrue = np.ones(n)
    b = np.dot(A, xtrue)
    cond2 = np.linalg.cond(A, 2)
    condInf = np.linalg.cond(A, np.Inf)
    x = np.linalg.solve(A,b)
    absolute_err = np.linalg.norm(xtrue - x, 2)
    relative_err = absolute_err / np.linalg.norm(xtrue, 2)
    array_relErr.append(relative_err)
    array_norm2.append(cond2)
    array_normInf.append(condInf)

x_plot = np.arange(4, 13, 1)

plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
plt.plot(x_plot, array_relErr)
plt.title("Plot of relative errors")
plt.xlabel("N")
plt.ylabel("Relative Error")
plt.grid()

plt.subplot(1,2,2)
plt.plot(x_plot, array_norm2)
plt.plot(x_plot, array_normInf)
plt.legend(["Norm 2", "Norm Inf"])
plt.title("Plot of norm 2 and norm Inf")
plt.xlabel("N")
plt.ylabel("Condition number")
plt.grid()

plt.show()
```



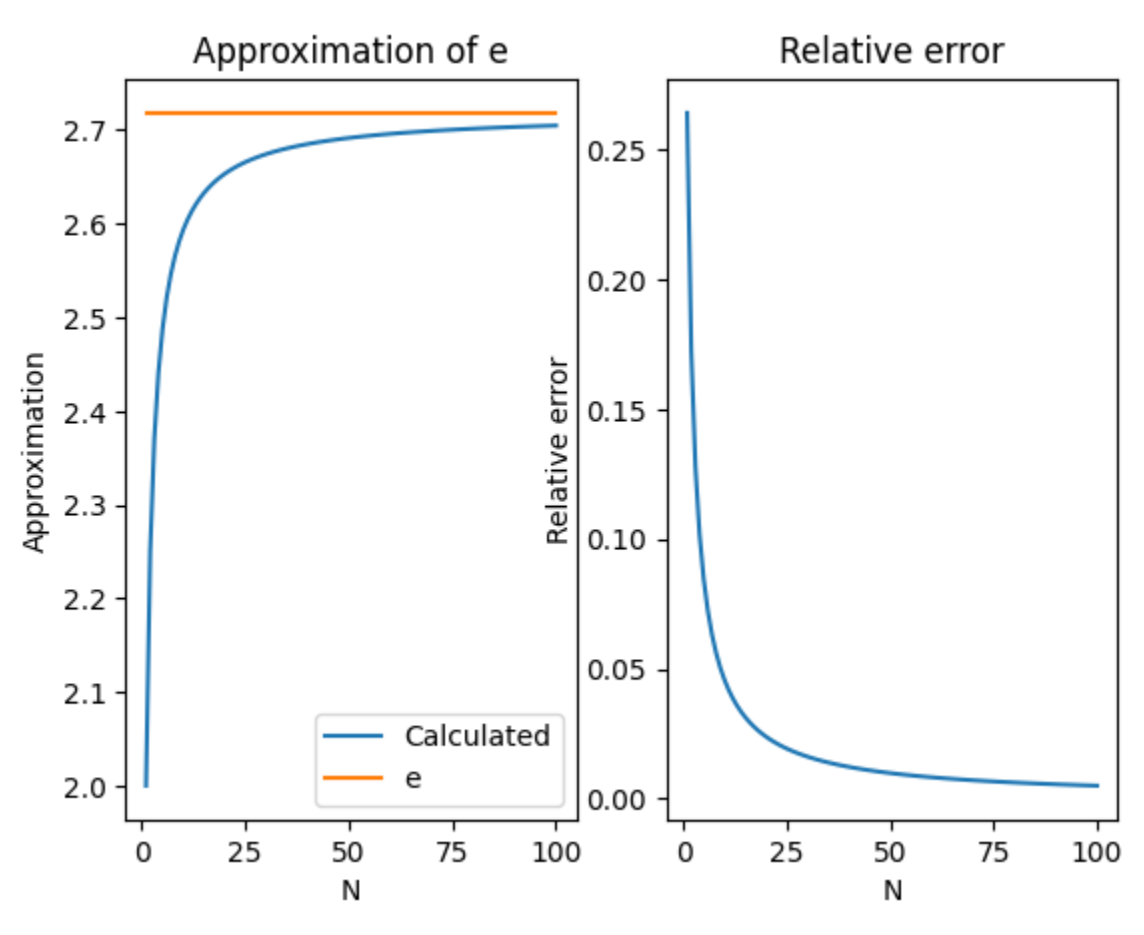
FLOATING POINTS

```python
########################## 1 ##########################
oldeps = 1
eps = 1
while(1 + eps > 1):
    oldeps = eps
    eps = eps / 2

print("Calculated eps:", oldeps)
```

```
Calculated eps: 2.220446049250313e-16
```

```python
########################## 2 ##########################

N = 100
x_plot = np.arange(1, N+1, 1)
y = []
err = []

for n in range(1, N+1, 1):
    approx = (1+1/n)**n
    y.append(approx)
    err.append(np.abs(np.e - approx)/np.e)

plt.subplot(1,2,1)
plt.plot(x_plot, y)
plt.plot(x_plot, [np.e]*N)
plt.title("Approximation of e")
plt.xlabel("N")
plt.ylabel("Approximation")
plt.legend(["Calculated", "e"])
plt.subplot(1,2,2)
plt.plot(x_plot, err)
plt.title("Relative error")
plt.xlabel("N")
plt.ylabel("Relative error")
```

```
Text(0, 0.5, 'Relative error')
```




```python
########################## 3 ##########################
A = np.array([[4,2],[1,3]])
B = np.array([[4,2],[2,1]])

eigA = np.linalg.eigvals(A)
eigB = np.linalg.eigvals(B)

print("Rank A: ", np.linalg.matrix_rank(A))
print("Eigenvalues A: ", eigA)
print("Rank B: ", np.linalg.matrix_rank(B))
print("Eigenvalues B: ", eigB)
```

```
Rank A:  2
Eigenvalues A:  [5. 2.]
Rank B:  1
Eigenvalues B:  [5. 0.]
```