Scilab for Chemical Engineers

TSEC - Online Certificate Course

SAMPLE PROBLEMS

Aditya Ganesh Date: 08 - 02 - 2025

Numerical Methods continued...

1 Numerical Integration

Let's consider the following data points:

x	y
0.0	0.0000
0.5	0.4794
1.0	0.8415
1.5	0.9975
2.0	0.9093
2.5	0.5985
3.0	0.1411
3.5	-0.3508
4.0	-0.7568
4.5	-0.9775
5.0	-0.9589
5.5	-0.7055
6.0	-0.2794

Table 1: Discrete data points for numerical integration

1.1 Trapezoidal Method

Compute the integral of the given data points using the trapezoidal method.

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{N-1} \frac{(x_{i+1} - x_i)}{2} \left[f(x_{i+1}) + f(x_i)\right] \tag{1}$$

```
// Trapezoidal Method for discrete data
function I = trapezoidal_data(x, y)
    n = length(x);
    I = 0;
    for k = 1:n-1
        I = I + 0.5 * (x(k+1) - x(k)) * (y(k) + y(k+1));
    end
endfunction

// Data points
x = [0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0,
   5.5, 6.0];
y = [0, 0.4794, 0.8415, 0.9975, 0.9093, 0.5985, 0.1411,
   -0.3508, -0.7568, -0.9775, -0.9589, -0.7055, -0.2794];

I_trap = trapezoidal_data(x, y);
disp(I_trap);
```

1.2 Simpson's 1/3rd Rule

Compute the integral of the given data points using Simpson's 1/3rd rule.

$$\int_a^b f(x)dx \approx \frac{h}{3}[f(x_1)+4f(x_2)+2f(x_3)+4f(x_4)+\ldots+2f(x_{n-2})+4f(x_{n-1})+f(x_n)] \tag{2}$$

$$\int_a^b f(x)\,dx \approx \frac{x_2 - x_1}{3}\left[f(x_1) + f(x_N) + 4\sum_{i=2}^{N-1} f(x_{\text{even}}) + 2\sum_{i=3}^{N-1} f(x_{\text{odd}})\right] \tag{3}$$

```
// Simpson's 1/3rd Rule for discrete data
function Int = Simpsons_one_third(x,y)
    Int = (x(2) - x(1))/3 * (y(1) + y($) + 4* sum(y(2:2:$-1))
        + 2* sum(y(3:2:$-1)));
endfunction

```

```
// Data points
x = [0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0,
   5.5, 6.0];
y = [0, 0.4794, 0.8415, 0.9975, 0.9093, 0.5985, 0.1411,
   -0.3508, -0.7568, -0.9775, -0.9589, -0.7055, -0.2794];

disp(Simpsons_one_third(x,y))
```

1.3 Simpson's 3/8th Rule

Compute the integral of the given data points using Simpson's 3/8th rule.

$$\int_a^b f(x)\,dx \approx \frac{3h}{8}\left(f(x_1) + 3f(x_2) + 3(x_3) + 2(x_4) + 3(x_5) + ...f(x_n)\right) \tag{4}$$

```
// Simpson's 3/8th Rule for discrete data
function Int = Simpsons_three_eighth(x,y)
    Int = 3*(x(2) - x(1))/8 * ( y(1) + y($) + 3*sum(y(2:$-1))
        - sum(y(4:3:$-1)))
endfunction

// Data points
x = [0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0,
   5.5, 6.0];
y = [0, 0.4794, 0.8415, 0.9975, 0.9093, 0.5985, 0.1411,
   -0.3508, -0.7568, -0.9775, -0.9589, -0.7055, -0.2794];

disp(Simpsons_three_eighth(x,y))
```

2 Regression

2.1 Line Fitting (Linear Regression)

Fit a straight line to the following data points and compute the coefficients of the line $y = ax + b$.

Data points:

$$x = [1, 2, 3, 4, 5]$$

$$y = [2, 3, 5, 4, 6]$$

The linear regression formula for the slope a and intercept b is given by:

$$a = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - (\sum x)^2}$$

$$b = \frac{\sum y - a\sum x}{n}$$

The normal equations for linear regression in the matrix form are:

$$\mathbf{AX} = \mathbf{B}$$

Where:

$$\mathbf{A} = \begin{bmatrix} \sum x^2 & \sum x \\ \sum x & n \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \sum xy \\ \sum y \end{bmatrix}$$

```
function [a, b] = linear_regression(x, y)
    n = length(x);

    // Calculate sums
    sum_x = sum(x);
    sum_y = sum(y);
    sum_xy = sum(x .* y);
    sum_x2 = sum(x .^ 2);

    // Calculate slope (a) and intercept (b)
    a = (n * sum_xy - sum_x * sum_y) / (n * sum_x2 - sum_x^2)
       ;
    b = (sum_y - a * sum_x) / n;
endfunction

```

```
// Data points
x = [1, 2, 3, 4, 5];
y = [2, 3, 5, 4, 6];

// Perform linear regression
[a, b] = linear_regression(x, y);

disp("Coefficients of the line: ");
disp("Slope (a): " + string(a));
disp("Intercept (b): " + string(b));
```

2.2 Polynomial Fitting (Quadratic Polynomial)

Fit a quadratic polynomial $y = ax^2 + bx + c$ to the following data points and compute the coefficients.

Data points:

$$x = [1, 2, 3, 4, 5]$$

$$y = [2, 3, 5, 4, 6]$$

The normal equations for quadratic polynomial fitting are.

$$\mathbf{A} = \begin{bmatrix} \sum x^4 & \sum x^3 & \sum x^2 \\ \sum x^3 & \sum x^2 & \sum x \\ \sum x^2 & \sum x & n \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \sum x^2 y \\ \sum xy \\ \sum y \end{bmatrix}$$

Solve $\mathbf{AX} = \mathbf{B}$:

```
function [a, b, c] = polynomial_regression(x, y)
    n = length(x);

```

```
    // Calculate sums
    sum_x = sum(x);
    sum_y = sum(y);
    sum_x2 = sum(x .^ 2);
    sum_x3 = sum(x .^ 3);
    sum_x4 = sum(x .^ 4);
    sum_xy = sum(x .* y);
    sum_x2y = sum((x .^ 2) .* y);

    // Construct matrices
    A = [sum_x4, sum_x3, sum_x2;
         sum_x3, sum_x2, sum_x;
         sum_x2, sum_x, n];
    B = [sum_x2y; sum_xy; sum_y];

    // Solve for coefficients
    X = A \ B;

    a = X(1);
    b = X(2);
    c = X(3);
endfunction

// Data points
x = [1, 2, 3, 4, 5];
y = [2, 3, 5, 4, 6];

// Perform polynomial regression
[a, b, c] = polynomial_regression(x, y);

disp("Coefficients of the quadratic polynomial: ");
disp("a: " + string(a));
disp("b: " + string(b));
disp("c: " + string(c));
```

3 Ordinary Differential Equations

3.1 Euler's Method

Solve the first-order ODE $\frac{dy}{dt} = -2y$ with initial condition $y(0) = 1$.
Hint:

$$y_{n+1} = y_n + hf(x_n, y_n) \tag{5}$$

```
// Euler's Method for solving ODE
function [t, y] = euler_method(f, t0, y0, tf, h)
    t = t0:h:tf;
    y = zeros(1, length(t));
    y(1) = y0;
    for i = 1:length(t)-1
        y(i+1) = y(i) + h * f(t(i), y(i));
    end
endfunction

// Define the ODE as a function
function dydt = Euler_ode(t, y)
    dydt = -2 * y;
endfunction

// Initial condition
t0 = 0;
y0 = 1;
tf = 5;
h = 0.1;

// Solve the ODE using Euler's method
[t, y] = euler_method(Euler_ode, t0, y0, tf, h);

// Plot the solution
plot(t, y);
xlabel("t");
ylabel("y(t)");
title("Solution of the Simple ODE using Euler''s Method");
```

3.2 Runge-Kutta Method

Solve the first-order ODE $\frac{dy}{dt} = y - t^2 + 1$ with initial condition $y(0) = 0.5$.

Hint:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = hf(x_i, y_i)$$

$$k_2 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right)$$

$$k_3 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right)$$

$$k_4 = hf(x_i + h, y_i + k_3)$$

```
// Runge-Kutta Method for solving ODE
function [t, y] = runge_kutta(f, t0, y0, tf, h)
    t = t0:h:tf;
    y = zeros(1, length(t));
    y(1) = y0;
    for i = 1:length(t)-1
        k1 = h * f(t(i), y(i));
        k2 = h * f(t(i) + h/2, y(i) + k1/2);
        k3 = h * f(t(i) + h/2, y(i) + k2/2);
        k4 = h * f(t(i) + h, y(i) + k3);
        y(i+1) = y(i) + (k1 + 2*k2 + 2*k3 + k4) / 6;
    end
endfunction

// Define the ODE as a function
function dydt = RK_ode(t, y)
    dydt = y - t^2 + 1;
endfunction

// Initial condition
t0 = 0;
y0 = 0.5;
```

```
tf = 2;
h = 0.1;

// Solve the ODE using Runge-Kutta method
[t, y] = runge_kutta(RK_ode, t0, y0, tf, h);

// Plot the solution
plot(t, y);
xlabel("t");
ylabel("y(t)");
title("Solution of the Intermediate ODE using Runge-Kutta
   Method");
```

3.3 2nd order differential equation: Runge-Kutta Method

Solve the second-order ODE $\frac{d^2y}{dt^2} + 3\frac{dy}{dt} + 2y = 0$ with initial conditions $y(0) = 0$ and $\frac{dy}{dt}(0) = 1$.

Hint: Convert the second-order ODE into a system of first-order ODEs:

$$\frac{dy_1}{dt} = y_2$$

$$\frac{dy_2}{dt} = -3y_2 - 2y_1$$

```
// Runge-Kutta Method for solving ODE
function [t, y] = runge_kutta(f, t0, y0, tf, h)
    t = t0:h:tf;
    y = zeros(length(y0), length(t));
    y(:, 1) = y0;
    for i = 1:length(t)-1
        k1 = h * f(t(i), y(:, i));
        k2 = h * f(t(i) + h/2, y(:, i) + k1/2);
        k3 = h * f(t(i) + h/2, y(:, i) + k2/2);
        k4 = h * f(t(i) + h, y(:, i) + k3);
        y(:, i+1) = y(:, i) + (k1 + 2*k2 + 2*k3 + k4) / 6;
    end
endfunction
```

```
14
15 // Define the system of ODEs as a function
16 function dydt = RK_2_ode(t, y)
17     dydt = [y(2); -3 * y(2) - 2 * y(1)];
18 endfunction
19
20 // Initial conditions
21 t0 = 0;
22 y0 = [0; 1];
23 tf = 5;
24 h = 0.1;
25
26 // Solve the ODE using Runge-Kutta method
27 [t, y] = runge_kutta(RK_2_ode, t0, y0, tf, h);
28
29 // Plot the solution
30 plot(t, y(1, :));
31 xlabel("t");
32 ylabel("y(t)");
33 title("Solution of the Complex ODE using Runge-Kutta Method")
   ;
```