

exercise2 (Score: 13.0 / 13.0)

1. [Test cell](#) (Score: 2.0 / 2.0)
2. [Test cell](#) (Score: 2.0 / 2.0)
3. [Coding free-response](#) (Score: 2.0 / 2.0)
4. [Written response](#) (Score: 2.0 / 2.0)
5. [Test cell](#) (Score: 1.0 / 1.0)
6. [Coding free-response](#) (Score: 2.0 / 2.0)
7. [Written response](#) (Score: 2.0 / 2.0)

## Lab 3

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-3 \(https://yuanyuyuan.github.io/itcm/lab-3.html\)](https://yuanyuyuan.github.io/itcm/lab-3.html)，有任何問題歡迎找助教詢問。
4. **Deadline: 10/30(Wed.)**

In [1]:

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

## Exercise 2

It is known that when interpolating a function  $f(x)$  with a polynomial  $p_{m+1}$  of degree  $m$  that using  $x_j$  for  $j = 0, 1, \dots, m$  as interpolation points the error has the form

$$|f(x) - p_{m+1}(x)| = \frac{|f^{(m+1)}(\xi_x)|}{(m+1)!} \left| \prod_{k=0}^m (x - x_k) \right|,$$

where  $\xi_x \in [x_0, x_m]$ .

Therefore, the polynomial  $\omega_m(t) := \prod_{k=0}^m (t - x_k)$  influences the size of the interpolation error.

**1. Put  $m + 1$  *distinct equidistant points* in the interval  $[-1, 1]$ , and plot  $\omega_m(t)$  for  $m = 5, 10, 15, 20$ .**

**Part 0. Import libraries.**

In [2]:

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

## Part 1. Define $\omega_m(t)$ function.

In [3]:

(Top)

```
def omega_m(t, x):
    # ===== 請實做程式 =====
    prod = 1
    for i in range(len(x)):
        prod *= t - x[i]
    return(prod)
    # =====
```

In [4]:

(Top)

omega

```
# Test
print('w_5(0.5) =', omega_m(0.5, np.linspace(-1, 1, 6)))

### BEGIN HIDDEN TESTS
from random import random

rd_number = random()
x = np.linspace(-1, 1, 11)

m = len(x)
product = 1

for i in range(m):
    product *= (rd_number - x[i])

assert omega_m(rd_number, np.linspace(-1, 1, 11)) == product, 'omega_m is wrong!'
### END HIDDEN TESTS
```

w\_5(0.5) = 0.017325000000000007

## Part 2. Define the equidistant points function.

For example, if  $m = 4$ , then  $m + 1$  distinct equidistant points in the interval  $[-1, 1]$  should be  $[-1, -0.5, 0, 0.5, 1]$ .

So the results of `equidistant_points(4)` will be `[-1. -0.5 0. 0.5 1. ]`.

In [5]:

(Top)

```
def equidistant_points(m):
    # ===== 請實做程式 =====
    x = np.linspace(-1, 1, m+1)
    points = [xi for xi in x]
    return(points)
    # =====
```

In [6]:

points

(Top)

```
# Test
m = 4
print("Equidistant points:", equidistant_points(m))

### BEGIN HIDDEN TESTS
m = 10
assert np.mean(np.array(equidistant_points(m)) - np.linspace(-1, 1, m+1)) < 1e-7, 'equidistant_points is wrong!'
### END HIDDEN TESTS
```

Equidistant points: [-1.0, -0.5, 0.0, 0.5, 1.0]

### Part 3. plot $\omega_m(t)$ for $m = 5, 10, 15, 20$ .

Please refer parts of plotting in "lagrange.ipynb".

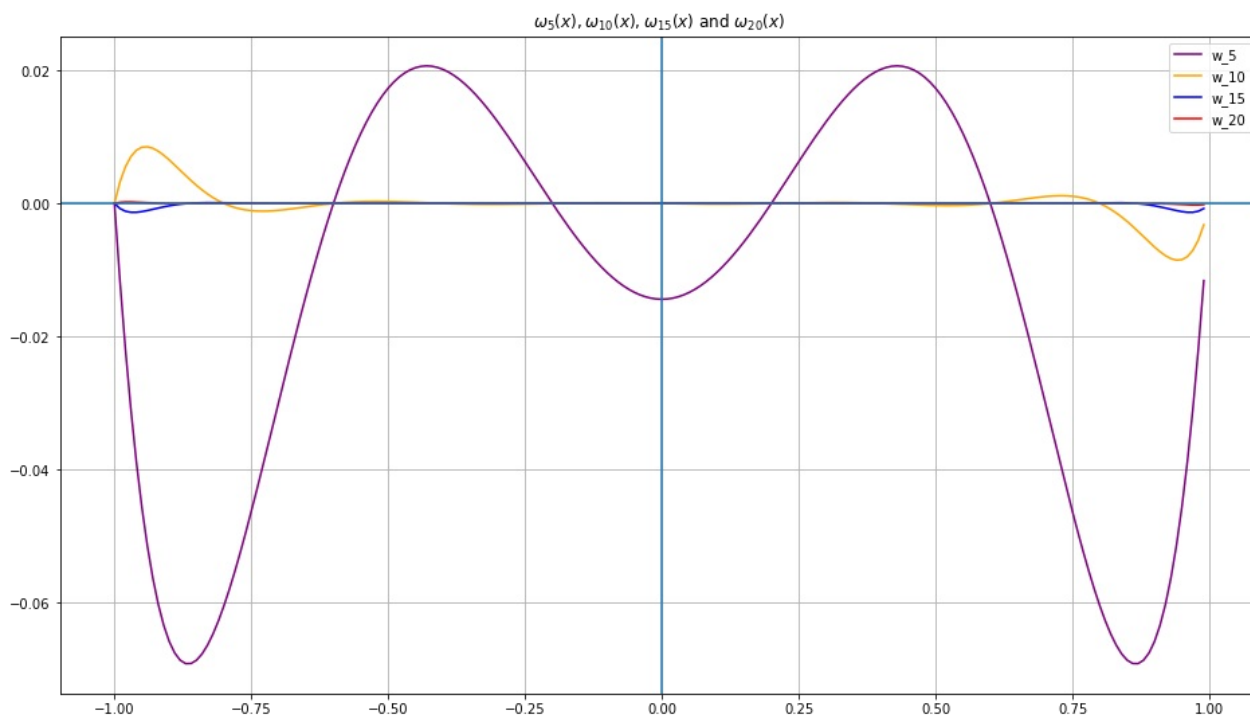
In [7]:

(Top)

```
x_range = np.arange(-1, 1, 0.01)
fig, ax = plt.subplots(figsize=(16, 9))

# Plot the function w_5(x), w_10(x), w_15(x) and w_20(x)
#
# Hint: ax.plot( x_points, y_points, color='?', label='?')
# ===== 請實做程式 =====
ax.plot( x_range, omega_m(x_range, equidistant_points(5)), color='purple', label='w_5')
ax.plot( x_range, omega_m(x_range, equidistant_points(10)), color='orange', label='w_10')
ax.plot( x_range, omega_m(x_range, equidistant_points(15)), color='blue', label='w_15')
ax.plot( x_range, omega_m(x_range, equidistant_points(20)), color='red', label='w_20')
# =====

# Add other text and items
ax.set_title(r'$\omega_{5}(x)$, $\omega_{10}(x)$, $\omega_{15}(x)$ and $\omega_{20}(x)$')
plt.legend(loc='upper right')
ax.grid(True)
ax.axhline(y=0)
ax.axvline(x=0)
plt.show()
```



Part 4. What's your observation of the above figure?

(Top)

用這個分割來估計的話兩邊的誤差較大。

2. Redo " Problem 1. " using **\*zeros of the Chebyshev polynomial (Chebyshev nodes)\*** as the interpolation points.

Part 1. Define Chebyshev nodes.

Please refer the part of Chebyshev nodes in " *lagrange.ipynb* ".

In [8]:

(Top)

```
def chebv_nodes(m):  
    # ===== 請實做程式 =====  
    x = -np.cos(np.linspace(0, np.pi, m+1))  
    points = [xi for xi in x]  
    return(points)  
    # =====
```

In [9]:

(Top)

chebv\_nodes

```
# Test  
m = 5  
print("Chebyshev nodes:", chebv_nodes(m))  
  
### BEGIN HIDDEN TESTS  
m = 10  
assert np.mean(np.array(chebv_nodes(m)) - np.cos(np.linspace(0, np.pi, m+1))) < 1e-7, 'chebv_nodes is wrong!'  
### END HIDDEN TESTS
```

Chebyshev nodes: [-1.0, -0.8090169943749475, -0.30901699437494745, 0.30901699437494734, 0.8090169943749473, 1.0]

Part 2. plot  $\omega(t)$  for  $m = 5, 10, 15, 20$ .

Please refer parts of plotting in " *lagrange.ipynb* ".

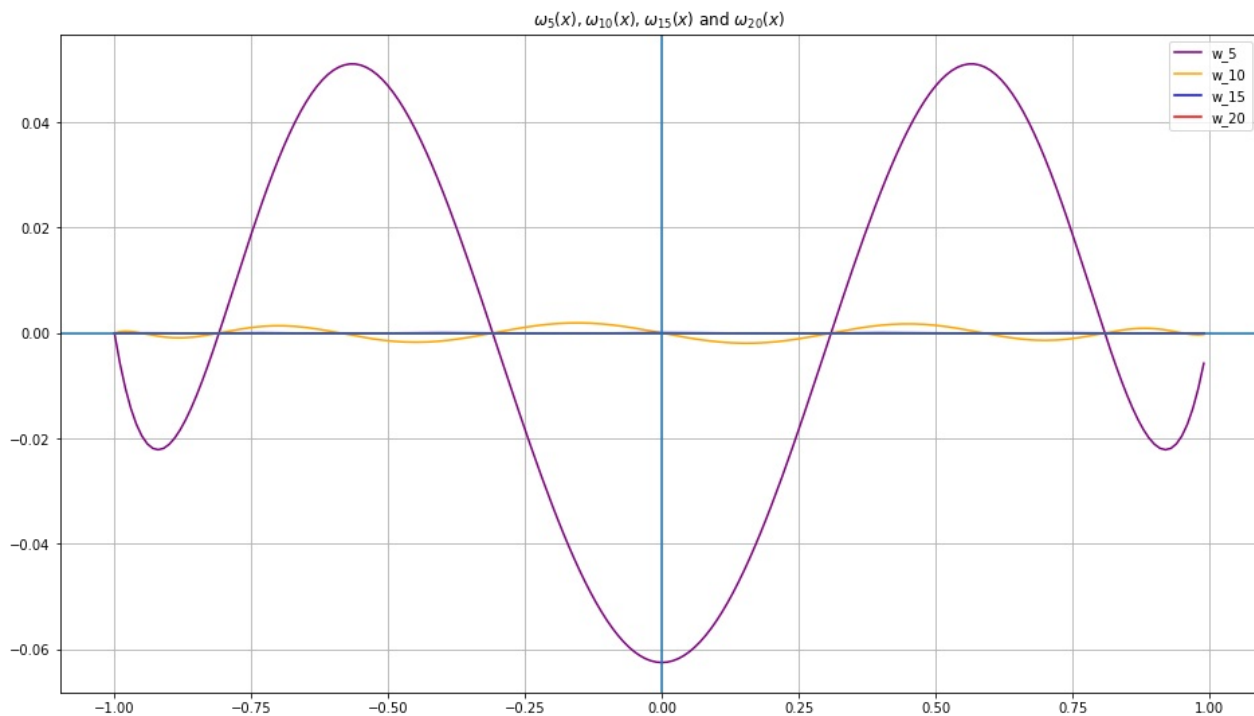
In [10]:

(Top)

```
x_range = np.arange(-1, 1, 0.01)
fig, ax = plt.subplots(figsize=(16, 9))

# Plot the function  $w_5(x)$ ,  $w_{10}(x)$ ,  $w_{15}(x)$  and  $w_{20}(x)$ 
#
# Hint: ax.plot( x_points, y_points, color='?', label='?')
# ===== 請實做程式 =====
ax.plot( x_range, omega_m(x_range,chebv_nodes(5)), color='purple', label='w_5')
ax.plot( x_range, omega_m(x_range,chebv_nodes(10)), color='orange', label='w_10')
ax.plot( x_range, omega_m(x_range,chebv_nodes(15)), color='blue', label='w_15')
ax.plot( x_range, omega_m(x_range,chebv_nodes(20)), color='red', label='w_20')
# =====

# Add other text and items
ax.set_title(r'$\omega_5(x)$, $\omega_{10}(x)$, $\omega_{15}(x)$ and $\omega_{20}(x)$')
plt.legend(loc='upper right')
ax.grid(True)
ax.axhline(y=0)
ax.axvline(x=0)
plt.show()
```



### Part 3. What's your observation of the above figure?

(Top)

用這個分割來估計的話兩邊的誤差和上面的比起來較小。