Combustion Toolbox: An open-source thermochemical code for gas- and condensed-phase problems involving chemical equilibrium

Alberto Cuadra*, César Huete, Marcos Vera

Departamento de Ingeniería Térmica y de Fluidos, Escuela Politécnica Superior, Universidad Carlos III de Madrid, 28911 Leganés, Spain

Abstract

We present a new open-source thermochemical code—hereafter referred to as Combustion Toolbox (CT)—that solves problems involving chemical equilibrium of gas- and condensed-phase species. The kernel of the code is based on the theoretical framework set forth by NASA's computer program CEA (Chemical Equilibrium with Applications) while incorporating new algorithms that significantly speed up the convergence rate. The thermochemical properties are computed under the ideal gas approximation using an up-to-date version of NASA's 9-coefficient polynomial fits. These fits use the Third Millennium database, which includes the available values from Active Thermochemical Tables. Combustion Toolbox is programmed in MATLAB with a modular architecture composed of three main modules: CT-EQUIL, CT-SD, and CT-ROCKET. The core module, CT-EQUIL, minimizes the Gibbs/Helmholtz free energy of the system using the technique of Lagrange multipliers combined with a multidimensional Newton-Raphson method, upon the condition that two state functions are used to define the mixture properties (e.g., enthalpy and pressure). CT-SD solves processes involving strong changes in dynamic pressure, such as steady shock and detonation waves under normal and oblique incidence angles. Finally, CT-ROCKET estimates rocket engine performance under highly idealized conditions. The new tool is equipped with a versatile Graphical User Interface and has been successfully used for teaching and research activities over the last four years. Results are in excellent agreement with CEA, Cantera within Caltech's Shock and Detonation Toolbox (SD-Toolbox), and the recent Thermochemical Equilibrium Abundances (TEA) code. CT is available under an open-source GPLv3 license via GitHub https://github.com/AlbertoCuadra/combustion_toolbox, and its documentation can be found in https://combustion-toolbox-website.readthedocs.io.

Keywords:
Thermochemistry, shock wave, detonation, oblique front, rocket performance, GUI

Program summary

Program Title: Combustion Toolbox (CT)
CPC Library link to program files: (to be added by Technical Editor)
Developer's repository link: https://github.com/AlbertoCuadra/combustion_toolbox
Code Ocean capsule: (to be added by Technical Editor)
Licensing provisions: GPLv3
Programming language: MATLAB
Nature of problem: Combustion Toolbox (CT) has been developed to unify a collection of routines for solving problems involving chemical equilibrium of gas- and condensed-phase species in a MATLAB software package. The code can be used either from the MATLAB console or through a Graphic User Interface. CT is written in a modular architecture and is composed of three main modules: CT-EQUIL, for the estimation of the final equilibrium conditions resulting from canonical thermochemical transformations; CT-SD, for the computation of shocks and detonations in a neutral or ionized gas phase under different flow configurations; and CT-ROCKET, for the theoretical estimation of rocket engine performance.
Solution method: Chemical equilibrium is solved via minimization of the Gibbs/Helmholtz free energy combining the method of Lagrange multipliers with a multidimensional Newton-Raphson (NR) method. The shock and detonation routines solve the Rankine-Hugoniot relations also using a NR method. The rocket engine performance module relies on an iterative procedure. All calculations are based on the ideal gas approximation.

1. Introduction

The computation of chemical equilibrium has been widely used during the last century to determine the composition of multi-component mixtures subject to complex thermochemical transformations. The resulting mathematical problem is simple in systems involving only a few species, such as the complete combustion of rich hydrocarbon-air mixtures, or dissociation of diatomic gas mixtures, e.g., air at moderate temperatures. However, the incomplete combustion of a typical hydrocarbon, for instance, methane, with air, involves hundreds of reactions and more than fifty species [1]. This makes finding the final equilibrium state of the products a challenging task.

Two equivalent methods can be employed to determine the composition of the products at equilibrium: using equilibrium

*Corresponding author.
E-mail address: acuadra@ing.uc3m.es (A. Cuadra)

constants or minimizing the Gibbs/Helmholtz free energy of the system [2]. The first method requires specifying a sufficiently large set of elementary reactions at equilibrium (see, e.g., Refs. [3–7]). This favors the second, where each species is treated independently, and the focus is shifted to the chemical potentials of the different species involved [8, 9]. The second method was first introduced by the pioneering work of White in 1958 [10] and has become the cornerstone in the development of virtually all state-of-the-art thermochemical codes [11–28].

The solution to the resulting minimization problem requires the evaluation of the thermodynamic properties of all species involved at a given temperature. For this purpose, extensive thermodynamic databases have been compiled, such as the NIST-JANAF tables [29, 30], NASA's polynomials [31] and, more recently, the Third Millennium (Burcat) Database [32] with updates from the Active Thermochemical Tables (ATcT) [33] to evaluate the enthalpies of formation. The ATcT rely on the use of a complete thermochemical network (TN) instead of the more traditional datasets based on individual reactions. The use of the full TN yields more accurate results that are, in addition, fully documented (uncertainties included). But more important is the fact that databases are easily updated with new knowledge, and readily provide new values for the thermochemical properties of all species [34].

A recent work by Scoggins et al. [35] reported an exhaustive review of over 1200 unique chemical species from several databases that fully (Goldsmith [36] and Blanquart [37–40]), partially (Burcat), or did not (NASA) rely on the ATcT. They identified significant differences between Burcat's and NASA's databases due to the inconsistency of the species enthalpy of formation in the latter. Although this type of analysis is out of the scope of this work, all chemical species from Burcat's database are also available in Combustion Toolbox (CT) and can be identified by the suffix "_M". Thus, when both databases contain a given species, the final choice is left to the user.

Thermochemistry is firmly rooted in the study of combustion problems, high-speed flows, reactive and non-reactive shocks, rocket engine performance, and high explosives [41, 42]. For instance, strong hypersonic shocks involve changes in the molecular structure of the gas, including vibrational excitation leading to dissociation [43, 44], and later electronic excitation leading to ionization [45], which eventually transform the gas into a plasma. Turbulent combustion and gaseous detonations have also been the topic of intense research due to their high thermodynamic efficiency in propulsion applications [46, 47]. But they often exhibit strong deviations from equilibrium due to the wide range of length and time scales involved, making it necessary to rely on complex fluid dynamical analyses and numerical simulations with a high computational cost [48, 49]. Despite the deep understanding provided by the latter approach, there are still cases in which a proper physical explanation can not be found based only on numerical results. In these cases, separation of scales may allow to split the problem into simpler ones, where the assumption of chemical equilibrium could be justified in some representative scenarios [43, 50, 51].

Chemical equilibrium can be formulated either for single-phase gas mixtures [5, 7, 10, 17, 19, 20, 23, 24], single-phase gas mixtures with pure-condensed species—as in NASA's CEA code [15], Refs. [3, 6, 12], and this work [28]—or for multi-phase systems [4, 13, 14, 16, 18, 21, 22, 25–27, 52, 53]. The latter case requires special treatments to ensure that the global minima are reached, due to the non-convexity of the Gibbs free energy [18, 21, 54]. These include global optimization methods like the tunneling method [55] or differential evolution [56, 57].

The above review has identified many thermochemical codes currently available to the community. Nevertheless, to the best of our knowledge, there is not yet an open-source code based on up-to-date databases, written in a high-level programming language, fully documented, with high-performance computing capabilities, able to model a wide variety of applications, and equipped with a user-friendly Graphic User Interface (GUI). Combustion Toolbox was conceived with these long-term goals in mind, and is now presented and validated in this work. This MATLAB-GUI thermochemical code represents the core of an ongoing research work the has been used to investigate a series of problems during the last few years [43, 50, 51, 58]. Results are in excellent agreement with NASA's CEA code [15], Cantera [59] within Caltech's Shock and Detonation Toolbox (SD-Toolbox) [60, 61], and the recent Thermochemical Equilibrium Abundances (TEA) code [23].

The paper is structured as follows. Section 2 starts with an initial overview of CT. The equilibrium kernel (CT-EQUIL) is presented in Section 3. The shock and detonation module (CT-SD) is discussed in Section 4, followed by the rocket performance module (CT-ROCKET) in Section 5. The results of all modules are validated against other codes in Sections 3 to 5. A detailed description of the GUI is given in Section 6. And, finally, the conclusions are presented in Section 7.

2. Overview of Combustion Toolbox

Combustion Toolbox [28] is a GUI-based thermochemical code written in MATLAB with an equilibrium kernel based on the mathematical formulation set forth by NASA in its CEA code [15]. The thermodynamic properties of the gaseous species are modeled with the ideal gas equation of state (EoS), and an up-to-date version of NASA's 9-coefficient polynomial fits from [31–33]. CT is a new thermochemical code written from scratch in a modular architectural format composed of three main modules: CT-EQUIL, CT-SD, and CT-ROCKET.

The first module, CT-EQUIL, computes the composition at the equilibrium of multi-component gas mixtures that undergo canonical thermochemical transformations from an initial state (reactants), defined by its initial composition, temperature, and pressure, to a final state (products), defined by a set of chemical species (in gaseous—included ions—or pure condensed phase) and two thermodynamic state functions, such as enthalpy and pressure, e.g., for isobaric combustion processes. CT-SD solves steady-state shock and detonation waves in either normal or oblique incidence. Finally, CT-ROCKET computes the theoretical performance of rocket engines under highly idealized conditions. Even though all modules are enclosed in a user-friendly GUI, they can also be accessed from MATLAB's command line (in plain code mode).

There is a fourth closed-source (i.e., proprietary) module, CT-EXPLO, that estimates the theoretical properties of high explosive mixtures and multi-component propellants with non-ideal EoS. Although still under development, CT-EXPLO is distributed in its current form as the thermochemical module of SimEx [58] subject to a proprietary license. Further details on this module will be provided elsewhere.

2.1. Software Architecture

As previously mentioned, the program was developed from scratch using MATLAB and a modular architectural design. All the modules rely on CT-EQUIL (core module) to compute the thermodynamic properties of the species involved and the chemical composition of the mixtures at equilibrium. The full package can be accessed from the GUI (see Section 6) or from MATLAB's command line (see Appendix B). Additionally, the main computations performed using the GUI are callbacks to the plain code. Consequently, any change made to the code is immediately reflected in the GUI, leading to a more flexible and adaptable tool. Overall, there are more than 10^4 lines of genuine code (built-in functions), all encapsulated in the GUI.

MATLAB was selected as programming language due to its excellent linear algebra, visualizing and debugging capabilities, extensive documentation, active community, and dedicated app development framework (App Designer). The code has been written using procedural techniques for their better performance compared to MATLAB Object-Oriented-Programming (OOP). Even though MATLAB is an interpreted language, which introduces a significant performance cost compared to other (compiled) languages [62], CT computational times are still competitive compared to similar codes. For instance, as shown in Section 4, CT is about one order of magnitude faster than the MATLAB version of Caltech's SD-Toolbox used with Cantera (written in C++). However, there is room for further improvement by isolating specific demanding tasks into C++ subroutines and accessing them from MATLAB using MEX files, an optimization technique that will be explored in future code releases.

Most CT routines have as first argument a variable called *self* that contains all the shared data required for the calculations. Thus, this variable has been organized in a hierarchical tree structure as shown in Fig. 1, namely:

- *self*: parent node; contains all the data of the code, e.g., databases, input values, and results.
- *Constants (C)*: contains constant values.
- *Elements (E)*: contains data of the chemical elements in the problem (names and indices for fast data access).
- *Species (S)*: contains data of the chemical species in the problem (names and indices for fast data access), as well as lists (cells) with the species for complete combustion.
- *Problem Description (PD)*: contains data of the problem to solve, e.g., initial mixture (composition, temperature, pressure), problem type, and its configuration.
- *Problem Solution (PS)*: contains results (mixtures).
- *Tuning Properties (TN)*: contains parameters that control the numerical error of the algorithms implemented in the different modules.
- *Miscellaneous (Misc)*: contains values that configure the auto-generated plots and export setup, as well as flags, e.g., setting `FLAG_RESULTS = true` (by default) the results are shown in the command window (only in the desktop environment).
- *Database master (DB_master)*: a structured thermochemical database including data from [31, 32].
- *Database (DB)*: a structured thermochemical database with *griddedInterpolant* objects (see MATLAB built-in function `griddedInterpolant.m`) that contain piecewise cubic Hermite interpolating polynomials (PCHIP) [63] for faster data access.

The use of *griddedInterpolant* objects in *DB* speeds up the data access by a factor of 200% with respect to the evaluation of NASA's polynomials. Furthermore, for temperatures outside the bounds, we avoid the higher order terms of the polynomials by linear extrapolation, similar to Ref. [5], extending the range of validity of the thermodynamic data available. It should be emphasized that this extension is limited to a narrow temperature range and may not apply to temperatures significantly outside of this range.

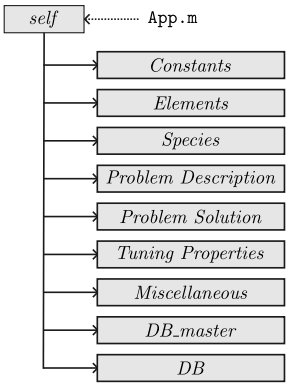


Figure 1: Combustion Toolbox hierarchical data tree structure, where `App.m` is the initialization function.

Data for nodes in the *self* parent node must be initialized before use. This can be easily done with one of the following sample statements:

```
self = App()
self = App('Soot formation extended')
self = App({'N2', 'O2', 'NO', 'N', 'O'})
self = App('Complete')
```

The first option, used by default, directs the code to select all possible species that can appear based on the elements present in the reactants (see routine `find_products.m`). The second option directs the code to use an expanded list of 94 species that typically appear in CHON mixtures, solid carbon $C_{(gr)}$ included.

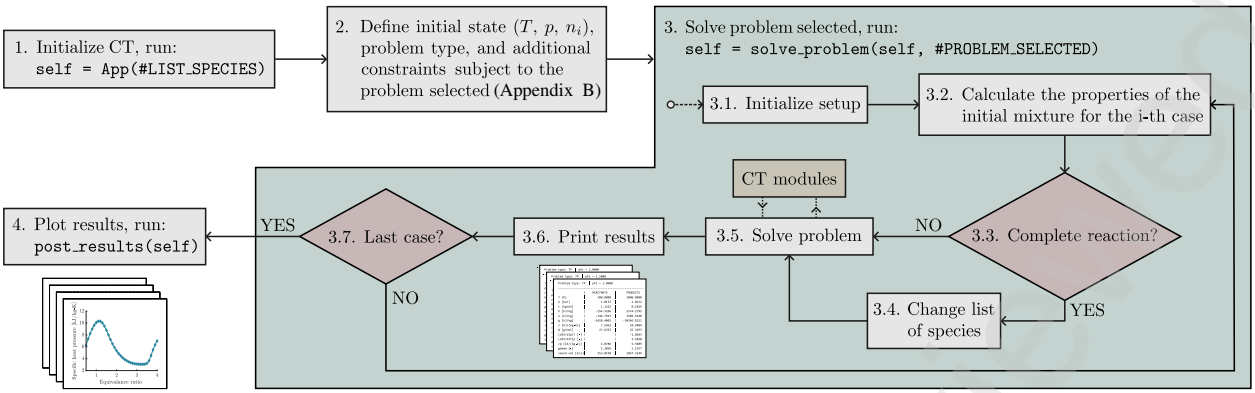


Figure 2: Combustion Toolbox simplified workflow.

The third option demonstrates how to limit the computations to a specific set of species. Finally, by specifying either *complete* or *complete_reaction*, the code carries out calculations under the assumption of complete combustion, which considers only the seven major species involved in $C_xH_yO_zN_w$ systems: CO_2, CO, H_2O, H_2, O_2, N2, and $C_{(gr)}$. This option is particularly relevant for the solution of combustion problems in academic contexts. Solid carbon $C_{(gr)}$ is known to appear in the products at equilibrium whenever the equivalence ratio ϕ is sufficiently large, $\phi \geq \phi_c$. The code estimates the critical equivalence ratio as $\phi_c \approx 2/(x - z)(x + y/4 - z/2)$ and, if the above inequality is satisfied, modifies the species list used for the calculations automatically (see functions `Species.m` and `define_F.m`).

The source code of the Combustion Toolbox is organized into several top-layer folders: *databases*, *examples*, *gui*, *installer*, *modules*, *utils*, and *validations*. The *databases* folder mainly consists of raw data and `.mat` files that contain the thermochemical properties of the individual chemical species [31–33]. The *examples* folder includes various examples that demonstrate the wide variety of problems that can be solved with CT. The *gui* folder contains the routines that are specifically designed for the GUI. The *installer* folder contains all the installation files of the GUI: the MATLAB toolbox and the royalty-free stand-alone version. This step is straightforward (see `INSTALL.m`), and for additional information we refer to the CT website (https://combustion-toolbox-website.readthedocs.io/en/latest/install.html). The *modules* folder contains the functions of the different modules, CT-EQUIL, CT-SD, and CT-ROCKET, as well as the routines for initializing CT. The *utils* folder houses utility functions with different purposes. Finally, the *validation* folder includes the routines used to validate CT with the results obtained with other codes, the unit testing files to ensure the correct functionality of the code, and all the graphs generated from these verifications.

Figure 2 summarizes the main steps required to solve a problem with CT. First, the system must be initialized using one of the statements indicated above, which defines the list of chemical species involved. Second, one must detail the initial state of the mixture (temperature, pressure, and molar composition) and the type of problem. Depending on the constraints of the selected problem, additional parameters may be required, e.g., for planar shocks, it is necessary to specify the pre-shock velocity in m/s or, equivalently, the pre-shock Mach number. When the setup is finished, CT calls the routine `solve_problem.m` to compute the results of the posed problem. Lastly, CT shows predefined plots using the function `post_results.m`, and it can save the results into a spreadsheet or as a `.mat` file (see routine `Miscellaneous.m`).

2.2. Collaborative framework and version control system

The collaborative process of open-source code can greatly benefit from the contributions of other authors. However, in the absence of adequate tools, this process may become disorderly. To mitigate this risk, we employ Git as version control system (VCS) and GitHub as online hosting service. These technologies enable comprehensive tracking of all changes made to the code while maintaining complete transparency throughout the development process [64–67]. To further ensure the integrity of the package, all contributions are subject to rigorous testing prior to being merged into the two primary branches (`master` and `develop`) by using GitHub Actions, giving a powerful tool to retrace errors in the code.

2.3. Documentation

A notable fraction of open-source codes lack sufficient documentation, which impedes the code's usability and accessibility. To amend this issue, we use Sphinx [68], a documentation generator written and used by the Python community, along with its MATLAB-domain extension [69]. All function headers are written following Google's Python-style docstrings. The online documentation is hosted on Read the Docs and is regularly updated from the GitHub repository. The new routines are automatically included in the online documentation using GitHub Actions. The framework allows having specific documentation for each distributed version. Additionally, the package includes several examples and all the validations carried out with other codes.

2.4. Benchmarks

The calculations presented in this work were performed on a laptop computer with the following specifications: Intel(R) Core(TM) i7-11800H CPU @ 2.30GHz with 8 physical cores

and 64GB of RAM, running on a 64-bit Windows 11 Pro system and using MATLAB R2022b. The computation time represents the elapsed time from CT's initialization until the end of the calculations. For reference, only loading the databases (~ 3600 species with their *griddedInterpolant* objects) takes an average of 0.9685 seconds and is included in the times presented in this manuscript.

3. Thermochemical equilibrium module

This section presents the kernel module of the code, CT-EQUIL, namely the thermochemical equilibrium module. This module is composed of four main routines:

- `equilibrium_gibbs.m`
- `equilibrium_helmholtz.m`
- `equilibrate_T.m`
- `equilibrate.m`

The first two are described in Sections 3.1 and 3.2 and are used to compute the chemical equilibrium composition for given temperature-pressure (TP), or temperature-volume (TV) states, respectively. The third is employed to compute the thermodynamic properties of the mixture and is an upper layer of the previous routines, as shown in Algorithm 1. The latter, described in Section 3.3, represents the top layer of CT-EQUIL and is implemented to compute the chemical equilibrium composition and thermodynamic properties for any of the following pairs of specified state functions: TP, HP, SP, TV, EV, and SV, where T stands for temperature, P for pressure, H for enthalpy, S for entropy, E for internal energy, and V for volume.

Combustion Toolbox enables the computation of chemical equilibrium under various assumptions regarding the final gas mixture, including calorically perfect gas, calorically imperfect gas with frozen chemistry, or calorically imperfect gas with equilibrium chemistry, including dissociation and ionization. An example of these calculations is presented in Section 4. It is also possible to freeze the composition of a subset of the species considered as possible products at equilibrium by defining them as inert species.

3.1. Equilibrium composition at specified temperature and pressure (TP)

The `equilibrium_gibbs.m` routine computes the molar equilibrium composition $\mathbf{n} = \{n_1, n_2, \ldots, n_{\mathrm{NS}}\}$ of a mixture $\mathbf{S} = \{S_1, S_2, \ldots, S_{\mathrm{NS}}\}$ of NS species at a given temperature T and pressure p for a closed system by minimizing the Gibbs free energy of the system $G(T, p, \mathbf{n})$. This is an equality-constrained problem (ECP) subject to mass conservation, namely

$$\min\, G(T,p,\mathbf{n}) \;\Leftrightarrow\; \mathrm{d}G(T,p,\mathbf{n}) = \sum_{j\in\mathbf{S}} \mu_j(T,p,\mathbf{n})\,\mathrm{d}n_j = 0 \quad \text{(1a)}$$

$$q_i = \sum_{j\in\mathbf{S}} a_{ij} n_j - b_i^\circ = 0, \quad \forall i \in \mathbf{E} \quad \text{(1b)}$$

where n_j and μ_j are the number of moles and chemical potential of species j, respectively, a_{ij} are the stoichiometric coefficients, i.e., the number of atoms of element i per molecule of species j, and b_i° is the number of atoms of the i-th element in the initial mixture. There are as many linear constraints q_i as NE elements $\mathbf{E} = \{E_1, E_2, \ldots, E_{\mathrm{NE}}\}$ involved. The NS species can be either gaseous or condensed, assuming pure components. Thus, there are NG gaseous species $\mathbf{S^G} = \{S_1, S_2, \ldots, S_{\mathrm{NG}}\}$ and NS − NG condensed species $\mathbf{S^C} = \{S_{\mathrm{NG+1}}, S_{\mathrm{NG+2}}, \ldots, S_{\mathrm{NS}}\}$, where $\mathbf{S^G}$ and $\mathbf{S^C}$ are the respective subsets of $\mathbf{S}$. In addition, the problem must satisfy NE ≤ NS. Equation (1a) must be supplemented with an EoS to define the thermodynamic functions. Our code implements the ideal gas EoS for the chemical potential

$$\mu_j(T,p,\mathbf{n}) = \mu_j^\circ(T) + \kappa_j RT\left(\ln\frac{n_j}{\sum_{j\in\mathbf{S^G}} n_j} + \ln\frac{p}{p^\circ}\right), \quad \forall j \in \mathbf{S} \quad \text{(2)}$$

where $\mu_j^\circ(T)$ is the chemical potential of species j at the reference pressure (p° = 1 bar) and the specified temperature, κ_j is either one or zero depending on whether the species is in gaseous or condensed phase, and R is the universal gas constant. As stated above, the CT-EQUIL module computes all thermochemical properties within the ideal gas approximation using an up-to-date version of NASA's 9-coefficient polynomial fits [31] that incorporates the Third Millennium database [32], including the available values from Active Thermochemical Tables. For more details on the calculation of the thermochemical properties from the values contained in these databases the reader is referred to Appendix A.

The ECP formulated in Eqs. (1) and (2) is solved using the method of Lagrange multipliers (an extended description thereof can be found in [70]), which introduces the Lagrangian function

$$\mathcal{L}(T,p,\mathbf{n},\boldsymbol{\lambda}) = G(T,p,\mathbf{n}) + \boldsymbol{\lambda}\,\mathbf{q}(\mathbf{n}), \quad \text{(3)}$$

where $\boldsymbol{\lambda}$ represents the multiplier vector, of length NE. If there is an infinitesimal change, $\mathrm{d}\mathbf{n}$ and $\mathrm{d}\boldsymbol{\lambda}$, the differential of $\mathcal{L}$ can be obtained from Eq. (3) with use made of (1) to yield $\mathrm{d}\mathcal{L} = 0$, namely

$$\sum_{j\in\mathbf{S}}\left[\mu_j(T,p,\mathbf{n}) + \sum_{i\in\mathbf{E}} a_{ij}\lambda_i\right]\mathrm{d}n_j + \sum_{i\in\mathbf{E}}\left[\sum_{j\in\mathbf{S}} a_{ij}n_j - b_i^\circ\right]\mathrm{d}\lambda_i = 0. \quad \text{(4)}$$

Additionally, the sum of the molar compositions must equal the total number of moles of gaseous species in the system, thus

$$\sum_{j\in\mathbf{S^G}} n_j = n. \quad \text{(5)}$$

Considering that $\mathrm{d}n_j$, $\mathrm{d}\lambda_i$, and n are independent, Eqs. (4) and (5) constitute a system of NS non-linear equations subject to a set of NE + 1 linear constraints. Furthermore, to ensure that

the molar composition n_j is strictly positive, it is convenient to work with the functions $\ln n_j$ and $\ln n$ in the gaseous species. There is no need to apply these definitions to the condensed-phase species because the algorithm solves the ECP problem for the gas phase first, and then considers the condensed species without specifying an initial estimate of their composition.

Using these definitions, Eqs. (4) and (5) can be rearranged in the form $\mathbf{f}(\mathbf{x}) = 0$, where $\mathbf{x}$ is the vector of unknowns composed of $\ln n_j$ (for $j \in \mathbf{S^G}$), n_j (for $j \in \mathbf{S^C}$), $\ln n$, and $\pi_i = -\lambda_i/RT$ (for $i \in \mathbf{E}$). To solve this system of equations, Combustion Toolbox uses a multidimensional Newton-Raphson (NR) method

$$\mathbf{J} \cdot \delta\mathbf{x} = -\mathbf{f}(\mathbf{x}), \tag{6}$$

where $\mathbf{J}$ is the Jacobian matrix with components $J_{ij} \equiv \partial f_i/\partial x_j$, $\delta\mathbf{x}$ is the correction vector composed of $\Delta \ln n_j$ (for $j \in \mathbf{S^G}$), Δn_j (for $j \in \mathbf{S^C}$), $\Delta \ln n$ and $\Delta\pi_i$, and $\mathbf{f}$ is the vector function. Equation (6) can be expanded as the following system of NS + NE + 1 linear equations

$$\Delta \ln n_j - \Delta \ln n - \sum_{i\in\mathbf{E}} a_{ij}\Delta\pi_i = -\frac{\mu_j}{RT}, \quad \forall j \in \mathbf{S^G} \tag{7a}$$

$$\sum_{i\in\mathbf{E}} a_{ij}\Delta\pi_i = \frac{\mu_j}{RT}, \quad \forall j \in \mathbf{S^C} \tag{7b}$$

$$\sum_{j\in\mathbf{S^G}} a_{ij}n_j\Delta \ln n_j + \sum_{j\in\mathbf{S^C}} a_{ij}\Delta n_j = b_i^\circ - a_{ij}n_j, \quad \forall i \in \mathbf{E} \tag{7c}$$

$$\sum_{j\in\mathbf{S^G}} n_j\Delta \ln n_j - n\Delta \ln n = n - \sum_{j\in\mathbf{S^G}} n_j, \tag{7d}$$

where the dimensionless Lagrange multiplier π_i has been taken equal to zero in the right hand side of Eqs. (7a) and (7b). This is a suitable simplification as long as λ_i appears linearly in the first square bracket of Eq. (4), as discussed in Ref. [11]. Algebraic manipulation of (7) allows to reduce the system's dimensions due to the spareness of the upper left corner of the Jacobian matrix $\mathbf{J}$. Thus, substituting $\Delta \ln n_j$ from (7a) in (7c) and (7d), NG equations are drop out from (7), providing a reduced system of NE + NS − NG + 1 equations, namely

$$\sum_{i\in\mathbf{E}}\sum_{j\in\mathbf{S}} a_{lj}a_{ij}n_j\Delta\pi_i + \left(\sum_{j\in\mathbf{S^G}} a_{lj}n_j\right)\Delta \ln n + \sum_{j\in\mathbf{S^C}} a_{lj}\Delta n_j = b_l^\circ - \sum_{j\in\mathbf{S}} a_{lj}n_j + \sum_{j\in\mathbf{S^G}} \frac{a_{lj}n_j\mu_j}{RT}, \quad \forall l \in \mathbf{E} \tag{8a}$$

$$\sum_{i\in\mathbf{E}} a_{ij}\Delta\pi_i = \frac{\mu_j}{RT}, \quad \forall j \in \mathbf{S^C} \tag{8b}$$

$$\sum_{i\in\mathbf{E}}\sum_{j\in\mathbf{S}} a_{ij}n_j\Delta\pi_i + \left(\sum_{j\in\mathbf{S^G}} n_j - n\right)\Delta \ln n = n - \sum_{j\in\mathbf{S^G}} n_j + \sum_{j\in\mathbf{S^G}} \frac{n_j\mu_j}{RT}, \tag{8c}$$

which is solved using MATLAB's linear programming routines built on LAPACK [71]. The updated solution vector $\mathbf{x}$ at the $(k+1)$-th iteration is given by $\mathbf{x}_{k+1} = \mathbf{x}_k + \tau_k\,\delta\mathbf{x}_k$, where τ_k is the step size, or relaxation, parameter for the k-th iteration, defined in Ref. [15]. As previously indicated, the solution vector $\mathbf{x}_k$ has $\pi_i = 0$ (for $i \in \mathbf{E}$); consequently, it is not necessary to relax the new value obtained, i.e., $\pi_{i,k+1} = \Delta\pi_{i,k}$. Note that in Eq. (8), $\mathbf{x}$ is composed of n_j (for $j \in \mathbf{S^C}$), $\ln n$, and $\pi_i = 0$ (for $i \in \mathbf{E}$). Consequently, to update the terms of the gaseous species $\ln n_j$, the correction $\Delta \ln n_j$ must be obtained from (7a) after each solution of (8), which requires defining a set of initial estimates $\mathbf{x}_0$ for the molar number n_j of all the possible products. We proceed by setting the number of condensed species to zero, and assuming that initially the gaseous species appear with a uniform molar distribution $n_j = 0.1/\mathrm{NG}$, obtained by considering 1 g of mixture with an average molecular mass of 10 g/mol [15]. However, when performing parametric sweeps, CT starts with the moles of the gaseous species n_j obtained from the previous calculation, provided that their magnitudes exceed a predefined threshold value of 10^{-6}. This approach facilitates the convergence of the iterative procedure and accelerates the overall computational efficiency.

Camberos and Moubry [72] evaluated other initial guesses for the molar composition using, e.g., a probability density function (PDF) based on the thermal enthalpy. Nevertheless, they concluded that the uniform distribution was, in fact, the best estimate of the tested distributions due to its combined simplicity and effectiveness. Mathematically, the uniform distribution PDF represents the maximum uncertainty in the molar composition.

Once convergence is achieved (by default, the tolerance for the molar composition is set to 10^{-14} and for the NR is 10^{-5}), if there are condensed species in the set of products, a second iteration process is conducted. The procedure is similar, but now we include in the set of unknowns the condensed species that satisfy the vapor pressure test, namely

$$\frac{1}{RT}\frac{\partial\mathcal{L}}{\partial n_j} = \frac{\mu_j^\circ}{RT} - \sum_{i\in\mathbf{E}} \Delta\pi_i a_{ij} < 0, \quad \forall j \in \mathbf{S^C} \tag{9}$$

whose addition to the system will reduce the Gibbs free energy of the system even further, corresponding with the first term of Eq. (4) in dimensionless form. Note that if Eq. (9) yields negative values, the Lagrange function may not be at equilibrium ($\mathrm{d}\mathcal{L} = 0$), which means that the added species can appear at the final state of equilibrium. When several condensed species satisfy this condition, CT only includes the species with the minimum value of $(\partial\mathcal{L}/\partial n_j)/W_j$, as suggested Mcbride [73], where W_j represents the molecular mass of the species. If in the new equilibrium state that considers condensed species the molar composition n_j of the added species is negative, or the Jacobian $\mathbf{J}$ is a singular matrix, these species are omitted from the set $\mathbf{S^C}$. The process is repeated until all the condensed species in $\mathbf{S^C}$ that satisfy $T \in [T_{\min}, T_{\max}]$, i.e., whose temperature range is compatible with the system's temperature, have been tested.

In the presence of ionized gases, the algorithm neglects the Coulombic interactions associated with ideal plasmas. In this case, there is only an additional restriction that is given by the electroneutrality of the mixture [74]

$$\sum_{j\in\mathbf{S^G}} a_{ej}n_j = 0, \tag{10}$$

where the stoichiometric coefficient a_{ej} represents the number of electrons in ion j relative to the neutral species. Thus, for $\{e^-, N_2^+\}$ we have $a_{ej} = \{1, -1\}$. This means that the electron E_e, with index e, is treated as an element. This assumption is valid only when the ion density is sufficiently small, i.e., for weakly ionized gases. The code directly detects if there are ions in the set of possible products **S** and calls another subroutine that ensures that condition (10) is met.

Algorithm 1 Pseudocode to compute the final equilibrium composition of an initial mixture mix_1, for given temperature-pressure (TP), or given temperature-volume (TV). For problems at constant volume, p is not used.

function `equilibrate_T`($self$, mix_1, p, T)
  Step 1. Check settings ▷ `ProblemDescription.m`
  if FLAG_TCHEM_FROZEN **then** ▷ calorically perfect
    $mix_2 \leftarrow$ `equilibrate_T_tchem`($self$, mix_1, p, T)
    `return`
  end if
  if FLAG_FROZEN **then** ▷ calorically imperfect frozen
    $mix_2 \leftarrow$ `equilibrate_T_frozen`($self$, mix_1, p, T)
    `return`
  end if
  Step 2. Set list of species LS for calculations
    $LS \leftarrow$ `set_LS_original`($self$)
  Step 3. Solve ECP at constant TP or TV
  if TP **then** ▷ calorically imperfect dissociation/ionization
    $n_j \leftarrow$ `equilibrium_gibbs`($self, p, T, mix_1$)
  else
    $n_j \leftarrow$ `equilibrium_helmholtz`($self, v, T, mix_1$)
  end if
  Step 4. Add inert species to n_j
    $n_j \leftarrow n_{j,\text{inert}}$
  Step 5. Compute property matrix $\boldsymbol{M_0}$
    $\boldsymbol{M_0} \leftarrow$ `set_species`($self, LS, n_j, T$)
  Step 6. Compute properties of the mixture
    $mix_2 \leftarrow$ `compute_properties`($self$, $\boldsymbol{M_0}$, p, T)
end function

3.2. Equilibrium composition at specified temperature and volume (TV)

For calculating the molar equilibrium composition of the mixture at a given temperature T and volume v, we have to minimize the Helmholtz free energy of the system, defined as $F = G - pv$. Upon use of this relation into the minimization condition $\mathrm{d}F(T, v, \mathbf{n}) = 0$, we get

$$\sum_{j\in\mathbf{S}} \mu_j(T, v, \mathbf{n})\, n_j - pv = 0 \tag{11}$$

to be used in substitution of Eq. (1a). For convenience, the chemical potential of species j is now expressed as a function of the mixture's volume v. For an ideal gas EoS we have

$$\mu_j(T, v, \mathbf{n}) = \mu_j^\circ(T) + \kappa_j RT \left(\ln \frac{n_j}{\sum\limits_{j\in\mathbf{S^G}} n_j} + \ln \frac{n_j RT}{p^\circ v} \right), \quad \forall j \in \mathbf{S} \tag{12}$$

which ultimately gives a different reduced system of equations

$$\sum_{i\in\mathbf{E}} \sum_{j\in\mathbf{S}} a_{lj} a_{ij} n_j \Delta\pi_i + \sum_{j\in\mathbf{S^C}} a_{lj} \Delta n_j = b_l^\circ - \sum_{j\in\mathbf{S}} a_{lj} n_j + \sum_{j\in\mathbf{S^G}} \frac{a_{lj} n_j \mu_j}{RT}, \quad \forall l \in \mathbf{E} \tag{13a}$$

$$\sum_{i\in\mathbf{E}} a_{ij} \Delta\pi_i = \frac{\mu_j}{RT}, \quad \forall j \in \mathbf{S^C}. \tag{13b}$$

to be solved instead of (8).

Unlike in the TP calculations, the linear system no longer includes the total number of moles, n, and its correction factor, $\Delta \ln n$, as they are drop out of the system of NE + NS − NG dimensions written above. Here, μ_j is given by Eq. (12) and the correction values $\Delta \ln n_j$ do not depend of $\Delta \ln n$, yielding

$$\Delta \ln n_j = \sum_{i\in\mathbf{E}} a_{ij} \Delta\pi_i - \frac{\mu_j}{RT}, \quad \forall j \in \mathbf{S^G}. \tag{14}$$

As indicated in Algorithm 1, the computation of the chemical composition and thermodynamic properties of a given mixture at specified temperature and volume is performed by the routine `equilibrium_helmholtz.m`.

3.3. Equilibrium composition for other pairs of state functions

In many practical applications, the equilibrium temperature of a system is not initially determined, thereby necessitating the provision of supplementary information to close the problem. This additional information may be obtained from an enthalpy, internal energy, or entropy conservation equation, subject to the requirement that the corresponding state function f remains unchanged, namely

$$\Delta f(T) \equiv f_\mathrm{F}(T) - f_\mathrm{I}(T_\mathrm{I}) = 0, \tag{15}$$

where the subscripts F and I refer here to the final and initial states of the mixture, respectively. Unlike in NASA's CEA code, we have increased the flexibility of the CT-EQUIL module by decoupling this additional equation and retrieved the new condition by using a second-order NR method

$$T_{k+1} = T_k - \frac{f(T_k)}{f'(T_k)}. \tag{16}$$

The derivatives of the state functions $f'(T)$ involved in the different transformations can be expressed analytically in the form: $(\partial h/\partial T)_p = c_p$, $(\partial e/\partial T)_v = c_v$, $(\partial s/\partial T)_p = c_p/T$, and $(\partial s/\partial T)_v = c_v/T$, for HP, EV, SP, and SV transformations, respectively. Following common practice, h, e, s, c_p, and c_v denote the enthalpy, internal energy, entropy, and the specific

heats at constant pressure and constant volume, respectively. It is worth noting that that although they are written in lower case letters, these variables refer here to extensive magnitudes.

The initial estimate T_0 is computed using a *regula falsi* method. Nevertheless, when carrying out parametric studies, the program uses the temperature obtained in the previous calculation as an initial estimate to accelerate convergence toward the new solution. However, if T_k is significantly distant from the actual solution, this approach can lead to unsatisfactory convergence. To overcome this issue, we can select a more robust root-finding method, as Eq. (15) has been purposely decoupled. In particular, the code has implemented an implicit third-order Newton-Steffensen root-finding algorithm [75], defined as follows:

$$T_{k+1} = T_k - \frac{f^2(T_k)}{f'(T_k)\left[f(T_k) - f\left(T^*_{k+1}\right)\right]}, \tag{17}$$

where the temperature at the $(k+1)$-th iteration, T_{k+1}, is re-estimated by using the provisional value T^*_{k+1} provided by the application of the classical method defined in Eq. (16). The convergence criterion $\max\{|(T_{k+1} - T_k)/T_{k+1}|, |\Delta f/f_F|\} < \epsilon_0$ is set by default to 10^{-3} in both methods and is generally reached in two to five iterations. Algorithm 2 describes the pseudocode to calculate the equilibrium composition for any given pair of state functions.

Algorithm 2 Pseudocode to compute the final equilibrium composition of an initial mixture mix_1 at pressure p, for a given pair of state functions XY (included in $self$ variable). For problems at constant volume, p is not used. In TP/TV problems, steps 1-3 are omitted.

```
function equilibrate(self, mix_1, p)
    Step 1. Get attribute        ▷ e.g., enthalpy 'h' in case of HP
        att ← get_attr_name(self)                ▷ depends on X
    Step 2. Get initial estimates of T
        T ← regula_guess(self, mix_1, p, att)
    Step 3. Solve Eq. (15)
        switch @root_method do  ▷ TuningProperties.m
            case newton                  ▷ second-order method
                T ← newton(self, mix_1, p, att, T)
            case nsteff                  ▷ third-order method
                T ← nsteff(self, mix_1, p, att, T)
    Step 4. Solve ECP at constant TP or TV
        mix_2 ← equilibrate_T(self, mix_1, p, T)
end function
```

3.4. Validations

To illustrate the wide variety of applications of Combustion Toolbox and asses the capabilities of the CT-EQUIL module, several validation tests have been conducted. In this work, we provide three of them in which only a reduced set of species are presented for clarity. Further validation tests can be easily accessed through the CT website or by just utilizing the user-interface validation add-on, *uivalidation*, which is implemented in the GUI (see Fig. 17 below). Alternatively, the user can also run the scripts included in the *validations* folder.

First test: In planetary science, thermochemical equilibrium codes like TEA [23], Fastchem [5, 7], and GG_{CHEM} [6], are used to model the atmospheric composition of giant planets, brown dwarfs, and other celestial bodies. Further examples can be found in Refs. [76–79]. Such models can help to unveil the physicochemical processes (chemical and radiative) that drive the evolution of these atmospheres, such as the formation of clouds and the escape of atmospheric gases into space [80]. This motivates the first validation case: the composition of the hot-Jupiter exoplanet WASP-43b's atmosphere. To this end, it is necessary to provide temperature and pressure profiles, as well as the planet's metallicity, which refers to the abundance of elements heavier than hydrogen and helium present in its composition. For example, Fig. 3 shows the variation of the species molar fractions $X_j = n_j / \sum_{j \in S} n_j$ with pressure (right panel) corresponding to the temperature-pressure profile given in [81] (left panel), and assuming an atmospheric 50× solar metallicity. This value is needed to determine the mixture's initial number of moles n_j, upon knowledge of the elemental solar abundances that are here estimated from [82], which considers H the reference element. A specific routine, `read_abundances.m`, reads the solar mass abundances that are then converted into molar abundances using the function `abundances2moles.m`, which considers a metallicity equal to unity by default. It is important to note that different exoplanets may necessitate the utilization of distinct solar abundances. Nevertheless, the program is compatible with additional datasets that follow the same format as the original dataset: `abundances.txt` (located in the *database* folder). The results (solid lines) show an excellent agreement with those obtained with the recently developed Thermochemical Equilibrium Abundances (TEA) code [23] (symbols) even down to the μbar level. The minor differences in HCN, C_2H_2 (acetylene), and HS_M (the subscript M denotes that is obtained from Burcat's database) come from the discrepancies of the free energies compared to the NIST-JANAF database [29, 30] that is implemented in TEA. In this test, the computation time with a tolerance of 10^{-32} for the molar composition was 1.17 seconds (TEA: 6.42 seconds) for a set of 26 species considered and a total of 90 case studies, which represents a 5.8× speed-up factor for our code (22× loading a specific database for this case).

Second test: As previously indicated, thermochemical codes are a crucial tool for understanding and predicting the intricate chemistry that takes place during combustion processes. As a result, it is essential to validate CT with a canonical combustion test. With this purpose, the second validation test involves the investigation of the adiabatic isobaric combustion of acetylene and air, a potential mixture for advanced internal combustion engines owing to the high energy density and low carbon content of acetylene. Specifically, the investigation focuses on the isobaric reactive mixture at an initial pressure of $p_1 = 1$ atm and temperature of $T_1 = 300$ K, and considers a wide range of equivalence ratios $\phi \in [0.5, 4]$. Figure 4 shows the variation of the molar composition of the products with ϕ (top panel) along with other mixture properties (Figs. 4a-h). It should be noted that, unlike the previous test, in this case the results obtained with CT (represented by solid lines) are compared to those of NASA's CEA [15] (represented by symbols). Once again, the

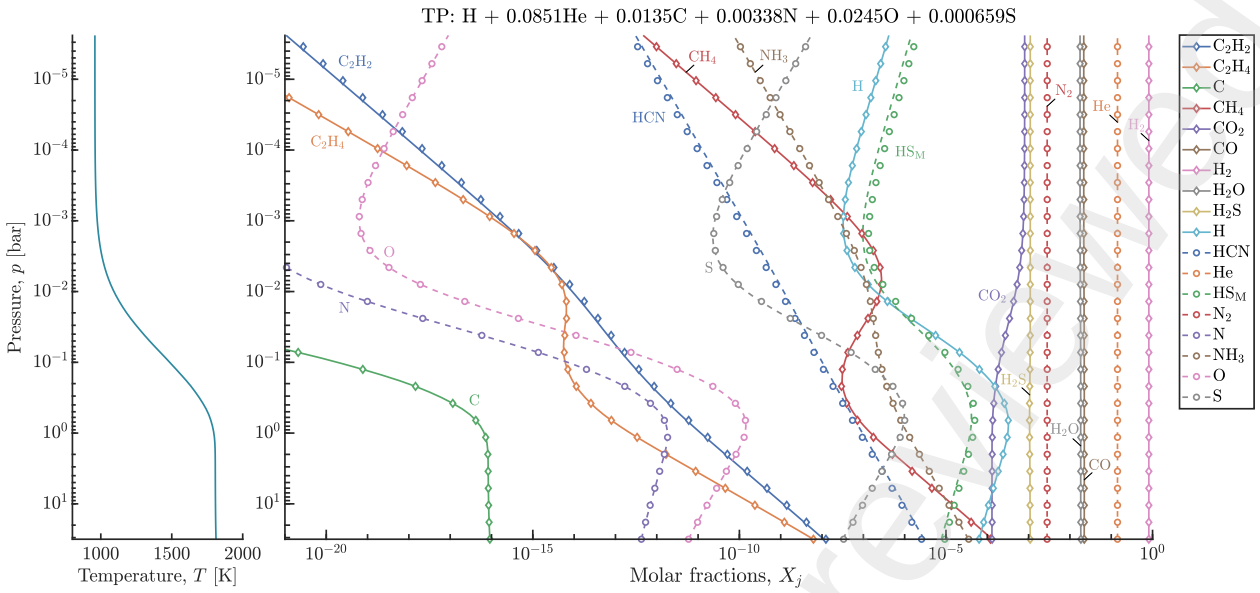


Figure 3: Variation of molar fraction with pressure (right panel) for the temperature-pressure profile of exoplanet WASP-43b (left panel) with an atmospheric 50× solar metallicity; solid line: numerical results obtained with CT; symbols: numerical results obtained with TEA [23]. The species denoted with subscript M is obtained from Burcat's database [32].

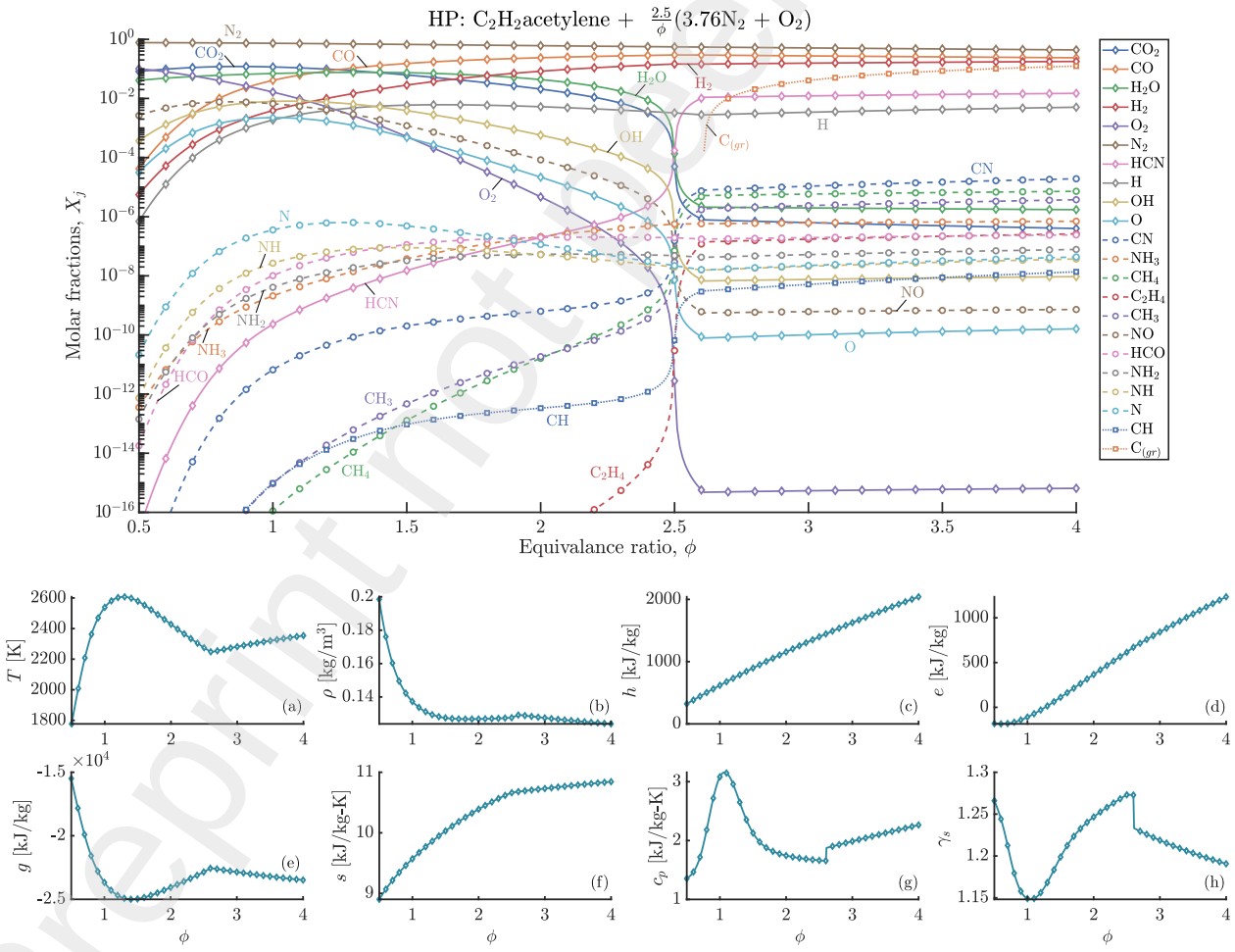


Figure 4: Variation of the molar fractions, X_j (top), and of different thermodynamic mixture properties: (a) temperature, T, (b) density, ρ, (c) enthalpy, h, (d) internal energy, e, (e) Gibbs energy, g, (f) entropy, s, (g) specific heat capacity at constant pressure, c_p, and (h) adiabatic index, γ_s, for an HP transformation in lean-to-rich acetylene (C_2H_2)-air mixtures at standard conditions (T_1 = 300 K, p_1 = 1 bar); solid line: numerical results obtained with CT; symbols: numerical results obtained with NASA's CEA [15]. The code snippet is shown in Appendix B Listing 1.

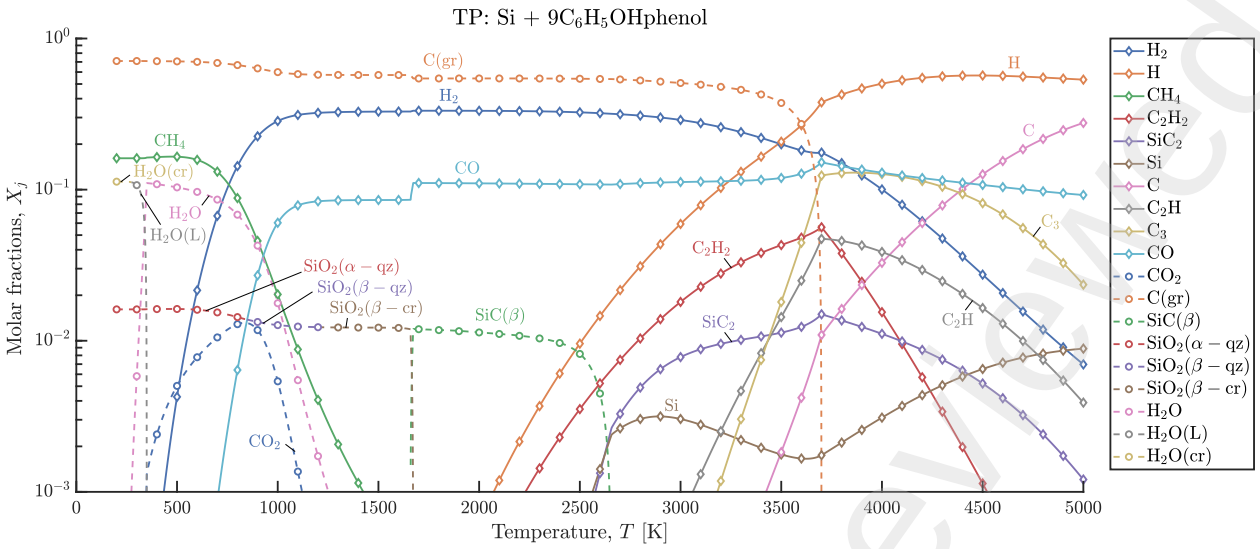


Figure 5: Variation of the molar fractions X_j for a Silica-Phenolic mixture at atmospheric pressure ($p = 1$ atm) with $T \in [200, 5000]$; solid line: numerical results obtained with CT; symbols: numerical results obtained with NASA's CEA [15].

results are in excellent agreement with the benchmark code, which includes the generation of solid carbon $C_{(gr)}$ when the equivalence ratio reaches or exceeds $\phi \approx 2.6$. This value is near the theoretical value $\phi_c = 2.5$ predicted by the complete combustion approximation. The computation time elapsed 4.57 seconds for a set of 94 species and 351 case studies. For this test, the tolerance was set to 10^{-18} for the molar composition and 10^{-3} for the root-finding method.

Third test: Recent advancements in chemical equilibrium solvers have opened up new avenues for studying the complex phenomena that take place on the surface of ablator materials during atmospheric reentry. For instance, Helber et al. [83] used a chemical equilibrium code to investigate the ablation of carbon-based materials under the conditions experienced during Earth's atmospheric reentry. Other studies focused on carbon-fiber-reinforced-polymers (CFRP) [84] or silicon-based ablative materials [85]. Regardless of the composition of the ablator, predicting the equilibrium species during ablation is a challenging task, as it involves the evaluation of numerous condensed species at very high enthalpies. To assess the capabilities of CT under more demanding scenarios than those of the previous tests, we recall in this third test the example presented in Ref. [22]. The problem consists of a parametric study of a Si-C_6H_5OH mixture at atmospheric pressure ($p = 1$ atm) for a wide range of temperatures $T \in [200, 5000]$. The variation with temperature of the molar fractions X_j for the Silica-Phenolic mixture is shown in Fig. 5, which also shows the same results computed with NASA's CEA [15] code. It is readily seen that there is a total agreement of the results even for the multiple condensed species. Previous work [22] reported that NASA's CEA code was not able to converge for $T < 400$. However, this is only true when computing the parametric study, not the individual cases, whereas CT converges in both situations. In this test, the computation time was 5.08 seconds for a set of 178 species (21 in condensed phase) and 481 case studies. The molar composition's tolerance was 10^{-18}.

4. Shock and detonation module

This section presents the routines of the shock and detonation module, CT-SD. This module determines the post-shock equilibrium state of steady non-reactive and reactive shocks with arbitrary incidence angles, β (see inset on the right panel of Fig. 6b). The routines are based on the algorithm outlined in NASA's Reference Publication 1311 [15, Chapters 7-9] for the solution of normal shocks and detonation waves, $\beta = \pi/2$, along with the CT-EQUIL module described in the previous section.

4.1. Oblique shocks

Let us first consider the problem of an undisturbed, planar, normal shock wave. The pre-shock density, pressure, enthalpy, and velocity (in the reference frame attached to the shock) are denoted, respectively, as ρ_1, p_1, h_1, and u_1. The corresponding flow variables in the post-shock gases are denoted as ρ_2, p_2, u_2, and h_2. The well-known Rankine-Hugoniot (RH) relations for the variation of pressure and enthalpy are, respectively

$$p_2 = p_1 + \rho_1 u_1^2 \left(1 - \frac{\rho_1}{\rho_2}\right), \tag{18a}$$

$$h_2 = h_1 + \frac{u_1^2}{2}\left[1 - \left(\frac{\rho_1}{\rho_2}\right)^2\right]. \tag{18b}$$

These equations must be supplemented by the equation of state, in our case, the ideal EoS $p = \rho RT/W$, where $W = \sum_{j\in\mathbf{S}^G}(n_j/\sum_{j\in\mathbf{S}^G} n_j)W_j$ stands for the average molecular mass of the gaseous mixture computed in terms of the gas-phase molar fractions, $n_j/\sum_{j\in\mathbf{S}^G} n_j$, and the molecular masses of the gaseous species, W_j. Also required is the caloric EoS, $h = \sum_{j\in\mathbf{S}} n_j H_j^\circ(T)$, that gives enthalpy in terms of the temperature and gas mixture composition. As discussed above, the molecular masses, W_j, and the molar specific enthalpies, $H_j^\circ(T)$, are evaluated from a combination of NASA's [31] and Burcat's (Third Millennium) [32] thermochemical databases.

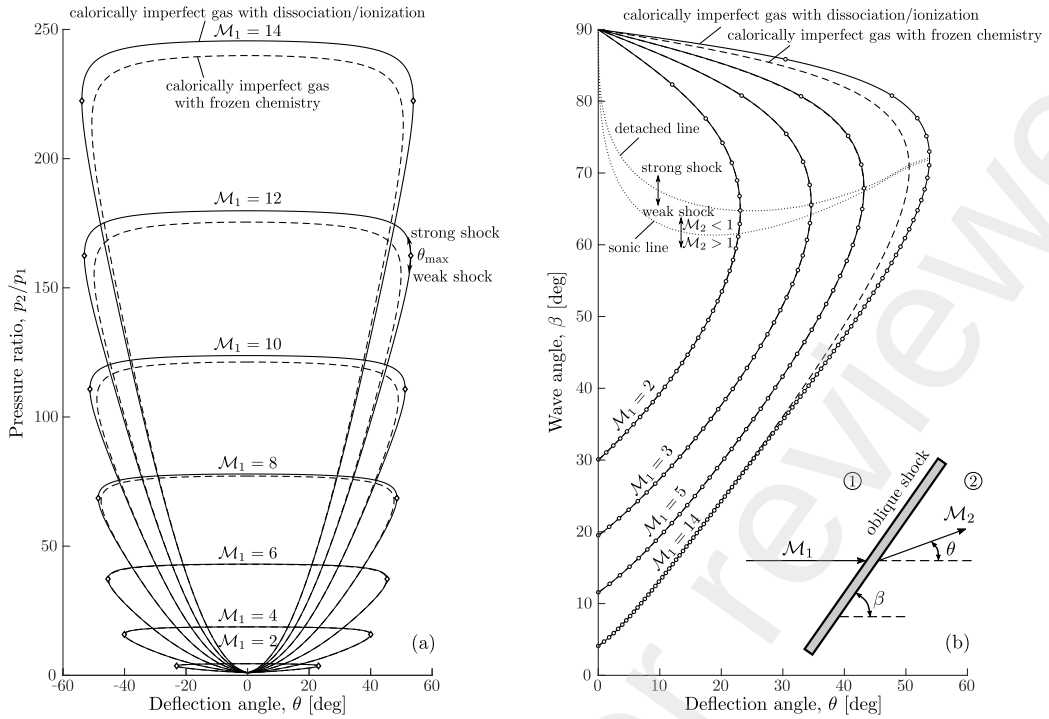


Figure 6: Pressure-deflection (a) and wave angle-deflection (b) shock polar diagrams for air (78% N_2, 21% O_2, and 1% Ar) at pre-shock temperature $T_1 = 300$ K and pressure $p_1 = 1$ atm, and a range of pre-shock Mach numbers $\mathcal{M}_1$ between 2 and 14; solid line: calorically imperfect gas with ionization/dissociation; dashed: calorically imperfect gas with frozen chemistry; circles: results obtained with Cantera [59] within Caltech's SD-Toolbox [61]; diamonds: maximum deflection angle $\theta_{\max}$. The code snippet is shown in Listing 2.

The solver employs a NR method (see Ref. [15, Chapters 7-9] for more information) to determine the roots of the system of equations governing both reactive and non-reactive shocks (see routines `det_cj.m` and `shock_incident.m`, respectively). Consequently, the jump relationships, such as p_2/p_1 and ρ_2/ρ_1, can be calculated with ease. In this subsection, we focus on the oblique shock configuration that implicitly includes the solution of the normal shock wave. When the angle formed by the shock plane and the upstream flow is not $\pi/2$, the pre- and post-shock velocities, u_1 and u_2, must be replaced by their respective components normal to the shock, u_{1n} and u_{2n}, in the RH equations (18). These equations can be readily reformulated in terms of the magnitudes of the pre- and post-shock velocities upon direct substitution of the trigonometric relationships $u_{1n} = u_1 \sin\beta$ and $u_{2n} = u_2 \sin(\beta - \theta)$, where β is the shock incidence angle and θ is the flow deflection angle, both measured with respect to the upstream flow direction. For oblique shocks, the continuity of the tangential velocity across the shock, $u_{1t} = u_{2t}$, or, equivalently, $u_1 \cos\beta = u_2 \cos(\beta - \theta)$, is also required.

In the case of a normal shock wave, the gas properties downstream of the shock are determined by two factors: the upstream thermodynamic state and a parameter that characterizes the intensity of the shock, usually either the shock speed u_1 relative to the upstream gas or the shock Mach number, $\mathcal{M}_1 = u_1/a_1$ (where a_1 represents the speed of sound upstream the shock), as given in Eq. (A.7). Other problems, such as those concerning blast waves, impose the post-shock pressure p_2 as the initial input parameter describing the shock intensity. Oblique shocks, on the other hand, require an additional geometrical restriction given by the value of the incidence angle β or the flow deflection angle θ. When the value of β is specified, there exists a unique solution for the post-shock fluid state (see routine `shock_oblique_beta.m`). However, in the case where θ is given, there are two possible solutions for β: a weak shock solution linked to a smaller value of β, representing the weakest possible shock, and a strong solution corresponding to a larger β (see routine `shock_oblique_theta.m`). The two solutions converge for $\theta = \theta_{\max}$, which corresponds to the maximum deflection angle of the shock for a given $\mathcal{M}_1$ (see Fig. 6), above which the problem does not admit solution. Remarkably close to the maximum deflection angle, in the weak-shock branch, we find the sonic condition $\mathcal{M}_2 = 1$, below and above which the post-shock flow is supersonic (weak shocks) and subsonic (strong shocks, and weak shocks between the sonic line and the maximum deflection angle), respectively. Then, the supersonic branch always corresponds to the weak shock solution.

In either scenario, when the value of θ is specified, β becomes an implicit variable that must be determined numerically. To this end, CT employs an iterative procedure based on the continuity of the tangential velocity across the shock, which can be manipulated using the above trigonometric identities to give

$$f(\beta) \equiv \theta + \tan^{-1}\left(\frac{u_{2n}}{u_1 \cos\beta}\right) - \beta = 0. \tag{19}$$

This equation must be solved for the shock incidence angle β (see Algorithm 5) with use made of the RH relations (18), the ideal gas EoS, and provided that $f'(\beta)$ and $f''(\beta)$ can be written

as explicit functions. This enables the use of Halley's third-order iterative method [86]

$$\beta_{k+1} = \beta_k - \frac{2f(\beta_k)\, f'(\beta_k)}{2f'(\beta_k) - f(\beta_k)\, f''(\beta_k)} \tag{20}$$

to find the root of Eq. (19). This approach exhibits rapid convergence and meets the default convergence criterion of 10^{-3} within two or three iterations. However, it is worth noting that like other root-finding methods, Halley's method only provides one of the possible roots of the nonlinear system, which generally corresponds to the root closest to the initial guess used during the iterative process. Therefore, we must supply sufficiently accurate guesses to cover the whole set of solutions that include both the weak- and strong-shock branches. One straightforward approach to acquiring such guesses is to anticipate the solution domain bounded by the acoustic weak-shock limit $\beta_{\min} = \sin^{-1}(1/\mathcal{M}_1)$ and the normal shock configuration $\beta_{\max} = \pi/2$. In particular, we choose $\beta_0 = 0.5\,(\beta_{\min} + \beta_{\max})$ for the weak-shock branch and $\beta_0 = 0.97\beta_{\max}$ for the strong-shock branch.

The left and right plots in Fig. 6 depict the pressure ratio-deflection angle and the incidence angle-deflection angle shock polar diagrams for dry air (consisting of 78% N_2, 21% O_2, and 1% Ar) initially at room conditions ($T_1 = 300$ K, $p_1 = 1$ atm). These results were obtained using the `shock_polar.m` routine. It is worth noting that CT provides users with a variety of gas models to choose from: *i)* calorically perfect gas with frozen chemistry (constant specific heat at constant pressure c_p, adiabatic index γ, and n_j), *ii)* calorically imperfect gas with frozen chemistry (constant n_j), and *iii)* calorically imperfect gas with variable composition, including dissociation, ionization, and recombination reactions at equilibrium. These effects are incorporated by specifying a sufficiently large set of species for the calculations, and using NASA's [31] and Burcat's (Third Millennium) [32] databases for evaluating the thermodynamic properties.

Figure 6 displays the results obtained with models *ii)* and *iii)* spanning a set of pre-shock Mach numbers $\mathcal{M}_1$ ranging from 2 to 14. The results are compared with Caltech's Shock and Detonation Toolbox [60], which uses Cantera [59] as kernel for the computations of chemical equilibrium. It is found that the lobes in the pressure ratio-deflection angle diagram expand due to dissociation/ionization effects, particularly in the hypersonic flow regime, $\mathcal{M}_1 > 5$. As a result, weak oblique shocks exhibit smaller pressure ratios while strong ones exhibit larger pressure ratios for the same deflection angle. Moreover, the endothermic (cooling) effect caused by dissociation/ionization in hypersonic oblique shocks leads to an increase in the post-shock density that also increases the wave deflection angle at all incidence angles. The results obtained from both codes are in complete agreement for all conditions tested. However, CT-SD exhibits superior performance compared to Caltech's SD-Toolbox with Cantera, reducing computation time by more than 95% (CT-SD: 4.12 s vs. Caltech's SD-Toolbox & Cantera: 99.72 s; for a large subset that contains 1200 points of all the cases represented in Fig. 6b, with both codes running on the same platform and with the same subset of 14 chemical species), which demonstrates the excellent performance of the CT-SD module.

4.2. Regular reflections

Understanding the reflection of shock waves off flat surfaces is a problem of great relevance to high-speed flows. The angle subtended by the incident shock and the flat surface determines the type of shock reflection, with $\beta = 0$ representing normal reflections and $0 < \beta < \pi/2$ oblique reflections. In a reference frame with origin at the point of contact of the shock with the wall, the latter exhibit an incoming free stream parallel to the wall with $\mathcal{M}_1 > 1$. For sufficiently small incidence angles, the incident shock deflects the free stream uniformly towards the wall an angle θ. The reflected shock then deflects back the perturbed stream to its original flow direction parallel to the wall. This type of reflection is said to be regular, and is depicted in Fig. 7. Regular reflections leave uniform flow patterns behind both the incident and reflected shocks that can be described with Combustion Toolbox.

By contrast, for incidence angles above a certain critical value, $\beta > \beta_{\max}(\mathcal{M}_1)$, the reflected wave, with $\mathcal{M}_2 < \mathcal{M}_1$ and thus a lower maximum deflection angle, is not able to deflect the flow back to its upstream direction parallel to the wall. This leads to the so-called irregular, or Mach, reflections, where the reflected and incident shocks merge into a single wave called the Mach stem, which connects the wall to the triple point where the three shocks meet. These reflections produce non-uniform flows that include a high-speed shear layer or slip-stream emanating from the triple point [87]. The properties of these flows cannot be determined solely by the polar-plot charts or the zero-dimensional RH equations and thus, are out of the scope of this work.

To calculate regular reflections, Combustion Toolbox uses the routine `shock_oblique_reflected_theta.m` to compute the incident wave by specifying the wave angle β (or the flow deflection θ) and the pre-shock velocity u_1. This results in the calculation of the post-shock state (2), which serves as pre-shock state for the reflected wave (see sketch on top of Fig. 7). If the incident shock is sufficiently strong, the transition to state (2) can result in significant thermochemical effects that may cause changes in the aerothermal properties compared to those of a calorically perfect fixed-composition gas. In this case, the values of $\mathcal{M}_2$ and θ may change accordingly. The reflected shock increases the gas pressure and temperature even further, and the properties in state (3) can be determined using the code routine employed for single oblique shocks imposing the counter deflection angle θ calculated for the incident shock. This guarantees that the streamlines in state (3) are parallel to the reflecting surface. If there is no solution for the reflected shock (which occurs for sufficiently large values of θ), irregular Mach reflections occur. As discussed above, these reflections involve non-uniform flow properties and non-steady solutions, and thus their computation is beyond the capabilities of CT.

As an illustrative example, Fig. 7 represents the pressure-deflection shock polar diagrams for a regular shock reflection in atmospheric air at 30 km above sea level with an incident Mach number of $\mathcal{M}_1 = 20$ and a deflection angle of $\theta = 35°$ under

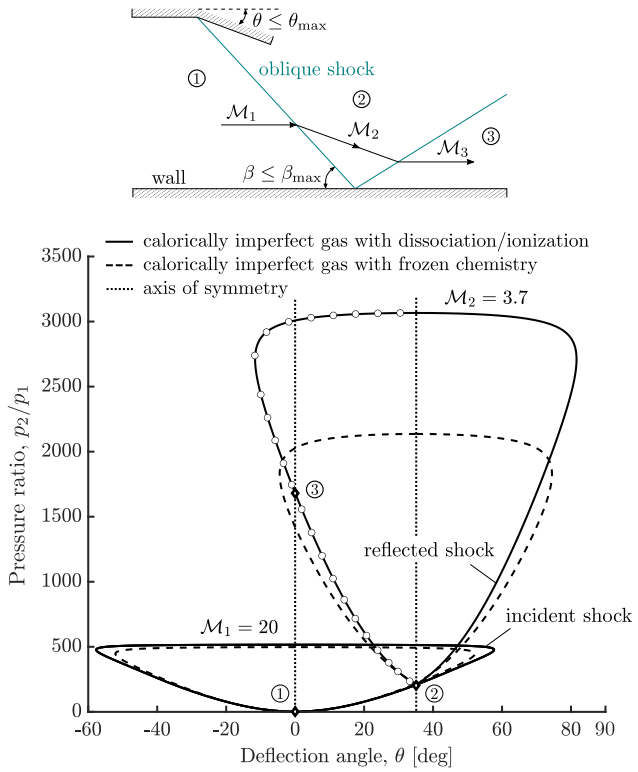


Figure 7: Pressure-deflection shock polar diagrams for a regular shock reflection in atmospheric air (78% N_2, 21% O_2, and 1% Ar) at 30 km above sea level (pre-shock temperature $T_1 = 226.51$ K and pressure $p_1 = 1.181 \cdot 10^{-2}$ atm), pre-shock Mach number $\mathcal{M}_1 = 20$, and deflection angle $\theta = 35°$; solid line: calorically imperfect gas with dissociation/ionization; dashed line: calorically imperfect gas with frozen chemistry; dotted line: axes of symmetry; circles: results of Zhang et al. [88]; diamonds: states 1, 2, and 3.

the same gas models *ii)* and *iii)* used in the oblique shock charts presented above. The polar plot starting from state (2), obtained by increasing the incident wave angle from 0 to $\pi/2$, determines the solution of state (3) for the deflection angle $\theta = 35°$. This, in turn, determines the reflected shock at the intersection of the second polar with the vertical axis ($\theta = 0$). As can be seen, the dissociation and ionization effects are more pronounced in the reflected shock, as the accumulated temperature jump in both shocks amplifies the endothermicity of the chemical reactions, resulting in substantially higher overall pressure ratios. Finally, the outcomes are compared with those acquired by Zhang et al. [88] under the same flow conditions, revealing excellent agreement in all instances. In our calculations, the computation time was 2.58 s (1.07 s) for a group of 28 species (3 species) and 200 case studies. These values depend on the tolerance, which was set to 10^{-14} for the molar composition and 10^{-5} for the root-finding method.

If the incident angle becomes larger, the polar diagram of the reflected shock moves farther away from the $\theta = 0$ axis, making it harder for the second shock to redirect the flow to its initial upstream direction parallel to the reflecting wall. For each $\mathcal{M}_1$, there exists a maximum value of β (and, consequently, of θ) beyond which regular reflection is impossible. This maximum value is determined by the condition at which the polar diagram of the reflected shock is tangent to the $\theta = 0$ axis. Our code is able to compute $\beta_{\max}$ and $\theta_{\max}$ in cases involving high-temperature thermochemical effects. To determine this limit, we impose the condition $\theta_{3,\max} - \theta = 0$, and employ an iterative algorithm based on Broyden's method [89], which makes use of the set of routines described above (see Algorithm 3 for further details).

Algorithm 3 Pseudocode to obtain pre-shock and post-shock states at the limit of regular reflections for a given mixture mix_1 and pre-shock velocity u_1.

function `shock_polar_limitRR`($self, mix_1, u_1$)
  Step 1. Get polar diagrams at state (1).
    $mix_{2,\text{polar}} \leftarrow$ `shock_polar`($self, mix_1, u_1$)
  Step 2. Set initial estimates
    $\theta \leftarrow \theta_{\max}/2$
    $f'(\theta) \leftarrow 2$ ▷ derivative of $f(\theta) \equiv \theta_{3,\max} - \theta$
    STOP, $k \leftarrow 1, 0$
  Step 3. Get θ_2 using Broyden's method
  while STOP $> \epsilon_{\text{limitRR}}$ & $k < k_{\max,\text{limitRR}}$ **do**
    $k \leftarrow k + 1$
    Step 3.1 Solve oblique shock (weak branch) for θ
      $mix_2 \leftarrow$ `shock_oblique_theta`($self, mix_1, u_1, \theta$)
    Step 3.2 Get polar diagrams at state (2)
      $mix_{3,\text{polar}} \leftarrow$ `shock_polar`($self, mix_2, u_2$)
    Step.3.3 Compute $f(\theta)$ and $f'(\theta)$
    Step.3.4 Update estimate θ
      $\theta \leftarrow \theta - f(\theta)/f'(\theta)$
    Step.3.5 Calculate STOP criteria
      $$\text{STOP} \leftarrow \max\left(\left|\frac{\theta_{k+1} - \theta_k}{\theta_k}\right|, \left|\frac{f(\theta_{k+1})}{\theta_{k+1}}\right|\right)$$
  end while
end function

Figure 8 represents the maximum incidence angle $\beta_{3,\max}$ for regular shock reflections as a function of the pre-shock Mach number $\mathcal{M}_1$ for three atmospheric conditions corresponding to increasing flight altitudes in the ISA model [90]. Results are presented for the three gas models defined above *i)*, *ii)*, and *iii)*. As expected, the upstream pressure and temperature do not have any effect on the calorically perfect gas solution, which can be written analytically for $\gamma = 1.4$. Nevertheless, the results including high-temperature thermochemical effects show large deviations from the calorically perfect gas solution. Thus, the endothermic effects associated with the dissociation of O_2 and N_2 are seen to increase the value of $\beta_{\max}$ and widen the domain of regular reflections. These effects occur primarily across the reflected shock, which by increasing its flow deflection angle also increases its ability to deflect the disturbed current back to its initial direction parallel to the surface. In conditions where endothermic effects occur mainly across the incident shock (for very high pre-shock temperatures or very high Mach numbers), the situation is reversed, and the maximum incidence angle for regular shock reflection exhibits a slight decrease.

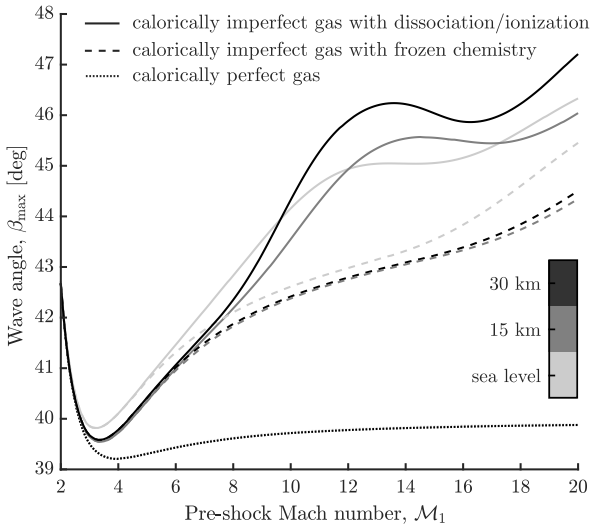


Figure 8: Maximum wave angle β_{max} in the limit of regular reflection as a function of pre-shock Mach number $\mathcal{M}_1$ for an air mixture (78% N_2, 21% O_2, and 1% Ar) in the atmosphere at different flight altitudes (0, 15, and 30 km) above sea level in the ISA model; solid line: calorically imperfect gas with dissociation/ionization; dashed line: calorically imperfect gas with frozen chemistry; dotted: calorically perfect gas.

4.3. Planar gaseous detonations

The thermochemical framework employed to describe shock waves involving endothermic molecular transformations can be easily extended to account for exothermic reactions, as occurs in planar detonations. As with shock waves, the computation of detonations requires knowledge of the pre-shock state and the degree of overdrive that measures the contribution of the external supporting mechanism. However, unlike shock waves, detonations can be self-sustained, i.e., propagate without any external contribution exerting additional pressure from behind. This propagation mode, named after Chapman-Jouguet (CJ), involves the maximum possible expansion of the hot products. Therefore, the burnt-gas state is obtained by imposing the sonic condition in the post-wave flow $\mathcal{M}_2 = 1$. If the pressure behind the detonation wave is larger than what is anticipated by the CJ condition, which can only be achieved using an external forcing mechanism, the detonation is considered *over-driven* and results in subsonic downstream conditions, with $\mathcal{M}_2 < 1$. Conversely, *under-driven* detonations occur when the burnt gas is in a supersonic state with $\mathcal{M}_2 > 1$, but this is not compatible with the internal structure of the detonation wave.

The over-driven/under-driven solutions can be determined numerically for a defined upstream mixture and a given degree of overdrive $\eta = u_1/u_{cj}$ (see functions `det_overdriven.m` and `det_underdriven.m`) by using the routines designed for CJ detonations (see `det_cj.m`) and normal shocks (see `shock_incident.m`). The former is necessary to determine the minimum velocity $u_1 = u_{cj}$ (or $\eta = 1$) required for a planar detonation to propagate, while the latter is employed to obtain the post-detonation state for a given degree of overdrive η.

Careful selection of the initial guesses is required to obtain the solutions for over-driven and under-driven detonations. For instance, $T_{2,guess}$ and $p_{2,guess}$ denote the estimated temperature and pressure after the detonation, and should be anticipated considering the significant variations arising from the degree of overdrive and the type of propagation mode. To obtain the initial guesses for over-driven detonations, $p_{2,guess}$ is computed using Eq. (18a), assuming a constant $\gamma = \gamma_1 = \gamma_{2,guess}$. This gives $p_{2,guess} = p_1(2\gamma\mathcal{M}1^2 - \gamma + 1)/(\gamma + 1)$. The temperature guess is based on the fact that, for sufficiently strong shocks, the kinetic energy downstream is much lower than upstream of the shock, $u_2^2/u_1^2 \sim (\rho_1/\rho_2)^2 \ll 1$. This simplifies Eq. (18b) to $h_{2,guess} = h_1 + u_1^2/2$, thus enabling the computation of $T_{2,guess}$ by solving the thermochemical equilibrium problem at specified enthalpy and pressure, $h_{2,guess}$ and $p_{2,guess}$. This approximation becomes more accurate with increasing degrees of overdrive, as the differences between u_1 and u_2 become more significant. This estimation method is the same as the one utilized in the incident normal shocks routine.

For under-driven detonations, a reasonable initial guess can be obtained by considering the range of acceptable values for the mean post-shock density. By defining the dimensionless parameter $\zeta \in (0, 1)$ to measure how close $\rho_{2,guess}$ is to the CJ state compared to the initial state, we can construct an initial guess for the density as follows: $\rho_{2,guess} = [\zeta/\rho_{2,cj}+(1-\zeta)/\rho_1]^{-1}$. The pressure and temperature values can then be determined using Eq. (18a) and the ideal gas EoS, respectively. It has been found that a value of $\zeta = 0.1$ is suitable for the set of Mach numbers tested. The pseudocode for under-driven/over-driven detonations is shown in Algorithm 4.

Algorithm 4 Pseudocode to solve under-driven detonations for a given mixture mix_1 and degree of overdrive η. For over-driven detonations step 3 is omitted.

function `det_underdriven`($self$, mix_1, η)
  Step 1. Solve CJ pre-shock and post-shock states
    $mix_{1,cj}, mix_{2,cj} \leftarrow$ `det_cj`($self, mix_1$)
  Step 2. Compute pre-shock velocity
    $u_1 \leftarrow u_{cj}\eta$
  Step 3. Calculate initial estimates of post-shock state
    $\rho_2 \leftarrow [\zeta/\rho_{2,cj} + (1-\zeta)/\rho_1]^{-1}$ ▷ $\zeta = 0.1$
    $p_2 \leftarrow$ Eq. (18a)
    $W_2 \leftarrow W_{2,cj}$ ▷ same as CJ post-shock state
    $T_2 \leftarrow p_2 W_2/(\rho_2 R)$ ▷ ideal EoS
  Step 4. Compute pre-shock and post-shock states for u_1
    $mix_1, mix_2 \leftarrow$ `shock_incident`($self, mix_1, u_1, mix_2$)
end function

Table 1 lists the CJ velocities of planar gaseous detonations computed by CT for various near-stoichiometric fuel-air/O_2 mixtures. The results are compared with the experimental data reported in the literature, showing good agreement in all cases. The validation with other codes is performed in the following subsection in the context of oblique detonations. However, for detonation velocities related to condensed-phase explosives, we refer to SimEx [58]. SimEx includes an extensive database of pure CHNO propellants and explosives, and its kernel from CT was adapted to solve the products' composition with the

ideal gas EoS following the norm UNE 31-002-94 [91]. SimEx also employs more complex computations based on the European Standard EN 13631-15 [92], which use the semi-empirical Becker–Kistiakowsky–Wilson (BKW) [93, 94] or the Heuzé (H9) EoS [95].

Mixture	ϕ	CT [m/s]	Exp. [m/s]	Source
CH_4-air	0.989	1797.21	1798.17	[96]
H_2-air	1	1965.45	1825.64	[97]
H_2-O_2	1	2838.12	2898.39	[98]
DME-O_2	1	2318.12	2299.71	[99]

Table 1: Chapman-Jouguet detonation propagation velocities for different near-stoichiometric fuel-air/O_2 mixtures at $p_1 = 1$ atm and $T_1 \sim 293.15$ K computed by Combustion Toolbox and measured experimentally by various authors.

4.4. Oblique gaseous detonations

Detonation waves can take on oblique configurations, which are less common compared to oblique shocks. However, such detonations are crucial in Oblique Detonation Wave Engines (ODWE) [100, 101], where the combustion process occurs along an oblique detonation that revolves around a cylindrical combustion chamber.

To compute oblique detonations, knowledge of the pre-shock state and the degree of overdrive caused by the supporting mechanism, typically a wedge deflecting a reactive supersonic stream and generating the oblique detonation, is required, just like in the oblique shocks. Thus, given the temperature, pressure, composition, and pre-shock velocity, one can calculate the detonation polar diagrams with the particularity that now the exothermicity of the reaction increases the number of possible solutions, as occurs for planar detonations [102, 103]. Then, for a given detonation angle β (or deflection angle θ), two solutions for the burnt-gas state can be found, associated with under-driven ($\mathcal{M}_{2n} > 1$) and over-driven ($\mathcal{M}_{2n} < 1$) conditions (see routines `det_oblique_beta.m` and `det_oblique_theta.m`).

At the Chapman-Jouguet condition the two solutions merge into a single solution. This state is characterized by a sonic normal component of the downstream velocity vector ($\mathcal{M}_{2n} = 1$). The corresponding values for the upstream Mach number and shock angle are $\mathcal{M}_1 = \mathcal{M}_{1,\mathrm{cj}}$ and $\beta = \beta_{\mathrm{cj}}$. Both in the under-driven and over-driven cases, the RH-equations only produce real solutions if the values for the upstream Mach number and shock angle are greater than the corresponding values for the CJ condition, $\mathcal{M}_1 \geq \mathcal{M}_{1,\mathrm{cj}}$ and $\beta \geq \beta_{\mathrm{cj}}$. For oblique detonations with angles in the range $\beta_{\mathrm{cj}} < \beta < \pi/2$, a given shock angle corresponds to two different deflection angles, namely θ_{over} and θ_{under}.

In the over-driven branch, the solution resembles that of an oblique shock (see Algorithm 5). The point where $\mathcal{M}_2 = 1$ is reached below the maximum deflection angle separating the strong and weak solutions. Since the solution is multi-valued, the computation of the different branches requires an accurate initial estimate regardless of the input parameter, be it the shock or the flow deflection angle (see Section 4.3). Figure 9 shows the pressure-deflection (a) and wave angle-deflection (b) polar diagrams for detonations in stoichiometric hydrogen (H_2)-air (79% N_2, 21% O_2) mixtures at pre-shock temperature $T_1 = 300$ K and pressure $p_1 = 1$ atm, for a range of pre-shock Mach numbers $\mathcal{M}_1$ between 5 and 10.

CT provides embedded functionalities for obtaining polar diagrams that characterize incident oblique detonations (see function `det_polar.m`). These functionalities perform a direct computation of a set of cases (100 by default) by sweeping the range of possible solutions for both the under-driven and over-driven branches, as depicted in Fig. 9. These results have been compared with the values from Zhang et al. [88], which are found to be in remarkable agreement. The computation time was 9.74 seconds for a set of 26 species and 1500 case studies, for a tolerance of 10^{-14} for the molar composition and 10^{-5} for the root-finding method. For comparative purposes, the computation time required by Caltech's SD-Toolbox [61] with Cantera [59], which only provides the over-driven branch, was 101.77 seconds, which represents a 10× speed-up factor

Algorithm 5 Pseudocode to solve weak and strong branches of over-driven oblique detonations for a given mixture mix_1, degree of overdrive η, and deflection angle θ. For non-reactive shocks step 1 and 2 are omitted, u_{1n} is calculated rather than η_n, and `det_overdriven` is changed to `shock_incident`($self$, mix_1, u_{1n}, mix_2). Calculations for a given wave angle β only require steps 1-3, 4.1, 4.2, and 5.

function `det_oblique_theta`($self$, mix_1, η, θ)
  Step 1. Obtain CJ pre-shock state
    $mix_{1,\mathrm{cj}} \leftarrow$ `det_cj`($self, mix_1$)
  Step 2. Get pre-shock velocity and sound velocity
    $u_1, a_1 \leftarrow u_{1,\mathrm{cj}}\eta,\ a_{1,\mathrm{cj}}$
  Step 3. Obtain wave angle limits and their initial guesses
    $\beta_{\min}, \beta_{\max} \leftarrow \sin^{-1}(a_1/u_1),\ \pi/2$
    $\beta_{\mathrm{weak}}, \beta_{\mathrm{strong}} \leftarrow (\beta_{\min} + \beta_{\max})/2,\ 0.97\beta_{\max}$
  Step 4. Solve weak/strong branch using Halley's method
  while STOP $> \epsilon_{\mathrm{oblique}}$ & $k < k_{\max,\mathrm{oblique}}$ **do**
    $k \leftarrow k + 1$
    Step 4.1 Get normal component degree-overdrive η_n
      $\eta_n \leftarrow \eta \sin(\beta)$
    Step 4.2 Obtain post-shock state and there fore u_{2n}
      $mix_2 \leftarrow$ `det_overdriven`($self, mix_1, \eta_n$)
    Step.4.3 Compute $f(\beta)$, $f'(\beta)$, and $f''(\beta)$
    Step.4.4 Update estimate β
      $$\beta \leftarrow \beta - \frac{2f(\beta)f'(\beta)}{2f'(\beta)^2 - f(\beta)f''(\beta)} \qquad \triangleright \text{Eq. (20)}$$
    Step.4.5 Calculate STOP criteria
      $$\mathrm{STOP} \leftarrow \max\left(\left|\frac{\beta_{k+1} - \beta_k}{\beta_k}\right|, \left|\frac{f(\beta_{k+1})}{\beta_{k+1}}\right|\right)$$
  end while
  Step 5. Calculate post-shock velocity u_2
    $u_2 \leftarrow u_{2,n} \sin^{-1}(\beta - \theta)$
end function

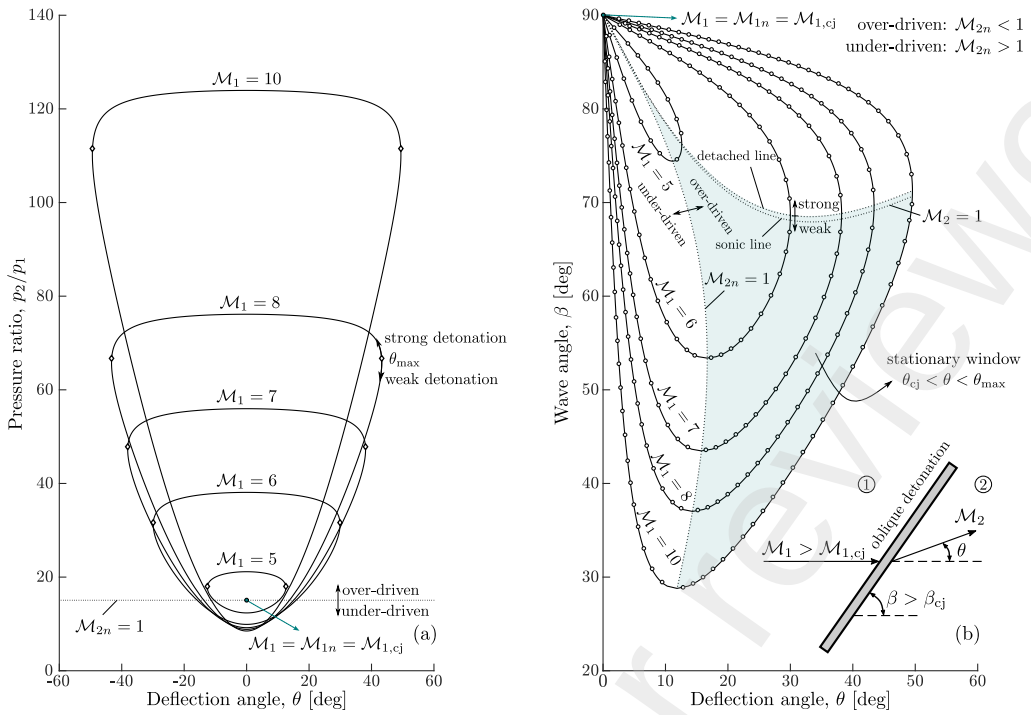


Figure 9: Pressure-deflection (a) and wave angle-deflection (b) detonation polar diagrams for a stoichiometric hydrogen (H_2) air (79% N_2, 21% O_2) mixture at pre-shock temperature $T_1 = 300$ K and pressure $p_1 = 1$ atm, and a range of pre-shock Mach numbers $\mathcal{M}_1$ between 5 and 10; solid line: results with CT considering a calorically imperfect gas with dissociation; circles: results from Zhang et al. [88].

for our code.

The blue-shaded area shown in Fig. 9 corresponds to weak over-driven conditions, which are the most likely to occur in oblique detonations. This region, bounded by $\theta_{cj} < \theta < \theta_{max}$, is of significant interest due to its applicability to the study of ODWE systems [104, 105]. For instance, recent research by Guo et al. [105] investigated the impact of pre-shock conditions on the stationary window for CH_4-air oblique detonations. The authors discovered that the limits θ_{cj} and θ_{max} depend strongly on the pre-shock velocity and heat release associated with the mixture, while variations with the upstream temperature and pressure are not as prominent. Combustion Toolbox allows to perform these calculations easily, highlighting its relevance in carrying out preliminary studies before tackling more complex flow configurations.

5. Rocket module

The calculation of the theoretical performance of rocket engines has drawn renewed attention in recent times, primarily driven by the emergence of private space companies such as Virgin Galactic, SpaceX, Blue Origin, Rocket Lab, and the Spanish PLD Space, which are focused on developing low-cost and reusable launch vehicles [106]. Despite the inherent complexities of these systems associated with the variety of physicochemical phenomena involved, a reasonably accurate estimation of engine performance can be achieved. Thus, since rocket engines typically operate at moderate pressures, the ideal gas assumption can be applied without the need for more complex equations of state. Additionally, the long residence times of the reacting gases in the combustion chamber compared to the chemical reaction times allow further simplification of the calculations. This simplification allows for the utilization of thermochemical equilibrium tools such as CT.

The initial release of CT-ROCKET incorporates the mathematical description proposed in [15]. The approach is based on a number of simplifying assumptions, such as one-dimensional flow, uniform cross-sectional area, negligible flow velocity at the inlet of the combustion chamber, adiabatic combustion, isentropic expansion at the nozzle, homogeneous flow, ideal equation of state, and continuity of temperatures and velocities between gaseous and condensed species. Further details on the numerical implementation can be found in Ref. [15, Chapter 6].

CT-ROCKET utilizes the CT-EQUIL module to determine the gas composition within the rocket engine at various points of interest, such as the injector (inj), the combustion chamber outlet (c), the nozzle throat (t), and different points between (t) and (c/inf) where the hot gases are compressed (subsonic region) or between (t) and the nozzle outlet (e) where the hot gases expand (supersonic region), as illustrated in Fig. 10 (top). The bottom panel shows the temperature and the Mach number of the fluid particles from the combustion chamber outlet (c) to the exit (e), passing through the throat (t) with $A_t = A_c/3$, for a LOX/RP1 mixture with equivalence ratio $\phi = 1.5$ (representing a 2.27 oxidizer/fuel weight ratio) in a high-pressure combustion chamber at $p_1 = 100$ atm. The area ratio is taken as the control variable, upon condition that thermochemical equilibrium is achieved at each position.

Additionally, the module calculates the thrust generated by

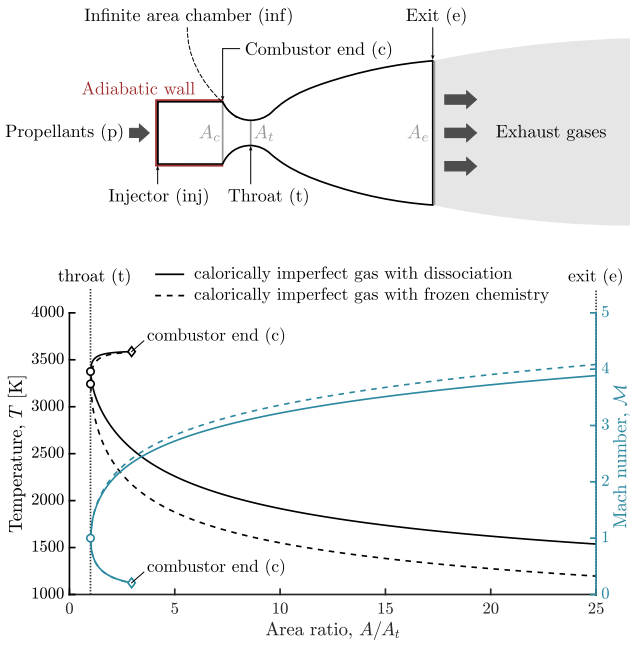


Figure 10: Sketch of the cross section of a finite area chamber (FAC) rocket engine. The dashed line represents the difference with an infinite area chamber (IAC), which is only included for the top region for clarity. Chemical transformations: (p-inj) and (p-inf) instant adiabatic combustion at constant pressure (HP); (inj-c) entropic process; (inf-t), (c-t), and (t-e) isentropic process at defined pressure (SP). Bottom: variation of the temperature (——) and Mach number (——) from the combustor end (c) to the exit (e) for a LOX/RP1 mixture with equivalence ratio $\phi = 1.5$ in a high-pressure combustion chamber $p_1 = 100$ atm considering calorically imperfect gas with dissociation (line) and calorically imperfect gas with frozen chemistry (dashed).

the rocket engine. CT-ROCKET allows for calculations using either frozen chemistry or chemical equilibrium, accounting for combustion chambers with both finite (entropic process) and infinite (isentropic process) dimensions. The frozen chemistry and chemical equilibrium approaches provide an estimate of the performance limits of rocket engine nozzles, as demonstrated by Grossi et al. [107] through two-dimensional numerical simulations based on finite-rate kinetics. This feature enables the performance of parametric analyses to determine the optimal theoretical configuration for a given launch condition or to evaluate the environmental impact at various stages of the rocket vehicle. For instance, with the increasing number of space launches [108, 109], there has been a shift away from highly toxic fuels such as unsymmetrical dimethylhydrazine (UDMH) during the early phases of launch, towards so-called "green" propellants like kerosene (RP1) [110]. It is expected that the use of toxic fuels will be further restricted or even prohibited in the near future, thus driving the need to gain more experience with alternative "green" propellants [111]. The CT-ROCKET module can prove to be a valuable tool in this endeavor.

As previously discussed, this module also includes various routines to compute the state of the mixture at different points of the rocket engine. These states can be modeled using either an infinite area chamber (IAC) or a finite area chamber (FAC). The main steps to calculate the mixture properties using the finite area chamber (FAC) model are defined in the top-layer routine `rocket_performance.m`, as described in Algorithm 6. In brief, an iterative procedure is used to determine the mixture states at the chamber outlet (c) and at the throat (t) by using the infinite area chamber (IAC) model upon defining the initial fresh mixture (composition, temperature, and pressure), A_c/A_t, and A_e/A_t.

Numerous validations were conducted using NASA's CEA code to ensure the reliability and robustness of CT-ROCKET. As an example, Fig. 11 displays a range of thermodynamic properties computed at the nozzle exit (e). The geometrical aspect ratios defining the combustion chamber and the nozzle are $A_c/A_t = 2$ and $A_s/A_t = 3$. Several reacting mixtures were utilized in the computations, including LOX/LH_2, LOX/RP1, LOX/LCH_4, and N_2O_4/MMH, the latter consisting of nitrogen tetroxide and monomethyl-hydrazine, both of which are highly toxic. The reactants were introduced in a combustion chamber where they reacted isobarically under high-pressure conditions, $p_1 = 100$ atm. The inlet temperature of the propellants was set to their respective boiling points, except for N_2O_4, which was evaluated at 300 K. The computations were performed over a wide range of equivalence ratios, $\phi \in [0.5, 4]$, to investigate the impact of the fuel-to-oxidizer ratio on the combustion process and the resulting reaction products.

As illustrated in Fig. 11, CT-ROCKET accurately predicts the properties of primary interest at the nozzle exit, including temperature (T) in (a), pressure (p) in (b), enthalpy (h) in (c), specific heat capacity at constant pressure (c_p) in (d), adiabatic index (γ_s) in (e), gas velocity (u) in (f), specific impulse at sea level (I_{sp}) in (g), and specific impulse in a vacuum (I_{vac}) in (h). The results demonstrate excellent agreement with the CEA code, with uniform convergence. However, the NASA code showed numerical instabilities for certain cases, such as LOX/RP1 at $\phi = 3$ and LOX/LCH4 at $\phi = 4$. The computation time for LOX/H2 was 19.53 seconds for a set of 11 species and a total of 351 cases. The other mixtures were computed using 94 species, with an average computation time of 57.25 seconds. It is noteworthy that the computation time per species is almost three times less for the latter cases.

Algorithm 6 Pseudocode to obtain the theoretical performance of a rocket engine at the points displayed in Fig. 10, for the FAC model, a given mixture mix_1, and aspect ratios A_e/A_t and A_c/A_t. The value of A_c/A_t is included in the *self* variable.

function `rocket_performance`($self$, mix_1, A_e/A_t)
  Step 1. Calculate mixture state at the injector, chamber outlet, and throat
    $mix_{2,\text{inj}}, mix_{2,\text{c}}, mix_3 \leftarrow$ `compute_FAC`($self, mix_1$)
      ▷ includes callbacks to the IAC model
  Step 2. Calculate mixture state at the exit points
    $mix_4 \leftarrow$ `compute_exit`($self, mix_{2,\text{c}}, mix_3, mix_4, ...$
      $A_e/A_t, mix_{2,\text{inj}}$)
  Step 3. Calculate performance parameters
    $mix_3, mix_{2,\text{c}}, mix_4 \leftarrow$ `rocket_parameters`(...
      $mix_{2,\text{inj}}, mix_3, self.C.gravity, mix_{2,\text{c}}, mix_4$)
end function

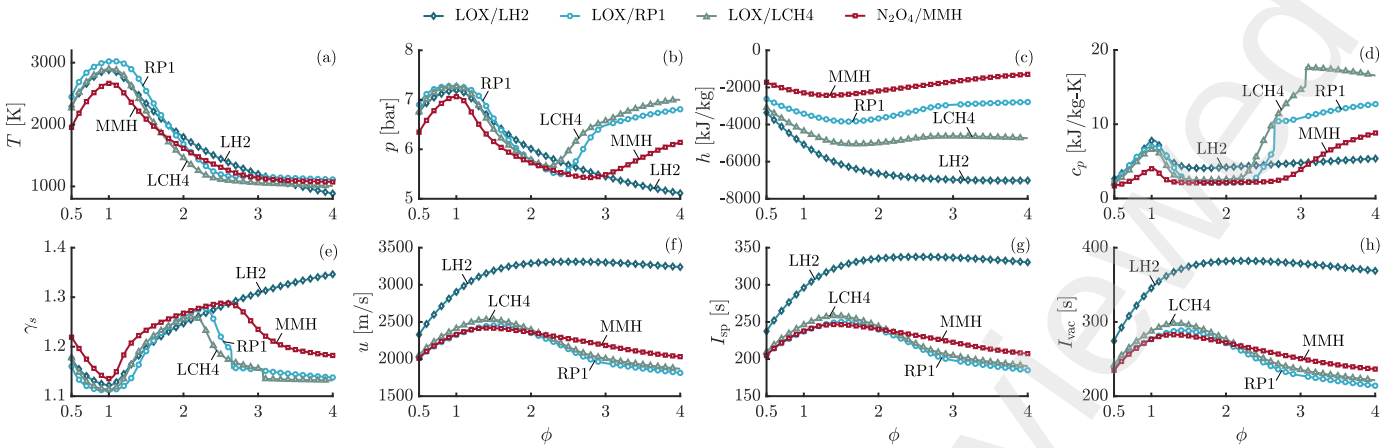


Figure 11: Thermodynamic properties at the nozzle exit of a rocket engine with aspect ratios $A_c/A_t = 2$ and $A_e/A_t = 3$ for different liquid bi-propellant mixtures in a high-pressure combustion chamber, $p_1 = 100$ atm, with equivalence ratios $\phi \in [0.5, 4]$: temperature, T (a), pressure, p (b), enthalpy, h (c), specific heat capacity at constant pressure, c_p (d), adiabatic index, γ_s (e), gas velocity, u (f), specific impulse at sea level, I_{sp} (g), specific impulse in a vacuum, I_{vac} (h); solid line: results obtained with CT; symbols: results obtained with the NASA's CEA [15]: LOX/LH_2 (◇), LOX/RP1 (○), LOX/LCH_4 (△), N_2O_4/MMH (□). The code snippet for the LOX/LH_2 case is shown in Listing 4.

Description	Module	GUI	Main callbacks
CE at defined TP, TV	CT-EQUIL	✓	`equilibrate_T`
CE at defined HP, SP, EV, SV	CT-EQUIL	✓	`equilibrate`
CE for a given TP profile - Exoplanets	CT-EQUIL		`equilibrate_T`
Normal shocks	CT-SD	✓	`shock_incident`
Reflected normal shocks	CT-SD	✓	`shock_reflected`
Oblique shocks for a given β	CT-SD	✓	`shock_oblique_beta`
Oblique shocks for a given θ	CT-SD	✓	`shock_oblique_theta`
Shock polar diagrams	CT-SD	✓	`shock_polar`
Reflected oblique shocks for a given θ	CT-SD		`shock_oblique_theta`, `shock_oblique_beta`
Reflected shock polar diagrams for a given β	CT-SD	✓	`shock_polar`, `shock_oblique_beta`
Reflected shock polar diagrams for a given θ	CT-SD	✓	`shock_polar`, `shock_oblique_theta`
Shocks in the limit of regular reflections	CT-SD		`shock_polar_limitRR`
CJ detonations	CT-SD	✓	`det_cj`
Over-driven detonations	CT-SD	✓	`det_overdriven`
Under-driven detonations	CT-SD	✓	`det_underdriven`
Reflected CJ detonations	CT-SD	✓	`det_cj`, `shock_reflected`
Reflected over-driven detonations	CT-SD	✓	`det_overdriven`, `shock_reflected`
Reflected under-driven detonations	CT-SD	✓	`det_underdriven`, `shock_reflected`
Oblique detonation for a given β	CT-SD	✓	`det_oblique_beta`
Oblique detonation for a given θ	CT-SD	✓	`det_oblique_theta`
Detonation polar diagrams	CT-SD	✓	`det_polar`
Rocket engine performance assuming IAC	CT-ROCKET	✓	`rocket_performance`
Rocket engine performance assuming FAC	CT-ROCKET	✓	`rocket_performance`

Table 2: Summary of problems that can be solved using the Graphic User Interface.

6. Graphic User Interface

This section presents a detailed overview of the Graphic User Interface (GUI) developed in this study. The GUI is intended to provide a user-friendly and intuitive interface for the visualization and analysis of data. Most of the functions presented in the manuscript's code, listed in Table 2, are encapsulated in the GUI. The remainder will be included in the next update of the package. Figures 12 - 14 depict the fundamental elements of the GUI: the menu bar, the tabs and sub-tabs, the control panel, the command window, the dialog box, the lamp, the tree, and the data visualization area. Each of these objects is described in detail below.

- The *menu bar* comprises predefined actions such as clear, save, snapshot, and check for updates, along with options to access online documentation, tutorials, examples, and license. It also provides access to additional tools, or add-ons, that can expand the GUI's capabilities. These add-ons include controls for numerical errors and visualization settings, species selection, as well as the ability to perform code validations and provide feedback to the development team.
- The interface is divided into two main *tabs* (setup and results) and additional *sub-tabs* designed to organize the content and prevent the user from feeling overwhelmed. The setup tab contains a control panel to configure the problem to be addressed. The results tab contains a data visualization area for post-processing all collected data.
- The *control panel* is a crucial part of the GUI that enables users to configure the problem conditions. It provides a range of controls and options to adjust parameters such as the chemical species (reactants and products), the initial state (composition, temperature, and pressure), the type of problem to be solved, and other parameters for the setup of single-case or parametric studies.
- The *command window* provides a command-line interface that allows users to interact with the GUI through a series of text commands. This feature is particularly useful for advanced users who prefer to work with code or scripts. Through this tool, users can input commands and execute scripts, while the *dialog box* prompts the user for further input or confirmation before executing a command. The dialog box displays practical information like warnings, errors, and execution time, providing valuable feedback to the user.
- The *lamp* component serves as a visual indicator of the analysis status. When the analysis is complete, the lamp emits a green light, while a yellow light indicates that the computations are still in progress. A red light indicates an error in the analysis.
- The *data visualization area* displays the computed results in a visual format, such as plots or tables. It allows users to interact with and explore the data in a way that's intuitive and easy to understand.
- The *tree* component collects all the data and exhibits the hierarchical organization of the obtained outcomes, which enables users to explore and access distinct aspects of each case.
- Additional features have been incorporated to improve the GUI's usability, including context menus and keyboard shortcuts. These functionalities enable users to perform intricate tasks with ease and speed.

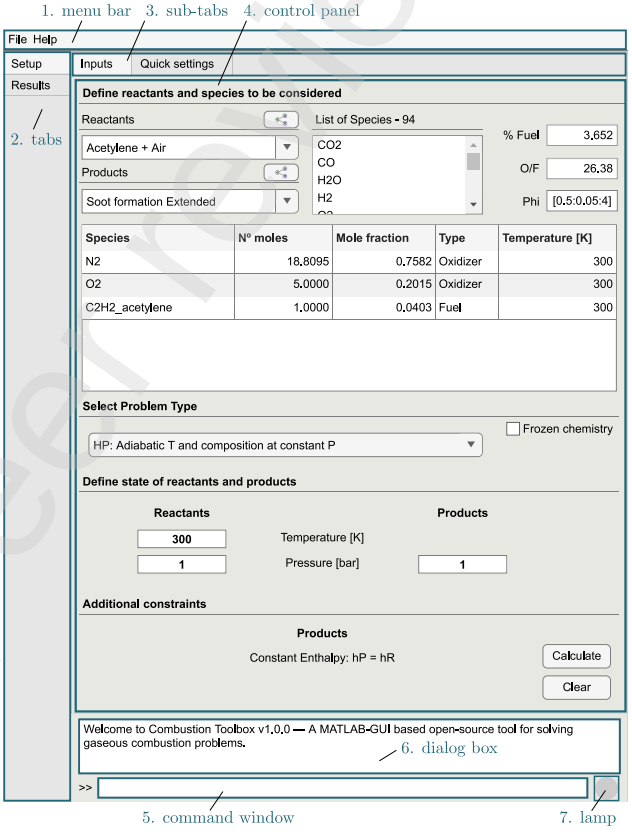
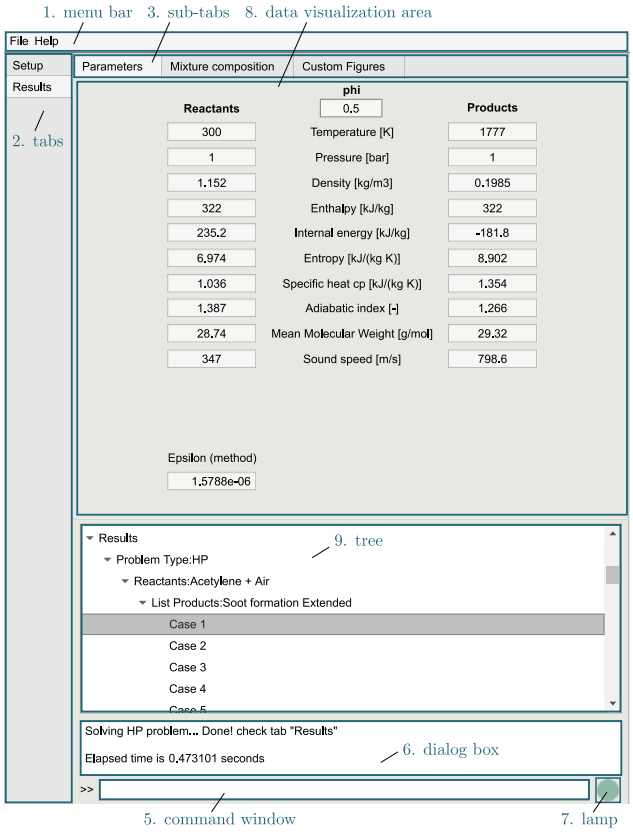
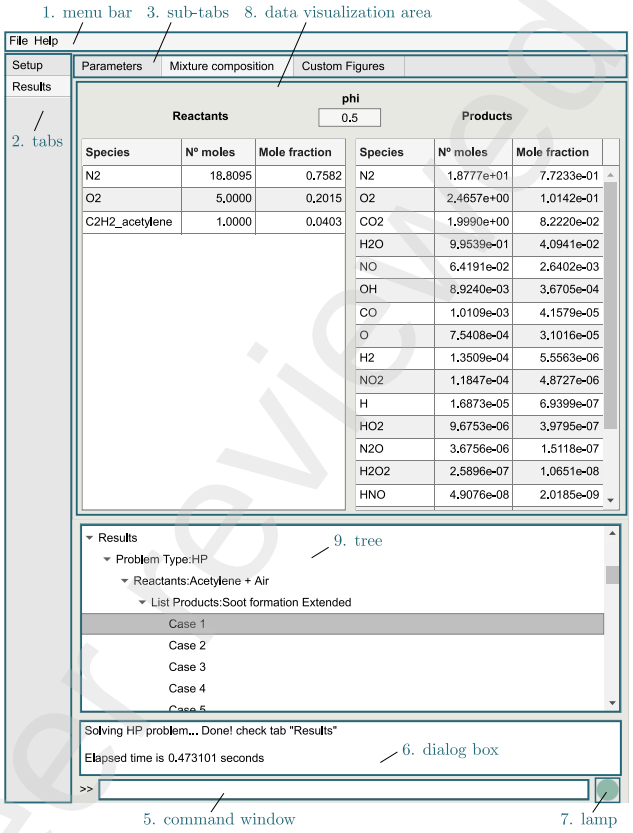


Figure 12: Example of how to configure the GUI to reproduce the results of Fig. 4.

For illustrative purposes, Figs. 12-14 exhibit Combustion Toolbox GUI screenshots captured during a parametric study of the adiabatic and isobaric combustion of acetylene-air mixtures for a wide range of equivalence ratios. This case corresponds to the data displayed in Fig. 4 in Section 3.4. As observed in Fig. 12, the first step includes setting up the problem conditions, which include *i)* the initial mixture (composition, temperature, and pressure), that can be chosen from the predefined mixtures via the *reactants* drop-down menu, or by manually adding the appropriate species name one-by-one to the same object; *ii)* the problem configuration (adiabatic at constant pressure, etc.); *iii)* the control parameter for a parametric or individual study, such as the equivalence ratio; and *iv)* additional input parameters that may be required based on the type of problem.

If a parametric study is selected with the equivalence ratio as the control parameter, the post-processing step can be initiated (as depicted in Fig. 13). If the problem is well-posed,





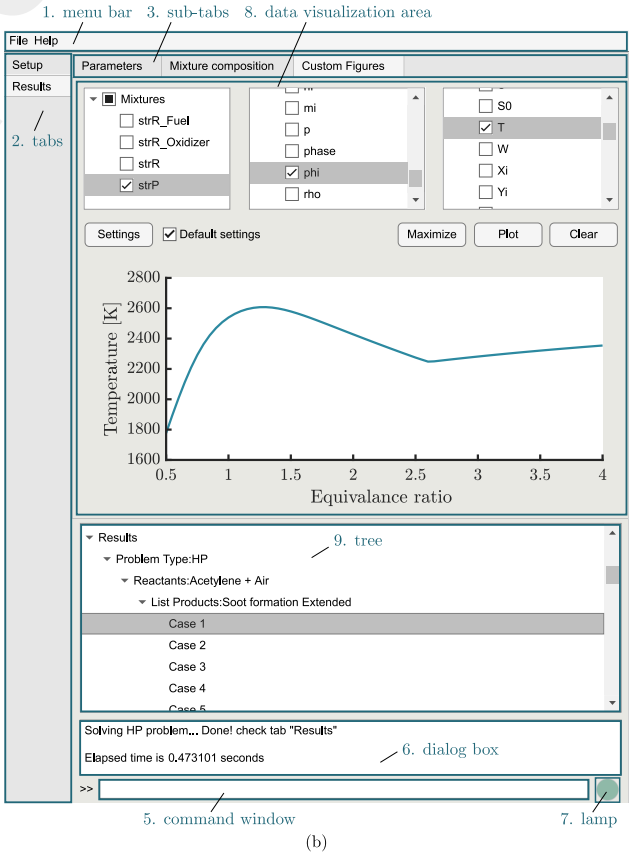
(a)

Figure 13: Post-processing the results of Fig. 4 through the GUI. In particular, the thermodynamic properties correspond to the case selected in the tree object ($\phi = 0.5$).

the *lamp* object will turn green, meaning that the computations were obtained successfully, and a message will appears in the *dialog box* of the GUI. Subsequently, the obtained dataset is re-structured in the background to fit into the *tree* object for result post-processing. By selecting each solution of the *tree* object, the GUI automatically updates the thermodynamic properties of the mixtures in the data visualization area (see Figs. 12 and 13). Additionally, in the *results* ↪ *custom figures* tab, all mixture properties can be analyzed by plotting the results, as illustrated in Fig. 14. The datasets collected using the GUI can also be exported to a structured spreadsheet or *.mat* file.

As commented before, CT has implemented several add-ons to enhance the capabilities of the main GUI. For example, the add-on *uielements* (see Fig. 15) facilitates the selection of the chemical species allocated in the databases. It also allows to evaluate and plot the thermodynamic data of the individual species. The CT settings can be conveniently adjusted using the *uipreferences* add-on, as illustrated in Fig. 16. Of particular interest for potential users, the validations conducted in CT can be replicated using the *uivalidations* add-on, which reads all the tests stored in the *validations* folder. Lastly, all users can report bugs, ideas, or queries using the *uifeedback* add-on, which will include predefined templates to maintain consistent standards.

In brief, the GUI offers a broad range of tools for examining problems related to chemical equilibrium. The user-friendly design and intuitive features make it accessible to a variety of



(b)

Figure 14: Post-processing the results of Fig. 4 through the GUI: (a) chemical composition and (b) custom plots. The tab *results* ↪ *mixture composition* shows the mixture composition for the case selected in the tree object ($\phi = 0.5$).

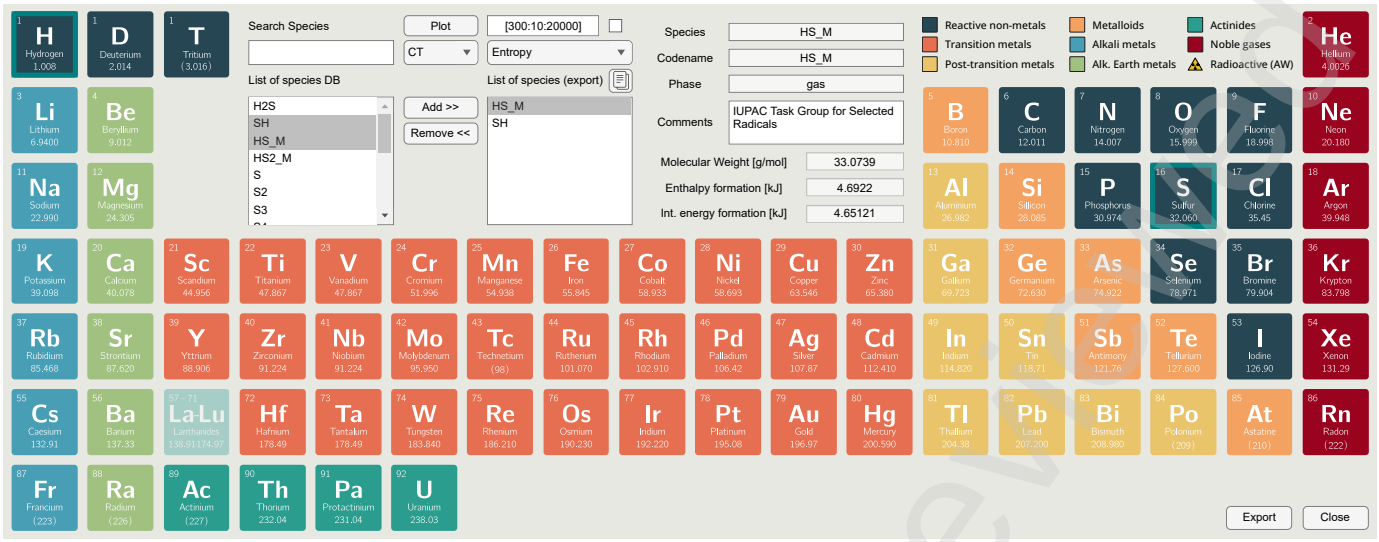


Figure 15: Periodic table add-on used to browse and select the species from the database that contain a particular set of elements. The suffix _M in the name of a given species from the drop-down list indicates that its thermochemical properties are obtained from the Third Millennium database [32].

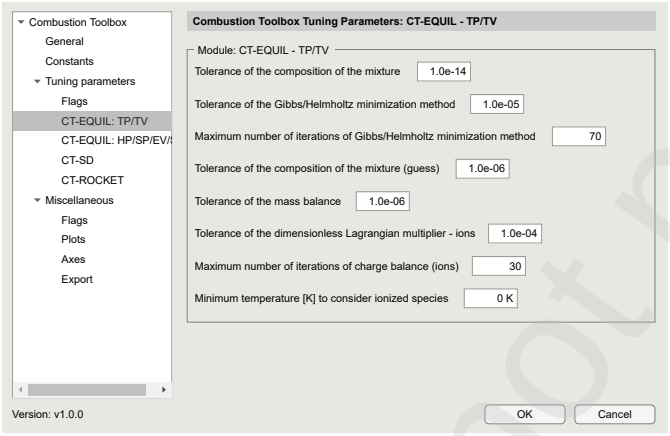


Figure 16: Add-on to set all the preferences of the Combustion Toolbox.

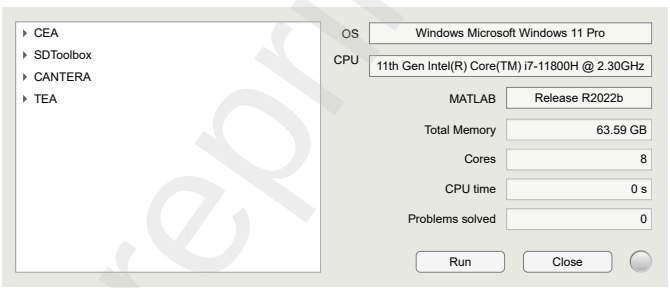


Figure 17: Add-on to reproduce all the validations of the Combustion Toolbox.

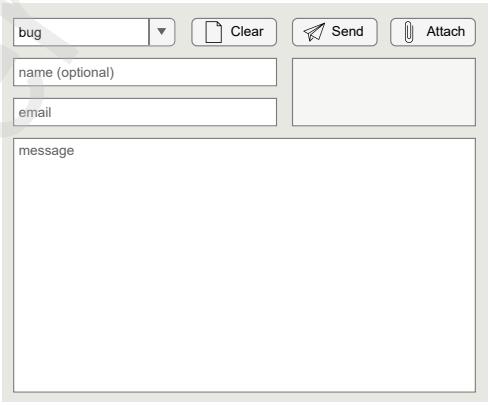


Figure 18: Add-on to report bug/inquiries of the Combustion Toolbox.

users, including those with a limited technical background. The GUI is an essential tool for researchers and practitioners who need to perform and analyse extensive parametric studies of the wide range of problems that can be addressed by the code. However, it is important to note that the GUI is intended to supplement traditional coding approaches instead of replacing them. Despite its ability to streamline tasks for non-experts, plain code actually exhibits greater versatility. The GUI itself is constructed upon an existing codebase, whose fundamental functions and calculations are still available for access and manipulation via the command line interface. In fact, proficient users interested in intricate analytical requirements may find that direct coding and execution is a more efficient and effective method than relying exclusively on the GUI. Thus, the GUI ought to be regarded as a tool that assists users with specific tasks rather than as a replacement for the potent and flexible nature of traditional coding.

7. Conclusions

In this work, we present the Combustion Toolbox (CT), an innovative open-source thermochemical code developed for the calculation of equilibrium states in gaseous reacting systems. While CT primarily focuses on combustion problems that may involve the formation of condensed-phase species, its abilities extend to other areas of interest, including the calculation of the atmospheric compositions of gaseous exoplanets, ablation processes, hypersonic shocks, and detonations. CT has been implemented in MATLAB and designed with a modular architecture, making it both user- and developer-friendly. Additionally, CT is equipped with an advanced Graphic User Interface that encapsulates the three modules and multiple built-in functions, providing users with a convenient operating experience.

At present, the three modules included in CT are CT-EQUIL, CT-SD, and CT-ROCKET. The first module, CT-EQUIL, is the kernel of the Combustion Toolbox and is responsible for solving the chemical composition of the system at equilibrium. This is achieved by minimizing the Gibbs/Helmholtz free using the Lagrange multiplier approach coupled with a multidimensional Newton-Raphson method. The second module, CT-SD, solves post-shock/detonation states for normal and oblique incident flows, including the computation of reflected waves. The third module, CT-ROCKET, is designed to determine the mixture composition at various points of interest within rocket engines, along with the calculation of the theoretical rocket performance.

The modules have been validated against existing state-of-the-art codes, including NASA's Chemical Equilibrium with Applications (CEA) [15], Cantera [59] within Caltech's Shock and Detonation Toolbox (SD-Toolbox) [60, 61], and the newly developed Thermochemical Equilibrium Abundances (TEA) code [23]. All tests showed excellent agreement. As a matter of fact, CT exhibits superior computational performance in terms of cost and time, outperforming Caltech's SD-Toolbox and TEA by a significant margin. Additional validations can be accessed through the web at https://combustion-toolbox-website.readthedocs.io, which also provides further documentation and examples. The tool is actively maintained and can be accessed at https://github.com/AlbertoCuadra/combustion_toolbox.

While Combustion Toolbox has obtained promising results, it is still an ongoing research project that requires additional development to enhance its capabilities. We aim to introduce additional functionalities in future versions of the code, such as the incorporation of non-ideal equations of state (currently under implementation), the analysis of multi-phase systems, a more accurate model for rocket propellant performance, and the extension of the database to include transport properties. We are also considering expanding the code to other well-known open-source programming languages, such as C++ and Python. The former is preferred due to its exceptional performance compared to MATLAB [62], while the latter is preferred due to its simplicity [112]. Additionally, a functional version of the CT-EQUIL module has been developed in Python. An intermediate step will involve using MEX functions in the kernel of the code to combine C++ and MATLAB for calculating chemical equilibrium at defined temperature and pressure/volume, which is anticipated to substantially improve the speed of the code.

Acknowledgements

The authors acknowledge funding from Comunidad de Madrid under the Multiannual Agreement H2SFE-CM-UC3M. C.H. also received support from project TED2021-129446B-C41 (MICINN/FEDER, UE). M.V. would like to express his gratitude for the invaluable and inspiring collaborations with Professors Amable Liñán and Antonio L. Sánchez over the past twenty-five years. Their contributions and insights played a pivotal role in initiating and promoting this project.

Declaration of Competing Interest

The authors declare that they have no competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Thermodynamic data

Appendix A.1. Polynomials

The equilibrium computations carried out by CT rely on NASA's 9-coefficient polynomial fits, which provide the thermodynamic data of the individual species, including the molar specific heat at constant pressure, enthalpy, and entropy as a function of temperature in dimensionless form

$$\frac{C_p^\circ}{R} = a_1 T^{-2} + a_2 T^{-1} + a_3 + a_4 T + a_5 T^2 + a_6 T^3 + a_7 T^4, \tag{A.1a}$$

$$\frac{H^\circ}{RT} = -a_1 T^{-2} + a_2 T^{-1} \ln T + a_3 + a_4 T/2 + a_5 T^2/3 + a_6 T^3/4 + a_7 T^4/5 + a_8/T, \tag{A.1b}$$

$$\frac{S^\circ}{R} = -a_1 T^{-2}/2 - a_2 T^{-1} + a_3 \ln T + a_4 T + a_5 T^2/2 + a_6 T^3/3 + a_7 T^4/4 + a_9, \tag{A.1c}$$

where the a_i are the specific heat temperature coefficients for $i = 1, \ldots, 7$ and the two integration constants required for the computation of enthalpy and entropy for $i = 8, 9$. Depending on the species, the fits range from 50 K to 20000 K and are typically divided into two or three temperature ranges. The implemented thermodynamic database is a combination of NASA's [31] and Burcat's (Third Millennium) [32] databases.

To compute the dimensionless Gibbs free energy G°/RT from NASA's polynomials, we use the following expression

$$G^\circ/RT = H^\circ/RT - S^\circ/R, \tag{A.2}$$

or equivalently

$$\frac{G^\circ}{RT} = -a_1 T^{-2}/2 + a_2 T^{-1}(1 + \ln T) + a_3(1 - \ln T) - a_4 T/2 - a_5 T^2/6 - a_6 T^3/12 - a_7 T^4/20 + a_8/T - a_9.$$

This data is collected from the `thermo_CT.inp` file and next formatted into a more accessible structure (*DB_master*) with the built-in function `generate_DB_master.m`. Then, for faster data access, we generate a new database (*DB*) that contains *griddedInterpolant* objects that allows the evaluation of the thermodynamic functions at a given temperature. CT implements routines that facilitate the evaluation of these functions. Thus, obtaining the value of the previous functions for methane CH_4 at 3000 K is as simple as follows:

```
cp0 = species_cp('CH4', 3000, self.DB)
h0  = species_h0('CH4', 3000, self.DB)
s0  = species_s0('CH4', 3000, self.DB)
g0  = species_g0('CH4', 3000, self.DB)
```

returning the values in J/(mol-K) for c_p°, in kJ/mol for h° and g°, and in kJ/(mol-K) for s°.

Appendix A.2. Thermodynamic mixture properties

The thermodynamic properties of a mixture at equilibrium require the evaluation of the contributions of the individual species contained in the mixture, as described in the previous subsection. Due to the mixing process, these properties change when solids, liquids, or gases are considered. Therefore, CT computes them considering a linear mixing rule. In this case, the enthalpy, internal energy, entropy, and Gibbs energy of the mixture read, respectively:

$$h = \sum_{j\in\mathbf{S}} n_j H_j^\circ, \tag{A.3a}$$

$$e = h - nRT, \tag{A.3b}$$

$$s = \sum_{j\in\mathbf{S}} n_j S_j^\circ - R\sum_{j\in\mathbf{S^G}} n_j \ln\left(\frac{n_j p}{n}\right), \tag{A.3c}$$

$$g = h - Ts, \tag{A.3d}$$

where $n = \sum_{j\in\mathbf{S^G}} n_j$ represents the total number of moles in the gas phase. By differentiating Eq. (A.3a) with respect to n_j and T at constant p, we obtain the specific heat at constant pressure

$$c_p = \sum_{j\in\mathbf{S}} n_j C_{p,j}^\circ + \sum_{j\in\mathbf{S^G}} n_j \frac{H_j^\circ}{T}\left(\frac{\partial \ln n_j}{\partial \ln T}\right)_p + \sum_{j\in\mathbf{S^C}} \frac{H_j^\circ}{T}\left(\frac{\partial n_j}{\partial \ln T}\right)_p, \tag{A.4}$$

defined as the sum of the frozen contribution (first term) and the reaction contribution (the remainder). The partial derivatives $(\partial \ln n_j/\partial T)_p$ and $(\partial n_j/\partial T)_p$ are obtained using the solution of the Jacobian matrix $\mathbf{J}$ obtained at equilibrium (see Ref. [15] for more details.). On the other hand, by differentiating Eq. (A.3b) with respect to n_j and T at constant v, we get the specific heat at constant volume

$$c_v = c_p + nR\left(\frac{\partial \ln v}{\partial \ln T}\right)_p^2 \left(\frac{\partial \ln v}{\partial \ln p}\right)_T^{-1} \tag{A.5}$$

and from the ideal gas EoS we obtain the partial derivatives

$$\left(\frac{\partial \ln v}{\partial \ln T}\right)_p = 1 + \left(\frac{\partial \ln n}{\partial \ln T}\right)_p, \tag{A.6a}$$

$$\left(\frac{\partial \ln v}{\partial \ln p}\right)_T = -1 + \left(\frac{\partial \ln n}{\partial \ln p}\right)_T. \tag{A.6b}$$

Lastly, the speed of sound a is defined as

$$a^2 = \frac{p}{\rho}\left(\frac{\partial \ln p}{\partial \ln \rho}\right)_s, \tag{A.7}$$

with

$$\gamma_s = \left(\frac{\partial \ln p}{\partial \ln \rho}\right)_s = -\gamma\left(\frac{\partial \ln v}{\partial \ln p}\right)_T^{-1}, \tag{A.8}$$

where the subscript s denotes constant entropy and $\gamma = c_p/c_v$ is the specific heat ratio.

Combustion Toolbox calculates all these quantities with the function `compute_properties.m`, but first, it is necessary to fill the property matrix

$$\mathbf{M_0} = \begin{bmatrix} h_{f,1} & e_{f,1} & W_1 & \text{phase}_1 & n_1 & h_1^\circ & c_{p,1}^\circ & s_1^\circ \\ h_{f,2} & e_{f,2} & W_2 & \text{phase}_2 & n_2 & h_2^\circ & c_{p,2}^\circ & s_2^\circ \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{f,\text{NS}} & e_{f,\text{NS}} & W_\text{NS} & \text{phase}_\text{NS} & n_\text{NS} & h_\text{NS}^\circ & c_{p,\text{NS}}^\circ & s_\text{NS}^\circ \end{bmatrix} \tag{A.9}$$

containing the necessary thermodynamic parameters of all the species in the system required to calculate the thermodynamic properties of the mixture. These are the enthalpy of formation h_f, the internal energy of formation e_f, the molecular mass W, and the phase, which is zero or one depending if the species is in gas or condensed phase, respectively, the number of moles n, the enthalpy h°, the specific heat at constant pressure c_p°, and the entropy s°. Note that the first four columns are constants; however, the following columns change with temperature and pressure, i.e., the chemical equilibrium state. These values are filled with the routines `set_species_initialize.m` and `set_species.m`, respectively.

Appendix B. Code examples

In this paper, we have presented the Combustion Toolbox, its capabilities, and the type of problems that can be solved with the first version of the code. Each problem has to be called with a specific abbreviation, as demonstrated in line 11 of Listing 1. For example, for the thermochemical equilibrium module CEQUIL, these are:

- TP: equilibrium at specified temperature and pressure.
- HP: equilibrium at constant enthalpy and pressure.
- SP: equilibrium at constant entropy and pressure.
- TV: equilibrium at specified temperature and volume.
- EV: equilibrium at constant internal energy and volume.
- SV: equilibrium at constant entropy and volume.

As one may expect, constant parameters imply that the value, e.g., enthalpy of the HP problem, is the same for the initial (mix1) and final mixture (mix2). However, there are problems where it is necessary to use specific values. To do this, using the same example, we can proceed as follows:

```
function mix2 = example(mix1, h, p)
    mix1.h = h;
    mix2 = equilibrate(self, mix1, p);
end
```

solving the HP problem for a defined enthalpy in kJ and pressure in bar. By using a function, we can avoid the use of an auxiliary variable to store the actual value of mix1.

```
% Initialization
self = App('Soot formation extended');
% Set initial conditions
self = set_prop(self, 'TR', 300, 'pR', 1);
self = set_prop(self, 'phi', 0.5:0.05:4);
self.PD.S_Fuel = {'C2H2_acetylene'};
self.PD.S_Oxidizer = {'N2', 'O2'};
self.PD.ratio_oxidizers_O2 = [79, 21]/21;
% Additional inputs
self = set_prop(self, 'pP', 1);
% Solve problem
self = solve_problem(self, 'HP');
% Display results (plots)
post_results(self);
```

Listing 1: Snippet of the code necessary to reproduce the case shown in Fig. 12 using the desktop environment.

The CT-SD module for shock and detonation calculations contains multiple possible problems, with these abbreviations:

- SHOCK_I: planar shock wave, incident.
- SHOCK_R: planar shock wave, incident and reflected.
- SHOCK_OBLIQUE : oblique shock wave, incident.
- SHOCK_OBLIQUE_R: oblique shock wave, incident and reflected.
- SHOCK_POLAR: shock polar diagrams.
- SHOCK_POLAR_R: shock polar diagrams, incident and reflected.
- SHOCK_IDEAL_GAS: planar shock wave, incident for a fixed adiabatic index.
- DET: Chapman-Jouguet detonation.
- DET_R: Chapman-Jouguet detonation, incident and reflected.
- DET_OBLIQUE: oblique detonation.
- DET_POLAR: detonation polar diagrams.
- DET_OVERDRIVEN: over-driven detonation.
- DET_OVERDRIVEN_R: over-driven detonation, incident and reflected.
- DET_UNDERDRIVEN: under-driven detonation.
- DET_UNDERDRIVEN_R: under-driven detonation, incident and reflected.

```
% Initialization
self = App('Air_ions');
% Set initial conditions
self = set_prop(self, 'TR', 300);
self = set_prop(self, 'pR', 1.01325);
self.PD.S_Oxidizer = {'N2', 'O2', 'Ar'};
self.PD.ratio_oxidizers_O2 = [78, 21, 1]/21;
% Additional inputs
self = set_prop(self, 'M1', 2:2:14);
% Tuning properties
self.TN.N_points_polar = 300;
% Solve problem
self = solve_problem(self, 'SHOCK_POLAR');
% Display results (plots)
post_results(self);
```

Listing 2: Snippet of the code necessary to reproduce the case shown in Fig. 6 using the desktop environment.

```
------------------------------------------------------------------------------
Problem type: SHOCK_OBLIQUE  | phi =  NaN
------------------------------------------------------------------------------
                |     STATE 1     |     STATE 2-W   |     STATE 2-S
T [K]           |       300.0000  |      1233.3991  |      1525.1611
p [bar]         |         1.0132  |        20.9924  |        28.2923
r [kg/m3]       |         1.1768  |         5.9302  |         6.4635
h [kJ/kg]       |         1.8585  |      1019.1807  |      1371.9119
e [kJ/kg]       |       -84.2432  |       665.1911  |       934.1900
g [kJ/kg]       |     -2057.6172  |     -8236.8991  |    -10334.2938
s [kJ/(kg-K)]   |         6.8649  |         7.5045  |         7.6754
W [g/mol]       |        28.9697  |        28.9699  |        28.9702
(dlV/dlp)T [-]  |                 |        -1.0000  |        -1.0000
(dlV/dlT)p [-]  |                 |         1.0000  |         1.0000
cp [kJ/(kg-K)]  |         1.0044  |         1.1844  |         1.2333
gamma [-]       |         1.4001  |         1.3198  |         1.3032
gamma_s [-]     |         1.4001  |         1.3198  |         1.3032
sound vel [m/s] |       347.2019  |       683.5102  |       755.2816
u [m/s]         |      1736.0094  |       989.2476  |       523.1743
Mach number [-] |         5.0000  |         1.4473  |         0.6927
------------------------------------------------------------------------------
PARAMETERS
min wave  [deg] |                 |        11.5370  |        11.5370
wave angle[deg] |                 |        56.9743  |        75.8636
deflection[deg] |                 |        40.0000  |        40.0000
------------------------------------------------------------------------------
STATE 1              Xi [-]
N2                7.8000e-01
O2                2.1000e-01
Ar                1.0000e-02
MINORS[+48]       0.0000e+00

TOTAL             1.0000e+00
------------------------------------------------------------------------------
STATE 2-W            Xi [-]
N2                7.7987e-01
O2                2.0986e-01
Ar                1.0000e-02
NO                2.4927e-04
NO2               1.6587e-05
N2O               7.8179e-08
O                 4.9856e-09
O3                1.1427e-10
NO3               2.2089e-11
N2O3              5.2354e-13
MINORS[+41]       0.0000e+00

TOTAL             1.0000e+00
------------------------------------------------------------------------------
STATE 2-S            Xi [-]
N2                7.7931e-01
O2                2.0928e-01
Ar                1.0000e-02
NO                1.3710e-03
NO2               3.6374e-05
O                 4.8812e-07
N2O               4.4256e-07
O3                1.9535e-09
NO3               1.2256e-10
N2O3              4.7846e-12
N                 1.8984e-14
MINORS[+40]       0.0000e+00

TOTAL             1.0000e+00
------------------------------------------------------------------------------
```

Listing 3: Results shown in MATLAB's command window after solving an oblique shock for air (78% N_2, 21% O_2, and 1% Ar) at standard conditions (T_1 = 300 K, p_1 = 1 atm), pre-shock Mach number $\mathcal{M}_1 = 5$, and deflection angle $\theta = 40$ deg. The states 1, 2-W, and 2-S denote initial state, weak solution, and strong solution, respectively.

For further details on how to introduce the additional inputs required for each problem, the reader is referred to the *examples* folder or the website. As two illustrative examples, Listing 2 shows the snippet of the code necessary to obtain the shock polar diagrams of Fig. 6, and Listing 3 displays an example of the report prompted in the command window after computing an oblique shock in air, at a given pre-shock Mach number and deflection angle.

The calculations of the IAC and FAC models in the CT-ROCKET module are called with the same abbreviation:

- ROCKET: rocket engine performance under ideal conditions,

and the model is specified by setting the variable `FLAG_IAC` (in `ProblemDescription.m`) to true or false, corresponding with the IAC and FAC models, respectively. This is shown in Listing 4. Note that, in this case, the temperature of the cryogenic reactants has not been specified because CT directly assigns the temperature corresponding to their boiling points. For calculations assuming frozen chemistry (post-combustion), the `FLAG_FROZEN` attribute of the *Problem Description* must be set to true (by default, it is set to false).

```
% Initialization
 self = App('Hydrogen_L');
% Set initial conditions
 self = set_prop(self, 'pR', 101.325);
 self = set_prop(self, 'phi', 0.5:0.05:4)
 self.PD.S_Fuel = {'H2bLb'};
 self.PD.S_Oxidizer = {'O2bLb'};
 self.PD.FLAG_IAC = false;
 self.PD.FLAG_FROZEN = false;
% Additional inputs
 self = set_prop(self, 'Aratio_c', 2);
 self = set_prop(self, 'Aratio', 3);
% Solve problem
 self = solve_problem(self, 'ROCKET');
% Display results (plots)
 post_results(self);
```

Listing 4: Snippet of the code necessary to partially reproduce the case shown in Fig. 11 using the desktop environment.

References

[1] G. P. Smith, GRI-Mech 3.0, http://www.me.berkley.edu/gri_mech.

[2] F. J. Zeleznik, S. Gordon, An analytical investigation of three general methods of calculating chemical-equilibrium compositions, National Aeronautics and Space Administration, 1960.

[3] S. R. Brinkley Jr, Calculation of the equilibrium composition of systems of many constituents, The Journal of Chemical Physics 15 (2) (1947) 107–110. doi:10.1063/1.1746420.

[4] J. M. Paz-García, B. Johannesson, L. M. Ottosen, A. B. Ribeiro, J. M. Rodríguez-Maroto, Computing multi-species chemical equilibrium with an algorithm based on the reaction extents, Computers & Chemical Engineering 58 (2013) 135–143. doi:10.1016/j.compchemeng.2013.06.013.

[5] J. W. Stock, D. Kitzmann, A. B. C. Patzer, E. Sedlmayr, FastChem: A computer program for efficient complex chemical equilibrium calculations in the neutral/ionized gas phase with applications to stellar and planetary atmospheres, Monthly Notices of the Royal Astronomical Society 479 (1) (2018) 865–874. doi:10.1093/mnras/sty1531.

[6] P. Woitke, C. Helling, G. H. Hunter, J. D. Millard, G. E. Turner, M. Worters, J. Blecic, J. W. Stock, Equilibrium chemistry down to 100 K. Impact of silicates and phyllosilicates on the carbon to oxygen ratio, Astronomy & Astrophysics 614 (2018) A1. doi:10.1051/0004-6361/201732193.

[7] J. W. Stock, D. Kitzmann, A. B. C. Patzer, FastChem 2: an improved computer program to determine the gas-phase chemical equilibrium composition for arbitrary element distributions, Monthly Notices of the Royal Astronomical Society 517 (3) (2022) 4070–4080. doi:10.1093/mnras/stac2623.

[8] F. Van Zeggeren, S. H. Storey, The Computation of Chemical Equilibria, Cambridge University Press, Cambridge, 1970.

[9] W. R. Smith, R. W. Missen, Chemical reaction equilibrium analysis, Wiley, 1982.

[10] W. B. White, S. M. Johnson, G. B. Dantzig, Chemical equilibrium in complex mixtures, The Journal of Chemical Physics 28 (5) (1958) 751–755. doi:10.1063/1.1744264.

[11] F. J. Zeleznik, S. Gordon, Calculation of complex chemical equilibria, Industrial & Engineering Chemistry 60 (6) (1968) 27–57. doi:10.1021/ie50702a006.

[12] G. Eriksson, Thermodynamic studies of high temperature equilibria III. SOLGAS, a computer program for calculating the composition and heat condition of an equilibrium mixture., Acta Chemica Scandinavica 25 (1971) 2651–2658.

[13] W. C. Reynolds, The element potential method for chemical equilibrium analysis: implementation in the interactive program STANJAN, Technical Rept., Dept. of Mechanical Engineering, Stanford University, 1986.

[14] M. L. Michelsen, Calculation of multiphase ideal solution chemical equilibrium, Fluid Phase Equilibria 53 (1989) 73–80. doi:10.1016/0378-3812(89)80073-1.

[15] S. Gordon, B. J. McBride, Computer program for calculation of complex chemical equilibrium compositions and applications. Part 1: Analysis, No. NAS 1.61:1311.

[16] P. Voňka, J. Leitner, Calculation of chemical equilibria in heterogeneous multicomponent systems, Calphad 19 (1) (1995) 25–36. doi:10.1016/0364-5916(95)00004-X.

[17] L. Eriksson, CHEPP-A chemical equilibrium program package for Matlab, SAE transactions (2004) 730–741, Available: https://www.jstor.org/stable/44740797.

[18] D. V. Nichita, S. Gomez, E. Luna, Multiphase equilibria calculation by direct minimization of gibbs free energy with a global optimization method, Computers & Chemical Engineering 26 (12) (2002) 1703–1724. doi:10.1016/S0098-1354(02)00144-8.

[19] S. B. Pope, The computation of constrained and unconstrained equilibrium compositions of ideal gas mixtures using Gibbs function continuation, Cornell University Report FDA (2003) 03–02.

[20] S. B. Pope, Gibbs function continuation for the stable computation of chemical equilibrium, Combustion and Flame 139 (3) (2004) 222–226. doi:10.1016/j.combustflame.2004.07.008.

[21] A. Néron, G. Lantagne, B. Marcos, Computation of complex and constrained equilibria by minimization of the gibbs free energy, Chemical engineering science 82 (2012) 260–271. doi:10.1016/j.ces.2012.07.041.

[22] J. B. Scoggins, T. E. Magin, Gibbs function continuation for linearly constrained multiphase equilibria, Combustion and Flame 162 (12) (2015) 4514–4522. doi:10.1016/j.combustflame.2015.08.027.

[23] J. Blecic, J. Harrington, M. O. Bowman, TEA: A code calculating thermochemical equilibrium abundances, The Astrophysical Journal Supplement Series 225 (1) (2016) 4 1–14. doi:10.3847/0067-0049/225/1/4.

[24] J. Gray, J. Chin, T. Hearn, E. Hendricks, T. Lavelle, J. R. Martins, Chemical-Equilibrium Analysis with Adjoint Derivatives for Propulsion Cycle Analysis, Journal of Propulsion and Power 33 (5) (2017) 1041–1052. doi:10.2514/1.B36215.

[25] C. Tsanas, E. H. Stenby, W. Yan, Calculation of multiphase chemical equilibrium by the modified rand method, Industrial & Engineering Chemistry Research 56 (41) (2017) 11983–11995. doi:10.1021/acs.iecr.7b02714.

[26] C. Tsanas, E. H. Stenby, W. Yan, Calculation of simultaneous chemical and phase equilibrium by the method of lagrange multipliers, Chemical Engineering Science 174 (2017) 112–126. doi:10.1016/j.ces.2017.08.033.

[27] J. Coatléven, A. Michel, A successive substitution approach with embedded phase stability for simultaneous chemical and phase equilibrium calculations, Computers & Chemical Engineering (2022) 108041doi:10.1016/j.compchemeng.2022.108041.

[28] A. Cuadra, C. Huete, M. Vera, Combustion Toolbox: A MATLAB-GUI based open-source tool for solving combustion problems, version 1.0.0 (2023). doi:10.5281/zenodo.5554911.

[29] M. W. Chase, NIST-JANAF thermochemical tables 4th edition, Journal of Physical and Chemical Reference Data, Monograph 9 (1998) 1529–1564.

[30] O. V. Dorofeeva, V. P. Novikov, D. B. Neumann, NIST-JANAF thermochemical tables. I. Ten organic molecules related to atmospheric chemistry, Journal of Physical and Chemical Reference Data 30 (2) (2001) 475–513. doi:10.1063/1.1364518.

[31] B. J. McBride, NASA Glenn coefficients for calculating thermodynamic properties of individual species, National Aeronautics and Space Administration, Glenn Research Center, 2002.

[32] A. Burcat, B. Ruscic, Third millenium ideal gas and condensed phase thermochemical database for combustion (with update from active thermochemical tables), Tech. rep., Argonne National Lab. (ANL), Argonne, IL (United States) (2005). doi:10.2172/925269.

[33] B. Ruscic, R. E. Pinzon, G. Von Laszewski, D. Kodeboyina, A. Burcat, D. Leahy, D. Montoy, A. F. Wagner, Active Thermochemical Tables: thermochemistry for the 21st century, in: Journal of Physics: Conference Series, Vol. 16, IOP Publishing, 2005, p. 078. doi:10.1088/1742-6596/16/1/078.

[34] B. Ruscic, D. H. Bross, Chapter 1 - Thermochemistry, in: T. Faravelli, F. Manenti, E. Ranzi (Eds.), Mathematical Modelling of Gas-Phase Complex Reaction Systems: Pyrolysis and Combustion, Vol. 45 of Computer Aided Chemical Engineering, Elsevier, 2019, pp. 3–114. doi:https://doi.org/10.1016/B978-0-444-64087-1.00001-2.

[35] J. B. Scoggins, J. Rabinovitch, B. Barros-Fernandez, A. Martin, J. Lachaud, R. L. Jaffe, N. N. Mansour, G. Blanquart, T. E. Magin, Thermodynamic properties of carbon–phenolic gas mixtures, Aerospace Science and Technology 66 (2017) 177–192. doi:10.1016/j.ast.2017.02.025.

[36] C. F. Goldsmith, G. R. Magoon, W. H. Green, Database of small molecule thermochemistry for combustion, The Journal of Physical Chemistry A 116 (36) (2012) 9033–9057. doi:10.1021/jp303819e.

[37] G. Blanquart, H. Pitsch, Thermochemical properties of polycyclic aromatic hydrocarbons (pah) from g3mp2b3 calculations, The Journal of Physical Chemistry A 111 (28) (2007) 6510–6520. doi:10.1021/jp068579w.

[38] G. Blanquart, P. Pepiot-Desjardins, H. Pitsch, Chemical mechanism for high temperature combustion of engine relevant fuels with emphasis on soot precursors, Combustion and Flame 156 (3) (2009) 588–607. doi:10.1016/j.combustflame.2008.12.007.

[39] K. Narayanaswamy, G. Blanquart, H. Pitsch, A consistent chemical mechanism for oxidation of substituted aromatic species, Combustion and Flame 157 (10) (2010) 1879–1898. doi:10.1016/j.combustflame.2010.07.009.

[40] G. Blanquart, Effects of spin contamination on estimating bond dissociation energies of polycyclic aromatic hydrocarbons, International Journal of Quantum Chemistry 115 (12) (2015) 796–801. doi:10.1002/qua.24904.

[41] J. A. Miller, R. J. Kee, C. K. Westbrook, Chemical kinetics and combustion modeling, Annual Review of Physical Chemistry 41 (1) (1990) 345–387.

[42] N. Kubota, Propellants and explosives: thermochemical aspects of combustion, John Wiley & Sons, 2015.

[43] C. Huete, A. Cuadra, M. Vera, J. Urzay, Thermochemical effects on hypersonic shock waves interacting with weak turbulence, Physics of Fluids 33 (8) (2021) 086111. doi:10.1063/5.0059948.

[44] H. Jiang, J. Liu, S. Luo, W. Huang, J. Wang, M. Liu, Thermochemical non-equilibrium effects on hypersonic shock wave/turbulent boundary-layer interaction, Acta Astronautica 192 (2022) 1–14. doi:10.1016/j.actaastro.2021.12.010.

[45] G. V. Candler, R. W. MacCormack, Computation of weakly ionized hypersonic flows in thermochemical nonequilibrium, Journal of Thermophysics and Heat Transfer 5 (3) (1991) 266–273. doi:10.2514/3.260.

[46] P. A. Libby, F. A. Williams (Eds.), Turbulent Reacting Flows, Academic Press, New York, 1994.

[47] P. Wolański, Detonative propulsion, Proceedings of the Combustion Institute 34 (1) (2013) 125–158. doi:10.1016/j.proci.2012.10.005.

[48] M. Di Renzo, J. Urzay, Direct numerical simulation of a hypersonic transitional boundary layer at suborbital enthalpies, Journal of Fluid Mechanics 912 (2021) A29 1–36. doi:10.1017/jfm.2020.1144.

[49] V. Raman, S. Prakash, M. Gamba, Nonidealities in rotating detonation engines, Annual Review of Fluid Mechanics 55. doi:10.1146/annurev-fluid-120720-032612.

[50] A. Cuadra, C. Huete, M. Vera, Effect of equivalence ratio fluctuations on planar detonation discontinuities, Journal of Fluid Mechanics 903 (2020) A30 1–39. doi:10.1017/jfm.2020.651.

[51] A. Cuadra, M. Vera, M. Di Renzo, C. Huete, Linear theory of hypersonic shocks interacting with turbulence in air, in: AIAA SciTech 2023 Forum, AIAA paper 2023–0075, 2023. doi:10.2514/6.2023-0075.

[52] M. Michelsen, Calculation of multiphase equilibrium, Computers & chemical engineering 18 (7) (1994) 545–550. doi:10.1016/0098-1354(93)E0017-4.

[53] J. B. Scoggins, V. Leroy, G. Bellas-Chatzigeorgis, B. Dias, T. E. Magin, Mutation++: Multicomponent thermodynamic and transport properties for ionized gases in C++, SoftwareX 12 (2020) 100575. doi:10.1016/j.softx.2020.100575.

[54] A. M. Leal, D. A. Kulik, W. R. Smith, M. O. Saar, An overview of computational methods for chemical equilibrium and kinetic calculations for geochemical and reactive transport modeling, Pure and Applied Chemistry 89 (5) (2017) 597–643. doi:10.1515/pac-2016-1107.

[55] P. J. Groenen, W. J. Heiser, The tunneling method for global optimization in multidimensional scaling, Psychometrika 61 (3) (1996) 529–550. doi:10.1007/BF02294553.

[56] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Gptimization 11 (4) (1997) 341–359. doi:10.1023/A:1008202821328.

[57] K. V. Price, Differential evolution, in: Handbook of optimization, Springer, 2013, pp. 187–214.

[58] J. Sánchez-Monreal, A. Cuadra, C. Huete, M. Vera, SimEx: A Tool for the Rapid Evaluation of the Effects of Explosions, Applied Sciences 12 (18). doi:10.3390/app12189101.

[59] D. G. Goodwin, R. L. Speth, H. K. Moffat, B. W. Weber, Cantera: An object-oriented software toolkit for chemical kinetics, thermodynamics, and transport processes, https://www.cantera.org, version 2.5.1 (2021). doi:10.5281/zenodo.4527812.

[60] S. Browne, J. Ziegler, N. Bitter, B. Schmidt, J. Lawson, J. E. Shepherd, SDToolbox - Numerical Tools for Shock and Detonation Wave Modeling, GALCIT Technical Report FM2018.001 Revised January 2021, California Institute of Technology, Pasadena, CA, https://shepherd.caltech.edu/EDL/PublicResources/sdt.

[61] S. Browne, J. Ziegler, J. E. Shepherd, Numerical solution methods for shock and detonation jump conditions, GALCIT report FM2006 6 (2008) 1–90.

[62] T. Andrews, Computation time comparison between Matlab and C++ using launch windows, Tech. rep., California Polytechnic State University San Luis Obispo, Available: https://digitalcommons.calpoly.edu/aerosp/78 (2012).

[63] F. N. Fritsch, R. E. Carlson, Monotone piecewise cubic interpolation, SIAM Journal on Numerical Analysis 17 (2) (1980) 238–246. doi:10.1137/0717021.

[64] K. Ram, Git can facilitate greater reproducibility and increased transparency in science, Source code for biology and medicine 8 (1) (2013) 1–8. doi:10.1186/1751-0473-8-7.

[65] S. Chacon, B. Straub, Pro git, Apress, 2014.

[66] J. D. Blischak, E. R. Davenport, G. Wilson, A quick introduction to version control with git and github, PLOS computational biology 12 (1) (2016) e1004668. doi:10.1371/journal.pcbi.1004668.

[67] Y. Perez-Riverol, L. Gatto, R. Wang, T. Sachsenberg, J. Uszkoreit, F. da Veiga Leprevost, C. Fufezan, T. Ternent, S. J. Eglen, D. S. Katz, et al., Ten simple rules for taking advantage of git and github (2016). doi:10.1371/journal.pcbi.1004947.

[68] G. Brandl, Sphinx documentation, Available: http://sphinx-doc.org/sphinx.pdf.

[69] J. Cederberg, M. Mikofski, jcewidex, D. Rosset, I. Lenton, N. Martin, M. Topper, C. Markiewicz, C. Boeddeker, C. Vergari, cleobis, D. Stansby, florianjacob, PekaryGergelyR, H. Leblanc, M.-H. Bleu-Laine, N. N. Oosterhof, R. Zimmerman, ptitrex, sphinx-contrib/matlabdomain: 0.18.0 (Apr. 2023). doi:10.5281/zenodo.7829592.

[70] D. P. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods, Academic press, New York, 2014.

[71] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, et al., LAPACK users' guide, SIAM, 1999.

[72] J. Camberos, J. Moubry, Chemical equilibrium analysis with the method of element potentials, in: 39th Aerospace Sciences Meeting and Exhibit, AIAA paper 2001–873, 2001. doi:10.2514/6.2001-873.

[73] B. J. McBride, Computer program for Calculation of Complex Chemical Equilibrium Compositions and Applications, Vol. 2, NASA Lewis Research Center, 1996.

[74] B. M. Smirnov, Fundamentals of ionized gases: basic topics in plasma physics, John Wiley & Sons, 2012.

[75] J. Sharma, A composite third order Newton–Steffensen method for solving nonlinear equations, Applied Mathematics and Computation 169 (1) (2005) 242–246. doi:10.1016/j.amc.2004.10.040.

[76] B. Fegley Jr, K. Lodders, Atmospheric chemistry of the brown dwarf Gliese 229B: Thermochemical equilibrium predictions, The Astrophysical Journal 472 (1) (1996) L37. doi:10.1086/310356.

[77] C. Visscher, K. Lodders, B. Fegley Jr, Atmospheric chemistry in giant planets, brown dwarfs, and low-mass dwarf stars. II. Sulfur and phosphorus, The Astrophysical Journal 648 (2) (2006) 1181. doi:10.1086/506245.

[78] M. Agúndez, V. Parmentier, O. Venot, F. Hersant, F. Selsis, Pseudo 2D chemical model of hot-Jupiter atmospheres: application to HD 209458b and HD 189733b, Astronomy & Astrophysics 564 (2014) A73. doi:10.1051/0004-6361/201322895.

[79] V. Parmentier, M. R. Line, J. L. Bean, M. Mansfield, L. Kreidberg, R. Lupu, C. Visscher, J.-M. Désert, J. J. Fortney, M. Deleuil, et al., From thermal dissociation to condensation in the atmospheres of ultra hot Jupiters: WASP-121b in context, Astronomy & Astrophysics 617 (2018) A110. doi:10.1051/0004-6361/201833059.

[80] N. Madhusudhan, Exoplanetary atmospheres: Key insights, challenges, and prospects, Annual Review of Astronomy and Astrophysics 57 (2019) 617–663. doi:10.1146/annurev-astro-081817-051846.

[81] K. B. Stevenson, J.-M. Désert, M. R. Line, J. L. Bean, J. J. Fortney, A. P. Showman, T. Kataria, L. Kreidberg, P. R. McCullough, G. W. Henry, et al., Thermal structure of an exoplanet atmosphere from phase-resolved emission spectroscopy, Science 346 (6211) (2014) 838–841. doi:10.1126/science.1256758.

[82] M. Asplund, N. Grevesse, A. J. Sauval, P. Scott, The chemical composition of the sun, Annual Review of Astronomy and Astrophysics 47 (2009) 481–522. doi:10.1146/annurev.astro.46.060407.145222.

[83] B. Helber, C. O. Asma, Y. Babou, A. Hubin, O. Chazot, T. E. Magin, Material response characterization of a low-density carbon composite ablator in high-enthalpy plasma flows, Journal of materials science 49 (2014) 4530–4543. doi:10.1007/s10853-014-8153-z.

[84] F. Bariselli, A. Frezzotti, A. Hubin, T. E. Magin, Aerothermodynamic modelling of meteor entry flows, Monthly Notices of the Royal Astronomical Society 492 (2) (2020) 2308–2325. doi:10.1093/mnras/stz3559.

[85] S.-H. Park, J. N. Laboulais, P. Leyland, S. Mischler, Re-entry survival analysis and ground risk assessment of space debris considering by-products generation, Acta Astronautica 179 (2021) 604–618. doi:10.1016/j.actaastro.2020.09.034.

[86] T. Scavo, J. Thoo, On the geometry of halley's method, The American mathematical monthly 102 (5) (1995) 417–426. doi:doi.org/10.1080/00029890.1995.12004594.

[87] H. Hornung, Regular and Mach reflection of shock waves, Annual review of fluid mechanics 18 (1986) 33–58. doi:10.1146/annurev.fl.18.010186.000341.

[88] Z. Zhang, C. Wen, W. Zhang, Y. Liu, Z. Jiang, A theoretical method for solving shock relations coupled with chemical equilibrium and its applications, Chinese Journal of Aeronautics 35 (6) (2022) 47–62. doi:10.1016/j.cja.2021.08.021.

[89] J. E. Dennis Jr, R. B. Schnabel, Numerical methods for unconstrained optimization and nonlinear equations, SIAM, 1996.

[90] M. Cavcar, The international standard atmosphere (ISA), Anadolu University, Turkey 30 (9) (2000) 1–6.

[91] Norma UNE 31-002-94: Cálculo de las principales características teóricas de los explosivos, Asociación Española de Normalización UNE, 1994.

[92] European Standard EN 13631-15: Explosives for Civil use – High Explosives – Part 15: Calculation of the Thermodynamic Properties, 2005.

[93] C. L. Mader, Detonation Properties of Condensed Explosives Computed Using the Becker-Kistiakowsky-Wilson Equation of State. Report LA-2900, Los Alamos Scientific Laboratory, Los Alamos, NM, USA, 1963.

[94] M. L. Hobbs, M. R. Baer, Nonideal thermoequilibrium calculations using a large product species data base, Shock Waves 2 (3) (1992) 177–187. doi:10.1007/BF01414640.

[95] O. Heuzé, Equations of state of detonation products: Influence of the repulsive intermolecular potential, Phys. Rev. A 34 (1) (1986) 428–432. doi:10.1103/PhysRevA.34.428.

[96] R. Zipf Jr, V. Gamezo, M. Sapko, W. Marchewka, K. Mohamed, E. Oran, D. Kessler, E. Weiss, J. Addis, F. Karnack, et al., Methane–air detonation experiments at NIOSH Lake Lynn Laboratory, Journal of Loss Prevention in the Process Industries 26 (2) (2013) 295–301. doi:10.1016/j.jlp.2011.05.003.

[97] S. Zhou, H. Ma, S. Chen, Y. Zhong, C. Zhou, Experimental investigation on propagation characteristics of rotating detonation wave with a hydrogen-ethylene-acetylene fuel, Acta Astronautica 157 (2019) 310–320. doi:10.1016/j.actaastro.2019.01.009.

[98] W. Strauss, S. JN, Experimental investigation of the detonation properties of hydrogen-oxygen and hydrogen-nitric oxide mixtures at initial pressures up to 40 atmospheres, Combustion and Flame 19 (1972) 141–143.

[99] T. Mogi, S. Horiguchi, Explosion and detonation characteristics of dimethyl ether, Journal of hazardous materials 164 (1) (2009) 114–119. doi:10.1016/j.jhazmat.2008.07.133.

[100] C. Li, K. Kailasanath, E. S. Oran, Detonation structures behind oblique shocks, Physics of Fluids 6 (4) (1994) 1600–1611.

[101] K. Kailasanath, Review of propulsion applications of detonation waves, AIAA journal 38 (9) (2000) 1698–1708.

[102] J. M. Powers, K. A. Gonthier, Reaction zone structure for strong, weak overdriven, and weak underdriven oblique detonations, Physics of Fluids A: Fluid Dynamics 4 (9) (1992) 2082–2089.

[103] Z. Zhang, Y. Liu, C. Wen, Mechanisms of the destabilized Mach reflection of inviscid oblique detonation waves before an expansion corner, Journal of Fluid Mechanics 940. doi:10.1017/jfm.2022.226.

[104] X. Zhuo, D. Gang, P. Zhenhua, G. Mingyue, Standing window of oblique detonation with pathological behaviour, Chinese Journal of Aeronautics 34 (5) (2021) 496–503. doi:10.1016/j.cja.2020.12.005.

[105] H. Guo, N. Zhao, H. Yang, S. Li, H. Zheng, Analysis on stationary window of oblique detonation wave in methane-air mixture, Aerospace Science and Technology 118 (2021) 107038. doi:10.1016/j.ast.2021.107038.

[106] F. García, M. Nürmberger, R. Torres, J. Crespo, ARION 1 reusable sounding rocket: The new microgravity platform in Europe, Adv. Astronaut. Sci. 1 (1) (2018) 23–30.

[107] M. Grossi, A. Sereno, D. Bianchi, B. Favini, Role of finite-rate kinetics on the performance predictions of solid rocket motor nozzles, in: AIAA SCITECH 2023 Forum, 2023, p. 1314. doi:10.2514/6.2023-1314.

[108] H. Jones, The recent large reduction in space launch cost, in: 48th International Conference on Environmental Systems, 2018, pp. 1–12, Available: http://hdl.handle.net/2346/74082.

[109] I. W. Kokkinakis, D. Drikakis, Atmospheric pollution from rockets, Physics of Fluids 34 (5) (2022) 056107. doi:10.1063/5.0090017.

[110] A. S. Gohardani, J. Stanojev, A. Demairé, K. Anflo, M. Persson, N. Wingborg, C. Nilsson, Green space propulsion: Opportunities and prospects, Progress in Aerospace Sciences 71 (2014) 128–149. doi:10.1016/j.paerosci.2014.08.001.

[111] J. Dallas, S. Raval, J. A. Gaitan, S. Saydam, A. Dempster, The environmental impact of emissions from space launches: A comprehensive review, Journal of Cleaner Production 255 (2020) 120209. doi:10.1016/j.jclepro.2020.120209.

[112] H. Fangohr, A comparison of C, MATLAB, and Python as teaching languages in engineering, in: International Conference on Computational Science, Springer, 2004, pp. 1210–1217. doi:10.1007/978-3-540-25944-2_157.