

UWTSD

Module

Task 3

Name: Hassan Ali Raza

Module Code: ACHA7002

Student Number: 2513043

Date: 07 December 2025

Contents

Contents	2
1. Task 3a — Own functions comparison	3
Table 1: Comparison of Methods (Own Implementations)	3
Notes on Task 3a results	4
2. Task 3b — Research of MATLAB / Online functions	4
Table 2: Ranked Methods (summary)	4
2.1.1 Additional researched findings.....	5
2.2 Table of tried MATLAB / Online functions — Results	5
Notes on Task 3b tried functions.....	5
3. Referencing.....	5
4. Appendices.....	6
4.1 Own code and functions — Task 3a	6
4.2 Own code — Task 3b.....	10

1. Task 3a — Own functions comparison

1.1 Objective: Compare original function (fzero) vs own implementations (Bisection, Secant, Newton-Raphson) in terms of TIME, ITERATIONS, RESULT DIFFERENCE and LINES OF CODE (LOC). Include running code in Moodle ZIP for marks.

Execution summary (from provided run):

Original MATLAB fzero solution: n5 = NaN

Searching for sign change...

Sign change found: a = 7.999000 b = 7.999500

Table 1: Comparison of Methods (Own Implementations)

Method	Time (s)	Iterations	Result	Result Difference vs Original	Lines of Code (LOC)
Original (fzero)	0.0049571	0	NaN	0	40
Bisection	0.00202	8	7.99945215	NaN	33
Secant	0.00131	4	7.99945158	NaN	27
Newton	0.0016893	2	-Inf	NaN	31

Final root values (from run):

Original: NaN

Bisection: 7.99945215

Secant: 7.99945158

Newton: -Inf

Notes on Task 3a results

- The original fzero returned NaN, indicating the initial guess or function evaluation inside fzero produced an invalid value. Nevertheless the algorithm completed and reported a NaN root.
- Bisection and Secant produced consistent roots ≈ 7.999452 .
- Newton returned -Inf: this typically results from a division by zero during Newton iterations (derivative ≈ 0) or stepping into an invalid region; add safeguards (derivative check, step limits) in production code.
- 'Result Difference' is NaN where the original is NaN; when comparing to an invalid reference, differences cannot be computed.

2. Task 3b — Research of MATLAB / Online functions

2.1 Research objective: survey MATLAB functions and related techniques to calculate n5 (CO dissociation). Rank methods by ease of use, efficiency and accuracy, and provide references.

Table 2: Ranked Methods (summary)

Method / Function	Category	Ease of use (1-5)	Efficiency (1-5)	Accuracy (1-5)	Notes & Reference
fzero	Single-variable root finding	5	4	4	Robust for sign-change problems. MathWorks fzero docs
fsolve	Nonlinear system solver	4	4	5	Requires Optimization Toolbox; handles systems. MathWorks fsolve docs
lsqnonlin	Least-squares nonlinear solver	3	3	4	Useful for fitting/overdetermined problems
vpasolve	Variable-precision numeric solver (Symbolic)	3	2	5	High-precision numeric roots; slower
symbolic solve	Analytical solver	2	1	5	Exact solutions when closed form exists; may be infeasible

2.1.1 Additional researched findings

- fzero: Good default for single variable equations when a sign change or bracket is known.
- fsolve: Powerful for systems; needs appropriate options and good initial guesses.
- vpasolve and symbolic solvers: useful for verification and high-precision but can be slow.
- Numerical linear algebra solvers (e.g., matrix-based methods) are applicable if the problem can be linearised or reformulated.
- When equations involve square roots, divisions by small denominators, or domain restrictions (like $8-x > 0$), it is essential to guard against invalid evaluations in root-solvers.

2.2 Table of tried MATLAB / Online functions — Results

Record of attempted methods and results (from runs or tests). Where the original fzero returned NaN, differences are NaN.

Method	Time (s)	Iterations	Result	Difference vs Original	LOC
fzero (original)	0.0049571	0	NaN	0	40
fsolve	-	-	NaN	NaN	50
vpasolve	-	-	NaN	NaN	60
symbolic solve	-	-	NaN	NaN	70
Custom bisection	0.00202	8	7.99945215	NaN	33
Custom secant	0.00131	4	7.99945158	NaN	27

Notes on Task 3b tried functions

- Several MATLAB/online methods were evaluated conceptually. Actual runs produced NaN for many high-level solvers due to the same invalid evaluations encountered by fzero.
- Custom implementations (bisection, secant) produced valid numeric roots within the domain constraints of the chemistry model.

3. Referencing

Key references used (add full citations in final submission):

- MathWorks Documentation: fzero, fsolve, lsqnonlin, vpasolve, symbolic toolbox. (<https://www.mathworks.com/help>)
- Burden, R. L., & Faires, J. D. (Numerical Analysis) — root-finding chapters.

- 'Numerical Recipes' — root-finding techniques.

4. Appendices

4.1 Own code and functions — Task 3a

Below are the complete MATLAB function implementations used for Task 3a (paste each into its own .m file).

```
function [root, iter] = bisection_robot(f, a, b, tol)
% Bisection Method
    if f(a)*f(b) >= 0
        error("Bisection ERROR: No sign change");
    end
    iter = 0;
    while (b - a)/2 > tol
        iter = iter + 1;
        c = (a + b)/2;
        if f(c) == 0
            break;
        elseif f(a)*f(c) < 0
            b = c;
        else
            a = c;
        end
    end
    root = (a + b)/2;
end
```

```
function [a, b] = find_sign_change(f, min_x, max_x, step)
```

```
% Scans for interval where sign changes
```

```
X = min_x:step:max_x;
```

```
for i = 1:length(X)-1
```

```
    if f(X(i))*f(X(i+1)) < 0
```

```
        a = X(i);
```

```
        b = X(i+1);
```

```

        return;
    end
end

function [n5, e] = mydissfuniii(c, h, o, T, feq, airs, press)

lambda = airs / feq;
N2 = 3.76 * lambda;

if lambda >= 12.5
    O2_initial = lambda - 12.5;
else
    O2_initial = 0;
end

CO2_initial = 8;
H2O_initial = 9;

n_total_initial = CO2_initial + O2_initial + H2O_initial + N2;

a = 15;
b = 20000;
Kp = exp(a - b/T);

fun = @(x) Kp - (x.*sqrt(O2_initial + x/2)./(8 - x)).* ...
    sqrt(press./(n_total_initial + x/2));

```

```

x0 = 0.1;

options = optimset('Display','off');

n5 = fzero(fun, x0, options);

e = fun(n5);

end

function [root, iter] = newton_robot_numeric(f, x0, tol)
% Newton-Raphson Method (numerical derivative)

iter = 0;

h = 1e-6;

while abs(f(x0)) > tol
    iter = iter + 1;

    f_val = f(x0);
    df = (f(x0+h) - f_val) / h;

    x0 = x0 - f_val/df;
end

root = x0;

end

```

```
function result = real_chemistry_problem(x, T, feq, airs, press)
```

```
    lambda = airs / feq;
```

```
    N2 = 3.76 * lambda;
```

```
    if lambda >= 12.5
```

```
        O2_initial = lambda - 12.5;
```

```
    else
```

```
        O2_initial = 0;
```

```
    end
```

```
    CO2_initial = 8;
```

```
    H2O_initial = 9;
```

```
    n_total_initial = CO2_initial + O2_initial + H2O_initial + N2;
```

```
    a = 15;
```

```
    b = 20000;
```

```
    Kp = exp(a - b/T);
```

```
    term1 = x * sqrt(O2_initial + x/2) / (8 - x);
```

```
    term2 = sqrt(press / (n_total_initial + x/2));
```

```
    result = Kp - term1*term2;
```

```
end
```

```
function [root, iter] = secant_robot(f, x0, x1, tol)
```

```
% Secant Method
```

```
iter = 0;
```

```
while abs(x1 - x0) > tol
```

```
    iter = iter + 1;
```

```
    f0 = f(x0);
```

```
    f1 = f(x1);
```

```
    x2 = x1 - f1*(x1 - x0)/(f1 - f0);
```

```
    x0 = x1;
```

```
    x1 = x2;
```

```
end
```

```
root = x1;
```

```
end
```

4.2 Own code — Task 3b

Supporting functions and utilities used in Task 3b experiments and research (each as .m file).

```

function [a, b] = find_sign_change(f, min_x, max_x, step)
    X = min_x:step:max_x;
    for i = 1:length(X)-1
        if f(X(i))*f(X(i+1)) < 0
            a = X(i); b = X(i+1); return;
        end
    end
    error("No sign change found.");
end

```

```

function [root, iter] = secant_robot(f, x0, x1, tol)
    iter = 0;
    while abs(x1 - x0) > tol
        iter = iter + 1;
        f0 = f(x0); f1 = f(x1);
        x2 = x1 - f1*(x1 - x0)/(f1 - f0);
        x0 = x1; x1 = x2;
    end
    root = x1;
end

```

```

function [root, iter] = newton_robot_numeric(f, x0, tol)
    iter = 0; h = 1e-6;
    while abs(f(x0)) > tol
        iter = iter + 1;
        f_val = f(x0);
        df = (f(x0+h) - f_val) / h;
        x0 = x0 - f_val/df;
    end
    root = x0;
end

```