MATH 211.3 Winter Term 2024 Assignment

Assignment #02

Name: Kanwar Pannu

Student Number: 11336732

NSID: eza147

PROBLEM 1

**A close-up of a piece of paper

Description automatically generated**

**1c**

clear;

clc;

% Function handle for g(x)

g = @(x) log(4 - sin(x));

% Initial guess (x0)

x0 = 1.5;

% Number of iteration steps (k)

k = 100;

% Call the fixed-point iteration function

xc = fpi(g, x0, k);

% Display the result

disp(['Approximate solution: ' num2str(xc)]);

%Program 1.2 Fixed-Point Iteration

%Computes approximate solution of g(x)=x

%Input: function handle g, starting guess x0,

% number of iteration steps k

%Output: Approximate solution xc

function xc=fpi(g, x0, k)

x(1)=x0;

for i=1:k

x(i+1)=g(x(i));

end

xc=x(k+1);

end

**3b**

clear;

clc;

% Function handle for g(x)

g = @(x) 0.5 \* (x + 5 / x);

%sqrt of 5 can be represented as avg of x and 5/x

% Initial guess (x0)

x0 = 2;

% Number of iteration steps (k)

k = 100;

% Call the fixed-point iteration function

xc = fpi(g, x0, k);

% Display the result

disp(['Approximate solution: ' num2str(xc)]);

%Program 1.2 Fixed-Point Iteration

%Computes approximate solution of g(x)=x

%Input: function handle g, starting guess x0,

% number of iteration steps k

%Output: Approximate solution xc

function xc=fpi(g, x0, k)

x(1)=x0;

for i=1:k

x(i+1)=g(x(i));

end

xc=x(k+1);

end

**5**

clear;

clc;

g = @(x) cos(x)^2;

% Initial guess (x0)

x0 = 0.5;

% Number of iteration steps (k)

k = 1000;

% Call the fixed-point iteration function

xc = fpi(g, x0, k);

% Display the result

disp(['Approximate solution: ' num2str(xc)]);

%Program 1.2 Fixed-Point Iteration

%Computes approximate solution of g(x)=x

%Input: function handle g, starting guess x0,

% number of iteration steps k

%Output: Approximate solution xc

function xc=fpi(g, x0, k)

x(1)=x0;

for i=1:k

x(i+1)=g(x(i));

end

xc=x(k+1);

end

PROBLEM 2

**A close-up of a math problem

Description automatically generated**

**A close-up of a paper with writing

Description automatically generated**

A close-up of a paper with mathematical equations

Description automatically generated

**1c**

clear;

clc;

f = @(x) exp(x) + sin(x) - 4;

% Function handle for f'(x)

df = @(x) exp(x) + cos(x);

% Initial guess (x0)

x0 = 1.5;

% Tolerance for convergence

tol = 1e-8;

% Maximum number of iterations

max\_iter = 100;

% Call the Newton's method function

xc = newton(f, df, x0, tol, max\_iter);

disp(['Approximate solution: ' num2str(xc)]);

function xc = newton(f, df, x0, tol, max\_iter)

x = x0;

for i = 1:max\_iter

x\_new = x - f(x) / df(x);

if abs(x\_new - x) < tol

xc = x\_new;

return;

end

x = x\_new;

end

xc = x; % Return the last value if max\_iter is reached

end

**3a**

**A graph with a line drawn on it

Description automatically generated**

**5**

clear;

clc;

% Given values

total\_volume = 400; % Total volume in cubic meters

height = 10; % Height of the cylinder in meters

V = @(r) pi \* r^2 \* height + (2/3) \* pi \* r^3 - total\_volume;

dV = @(r) 2 \* pi \* r \* height + 2 \* pi \* r^2;

% Initial guess for the radius

r0 = 5;

% Tolerance for convergence

tol = 1e-4;

% Maximum number of iterations

max\_iter = 100;

% Call the Newton's method function

radius = newton(V, dV, r0, tol, max\_iter);

disp(['Base radius of the silo: ' num2str(radius)]);

function xc = newton(f, df, x0, tol, max\_iter)

x = x0;

for i = 1:max\_iter

x\_new = x - f(x) / df(x);

if abs(x\_new - x) < tol

xc = x\_new;

return;

end

x = x\_new;

end

xc = x; % Return the last value if max\_iter is reached

end

PROBLEM 3

**A close-up of a math problem

Description automatically generated**

**1c**

clear;

clc;

f = @(x) exp(x) + sin(x) - 4;

% Initial guesses (x0 and x1)

x0 = 1;

x1 = 2;

% Tolerance for convergence

tol = 1e-8;

% Maximum number of iterations

max\_iter = 100;

% Call the Secant method function

xc = secant(f, x0, x1, tol, max\_iter);

disp(['Approximate solution: ' num2str(xc)]);

function xc = secant(f, x0, x1, tol, max\_iter)

for i = 1:max\_iter

f\_x0 = f(x0);

f\_x1 = f(x1);

x2 = x1 - f\_x1 \* (x1 - x0) / (f\_x1 - f\_x0);

if abs(x2 - x1) < tol

xc = x2;

return;

end

x0 = x1;

x1 = x2;

end

xc = x1; % Return the last value if max\_iter is reached

end