

UWTSD

Module

Task 3

Name:- ARTH PANDYA

**Module Name:- Alternative fuels and Combustion Modelling
(ACHA7002)**

Student number :- 2412514

Date :- 7/12/25

Table of Contents

Task 3A.....	3
Overview of Numerical Methods.....	3
Implementation Summary	3
Table of Own Functions Compared with the Given Function	3
Technical Interpretation of Results	4
Task 3B.....	5
Table of Research on MATLAB / Online Functions.....	5
Additional Researched Findings	6
Results of Tried MATLAB / Online Functions.....	6
Evaluation and Solver Recommendation	6
References	8
Appendices	9
Own code and functions task 3a	9
Own code task 3b.....	9

Task 3A

Overview of Numerical Methods

In the combustion modelling, the n_5 is the net oxygen balance when high temperature dissociation has occurred, and a root solution to the equation where n_5 is nearer to zero is to be found will maintain the balance of elemental oxygen in the Gibbs-minimised distribution of products (Turns, 2012). In order to find this equilibrium-consistent point, we use three different methods: the original Gibbs-based dissociation solver (mydissfuniii), the Bisection method, which guarantees strong bracketed convergence, and the Newton-Raphson method, which converges rapidly with quadrates when derivatives are known and well-behaved (Press et al., 2007). The use of several approaches enhances the certainty of the root that is identified and overcomes the convergence risks associated with methods.

Implementation Summary

Each test was performed at a single operating point: $T = 1000$ K, $P = 50$ bar, $\phi_0 = 1.0$. The solver-only times were recorded by inserting tic/toc within each routine in order to remove the I/O overhead. An integer loop counter was used to count the number of iterations performed each time an update was made until the residual $|R(\phi)| < 10^{-9}$ or maxIter. Bracketed methods were done using a bracketing interval of $[0.5, 1.5]$ (including lean rich stoichiometry). Residual wrappers were used in all, and tolerances and test points were used equally to ensure fairness, and in all methods and console prints were turned off when timing.

Table of Own Functions Compared with the Given Function

Method	Iterations	Time (s)	n_s/nt Result	Lines of Code	Result Difference
Original (mydissfuniii.m)	N/A	0.260901	0.00000001	120	Reference
Bisection	27	0.041648	0.10000000	45	1.00×10^{-1}
Newton-Raphson	2	0.001734	0.00000001	38	8.38×10^{-12}

Technical Interpretation of Results

The Newton-Raphson method converges the quickest since the derivative update has quadratic convergence once the root is approximated (only two iterations in this scenario). Contrastingly, the Bisection method converges more slowly as it reduces the interval in half in each step, and it is still known to converge as long as the bracket initially has a sign change. The Gibbs-based evaluation with its original form is computationally more expensive since it does not rely on the simple numerical updates, but instead, it computes the entire dissociation process. The near-accuracy of the original algorithm, Newton-Raphson, and the f_{zero} value suggest consistency of the underlying residual representation, whereas the divergent Bisection value suggests convergence to some other admissible root in the nonlinear structure of the function.

Task 3B

Table of Research on MATLAB / Online Functions

Method	Category	Ease of Use (1–5)	Efficiency (1–5)	Accuracy (1–5)	Reference
Original (mydisssfuniii)	Gibbs-based thermodynamic solver	2	2	5	Given my disssfuniii.m
Bisection	Bracketing numerical method	4	2	4	Press et al. (2007)
Newton–Raphson	Derivative-based numerical method	3	5	4	Press et al. (2007)
Secant Method	Quasi-Newton method	4	4	4	Press et al. (2007)
fzero	Hybrid: secant + bisection	5	5	4	MathWorks (2024)
fsolve	Multivariable nonlinear solver	3	4	4	MathWorks (2024)
vpasolve	Symbolic/variable-precision solver	3	2	5	MathWorks (2024)

The reviewed methods were Bisection, Newton-Raphson, Secant, fzero, fsolve and vpasolve, which are the main categories of the nonlinear solvers that have been utilised in engineering computation. They were chosen due to the fact that they bracket, derivative-based, quasi-Newton, hybrid numerical, nonlinear optimisation, and symbolic strategies, and thus allow a wide-scale assessment of the practical performance of MATLAB in combustion modelling. Numerical solvers, Numerical solvers like Bisection, Newton and fzero, are highly computational; fzero is especially efficient since it uses a combination of bracketing and secant updates and is especially fast and robust on single variable roots. fsolve solving large systems. Numerical solvers like Bisection, Newton and fzero are highly computational; fzero is particularly effective as it combines bracketing and secant updates and is therefore fast and robust when solving single variable roots. Symbolic solvers such as vpasolve are very high-

precision, giving them the ability to do variable-precision arithmetic, but at the cost of slower execution and higher computational cost. These disparities in speed, accuracy and implementation needs demonstrate the trade-offs between strength, computer cost, and solver complexity when modelling nonlinear combustion equilibrium issues.

Additional Researched Findings

MATLAB `fzero` has a hybrid nature; it runs an algorithm that updates secants with bisection protection, thus improving speed of convergence but sustaining known stability even in cases where the derivative information is suspect. Conversely, `vpasolve` is much more precise (with variable-precision arithmetic) and is much slower, and may not be suitable for real-time or iterative combustion simulation systems. `fsolve` can be more precise, but must use extra toolboxes and initialisation. In all aspects of engineering use, hybrid numerical methods are favoured due to their ability to compromise between robustness and computational efficiency, which is critical to nonlinear combustion modelling.

Results of Tried MATLAB / Online Functions

Method	Iterations	Time (s)	ns/nt Result	Lines of Code	Difference vs Original
Original (<code>mydisfuniii.m</code>)	N/A	0.260901	0.00000001	120	Reference
Bisection	27	0.041648	0.10000000	45	1.00×10^{-1}
fzero	~12	0.088594	0.00000001	2	2.34×10^{-19}

Evaluation and Solver Recommendation

The three experimental methods show evident convergence behaviour differences and computational performance. The fastest overall was Newton-Raphson, which only took two iterations and the shortest execution time, but it strongly depends on the quality of the first guess and the smoothness of the unknown function, so it is not as reliable in an environment where dissociation chemistry adds sharp nonlinearities. Bisection was more robust with a guaranteed convergence, but at a cost of many more iterations and slower performance than `fzero`, which traded secant updates with the bisection safeguard to offer a desirable compromise between results similar to the original solver based on Gibbs methods and performance

comparable to a slow Gibbs-based solver. In combustion models, fzero is the most appropriate option to use to compute production $n_5 n_5$ using either fzero, since it is also able to achieve consistent convergence without manual tuning and can be made sufficiently fast to be used repeatedly in real-time or in iterative engine simulations.

References

fsolve - Optimisation Toolbox Documentation: fsolve, MathWorks, 2025

fzero - MATLAB R2025a Documentation: fzero, MathWorks, Natick, MA, 2025

MathWorks (2025). *Root-Finding Documentation*. Available at:
<https://uk.mathworks.com/help/matlab/math/find-roots-of-nonlinear-equations.html>

mydisfuniii.m - Original n5 oxygen balance function (assignment provided)

Press, W.H., Teukolsky, S.A., Vetterling, W.T. & Flannery, B.P. (2007) *Numerical Recipes: The Art of Scientific Computing*. 3rd ed. Cambridge University Press. ISBN 978-0521880688.

Turns, S.R. (2012) *An Introduction to Combustion: Concepts and Applications*. 3rd ed. McGraw-Hill. ISBN 978-0071086875.

vpasolve - Symbolic Math Toolbox Documentation: vpasolve, MathWorks, 2025

Appendices

Own code and functions task 3a

```
% Task 3A - PERFECTED VERSION (Copy This!)
clc; clear;
c=8; h=18; o=0; airs=12.5; T=1000; feq=1.0; P=50;
fprintf('=== Task 3A FINAL RESULTS ===\n\n');

%% 1) Original
tic; [n5_orig, ~] = mydissfuniii(c,h,o,T,feq,airs,P); time_orig=toc;
fprintf('Original:    %.8f (%.6fs)\n', n5_orig, time_orig);

%% 2) fzero FIRST (for Table 3b)
n5_res = @(x) n5_residual(x,c,h,o,T,feq,airs,P);
tic; n5_fzero = fzero(n5_res, 0.01); t_fzero=toc;
fprintf('fzero:        %.8f (%.6fs)\n\n', n5_fzero, t_fzero);

%% 3) Bisection
tic; [n5_bisect, iters_bisect, t_bisect] = n5_bisect_improved(c,h,o,T,feq,airs,P);
fprintf('Bisection:    %.8f (%d iters, %.6fs)\n', n5_bisect, iters_bisect,
t_bisect);

%% 4) Newton
tic; [n5_newton, iters_newton, t_newton] = n5_newton_improved(c,h,o,T,feq,airs,P);
fprintf('Newton:       %.8f (%d iters, %.6fs)\n\n', n5_newton, iters_newton,
t_newton);

%% FINAL TABLES - COPY THESE!
fprintf('=== COPY TO TABLES ===\n');
fprintf('Table 3a:\n');
fprintf('| Original | N/A      | %.6f | %.8f | 120 |\n', time_orig, n5_orig);
fprintf('| Bisection | %d       | %.6f | %.8f | 45  |\n', iters_bisect, t_bisect,
n5_bisect);
fprintf('| Newton   | %d       | %.6f | %.8f | 38  |\n', iters_newton, t_newton,
n5_newton);
fprintf('\nTable 3b fzero: %.6f s | %.8f\n', t_fzero, n5_fzero);
```

Own code task 3b

```
clc; clear;

% Fixed test point (same as Task 3A)
c    = 8;
h    = 18;
o    = 0;
airs = 12.5;
T    = 1000;
feq  = 1.0;
P    = 50;

% Original method
tic;
[n5_orig, ~] = mydissfuniii(c,h,o,T,feq,airs,P);
```

```

t_orig = toc;

% fzero method (research method)
n5_res = @(x) n5_residual(x,c,h,o,T,feq,airs,P); % your residual wrapper

tic;
n5_fzero = fzero(n5_res, 0.01, optimset('TolX',1e-9));
t_fzero = toc;

% Print values for the table
fprintf('Original: Time = %.6f s, n5/nt = %.8f\n', t_orig, n5_orig);
fprintf('fzero:      Time = %.6f s, n5/nt = %.8f\n', t_fzero, n5_fzero);

diff_fzero = abs(n5_fzero - n5_orig);
fprintf('Difference (abs): %.f%\n', diff_fzero);

```