高等應用數學　　HW3　　黃展皇 110621013

1.

高等應數 hw3. 黃展皇 110621013

① (a)

for Eu-La e.q. $u''+u+x=0$ and B.C. $\boxed{\begin{matrix} u(0)=0 \\ u(1)=0 \end{matrix}}$

$\Rightarrow u \cong \phi_0(x) \stackrel{令}{=} cx(1-x)$

$u=cx(1-x)$，$u''=-2c$，Eu-La e.q $\Rightarrow -2c+cx(1-x)+x=0$

$c(-x^2+x-2)=-x$，$c=\dfrac{x}{x^2-x+2}$，代⊙ u近似

$\Rightarrow \dfrac{x}{x^2-x+2}x(1-x) = \boxed{\dfrac{-x^3+x^2}{x^2-x+2}=u}$ #

(b)

$u''+u+x=0$，$u''+u=-x$，for $y''+py'+qy=r$，$\begin{matrix} p=0 \\ q=1 \\ r=-x \end{matrix}$

$u=u_h+u_p$，u_h solve: $u''+u=0$，$p^2-4q=0-4<0$

$\Rightarrow$ case III，$u_h=\cancel{e^{-\frac{px}{2}}}(A\cos\omega x+B\sin\omega x)$

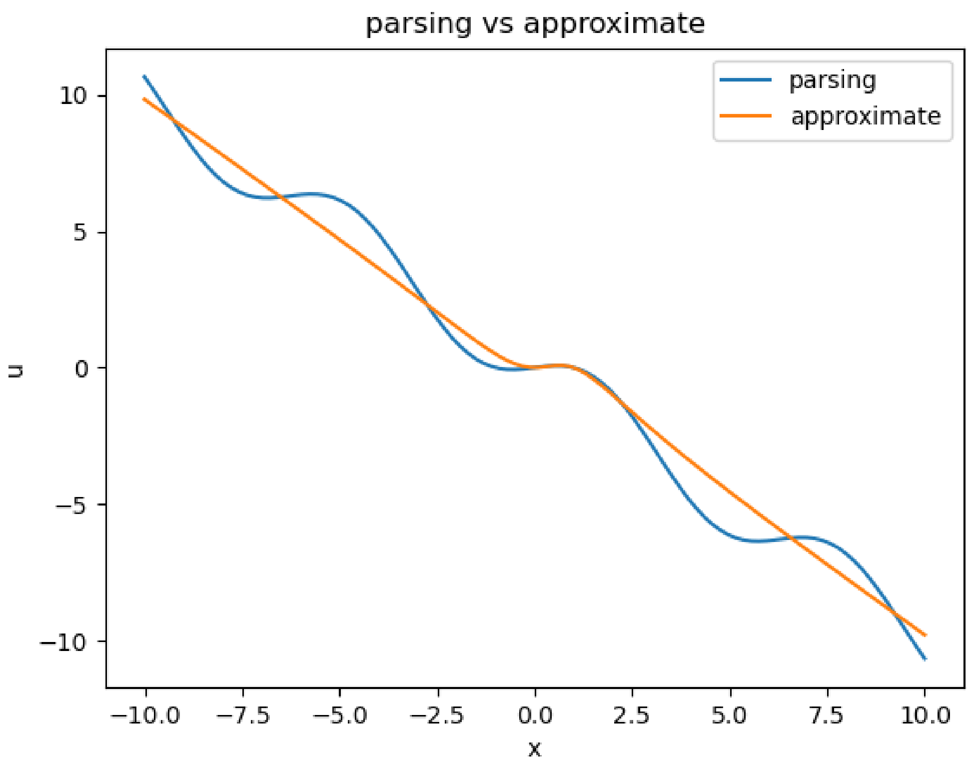
for u_p，$r=-x \in kx^n$，$u_p=-x$，$u=u_h+u_p=A\cos x+B\sin x-x$

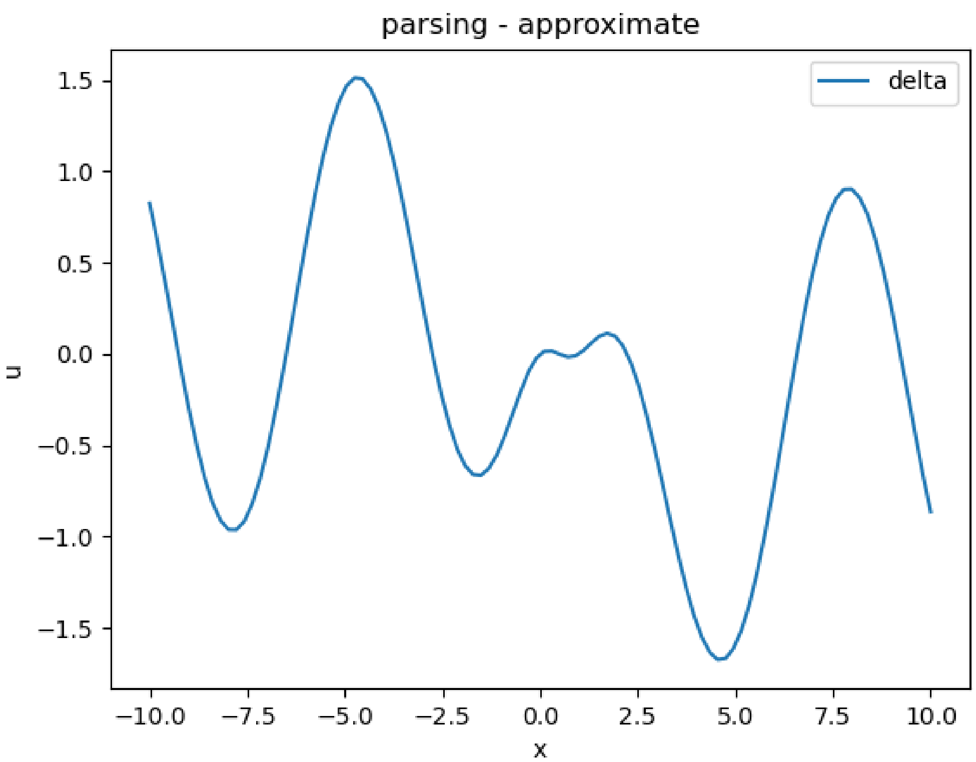
I.C. $\Rightarrow u(0)=0$，$A=0$；$u(1)=0$，$B\sin(1)-1=0$，$B=\dfrac{1}{\sin(1)}$

代⊙，$\boxed{u=\dfrac{\sin x}{\sin(1)}-x}$ #

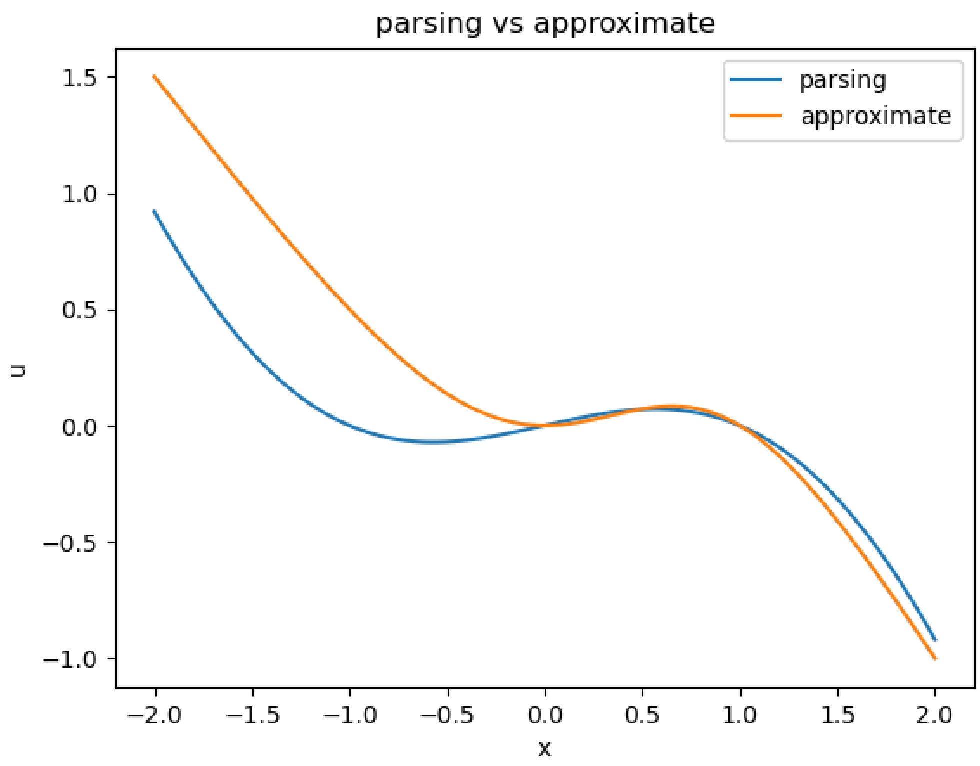
繪圖和 $x>[0.25, 0.5, 0.75]$ 之值比較見後

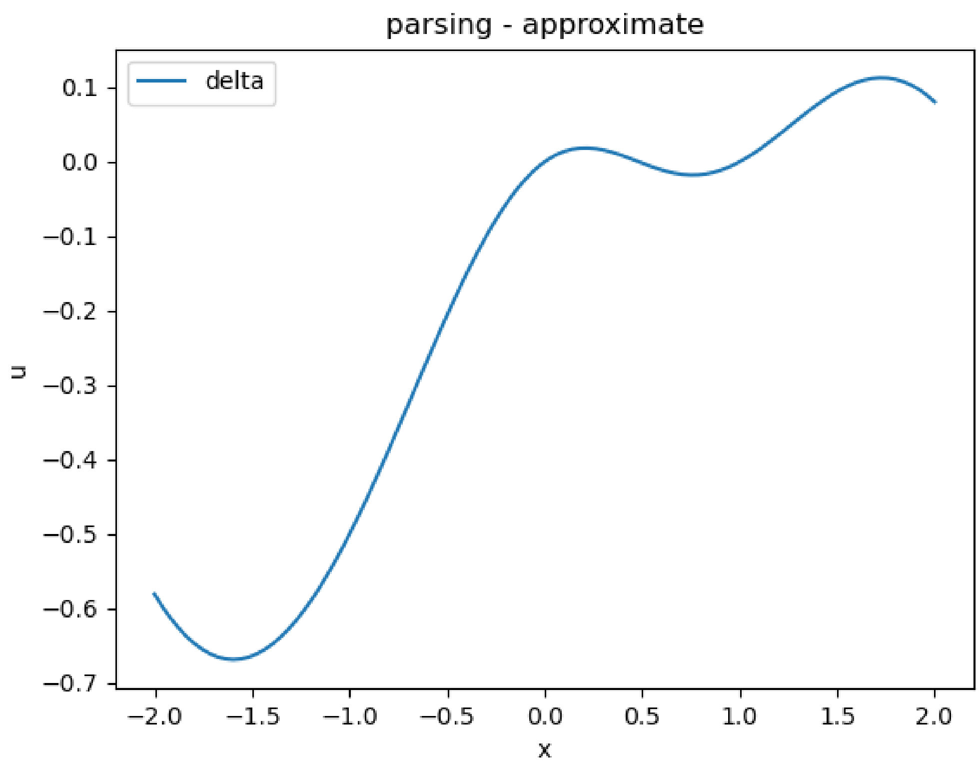
在-10~10 區間的解析解 vs.近似解，以及解析解-近似解的差值：





在-2~2 區間的解析解 vs.近似解，以及解析解-近似解的差值：





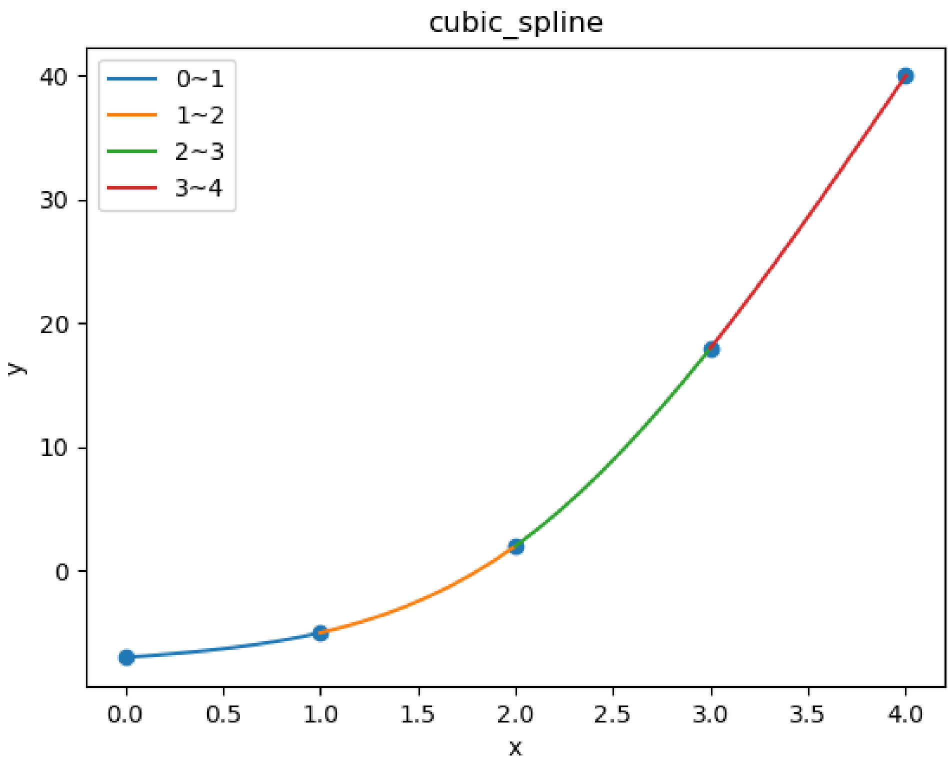
最後可知在 x=[0.25, 0.5, 0.75]時

解析解：[0.0440, 0.0697, 0.0601]

近似解：[0.0259, 0.0714, 0.0776]

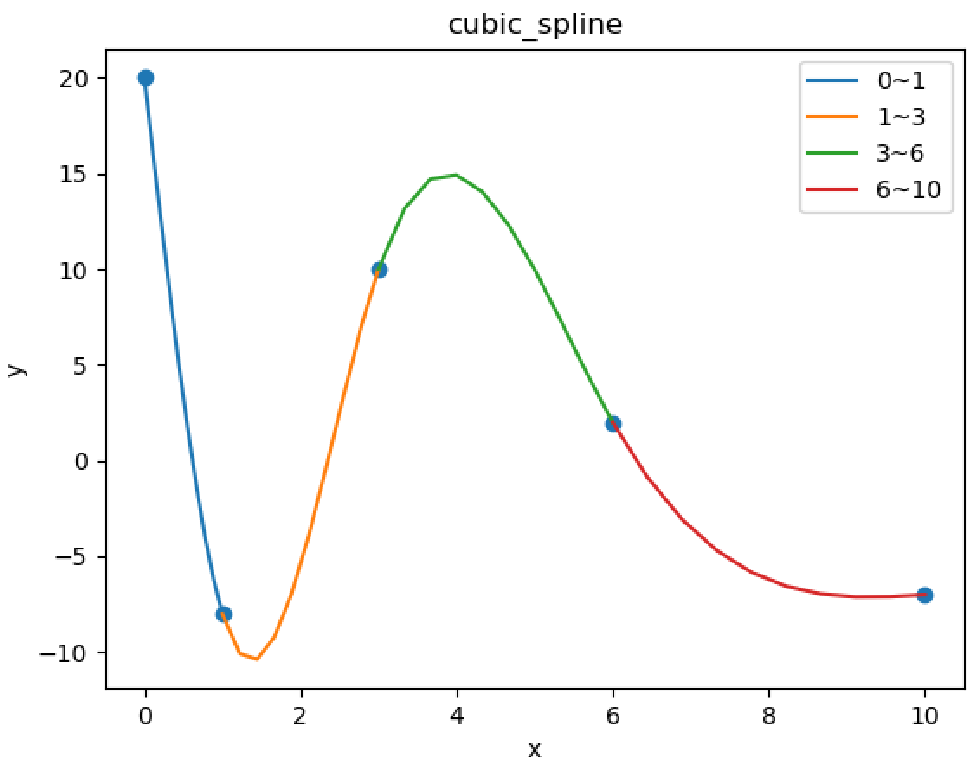
2.如下兩圖，分別為 cubic spline 計算各區段係數與繪圖結果，自然邊界條件

	a	b	c	d
i =0	0.8036	0	1.1964	-7
i =1	0.9821	2.4107	3.6071	-5
i =2	-0.7321	5.3571	11.375	2
i =3	-1.0536	3.1607	19.8929	18



3. 如下兩圖，分別為 cubic spline 計算各區段係數與繪圖結果，自然邊界條件

	a	b	c	d
i =0	7.0906	0	-35.0906	20
i =1	-4.9313	21.2719	-13.8187	-8
i =2	1.1319	-8.3157	12.0936	10
i =3	-0.1559	1.8712	-7.2400	2



提供題 2 題 3 原始碼：(環境 python3.8.10，numpy、matplotlib 套件)

```
import numpy as np

# give arr -> LU, check ok
def LU_decomposition(arr):
    assert len(arr.shape)==2
    assert arr.shape[0]==arr.shape[1]
```

```
    n = arr.shape[0]

    L = np.full_like(arr, 0.0, dtype=np.double)
    for i in range(n):
        L[i][i] = 1.0

    for col in range(n-1):
        for row in range(col+1, n):
            L[row][col] = arr[row][col] / arr[col][col]
            arr[row] = arr[row] - L[row][col]*arr[col]
    return L, arr
# use LU_decomposition to solve linear eq, check ok
def LU_solve_eq(A, B):
    n = B.shape[0]
    L, U = LU_decomposition(A)
    D, m = np.zeros_like(B), np.zeros_like(B)

    # LD = B
    for i in range(n):
        D[i] = B[i]
        for j in range(i+1, n):
            B[j] -= L[j][i] * D[i]

    # Um = D
    for i in range(n-1, -1, -1):
        m[i] = D[i] / U[i][i]
        for j in range(i-1, -1, -1):
            D[j] -= U[j][i] * m[i]
    return m



def get_cubic_spline_coefficient(x, y):
    n = len(x)-1 # intervel nums
    h = [x[i+1]-x[i] for i in range(n)]
    A = np.zeros((n+1, n+1))
    B = np.zeros((n+1))

    # natural condition
```

```
    A[0][0] = 1
    A[n][n] = 1
    for i in range(1, n):
        A[i][i] = 2*(h[i-1]+h[i])
        A[i][i-1] = h[i-1]
        A[i][i+1] = h[i]
        B[i] = 6*((y[i+1]-y[i])/h[i] - (y[i]-y[i-1])/h[i-1])
    m = LU_solve_eq(A, B)

    coefficient = []
    for i in range(n):
        coef = []
        coef.append((m[i+1]-m[i])/6/h[i])
        coef.append(m[i]/2)
        coef.append((y[i+1]-y[i])/h[i] - h[i]*m[i]/2 - h[i]*(m[i+1]-m[i])/6)
        coef.append(y[i])
        coefficient.append(coef)
    return coefficient

def cubic_spline_plot(x, y, coefficient):
    import matplotlib.pyplot as plt
    n = len(x)-1 # intervel nums

    plt.scatter(x, y)
    for i in range(n):
        plot_x = np.linspace(x[i], x[i+1], 10)
        plot_x_to_y = np.linspace(0.0, x[i+1]-x[i], 10)
        plot_y = coefficient[i][0]*plot_x_to_y**3 + coefficient[i][1]*plot_x_to_y**2 +
coefficient[i][2]*plot_x_to_y**1 + coefficient[i][3]*plot_x_to_y**0
        plt.plot(plot_x, plot_y, label='{}~{}'.format(str(x[i]), str(x[i+1])))
    plt.legend()
    plt.title('cubic_spline')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.show()
    plt.close()

if __name__ == '__main__':
```

```
print('gogo')
x = [0, 1, 2, 3, 4]
y = [-7, -5, 2, 18, 40]
coefficient = get_cubic_spline_coefficient(x, y)
print('coefficient:', coefficient)
cubic_spline_plot(x, y, coefficient)

x = [0, 1, 3, 6, 10]
y = [20, -8, 10, 2, -7]
coefficient = get_cubic_spline_coefficient(x, y)
print('coefficient:', coefficient)
cubic_spline_plot(x, y, coefficient)
```