exercise1-newton (Score: 13.0 / 13.0)

1. Test cell (Score: 1.0 / 1.0)
2. Test cell (Score: 1.0 / 1.0)
3. Test cell (Score: 1.0 / 1.0)
4. Written response (Score: 1.0 / 1.0)
5. Test cell (Score: 1.0 / 1.0)
6. Coding free-response (Score: 2.0 / 2.0)
7. Test cell (Score: 1.0 / 1.0)
8. Coding free-response (Score: 2.0 / 2.0)
9. Written response (Score: 3.0 / 3.0)

Lab 2

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. 四個求根演算法的實作可以參考lab-2 (https://yuanyuyuan.github.io/itcm/lab-2.html)，裡面有教學影片也有範例程式可以套用。
4. **Deadline: 10/9(Wed.)**

In [1]:

```
name = "李澤諺"
student_id = "B05902023"
```

Exercise 1 - Newton

Use the Newton's method to find roots of

$$f(x) = cosh(x) + cos(x) - c, \text{for } c = 1, 2, 3,$$

Import libraries

In [2]:

```
import matplotlib.pyplot as plt
import numpy as np
```

1. Define the function $g(c)(x) = f(x) = cosh(x) + cos(x) - c$ with parameter $c = 1, 2, 3$ and its derivative df.

In [3]:

(Top)

```
def g(c):
    assert c == 1 or c == 2 or c == 3
    def f(x):
        return np.cosh(x) + np.cos(x) - c
    return f

def df(x):
    return np.sinh(x) - np.sin(x)
```

Pass the following assertion.

In [4]:

cell-b59c94b754b1fc9e (Top)

```
assert g(1)(0) == np.cosh(0) + np.cos(0) - 1
assert df(0) == 0
### BEGIN HIDDEN TESTS
assert g(2)(0) == np.cosh(0) + np.cos(0) - 2
assert g(3)(0) == np.cosh(0) + np.cos(0) - 3
assert df(1) == np.sinh(1) - np.sin(1)
### END HIDDEN TESTS
```

2. Implement the algorithm

In [5]:

(Top)

```python
def newton(
    func,
    d_func,
    x_0,
    tolerance=1e-7,
    max_iterations=5,
    report_history=False
):
    '''
    Parameters
    ----------
    func : function
        The target function.
    d_func : function
        The derivative of the target function.
    x_0 : float
        Initial guess point for a solution f(x)=0.
    tolerance : float
        One of the termination conditions. Error tolerance.
    max_iterations : int
        One of the termination conditions. The amount of iterations allowed.
    report_history: bool
        Whether to return history.

    Returns
    -------
    solution : float
        Approximation of the root.
    history: dict
        Return history of the solving process if report_history is True.
    '''

    x = x_0
    iteration = 0
    history = {'estimation' : [] , 'error' : []}
    while True:
        error = abs(func(x))

        history['estimation'].append(x)
        history['error'].append(error)

        iteration += 1
        if (iteration >= max_iterations or error < tolerance):
            return (x , history) if report_history else x

        if (d_func(x) == 0):
            return (None , history) if report_history else None

        x = x - func(x) / d_func(x)
```

Test your implementation with the assertion below.

In [6]:

cell-4d88293f2527c82d (Top)

```python
root = newton(
    lambda x: x**2 - x - 1,
    lambda x: 2*x - 1,
    1.2,
    max_iterations=100,
    tolerance=1e-7,
    report_history=False
)
assert abs(root - ((1 + np.sqrt(5)) / 2)) < 1e-7
```

---

3. Answer the following questions under the case $c = 1$.

Plot the function to find an interval that contains the zero of f if possible.

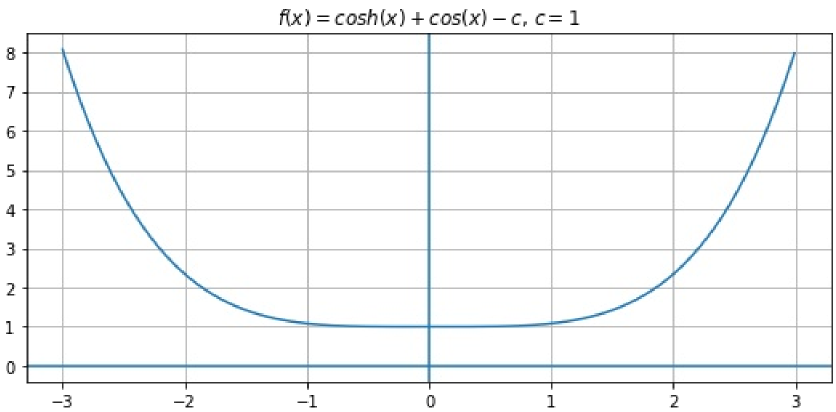
In [7]:

(Top)

```
c = 1
f = g(c)

search_range = np.arange(-3 , 3 , 0.01)

fig, ax = plt.subplots(figsize=(9, 4))
ax.plot(search_range, f(search_range))
ax.set_title(r'$f(x)=cosh(x)+cos(x)-c$, $c=$%d' % c)
ax.grid(True)
ax.axhline(y=0)
ax.axvline(x=0)
plt.show()
```



According to the figure above, estimate the zero of f.

For example,

```
root = 3        # 單根
root = -2, 1    # 多根
root = None     # 無解
```

In [8]:

(Top)

```
root = None
```

In [9]:

cell-d872c7c57f11c968 (Top)

```
print('My estimation of root:', root)
### BEGIN HIDDEN TESTS
if root == None:
    print('Right answer!')
else:
    raise AssertionError('Wrong answer!')
### END HIDDEN TESTS
```

```
My estimation of root: None
Right answer!
```

Try to find the zero with a tolerance of 10^{-10}. If it works, plot the error and estimation of each step. Otherwise, state the reason why the method failed on this case.

(Top)

因為 $\forall x \in \mathrm{R}$，皆有
$coshx + cosx - 1 = \frac{e^x + e^{-x}}{2} + cosx - 1 = \frac{1}{2}(\sum_{n=0}^{\infty}\frac{x^n}{n!} + \sum_{n=0}^{\infty}\frac{(-x)^n}{n!}) + \sum_{n=0}^{\infty}(-1)^n\frac{x^n}{(2n)!} - 1 = \frac{1}{2}((1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + \frac{x^7}{7!} + \cdots) + (1 -$
所以 $coshx + cosx - 1 = 0$ 無解
(或是由上圖也可以直接看出 $coshx + cosx - 1 = 0$ 無解)

---

4. Answer the following questions under the case $c = 2$.

Plot the function to find an interval that contains the zero of f if possible.

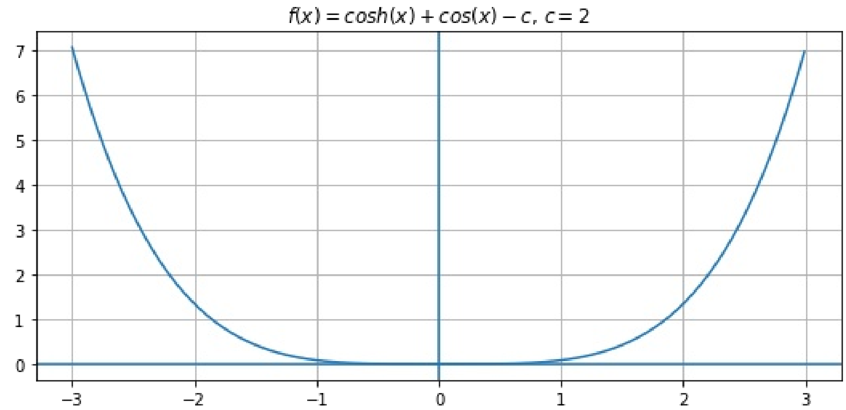
In [10]:

(Top)

```
c = 2
f = g(c)

search_range = np.arange(-3 , 3 , 0.01)

fig, ax = plt.subplots(figsize=(9, 4))
ax.plot(search_range, f(search_range))
ax.set_title(r'$f(x)=cosh(x)+cos(x)-c$, $c=$%d' % c)
ax.grid(True)
ax.axhline(y=0)
ax.axvline(x=0)
plt.show()
```



According to the figure above, estimate the zero of f.

For example,

```
root = 3        # 單根
root = -2, 1    # 多根
root = None     # 無解
```

In [11]:

(Top)

```
root = 0
```

In [12]:

```
cell-20fddbe6fa4c437b                                                    (Top)

print('My estimation of root:', root)

### BEGIN HIDDEN TESTS
assert type(root) is float or int, 'Wrong type!'
### END HIDDEN TESTS
```

```
My estimation of root: 0
```

Try to find the zero with a tolerance of 10^{-10}. If it works, plot the error and estimation of each step. Otherwise, state the reason why the method failed on this case.

In [13]:

```
                                                                         (Top)

(x , history) = newton(func = g(2) , d_func = df , x_0 = 1 , tolerance = 1e-10 , max_iterations = 100 , r
eport_history = True)
```

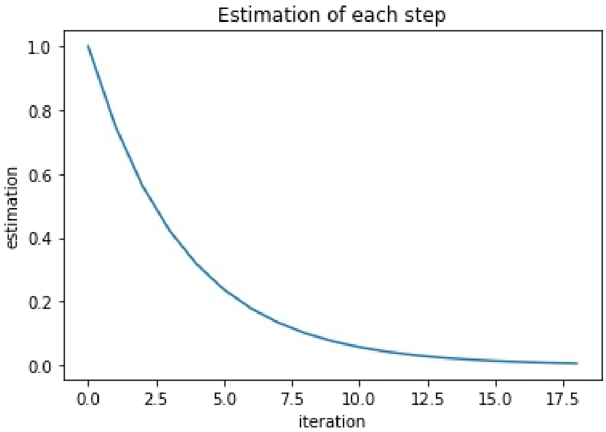
In [14]:

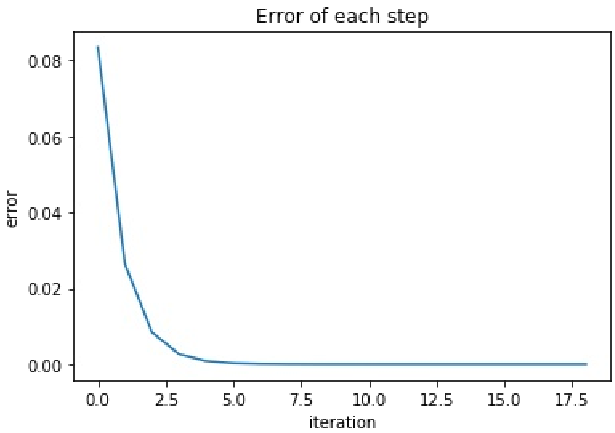
```
print('Estimation of root by Newton\'s method:' , x)

plt.plot(history['estimation'])
plt.title('Estimation of each step')
plt.xlabel('iteration')
plt.ylabel('estimation')
plt.show()

plt.plot(history['error'])
plt.title('Error of each step')
plt.xlabel('iteration')
plt.ylabel('error')
plt.show()
```

```
Estimation of root by Newton's method: 0.005639347364278358
```





5. Answer the following questions under the case $c = 3$.

Plot the function to find an interval that contains the zeros of f if possible.

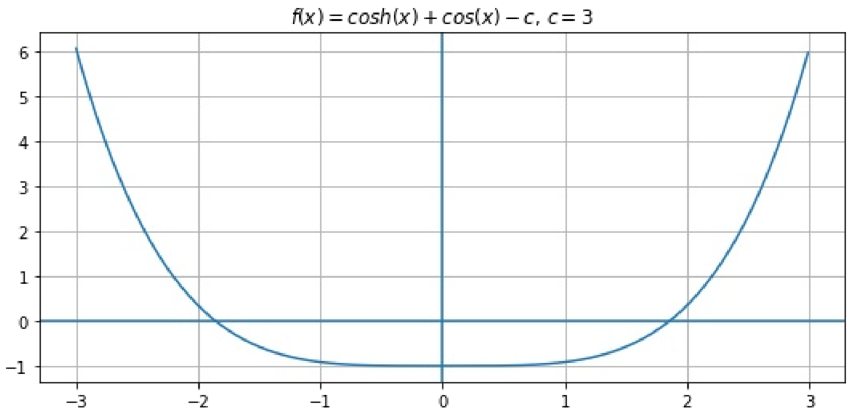
In [15]:

(Top)

```
c = 3
f = g(c)

search_range = np.arange(-3 , 3 , 0.01)

fig, ax = plt.subplots(figsize=(9, 4))
ax.plot(search_range, f(search_range))
ax.set_title(r'$f(x)=cosh(x)+cos(x)-c$, $c=$%d' % c)
ax.grid(True)
ax.axhline(y=0)
ax.axvline(x=0)
plt.show()
```



According to the figure above, estimate the zero of f.

For example,

```
root = 3        # 單根
root = -2, 1    # 多根
root = None     # 無解
```

In [16]:

(Top)

```
root = (-1.8 , 1.8)
```

In [17]:

cell-06ec0b20844075c7

(Top)

```
print('My estimation of root:', root)

### BEGIN HIDDEN TESTS
assert type(root) == tuple, 'Should be multiple roots!'
### END HIDDEN TESTS
```

```
My estimation of root: (-1.8, 1.8)
```

Try to find the zero with a tolerance of 10^{-10}. If it works, plot the error and estimation of each step. Otherwise, state the reason why the method failed on this case.

In [18]:

(Top)

```
(x , history) = newton(func = g(3) , d_func = df , x_0 = 2 , tolerance = 1e-10 , max_iterations = 100 , r
eport_history = True)
```

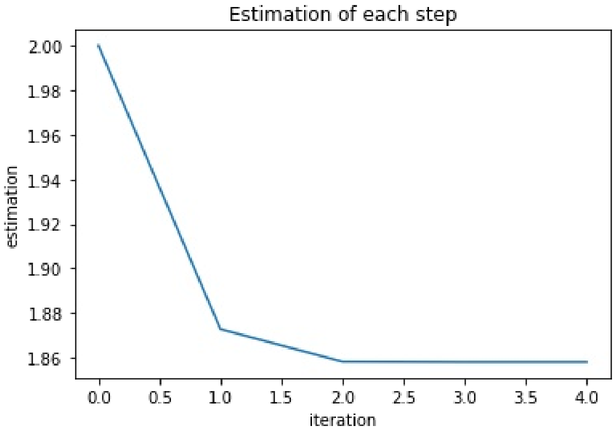
In [19]:

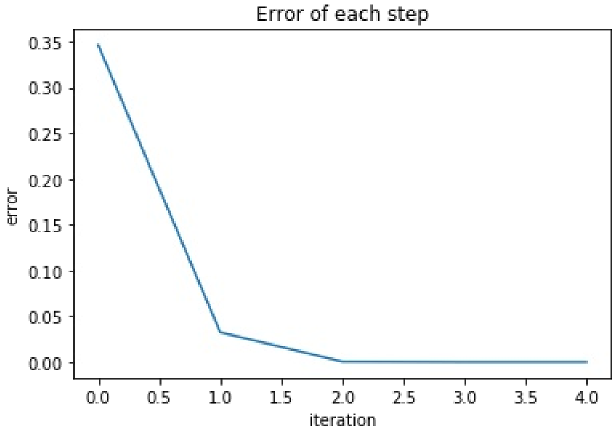
```
print('Estimation of root by Newton\'s method:' , x)

plt.plot(history['estimation'])
plt.title('Estimation of each step')
plt.xlabel('iteration')
plt.ylabel('estimation')
plt.show()

plt.plot(history['error'])
plt.title('Error of each step')
plt.xlabel('iteration')
plt.ylabel('error')
plt.show()
```

Estimation of root by Newton's method: 1.8579208291501987





In [20]:

```
(x , history) = newton(func = g(3) , d_func = df , x_0 = -2 , tolerance = 1e-10 , max_iterations = 100 , r
eport_history = True)
```

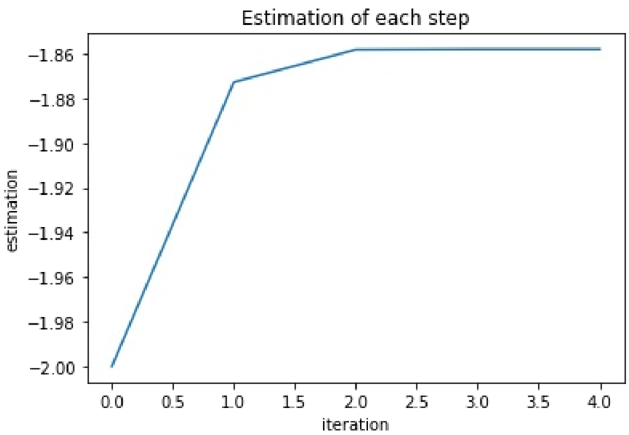
In [21]:

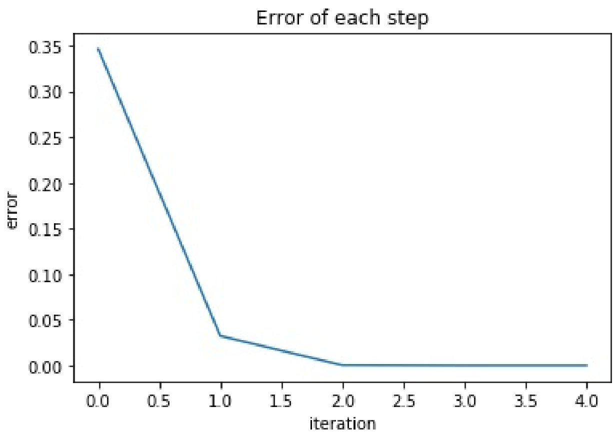
```
print('Estimation of root by Newton\'s method:' , x)

plt.plot(history['estimation'])
plt.title('Estimation of each step')
plt.xlabel('iteration')
plt.ylabel('estimation')
plt.show()

plt.plot(history['error'])
plt.title('Error of each step')
plt.xlabel('iteration')
plt.ylabel('error')
plt.show()
```

```
Estimation of root by Newton's method: -1.8579208291501987
```





Discussion

For all cases above(c=1,2,3), do the results(e.g. error behaviors, estimations, etc) agree with the theoretical analysis?

(Top)

當 $c = 1$ 時，方程式無解。
當 $c = 2$ 時，由上圖大致可以看出 $|x_{n+1} - \alpha| \leq C|x_n - \alpha|$ ，其中 $0 \leq C < 1$，故estimation為linearly converge(其與理論中當方程式有重根時，Newton's method的rate of convergence相符)。
當 $c = 3$ 時，由上圖大致可以看出 $|x_{n+1} - \alpha| \leq C|x_n - \alpha|^2$，故estimation為quadratically converge(其與理論中Newton's method的rate of convergence相符)。

In [ ]: