MATH 211.3 Winter Term 2024 Assignment

Assignment #07

Name: Kanwar Pannu

Student Number: 11336732

NSID: eza147

**PROBLEM 1  
A close-up of a paper

Description automatically generated**

**A close-up of a paper with mathematical equations

Description automatically generated**

**1 c e h**

clear;

clc;

%integrand functions

f1 = @(x) x .\* exp(x);

f2 = @(x) x.^2 .\* sin(x);

f3 = @(x) x ./ sqrt(x.^4 + 1);

%limits of integration

a = [0, 0, 0];

b = [1, pi, 1];

%panels

m1 = 16;

m2 = 32;

h1 = (b - a) / m1;

h2 = (b - a) / m2;

x1 = [a; a+h1; b];

x2 = [a; a+h2; b];

Fx1 = [f1(x1(1,:)); f1(x1(2,:)); f1(x1(3,:))];

Fx2 = [f2(x1(1,:)); f2(x1(2,:)); f2(x1(3,:))];

Fx3 = [f3(x1(1,:)); f3(x1(2,:)); f3(x1(3,:))];

T1 = h1 .\* (Fx1(1,:) + 2 .\* sum(Fx1(2:end-1,:)) + Fx1(end,:)) ./ 2;

T2 = h2 .\* (Fx3(1,:) + 2 .\* sum(Fx3(2:end-1,:)) + Fx3(end,:)) ./ 2;

I\_exact = [integral(f1, a(1), b(1)), integral(f2, a(2), b(2)), integral(f3, a(3), b(3))];

error1 = abs(I\_exact - T1);

error2 = abs(I\_exact - T2);

% Display the results

disp(['Trapezoidal Rule results with m=16: ', num2str(T1)]);

disp(['Trapezoidal Rule results with m=32: ', num2str(T2)]);

disp(['Exact integral values: ', num2str(I\_exact)]);

disp(['Errors with m=16: ', num2str(error1)]);

disp(['Errors with m=32: ', num2str(error2)]);

**A screenshot of a computer

Description automatically generated**

**3 b g h**

clear;

clc;

%integrand functions

f4 = @(x) sin(x.^2);

f5 = @(x) x.^x;

f6 = @(x) log(cos(x) + sin(x));

a = [0, 0, 0];

b = [sqrt(pi), 1, pi/2];

%panels

m1 = 16;

m2 = 32;

h1 = (b - a) / m1;

h2 = (b - a) / m2;

T1\_16 = h1(1) / 2 \* (f4(a(1)) + 2 \* sum(arrayfun(f4, a(1)+h1(1):h1(1):b(1)-h1(1))) + f4(b(1)));

T1\_32 = h1(2) / 2 \* (f5(a(2)) + 2 \* sum(arrayfun(f5, a(2)+h2(2):h2(2):b(2)-h2(2))) + f5(b(2)));

T1\_64 = h1(3) / 2 \* (f6(a(3)) + 2 \* sum(arrayfun(f6, a(3)+h2(3):h2(3):b(3)-h2(3))) + f6(b(3)));

T2\_16 = h2(1) / 2 \* (f4(a(1)) + 2 \* sum(arrayfun(f4, a(1)+h1(1):h1(1):b(1)-h1(1))) + f4(b(1)));

T2\_32 = h2(2) / 2 \* (f5(a(2)) + 2 \* sum(arrayfun(f5, a(2)+h2(2):h2(2):b(2)-h2(2))) + f5(b(2)));

T2\_64 = h2(3) / 2 \* (f6(a(3)) + 2 \* sum(arrayfun(f6, a(3)+h2(3):h2(3):b(3)-h2(3))) + f6(b(3)));

% Display the results for m=16

fprintf('Approximation with m=16 for first integral: %f\n', T1\_16);

fprintf('Approximation with m=16 for second integral: %f\n', T1\_32);

fprintf('Approximation with m=16 for third integral: %f\n', T1\_64);

% Display the results for m=32

fprintf('Approximation with m=32 for first integral: %f\n', T2\_16);

fprintf('Approximation with m=32 for second integral: %f\n', T2\_32);

fprintf('Approximation with m=32 for third integral: %f\n', T2\_64);

**A screenshot of a computer

Description automatically generated**

**5**

clear;

clc;

ms = [10, 100, 1000];

%integrands for (b), (g), and (h)

f\_b = @(x) sin(x.^2);

f\_g = @(x) x.^x;

f\_h = @(x) log(cos(x) + sin(x));

for part = {'b', 'g', 'h'}

for m = ms

switch char(part)

case 'b'

a = 0; b = sqrt(pi);

f = f\_b;

exact\_value = integral(f\_b, a, b);

case 'g'

a = 0; b = 1;

f = f\_g;

exact\_value = integral(f\_g, a, b);

case 'h'

a = 0; b = pi/2;

f = f\_h;

exact\_value = integral(f\_h, a, b);

end

I = midpoint\_rule(f, a, b, m);

error = abs(I - exact\_value);

fprintf('Part (%s) using m=%d: Integral Approx = %f, Error = %f\n', char(part), m, I, error);

end

end

%midpoint rule function

function I = midpoint\_rule(f, a, b, m)

h = (b - a) / m;

x = linspace(a + h/2, b - h/2, m);

I = h \* sum(f(x));

end

**A black and white text on a white background

Description automatically generated**

**7a**

clear;

clc;

midpoint\_rule = @(f, a, b, m) (b - a) / m \* sum(f(linspace(a + (b - a) / (2\*m), b - (b - a) / (2\*m), m)));

f = @(x) x ./ sin(x);

%interval

a = 0;

b = pi / 2;

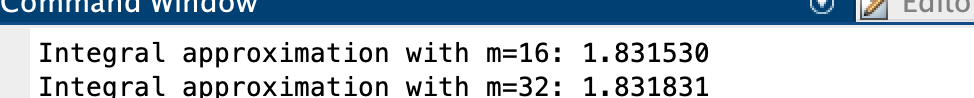
I\_m16 = midpoint\_rule(f, a, b, 16);

I\_m32 = midpoint\_rule(f, a, b, 32);

% Display the results

fprintf('Integral approximation with m=16: %f\n', I\_m16);

fprintf('Integral approximation with m=32: %f\n', I\_m32);

****

**PROBLEM 2A close-up of a paper

Description automatically generated**

**A paper with math equations

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A close-up of a paper with mathematical equations

Description automatically generated**

**1**

clear;

clc;

%(a): y' = t

dydt\_a = @(t, y) t;

% (c): y' = 2(t + 1)y

dydt\_c = @(t, y) 2 \* (t + 1) \* y;

% (f): y' = t^3 / y^2

dydt\_f = @(t, y) t^3 / y^2;

% Placeholder for the exact solution functions

exact\_solution\_a = @(t) nan;

exact\_solution\_c = @(t) nan;

exact\_solution\_f = @(t) nan;

t0 = 0;

y0\_a = 1;

y0\_c = 1;

y0\_f = 1;

t\_end = 1;

h = 0.1;

[t\_values\_a, y\_values\_a, errors\_a] = euler\_method(dydt\_a, t0, y0\_a, t\_end, h, exact\_solution\_a);

[t\_values\_c, y\_values\_c, errors\_c] = euler\_method(dydt\_c, t0, y0\_c, t\_end, h, exact\_solution\_c);

[t\_values\_f, y\_values\_f, errors\_f] = euler\_method(dydt\_f, t0, y0\_f, t\_end, h, exact\_solution\_f);

% Euler's method function

function [t\_values, y\_values, errors] = euler\_method(dydt, t0, y0, t\_end, h, exact\_solution)

t\_values = t0:h:t\_end;

y\_values = zeros(size(t\_values));

errors = zeros(size(t\_values));

y\_values(1) = y0;

errors(1) = y\_values(1) - exact\_solution(t\_values(1));

for i = 2:numel(t\_values)

y\_values(i) = y\_values(i-1) + h \* dydt(t\_values(i-1), y\_values(i-1));

errors(i) = y\_values(i) - exact\_solution(t\_values(i));

end

% Display results in a table

T = table(t\_values', y\_values', errors', 'VariableNames', {'t', 'Euler Approximation', 'Error'});

disp(T);

end

**t Euler Approximation Error**

**\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_**

**0 1 NaN**

**0.1 1 NaN**

**0.2 1.01 NaN**

**0.3 1.03 NaN**

**0.4 1.06 NaN**

**0.5 1.1 NaN**

**0.6 1.15 NaN**

**0.7 1.21 NaN**

**0.8 1.28 NaN**

**0.9 1.36 NaN**

**1 1.45 NaN**

**t Euler Approximation Error**

**\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_**

**0 1 NaN**

**0.1 1.2 NaN**

**0.2 1.464 NaN**

**0.3 1.8154 NaN**

**0.4 2.2874 NaN**

**0.5 2.9278 NaN**

**0.6 3.8062 NaN**

**0.7 5.0241 NaN**

**0.8 6.7323 NaN**

**0.9 9.156 NaN**

**1 12.635 NaN**

**t Euler Approximation Error**

**\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_**

**0 1 NaN**

**0.1 1 NaN**

**0.2 1.0001 NaN**

**0.3 1.0009 NaN**

**0.4 1.0036 NaN**

**0.5 1.0099 NaN**

**0.6 1.0222 NaN**

**0.7 1.0429 NaN**

**0.8 1.0744 NaN**

**0.9 1.1188 NaN**

**1 1.177 NaN**

**>>**

**3**

clear;

clc;

dydt\_b = @(t, y) t - y;

exact\_solution\_b = @(t) exp(-t) .\* (exp(t) - 1);

t0 = 0;

y0 = 1;

t\_end = 1;

h\_values = [0.1, 0.05, 0.025];

figure;

hold on;

colors = ['r', 'g', 'b'];

for i = 1:length(h\_values)

h = h\_values(i);

y\_approx = euler\_method(dydt\_b, t0, y0, t\_end, h);

t\_approx = t0:h:t\_end;

plot(t\_approx, y\_approx, colors(i), 'DisplayName', sprintf('h = %.3f', h));

end

t\_fine = t0:0.001:t\_end;

y\_exact = exact\_solution\_b(t\_fine);

plot(t\_fine, y\_exact, 'k', 'LineWidth', 2, 'DisplayName', 'Exact Solution');

xlabel('t');

ylabel('y');

legend;

title('Euler''s Method Approximations and Exact Solution');

grid on;

hold off;

% Euler's method function

function y\_values = euler\_method(dydt, t0, y0, t\_end, h)

t\_values = t0:h:t\_end;

y\_values = zeros(size(t\_values));

y\_values(1) = y0;

for i = 2:numel(t\_values)

y\_values(i) = y\_values(i-1) + h \* dydt(t\_values(i-1), y\_values(i-1));

end

end

**A screen shot of a graph

Description automatically generated**

**5**

clear;

clc;

% Define the differential equation

dydt\_b = @(t, y) t - y;

exact\_solution\_b = @(t) exp(-t) + t - 1;

errors = zeros(1, 6);

h\_values = zeros(1, 6);

% Calculate errors for different step sizes

y0 = 0;

for k = 0:5

h = 0.1 \* 2^(-k);

h\_values(k+1) = h;

y\_approx = euler\_method\_end(dydt\_b, y0, h);

y\_exact = exact\_solution\_b(1);

errors(k+1) = abs(y\_exact - y\_approx);

end

figure;

loglog(h\_values, errors, 'bo-', 'LineWidth', 2, 'MarkerSize', 8);

xlabel('Step size h');

ylabel('Error at t = 1');

title('Log-Log Plot of Error vs Step Size for Euler’s Method');

grid on;

% Euler's method function to return the value at t=1

function y\_end = euler\_method\_end(dydt, y0, h)

t = 0:h:1;

y = y0;

for i = 1:length(t)-1

y = y + h \* dydt(t(i), y);

end

y\_end = y;

end

**A graph with a line

Description automatically generated**

**7**

clear;

clc;

% Differential equation

dydt = @(y) 1 + y^2;

exact\_solution\_y0\_0 = @(t) tan(t);

exact\_solution\_y0\_1 = @(t) tan(t + pi/4);

t0 = 0; t\_end = 1;

step\_sizes = [0.1, 0.05];

y0\_values = [0, 1];

figure;

hold on;

for y0 = y0\_values

% Exact solution plot

t\_fine = t0:0.001:t\_end;

if y0 == 0

y\_exact = exact\_solution\_y0\_0(t\_fine);

else

y\_exact = exact\_solution\_y0\_1(t\_fine);

end

plot(t\_fine, y\_exact, 'DisplayName', sprintf('Exact y(0)=%d', y0));

% Loop over step sizes

for h = step\_sizes

t\_values = t0:h:t\_end;

y\_approx = euler\_method(dydt, y0, t0, t\_end, h);

plot(t\_values, y\_approx, 'o-', 'DisplayName', sprintf('Euler y(0)=%d, h=%.2f', y0, h));

end

end

xlabel('t');

ylabel('y(t)');

title('Euler’s Method Approximation vs. Exact Solution');

legend('Location', 'best');

grid on;

hold off;

% Euler's method function

function y\_values = euler\_method(dydt, y0, t0, t\_end, h)

t\_values = t0:h:t\_end;

y\_values = zeros(size(t\_values));

y\_values(1) = y0;

for i = 1:(length(t\_values) - 1)

y\_values(i + 1) = y\_values(i) + h \* dydt(y\_values(i));

end

end

**A screen shot of a graph

Description automatically generated**