TMA4215 - Project 1 - Emil Myhre

Problem 1:

The induced matrix norm $\|\cdot\|_2$ is given as $\|A\|_2 = \sqrt{p(A^TA)}$, where p is the spectral radius of a given matrix, meaning the maximum absolute value of its eigenvalues.

In order to proove $\|uv^T\|_2 = \|u\|_2 \cdot \|v\|_2$

we use the definition of the induced matrix

$$\|uv^T\|_2 = \sqrt{p((uv^T)^T uv^T)} = \sqrt{p(vu^T uv^T)}$$

From the definition

$$\|A\|_2 = \sqrt{p(A^TA)}$$

we can write

$$\sqrt{\|u\|_2^2 p(vv^T)} = \|u\|_2 \cdot \sqrt{p(vv^T)}$$

By solving the equation

$$vv^Tx = x\lambda$$

Substituting x for v:

$$vv^Tv = v\lambda$$

$$v^Tv = \lambda$$

Now we can substitute $p(vv^T)$ with $\|v\|_2$

Thus

$$\|uv^T\|_2 = \|u\|_2 \cdot \|v\|_2$$

Problem 2:

From the definition of condition numbers, we have:

$$k_2(A) = \|A\|_2 \cdot \|A^{-1}\|_2$$

Given a symmetrical matrix, we have $A^T = A$, which leads to:

$$\|A\|_2 = \sqrt{p(A^TA)} = \sqrt{p(A^2)} = \sqrt{(p(A))^2} = p(A)$$

where $p(A)$ is defined as $|\lambda_{max}|$

Similarly, we have for $\|A^{-1}\|_2$:

$$\|A^{-1}\|_2 = \sqrt{p((A^{-1})^TA^{-1})} = \sqrt{p(A^{-1})^2)} = \sqrt{(p(A^{-1}))^2} = p(A^{-1})$$

The eigenvalues of A^{-1} will naturally be $\frac{1}{\lambda}$, thus the spectral radius, $p(A^{-1})$, is given as $|\frac{1}{\lambda_{min}}|$

Finally, this leads to $k_2(A) = \|A\|_2 \cdot \|A^{-1}\|_2 =| \lambda_{max} | \cdot | \frac{1}{\lambda_{min}} |=| \frac{\lambda_{max}}{\lambda_{min}}|$

$$k_2(A) =| \frac{\lambda_{max}}{\lambda_{min}}|$$

Problem 3:

Given the matrix nxn matrix A, such that $a_{ij} = 1$ for j $\geq$ i and $a_{ij} = 0$ for $j < i$, we can find A^{-1} by performing row reduction to the following matrix

$$\begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{vmatrix}$$

consisting of A and I. By reducing the left side to I, the right side of the matrix will convert to A^{-1}, because $A \cdot A^{-1} = I$.

$$\begin{vmatrix} 1 & 0 & 0 & 0 & \mathbf{1} & \mathbf{-1} & \mathbf{0} & \mathbf{0} \\ 0 & 1 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{-1} & \mathbf{0} \\ 0 & 0 & 1 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{-1} \\ 0 & 0 & 0 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{vmatrix}$$

As we see, A^{-1} can be expressed as an nxn matrix with $a_{ij} = -1$ for $j = i + 1$, $a_{ij} = 1$ for $j = i$, and else, $a_{ij} = 0$.

To determine $|\frac{\lambda_{max}}{\lambda_{min}}|$, I have implemented a simple python code to calculate the eigenvalues of the matrix.

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt
import math

n = 13
matrix = np.zeros([n,n])
for i in range(n):
    for j in range(n):
        if (i == j):
            matrix[i][j] = 1
        if (j == i + 1):
            matrix[i][j] = 1
        if (j > i):
            matrix[i][j] = 1
print(np.linalg.eigvals(matrix))
```

```
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.]
```

Clearly, all the eigenvalues are equal to 1. Thus, $|\frac{\lambda_{max}}{\lambda_{min}} |= 1$ as well.

To determine the condition numbers $k_1(A)$ and $k_\infty(A)$, we use the definitions of condition numbers.

$$k_1(A) = \|A\|_1 \cdot \|A^{-1}\|_1$$

Where $\|A\|_1$ is defined as $\max\limits_{1\leq j\leq n} \sum_{i=1}^n | a_{ij}|$

From this definition we can easily calculate

$\|A\|_1 = n$ and $\|A^{-1}\|_1 = 2$

Which yields $k_1(A) = 2n$

Similarily, we have

$$k_\infty(A) = \|A\|_\infty \cdot \|A^{-1}\|_\infty$$

where $\|A\|_\infty$ is defined as $\max\limits_{1\leq i\leq n} \sum_{j=1}^n | a_{ij}|$

Equally, we have $\|A\|_\infty = n$ and $\|A^{-1}\|_\infty = 2$

Which gives Which yields $k_\infty(A) = 2n$

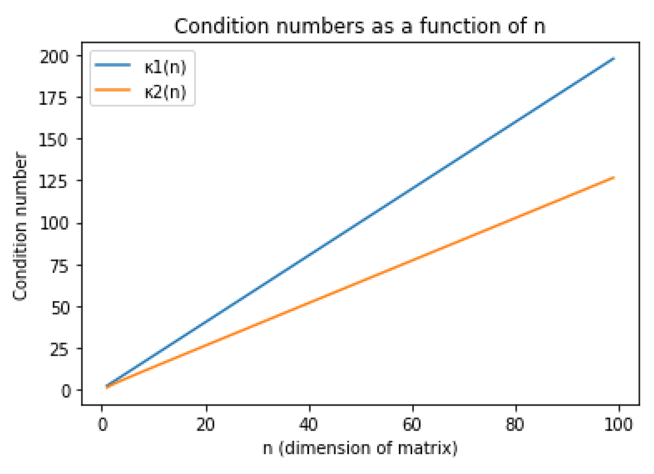
In the following python code, the condition number k_2 is investigated numerically.

In [10]:
```
import numpy as np
import matplotlib.pyplot as plt
import math

k1 = []
k2 = []
list = []

for n in range(1,100):
    list.append(n)
    k1.append(2*n)
    matrix = np.zeros([n,n])
    inverseMatrix = np.zeros([n,n])
    for i in range(n):
        for j in range(n):
            if (i == j):
                matrix[i][j] = 1
                inverseMatrix[i][j] = 1
            if (j == i + 1):
                inverseMatrix[i][j] = -1
                matrix[i][j] = 1
            if (j > i):
                matrix[i][j] = 1
    matrixProduct = np.dot(matrix,np.transpose(matrix))
    inverseMatrixProduct = np.dot(inverseMatrix,np.transpose(inverseMatrix))
    eigenvalues1 = np.linalg.eigvals(matrixProduct)
    eigenvalues2 = np.linalg.eigvals(inverseMatrixProduct)
    for eigenvalue in eigenvalues1:
        eigenvalue = abs(eigenvalue)
    for eigenvalue in eigenvalues2:
        eigenvalue = abs(eigenvalue)
    specrad1 = np.sqrt(max(eigenvalues1))
    specrad2 = np.sqrt(max(eigenvalues2))
    k_2 = specrad1 * specrad2
    k2.append(k_2)

plt.plot(list,k1,label='κ1(n)')
plt.plot(list,k2,label='κ2(n)')
plt.title("Condition numbers as a function of n")
plt.xlabel('n (dimension of matrix)')
plt.ylabel('Condition number')
plt.legend()
plt.show()
```



By studiyng the graph, we observe that $k_2(A)$ is approximately linear, at least not constant equal to 1. Thus we can conclude that the relation $k_2(A) =| \frac{\lambda_{max}}{\lambda_{min}}|$ does not hold for non-symmetric matrices.

Problem 4:

In this problem I am going to analyze the bound $\frac{\|\delta x\|}{\|x\|} \leq \frac{k(A)}{1-k(A)\cdot\frac{\|\delta A\|}{\|A\|}}(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|})$

By simplifying the problem, we set $\|\delta A\| = 0$, which yields the bound

$$\frac{\frac{\|\delta x\|}{\|x\|}}{\frac{\|\delta b\|}{\|b\|}} \leq k(A)$$

In the following code I am calculating "kappa*est", $\frac{\frac{\|\delta x\|}{\|x\|}}{\frac{\|\delta b\|}{\|b\|}}$, numerically, and comparing this to "K2", $k{2}(A)$.*

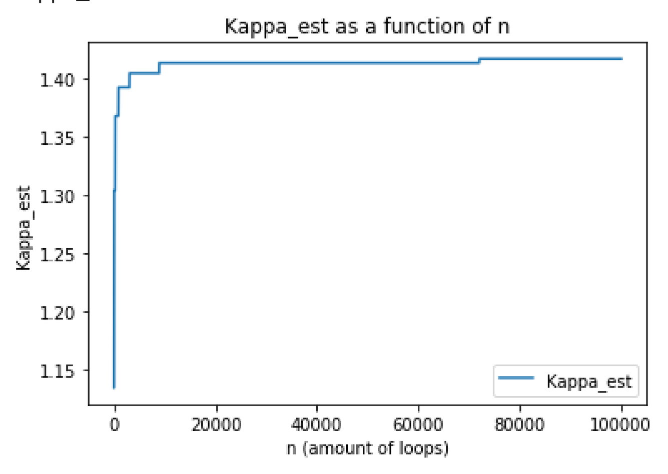
In [9]:
```
import numpy as np
import matplotlib.pyplot as plt
import math

n = 13
matrix = np.zeros([n,n])  #matrisen A
inverseMatrix = np.zeros([n,n])
for i in range(n):
    for j in range(n):
        if (i == j):
            matrix[i][j] = 1
            inverseMatrix[i][j] = 1
        if (j == i + 1):
            inverseMatrix[i][j] = -1
            matrix[i][j] = 1
        if (j > i):
            matrix[i][j] = 1
b = []
for i in range(n):
    b.append(1- 2*np.random.random_sample())
b = np.asarray(b)

x = np.linalg.solve(matrix,b) #finner løsingen på likningen Ax = b
matrix_norm = np.linalg.norm(matrix,2) #normen til matrisen A
invMatrix_norm = np.linalg.norm(inverseMatrix,2) #normen til inversA
b_norm = np.linalg.norm(b,2) #normen til vektoren B
x_norm = np.linalg.norm(x,2)
k2 = matrix_norm * invMatrix_norm #condition number k2(A)

steps = []
kappa_est_list = []
kappa_est = 0
NEXPS = 100000
for k in range(0,NEXPS):
    delta_b = []
    for i in range(n):
        delta_b.append((1- 2*np.random.random_sample())/10)
    delta_b = np.asarray(delta_b)
    delta_x = np.linalg.solve(matrix,delta_b)
    delta_x_norm = np.linalg.norm(delta_x)
    delta_b_norm = np.linalg.norm(delta_b)
    kappa_c = (delta_x_norm / x_norm) / (delta_b_norm / b_norm)
    kappa_est = max(kappa_est,kappa_c)
    kappa_est_list.append(kappa_est)
    steps.append(k)
print("K2: ",k2)
print("Kappa_est:", kappa_est)
plt.plot(steps,kappa_est_list,label="Kappa_est")
plt.title("Kappa_est as a function of n")
plt.xlabel('n (amount of loops)')
plt.ylabel('Kappa_est')
plt.legend()
plt.show()
```

```
K2:  17.0821442999
Kappa_est: 1.41717198644
```



We observe that the bound is satisfied, because of the randomly generated numbers, the result does vary. However, usually I get to approximately 10% of the bound.

In [ ]: