MATH 446: Project 04

Zachary Ferguson

Contents

Code

Secant Method

```
% Computes the roots of a function using the Secant Method.
% Written by Zachary Ferguson

function xc = secant_method(f, x0, x1, tol)
    % Compute the root to f(x) using the Secant Method
    % Input:
    %   f - function to find the roots of
    %   x0, x1 - intial guesses
    %   tol - tolerance for the root
    % Output:
    %   xc - computed root to the function f(x).
    if nargin < 4
        tol = 1e-9;
    end

    n = 0;
    xi = x1;
    xi_1 = x0;
    while (abs(f(xi)) >= 0.5 * tol)
        x = xi - (f(xi) * (xi - xi_1)) / (f(xi) - f(xi_1));
        xi_1 = xi;
        xi = x;
        n = n + 1;
    end
    fprintf('\tn = %d\n', n)
    xc = xi;
end
```

Method of False Position

```
% Computes the roots of a function using the Method of False Position.
% Written by Zachary Ferguson

function xc = method_of_false_position(f, x0, x1, tol)
    % Compute the root to f(x) using the Method of False Position
```

```
    % Input:
    %   f - function to find the roots of
    %   x0, x1 - intial guess range (f(x0)*f(x1) < 0)
    %   tol - tolerance for the root
    % Output:
    %   xc - computed root to the function f(x).
    if nargin < 4
        tol = 1e-9;
    end

    n = 0;
    a = x0;
    b = x1;
    c = b;
    while (abs(f(c)) >= 0.5 * tol)
        if (f(a) * f(b)) < 0
            b = c;
        else
            a = c;
        end
        % Next value of c closer to the root
        c = (b * f(a) - a * f(b)) / (f(a) - f(b));
        n = n + 1;
    end
    fprintf('\tn = %d\n', n);
    xc = c;
end
```

Inverse Quadratic Interpolation

```
% Computes the roots of a function using the Inverse Quadratic Interpolation.
% Written by Zachary Ferguson

function xc = inverse_quadratic_interpolation(f, x0, x1, x2, tol)
    % Compute the root to f(x) using the Inverse Quadratic Interpolation
    % Input:
    %   f - function to find the roots of
    %   x0, x1, x2 - intial guess defining the parabola
    %   tol - tolerance for the root
    % Output:
    %   xc - computed root to the function f(x).
    if nargin < 5
        tol = 1e-9;
    end

    n = 0;
    a = x0;
    b = x1;
    c = x2;
    div = @(x,y) (f(x) / f(y));
    while (abs(f(c)) >= 0.5 * tol)
        q = div(a, b);
        r = div(c, b);
```

```
        s = div(c, a);

        % Next value of c closer to the root
        d = c - (r * (r - q) * (c - b) + (1 - r) * s * (c - a)) / ...
            ((q - 1) * (r - 1) * (s - 1));

        a = b;
        b = c;
        c = d;
        n = n + 1;
    end
    fprintf('\tn = %d\n', n)
    xc = c;
end
```

Main

```
% MATH 446: Project 04
% Written by Zachary Ferguson

function main()
    fprintf('MATH 446: Project 04\nWritten by Zachary Ferguson\n\n');

    init_guess_str = 'x0 = 1, x1 = 2';
    x0 = 1;
    x1 = 2;

    f1_str = 'f(x) = x^3 - 2x - 2 = 0';
    f1 = @(x) x^3 - 2*x - 2;

    f2_str = 'f(x) = e^x + x - 7 = 0';
    f2 = @(x) exp(x) + x - 7;

    f3_str = 'f(x) = e^x + sin(x) - 4 = 0';
    f3 = @(x) exp(x) + sin(x) - 4;

    fprintf('Secant Method:\n\n');

    % Q1a
    fprintf('Q1a:\n\t%s\n', f1_str);
    fprintf('\t%s\n', init_guess_str);
    fprintf('\txc = %.10f\n', secant_method(f1, x0, x1));

    % Q1b
    fprintf('Q1b:\n\t%s\n', f2_str);
    fprintf('\t%s\n', init_guess_str);
    fprintf('\txc = %.10f\n', secant_method(f2, x0, x1));

    % Q1a
    fprintf('Q1c:\n\t%s\n', f3_str);
    fprintf('\t%s\n', init_guess_str);
    fprintf('\txc = %.10f\n', secant_method(f3, x0, x1));
```

```
    fprintf('\nMethod of False Position:\n\n');

    % Q2a
    fprintf('Q2a:\n\t%s\n', f1_str);
    fprintf('\t%s\n', init_guess_str);
    fprintf('\txc = %.10f\n', method_of_false_position(f1, x0, x1));

    % Q2b
    fprintf('Q2b:\n\t%s\n', f2_str);
    fprintf('\t%s\n', init_guess_str);
    fprintf('\txc = %.10f\n', method_of_false_position(f2, x0, x1));

    % Q2c
    fprintf('Q2c:\n\t%s\n', f3_str);
    fprintf('\t%s\n', init_guess_str);
    fprintf('\txc = %.10f\n', method_of_false_position(f3, x0, x1));

    fprintf('\nInverse Quadratic Interpolation:\n\n');

    x2 = 0;

    % Q2a
    fprintf('Q3a:\n\t%s\n', f1_str);
    fprintf('\t%s, x2 = %d\n', init_guess_str, x2);
    fprintf('\txc = %.10f\n', ...
        inverse_quadratic_interpolation(f1, x0, x1, x2));

    % Q2b
    fprintf('Q3b:\n\t%s\n', f2_str);
    fprintf('\t%s, x2 = %d\n', init_guess_str, x2);
    fprintf('\txc = %.10f\n', ...
        inverse_quadratic_interpolation(f2, x0, x1, x2));

    % Q2c
    fprintf('Q3c:\n\t%s\n', f3_str);
    fprintf('\t%s, x2 = %d\n', init_guess_str, x2);
    fprintf('\txc = %.10f\n', ...
        inverse_quadratic_interpolation(f3, x0, x1, x2));
end
```

Output

```
MATH 446: Project 04
Written by Zachary Ferguson

Secant Method:

Q1a:
    f(x) = x^3 - 2x - 2 = 0
    x0 = 1, x1 = 2
    n = 6
    xc = 1.7692923542
Q1b:
```

```
    f(x) = e^x + x - 7 = 0
    x0 = 1, x1 = 2
    n = 6
    xc = 1.6728216986
Q1c:
    f(x) = e^x + sin(x) - 4 = 0
    x0 = 1, x1 = 2
    n = 5
    xc = 1.1299804986

Method of False Position:

Q2a:
    f(x) = x^3 - 2x - 2 = 0
    x0 = 1, x1 = 2
    n = 13
    xc = 1.7692923542
Q2b:
    f(x) = e^x + x - 7 = 0
    x0 = 1, x1 = 2
    n = 7
    xc = 1.6728216986
Q2c:
    f(x) = e^x + sin(x) - 4 = 0
    x0 = 1, x1 = 2
    n = 5
    xc = 1.1299804986

Inverse Quadratic Interpolation:

Q3a:
    f(x) = x^3 - 2x - 2 = 0
    x0 = 1, x1 = 2, x2 = 0
    n = 30
    xc = 1.7692923542
Q3b:
    f(x) = e^x + x - 7 = 0
    x0 = 1, x1 = 2, x2 = 0
    n = 5
    xc = 1.6728216986
Q3c:
    f(x) = e^x + sin(x) - 4 = 0
    x0 = 1, x1 = 2, x2 = 0
    n = 4
    xc = 1.1299804987
```