exercise1 (Score: 3.0 / 14.0)

1. Comment
2. Test cell (Score: 0.0 / 2.0)
3. Comment
4. Test cell (Score: 0.0 / 1.0)
5. Comment
6. Test cell (Score: 0.0 / 1.0)
7. Comment
8. Test cell (Score: 0.0 / 1.0)
9. Comment
10. Test cell (Score: 0.0 / 1.0)
11. Comment
12. Comment
13. Test cell (Score: 3.0 / 3.0)
14. Comment
15. Comment
16. Test cell (Score: 0.0 / 2.0)
17. Task (Score: 0.0 / 3.0)

Exercise 1

An $m \times m$ *Hilbert matrix* H_m has entries $h_{ij} = 1/(i+j-1)$ for $1 \leq i,j \leq m$, 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 [1]:

```
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 [2]:

(Top)

```
def hilbert_matrix(m):
    '''
    Return:
        2D np.array, the Hildert Matrix of order m
    '''
    # ===== 請實做程式 =====

    # ====================
```

Comments:
No response.

Test your function.

In [3]:

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:
 None

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-c9e75a708675> in <module>
      1 print('H_2:\n', hilbert_matrix(2))
      2 ### BEGIN HIDDEN TESTS
----> 3 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
      4 ### END HIDDEN TESTS

TypeError: unsupported operand type(s) for -: 'NoneType' and 'float'
```

Part 1.2

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

In [4]:

(Top)

```
H_m = []
# ===== 請實做程式 =====

# ====================
```

Comments:
No response.

Check your Hilbert matrix list.

In [5]:

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
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-5-5815594e27ed> in <module>
      6 error = 0
      7 for m in range(2, 13):
----> 8     error += LA.norm(hilbert_matrix(m) - np.array([[1/(i + j + 1) for j in range(m)]
for i in range(m)]))
      9 assert error < 1e-16
     10 ### END HIDDEN TESTS

TypeError: unsupported operand type(s) for -: 'NoneType' and 'float'
```

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 [6]:

(Top)

```
one_norm = []
# ===== 請實做程式 =====

# ====================
```

Comments:
No response.

In [7]:

kappa_one_norm (Top)

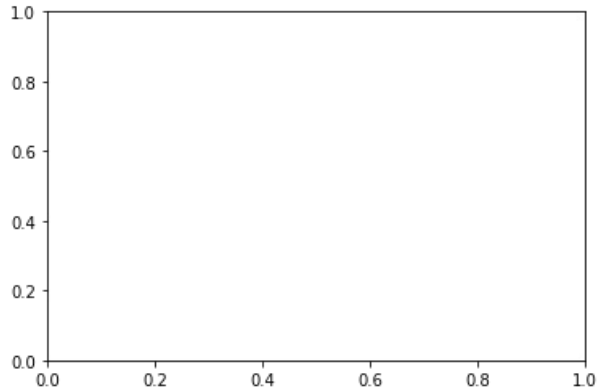
```
print('one_norm:\n', one_norm)
### BEGIN HIDDEN TESTS
assert len(one_norm) == 11
### END HIDDEN TESTS
```

```
one_norm:
 []

---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-7-721ea82aa40c> in <module>
      1 print('one_norm:\n', one_norm)
      2 ### BEGIN HIDDEN TESTS
----> 3 assert len(one_norm) == 11
      4 ### END HIDDEN TESTS

AssertionError:
```

In [8]:

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-8-0b0cebf9fb51> in <module>
----> 1 plt.plot(range(2,13), one_norm)
      2 plt.xlabel('m')
      3 plt.title(r'$\kappa(H_m)$ in 1-norm')
      4 plt.show()

/usr/local/lib/python3.6/dist-packages/matplotlib/pyplot.py in plot(scalex, scaley, data, *ar
gs, **kwargs)
   2793     return gca().plot(
   2794         *args, scalex=scalex, scaley=scaley, **({"data": data} if data
-> 2795         is not None else {}), **kwargs)
   2796
   2797

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_axes.py in plot(self, scalex, scaley,
data, *args, **kwargs)
   1664         """
   1665         kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D._alias_map)
-> 1666         lines = [*self._get_lines(*args, data=data, **kwargs)]
   1667         for line in lines:
   1668             self.add_line(line)

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_base.py in __call__(self, *args, **kw
args)
    223                 this += args[0],
    224                 args = args[1:]
--> 225             yield from self._plot_args(this, kwargs)
    226
    227     def get_next_color(self):

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_base.py in _plot_args(self, tup, kwar
gs)
    389             x, y = index_of(tup[-1])
    390
--> 391         x, y = self._xy_from_xy(x, y)
    392
    393         if self.command == 'plot':

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_base.py in _xy_from_xy(self, x, y)
    268         if x.shape[0] != y.shape[0]:
    269             raise ValueError("x and y must have same first dimension, but "
--> 270                              "have shapes {} and {}".format(x.shape, y.shape))
    271         if x.ndim > 2 or y.ndim > 2:
    272             raise ValueError("x and y can be no greater than 2-D, but have "

ValueError: x and y must have same first dimension, but have shapes (11,) and (0,)
```



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

In [9]:

(Top)

```
two_norm = []
# ===== 請實做程式 =====

# ====================
```

Comments:
No response.

In [10]:

kappa_two_norm (Top)

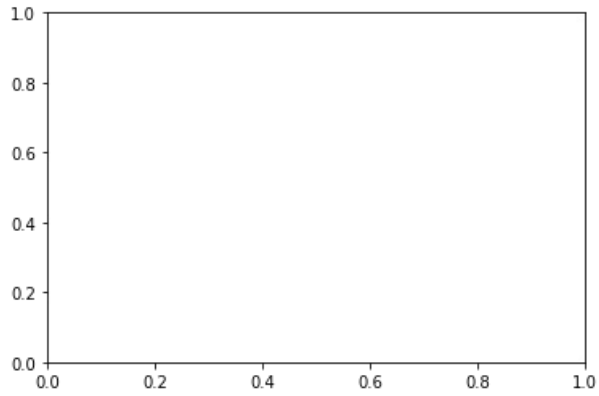
```
print('two_norm:\n', two_norm)
### BEGIN HIDDEN TESTS
assert len(two_norm) == 11
### END HIDDEN TESTS
```

```
two_norm:
 []

---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-10-7a90049dcc97> in <module>
      1 print('two_norm:\n', two_norm)
      2 ### BEGIN HIDDEN TESTS
----> 3 assert len(two_norm) == 11
      4 ### END HIDDEN TESTS

AssertionError:
```

In [11]:

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-11-f455ca31ae41> in <module>
----> 1 plt.plot(range(2,13), two_norm)
      2 plt.xlabel('m')
      3 plt.title(r'$\kappa(H_m)$ in 2-norm')
      4 plt.show()

/usr/local/lib/python3.6/dist-packages/matplotlib/pyplot.py in plot(scalex, scaley, data, *ar
gs, **kwargs)
   2793     return gca().plot(
   2794         *args, scalex=scalex, scaley=scaley, **({"data": data} if data
-> 2795         is not None else {}), **kwargs)
   2796
   2797

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_axes.py in plot(self, scalex, scaley,
data, *args, **kwargs)
   1664         """
   1665         kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D._alias_map)
-> 1666         lines = [*self._get_lines(*args, data=data, **kwargs)]
   1667         for line in lines:
   1668             self.add_line(line)

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_base.py in __call__(self, *args, **kw
args)
    223                 this += args[0],
    224                 args = args[1:]
--> 225             yield from self._plot_args(this, kwargs)
    226
    227     def get_next_color(self):

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_base.py in _plot_args(self, tup, kwar
gs)
    389             x, y = index_of(tup[-1])
    390
--> 391         x, y = self._xy_from_xy(x, y)
    392
    393         if self.command == 'plot':

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_base.py in _xy_from_xy(self, x, y)
    268         if x.shape[0] != y.shape[0]:
    269             raise ValueError("x and y must have same first dimension, but "
--> 270                              "have shapes {} and {}".format(x.shape, y.shape))
    271         if x.ndim > 2 or y.ndim > 2:
    272             raise ValueError("x and y can be no greater than 2-D, but have "

ValueError: x and y must have same first dimension, but have shapes (11,) and (0,)
```



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 [12]:

(Top)

```
'''
Hint:
    b_m = ?
'''
# ===== 請實做程式 =====

# ====================
```

Comments:
No response.

Out[12]:

```
'\nHint:\n    b_m = ?\n'
```

Print `b_m`

In [13]:

`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
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-13-03ed81c14c8f> in <module>
----> 1 for i in range(len(b_m)):
      2     print('b_%d:' % (i+2))
      3     print(b_m[i])
      4     print()
      5 ### BEGIN HIDDEN TESTS

NameError: name 'b_m' is not defined
```

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.)

In [14]:

(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 =====

    # ====================
```

Comments:
No response.

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

In [15]:

```
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 [16]:

(Top)

```
r_m = []
# ===== 請實做程式 =====

# ====================
```

Comments:
No response.

In [17]:

infty_norm (Top)

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

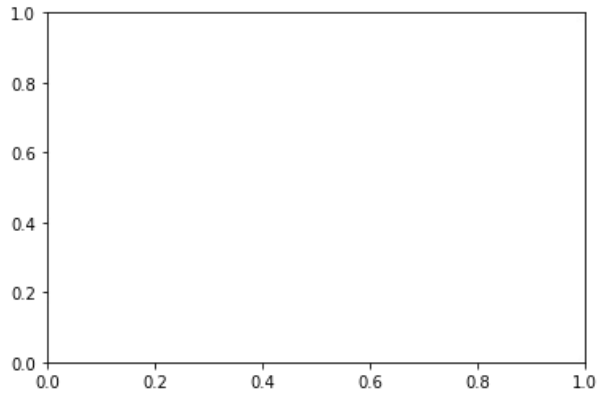
```
r_m:
 []
```

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

In [18]:

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-18-f2a2dd57ac83> in <module>
----> 1 plt.plot(range(2,13), r_m)
      2 plt.xlabel('m')
      3 plt.title(r'$||r_m||_\infty$')
      4 plt.show()

/usr/local/lib/python3.6/dist-packages/matplotlib/pyplot.py in plot(scalex, scaley, data, *ar
gs, **kwargs)
   2793     return gca().plot(
   2794         *args, scalex=scalex, scaley=scaley, **({"data": data} if data
-> 2795         is not None else {}), **kwargs)
   2796
   2797

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_axes.py in plot(self, scalex, scaley,
data, *args, **kwargs)
   1664         """
   1665         kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D._alias_map)
-> 1666         lines = [*self._get_lines(*args, data=data, **kwargs)]
   1667         for line in lines:
   1668             self.add_line(line)

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_base.py in __call__(self, *args, **kw
args)
    223                 this += args[0],
    224                 args = args[1:]
--> 225             yield from self._plot_args(this, kwargs)
    226
    227     def get_next_color(self):

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_base.py in _plot_args(self, tup, kwar
gs)
    389             x, y = index_of(tup[-1])
    390
--> 391         x, y = self._xy_from_xy(x, y)
    392
    393         if self.command == 'plot':

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_base.py in _xy_from_xy(self, x, y)
    268         if x.shape[0] != y.shape[0]:
    269             raise ValueError("x and y must have same first dimension, but "
--> 270                              "have shapes {} and {}".format(x.shape, y.shape))
    271         if x.ndim > 2 or y.ndim > 2:
    272             raise ValueError("x and y can be no greater than 2-D, but have "

ValueError: x and y must have same first dimension, but have shapes (11,) and (0,)
```



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 [19]:

(Top)

```
delta_x = []
# ===== 請實做程式 =====

# ====================
```

Comments:
No response.

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

In [20]:

(Top)

```
delta_x_two_norm = []
# ===== 請實做程式 =====

# ====================
```

Comments:
No response.

In [21]:

`delta_x_two_norm` (Top)

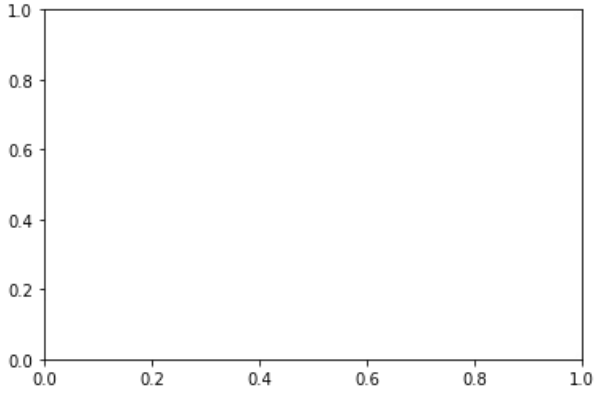
```
print('delta_x_two_norm =', delta_x_two_norm)
### BEGIN HIDDEN TESTS
assert (len(delta_x_two_norm) == 11) and (np.mean(delta_x_two_norm) <= 0.1)
### END HIDDEN TESTS
```

```
delta_x_two_norm = []

---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-21-2e1d931ec035> in <module>
      1 print('delta_x_two_norm =', delta_x_two_norm)
      2 ### BEGIN HIDDEN TESTS
----> 3 assert (len(delta_x_two_norm) == 11) and (np.mean(delta_x_two_norm) <= 0.1)
      4 ### END HIDDEN TESTS

AssertionError:
```

In [22]:

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-22-dc8a443fdad0> in <module>
----> 1 plt.plot(range(2,13), delta_x_two_norm)
      2 plt.xlabel('m')
      3 plt.title(r'$||\delta_x||_2$')
      4 plt.show()

/usr/local/lib/python3.6/dist-packages/matplotlib/pyplot.py in plot(scalex, scaley, data, *ar
gs, **kwargs)
   2793     return gca().plot(
   2794         *args, scalex=scalex, scaley=scaley, **({"data": data} if data
-> 2795         is not None else {}), **kwargs)
   2796
   2797

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_axes.py in plot(self, scalex, scaley,
data, *args, **kwargs)
   1664         """
   1665         kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D._alias_map)
-> 1666         lines = [*self._get_lines(*args, data=data, **kwargs)]
   1667         for line in lines:
   1668             self.add_line(line)

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_base.py in __call__(self, *args, **kw
args)
    223                 this += args[0],
    224                 args = args[1:]
--> 225             yield from self._plot_args(this, kwargs)
    226
    227     def get_next_color(self):

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_base.py in _plot_args(self, tup, kwar
gs)
    389             x, y = index_of(tup[-1])
    390
--> 391         x, y = self._xy_from_xy(x, y)
    392
    393         if self.command == 'plot':

/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_base.py in _xy_from_xy(self, x, y)
    268         if x.shape[0] != y.shape[0]:
    269             raise ValueError("x and y must have same first dimension, but "
--> 270                              "have shapes {} and {}".format(x.shape, y.shape))
    271         if x.ndim > 2 or y.ndim > 2:
    272             raise ValueError("x and y can be no greater than 2-D, but have "

ValueError: x and y must have same first dimension, but have shapes (11,) and (0,)
```



(Top)

Part 3.3

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

Please write down your answer here.