

UWTSD

Module

Task 3

Alternative Fuels Combustion Modelling

ACHA7002

Koa Brook

2203480

07/12/2025

Contents

1	Task 3a	1
1.1	Table of own functions results compared with given function.....	1
2	Task 3b	3
2.1	Table of research of MATLAB/Online functions	3
2.1.1	Additional researched functions findings	5
2.2	Table of tried MATLAB/Online functions results	7
3	Referencing.....	9
4	Appendices.....	10
4.1	Own code and functions task 3a	10
4.1.1	Main script	10
4.1.2	Functions	13
4.2	Own code task 3b	20
4.2.1	Main script	20
4.2.2	Functions	22
4.3	Full iteration tables (1.1) & (2.2)	23
4.3.1	Bisection	23
4.3.2	Secant	27
4.3.3	Newton-Raphson	29
4.3.4	Mydissfuniii.....	31
4.3.5	Ridder's.....	34
4.3.6	Fzero	38

1 Task 3a

1.1 Table of own functions results compared with given function

<u>Bisection</u>						
Temperature (Kelvin)	Pressure (Bar)	phi	Root found	iterations	residuals	Running time
800	20	0.9	0.01967	27	1.32E-10	0.015577
800	40	0.9	0.01967	27	1.32E-10	0.002324
800	60	0.9	0.01967	27	1.32E-10	0.000928
800	20	1	9.31E-10	29	8.99E-10	0.000965
800	40	1	9.31E-10	29	9.09E-10	0.010121
800	60	1	9.31E-10	29	9.13E-10	0.000841
800	20	1.1	9.31E-10	29	0.019395	0.001001
800	40	1.1	9.31E-10	29	0.019395	0.0008
800	60	1.1	9.31E-10	29	0.019395	0.001374
...	...	...	...	...	...	...

<u>Secant</u>						
Temperature (Kelvin)	Pressure (Bar)	phi	Root found	iterations	residuals	Running time
800	20	0.9	0.01967	2	3.20E-15	0.004723
800	40	0.9	0.01967	2	2.27E-15	0.00107
800	60	0.9	0.01967	2	1.83E-15	0.000705
800	20	1	1.40E-11	5	2.48E-10	0.000333
800	40	1	1.08E-11	5	2.01E-10	0.001092
800	60	1	9.30E-12	5	1.77E-10	0.000186
800	20	1.1	NaN	NaN	NaN	NaN
800	40	1.1	NaN	NaN	NaN	NaN
800	60	1.1	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...

<u>Newton-Raphson</u>						
Temperature (Kelvin)	Pressure (Bar)	phi	Root found	iterations	residuals	Running time
800	20	0.9	0.01967	2	0	0.004922
800	40	0.9	0.01967	2	0	0.000599
800	60	0.9	0.01967	2	0	0.001546
800	20	1	2.33E-10	4	1.68E-10	0.000413
800	40	1	1.80E-10	4	1.28E-10	0.001145
800	60	1	1.55E-10	4	1.09E-10	0.000298
800	20	1.1	NaN	NaN	NaN	NaN
800	40	1.1	NaN	NaN	NaN	NaN
800	60	1.1	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...

<u>mydissfuniii</u>						
Temperature (Kelvin)	Pressure (Bar)	phi	Root found	iterations	residuals	Running time
800	20	0.9	4739	7.69E-13	1.25	63.55
800	40	0.9	4739	7.71E-13	1.25	63.55
800	60	0.9	4739	7.72E-13	1.25	63.55
800	20	1	2101	-1.21E-05	2.06E-09	64
800	40	1	2101	-8.56E-06	1.46E-09	64
800	60	1	2101	-6.99E-06	1.19E-09	64
800	20	1.1	3101	0.019395	-1.25	64.45
800	40	1.1	3101	0.019395	-1.25	64.45
800	60	1.1	3101	0.019395	-1.25	64.45
...	...	...	...	...	...	...

Across the operating range, all three numerical methods—bisection, secant and Newton–Raphson—produce values of $x = n_5/n_t$ that are extremely close when convergence is achieved, typically agreeing to within 10^{-9} . The main differences between the methods appear not in the final value but in the residual error, iteration count, and failure behaviour. In terms of error, the secant and Newton–Raphson methods consistently achieve the lowest residuals, usually on the order of 10^{-11} or smaller,

whereas bisection frequently stops with much larger residuals (often between 10^{-3} and 10^{-2}). This is expected, as bisection shrinks the interval rather than directly forcing $f(x)$ to zero, while the open methods iteratively target the root more aggressively.

The iteration counts show the same pattern. Bisection is always the slowest, requiring around 23–29 iterations per case. Secant and Newton–Raphson are both significantly faster: secant typically converges in 3–8 iterations, and Newton frequently converges in as few as 2–5 iterations. Newton generally uses the fewest iterations of all methods that successfully converge.

However, this improved speed comes at the cost of robustness. Bisection converged for every single case, giving it a 0% failure rate. Secant and Newton–Raphson both show a substantial number of NaN results—indicating divergence or invalid intermediate values—particularly at rich conditions or higher pressures. Newton, in particular, is sensitive to its initial guess and derivative behaviour, making it the fastest method when successful but the least reliable across the full parameter sweep.

The legacy solver (mydissfuniii) provides an important baseline for comparison. It almost always returns a physically valid value for x , but it does so with very high iteration counts, often in the tens of thousands or tens of millions depending on the case, because it increments the oxygen balance by a fixed extremely small step. This brute-force stepping explains its long runtimes and generally poorer computational efficiency compared with modern numerical methods. Additionally, the legacy solver’s final balance error is often larger than the residuals achieved by Newton or Secant, highlighting that although it eventually finds a value, it is neither precise nor efficient. Compared to the legacy function, all three modern numerical methods dramatically reduce iteration count and computational cost, with bisection offering guaranteed convergence and Newton offering high speed where conditions permit.

2 Task 3b

2.1 Table of research of MATLAB/Online functions

<i>Method</i>	<i>Underlying Algorithm</i>	<i>Ease of Use</i>	<i>Efficiency</i>	<i>Accuracy</i>	<i>Limitations</i>	<i>Academic Reference</i>
---------------	-----------------------------	--------------------	-------------------	-----------------	--------------------	---------------------------

<i>Bisection Method</i>	Bracketing method using interval halving; requires $f(a)f(b) < 0$.	Very easy to apply; guaranteed convergence if sign change exists.	Slower convergence: many iterations required.	Moderate; accuracy increases linearly with iterations.	Requires sign change; slow compared with Newton-type methods.	<i>(Milani & Grande, 2020)</i>
<i>Newton-Raphson Method</i>	First-order Taylor linearisation; uses derivative $f'(x)$.	Requires derivative; sensitive to initial guess.	Very fast when derivative well-behaved; quadratic convergence.	High accuracy when derivative exists and is non-zero.	Fails when $f' \approx 0$; may diverge if initial guess is poor.	<i>(Liu et al., 2020)</i>
<i>Secant Method</i>	Approximates derivative using two previous points; derivative-free quasi-Newton method.	Moderate difficulty; no derivative required.	Faster than bisection; generally slower than Newton; superlinear convergence.	Good accuracy with well-chosen starting guesses.	Can fail if $f(x_0) = f(x_1)$; not guaranteed to converge.	<i>(Kopecky, 2007)</i>
<i>fzero (MATLAB)</i>	Hybrid bracketing–open method derived from Brent-like behaviour; combines bisection + secant-like steps + safeguarded interpolation.	Very easy; fully automatic solver.	High efficiency; robust; automatically handles poor guesses.	High accuracy; typically achieves machine precision.	MATLAB does not specify exact formula; internal implementation not fully documented.	<i>(Cai, n.d.)</i>
<i>Ridder's Method</i>	Bracketing method using exponential interpolation to enforce a sign-changing midpoint.	Easy to apply; no derivative required; always maintains a bracket.	Very fast initial convergence: in tests often outperforms bisection and false-position.	High accuracy (e.g., errors $\sim 10^{-5}$ in provided test cases).	Still requires a valid sign-change; can be slower than Newton when derivative is available.	Stage (2013)

2.1.1 Additional researched functions findings

2.1.1.1 Bracketing Methods

2.1.1.1.1 Bisection Method

The bisection method, as described in (*Milani & Grande, 2020*), is a bracketing numerical technique that repeatedly halves the interval containing the root. Its greatest advantage is reliability: as long as the function changes sign over the interval, convergence is guaranteed. The method is also simple to implement and does not require derivatives. However, its convergence rate is slow compared with open methods, with accuracy increasing only linearly with each iteration. This often results in a higher iteration count than more advanced solvers. Despite being reliable, the bisection method is inefficient for problems such as combustion equilibrium where many evaluations are required, but it is still valuable as a baseline or fallback method because of its robustness.

2.1.1.1.2 Brent's Method

The paper by (*Kopecky, 2007*) presents Brent's method as a hybrid algorithm that combines bisection, the secant method, and inverse quadratic interpolation. The method maintains a bracket around the root but attempts interpolation steps for faster convergence. When interpolation fails or becomes unreliable, the method safely falls back to bisection. This gives the algorithm both robustness and practical efficiency. The paper highlights that the method guarantees convergence as long as an initial sign-changing bracket exists.

2.1.1.1.3 Ridder's Method

As shown in Stage (2013), Ridder's method is a derivative-free bracketing technique that applies exponential interpolation to compute a refined midpoint while enforcing a sign change. This guarantees that the bracketing interval continually shrinks, providing improved reliability over methods such as the secant method. The paper demonstrates that Ridder's method achieves very fast initial convergence and, in the provided numerical tests, matches or outperforms Newton–Raphson in iteration count while maintaining bracketing safety. It also achieves high accuracy, with absolute errors in the order of 10^{-5} for several test problems. The primary limitation is that, like all bracketing methods, it requires the initial interval to contain a sign change. Additionally, when the derivative is easy to compute, Newton–Raphson can still outperform Ridder's method. However, for difficult nonlinear functions—such as

equilibrium dissociation equations—Ridder’s method offers an excellent balance between speed and robustness. Open / Derivative-Based Methods

2.1.1.2 Open methods

2.1.1.2.1 Newton–Raphson Method

The Newton–Raphson method, detailed in (Liu et al., 2020), uses a first-order Taylor expansion around an estimate of the root and relies on evaluating the derivative $f'(x)$. When the derivative is well-behaved and the initial guess is sufficiently close to the true root, the method converges extremely rapidly, exhibiting quadratic convergence. This makes it one of the fastest classical root-finding techniques. However, the method has two major limitations: it can diverge if the initial guess is poor, and it fails entirely when the derivative is zero or near zero. For highly nonlinear thermodynamic functions, these issues can occur, making the method highly efficient but not universally reliable without safeguards.

2.1.1.2.2 Secant Method

The secant method, described in the (*Kopecky, 2007*) paper, approximates the Newton derivative using two previous function values instead of computing $f'(x)$ directly. This results in a derivative-free method that is generally faster than bisection while maintaining simpler requirements than Newton–Raphson. The method exhibits superlinear convergence and requires only one function evaluation per step. However, it is not guaranteed to converge and can fail if the two starting values produce identical function values. As an open method, it can also diverge when the function is poorly behaved. In practice, the secant method is a good compromise between bisection’s robustness and Newton’s speed. Software / Hybrid Implementation

2.1.1.3 Hybrid (MATLAB)

2.1.1.3.1 MATLAB Fzero

(Cai, n.d.) explains that MATLAB’s `fzero` function is based on a safeguarded hybrid approach combining bisection, secant, and inverse quadratic interpolation. This hybridisation allows `fzero` to avoid the major pitfalls of purely open methods (e.g., divergence) while still achieving rapid convergence. The implementation automatically brackets roots when possible and dynamically switches between methods depending on function behaviour, making it robust and easy to use. `fzero` typically achieves very high accuracy but does not publicly document the exact internal algorithm; thus, it is normally described as “Brent-type” rather than a strict implementation of a known

method. Nevertheless, it is the most robust general-purpose solver among the methods tested.

2.2 Table of tried MATLAB/Online functions results

For the second stage of the investigation, two additional numerical solvers were selected: MATLAB's `fzero` function and Ridder's Method. These functions were chosen because they represent robust, well-established root-finding strategies that complement the behaviour of the student-written solvers. Ridder's Method is a high-accuracy bracketing method that improves upon bisection using exponential interpolation, offering faster convergence while maintaining guaranteed bracketing safety. Conversely, `fzero` is MATLAB's hybrid Brent-type solver, automatically combining bisection, secant, and inverse quadratic interpolation. It is widely used in engineering practice due to its reliability, lack of tuning requirements, and ability to converge rapidly across a wide range of initial conditions. Together, these methods allow a fair comparison with the custom solvers from Task 3a and illustrate the performance of professionally implemented numerical routines.

<u>Ridder's method</u>								
Temperature (Kelvin)	Pressure (Bar)	ph i	Root found	iterations	residuals	Running time	n5 mol (O_2)	Total mol
800	20	0.9	0.01967	6.94E-18	12	0.006784	1.25	63.55
800	40	0.9	0.01967	1.39E-17	12	0.002225	1.25	63.55
800	60	0.9	0.01967	6.94E-18	12	0.001116	1.25	63.55
800	20	1	8.05E-10	7.71E-10	36	0.002236	1.84E-08	64
800	40	1	8.05E-10	7.81E-10	36	0.003075	1.54E-08	64
800	60	1	8.05E-10	7.85E-10	36	0.002546	1.39E-08	64
800	20	1.1	9.10E-10	0.019395	60	0.003879	-1.25	64.45

800	40	1. 1	9.10E-10	0.01939 5	60	0.00340 2	-1.25	64.45
800	60	1. 1	9.10E-10	0.01939 5	60	0.00290 6	-1.25	64.45
...	...	...	...	...	...	...	...	...

<u>Fzero</u>									
Temperat ure (Kelvin)	Pressu re (Bar)	ph i	Root found	iteratio ns	residua ls	Runni ng time	exitfl ag	n5 mol (O ₂)	Tota l mol
800	20	0. 9	0.019 67	3.47E- 18	2	0.0637 02	1	1.25	63.5 5
800	40	0. 9	0.019 67	1.73E- 17	2	0.0064 69	1	1.25	63.5 5
800	60	0. 9	0.019 67	0	1	0.0034 5	1	1.25	63.5 5
800	20	1	1.17E- 11	2.76E- 10	8	0.0050 32	1	1.84 E-08	64
800	40	1	8.33E- 12	2.33E- 10	8	0.0078 51	1	1.54 E-08	64
800	60	1	6.85E- 12	2.10E- 10	8	0.0026 71	1	1.39 E-08	64
800	20	1. 1	5.25E- 10	0.01939 5	28	0.0032 57	1	-1.25	64.4 5
800	40	1. 1	5.25E- 10	0.01939 5	28	0.0017 69	1	-1.25	64.4 5
800	60	1. 1	5.25E- 10	0.01939 5	28	0.0025 24	1	-1.25	64.4 5
...	...	...	...	...	...	...	...	...	...

Analysis of the numerical results for the additional methods shows that both Ridder's Method and fzero produce physically consistent equilibrium values, matching each other to machine precision for all converged cases. In terms of accuracy, fzero typically achieves the smallest residuals, frequently reaching values on the order of 10^{-17} to

10^{-18} , while Ridder's Method produces residuals around 10^{-18} for lean mixtures and slightly larger values (approximately 10^{-10}) under stoichiometric conditions. Despite this, both methods return errors far smaller than those obtained from bisection in Task 3a, demonstrating their superior convergence characteristics.

Regarding iteration count, Ridder's Method and fzero differ substantially. Ridder's Method generally requires 12 iterations for most lean conditions and up to 36 iterations for stoichiometric cases. In contrast, MATLAB's fzero converges extremely quickly—typically in 1–2 iterations for lean mixtures and around 8 iterations for stoichiometric mixtures. This reflects the advantage of fzero's hybrid strategy, which aggressively adapts between bracketing and open-method behaviour.

In terms of computational time, Ridder's Method is consistently fast, with execution times between 0.001–0.007 seconds. fzero is slightly slower per call, ranging from 0.002–0.064 seconds, but still significantly faster overall than the legacy method from Task 3a. Despite the occasional higher runtime, fzero remains the more efficient method when normalised by convergence behaviour and iteration count.

Finally, in evaluating failure rate, neither Ridder's Method nor fzero failed for any test condition—the `exitflag_fzero` column was always equal to 1, indicating successful convergence. This contrasts with the custom secant and Newton–Raphson implementations, which experienced divergence or NaN values for several parameter combinations. The reliability of both researched methods confirms why these solvers are widely adopted in engineering thermodynamics and why they outperform simple open methods in robustness and consistency.

Overall, MATLAB's fzero was the most efficient and practical method, converging fastest with no failures. Ridder's method was the next most reliable, while Newton–Raphson was fastest only when it converged. Therefore, fzero is the best overall choice for this problem.

3 Referencing

- Milani, G. and Grande, E. (2020) *Simple bisection procedure in quickly convergent explicit ODE solver to numerically analyze FRCM strengthening systems. Composites Part B*, 199, 108322.

Available at:

https://www.sciencedirect.com/science/article/pii/S1359836820333710?fr=RR-2&ref=pdf_download&rr=9aa771a44fcbb325

(Accessed: 07/12/2025)

- Kopecky, K. A. (2007) *Root-Finding Methods*. ECO 613/614 Lecture Notes, Emory University.

Available at:

https://paulklein.ca/newsite/teaching/Notes_RootFindingMethods.pdf

(Accessed: 07/12/2025).

- Cai, C. L. (n.d.) *A More Reliable Equation Solver – My_fzero (Matlab Version)*. Math 128B Report.

Available at:

https://people.eecs.berkeley.edu/~wkahan/Math128/Projects/My_fzero_report.pdf

(Accessed: 07/12/2025).

- Liu, Z., Zhang, X., Su, M., Sun, Y., Han, H. and Wang, P. (2020) *Convergence analysis of Newton–Raphson method in feasible power-flow for DC network*. *IEEE Transactions on Power Systems*, 35(5), pp. 4100–4104.

Available at: <https://ieeexplore.ieee.org/abstract/document/9064609>

(Accessed: 07/12/2025).

- Stage, S. A. (2013). *Comments on An Improvement to the Brent’s Method*. *International Journal of Experimental Algorithms (IJEa)*, 4(1).

Available at: <https://iem.com/wp-content/uploads/2018/03/IJEa-33.pdf>

(Accessed: 07/12/2025).

4 Appendices

4.1 Own code and functions task 3a

4.1.1 Main script

```
%% Task 3 (Part A)
```

```
clc; clear;
```

```
%% USER INPUTS
```

```
% Fuel: example C8H18
```

```
c = 8;
```

```

h = 18;
o = 0;

airs = 12.5;

temps = (800:200:2000)';
press = (20:20:60)';
phis = (0.9:0.1:1.1)';

tol = 1e-9;
imax = 200;

Rows = {};

%% MAIN NESTED LOOPS
for i = 1:numel(temps)
    for j = 1:numel(phis)
        for k = 1:numel(press)

            T = temps(i);
            phi = phis(j);
            P = press(k);

%% mydissfuniii
            [~, legacy_error, it_legacy] = mydissfuniii(c,h,o,T,phi,airs,P);

%% Residual for numerical methods
            % f(x) = x - n5/nt(x)
            f = @(x) n5_balance_residual(x, c,h,o, T,phi, airs, P);
            FD = 1e-6;
            df = @(x) (f(x+FD) - f(x))/FD;

%% BISECTION METHOD
            a = 0;
            b = 0.5;
            fa = f(a);
            fb = f(b);

            if sign(fa)==sign(fb)
                for bb = [0.6 0.7 0.8 0.9]
                    if sign(fa) ~= sign(f(bb))
                        b = bb;
                        fb = f(bb);
                        break
                    end
                end
            end

            t0 = tic;
            [d_vec, it_bis] = bisect_silent(f, a, b, imax, tol);
            t_bis = toc(t0);

            x_bis = d_vec(end);
            f_bis = abs(f(x_bis));

%% SECANT METHOD
            try
                x0_sec = a;
                x1_sec = b;

```

```

        t0 = tic;
        [x_sec, it_sec] = secant_silent(x0_sec, x1_sec, f, imax, tol);
        t_sec = toc(t0);
        f_sec = abs(f(x_sec));
    catch
        x_sec = NaN;
        it_sec = NaN;
        t_sec = NaN;
        f_sec = NaN;
    end

%% NEWTON METHOD
x0 = 0.1;

    try
        t0 = tic;
        [x_newt, it_newt] = newton_silent(x0, f, df, imax, tol);
        t_newt = toc(t0);
        f_newt = abs(f(x_newt));
    catch
        x_newt = NaN;
        it_newt = NaN;
        t_newt = NaN;
        f_newt = NaN;
    end

%% Evaluate product moles at bisection root
[~, n_all] = n5_balance_residual(x_bis, c,h,o, T,phi, airs, P);

%% STORE RESULT ROW
Rows(end+1,:) = {T, P, phi, ...
                 x_bis, it_bis, f_bis, t_bis, ...
                 x_sec, it_sec, f_sec, t_sec, ...
                 x_newt, it_newt, f_newt, t_newt, ...
                 it_legacy, legacy_error, ...
                 n_all.n5, n_all.nt};

    end
end
end

%% TABLE OUTPUT
VarNames = {'T_K','P_bar','phi', ...
            'x_bis_n5_over_nt','iter_bis','residual_bis','time_bis_s', ...
            'x_sec_n5_over_nt','iter_sec','residual_sec','time_sec_s', ...
            'x_newt_n5_over_nt','iter_newt','residual_newt','time_newt_s', ...
            'legacy_iters','legacy_balance_error','n5_mol','nt_total_mol'};

Tresults = cell2table(Rows, 'VariableNames', VarNames);

%% Preview
figh = figure('Color','w','Position',[100 100 1200 600]);
uitable(figh,'Data',Tresults{1:min(40,height(Tresults)),:}, ...
        'ColumnName',Tresults.Properties.VariableNames, ...
        'Units','normalized','Position',[0 0 1 1]);

exportgraphics(figh,'Task3_PartA_MethodComparison.pdf','ContentType','vector');
disp('Saved Task3_PartA_MethodComparison.pdf');

```

```
writetable(Tresults,'Task3_PartA_MethodComparison.csv');
```

4.1.2 Functions

4.1.2.1 Mydissfuniii

```
function [out,error,itors] =  
mydissfuniii(carbon,hydrogen,oxygen,temperature,fuelequiv,molar_air_stoich,pressur  
e)  
% mydissfuniii - legacy Goodger dissociation solver with iteration count  
%  
% out    = final n5/nt value (oxygen balance fraction)  
% error  = final error = n5ntvalue - n5/ntsum  
% iters  = total number of internal iterations taken
```

```
n5ntvalue = 0.1;  
error      = 1;  
itors      = 0;    % iteration counter
```

%TEMP	CO2	H2O	CO	(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 -> ??? units as in original code)
gco2 = interp1(gibbseng(1:end,1),gibbseng(1:end,2),temperature)*10^3;
gh2o = interp1(gibbseng(1:end,1),gibbseng(1:end,3),temperature)*10^3;
gco = interp1(gibbseng(1:end,1),gibbseng(1:end,4),temperature)*10^3;

% Equilibrium constants (using original author's formula)
kco2 = exp(-(gco2-gco)*10^3/(8314.46*temperature));
kh2o = exp(-(gh2o)*10^3/(8314.46*temperature));

% ----- Goodger method for calculation of dissociation -----
% Iterate through n5/nt guess until error goes negative or guess reaches 0
while error>0
    a = sqrt(n5ntvalue*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;
    ntsum = n1+n2+n3+n4+n5+n6;
    error = n5ntvalue-n5/ntsum;

    n5ntvalue = n5ntvalue-0.001;
    iters = iters + 1; % count iteration

    if n5ntvalue<0
        break
    end
end

%-----
% If guess reached 0, reset guess to 0.001 and iterate down in 10^-3 steps
```

```

if n5ntvalue<0
    n5ntvalue=0.001;
    while error>0
        a = sqrt(n5ntvalue*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;
        ntsum = n1+n2+n3+n4+n5+n6;
        error = n5ntvalue-n5/ntsum;

        n5ntvalue = n5ntvalue-0.000001;
        iters = iters + 1;          % count iteration

        if n5ntvalue<0
            break
        end
    end
    % if error went negative, iterate guess in +10^-6 increments until error is
    % positive
else
    while error<0
        a = sqrt(n5ntvalue*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;
        ntsum = n1+n2+n3+n4+n5+n6;
        error = n5ntvalue-n5/ntsum;

        n5ntvalue = n5ntvalue+0.000001; %-6
        iters = iters + 1;          % count iteration
    end
end

%-----
% repeat with smaller increments
% now positive error
if n5ntvalue<0
    n5ntvalue=0.000001;
    while error>0
        a = sqrt(n5ntvalue*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;
        ntsum = n1+n2+n3+n4+n5+n6;
        error = n5ntvalue-n5/ntsum;

        n5ntvalue = n5ntvalue-0.00000001;
        iters = iters + 1;          % count iteration

        if n5ntvalue<0

```

```

        break
    end
end

else
    while error>0
        a = sqrt(n5ntvalue*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;
        ntsum = n1+n2+n3+n4+n5+n6;
        error = n5ntvalue-n5/ntsum;

        n5ntvalue = n5ntvalue-0.000000001; %-9
        iters = iters + 1;          % count iteration
    end
end

%-----
% error now negative
if n5ntvalue<0
    n5ntvalue=0.000000001;
    while error>0
        a = sqrt(n5ntvalue*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;
        ntsum = n1+n2+n3+n4+n5+n6;
        error = n5ntvalue-n5/ntsum;

        n5ntvalue = n5ntvalue-0.000000000001;
        iters = iters + 1;          % count iteration

        if n5ntvalue<0
            break
        end
    end
end

else
    while error<0
        a = sqrt(n5ntvalue*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;
        ntsum = n1+n2+n3+n4+n5+n6;
        error = n5ntvalue-n5/ntsum;

        n5ntvalue = n5ntvalue+0.000000000001; %-12
        iters = iters + 1;          % count iteration
    end
end

```

end

% Final output: equilibrium oxygen fraction according to this method
out = n5ntvalue;

end

4.1.2.2 *Bisect*

```
function [d, iters] = bisect_silent(f,a,b,imax,tol)
```

```
x1 = a;  
x3 = b;  
f1 = feval(f,x1);  
f3 = feval(f,x3);
```

```
if f1*f3 > 0  
    error('No root in the specified interval');  
end
```

```
d = zeros(1,imax);
```

```
for k = 1:imax  
    x2 = 0.5*(x1 + x3);  
    f2 = feval(f,x2);  
  
    d(k) = x2;  
  
    if abs(f2) < tol || 0.5*abs(x3 - x1) < tol  
        iters = k;  
        d = d(1:k);  
        return;  
    end  
  
    if f1*f2 < 0  
        x3 = x2; f3 = f2;  
    else  
        x1 = x2; f1 = f2;  
    end  
end
```

```
iters = imax;  
d = d(1:imax);  
end
```

4.1.2.3 *Newton-Raphson*

```
function [root, iters] = newton_silent(x0, f, df, imax, tol)
```

```
x = x0;  
iters = 0;  
abs_diff = inf;
```

```
while (abs_diff > tol) && (iters < imax)
```

```
    fx = feval(f, x);  
    dfx = feval(df, x);
```

```
    if dfx == 0  
        error('Derivative is zero - cannot continue.');
```

```

end

x_new = x - fx/dfx;

abs_diff = abs(x_new - x);
x = x_new;
iters = iters + 1;
end

root = x;
end

```

4.1.2.4 Secant

```

function [root, iters] = secant_silent(x0, x1, f, imax, tol)

iters = 0;

if x0 == x1
    error('Initial guesses x0 and x1 must be different for secant method.');
```

```

end

f0 = f(x0);
f1 = f(x1);

while (iters < imax) && (abs(x1 - x0) > tol) && (abs(f1) > tol)

    if f1 == f0
        error('Secant method breakdown: f(x1) == f(x0) -> division by zero.');
```

```

    end

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

    x0 = x1;
    f0 = f1;
    x1 = x2;
    f1 = f(x1);

    iters = iters + 1;
end

root = x1;
end

```

4.1.2.5 n5 balance residual

```

function [fval, n_out] = n5_balance_residual(x, carbon, hydrogen, oxygen,
temperature, fuelequiv, molar_air_stoich, pressure)
% f(x) = x - n5/nt(x), using the same relations as in mydissfuniii

gibbseng = [ ... % Temp(K)    g_CO2    g_H2O    g_CO    (kJ/mol)
    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 -228.500 -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 -192.590 -200.275
1100 -396.001 -187.033 -209.075
1200 -396.098 -181.425 -217.819
1300 -396.177 -175.774 -226.509
1400 -396.240 -170.089 -235.149
1500 -396.288 -164.376 -243.740
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 -65.430 -383.792
3300 -394.932 -59.558 -391.730
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 -392.550 -6.516 -461.881
4300 -392.210 -0.600 -469.539
4400 -391.857 5.320 -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 -529.890
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 -559.480
5600 -386.439 76.777 -566.818
5700 -385.890 82.769 -574.133
5800 -385.324 88.767 -581.424
5900 -384.745 94.770 -588.694
6000 -384.148 100.780 -595.939];

```

```
% Interpolate to J/mol
```

```
gco2 = 1e3*interp1(gibbseng(:,1),gibbseng(:,2),temperature,'linear','extrap');
gh2o = 1e3*interp1(gibbseng(:,1),gibbseng(:,3),temperature,'linear','extrap');
```

```

gco = 1e3*interp1(gibbseng(:,1),gibbseng(:,4),temperature,'linear','extrap');
R = 8.31446;

Kco2 = exp(-(gco2 - gco)/(R*temperature));
Kh2o = exp(-(gh2o)/(R*temperature));

x = max(x,0);
a = sqrt(x*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;
nt = n1+n2+n3+n4+n5+n6;

fval = x - n5/nt;

if nargout>1
    n_out = struct('n1',n1,'n2',n2,'n3',n3,'n4',n4,'n5',n5,'n6',n6,'nt',nt);
end
end

```

4.2 Own code task 3b

4.2.1 Main script

```
%% PART B
```

```
RowsB = {};
```

```
opts = optimset('TolX',tol,'TolFun',tol,'Display','off');
```

```

for i = 1:numel(temps)
    for j = 1:numel(phis)
        for k = 1:numel(press)

            T = temps(i);
            phi = phis(j);
            P = press(k);

            f = @(x) n5_balance_residual(x, c,h,o, T,phi, airs, P);

%% Ridder's Method (bracketing)
            aR = 0.0;
            bR = 0.5;
            fa = f(aR);
            fb = f(bR);

            if sign(fa) == sign(fb)
                for bb = [0.6 0.7 0.8 0.9]
                    if sign(fa) ~= sign(f(bb))
                        bR = bb;
                        fb = f(bR);
                        break
                    end
                end
            end

            try

```

```

        if sign(fa) ~= sign(fb)
            t0 = tic;
            [x_rid, it_rid] = ridder_silent(f, aR, bR, imax, tol);
            t_rid = toc(t0);
            f_rid = abs(f(x_rid));
        else
            x_rid = NaN;
            it_rid = NaN;
            t_rid = NaN;
            f_rid = NaN;
        end
    catch
        x_rid = NaN;
        it_rid = NaN;
        t_rid = NaN;
        f_rid = NaN;
    end
end

%% fzero
try
    x0_fz = 0.02;

    t0 = tic;
    [x_fzero, fval_fzero, exitflag, output] = ...
        fzero(f, x0_fz, opts);
    t_fzero = toc(t0);
    it_fzero = output.iterations;

catch
    x_fzero = NaN;
    fval_fzero = NaN;
    exitflag = -1;
    it_fzero = NaN;
    t_fzero = NaN;
end

if ~isnan(x_fzero)
    [~, nB] = n5_balance_residual(x_fzero, c,h,o, T,phi, airs, P);
    n5_B = nB.n5;
    nt_B = nB.nt;
else
    n5_B = NaN;
    nt_B = NaN;
end

RowsB(end+1,:) = {T, P, phi, ...
                  x_rid, f_rid, it_rid, t_rid, ...
                  x_fzero, abs(fval_fzero), it_fzero, t_fzero, ...
                  exitflag, n5_B, nt_B};

end
end
end

%% table for Part B
VarNamesB = {'T_K', 'P_bar', 'phi', ...

'x_ridder_n5_over_nt', 'residual_ridder', 'iter_ridder', 'time_ridder_s', ...

```

```

'x_fzero_n5_over_nt','residual_fzero','iter_fzero','time_fzero_s','exitflag_fzero'
, ...
    'n5_mol','nt_total_mol'});

TresultsB = cell2table(ResultsB,'VariableNames',VarNamesB);

%% Preview table
fighB = figure('Color','w','Position',[100 100 1200 600]);
uitable(fighB,'Data',ResultsB{1:min(40,height(ResultsB)),:}, ...
    'ColumnName',ResultsB.Properties.VariableNames, ...
    'Units','normalized','Position',[0 0 1 1]);

exportgraphics(fighB,'Task3_PartB_fzeroResults.pdf','ContentType','vector');
disp('Saved Task3_PartB_fzeroResults.pdf');

writetable(TresultsB,'Task3_PartB_fzeroResults.csv');

```

4.2.2 Functions

4.2.2.1 *Ridders*

```

function [root, iters] = ridder_silent(f, a, b, imax, tol)

    fa = f(a);
    fb = f(b);

    if fa*fb > 0
        error('Ridder method requires f(a) and f(b) of opposite sign.');
```

end

```

    iters = 0;
    root = NaN;

    while iters < imax
        c = 0.5*(a + b);
        fc = f(c);

        if abs(fc) < tol
            root = c;
            return;
        end

        s = sqrt(fc^2 - fa*fb);
        if s == 0
            root = c;
            return;
        end

        if fc >= 0
            sign_fc = 1;
        else
            sign_fc = -1;
        end

        x = c + (c - a)*sign_fc*fc/s;
        fx = f(x);

        if fc*fx < 0

```

```

        a = c;  fa = fc;
        b = x;  fb = fx;
elseif fa*fx < 0
    b = x;  fb = fx;
else
    a = x;  fa = fx;
end

iters = iters + 1;

if abs(b-a) < tol || abs(fx) < tol
    root = x;
    return;
end
end

root = 0.5*(a + b);
end

```

4.3 Full iteration tables (1.1) & (2.2)

4.3.1 Bisection

<u>Bisection</u>						
Temperature (Kelvin)	Pressure (Bar)	phi	x_bis_n5_over_n t	iterations	residuals	Running time
800	20	0.9	0.01967	27	1.32E-10	0.01557 7
800	40	0.9	0.01967	27	1.32E-10	0.00232 4
800	60	0.9	0.01967	27	1.32E-10	0.00092 8
800	20	1	9.31E-10	29	8.99E-10	0.00096 5
800	40	1	9.31E-10	29	9.09E-10	0.01012 1
800	60	1	9.31E-10	29	9.13E-10	0.00084 1
800	20	1.1	9.31E-10	29	0.019395	0.00100 1
800	40	1.1	9.31E-10	29	0.019395	0.0008

800	60	1.1	9.31E-10	29	0.019395	0.00137 4
1000	20	0.9	0.01967	27	1.46E-10	0.00070 1
1000	40	0.9	0.01967	27	1.42E-10	0.00068 2
1000	60	0.9	0.01967	27	1.40E-10	0.00110 6
1000	20	1	1.68E-08	28	3.62E-10	0.00067
1000	40	1	1.30E-08	28	6.93E-10	0.00067
1000	60	1	1.12E-08	27	9.34E-10	0.00065 1
1000	20	1.1	9.31E-10	29	0.019395	0.00068 1
1000	40	1.1	9.31E-10	29	0.019395	0.00069 3
1000	60	1.1	9.31E-10	29	0.019395	0.00071 4
1200	20	0.9	0.01967	26	7.93E-10	0.00068 2
1200	40	0.9	0.01967	28	2.50E-10	0.00068
1200	60	0.9	0.01967	28	1.14E-10	0.00074 3
1200	20	1	5.80E-07	29	1.31E-10	0.00068
1200	40	1	4.60E-07	28	7.69E-10	0.00080 4
1200	60	1	4.02E-07	25	4.14E-11	0.00070 4
1200	20	1.1	9.31E-10	29	0.019379	0.00066 4
1200	40	1.1	9.31E-10	29	0.019383	0.00069 3
1200	60	1.1	9.31E-10	29	0.019386	0.00099 7

1400	20	0.9	0.01967	28	7.35E-10	0.00067 6
1400	40	0.9	0.01967	26	9.98E-11	0.00062 5
1400	60	0.9	0.01967	28	3.84E-10	0.00063 5
1400	20	1	7.55E-06	26	3.75E-10	0.00067 9
1400	40	1	5.99E-06	24	5.96E-10	0.00058 4
1400	60	1	5.23E-06	29	3.76E-10	0.00070 1
1400	20	1.1	9.31E-10	29	0.018643	0.00065 6
1400	40	1.1	9.31E-10	29	0.018862	0.00077 9
1400	60	1.1	9.31E-10	29	0.01896	0.00071 7
1600	20	0.9	0.019672	26	7.64E-10	0.00062 5
1600	40	0.9	0.019671	23	7.83E-10	0.00060 3
1600	60	0.9	0.019671	28	4.14E-10	0.00064 5
1600	20	1	5.26E-05	27	1.20E-10	0.00071 3
1600	40	1	4.17E-05	25	4.58E-10	0.00066 4
1600	60	1	3.65E-05	25	6.37E-10	0.00062 1
1600	20	1.1	9.31E-10	29	0.007086	0.00066 8
1600	40	1.1	9.31E-10	29	0.010389	0.00067 5

1600	60	1.1	9.31E-10	29	0.011926	0.00069 2
1800	20	0.9	0.019693	27	9.41E-10	0.00065 9
1800	40	0.9	0.019686	28	1.45E-10	0.00064 3
1800	60	0.9	0.019683	27	2.65E-11	0.00061 8
1800	20	1	0.000239	29	3.85E-10	0.00069 4
1800	40	1	0.00019	26	5.12E-10	0.00062 1
1800	60	1	0.000166	28	8.82E-10	0.00072
1800	20	1.1	2.89E-08	29	0.000178	0.00063 3
1800	40	1.1	1.40E-08	29	0.000439	0.00067
1800	60	1.1	1.02E-08	29	0.000312	0.00068 9
2000	20	0.9	0.019814	24	1.02E-10	0.00054 9
2000	40	0.9	0.019772	25	5.24E-10	0.00056 7
2000	60	0.9	0.019753	28	7.91E-10	0.00061 8
2000	20	1	0.000802	28	1.09E-10	0.00068 2
2000	40	1	0.000637	29	7.53E-11	0.00066 5
2000	60	1	0.000557	28	2.36E-10	0.00065 1
2000	20	1.1	1.10E-06	29	4.62E-06	0.00065 6
2000	40	1.1	5.50E-07	29	1.18E-05	0.00065 4

2000	60	1.1	3.68E-07	29	7.79E-06	0.00064 3
------	----	-----	----------	----	----------	--------------

4.3.2 Secant

<u>Secant</u>						
Temperature (Kelvin)	Pressure (Bar)	phi	Root found	iterations	residuals	Running time
800	20	0.9	0.01967	2	3.20E-15	0.004723
800	40	0.9	0.01967	2	2.27E-15	0.00107
800	60	0.9	0.01967	2	1.83E-15	0.000705
800	20	1	1.40E-11	5	2.48E-10	0.000333
800	40	1	1.08E-11	5	2.01E-10	0.001092
800	60	1	9.30E-12	5	1.77E-10	0.000186
800	20	1.1	NaN	NaN	NaN	NaN
800	40	1.1	NaN	NaN	NaN	NaN
800	60	1.1	NaN	NaN	NaN	NaN
1000	20	0.9	0.01967	2	7.21E-12	0.000242
1000	40	0.9	0.01967	2	5.10E-12	9.24E-05
1000	60	0.9	0.01967	2	4.16E-12	9.67E-05
1000	20	1	1.66E-08	8	6.50E-10	0.000231
1000	40	1	1.31E-08	8	5.53E-10	0.000219
1000	60	1	1.15E-08	8	5.03E-10	0.000238
1000	20	1.1	NaN	NaN	NaN	NaN
1000	40	1.1	NaN	NaN	NaN	NaN
1000	60	1.1	NaN	NaN	NaN	NaN
1200	20	0.9	0.01967	3	7.98E-17	0.000157
1200	40	0.9	0.01967	3	4.16E-17	0.00011
1200	60	0.9	0.01967	2	8.30E-10	8.90E-05
1200	20	1	5.80E-07	9	1.06E-10	0.000258
1200	40	1	4.61E-07	9	1.01E-10	0.000307
1200	60	1	4.02E-07	9	9.84E-11	0.000252
1200	20	1.1	NaN	NaN	NaN	NaN
1200	40	1.1	NaN	NaN	NaN	NaN

1200	60	1.1	NaN	NaN	NaN	NaN
1400	20	0.9	0.01967	3	1.65E-13	0.000114
1400	40	0.9	0.01967	3	8.27E-14	0.000128
1400	60	0.9	0.01967	3	5.52E-14	0.000103
1400	20	1	7.55E-06	9	9.74E-11	0.00026
1400	40	1	5.99E-06	9	1.03E-10	0.000253
1400	60	1	5.23E-06	9	1.06E-10	0.000263
1400	20	1.1	NaN	NaN	NaN	NaN
1400	40	1.1	NaN	NaN	NaN	NaN
1400	60	1.1	NaN	NaN	NaN	NaN
1600	20	0.9	0.019672	3	5.60E-11	0.000112
1600	40	0.9	0.019671	3	2.80E-11	0.000109
1600	60	0.9	0.019671	3	1.87E-11	0.000108
1600	20	1	5.26E-05	9	3.18E-11	0.000242
1600	40	1	4.17E-05	9	3.90E-11	0.00024
1600	60	1	3.65E-05	9	4.35E-11	0.000339
1600	20	1.1	NaN	NaN	NaN	NaN
1600	40	1.1	NaN	NaN	NaN	NaN
1600	60	1.1	NaN	NaN	NaN	NaN
1800	20	0.9	0.019693	4	1.45E-15	0.000142
1800	40	0.9	0.019686	4	3.61E-16	0.000131
1800	60	0.9	0.019683	4	1.60E-16	0.000123
1800	20	1	0.000239	9	4.02E-12	0.000236
1800	40	1	0.00019	9	6.10E-12	0.000235
1800	60	1	0.000166	9	7.62E-12	0.000271
1800	20	1.1	NaN	NaN	NaN	NaN
1800	40	1.1	NaN	NaN	NaN	NaN
1800	60	1.1	NaN	NaN	NaN	NaN
2000	20	0.9	0.019814	4	2.01E-12	0.000126
2000	40	0.9	0.019772	4	5.11E-13	0.000123
2000	60	0.9	0.019753	4	2.29E-13	0.000122
2000	20	1	0.000802	9	1.73E-13	0.00024
2000	40	1	0.000637	9	3.66E-13	0.000239
2000	60	1	0.000557	9	5.47E-13	0.000245

2000	20	1.1	NaN	NaN	NaN	NaN
2000	40	1.1	NaN	NaN	NaN	NaN
2000	60	1.1	NaN	NaN	NaN	NaN

4.3.3 Newton-Raphson

<u>Newton-Raphson</u>						
Temperature (Kelvin)	Pressure (Bar)	phi	Root found	iterations	residuals	Running time
800	20	0.9	0.01967	2	0	0.004922
800	40	0.9	0.01967	2	0	0.000599
800	60	0.9	0.01967	2	0	0.001546
800	20	1	2.33E-10	4	1.68E-10	0.000413
800	40	1	1.80E-10	4	1.28E-10	0.001145
800	60	1	1.55E-10	4	1.09E-10	0.000298
800	20	1.1	NaN	NaN	NaN	NaN
800	40	1.1	NaN	NaN	NaN	NaN
800	60	1.1	NaN	NaN	NaN	NaN
1000	20	0.9	0.01967	2	0	0.000313
1000	40	0.9	0.01967	2	0	0.000139
1000	60	0.9	0.01967	2	0	0.000144
1000	20	1	1.67E-08	9	4.27E-10	0.000604
1000	40	1	1.32E-08	9	3.75E-10	0.000614
1000	60	1	1.16E-08	9	3.50E-10	0.000638
1000	20	1.1	NaN	NaN	NaN	NaN
1000	40	1.1	NaN	NaN	NaN	NaN
1000	60	1.1	NaN	NaN	NaN	NaN
1200	20	0.9	0.01967	3	0	0.000205
1200	40	0.9	0.01967	2	3.47E-18	0.000137
1200	60	0.9	0.01967	2	1.39E-17	0.000135
1200	20	1	5.80E-07	7	2.46E-10	0.000457
1200	40	1	4.61E-07	8	1.22E-10	0.000632
1200	60	1	4.02E-07	8	1.80E-10	0.000542
1200	20	1.1	NaN	NaN	NaN	NaN

1200	40	1.1	NaN	NaN	NaN	NaN
1200	60	1.1	NaN	NaN	NaN	NaN
1400	20	0.9	0.01967	3	0	0.000204
1400	40	0.9	0.01967	3	0	0.000193
1400	60	0.9	0.01967	3	0	0.000226
1400	20	1	7.55E-06	7	8.81E-12	0.000489
1400	40	1	5.99E-06	7	6.36E-12	0.000462
1400	60	1	5.23E-06	7	3.26E-12	0.000476
1400	20	1.1	NaN	NaN	NaN	NaN
1400	40	1.1	NaN	NaN	NaN	NaN
1400	60	1.1	NaN	NaN	NaN	NaN
1600	20	0.9	0.019672	3	0	0.000198
1600	40	0.9	0.019671	3	0	0.000193
1600	60	0.9	0.019671	3	0	0.000193
1600	20	1	5.26E-05	7	6.57E-12	0.000531
1600	40	1	4.17E-05	8	7.64E-14	0.000569
1600	60	1	3.65E-05	8	1.17E-13	0.001021
1600	20	1.1	NaN	NaN	NaN	NaN
1600	40	1.1	NaN	NaN	NaN	NaN
1600	60	1.1	NaN	NaN	NaN	NaN
1800	20	0.9	0.019693	3	1.39E-17	0.000258
1800	40	0.9	0.019686	3	0	0.000192
1800	60	0.9	0.019683	3	6.94E-18	0.000184
1800	20	1	0.000239	7	4.05E-13	0.0005
1800	40	1	0.00019	7	5.93E-13	0.000456
1800	60	1	0.000166	7	6.88E-13	0.000456
1800	20	1.1	NaN	NaN	NaN	NaN
1800	40	1.1	NaN	NaN	NaN	NaN
1800	60	1.1	NaN	NaN	NaN	NaN
2000	20	0.9	0.019814	3	6.25E-17	0.000189
2000	40	0.9	0.019772	3	3.47E-18	0.000183
2000	60	0.9	0.019753	3	6.94E-18	0.000183
2000	20	1	0.000802	7	6.69E-15	0.000453
2000	40	1	0.000637	7	1.85E-14	0.000442

2000	60	1	0.000557	7	3.19E-14	0.000463
2000	20	1.1	NaN	NaN	NaN	NaN
2000	40	1.1	NaN	NaN	NaN	NaN
2000	60	1.1	NaN	NaN	NaN	NaN

4.3.4 Mydissfuniii

<u>mydissfuniii</u>						
Temperature (Kelvin)	Pressure (Bar)	phi	Root found	iterations	residuals	Running time
800	20	0.9	4739	7.69E-13	1.25	63.55
800	40	0.9	4739	7.71E-13	1.25	63.55
800	60	0.9	4739	7.72E-13	1.25	63.55
800	20	1	2101	-1.21E-05	2.06E-09	64
800	40	1	2101	-8.56E-06	1.46E-09	64
800	60	1	2101	-6.99E-06	1.19E-09	64
800	20	1.1	3101	0.019395	-1.25	64.45
800	40	1.1	3101	0.019395	-1.25	64.45
800	60	1.1	3101	0.019395	-1.25	64.45
1000	20	0.9	4753	7.27E-13	1.25	63.55
1000	40	0.9	4749	8.42E-13	1.25	63.55
1000	60	0.9	4747	6.65E-13	1.25	63.55
1000	20	1	3090	8.48E-13	1.10E-06	64
1000	40	1	3586	8.56E-13	8.79E-07	64
1000	60	1	3882	1.17E-12	7.75E-07	64
1000	20	1.1	2101	-0.00511	-1.24999	64.45001
1000	40	1.1	3101	0.019393	-1.25	64.45
1000	60	1.1	3101	0.019393	-1.25	64.45

1200	20	0.9	4536	8.69E-14	1.25	63.55
1200	40	0.9	4717	3.89E-13	1.25	63.55
1200	60	0.9	4354	7.97E-13	1.25	63.55
1200	20	1	2824	8.03E-13	3.71E-05	64.00004
1200	40	1	3228	1.53E-13	2.95E-05	64.00003
1200	60	1	3059	3.17E-14	2.58E-05	64.00003
1200	20	1.1	2101	-0.107	-1.24898	64.45102
1200	40	1.1	2101	-0.10589	-1.24928	64.45072
1200	60	1.1	2101	-0.10505	-1.24941	64.45059
1400	20	0.9	4989	7.86E-13	1.250009	63.55001
1400	40	0.9	4546	2.58E-13	1.250006	63.55001
1400	60	0.9	4516	7.74E-13	1.250005	63.55
1400	20	1	1548784	7.78E-13	0.000483	64.00048
1400	40	1	1991761	3.92E-14	0.000383	64.00038
1400	60	1	1234450	9.51E-13	0.000335	64.00033
1400	20	1.1	2101	-0.10969	-1.20244	64.49756
1400	40	1.1	2101	-0.10966	-1.21631	64.48369
1400	60	1.1	2101	-0.10964	-1.22247	64.47753

1600	20	0.9	4142	8.35E-13	1.250157	63.55016
1600	40	0.9	5025	4.84E-13	1.250111	63.55011
1600	60	0.9	4748	3.40E-13	1.25009	63.55009
1600	20	1	1583443	9.69E-13	0.003365	64.00337
1600	40	1	1738909	1.25E-12	0.002671	64.00267
1600	60	1	1463783	6.20E-14	0.002334	64.00233
1600	20	1.1	2101	-0.10975	-0.46227	65.23773
1600	40	1.1	2101	-0.10975	-0.67551	65.02449
1600	60	1.1	2101	-0.10975	-0.77428	64.92572
1800	20	0.9	4873	7.46E-13	1.251522	63.55152
1800	40	0.9	4645	9.45E-13	1.251076	63.55108
1800	60	0.9	5091	5.11E-13	1.250879	63.55088
1800	20	1	1180590	3.35E-13	0.015311	64.01531
1800	40	1	1908420	8.72E-14	0.012156	64.01216
1800	60	1	1930367	1.21E-12	0.010621	64.01062
1800	20	1.1	3603	1.12E-07	0.011706	65.71171
1800	40	1.1	3853	9.75E-08	0.028877	65.72888

1800	60	1.1	3937	6.26E-07	- 0.02052	65.67948
2000	20	0.9	5196	8.29E-13	1.25936 8	63.55937
2000	40	0.9	4343	4.76E-15	1.25663 5	63.55663
2000	60	0.9	5012	9.43E-13	1.25542 1	63.55542
2000	20	1	1098070	1.04E-12	0.05137 5	64.05138
2000	40	1	1459984	9.28E-13	0.04082 3	64.04082
2000	60	1	1227763	1.12E-13	0.03568 2	64.03568
2000	20	1.1	103510	4.86E-09	0.00037 6	65.70038
2000	40	1.1	2795	1.13E-08	0.00080 8	65.70081
2000	60	1.1	3240	1.82E-08	- 0.00049	65.69951

4.3.5 Ridder's

<u>Ridder's method</u>								
Temperat ure (Kelvin)	Pressu re (Bar)	p hi	Root found	iteration s	residu als	Runnin g time	n5 mol (O ₂)	Total mol
800	20	0. 9	0.01967	6.94E- 18	12	0.00678 4	1.25	63.55
800	40	0. 9	0.01967	1.39E- 17	12	0.00222 5	1.25	63.55
800	60	0. 9	0.01967	6.94E- 18	12	0.00111 6	1.25	63.55
800	20	1	8.05E-10	7.71E- 10	36	0.00223 6	1.84E- 08	64

800	40	1	8.05E-10	7.81E-10	36	0.003075	1.54E-08	64
800	60	1	8.05E-10	7.85E-10	36	0.002546	1.39E-08	64
800	20	1.1	9.10E-10	0.019395	60	0.003879	-1.25	64.45
800	40	1.1	9.10E-10	0.019395	60	0.003402	-1.25	64.45
800	60	1.1	9.10E-10	0.019395	60	0.002906	-1.25	64.45
1000	20	0.9	0.01967	3.46E-15	12	0.001269	1.25	63.55
1000	40	0.9	0.01967	2.44E-15	12	0.000707	1.25	63.55
1000	60	0.9	0.01967	1.98E-15	12	0.000618	1.25	63.55
1000	20	1	1.72E-08	3.31E-10	34	0.002028	1.12E-06	64
1000	40	1	1.29E-08	9.45E-10	32	0.001787	8.51E-07	64
1000	60	1	1.18E-08	4.01E-11	35	0.001769	7.48E-07	64
1000	20	1.1	9.10E-10	0.019395	60	0.002852	-1.24999	64.45001
1000	40	1.1	9.10E-10	0.019395	60	0.003062	-1.25	64.45
1000	60	1.1	9.10E-10	0.019395	60	0.003303	-1.25	64.45
1200	20	0.9	0.01967	6.87E-13	12	0.000552	1.25	63.55
1200	40	0.9	0.01967	4.86E-13	12	0.00067	1.25	63.55

1200	60	0. 9	0.01967	3.97E- 13	12	0.00068 1	1.25	63.55
1200	20	1	5.80E-07	5.92E- 10	30	0.00143 3	3.72E- 05	64.000 04
1200	40	1	4.61E-07	3.72E- 12	36	0.00155 1	2.95E- 05	64.000 03
1200	60	1	4.02E-07	4.63E- 10	30	0.00133 1	2.58E- 05	64.000 03
1200	20	1. 1	9.09E-10	0.01937 9	60	0.00272 6	- 1.2489 9	64.451 01
1200	40	1. 1	9.09E-10	0.01938 3	60	0.00333	- 1.2492 4	64.450 76
1200	60	1. 1	9.09E-10	0.01938 5	60	0.00368 2	- 1.2493 5	64.450 65
1400	20	0. 9	0.01967	3.22E- 11	12	0.00058 9	1.2500 09	63.550 01
1400	40	0. 9	0.01967	2.28E- 11	12	0.00063 9	1.2500 06	63.550 01
1400	60	0. 9	0.01967	1.86E- 11	12	0.0007	1.2500 05	63.55
1400	20	1	7.55E-06	4.22E- 10	29	0.00141 1	0.0004 83	64.000 48
1400	40	1	5.99E-06	4.11E- 10	64	0.00320 6	0.0003 83	64.000 38
1400	60	1	5.23E-06	2.36E- 10	194	0.00986 9	0.0003 35	64.000 33
1400	20	1. 1	8.71E-10	0.01861 8	60	0.00267 6	- 1.1421 9	64.557 81

1400	40	1. 1	8.82E-10	0.01884 8	60	0.00248 4	- 1.2036 2	64.496 38
1400	60	1. 1	8.87E-10	0.01894 9	60	0.00250 8	- 1.2306	64.469 4
1600	20	0. 9	0.019672	5.90E- 10	12	0.00053 8	1.2501 57	63.550 16
1600	40	0. 9	0.019671	4.18E- 10	12	0.00056	1.2501 11	63.550 11
1600	60	0. 9	0.019671	3.42E- 10	12	0.00053 2	1.2500 9	63.550 09
1600	20	1	5.26E-05	2.46E- 10	37	0.00154 8	0.0033 65	64.003 37
1600	40	1	4.17E-05	6.21E- 15	137	0.00590 1	0.0026 71	64.002 67
1600	60	1	3.65E-05	2.61E- 16	140	0.00558 6	0.0023 34	64.002 33
1600	20	1. 1	9.13E-10	0.00698	58	0.00237 9	- 0.5611 5	65.138 85
1600	40	1. 1	7.33E-10	0.00935	59	0.00241 1	0.2672 68	65.967 27
1600	60	1. 1	8.36E-10	0.01154	59	0.00245 2	- 0.8815 9	64.818 41
1800	20	0. 9	0.020061	0.00036 9	200	0.00814 6	1.2515 22	63.551 52
1800	40	0. 9	0.020062	0.00037 6	200	0.00808 1	1.2510 76	63.551 08
1800	60	0. 9	0.020062	0.00037 9	200	0.00835 3	1.2508 79	63.550 88
1800	20	1	0.000246	1.03E- 05	200	0.00791 4	0.0153 11	64.015 31

1800	40	1	0.00019	3.62E-11	71	0.002845	0.012156	64.01216
1800	60	1	0.000166	9.50E-15	96	0.003851	0.010621	64.01062
1800	20	1.1	3.23E-08	0.000704	200	0.007848	0.005288	65.70529
1800	40	1.1	1.53E-08	0.000276	200	0.008149	-0.02377	65.67623
1800	60	1.1	1.08E-08	0.000723	200	0.007698	0.024671	65.72467
2000	20	0.9	0.020055	0.000241	200	0.007806	1.259368	63.55937
2000	40	0.9	0.020056	0.000285	200	0.007468	1.256635	63.55663
2000	60	0.9	0.020057	0.000305	200	0.007363	1.255421	63.55542
2000	20	1	0.000839	5.49E-05	200	0.007134	0.051375	64.05138
2000	40	1	0.000637	4.29E-16	89	0.003339	0.040823	64.04082
2000	60	1	0.000572	2.12E-05	200	0.007206	0.035682	64.03568
2000	20	1.1	1.13E-06	0.00016	200	0.007301	-1.21E-05	65.69999
2000	40	1.1	6.00E-07	0.000647	200	0.007168	4.38E-05	65.70004
2000	60	1.1	3.75E-07	0.000148	200	0.007375	0.000131	65.70013

4.3.6 Fzero

<u>Fzero</u>

Temperature (Kelvin)	Pressure (Bar)	p hi	Root found	iterations	residuals	Running time	exitflag	n5 mol (O ₂)	Total mol
800	20	0. 9	0.0196 7	3.47E- 18	2	0.0637 02	1	1.25	63.55
800	40	0. 9	0.0196 7	1.73E- 17	2	0.0064 69	1	1.25	63.55
800	60	0. 9	0.0196 7	0	1	0.0034 5	1	1.25	63.55
800	20	1	1.17E- 11	2.76E- 10	8	0.0050 32	1	1.84E- 08	64
800	40	1	8.33E- 12	2.33E- 10	8	0.0078 51	1	1.54E- 08	64
800	60	1	6.85E- 12	2.10E- 10	8	0.0026 71	1	1.39E- 08	64
800	20	1. 1	5.25E- 10	0.0193 95	28	0.0032 57	1	-1.25	64.45
800	40	1. 1	5.25E- 10	0.0193 95	28	0.0017 69	1	-1.25	64.45
800	60	1. 1	5.25E- 10	0.0193 95	28	0.0025 24	1	-1.25	64.45
1000	20	0. 9	0.0196 7	1.04E- 15	2	0.0043 29	1	1.25	63.55
1000	40	0. 9	0.0196 7	7.39E- 16	2	0.0003 99	1	1.25	63.55
1000	60	0. 9	0.0196 7	6.21E- 16	2	0.0005 06	1	1.25	63.55
1000	20	1	1.60E- 08	1.48E- 09	9	0.0014 7	1	1.12E- 06	64
1000	40	1	1.39E- 08	6.26E- 10	9	0.0014 29	1	8.51E- 07	64
1000	60	1	1.20E- 08	3.17E- 10	8	0.0016 07	1	7.48E- 07	64

1000	20	1. 1	5.27E- 10	0.0193 95	28	0.0018 36	1	- 1.2499 9	64.450 01
1000	40	1. 1	5.27E- 10	0.0193 95	28	0.0018 68	1	-1.25	64.45
1000	60	1. 1	5.26E- 10	0.0193 95	28	0.0014 45	1	-1.25	64.45
1200	20	0. 9	0.0196 7	2.10E- 13	2	0.0004 79	1	1.25	63.55
1200	40	0. 9	0.0196 7	1.49E- 13	2	0.0003 75	1	1.25	63.55
1200	60	0. 9	0.0196 7	1.21E- 13	2	0.0003 46	1	1.25	63.55
1200	20	1	5.80E- 07	1.17E- 09	10	0.0011 51	1	3.72E- 05	64.000 04
1200	40	1	4.60E- 07	6.25E- 10	10	0.0009 68	1	2.95E- 05	64.000 03
1200	60	1	4.02E- 07	3.33E- 10	10	0.0014 67	1	2.58E- 05	64.000 03
1200	20	1. 1	9.55E- 10	0.0193 79	28	0.0023 67	1	- 1.2489 9	64.451 01
1200	40	1. 1	8.29E- 10	0.0193 83	28	0.0019 6	1	- 1.2492 4	64.450 76
1200	60	1. 1	7.73E- 10	0.0193 85	28	0.0016 29	1	- 1.2493 5	64.450 65
1400	20	0. 9	0.0196 7	9.86E- 12	2	0.0006 65	1	1.2500 09	63.550 01
1400	40	0. 9	0.0196 7	6.97E- 12	2	0.0004 39	1	1.2500 06	63.550 01
1400	60	0. 9	0.0196 7	5.69E- 12	2	0.0004 96	1	1.2500 05	63.55

1400	20	1	7.55E-06	1.82E-10	9	0.001187	1	0.000483	64.00048
1400	40	1	5.99E-06	3.18E-12	9	0.001223	1	0.000383	64.00038
1400	60	1	5.23E-06	1.88E-11	8	0.001054	1	0.000335	64.00033
1400	20	1.1	1.79E-10	0.017692	29	0.001403	1	-1.14219	64.55781
1400	40	1.1	4.90E-10	0.018662	29	0.001547	1	-1.20362	64.49638
1400	60	1.1	1.88E-09	0.019088	27	0.001339	1	-1.2306	64.4694
1600	20	0.9	0.019672	1.82E-10	2	0.000308	1	1.250157	63.55016
1600	40	0.9	0.019671	1.29E-10	2	0.00024	1	1.250111	63.55011
1600	60	0.9	0.019671	1.05E-10	2	0.000213	1	1.25009	63.55009
1600	20	1	5.26E-05	7.07E-12	10	0.000851	1	0.003365	64.00337
1600	40	1	4.17E-05	1.47E-11	10	0.001051	1	0.002671	64.00267
1600	60	1	3.65E-05	1.96E-11	10	0.000982	1	0.002334	64.00233
1600	20	1.1	1.25E-09	0.008615	29	0.001437	1	-0.56115	65.13885
1600	40	1.1	1.00E-10	0.004052	29	0.001505	1	0.267268	65.96727
1600	60	1.1	1.60E-09	0.013601	29	0.001395	1	-0.88159	64.81841

1800	20	0. 9	0.0196 93	2.03E- 10	2	0.0004 36	1	1.2515 22	63.551 52
1800	40	0. 9	0.0196 86	7.34E- 10	2	0.0004 85	1	1.2510 76	63.551 08
1800	60	0. 9	0.0196 83	9.69E- 10	2	0.0004 88	1	1.2508 79	63.550 88
1800	20	1	0.0002 39	2.53E- 13	9	0.0009 23	1	0.0153 11	64.015 31
1800	40	1	0.0001 9	8.57E- 12	9	0.0008 76	1	0.0121 56	64.012 16
1800	60	1	0.0001 66	3.63E- 11	9	0.0008 33	1	0.0106 21	64.010 62
1800	20	1. 1	2.92E- 08	8.05E- 05	26	0.0011 77	1	0.0052 88	65.705 29
1800	40	1. 1	1.55E- 08	0.0003 62	25	0.0012 2	1	- 0.0237 7	65.676 23
1800	60	1. 1	9.39E- 09	0.0003 75	29	0.0013 4	1	0.0246 71	65.724 67
2000	20	0. 9	0.0198 14	3.99E- 15	3	0.0003 48	1	1.2593 68	63.559 37
2000	40	0. 9	0.0197 72	2.38E- 15	3	0.0002 57	1	1.2566 35	63.556 63
2000	60	0. 9	0.0197 53	1.68E- 15	3	0.0003 12	1	1.2554 21	63.555 42
2000	20	1	0.0008 02	1.27E- 09	8	0.0007 4	1	0.0513 75	64.051 38
2000	40	1	0.0006 37	1.41E- 09	8	0.0007 52	1	0.0408 23	64.040 82
2000	60	1	0.0005 57	9.03E- 10	8	0.0007 23	1	0.0356 82	64.035 68
2000	20	1. 1	1.10E- 06	1.29E- 06	25	0.0011 11	1	- 1.21E- 05	65.699 99

2000	40	1. 1	5.51E- 07	1.16E- 07	22	0.0009 84	1	4.38E- 05	65.700 04
2000	60	1. 1	3.67E- 07	1.63E- 06	25	0.0010 96	1	0.0001 31	65.700 13