exercise1 (Score: 14.0 / 14.0)

1. Test cell (Score: 2.0 / 2.0)
2. Test cell (Score: 1.0 / 1.0)
3. Test cell (Score: 1.0 / 1.0)
4. Test cell (Score: 1.0 / 1.0)
5. Test cell (Score: 1.0 / 1.0)
6. Test cell (Score: 3.0 / 3.0)
7. Test cell (Score: 2.0 / 2.0)
8. Task (Score: 3.0 / 3.0)

Lab 5

1. 提交作業之前，建議可以先點選上方工具列的**Kernel**，再選擇**Restart & Run All**，檢查一下是否程式跑起來都沒有問題，最後記得儲存。
2. 請先填上下方的姓名(name)及學號(stduent_id)再開始作答，例如：

```
name = "我的名字"
student_id= "B06201000"
```

3. 演算法的實作可以參考lab-5 (https://yuanyuyuan.github.io/itcm/lab-5.html), 有任何問題歡迎找助教詢問。
4. **Deadline: 12/11(Wed.)**

In [1]:

```
name = "馬宗儀"
student_id = "b06201006"
```

Exercise 1

An $m \times m$ *Hilbert matrix* H_m has entries $h_{ij} = 1/(i+j-1)$ for $1 \le i,j \le m$, and so it has the form

$$\left [

$$\begin{matrix} 1 & 1/2 & 1/3 & \dots \\ 1/2 & 1/3 & 1/4 & \dots \\ 1/3 & 1/4 & 1/5 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{matrix}$$

\right].$$

In [2]:

```
import numpy as np
from numpy import linalg as LA
import matplotlib.pyplot as plt
```

Part 1

Generate the Hilbert matrix of order m, for $m = 2, 3, \dots, 12$.

For each m, compute the condition number of H_m, *ie* , in p-norm for $p = 1$ and 2, and make a plot of the results.

Part 1.1

Define the function of Hilbert matrix

In [3]:

(Top)

```
def hilbert_matrix(m):
    '''
    Return:
        2D np.array, the Hildert Matrix of order m
    '''
    # ===== 請實做程式 =====
    U=[]
    for i in range(1,m+1,1):
        A=[]
        for j in range(1,m+1,1):
            A.append(1/(i+j-1))
        U.append(A)
    return U
    # ====================
```

Test your function.

In [4]:

hilbert_matrix (Top)

```
print('H_2:\n', hilbert_matrix(2))
### BEGIN HIDDEN TESTS
assert np.mean(np.array(hilbert_matrix(3)) - np.array([[1, 1/2, 1/3], [1/2, 1/3, 1/4], [1/3, 1/4, 1/5]]))
< 1e-7
### END HIDDEN TESTS
```

```
H_2:
 [[1.0, 0.5], [0.5, 0.3333333333333333]]
```

Part 1.2

Collect all Hilbert matrices into the list `H_m` for m = 2, 3, ..., 12.

In [5]:

(Top)

```
H_m = []
# ===== 請實做程式 =====
for m in range(2,13,1):
    H_m.append(hilbert_matrix(m))
# ====================
```

Check your Hilbert matrix list.

In [6]:

hilbert_matrices (Top)

```
for i in range(len(H_m)):
    print('H_%d:' % (i+2))
    print(H_m[i])
    print()
### BEGIN HIDDEN TESTS
error = 0
for m in range(2, 13):
    error += LA.norm(hilbert_matrix(m) - np.array([[1/(i + j + 1) for j in range(m)] for i in range(m)]))
assert error < 1e-16
### END HIDDEN TESTS
```

```
H_2:
[[1.0, 0.5], [0.5, 0.3333333333333333]]

H_3:
[[1.0, 0.5, 0.3333333333333333], [0.5, 0.3333333333333333, 0.25], [0.3333333333333333, 0.25,
0.2]]

H_4:
[[1.0, 0.5, 0.3333333333333333, 0.25], [0.5, 0.3333333333333333, 0.25, 0.2], [0.3333333333333
333, 0.25, 0.2, 0.16666666666666666], [0.25, 0.2, 0.16666666666666666, 0.14285714285714285]]

H_5:
[[1.0, 0.5, 0.3333333333333333, 0.25, 0.2], [0.5, 0.3333333333333333, 0.25, 0.2, 0.1666666666
6666666], [0.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0.14285714285714285], [0.25, 0
.2, 0.16666666666666666, 0.14285714285714285, 0.125], [0.2, 0.16666666666666666, 0.1428571428
5714285, 0.125, 0.1111111111111111]]

H_6:
[[1.0, 0.5, 0.3333333333333333, 0.25, 0.2, 0.16666666666666666], [0.5, 0.3333333333333333, 0.
25, 0.2, 0.16666666666666666, 0.14285714285714285], [0.3333333333333333, 0.25, 0.2, 0.1666666
6666666666, 0.14285714285714285, 0.125], [0.25, 0.2, 0.16666666666666666, 0.14285714285714285
, 0.125, 0.1111111111111111], [0.2, 0.16666666666666666, 0.14285714285714285, 0.125, 0.111111
1111111111, 0.1], [0.16666666666666666, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1,
0.09090909090909091]]

H_7:
[[1.0, 0.5, 0.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0.14285714285714285], [0.5, 0
.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125], [0.3333333333
333333, 0.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125, 0.1111111111111111], [0.2
5, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1], [0.2, 0.16
666666666666666, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091], [
0.16666666666666666, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091
, 0.08333333333333333], [0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.0909090909090
9091, 0.08333333333333333, 0.07692307692307693]]

H_8:
[[1.0, 0.5, 0.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125],
[0.5, 0.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125, 0.11111
11111111111], [0.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125
, 0.1111111111111111, 0.1], [0.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125, 0.11
111111111111111, 0.1, 0.09090909090909091], [0.2, 0.16666666666666666, 0.14285714285714285, 0.
125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.08333333333333333], [0.16666666666666666
, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.083333333333333
33, 0.07692307692307693], [0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090
909091, 0.08333333333333333, 0.07692307692307693, 0.07142857142857142], [0.125, 0.11111111111
11111, 0.1, 0.09090909090909091, 0.08333333333333333, 0.07692307692307693, 0.0714285714285714
2, 0.06666666666666667]]

H_9:
[[1.0, 0.5, 0.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125, 0
.1111111111111111], [0.5, 0.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0.1428571428571
4285, 0.125, 0.1111111111111111, 0.1], [0.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0
.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091], [0.25, 0.2, 0.16666
666666666666666, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.083
33333333333333], [0.2, 0.16666666666666666, 0.14285714285714285, 0.125, 0.1111111111111111, 0
.1, 0.09090909090909091, 0.08333333333333333, 0.07692307692307693], [0.16666666666666666, 0.1
4285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.08333333333333333, 0
.07692307692307693, 0.07142857142857142], [0.14285714285714285, 0.125, 0.1111111111111111, 0.
1, 0.09090909090909091, 0.08333333333333333, 0.07692307692307693, 0.07142857142857142, 0.0666
6666666666667], [0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.08333333333333333, 0.
07692307692307693, 0.07142857142857142, 0.06666666666666667, 0.0625], [0.1111111111111111, 0.
1, 0.09090909090909091, 0.08333333333333333, 0.07692307692307693, 0.07142857142857142, 0.0666
6666666666667, 0.0625, 0.05882352941176471]]
```

H_10:
[[1.0, 0.5, 0.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125, 0
.1111111111111111, 0.1], [0.5, 0.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0.14285714
285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091], [0.3333333333333333, 0.25, 0
.2, 0.16666666666666666, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.0909090909090
9091, 0.08333333333333333], [0.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125, 0.11
11111111111111, 0.1, 0.09090909090909091, 0.08333333333333333, 0.07692307692307693], [0.2, 0.
16666666666666666, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091,
0.08333333333333333, 0.07692307692307693, 0.07142857142857142], [0.16666666666666666, 0.14285
714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.08333333333333333, 0.076
92307692307693, 0.07142857142857142, 0.06666666666666667], [0.14285714285714285, 0.125, 0.111
1111111111111, 0.1, 0.09090909090909091, 0.08333333333333333, 0.07692307692307693, 0.07142857
142857142, 0.06666666666666667, 0.0625], [0.125, 0.1111111111111111, 0.1, 0.09090909090909091
, 0.08333333333333333, 0.07692307692307693, 0.07142857142857142, 0.06666666666666667, 0.0625,
0.058823529411764705], [0.1111111111111111, 0.1, 0.09090909090909091, 0.08333333333333333, 0.
07692307692307693, 0.07142857142857142, 0.06666666666666667, 0.0625, 0.058823529411764705, 0.
05555555555555555], [0.1, 0.09090909090909091, 0.08333333333333333, 0.07692307692307693, 0.07
142857142857142, 0.06666666666666667, 0.0625, 0.058823529411764705, 0.05555555555555555, 0.05
263157894736842]]

H_11:
[[1.0, 0.5, 0.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125, 0
.1111111111111111, 0.1, 0.09090909090909091], [0.5, 0.3333333333333333, 0.25, 0.2, 0.16666666
666666666, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.083333
33333333333], [0.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125
, 0.1111111111111111, 0.1, 0.09090909090909091, 0.08333333333333333, 0.07692307692307693], [0
.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.0909090
9090909091, 0.08333333333333333, 0.07692307692307693, 0.07142857142857142], [0.2, 0.166666666
66666666, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.0833333
3333333333, 0.07692307692307693, 0.07142857142857142, 0.06666666666666667], [0.16666666666666
666, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.083333333333
33333, 0.07692307692307693, 0.07142857142857142, 0.06666666666666667, 0.0625], [0.14285714285
714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.08333333333333333, 0.076923076
92307693, 0.07142857142857142, 0.06666666666666667, 0.0625, 0.058823529411764705], [0.125, 0.
1111111111111111, 0.1, 0.09090909090909091, 0.08333333333333333, 0.07692307692307693, 0.07142
857142857142, 0.06666666666666667, 0.0625, 0.058823529411764705, 0.05555555555555555], [0.111
1111111111111, 0.1, 0.09090909090909091, 0.08333333333333333, 0.07692307692307693, 0.07142857
142857142, 0.06666666666666667, 0.0625, 0.058823529411764705, 0.05555555555555555, 0.05263157
894736842], [0.1, 0.09090909090909091, 0.08333333333333333, 0.07692307692307693, 0.0714285714
2857142, 0.06666666666666667, 0.0625, 0.058823529411764705, 0.05555555555555555, 0.0526315789
4736842, 0.05], [0.09090909090909091, 0.08333333333333333, 0.07692307692307693, 0.07142857142
857142, 0.06666666666666667, 0.0625, 0.058823529411764705, 0.05555555555555555, 0.05263157894
736842, 0.05, 0.047619047619047616]]

H_12:
[[1.0, 0.5, 0.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125, 0
.1111111111111111, 0.1, 0.09090909090909091, 0.08333333333333333], [0.5, 0.3333333333333333,
0.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.090909
09090909091, 0.08333333333333333, 0.07692307692307693], [0.3333333333333333, 0.25, 0.2, 0.166
66666666666666, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.0
8333333333333333, 0.07692307692307693, 0.07142857142857142], [0.25, 0.2, 0.16666666666666666,
0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.08333333333333333
, 0.07692307692307693, 0.07142857142857142, 0.06666666666666667], [0.2, 0.16666666666666666,
0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.08333333333333333
, 0.07692307692307693, 0.07142857142857142, 0.06666666666666667, 0.0625], [0.1666666666666666
6, 0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.08333333333333
333, 0.07692307692307693, 0.07142857142857142, 0.06666666666666667, 0.0625, 0.058823529411764
705], [0.14285714285714285, 0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.0833333333
3333333, 0.07692307692307693, 0.07142857142857142, 0.06666666666666667, 0.0625, 0.05882352941
1764705, 0.05555555555555555], [0.125, 0.1111111111111111, 0.1, 0.09090909090909091, 0.083333
33333333333, 0.07692307692307693, 0.07142857142857142, 0.06666666666666667, 0.0625, 0.0588235
29411764705, 0.05555555555555555, 0.05263157894736842], [0.1111111111111111, 0.1, 0.090909090
90909091, 0.08333333333333333, 0.07692307692307693, 0.07142857142857142, 0.06666666666666667,
0.0625, 0.058823529411764705, 0.05555555555555555, 0.05263157894736842, 0.05], [0.1, 0.090909
09090909091, 0.08333333333333333, 0.07692307692307693, 0.07142857142857142, 0.066666666666666
67, 0.0625, 0.058823529411764705, 0.05555555555555555, 0.05263157894736842, 0.05, 0.047619047
619047616], [0.09090909090909091, 0.08333333333333333, 0.07692307692307693, 0.071428571428571
42, 0.06666666666666667, 0.0625, 0.058823529411764705, 0.05555555555555555, 0.052631578947368
42, 0.05, 0.047619047619047616, 0.045454545454545456], [0.08333333333333333, 0.07692307692307
693, 0.07142857142857142, 0.06666666666666667, 0.0625, 0.058823529411764705, 0.05555555555555
555, 0.05263157894736842, 0.05, 0.047619047619047616, 0.045454545454545456, 0.043478260869565
216]]

Part 1.3

Plot the condition number of H_m for $m = 2, 3, \ldots, 12$

Collect all condition numbers in 1-norm of `H_m` into a list `one_norm`

In [7]:

(Top)

```
one_norm = []
# ===== 請實做程式 =====
for m in range(2,13,1):
    norm=0
    for i in range(1,m+1,1):
        s=0
        for j in range(1,m+1,1):
            s+=hilbert_matrix(m)[j-1][i-1]
        if s>norm:
            norm=s
    one_norm.append(norm)
# ====================
```

In [8]:

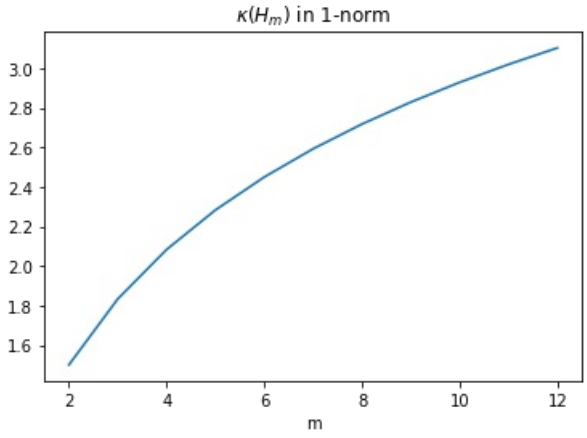
kappa_one_norm (Top)

```
print('one_norm:\n', one_norm)
### BEGIN HIDDEN TESTS
assert len(one_norm) == 11
### END HIDDEN TESTS
```

```
one_norm:
 [1.5, 1.8333333333333333, 2.083333333333333, 2.283333333333333, 2.4499999999999997, 2.592857
1428571425, 2.7178571428571425, 2.8289682539682537, 2.9289682539682538, 3.0198773448773446, 3
.103210678210678]
```

In [9]:

```
plt.plot(range(2,13), one_norm)
plt.xlabel('m')
plt.title(r'$\kappa(H_m)$ in 1-norm')
plt.show()
```



Collect all condition numbers in 2-norm of `H_m` into a list `two_norm`

In [10]:

(Top)

```
two_norm = []
# ===== 請實做程式 =====
for m in range(2,13,1):
    norm=0
    for i in range(1,m+1,1):
        s=0
        for j in range(1,m+1,1):
            s+=hilbert_matrix(m)[j-1][i-1]**2
        if s>norm:
            norm=s
    two_norm.append(norm**(1/2))
# ====================
```

In [11]:

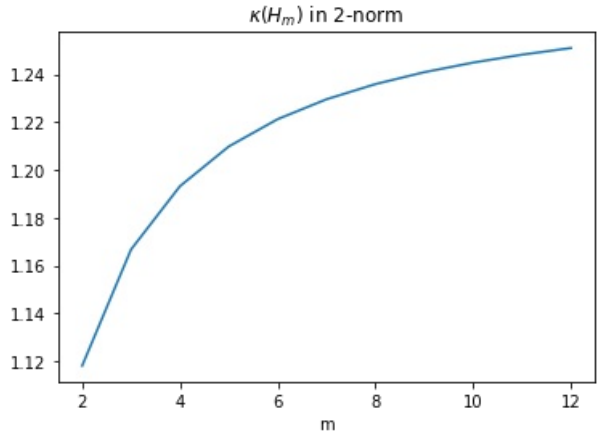
kappa_two_norm (Top)

```
print('two_norm:\n', two_norm)
### BEGIN HIDDEN TESTS
assert len(two_norm) == 11
### END HIDDEN TESTS
```

```
two_norm:
 [1.118033988749895, 1.1666666666666667, 1.1931517552730295, 1.209797962930634, 1.22122434011
48246, 1.2295515654718165, 1.235889174705481, 1.240873777290237, 1.2448966748957686, 1.248211
5982382387, 1.2509902631199423]
```

In [12]:

```
plt.plot(range(2,13), two_norm)
plt.xlabel('m')
plt.title(r'$\kappa(H_m)$ in 2-norm')
plt.show()
```



Part 2

Now generate the m-vector $b_m = H_m x$ also, where x is the m-vector with all of its components equal to 1.

Use Gaussian elimination to solve the resulting linear system $H_m x = b_m$ with H_m and b given above, obtaining an approximate solution $\tilde{x}$.

Part 2.1

Construct the m-vector b_m for $m = 2, 3, \ldots, 12$. Store all 1D `np.array` b_m into the list `b_m` .

In [13]:

(Top)

```
'''
Hint:
    b_m = ?
'''
# ===== 請實做程式 =====
b_m=[]
for i in range(0,11,1):
    x=np.linspace(1,1,i+2)
    b_m.append(H_m[i]@x)
# ====================
```

Print b_m

In [14]:

b_m (Top)

```
for i in range(len(b_m)):
    print('b_%d:' % (i+2))
    print(b_m[i])
    print()
### BEGIN HIDDEN TESTS
error = 0
for m in range(2,13):
    error += LA.norm(b_m[m-2] - np.array([[1/(i + j + 1) for j in range(m)] for i in range(m)])@np.ones(m
))
assert error < 1e-16
### END HIDDEN TESTS
```

```
b_2:
[1.5        0.83333333]

b_3:
[1.83333333 1.08333333 0.78333333]

b_4:
[2.08333333 1.28333333 0.95       0.75952381]

b_5:
[2.28333333 1.45       1.09285714 0.88452381 0.74563492]

b_6:
[2.45       1.59285714 1.21785714 0.99563492 0.84563492 0.73654401]

b_7:
[2.59285714 1.71785714 1.32896825 1.09563492 0.93654401 0.81987734
 0.73013376]

b_8:
[2.71785714 1.82896825 1.42896825 1.18654401 1.01987734 0.89680042
 0.80156233 0.72537185]

b_9:
[2.82896825 1.92896825 1.51987734 1.26987734 1.09680042 0.96822899
 0.86822899 0.78787185 0.72169538]

b_10:
[2.92896825 2.01987734 1.60321068 1.34680042 1.16822899 1.03489566
 0.93072899 0.84669538 0.77725094 0.7187714 ]

b_11:
[3.01987734 2.10321068 1.68013376 1.41822899 1.23489566 1.09739566
 0.98955252 0.90225094 0.82988251 0.7687714  0.71639045]

b_12:
[3.10321068 2.18013376 1.75156233 1.48489566 1.29739566 1.15621919
 1.04510808 0.95488251 0.87988251 0.81639045 0.761845   0.71441417]
```

Part 2.2

Implement the function of **Gaussian elimination**.

(Note that you need to implement it by hand, simply using some package functions is not allowed.)

Store all approximate solutions $\tilde{x}$ of H_m into a list `x_m` for $m = 2, 3, \ldots, 12$

In [15]:

(Top)

```
def gaussian_elimination(
    A,
    b
):
    '''
    Arguments:
        A : 2D np.array
        b : 1D np.array

    Return:
        x : 1D np.array, solution to Ax=b
    '''

    # ===== 請實做程式 =====
    n=len(b)
    for i in range(0,n):
        w=0
        for j in range(i,n):
            if A[j][i]!=0:
                break
            w+=1
        if w==n-i:
            break
        for k in range(0,n):
            A[i][k]+=A[i+w][k]
        b[i]+=b[i+w]
        for j in range(i+1,n):
            v=A[j][i]/A[i][i]
            for k in range(0,n):
                A[j][k]-=v*A[i][k]
            b[j]-= v*b[i]
    x=[b[n-1]/A[n-1][n-1]]
    for i in range(0,n-1):
        u=b[n-2-i]
        for j in range(n-i-1,n):
            u-=A[n-2-i][j]*x[j-n+i+1]
        x.insert(0,u/A[n-2-i][n-2-i])
    return(x)
    # ====================
```

In [ ]:

In [16]:

```
x_m = []
for i in range(len(H_m)):
    x = gaussian_elimination(H_m[i], b_m[i])
    x_m.append(x)
```

Part 3

Investigate the error behavior of the computed solution $\tilde{x}$.

(i) Compute the ∞-norm of the residual $r = b - H_m\tilde{x}$.

(ii) Compute the error $\delta x = \tilde{x} - x$, where x is the vector of all ones.

(iii) How large can you take m before there is no significant digits in the solution ?

Part 3.1

Compute the ∞-norm of the residual $r_m = b_m - H_m\tilde{x}$ for $m = 2, 3, \ldots, 12$. And store the values into the list `r_m` .

In [17]:

(Top)

```
r_m = []
# ===== 請實做程式 =====
for i in range(len(H_m)):
    c=b_m[i]-H_m[i]@np.array(x_m[i])
    d=0
    for j in range (0,i):
        if d<c[j]:
            d=c[j]
    r_m.append(d)
# =====================
```

In [18]:

(Top)

infty_norm

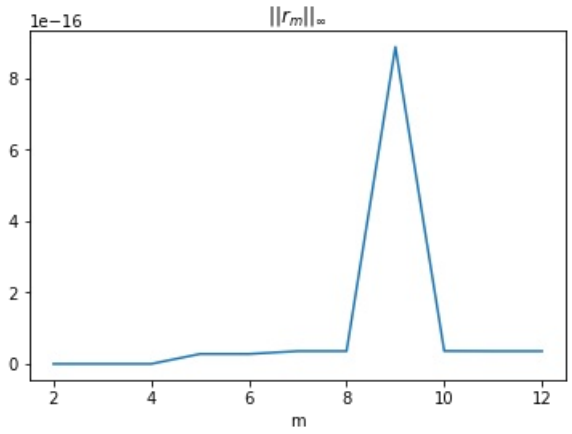
```
print('r_m:\n', r_m)
### BEGIN HIDDEN TESTS
assert np.sum(r_m) < 1e-12
### END HIDDEN TESTS
```

```
r_m:
 [0, 0, 0, 2.7755575615628914e-17, 2.7755575615628914e-17, 3.5670251474773096e-17, 3.56702514
74773096e-17, 8.881784197001252e-16, 3.599551212651875e-17, 3.5561831257524545e-17, 3.5561831
257524545e-17]
```

Plot the figure of the ∞-norm of the residual for $m = 2, 3, \ldots, 12$

In [19]:

```
plt.plot(range(2,13), r_m)
plt.xlabel('m')
plt.title(r'$||r_m||_\infty$')
plt.show()
```



Part 3.2

Compute the error $\delta x = \tilde{x} - x$, where x is the vector of all ones. And store the values into the list `delta_x` .

In [20]:

(Top)

```
delta_x = []
# ===== 請實做程式 =====
for i in range(len(H_m)):
    x=np.linspace(1,1,i+2)
    delta_x.append(x_m[i]-x)
# ====================
```

Collect all errors δx in 2-norm into the list `delta_x_two_norm` for $m = 2, 3, \ldots, 12$

In [21]:

(Top)

```
delta_x_two_norm = []
# ===== 請實做程式 =====
for i in range(len(H_m)):
    d=0
    for j in range (len(delta_x[i])):
        d+=(delta_x[i][j])**2
    delta_x_two_norm.append(d**(1/2))
# ====================
```

In [22]:

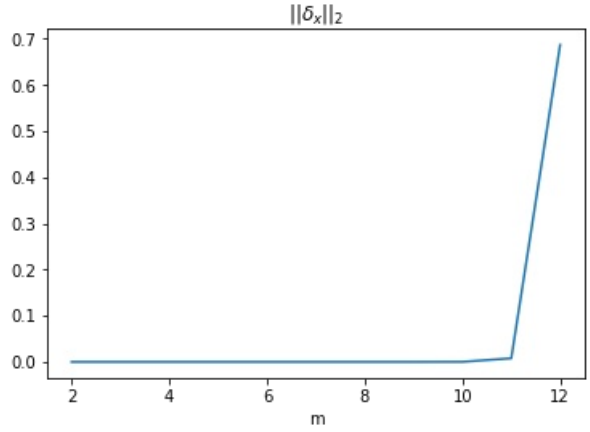
`delta_x_two_norm` (Top)

```
print('delta_x_two_norm =', delta_x_two_norm)
### BEGIN HIDDEN TESTS
assert (len(delta_x_two_norm) == 11) and (np.mean(delta_x_two_norm) <= 0.1)
### END HIDDEN TESTS
```

```
delta_x_two_norm = [8.005932084973442e-16, 1.7609551143107597e-14, 1.531093756529614e-13, 3.4
727313768072255e-12, 3.380199028306033e-10, 9.687977630572595e-09, 4.2446917552043934e-07, 1.
7393602592024017e-07, 0.00014527351138774312, 0.007757231295820366, 0.6867691718060671]
```

In [23]:

```
plt.plot(range(2,13), delta_x_two_norm)
plt.xlabel('m')
plt.title(r'$||\delta_x||_2$')
plt.show()
```



(Top)

Part 3.3

How large can you take m before there is no significant digits in the solution ?

about m=11

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```