



University of Wales Trinity St David

Task - 3

Student name: Aangan Nitinkumar Patel

Student ID number: 2511974

Program: MSc – Automotive Engineering

Engineering Module: Alternative fuels and Combustion Modelling

Module code: ACHA7002

Date: 07/12/2025

Contents

1	Task 3a.....	1
1.1	Table of own functions results compared with given function.....	1
1.2	Running code (Moodle zip) with all associated files must be submitted to receive marks for your results table	1
2	Task 3b.....	1
2.1	Table of research of MATLAB/Online functions	1
2.1.1	Additional researched functions findings.....	2
2.2	Table of tried MATLAB/Online functions results.....	2
3	Referencing	3
4	Appendices.....	3
4.1	Own code and functions task 3a.....	3
4.1.1	Function.....	3
4.1.2	Script	8
4.2	Own code task 3b.....	9

1 Task 3a

1.1 Table of own functions results compared with given function

Compare original function TIME, ITERATIONS, RESULT DIFFERENCE, LINES of CODE with

Your results for Bisection, Secant and Newton-Raphson in a table.

Method	Avg. Time (s)	Iterations	Result Difference	Lines of code
Bisection	0.0029	25	4.8785×10^{-10}	35
Secant	0.0018	8	2.7664×10^{-10}	30
Newton-Raphson	0.0020	6	2.0120×10^{-10}	35

1.2 Running code (Moodle zip) with all associated files must be submitted to receive marks for your results table



T 3a.zip

2 Task 3b

2.1 Table of research of MATLAB/Online functions

Function/Method	Ease of Use	Efficiency	Accuracy	Notes/References
fzero	High	High	High	Bracketed root; robust scalar solver
fminbnd	High	High	High	Scalar minimiser; good for energy methods
fsolve	Medium	Medium	High	Vector systems; needs good initial guess
vpasolve	Low	Low	Highest	High-precision reference; slow symbolic solver
Custom Newton-Raphson	Low	High	High	Educational; requires derivative and

				convergence checks
Isqnonlin	Medium	Medium	High	Nonlinear least-squares; efficient for residual systems

2.1.1 Additional researched functions findings

- Isqnonlin is adds little bit lower bounds, so avoids negative roots where fsolve fails.
- Fsolve is fast but unbounded, it can step up into impossible negative values.
- Vpasolve is slow but very accurate, used to define solution.
- Fzero is robust for only one variable, unsuitable for too many equations.
- Fminbnd and fmincon diminish Gibbs energy, downpour but physically reliable.
- Custom newton Raphson shows theory but requires Jacobian and careful coding.
- Slopers use backslash linear solves, more steady and faster than inv.

2.2 Table of tried MATLAB/Online functions results

Compare original functions TIME, ITERATIONS, RESULT DIFFERENCE, LINES of CODE with your results for researched code functions.

Method	Avg. Time (s)	Iterations	Result Difference	Lines of code
Original (Goodger)	0.069449	1847850	0	160
fzero (researched)	0.0040463	16	1.1854x10-12	20

Running code (Moodle zip) with all associated files must be submitted to receive marks for your results table



T 3b.zip

3 Referencing

1. MathWorks (2024). *MATLAB and Simulink Documentation: Optimization and Equation Solving Functions (fzero, fsolve, lsqnonlin, vpasolve, fminbnd)*.
2. Goodger, E.M. (1977). *Combustion Calculations: Theory, Reaction and Applications*. London: Macmillan Education.
3. Chapra, S.C. and Canale, R.P. (2015). *Numerical Methods for Engineers*. 7th Edition. New York: McGraw-Hill Education.
4. Cutlip, M.B. and Shacham, M. (1999). *Problem Solving in Chemical Engineering with Numerical Methods*. Prentice Hall.
5. Shacham, M. (1984). 'Numerical solution of constrained non-linear algebraic equations', *International Journal for Numerical Methods in Engineering*, 20(1), pp. 177-184.

4 Appendices

4.1 Own code and functions task 3a

4.1.1 Function

```

• mydissfunii
function [out,error,iter] =
mydissfunii(carbon,hydrogen,oxygen,temperature,fuelequiv,molar_air_stoich,pressure,varargin)
% MYDISSFUNII
% Original Goodger method (7 inputs) + residual mode for root solvers (8 inputs).
%
% [out,error,iter] = mydissfunii(c,h,o,T,phi,airs,p)
%   out = n5/nt (Goodger stepping)
%   error = final n5ntvalue - n5/ntsum
%   iter = total iterations
%
% out = mydissfunii(c,h,o,T,phi,airs,p,n5ntguess)
%   out = residual F = n5ntguess - n5/ntsum (for root finding)
%   error = NaN, iter = 0 in this mode

% ----- Gibbs free energy table [kJ/mol] -----
gibbseng = [ ...
0      -393.151      -238.921      -113.805
100     -393.683      -236.584      -120.239
200     -394.085      -232.766      -128.526
298.15  -394.389      -228.582      -137.163
300     -394.394      -228.5      -137.328
400     -394.675      -223.901      -146.338
500     -394.939      -219.051      -155.414
600     -395.182      -214.007      -164.486
700     -395.398      -208.812      -173.518
800     -395.586      -203.496      -182.497
900     -395.748      -198.083      -191.416

```

1000	-395.886	-192.59	-200.275
1100	-396.001	-187.033	-209.075
1200	-396.098	-181.425	-217.819
1300	-396.177	-175.774	-226.509
1400	-396.24	-170.089	-235.149
1500	-396.288	-164.376	-243.74
1600	-396.323	-158.639	-252.284
1700	-396.344	-152.883	-260.784
1800	-396.353	-147.111	-269.242
1900	-396.349	-141.325	-277.658
2000	-396.333	-135.528	-286.034
2100	-396.304	-129.721	-294.372
2200	-396.262	-123.905	-302.672
2300	-396.209	-118.082	-310.936
2400	-396.142	-112.252	-319.165
2500	-396.062	-106.416	-327.358
2600	-395.969	-100.575	-335.517
2700	-395.862	-94.729	-343.643
2800	-395.742	-88.878	-351.736
2900	-395.609	-83.023	-359.797
3000	-395.461	-77.163	-367.826
3100	-395.298	-71.298	-375.824
3200	-395.122	-65.43	-383.792
3300	-394.932	-59.558	-391.73
3400	-394.726	-53.681	-399.638
3500	-394.506	-47.801	-407.517
3600	-394.271	-41.916	-415.366
3700	-394.022	-36.027	-423.188
3800	-393.756	-30.133	-430.981
3900	-393.477	-24.236	-438.746
4000	-393.183	-18.334	-446.485
4100	-392.874	-12.427	-454.196
4200	-392.55	-6.516	-461.881
4300	-392.21	-0.6	-469.539
4400	-391.857	5.32	-477.171
4500	-391.488	11.245	-484.778
4600	-391.105	17.175	-492.359
4700	-390.706	23.111	-499.915
4800	-390.292	29.052	-507.446
4900	-389.862	34.998	-514.951
5000	-389.419	40.949	-522.433
5100	-388.959	46.906	-529.89
5200	-388.486	52.869	-537.323
5300	-387.996	58.838	-544.732
5400	-387.493	64.811	-552.118
5500	-386.974	70.791	-559.48
5600	-386.439	76.777	-566.818
5700	-385.89	82.769	-574.133
5800	-385.324	88.767	-581.424
5900	-384.745	94.77	-588.694
6000	-384.148	100.78	-595.939];

% Gibbs free energy [kJ/mol -> J/mol]

```
gco2 = interp1(gibbseng(:,1),gibbseng(:,2),temperature)*1e3;
gh2o = interp1(gibbseng(:,1),gibbseng(:,3),temperature)*1e3;
gco = interp1(gibbseng(:,1),gibbseng(:,4),temperature)*1e3;
```

% Equilibrium constants (same as original code)

```
kco2 = exp(-(gco2-gco)*1e3/(8314.46*temperature));
kh2o = exp(-(gh2o-gco)*1e3/(8314.46*temperature));
```

```

% ----- Mode selection: original vs residual -----
if nargin == 7
    % ===== ORIGINAL GOODGER METHOD =====
    n5ntvalue = 0.1;
    error = 1;
    iter = 0;

    % ---- First step: 10^-3 increments ----
    while error > 0
        iter = iter + 1;
        error = residual_n5nt(n5ntvalue);
        n5ntvalue = n5ntvalue - 1e-3;
        if n5ntvalue < 0, break; end
    end

    % ---- Second step: 10^-6 increments ----
    if n5ntvalue < 0
        n5ntvalue = 1e-3;
        while error > 0
            iter = iter + 1;
            error = residual_n5nt(n5ntvalue);
            n5ntvalue = n5ntvalue - 1e-6;
            if n5ntvalue < 0, break; end
        end
    else
        while error < 0
            iter = iter + 1;
            error = residual_n5nt(n5ntvalue);
            n5ntvalue = n5ntvalue + 1e-6;
        end
    end

    % ---- Third step: 10^-9 increments ----
    if n5ntvalue < 0
        n5ntvalue = 1e-6;
        while error > 0
            iter = iter + 1;
            error = residual_n5nt(n5ntvalue);
            n5ntvalue = n5ntvalue - 1e-9;
            if n5ntvalue < 0, break; end
        end
    else
        while error > 0
            iter = iter + 1;
            error = residual_n5nt(n5ntvalue);
            n5ntvalue = n5ntvalue - 1e-9;
        end
    end

    % ---- Fourth step: 10^-12 increments ----
    if n5ntvalue < 0
        n5ntvalue = 1e-9;
        while error > 0
            iter = iter + 1;
            error = residual_n5nt(n5ntvalue);
            n5ntvalue = n5ntvalue - 1e-12;
            if n5ntvalue < 0, break; end
        end
    else

```

```

while error < 0
    iter = iter + 1;
    error = residual_n5nt(n5ntvalue);
    n5ntvalue = n5ntvalue + 1e-12;
end
end

out = n5ntvalue; % final n5/nt from Goodger stepping

else
    % ===== RESIDUAL MODE FOR ROOT SOLVERS (8 inputs) =====
    n5ntvalue = varargin{1};
    iter = 0; % not used in this mode
    error = residual_n5nt(n5ntvalue);
    out = error; % F(x) = 0 at the root
end

% ----- nested helper: compute residual for given n5/nt -----
function err = residual_n5nt(n5nt)
    a = sqrt(n5nt*pressure);
    n3 = (carbon*fuelequiv) / (kco2*a + 1);
    n4 = ((hydrogen/2)*fuelequiv) / (kh2o*a + 1);
    n1 = carbon*fuelequiv - n3;
    n2 = (hydrogen/2)*fuelequiv - n4;
    n5 = (molar_air_stoich*2 + oxygen*fuelequiv - 2*n1 - n2 - n3)/2;
    n6 = molar_air_stoich*3.76;
    nsum = n1 + n2 + n3 + n4 + n5 + n6;
    err = n5nt - n5/nsum;
end

end

• my_bisection
function [root, iter] = my_bisection(fun, xL, xR, tol, maxit)
% [root, iter] = my_bisection(fun, xL, xR, tol, maxit)

fL = fun(xL);
fR = fun(xR);

if fL * fR > 0
    error('Bisection: interval does not bracket a root.');
```

```

end

for iter = 1:maxit
    root = 0.5 * (xL + xR);
    fM = fun(root);

    % Stop if function value or interval size is small enough
    if abs(fM) < tol || (xR - xL)/2 < tol
        return;
    end

    % Keep the sub-interval that still brackets the root
    if fL * fM < 0
        xR = root;
        fR = fM;
    else
        xL = root;
        fL = fM;
    end
end

```

```
end
end
```

```
warning('Bisection: maximum iterations reached.');
```

```
• my_newton
function [root, iter] = my_newton(fun, x0, tol, maxit)
% [root, iter] = my_newton(fun, x0, tol, maxit)
```

```
h = 1e-6; % step for numerical derivative
x = x0;
```

```
for iter = 1:maxit
    fx = fun(x);
    if abs(fx) < tol
        root = x;
        return;
    end
```

```
% central difference derivative
dfdx = (fun(x + h) - fun(x - h)) / (2*h);
```

```
if abs(dfdx) < eps
    warning('Newton: derivative ~ 0, stopping.');
```

```
    root = x;
    return;
end

x = x - fx / dfdx;
end
```

```
root = x;
warning('Newton: maximum iterations reached.');
```

```
• my_secant
function [root, iter] = my_secant(fun, x0, x1, tol, maxit)
% [root, iter] = my_secant(fun, x0, x1, tol, maxit)
```

```
f0 = fun(x0);
f1 = fun(x1);
```

```
for iter = 1:maxit
    if abs(f1 - f0) < eps
        root = x1;
        warning('Secant: small denominator, stopping.');
```

```
        return;
    end

    % Secant update
    root = x1 - f1 * (x1 - x0) / (f1 - f0);
    f2 = fun(root);
```

```
    if abs(f2) < tol
        return;
    end
```

```

    % Shift points
    x0 = x1; f0 = f1;
    x1 = root; f1 = f2;
end

warning('Secant: maximum iterations reached.');
```

4.1.2 Script

```

    • task3a
clc; clear;

% Operating point
c = 8; % C8H18
h = 18;
o = 0;
airs = 12.5;
T = 1500; % K
p = 40; % bar
phi = 1.0; % equivalence ratio

tol = 1e-9;
maxit = 1000;

fprintf('--- Task 3a: Comparison of methods ---\n');
fprintf('T = %.1f K, p = %.1f bar, phi = %.2f\n\n', T, p, phi);

% 1) Original Goodger method
tic;
[n5_orig, err_orig, it_orig] = mydissfunii(c,h,o,T,phi,airs,p);
t_orig = toc;

fprintf('Original method:\n');
fprintf(' n5/nt = %.10f\n', n5_orig);
fprintf(' iterations = %d\n', it_orig);
fprintf(' time [s] = %.6f\n', t_orig);
fprintf(' final error = %.3e\n\n', err_orig);

% Residual for root solvers
fun = @(x) mydissfunii(c,h,o,T,phi,airs,p,x);

% 2) Bisection
tic;
[n5_bis, it_bis] = my_bisection(fun, 0.0, 0.1, tol, maxit);
t_bis = toc;

% 3) Secant
tic;
[n5_sec, it_sec] = my_secant(fun, 0.0, 0.01, tol, maxit);
t_sec = toc;

% 4) Newton–Raphson
tic;
[n5_new, it_new] = my_newton(fun, 0.001, tol, maxit);
```

```

t_new = toc;

% Differences vs original
diff_bis = abs(n5_bis - n5_orig);
diff_sec = abs(n5_sec - n5_orig);
diff_new = abs(n5_new - n5_orig);

% Approx. lines of code (for report)
loc_orig = 160;
loc_bis = 35;
loc_sec = 30;
loc_new = 35;

% Comparison table
Method = {'Original (Goodger)', 'Bisection', 'Secant', 'Newton-Raphson'};
Time_s = [t_orig; t_bis; t_sec; t_new];
Iterations = [it_orig; it_bis; it_sec; it_new];
ResDiff = [0; diff_bis; diff_sec; diff_new];
LinesOfCode = [loc_orig; loc_bis; loc_sec; loc_new];

T3a = table(Time_s, Iterations, ResDiff, LinesOfCode, 'RowNames', Method);

disp('Comparison table (Task 3a):');
disp(T3a);

```

4.2 Own code task 3b

- task3b

```

clc;

% Operating point
c = 8; % C8H18
h = 18;
o = 0;
airs = 12.5;
T = 1500; % K
p = 40; % bar
phi = 1.0;

fprintf('--- Task 3b: Researched Method (fzero) ---\n');
fprintf('T = %.1f K, p = %.1f bar, phi = %.2f\n\n', T, p, phi);

% 1) Original Goodger method
tic;
[n5_orig, err_orig, it_orig] = mydisssfunii(c,h,o,T,phi,airs,p);
t_orig = toc;

fprintf('Original method:\n');
fprintf(' n5/nt = %.10f\n', n5_orig);
fprintf(' iterations = %d\n', it_orig);
fprintf(' time [s] = %.6f\n', t_orig);
fprintf(' final error = %.3e\n', err_orig);

% 2) Researched method: fzero (scalar root finder)
% mydisssfunii in residual mode: F(x) = 0 at correct n5/nt
fun_fzero = @(x) mydisssfunii(c,h,o,T,phi,airs,p,x);

```

```

% Physical bracket for n5/nt
x_bracket = [0.0, 0.1];

opts_fz = optimset('Display','off', 'TolX',1e-12, 'TolFun',1e-12);

tic;
[n5_fz, fval_fz, exitflag_fz, out_fz] = fzero(fun_fzero, x_bracket, opts_fz);
t_fz = toc;
it_fz = out_fz.iterations;

fprintf('fzero method:\n');
fprintf(' n5/nt      = %.10f\n', n5_fz);
fprintf(' iterations = %d\n',   it_fz);
fprintf(' time [s]    = %.6f\n',  t_fz);
fprintf(' final resid = %.3e\n',  fval_fz);
fprintf(' exitflag    = %d\n\n',  exitflag_fz);

% 3) Comparison metrics for report table
diff_fz = abs(n5_fz - n5_orig);

loc_orig = 160; % approx. lines in original Goodger code
loc_fz   = 20;  % approx. lines of fzero code

Method    = {'Original (Goodger)'; 'fzero (researched)'};
Time_s    = [t_orig; t_fz];
Iterations = [it_orig; it_fz];
ResDiff   = [0; diff_fz];
LinesOfCode = [loc_orig; loc_fz];

T3b = table(Time_s, Iterations, ResDiff, LinesOfCode, 'RowNames', Method);

disp('Task 3b - Table: Tried MATLAB/Online functions results');
disp(T3b);

```