DIRECT METHODS FOR THE SOLUTION OF LINEAR SYSTEMS 1

```
In [ ]: import numpy as np
        import scipy
        import matplotlib.pyplot as plt
```

a)

```
In [ ]: # Define matrix A nxn and vector X_true of n ones
        n = 10

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

        # Compute b = A x_true
        b = A @ x_true
```

b)

```
In [ ]: # Computation of the condition number of A in 2-norm
        c_2 = np.linalg.cond(A, 2)
        print(f'k(A) 2-norm: {c_2}')

        # Computation of the condition number of A in Inf-norm
        c_inf = np.linalg.cond(A, np.Inf)
        print(f'k(A) inf-norm: {c_inf}')
```

```
k(A) 2-norm: 411.1588693540122
k(A) inf-norm: 1261.0472757581674
```

c)

```
In [ ]: # Solution of the linear system Ax = b
        x_sol = np.linalg.solve(A, b)
```

d)

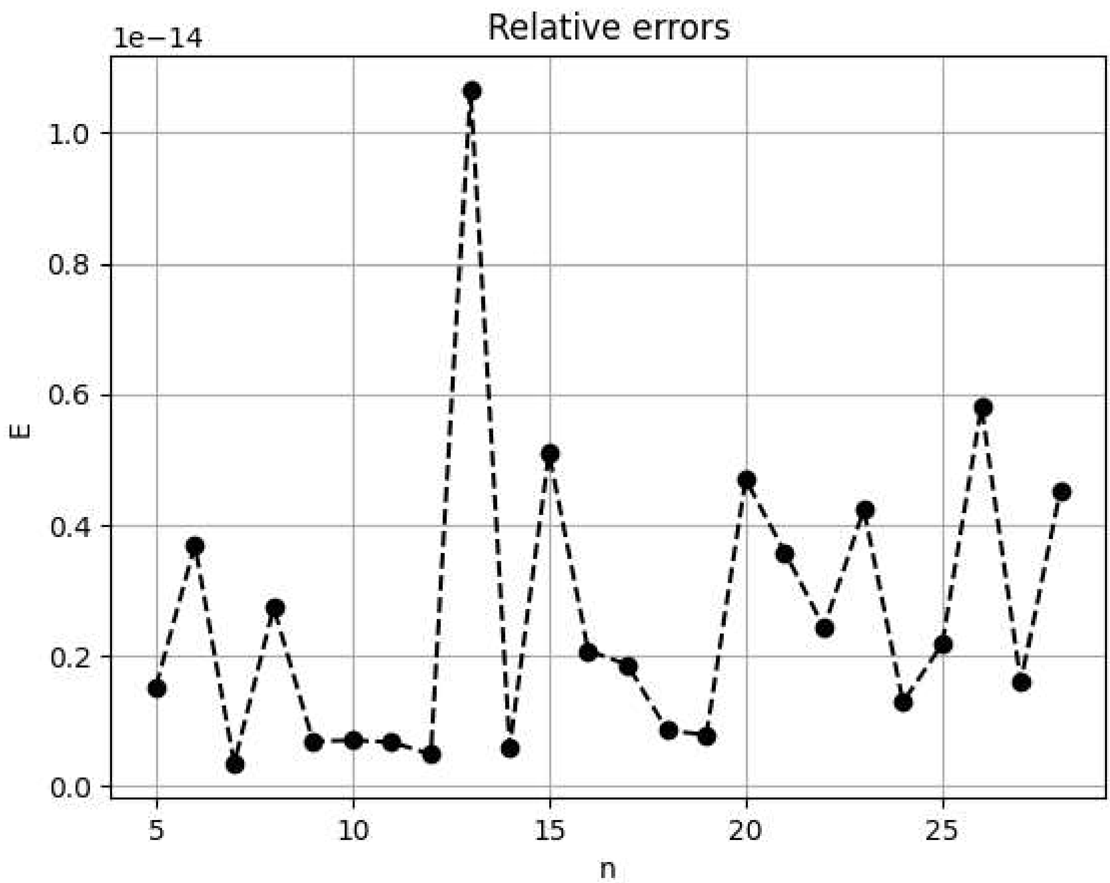
```
In [ ]: # Computation of the relative error between x_sol and x_true
        E = np.linalg.norm(x_sol - x_true) / np.linalg.norm(x_true)
        print(f'Relative error E = {E}')
```

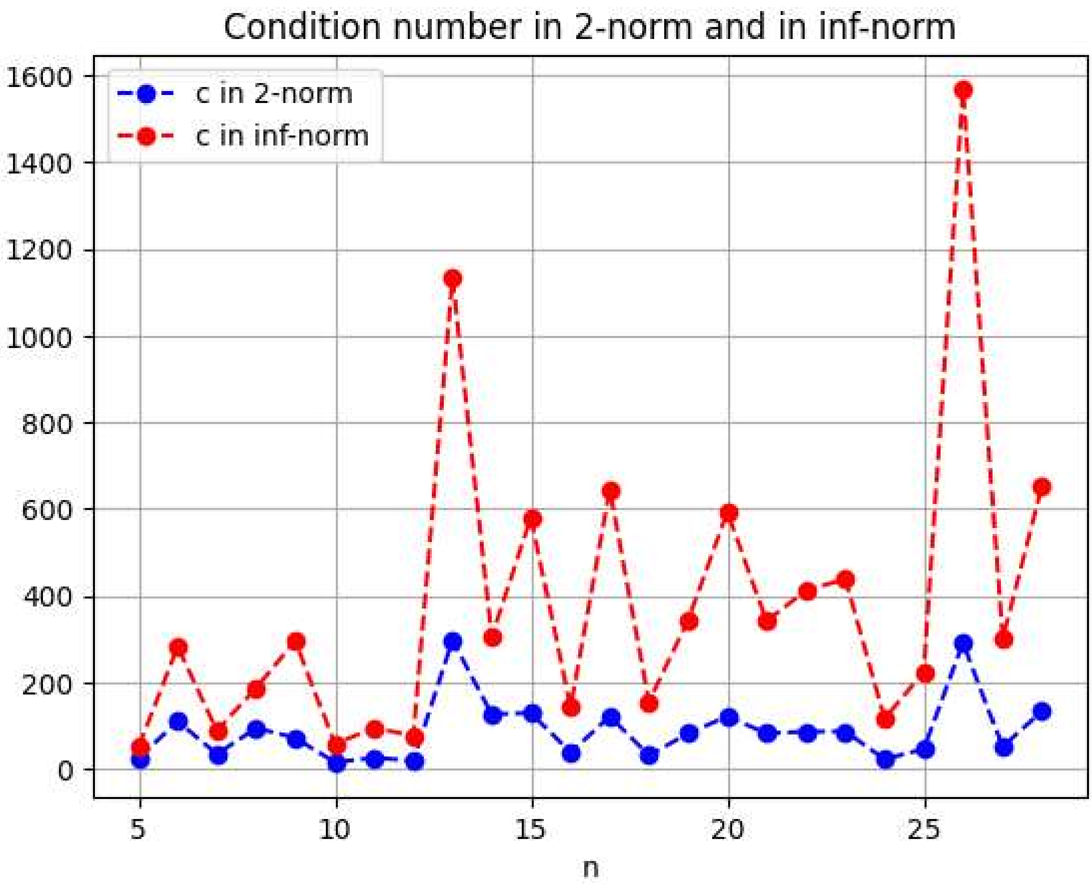
```
Relative error E = 8.315563043070463e-16
```

e)

```
In [ ]: # Plot a graph with the relative errors as function of n and
        # the condition number in 2-norm and inf-norm
        ns = range(5, 29, 1)

        Es = []
        c_2s = []
        c_infs = []

        for d in ns:
            A = np.random.randn(d, d)
            x_true = np.ones((d,))
```

```
    b = A @ x_true
    c_2s.append(np.linalg.cond(A, 2))
    c_infs.append(np.linalg.cond(A, np.Inf))
    x_sol = np.linalg.solve(A, b)
    Es.append(np.linalg.norm(x_sol - x_true) / np.linalg.norm(x_true))


plt.plot(ns, Es, 'ko--')
plt.grid()
plt.title('Relative errors')
plt.ylabel('E')
plt.xlabel('n')
plt.show()
plt.plot(ns, c_2s, 'bo--')
plt.plot(ns, c_infs, 'ro--')
plt.grid()
plt.title('Condition number in 2-norm and in inf-norm')
plt.xlabel('n')
plt.legend(['c in 2-norm', 'c in inf-norm'])
plt.show()
```





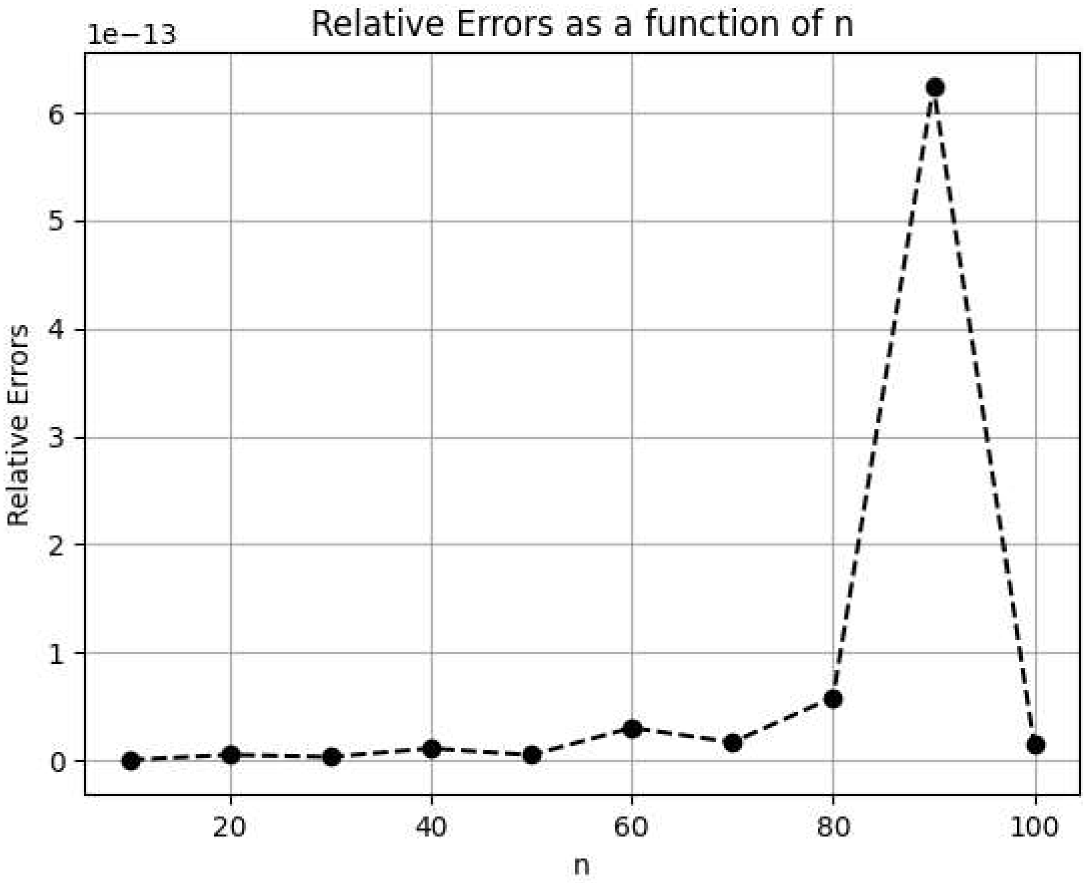
DIRECT METHODS FOR THE SOLUTION OF LINEAR SYSTEMS 2

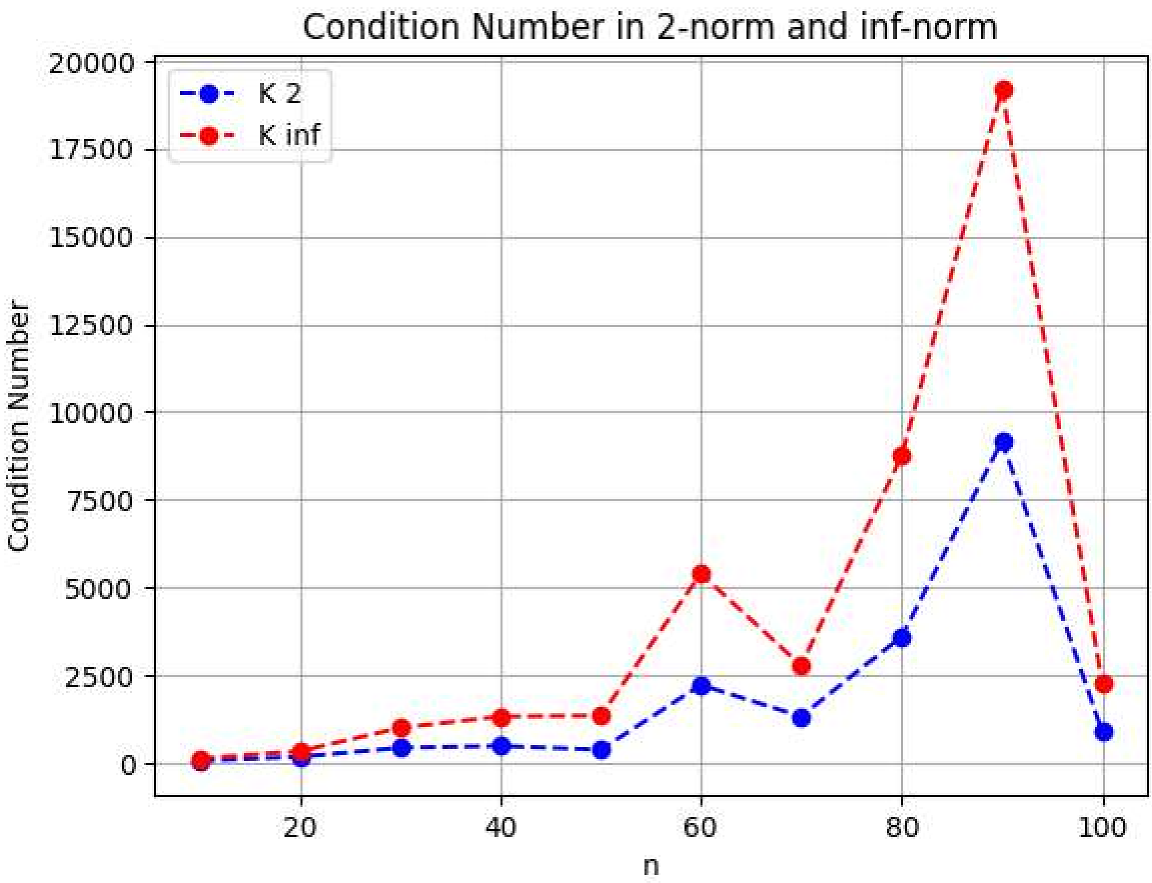
a)

In [ ]:
```
# Define a random matrix with size varying
n_sizes = np.arange(10, 110, 10)
print(n_sizes)
c_2 = []
c_inf = []
E = []

for n in n_sizes:
    x_true = np.ones((n,))
    A = np.random.rand(n, n)
    b = A @ x_true
    c_2.append(np.linalg.cond(A, 2))
    c_inf.append(np.linalg.cond(A, np.Inf))
    x_sol = np.linalg.solve(A, b)
    E.append(np.linalg.norm(x_sol - x_true) / np.linalg.norm(x_true))

plt.figure(1)
plt.plot(n_sizes, E, 'ko--')
plt.xlabel('n')
plt.ylabel('Relative Errors')
plt.title('Relative Errors as a function of n')
plt.grid(True)
plt.show()

plt.figure(2)
plt.plot(n_sizes, c_2, 'bo--', label='K 2')
plt.plot(n_sizes, c_inf, 'ro--', label='K inf')
```

```python
plt.xlabel('n')
plt.ylabel('Condition Number')
plt.title('Condition Number in 2-norm and inf-norm')
plt.legend()
plt.grid(True)
plt.show()
```

```
[ 10  20  30  40  50  60  70  80  90 100]
```

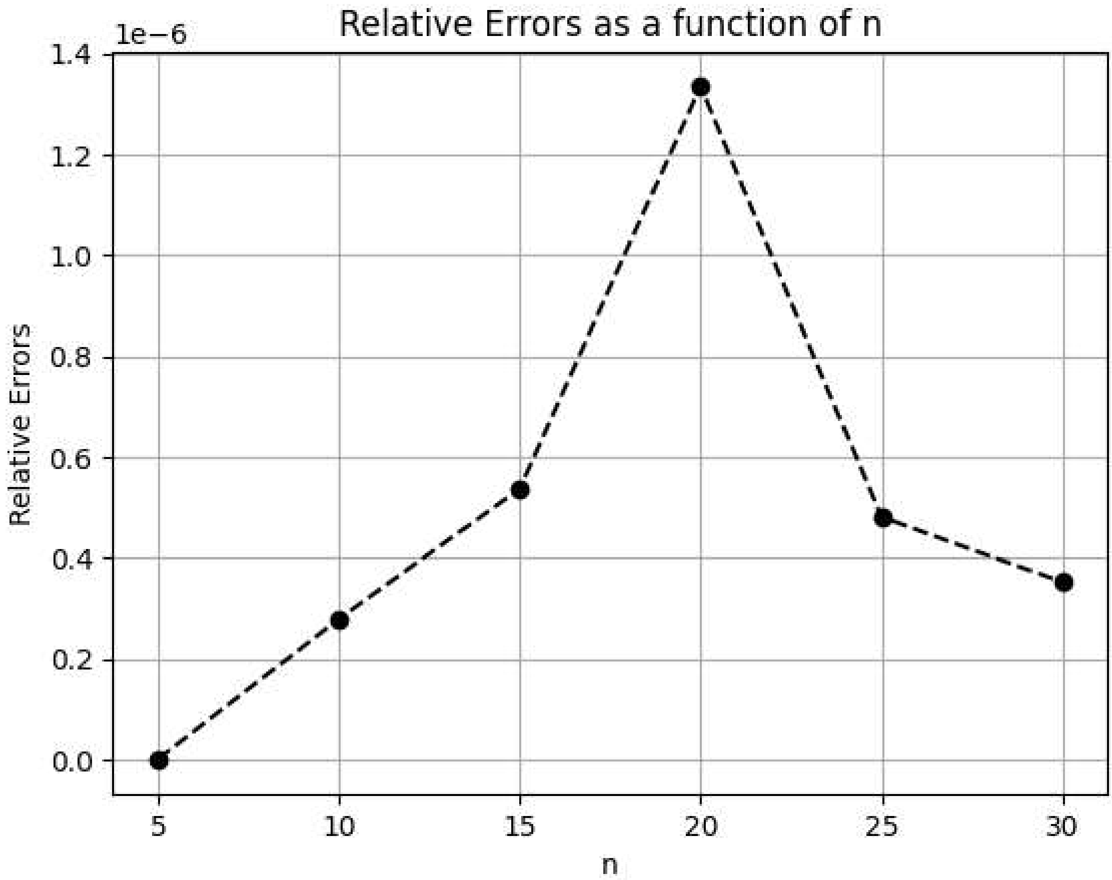


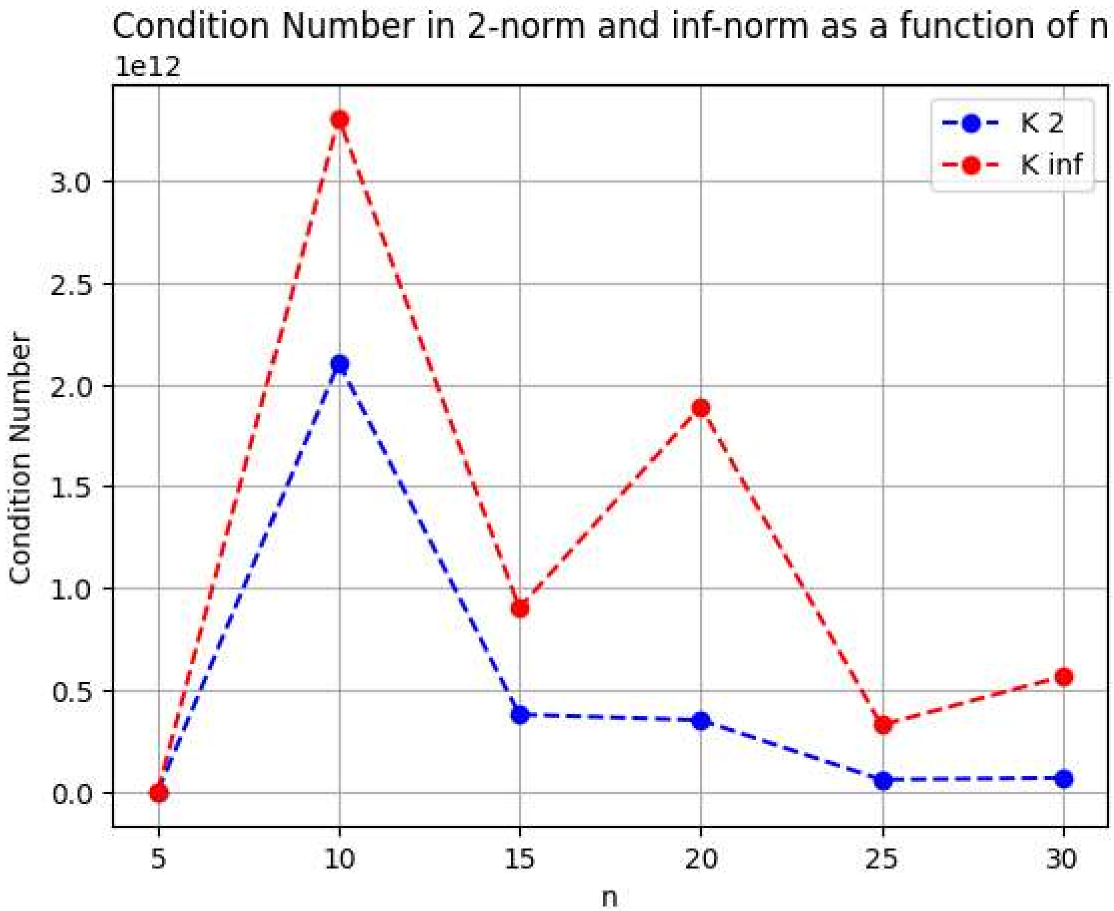


b)

```
In [ ]: # Test the program with Vendermonde matrix of variable dimension
        # wrt the vector x = {1, 2, 3, ...., n}
        n_sizes = np.arange(5, 35, 5)
        c_2 = []
        c_inf = []
        E = []

        for n in n_sizes:
            x_true = np.ones((n,))
            vander_vector = np.arange(1, n+1, 1)
            A = np.vander(vander_vector)
            b = A @ x_true
            c_2.append(np.linalg.cond(A, 2))
            c_inf.append(np.linalg.cond(A, np.Inf))
            x_sol = np.linalg.solve(A, b)
            E.append(np.linalg.norm(x_sol - x_true) / np.linalg.norm(x_true))

        plt.figure(1)
        plt.plot(n_sizes, E, 'ko--')
        plt.xlabel('n')
        plt.ylabel('Relative Errors')
        plt.title('Relative Errors as a function of n')
        plt.grid(True)
        plt.show()

        plt.figure(2)
        plt.plot(n_sizes, c_2, 'bo--', label='K 2')
        plt.plot(n_sizes, c_inf, 'ro--', label='K inf')
        plt.xlabel('n')
        plt.ylabel('Condition Number')
```

```
plt.title('Condition Number in 2-norm and inf-norm as a function of n')
plt.legend()
plt.grid(True)
plt.show()
```

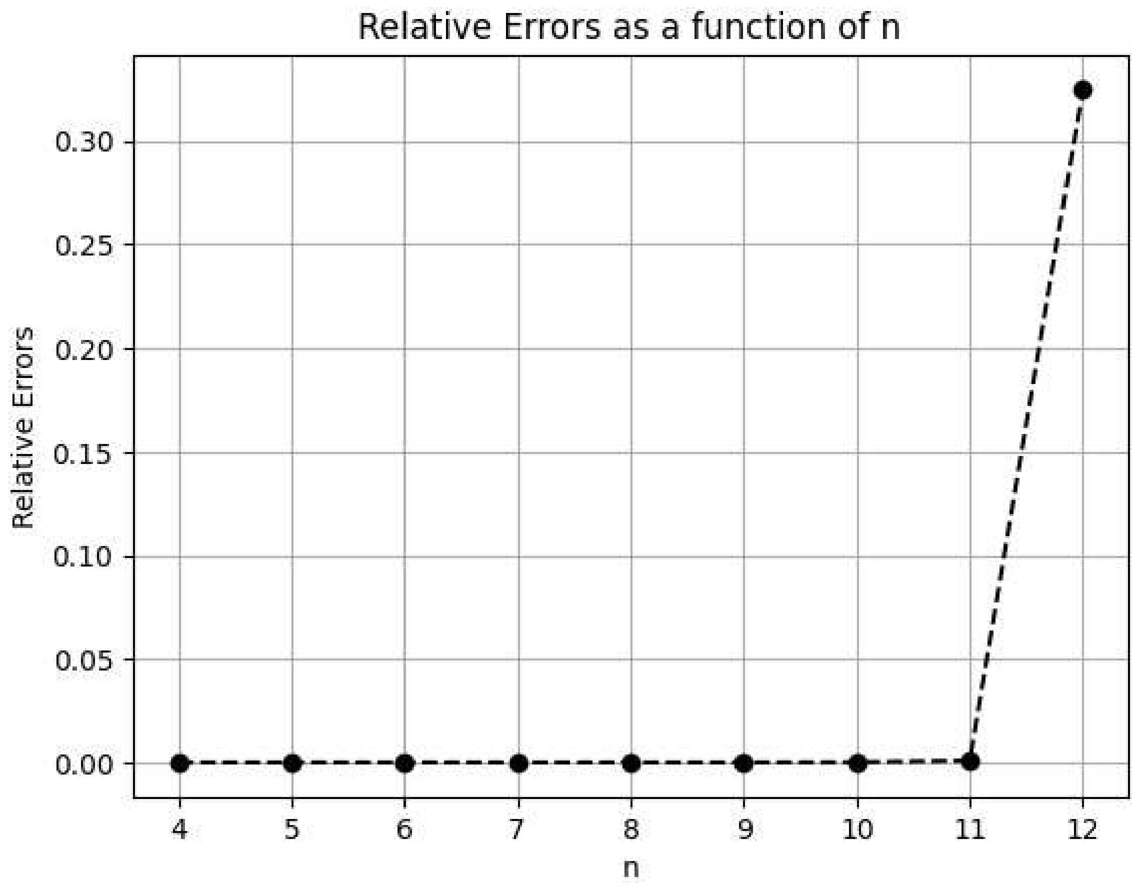




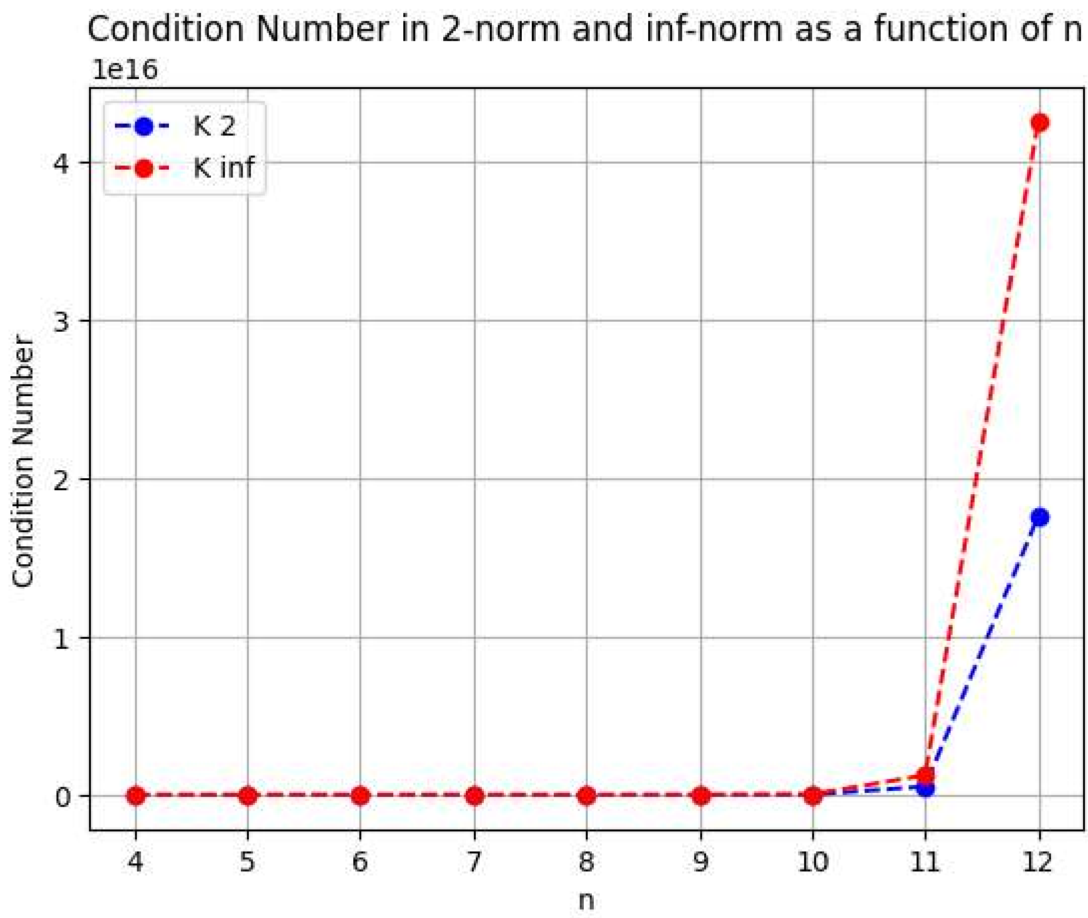
c)

In [ ]:
```
# Test the program with the Hilbert matrix of dimension n = {4, 5, 6, ..., 12}
n_sizes = np.arange(4, 13, 1)
print(n_sizes)
c_2 = []
c_inf = []
E = []


for n in n_sizes:
    x_true = np.ones((n,))
    A = scipy.linalg.hilbert(n)
    b = A @ x_true
    c_2.append(np.linalg.cond(A, 2))
    c_inf.append(np.linalg.cond(A, np.Inf))
    x_sol = np.linalg.solve(A, b)
    E.append(np.linalg.norm(x_sol - x_true) / np.linalg.norm(x_true))

plt.figure(1)
plt.plot(n_sizes, E, 'ko--')
plt.xlabel('n')
plt.ylabel('Relative Errors')
plt.title('Relative Errors as a function of n')
plt.grid(True)
plt.show()

plt.figure(2)
plt.plot(n_sizes, c_2, 'bo--',label='K 2')
plt.plot(n_sizes, c_inf, 'ro--', label='K inf')
plt.xlabel('n')
plt.ylabel('Condition Number')
plt.title('Condition Number in 2-norm and inf-norm as a function of n')
plt.legend()
plt.grid(True)
plt.show()
```

```
[ 4  5  6  7  8  9 10 11 12]
```



Relative Errors as a function of n
Relative Errors
0.30
0.25
0.20
0.15
0.10
0.05
0.00
4
5
6
7
8
9
10
11
12
n



Condition Number in 2-norm and inf-norm as a function of n
1e16
K 2
K inf
Condition Number
4
3
2
1
0
4
5
6
7
8
9
10
11
12
n