數值方法 作業 11

工科系 115 E14116401 張瑋哲

第一題：

```python
import numpy as np
from scipy.integrate import solve_ivp, quad

# 微分方程 y'' = -(x+1)y' + 2y + (1-x^2)e^(-x)
#          y'' + (x+1)y' - 2y = (1-x^2)e^(-x)
# 若 y'' = f(x, y, y') :
def ode_f_form(x, y_vec):
    # y_vec = [y, y']
    y, yp = y_vec
    ypp = -(x + 1) * yp + 2 * y + (1 - x**2) * np.exp(-x)
    return [yp, ypp]

# 若 y'' = p(x)y' + q(x)y + r(x)
# p(x) = -(x + 1)
def p_func(x):
    return -(x + 1)

# q(x) = 2
def q_func(x):
    return 2

# r(x) = (1-x^2)e^(-x)
def r_func(x):
    return (1 - x**2) * np.exp(-x)

# BC
x_a, x_b = 0.0, 1.0
y_a, y_b = 1.0, 2.0
h = 0.1

x_eval = np.arange(x_a, x_b + h, h)
x_fine_eval = np.linspace(x_a, x_b, 101)

print(f"求解微分方程: y'' = -(x+1)y' + 2y + (1-x^2)e^(-x)")
print(f"邊界條件: y({x_a})={y_a}, y({x_b})={y_b}")
print(f"步長 h = {h}\n")

# IVP 1: y1'' = p(x)y1' + q(x)y1 + r(x), y1(a)=alpha, y1'(a)=0
# y1'' = -(x-1)y1' + 2y1 + (1-x^2)e^(-x)
def ivp1_func(x, y_vec):
    # y_vec = [y1, y1']
    y1, y1p = y_vec
    y1pp = p_func(x) * y1p + q_func(x) * y1 + r_func(x)
    return [y1p, y1pp]
```

```python
# IVP 2: y2'' = p(x)y2' + q(x)y2, y2(a)=0, y2'(a)=1
# y2''= -(x-1)y2' + 2y2
def ivp2_func(x, y_vec):
    # y_vec = [y2, y2']
    y2, y2p = y_vec
    y2pp = p_func(x) * y2p + q_func(x) * y2
    return [y2p, y2pp]

# 初始條件 a=0,b=1
y1_init = [y_a, 0.0]  # [y1(a), y1'(a)]
y2_init = [0.0, 1.0]  # [y2(a), y2'(a)]

sol1 = solve_ivp(ivp1_func, [x_a, x_b], y1_init, dense_output=True, t_eval=x_eval, rtol=1e-8, atol=1e-8)
sol2 = solve_ivp(ivp2_func, [x_a, x_b], y2_init, dense_output=True, t_eval=x_eval, rtol=1e-8, atol=1e-8)

y1_at_b = sol1.y[0, -1] # y1(b)
y2_at_b = sol2.y[0, -1] # y2(b)

c = (y_b - y1_at_b) / y2_at_b
print(f"計算得常數 c = {c:.6f}\n")
# 計算最終解 y(x) = y1(x) + c*y2(x)
y_shooting = sol1.y[0] + c * sol2.y[0]

print("Solution by Shooting method:")
for i, x_val in enumerate(x_eval):
    print(f"y({x_val:.1f}) = {y_shooting[i]:.6f}")
print("-" * 30 + "\n")
```

圖 1、第一題計算式

```
PS D:\ForClass\1132\1132Numerical\HW11>
                                          > & C:/ProgramData/anaconda3/python.exe d:
/ForClass/1132/1132Numerical/HW11/a.py
求解微分方程: y'' = -(x+1)y' + 2y + (1-x^2)e^(-x)
邊界條件: y(0.0)=1.0, y(1.0)=2.0
步長 h = 0.1

計算得常數 c = 0.024158

Solution by Shooting method:
y(0.0) = 1.000000
y(0.1) = 1.016649
y(0.2) = 1.059290
y(0.3) = 1.124473
y(0.4) = 1.209118
y(0.5) = 1.310525
y(0.6) = 1.426374
y(0.7) = 1.554709
y(0.8) = 1.693915
y(0.9) = 1.842688
y(1.0) = 2.000000
------------------------------
```

圖 2、第一題計算結果

第二題：

```python
import numpy as np
from scipy.integrate import solve_ivp, quad

# 微分方程 y'' = -(x+1)y' + 2y + (1-x^2)e^(-x)
#          y'' + (x+1)y' - 2y = (1-x^2)e^(-x)
# 若 y'' = f(x, y, y') :
def ode_f_form(x, y_vec):
    # y_vec = [y, y']
    y, yp = y_vec
    ypp = -(x + 1) * yp + 2 * y + (1 - x**2) * np.exp(-x)
    return [yp, ypp]

# 若 y'' = p(x)y' + q(x)y + r(x)
# p(x) = -(x + 1)
def p_func(x):
    return -(x + 1)

# q(x) = 2
def q_func(x):
    return 2

# r(x) = (1-x^2)e^(-x)
def r_func(x):
    return (1 - x**2) * np.exp(-x)

# BC
x_a, x_b = 0.0, 1.0
y_a, y_b = 1.0, 2.0
h = 0.1

x_eval = np.arange(x_a, x_b + h, h)
x_fine_eval = np.linspace(x_a, x_b, 101)

print(f"求解微分方程: y'' = -(x+1)y' + 2y + (1-x^2)e^(-x)")
print(f"邊界條件: y({x_a})={y_a}, y({x_b})={y_b}")
print(f"步長 h = {h}\n")
```

```python
# 方程: (95 - 5*xi)y_{i-1} - 202*yi + (105 + 5*xi)y_{i+1} = (1-xi^2)e^(-xi)
# y0 = y_a, y_n+1 = y_b (n+1 = (b-a)/h = 10)
# 未知數 y1, ..., y9 (共9個)

N_fd = int((x_b - x_a) / h) -1 # 內部未知點的數量
# 內部節點 x_coords[i] = x_{i+1}
x_coords_fd = np.array([x_a + (i+1)*h for i in range(N_fd)])

# 構建矩陣 A 和向量 F
A_fd = np.zeros((N_fd, N_fd))
F_fd = np.zeros(N_fd)

# 填充矩陣和向量
# 公式: -(1+0.5*h*p_i)y_{i-1} + (2+h^2*q_i)y_i - (1-0.5*h*p_i)y_{i+1} = -h^2*r_i [cite: 2]
# 整理 (95 - 5x_i)y_{i-1} - 202y_i + (105 + 5x_i)y_{i+1} = (1-x_i^2)e^{-x_i}

for i in range(N_fd):
    xi = x_coords_fd[i]

    # 對角線元素係數 for y_i
    A_fd[i, i] = -202.0

    # 次對角線元素係數 for y_{i-1}
    if i > 0:
        A_fd[i, i-1] = 95.0 - 5.0 * xi

    # 超對角線元素係數 for y_{i+1}
    if i < N_fd - 1:
        A_fd[i, i+1] = 105.0 + 5.0 * xi

    # 右端項 F_i
    F_fd[i] = (1 - xi**2) * np.exp(-xi)
```

```python
        # BC:
        if i == 0: # 第一個方程
            # (95 - 5*x1)*y0 - 202*y1 + (105 + 5*x1)*y2 = (1-x1^2)e^(-x1)
            # -202*y1 + (105 + 5*x1)*y2 = (1-x1^2)e^(-x1) - (95 - 5*x1)*y_a
            F_fd[i] -= (95.0 - 5.0 * xi) * y_a
        if i == N_fd - 1: # 最後一個方程
            # (95 - 5*x_N)*y_{N-1} - 202*y_N + (105 + 5*x_N)*y_{N+1} = (1-x_N^2)e^(-x_N)
            # (95 - 5*x_N)*y_{N-1} - 202*y_N = (1-x_N^2)e^(-x_N) - (105 + 5*x_N)*y_b
            F_fd[i] -= (105.0 + 5.0 * xi) * y_b

    # 求解線性系統 {A}{Y} = {F}
    y_internal_fd = np.linalg.solve(A_fd, F_fd)

    # 組合完整解 (包括邊界點)
    y_fd = np.concatenate(([y_a], y_internal_fd, [y_b]))

    print("Solution byFinite-Difference method:")
    for i, x_val in enumerate(x_eval):
        print(f"y({x_val:.1f}) = {y_fd[i]:.6f}")
    print("-" * 30 + "\n")
```

圖 3、第二題計算式

```
PS D:\ForClass\1132\1132Numerical\HW11> & C:/ProgramData/anaconda3/python.exe d:/ForClass/1132/1132Numerical/HW11/B.py
求解微分方程: y'' = -(x+1)y' + 2y + (1-x^2)e^(-x)
邊界條件: y(0.0)=1.0, y(1.0)=2.0
步長 h = 0.1

--- b. Finite-Difference Method ---
Finite-Difference method 近似解:
y(0.0) = 1.000000
y(0.1) = 1.016532
y(0.2) = 1.059102
y(0.3) = 1.124251
y(0.4) = 1.208890
y(0.5) = 1.310313
y(0.6) = 1.426194
y(0.7) = 1.554570
y(0.8) = 1.693822
y(0.9) = 1.842642
y(1.0) = 2.000000
------------------------------
```

圖 4、第二題計算結果

第三題：

```python
import numpy as np
from scipy.integrate import quad, solve_bvp

# 微分方程 y'' = -(x+1)y' + 2y + (1-x^2)e^(-x)
#          y'' + (x+1)y' - 2y = (1-x^2)e^(-x)

# (e^(x^2/2 + x)y')' - 2e^(x^2/2 + x)y = (1-x^2)e^(x^2/2)
# 對應 -(P_s y')' + Q_s y = G_s
# P_s(x) = -e^(x^2/2 + x)
# Q_s(x) = -2e^(x^2/2 + x)
# G_s(x) = (1-x^2)e^(x^2/2)

def P_s_func(x):
    return -np.exp(x**2 / 2 + x)

def Q_s_func(x):
    return -2 * np.exp(x**2 / 2 + x)

def G_s_func(x):
    return (1 - x**2) * np.exp(x**2 / 2)

# y1(x) = 1 + x
def y1_func(x):
    return 1 + x

def y1_prime_func(x):
    return 1.0

# F_eff(x) = G_s(x) - [-d/dx(P_s(x)y1') + Q_s(x)y1(x)]
# -d/dx(P_s(x)y1') = (x+1)e3(x^2/2 + x)
# Q_s(x)y1(x) = -2e3(x^2/2 + x)(1+x)
# So, [-d/dx(P_s(x)y1') + Q_s(x)y1(x)] = -(x+1)e3(x^2/2+x)
def F_eff_func(x):
    term_from_y1 = -(x + 1) * np.exp(x**2 / 2 + x)
    return G_s_func(x) - term_from_y1

# BC
x_a, x_b = 0.0, 1.0
y_a, y_b = 1.0, 2.0
h = 0.1

print(f"求解微分方程: y'' = -(x+1)y' + 2y + (1-x^2)e^(-x)")
print(f"邊界條件: y({x_a})={y_a}, y({x_b})={y_b}")
```

```python
# φ_k(0) = 0, φ_k(1) = 0
phi_funcs = [
    lambda x: x * (1 - x),        # φ_1(x) = x - x^2
    lambda x: x**2 * (1 - x),     # φ_2(x) = x^2 - x^3
    lambda x: x**3 * (1 - x)      # φ_3(x) = x^3 - x^4
]

phi_prime_funcs = [
    lambda x: 1 - 2 * x,          # φ_1'(x)
    lambda x: 2 * x - 3 * x**2,   # φ_2'(x)
    lambda x: 3 * x**2 - 4 * x**3 # φ_3'(x)
]

n_ritz = 3
A = np.zeros((n_ritz, n_ritz))
b_vec = np.zeros(n_ritz)

# 計算矩陣 A 和向量 b
# A_ij = ∫ [ P_s(x)φ_i'(x)φ_j'(x) + Q_s(x)φ_i(x)φ_j(x) ] dx
# b_i = ∫ F_eff(x)φ_i(x) dx
for i in range(n_ritz):
    # 計算 b_i
    integrand_b = lambda x: F_eff_func(x) * phi_funcs[i](x)
    b_vec[i], _ = quad(integrand_b, x_a, x_b)

    for j in range(n_ritz):
        integrand_A = lambda x: (P_s_func(x) * phi_prime_funcs[i](x) * phi_prime_funcs[j](x) +
                                 Q_s_func(x) * phi_funcs[i](x) * phi_funcs[j](x))
        A[i, j], _ = quad(integrand_A, x_a, x_b)

c_coeffs = np.linalg.solve(A, b_vec)
print("\n計算得係數 c:", c_coeffs)

#Solution
def y_approx_ritz(x):
    y2_val = sum(c_coeffs[k] * phi_funcs[k](x) for k in range(n_ritz))
    return y1_func(x) + y2_val

x_eval = np.arange(x_a, x_b + h, h)
y_ritz_eval = np.array([y_approx_ritz(val) for val in x_eval])

print("\nSolution by Variation approach:")
for xi, yi in zip(x_eval, y_ritz_eval):
    print(f"y({xi:.1f}) = {yi:.6f}")
```

圖 5、第三題計算式

```
PS D:\ForClass\1132\1132Numerical\HW11> & C:/ProgramData/anaconda3/python.exe d:/ForClass/1132/1132Numerical/HW11/C3.py
求解微分方程: y'' = -(x+1)y' + 2y + (1-x^2)e^(-x)
邊界條件: y(0.0)=1.0, y(1.0)=2.0

計算得係數 c: [-0.97478616  0.50622565 -0.14540492]

Solution by Variation approach (n=3):
y(0.0) = 1.000000
y(0.1) = 1.016694
y(0.2) = 1.059303
y(0.3) = 1.124439
y(0.4) = 1.209065
y(0.5) = 1.310494
y(0.6) = 1.426385
y(0.7) = 1.554748
y(0.8) = 1.693942
y(0.9) = 1.842674
y(1.0) = 2.000000
```

圖 6、第三題計算結果