

exercise1 (Score: 18.0 / 20.0)

1. [Task](#) (Score: 4.0 / 4.0)
2. [Test cell](#) (Score: 2.0 / 2.0)
3. [Test cell](#) (Score: 4.0 / 4.0)
4. [Test cell](#) (Score: 2.0 / 2.0)
5. [Task](#) (Score: 4.0 / 4.0)
6. [Task](#) (Score: 2.0 / 4.0)
7. [Comment](#)

## Lab 4

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(student\_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-4 \(https://yuanyuyuan.github.io/itcm/lab-4.html\)](https://yuanyuyuan.github.io/itcm/lab-4.html), 有任何問題歡迎找助教詢問。
4. **Deadline: 11/20(Wed.)**

In [1]:

```
name = "林以翎"  
student_id = "B06201024"
```

## Exercise 1. Finite Difference

### Part 0.

**Import necessary libraries. Note that diags library from scipy is used to construct the differentiation matrix below.**

In [2]:

```
import numpy as np  
import matplotlib.pyplot as plt  
from scipy.sparse import diags
```

Part 1.

Given a function  $u(x)$  which we want to find its derivative with numerical methods.

Consider a uniform grid partitioning  $x$  into  $\{x_1, x_2, \dots, x_n\}$  with grid size  $\Delta x = x_{j+1} - x_j, j \in \{1, 2, \dots, n\}$ , and a set of corresponding data values  $U = \{U_1, U_2, \dots, U_n\}$ , where

$$U_{j+k} = u(x_j + k\Delta x) = u(x_{j+k}), j \in \{1, 2, \dots, n\}.$$

We want to use one-sided finite-difference formula

$$\alpha_1 U_j + \alpha_2 U_{j+1} + \alpha_3 U_{j+2}$$

to approximate the derivative of  $u$  at all the points  $x_j, j \in \{1, 2, \dots, n\}$ , that is

$$u'(x_j) \approx W_j \triangleq \alpha_1 U_j + \alpha_2 U_{j+1} + \alpha_3 U_{j+2}.$$

(Top)

Part 1.1

Find the coefficients  $\alpha_j$  for  $j = 1, 2, 3$  which make the stencil above accurate for as high degree polynomials as possible.  
Write down your derivation in detail with Markdown/LaTeX.

$$\begin{aligned} U_j &= u(x_j) \\ U_{j+1} &= u(x_j + 1 \cdot \Delta x) = u(x) + \Delta x u'(x) + \frac{(\Delta x)^2}{2} u''(x) + O(u^3) \\ U_{j+2} &= u(x_j + 2 \cdot \Delta x) = u(x) + 2\Delta x u'(x) + \frac{4(\Delta x)^2}{2} u''(x) + O(u^3) \end{aligned}$$

$$\Rightarrow \begin{cases} \alpha_1 + \alpha_2 + \alpha_3 = 0 \\ \Delta x \alpha_2 + 2\Delta x \alpha_3 = 1 \\ \frac{(\Delta x)^2}{2} \alpha_2 + \frac{4(\Delta x)^2}{2} \alpha_3 = 0 \end{cases}$$

$$\Rightarrow (\alpha_1, \alpha_2, \alpha_3) = \left( \frac{-3}{2\Delta x}, \frac{2}{\Delta x}, \frac{-1}{2\Delta x} \right)$$

Part 1.2

Fill in the tuple variable `alpha` of length 3 with your answer above. (Suppose  $\Delta x = 1$ )

In [3]:

(Top)

```
# Hint: alpha = [value of alpha_1, value of alpha_2, value of alpha_3]
# ===== 請實做程式 =====
alpha = [-3/2, 2, -1/2]
# =====
```

In [4]:

cell-e7c9469885bebc80

(Top)

```
print('My alpha =', alpha)
### BEGIN HIDDEN TESTS
assert alpha == [-1.5, 2, -0.5] or alpha == (-1.5, 2, -0.5)
### END HIDDEN TESTS
```

My alpha = [-1.5, 2, -0.5]

## Part 2.

Suppose we use the finite-difference formula above to approximate and assume the problem is periodic, i.e. take  $U_0 = U_n$ ,  $U_1 = U_{n+1}$ , and so on.

Find the differentiation matrix  $D$  so that the numerical differentiation problem can be represented as a matrix-vector multiplication  $W \triangleq DU$ , where  $D \in \mathbb{R}^{n \times n}$ ,  $U \in \mathbb{R}^n$ , and  $W \in \mathbb{R}^n$ .

### Part 2.1

Complete the following function to construct the desired differentiation matrix under the **periodic boundary condition** with given number of partition  $n$ , coefficients of 3-point finite-difference formula  $\alpha$ , and mesh size  $\Delta x$ .

In [5]:

(Top)

```
def construct_differentiation_matrix(n, alpha, delta_x):
    ''' Construct
    Parameters
    -----
    n : int
        number of partition
    alpha : tuple of length 3
        alpha = (a1, a2, a3)
    delta_x : float
        mesh size

    Returns
    -----
    D : scipy.sparse.diags
    '''
    # ===== 請實做程式 =====
    D = [
        alpha[0] * np.ones(n),
        alpha[1] * np.ones(n-1),
        alpha[2] * np.ones(n-2),
        alpha[2] * np.ones(2),
        alpha[1] * np.ones(1)
    ]
    diagonal_matrix = diags(D, offsets=[0, 1, 2, -(n-2), -(n-1)])
    diagonal_matrix /= delta_x
    return diagonal_matrix

return D
# =====
```

## Part 2.2

Print and check your implementation.

In [6]:

cell-2ca00ba5ff115302

(Top)

```
print("For n = 8 and mesh size 1, D in dense form is")
sparse_D = construct_differentiation_matrix(8, alpha, 1)
dense_D = sparse_D.toarray()
print(dense_D)
### BEGIN HIDDEN TESTS
answer = np.array([
    [-1.5, 2., -0.5, 0., 0., 0., 0., 0.],
    [0., -1.5, 2., -0.5, 0., 0., 0., 0.],
    [0., 0., -1.5, 2., -0.5, 0., 0., 0.],
    [0., 0., 0., -1.5, 2., -0.5, 0., 0.],
    [0., 0., 0., 0., -1.5, 2., -0.5, 0.],
    [0., 0., 0., 0., 0., -1.5, 2., -0.5],
    [-0.5, 0., 0., 0., 0., 0., -1.5, 2.],
    [2., -0.5, 0., 0., 0., 0., 0., -1.5]
])
assert np.linalg.norm(dense_D - answer) < 1e-7
### END HIDDEN TESTS
```

For n = 8 and mesh size 1, D in dense form is

```
[[-1.5 2. -0.5 0. 0. 0. 0. 0.]
 [0. -1.5 2. -0.5 0. 0. 0. 0.]
 [0. 0. -1.5 2. -0.5 0. 0. 0.]
 [0. 0. 0. -1.5 2. -0.5 0. 0.]
 [0. 0. 0. 0. -1.5 2. -0.5 0.]
 [0. 0. 0. 0. 0. -1.5 2. -0.5]
 [-0.5 0. 0. 0. 0. 0. -1.5 2.]
 [2. -0.5 0. 0. 0. 0. 0. -1.5]]
```

Part 3.

Take  $u(x) = e^{\sin x}$  on the domain  $[-\pi, \pi]$ . Find the finite difference approximation  $W$  for  $\{u'(x_j)\}_{j=1}^n$  for various values of  $n = 2^k$ ,  $k = 3, 4, \dots, 10$ , and analyze the errors.

Part 3.1

Define the functions  $u$  and  $u'(x)$ .

In [7]:

(Top)

```
def u(x):
    # ===== 請實做程式 =====
    return np.exp(np.sin(x))
    # =====

def d_u(x):
    # ===== 請實做程式 =====
    return np.exp(np.sin(x))*np.cos(x)
    # =====
```

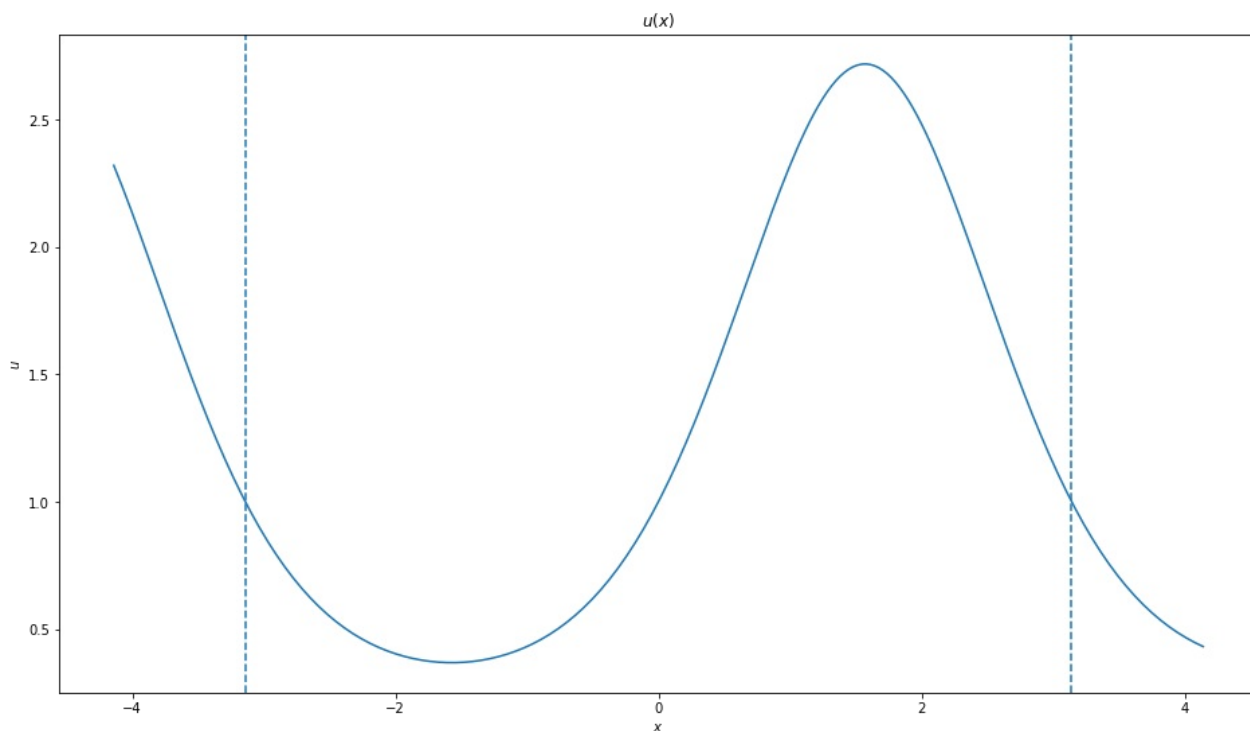
Plot and check the functions

In [8]:

cell-f97d6fb0842a6055

(Top)

```
x_range = np.linspace(-np.pi-1, np.pi+1, 2**8)
plt.figure(figsize=(16, 9))
plt.plot(x_range, u(x_range))
plt.axvline(x=np.pi, linestyle='--')
plt.axvline(x=-np.pi, linestyle='--')
plt.ylabel(r'$u$')
plt.xlabel(r'$x$')
plt.title(r'$u(x)$')
plt.show()
### BEGIN HIDDEN TESTS
assert u(1) == np.exp(np.sin(1))
assert u(3.14) == np.exp(np.sin(3.14))
assert d_u(1) == np.cos(1) * np.exp(np.sin(1))
assert d_u(0) == np.cos(0) * np.exp(np.sin(0))
### END HIDDEN TESTS
```



(Top)

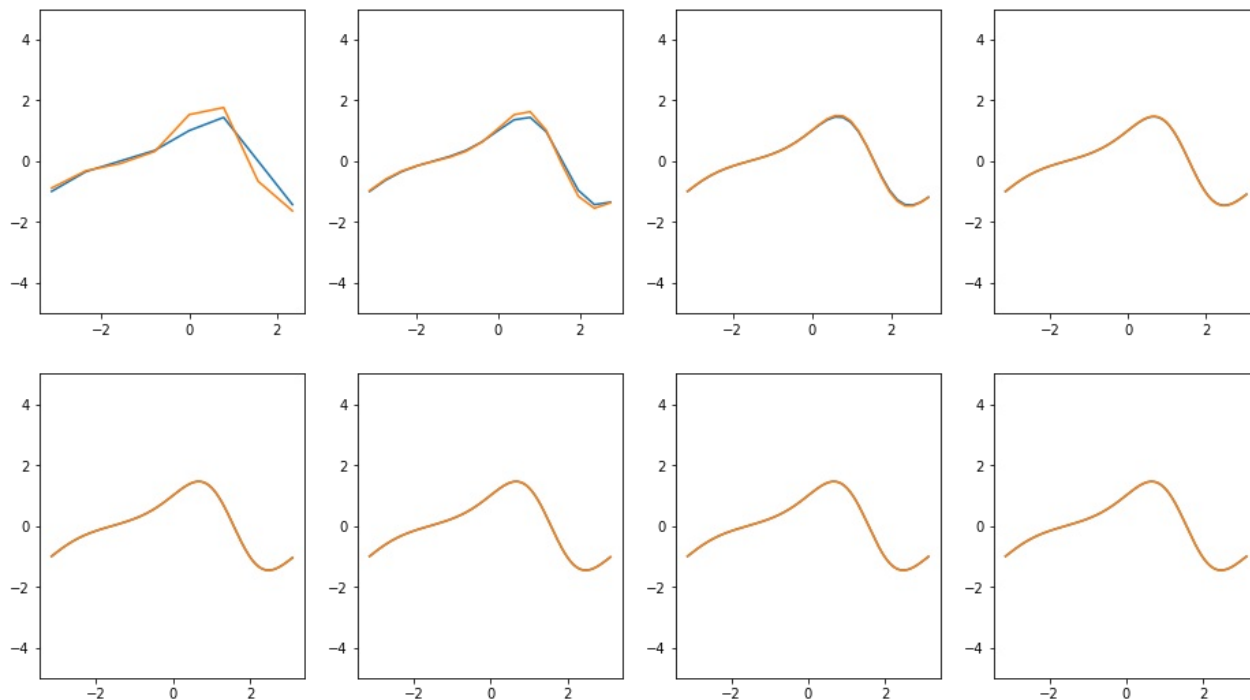
### Part 3.2

Plot the  $u'$  and  $W$  together for each point  $x_j, j \in \{1, 2, \dots, n\}$  with  $n = 2^k, k \in \{3, 4, \dots, 10\}$ . Note that there're total 8 figures to be plotted. And you need to compute the error, display them in the plots, and store them into the list variable `error_list` for further analysis below.

```

error_list = []
fig, axes = plt.subplots(2, 4, figsize=(16,9))
for idx, ax in enumerate(axes.flatten()):
    '''Hints:
    For each case in this for loop, you may follow the steps below
    1. Use idx to set k and n.
    2. Prepare n partition points of the domain.
    3. Construct D.
    4. Find u', U, and W.
    5. Compute the error between u' and W.
    6. Append the error into error_list.
    7. Use ax to plot u', W with proper labels, title
    8. Enable legend to show the labels of curves.
    9. To make the plots more readable, set a consistent range of y-axis e.g. ax.set_ylim([-3, 3])
    ...
    # ===== 請實做程式 =====
    k = idx + 3
    n = 2*k
    x = np.linspace(-np.pi, np.pi, n, endpoint=False)
    D = construct_differentiation_matrix(n, alpha, 2*np.pi/n)
    U = u(x)
    u_diff = d_u(x)
    W = D @ U
    error = np.abs(u_diff - W)
    ax.set_ylim([-5,5])
    ax.plot(x, u_diff)
    ax.plot(x, W)
    error_list.append(error)
    # =====

```

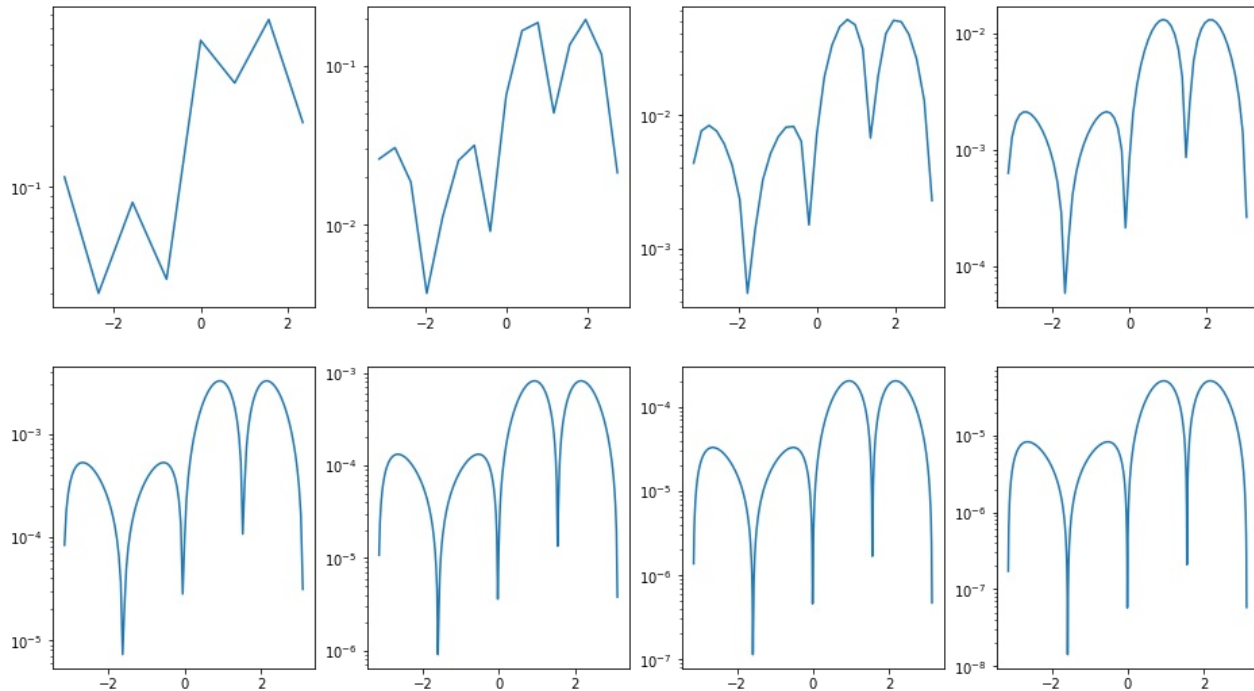


Plot the `error_list` with respect to  $k = 3, 4, \dots, 10$  in log scale to show the error behavior.

In [10]:

(Top)

```
# ===== 請實做程式 =====  
fig, axes = plt.subplots(2, 4, figsize=(16,9))  
for idx, ax in enumerate(axes.flatten()):  
    ax.set_yscale('log')  
    ax.plot(np.linspace(-np.pi, np.pi, 2**(idx+3), endpoint=False), error_list[idx])  
# =====
```



(Top)

### Part 3.3

From the figure above, what rates of convergence do you observe as  $\Delta x \rightarrow 0$ ?

#### Comments:

For what  $r$ ?

The rate of convergence  $\approx (\Delta x)^r$  for some  $r > 0$  as  $\Delta x \rightarrow 0$ .