CAPE Lab

Assignment 1

Name: Aditya Deep
Roll No: 22CH10003
Group No: 5

Date: 10 Jan 2025

Assignment 1

Objective: Numerical Solution of a single nonlinear algebraic equation

Problem Statement:

Find the molar volume (v) of ammonia at temperature T= 250 and pressure P=10 atm using the Van der Waals Equation of State:

$$(P + a/v^2)(v-b)=RT$$

$$a = 27*R^2*T^2/64Pc$$

$$b = RTc/8Pc$$

Given: Tc=407.5 K, Pc=111.3 atm, R=0.08206 L*atm*mol^(-1)*K^(-1)

Use the following numerical methods:

(a) Fixed-point iteration (Direct substitution method)
(b) Bisection method
(c) Newton's method
(d) MATLAB built-in function fzero

Compare the efficiency of these numerical methods in terms of accuracy, computation time, and number of iterations.

Algorithms

• Fixed Point Iteration

1. Start

2. Define function f(x)

3. Define function g(x) which is obtained from f(x)=0 such that x = g(x) and |g'(x) < 1|

4. Choose intial guess x0, Tolerable Error e
   and Maximum Iteration N

5. Initialize iteration counter: step = 1

6. Calculate x1 = g(x0)

7. Increment iteration counter: step = step + 1

8. If step > N then print "Not Convergent"
   and goto (12) otherwise goto (10)

9. Set x0 = x1 for next iteration

10. If |f(x1)| > e then goto step (6) otherwise goto step (11)

11. Display x1 as root.

12. Stop

• Bisection Iteration

1. start

2. Define function f(x)

3. Choose initial guesses x0 and x1 such that f(x0)f(x1) < 0

4. Choose pre-specified tolerable error e.

5. Calculate new approximated root as x2 = (x0 + x1)/2

6. Calculate f(x0)f(x2)
   a. if f(x0)f(x2) < 0 then x0 = x0 and x1 = x2
   b. if f(x0)f(x2) > 0 then x0 = x2 and x1 = x1
   c. if f(x0)f(x2) = 0 then goto (8)

7. if |f(x2)| > e then goto (5) otherwise goto (8)

8. Display x2 as root.

9. Stop

- **Newton Raphson Iteration**

1. Start

2. Define function as f(x)

3. Define first derivative of f(x) as g(x)

4. Input initial guess (x0), tolerable error (e)
   and maximum iteration (N)

5. Initialize iteration counter i = 1

6. If g(x0) = 0 then print "Mathematical Error"
   and goto (12) otherwise goto (7)

7. Calcualte x1 = x0 - f(x0) / g(x0)

8. Increment iteration counter i = i + 1

9. If i >= N then print "Not Convergent"
   and goto (12) otherwise goto (10)

10. If |f(x1)| > e then set x0 = x1
    and goto (6) otherwise goto (11)

11. Print root as x1

12. Stop

Results

Tolerance: 10^(-6)

	Solution	Interations	Time (sec)
Fixed-point Iteration	4.231243	7	0.002408
Bisection Method	4.231243	21	0.002745
Newton's Method	4.231244	4	0.000691
MATLAB fzero function	4.231243	NA	0.000350

Analysis of Result

Accuracy:

- All methods produced a solution close to 4.231243, with only Newton's Method showing a slight difference of 4.231244, likely due to numerical precision or convergence criteria
- This indicates that all methods are capable of finding a root to a high degree of accuracy

Iteration Counts:

- **Fixed-point Iteration:** Took 7 iterations, showing moderate convergence speed. This method often converges slower compared to others due to its reliance on transforming the equation into v=g(v) and ensuring convergence conditions are met
- **Bisection Method:** Required 21 iterations, making it the slowest method. This is expected because the bisection method reduces the interval by half in each iteration and is not as fast as Newton's or Fixed-point Iteration in terms of convergence rate

- **Newton's Method:** Needed only 4 iterations, showcasing its rapid convergence when a good initial guess is provided. It achieves quadratic convergence, making it much faster than the others.
- **MATLAB fzero:** The iterations are not directly reported (NaN), but the method internally uses advanced hybrid algorithms (bracketing and interpolation). Its efficiency is comparable to Newton's Method

Computation Time:

- **MATLAB fzero:** Fastest with 0.000350 seconds, owing to its optimized internal algorithms
- **Newton's Method:** Second fastest with 0.000691 seconds, benefiting from its quadratic convergence rate
- **Fixed-point Iteration:** Took 0.002408 seconds, reflecting its moderate speed due to a larger number of iterations
- **Bisection Method:** Slightly slower than Fixed-point Iteration at 0.002745 seconds, primarily because of its linear convergence rate and higher iteration count

Method Comparison:

- **Best Overall Performance:** MATLAB fzero is the fastest and most robust method.
- **Best for Speed and Simplicity:** Newton's Method, assuming derivative $f'(v)$ is readily available.
- **Reliable and Safe Option:** Bisection Method, especially for guaranteed convergence.
- **Least Preferred:** Fixed-point Iteration, unless transforming the equation to $v=g(v)$ is easy and convergence is ensured

MATLAB CODE:

```
clearvars, clc
%Constants
R = 0.08206;  % L·atm·mol⁻¹·K⁻¹
Tc = 407.5;   % K
Pc = 111.3;   % atm

%Given Conditions
T = 250 + 273.15;  % Convert °C to K
P = 10;       % atm

% Calculate a and b
a = (27 * R^2 * Tc^2) / (64 * Pc);
b = (R * Tc) / (8 * Pc);

% function definition as f(v) = 0
f = @(v) (P + a/v^2) * (v - b) - R * T;

% Derivative of f(v) for Newton's method
f_prime = @(v) ((-2) * a * (v - b)/ v^3 + (P + a/v^2));

% Fixed-point iteration
function [v, iterations] = fixed_point_iteration(f, tol, max_iter, R, T, P, a, b)
    g = @(v) (R * T) ./ (P + a ./ v.^2) + b;
    v = 1.0;  % Initial guess
    for i = 1:max_iter
        v_new = g(v);
        if abs(v_new - v) < tol
            iterations = i;
            return;
        end
        v = v_new;
    end
    error('Fixed-point iteration did not converge');
end
```

```
% Newton's method
function [v, iterations] = newtons_method(f, f_prime, tol, max_iter)
    v = 1.0;  % Initial guess
    for i = 1:max_iter
        v_new = v - f(v) / f_prime(v);
        if abs(v_new - v) < tol
            iterations = i;
            return;
        end
        v = v_new;
    end
    error('Newton''s method did not converge');
end

% MATLAB fzero
function v = matlab_fzero(f)
    v = fzero(f, 1.0);  % Initial guess
end
```

```
% Bisection method
function [v, iterations] = bisection_method(f, tol, max_iter)
    v_low = 0.01;
    v_high = 10;  % Initial bracket (needs adjustment if necessary)
    if f(v_low) * f(v_high) > 0
        error('Bisection method requires a sign change in the interval');
    end

    for i = 1:max_iter
        v_mid = (v_low + v_high) / 2;
        if abs(f(v_mid)) < tol
            v = v_mid;
            iterations = i;
            return;
        elseif f(v_low) * f(v_mid) < 0
            v_high = v_mid;
        else
            v_low = v_mid;
        end
    end
    error('Bisection method did not converge');
end
```