CEA2022: A Modernization of NASA Glenn's Software CEA (Chemical Equilibrium with Applications)

Mark K. Leader*, Thomas M. Lavelle, Xiao-yen J. Wang, Kevin W. Dickens, Michael McTague
NASA Glenn Research Center, Cleveland, OH, USA

Jeffrey P. Hill
NASA Ames Research Center, Mountain View, CA, USA

The software program "Chemical Equilibrium with Applications" (CEA) is used to solve chemical equilibrium, and compute thermodynamic and transport properties of the resulting mixture, and also has special solvers dedicated to rocket, shock, and detonation problems. We have recently completed a full re-write of CEA with modernization and improvements, called "CEA2022". In this paper, we will give an overview of CEA2022's features, and discuss some of the fundamental equations used by CEA2022, as well as the fundamental assumptions, in order to provide users with a complete understanding of the software's methodology. Several enhancements have been made to the software, including modern software development practices, interface improvements, and additional features. The feature enhancements include: running cases in parallel with thread safe solves, thermodynamic and transport database updates, and allowing for negative and inert reactants. In terms of interface improvements, we have made CEA a reusable library by adding APIs for multiple languages, including Python, Matlab, Excel, Fortran, and C. The subroutine interface allows for integration with other applications, including flow-solver integration (i.e. with CFD). We also compare results between CEA2022 and the previous version (CEA2) as a validation of the new software.

Nomenclature

A	=	stoichiometric coefficient matrix
$\mathbf{b}$, b_i	=	element amounts
c_p	=	specific heat capacity at constant pressure
g	=	Gibb's free energy
h	=	enthalpy
n	=	total moles of the product mixture
$\mathbf{n}$, n_j	=	species concentrations
n	=	total mixture moles
NE	=	number of elements
NG	=	number of gas-phase species
NS	=	number of species
P	=	mixture pressure
S	=	entropy
T	=	mixture temperature
η	=	viscosity
$\boldsymbol{\lambda}$, λ_i	=	Lagrange multipliers
λ	=	thermal conductivity
μ	=	chemical potential
$\boldsymbol{\pi}$	=	modified Lagrange multipliers

*mark.leader@nasa.gov

Subscripts

c = condensed-phase species
eq = equilibrium
fr = frozen
g = gas-phase species
re = reaction

I. Introduction

The NASA computer program "Chemical Equilibrium with Applications" (CEA) * is used to calculate chemical composition, and thermodynamic and transport properties for a variety of problem types, including: equilibrium with assigned thermodynamic states, theoretical rocket performance, Chapman-Jouget detonations, and shock-tube parameters for incident and reflected shocks. The work resulting in CEA began at NASA Lewis (now Glenn) in 1948, and has a strong history at the center since that time †.

The current version of CEA (called "CEA2") was completed in 2002, and uses a text-file interface to input problem parameters to the Fortran source code. CEA2 is a robust software program that has been utilized heavily in government, academia, and industry. However, because the last official release was in 2002, an update is in order to modernize the program and support several requested updates. That is the impetus behind this work. We have set out to maintain the existing functionality of CEA2, while implementing many upgrades, include software modernization, feature updates, and interface improvements.

In Section II, we will describe the approach used by CEA to solve each problem type. In Section III, we will discuss the software interface options for CEA2022. Next, in Section IV, we present validation cases and compare the accuracy of CEA2022 with CEA2 as the reference. Finally, we give concluding remarks in Section V.

II. Methodology

In this Section, we provide a non-exhaustive summary of the analysis procedure used in CEA2022 for each application. The full details of the solution procedures used by CEA2 can be found in Gordon and McBride [1]. We use the same solution procedures here, unless otherwise noted.

A. Equilibrium Problems

In this section, we describe the underlying equations and solution procedure for the chemical equilibrium problems in CEA. The equilibrium application of CEA takes a reactant mixture, and a pair of specified thermodynamic states, and then computes the resulting equilibrium composition and thermodynamic properties. The accepted pairs of prescribed thermodynamic states are:

- temperature and pressure, "TP"
- enthalpy and pressure, "HP"
- entropy and pressure, "SP"
- temperature and volume, "TV"
- energy and volume, "UV"
- entropy and volume, "SV"

where the "volume" problems can also accept a value for specific density. The equilibrium problem is also called by each of the other applications in CEA, so it is the most fundamental problem type.

A primary input of an equilibrium problem is the reactant mixture; this is often a mixture of a fuel and an oxidant, but that is not a requirement. A reactant mixture is the initial state, and the product mixture is the set of all possible species that can be formed through interactions of the reactants. The products can be specified by the user (typically done if the user wants to control which species are allowed), or the product mixture can be determined by CEA.

CEA2 is very flexible with how the amounts of each reactant are specified – the user can specify amounts in terms of weights, moles, oxidant-to-fuel ratios, fuel-to-air ratio, percent of fuel by weight, or equivalence ratios in terms of either

*https://www1.grc.nasa.gov/research-and-engineering/ceaweb/
†https://www1.grc.nasa.gov/research-and-engineering/ceaweb/ceahistory/

weight-ratios (ϕ) or valences (r). CEA2022 provides helper-functions for the user to convert any of these quantities into reactant weights before the solve is called - which is the only accepted parameter for the "solve" functions.

A primary assumption of CEA is that "all gases are ideal and interactions among phases are ideal"[1]. Thus, the equation of state governing the mixture is:

$$PV = n_g RT, \tag{1}$$

where n_g is the total moles of gas-phase species in the mixture, and is defined as

$$n_g = \sum_{j=1}^{NG} n_j \tag{2}$$

where n_j is the species concentration of the j-th species in the product mixture, and NG is the total number of gas species in the product mixture. CEA allows condensed species to be included as products and reactants, but they are assumed to contribute no volume.

The equilibrium state of the mixture is found through minimizing the Gibbs energy, given by

$$g = \sum_{j=1}^{NS} \mu_j n_j \tag{3}$$

where NS is the total number of species in the product mixture, and μ_j is the chemical potential of the j-th species.

The minimization of Gibbs energy is constrained by the element amounts of the initial reactant mixture. This constraint is expressed as:

$$\sum_{j=1}^{NS} a_{ij} n_j - b_i^\circ = 0, \quad i = 1, \ldots, NE \tag{4}$$

where a_{ij} are the elements in the stoichiometric coefficient matrix $\mathbf{A}$, b_i° are the amounts of each element in the reactant mixture, and NE is the number of elements in the mixture.

Combining equations (3) and (4), we can form a Lagrangian to find the minimum of the Gibbs energy while constraining the element amounts:

$$\mathcal{L} = g + \sum_{i}^{NE} \lambda_i \left(b_i - b_i^\circ \right) \tag{5}$$

Finding the stationary point of this Lagranigan results in the equations which determine chemical equilibrium. Some of the equations defining the equilibrium condition are nonlinear, and therefore require an iterative procedure to converge the system. CEA2022 uses the same Newton method as CEA2 to iterate to convergence. This solution procedure requires initial estimates for $\mathbf{n}$, n, and T. The initial estimate used by both CEA2 and CEA2022 is:

$$\begin{aligned} n &= 0.1 \text{ mol/kg-mol} \\ n_j &= \begin{cases} 0.1/NG, & j = 1, \ldots, NG \\ 0, & j = NG+1, \ldots, NS \end{cases} \\ T &= 3800 \text{ K} \end{aligned}$$

The details of how to get from (5) to the Gibbs iteration equations – and ultimately the matrix system of equations – can be found in Gordon and McBride [1].

1. Iteration Procedure and Convergence Criteria

The iterative procedure solution described above results in the following update variables: $\Delta \ln(\mathbf{n_g})$, $\Delta \mathbf{n_c}$, $\Delta \ln(n)$, $\Delta \ln(T)$, and $\boldsymbol{\pi}_i = -\boldsymbol{\lambda_i}/RT$. After each iteration of the Newton method solver, these update variables are translated to the solution variables of interest ($\mathbf{n}$, n, and T, primarily), and convergence checks are also performed. As part of the solution update procedure, a damped update is applied to improve convergence. A damped update factor, λ, is computed each iteration such that, if the solution is far for convergence, λ will be less than 1, and when the solution is close to

equilibrium, λ will be equal to 1. The procedure for computing λ can be found in Gordon and McBride [1]. Using the damped update factor, the new solution variables at the next iteration, $k+1$, are computed as:

$$\begin{aligned}
\ln(n_j)^{k+1} &= \ln(n_j)^k + \lambda(\Delta \ln(n_j)), \quad j = 1, \ldots, NG \\
n_j^{k+1} &= n_j^k + \lambda(\Delta n_j), \quad j = NG+1, \ldots, NS \\
\ln(n)^{k+1} &= \ln(n)^k + \lambda(\Delta \ln(n)) \\
\ln(T)^{k+1} &= \ln(T)^k + \lambda(\Delta \ln(T))
\end{aligned} \tag{6}$$

It is important to note that, in both versions of CEA, n_j does not necessarily equal $\exp(\ln(n_j))$. For species with very small concentration, beyond a certain threshold are treated as having a concentration of 0. This can be controlled using a factor called *trace*, but by default, species with concentrations lower than $n_j/n < 10^{-8}$ are treated as having a concentration of 0. However, the log of the species concentration ($\ln(n_j)$) does not get zeroed-out at the same time; this allows the species to come in and out of this threshold range without any loss of information. This explains why n_j and $\ln(n_j)$ are treated separately internally in the program.

After the solution variables are updated, a convergence test is performed. Convergence is checked separately for each of: $\mathbf{n}_g$, $\mathbf{n}_c$, n, $\mathbf{b}$, $\boldsymbol{\pi}$, T (if relevant), and s (if relevant). The exact convergence checks can be found in Gordon and McBride [1], but checks that each variable is similar to its value from the previous iteration, indicating that equilibrium has been found. If any of these solution variables changed noticeably from the previous iteration, another iteration is taken; otherwise, final checks and post-processing steps are performed.

2. Additional Considerations

CEA2022 has several useful features that utilize alternate modes of operation and require functionality beyond what has been described up to this point. These include condensed species, ionized species, negative reactants, and inert reactants.

Condensed Species CEA supports the inclusion of condensed species in both products and reactants. This is somewhat in contrast with the first assumption of CEA, that "all gasses are ideal", and governed by Eq. (1), but this is resolved with a secondary assumption that condensed species take up a negligible volume in the mixture. This is generally acceptable, but important for the end user to keep in mind when interpreting results.

Unless specified by the `insert` variable, the initial guess for the condensed species concentrations are assumed to be zero. CEA first converges a solution with only gas-phase species. At that point, CEA performs a test to determine if any condensed species should be added. This is done by checking if adding any would decrease the total Gibbs free energy of the mixture. If any meet this criteria, the condensed species that decreases the Gibbs energy the most is added to the mixture, and the iteration procedure is resumed. This cycle is repeated until convergence. Special considerations are considered for phase transitions, and for allowing multiple phases of the same species, and these details can be found in Gordon and McBride [1].

Ionized Species CEA supports ionized species for equilibrium calculations. Using this option adds an additional constraint to the problem to ensure charge-balance is met, which is expressed as:

$$\sum_{j=1}^{NG} a_{ej} n_j = 0 \tag{7}$$

where a_{ej} represents the charge of the j-th species. This feature is available in the other applications as well, because they call the equilibrium solver. There are additional adjustments made to improve convergence with this option, and the details can be found in Gordon and McBride [1].

Negative and Inert Reactants CEA2022 includes two new features for handling special cases of reactants: reactants with negative amounts, and inert reactants. Negative reactants could be useful in a scenario such as water precipitating from a flow. Inert reactants could be used in a case with incomplete combustion, where a user may wish to have part of the fuel react, and another portion not react. These are both non-equilibrium analyses, while CEA is by definition an equilibrium solver, so any results using these features should be used and interpreted carefully.

To use the negative reactant feature, any reactant weight can simply be specified with a negative value. This has two primary consequences:

1) The element constraint value is subtracted by the appropriate amount. The user should ensure that this does not cause any individual element amount to be negative, as CEA will not be able to find equilibrium with a net-negative amount of an element.
2) If the case uses fuel and oxidant temperatures to compute the constraint value (e.g. HP, SP, UV, or SV problems where $H^\circ/S^\circ/U^\circ$ are not specified by the user), CEA will subtract the appropriate amount when computing the constraint value.

To use the inert reactant feature, the inert species amount should be specified as a separate reactant, with "Inert" prefixed to the reactant name (e.g. "CH4" and "InertCH4", each with their own reactant weights). Only species containing some combination of the elements C, H, and O have an option to be treated as inert. A list of reactants that currently support an inert version is provided in the Appendix in Table V.B. When an inert element is used, internally in CEA, a separate version of each element is used, eg. "IC" instead of "C". This causes inert elements to have their own element-amount constraints, and also not react with the active versions of the species.

B. Thermodynamic Properties

A fundamental capability of CEA is the ability to evaluate thermodynamic properties for each species. The CEA thermodynamic database contains 2,012 products and 62 additional reactants. Thermodynamic properties of these species are evaluated using a piece-wise polynomial curve fit. For gas species, the curve fit uses the intervals (in Kelvin): [200, 600], [600, 1000], [1000, 6000], [6000, 20000], and condensed species each have custom temperature intervals defined in the data-set. The polynomial curve fit equations are provided in Appendix V.C, as well as Gordon and McBride [1].

In the case of specific heat capacity, the thermodynamic curve fit equations only provide the *frozen* specific heat capacity of the mixture, which is the weighted-sum of each of the species' heat capacity. However, this is not sufficient for an equilibrium calculation: consider the definition of specific heat capacity is

$$c_p \coloneqq \left(\frac{\partial h}{\partial T}\right)_P \tag{8}$$

where for a mixture, h is computed as:

$$h = \sum h_j n_j \tag{9}$$

By applying this to the original definition in Equation (8), we can expand as:

$$\left(\frac{\partial h}{\partial T}\right)_P = \sum \left(n_j \frac{\partial h_j}{\partial T} + \frac{\partial n_j}{\partial T} h_j\right) \tag{10}$$

The left-hand-side of the above equation is the equilibrium heat capacity of the mixture. The first term in the right-hand-side sum is the "frozen" heat capacity – again, this is the sum of each species' heat capacity weighted by the species concentration. The second term in the right-hand-side is the "reaction" heat capacity. This term is relevant because this is an equilibrium mixture, and species concentrations would change if temperature were to change.

As part of this modernization effort, the thermochemical and transport properties of the green propellants Ammonium DiNitramide (ADN), LMP-103S, HydroxylAmmonium Nitrate (HAN), and AF-M315E, a.k.a., ASCENT, as well as for the sustainable aviation fuels n-Butanol and Biodiesel have been included in the updated version of CEA, i.e., CEA2022. This will enable CEA2022 end users to perform chemical equilibrium product concentrations-related calculations from the given set of reactants for any of the aforementioned newly added green propellants and sustainable aviation fuels, in addition to any of the pre-existing chemical species, fuels, and propellants previously available.

C. Rocket Problem

CEA is able to analyze rocket performance assuming either an infinite-area combustor (IAC) or a finite-area combustor (FAC). In addition to specifying reactants, the rocket solver requires an assigned chamber pressure. Additionally, exit parameters should be specified in terms of either: chamber pressure to exit pressure ratio, P_c/P_e, or exit area to throat area ratio, A_e/A_t. For FAC problems, either contraction ratio, A_c/A_t, or the ratio of mass flow rate to chamber area, $\dot{m}/A_c$, is required. The rocket solver converges each station (combustor, throat, and exit), in order, and iterates until convergence using the update procedure described in Gordon and McBride [1]. By default, the equilibrium module is

called at each station to solve for the equilibrium chemical composition. However, the rocket solver can be called in "frozen" mode, and a station specified where the equilibrium analysis should be done. In this case, the equilibrium analysis is only performed up to the frozen station, and is fixed thereafter.

D. Shock Problem

Shock tube problems can be solved by CEA for both incident and reflected shock values. The shock problem uses either the incident shock velocity or Mach number, as well as un-shocked pressure and temperature values, as the input parameters. CEA then uses an iterative solution procedure to converge each stage of the shock problem.

E. Chapman-Jouguet Detonation Problem

Users may use CEA to compute parameters for Chapman-Jouguet detonation problems. The solution procedure starts with an initial guess of temperature and pressure, and then uses a Newton method to converge the parameters using an update procedure that is described in Gordon and McBride [1]; the solution procedure used in CEA2022 is the same as it was in CEA2 with no notable changes.

F. Transport Properties

Transport properties of an product solution can be output (optionally) after solving the equilibrium condition. Certain transport properties are also used internally in the rocket, shock, and detonation modules. CEA computes the viscosity, η, and thermal conductivity, λ, of the mixture using a least-squares based polynomial curve fit, given as:

$$\begin{cases} \ln \eta \\ \ln \lambda \end{cases} = A \ln T + \frac{B}{T} + \frac{C}{T^2} + D \tag{11}$$

where A, B, C, and D are the coefficients from the CEA transport database. For some species, an interaction value η_{ij} is added. There are other considerations when computing the transport properties that can be found in Gordon and McBride [1].

An important distinction with respect to transport properties is whether the user wants equilibrium properties or "frozen" properties. Thermal conductivity, for example, is computed as:

$$\lambda_{\text{eq}} = \lambda_{\text{fr}} + \lambda_{\text{re}} \tag{12}$$

where the subscripts "eq", "fr", and "re" denote "equilibrium", "frozen", and "reaction", respectively. Viscosity, on the other hand, only has a frozen value. The equations to compute λ_{fr} and λ_{re} can be found in Gordon and McBride [1].

As a note to the end user – especially for rocket problems: there is a difference between doing a "frozen" rocket analysis, and the "frozen" transport properties. "Frozen" rocket properties uses equilibrium computed at a specific location and treats this is constant for the downstream stations. This case will still have both equilibrium and frozen transport properties, and similarly, an equilibrium rocket problem will still report frozen transport properties. It is up to the user to decide whether frozen or equilibrium transport properties are more relevant for their purpose.

III. Software Interface

CEA2 operates exclusively through an input file interface. This makes interfacing with CEA challenging if another application wants to call CEA. For this reason, CEA2022 has emphasized the use of a subroutine interface in multiple programming languages. The primary programming language for CEA2022 is Fortran 2008. CEA2022 differs from CEA2 in that it supports a Fortran subroutine interface. CEA2022 also accepts the legacy input files used by CEA2 for backwards compatibility. The Fortran application programming interface (API) can be called directly from a Fortran script, or imported for use in another Fortran library. From the Fortran API, several other languages are made available, including: C, Python, and Matlab, and with Excel soon to come. The C-language interface calls the Fortran API, and is then used as the basis for all other interfaces. The call structure of the multi-language bindings in CEA2022 can be seen in Figure 1. Each of the interfaces will be described in the following subsection, but, at a high level: the C interface calls the Fortran API, the Python and Excel interfaces call the C API, and the Matlab interface calls the Python API. Regardless of the choice of interface that a user chooses, the analysis will be ultimately solved in Fortran. This results in faster execution times for the interpreted languages (Python, Excel, and Matlab).

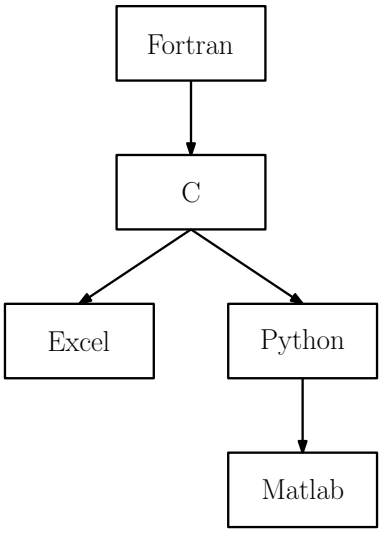

Fig. 1 Hierarchy of interface language calls.

A. Fortran Interface

Fortran 2008 is used as the primary programming language for CEA2022. The program is broken into one module for each application ("equilibrium", "rocket", "shock", "detonation"), as well as other ancillary support modules. Each module contains several functions which make up the basis for the Fortran API.

The decision to use Fortran 2008 for CEA2022 was made due to fast computational speed, and selecting a language similar to the original language of CEA2 was intentional to simplify the migration process. CEA2022 is a re-write of CEA2, not simply a translation into a more modern version of Fortran. In doing this, we took advantage of modern software development practices, and utilized the object-oriented nature of Fortran 2008 to improve the quality of the code both for the developers and the end-users. Each module in CEA2 contains custom `types` which are populated with relevant values. For example, the equilibrium module has custom types for `EqSolver` and `EqSolution`. To solve an equilibrium problem, the user first creates a *reactant* and *product* instance of a `Mixture` object. These reactant and product mixture objects store the species names, their curve fit data, and the reactant temperatures. The reactants and product mixtures are passed in to the `EqSolver` object when it is instantiated. The user can then call the `solve` method on the `EqSolver` object, which populates the `EqSolution` object. The `EqSolution` object store the mixture's temperature, species concentration, total moles, and other desired quantities. An example of the Fortran API is shown in Listing 1, which illustrates how to solve Example 2 from Gordon and McBride [2] using the Fortran API.

B. C Interface

The C interface of CEA is a direct wrapper of the Fortran interface using the Fortran `bind(c)` feature. The wrapper declares functions in Fortran using C-compatible data types, and declares the function with a `bind(c)` so that they may be called from a C program or script. The C interface is then available to be called from each of the additional programming languages available in CEA2022.

C. Python Interface

CEA2022 is callable through Python. The Python wrapper imports the C-binding of CEA2022 through the Cython[‡] package. To enable this, we create a definition file to declare the function calls and data types in the C interface, and then import that definition file into a Python wrapper file (`.pyx` extension). This allows us to write a CEA2022 Python package with an API that is more Python-ic, while using the function-based C binding.

[‡] `https://cython.org`

Listing 1 Fortran API example

```
use cea_equilibrium
use cea_thermo
use cea_mixture
use cea_units

type(ThermoDB) :: all_thermo
type(EqSolver) :: solver
type(EqSolution) :: solution
type(Mixture) :: reac, prod
real(dp) :: v_reac, weights(2), of_ratio

all_thermo = read_thermo('data/thermo.lib')
reac = Mixture(all_thermo, ['H2 ', 'Air'])
prod = Mixture(all_thermo, [ &
    'Ar  ', 'C   ', 'CO  ', 'CO2 ', 'H   ', &
    'H2  ', 'H2O ', 'HNO ', 'HO2 ', 'HNO2', &
    'HNO3', 'N   ', 'NH  ', 'NO  ', 'N2  ', &
    'N2O3', 'O   ', 'O2  ', 'OH  ', 'O3  '  &
])

solver = EqSolver(prod, reac)
solution = EqSolution(solver)

! Get the oxidant to fuel ratio
of_ratio = reac%of_from_equivalence([0.0d0, 1.0d0], [1.0d0, 0.0d0], 1.0d0)
weights = reac%weights_from_of([0.0d0, 1.0d0], [1.0d0, 0.0d0], of_ratio)

v_reac = convert_units_to_si(1.0d0/9.1864d-05, "cm**3/g")
call solver%solve(solution, 'tv', 3000.0d0, v_reac, weights)
```

D. Matlab Interface

The Matlab interface imports a custom Python library written specifically for Matlab. Matlab supports importing Python libraries, but does not allow passing object instances into other class functions. This paradigm breaks the standard Python interface, because we pass in class-type `Reactants` into the `EqSolution` and `EqSolver` classes, as just one example.

IV. Results

In this section, we provide validation examples for each application type of CEA by comparing results obtained using CEA2022 against results using CEA2. For each case, the problem inputs will be described, followed by a comparison of results.

A. Equilibrium Example

Here we provide a demonstration of the equilibrium application using Example 4 from Gordon and McBride [2]. This is a UV problem with $\rho = 14.428$ kg/m^3 and $u/R = -45.1343$ (kg-mol)(K)/kg. The oxidant-to-fuel ratio is 17. `Air` is used as the only oxidant, and has a temperature of 700 K. The fuel is a mixture of `C7H8(L)` and `C8H19(L)` with a 40/60 weight-fraction of each, and a temperature of 298.15 K. The results for this problem are provided in Table 1. We provide temperature, and a subset of species concentrations to compare. The log of the species concentration is provided, because this is what CEA uses internally, and this also provides a better basis for comparison because it is more independent of the absolute quantity of the species, and allows us to show the accuracy for species that show up in large concentration as well as trace species. This example shows that CEA2022 has an accuracy of $O(10^{-6})$ or better for each value shown, indicating very good agreement for the equilibrium solver.

Table 1 Equilibrium example of a UV problem.

	CEA2	CEA2022	Relative Error (%)
T (K)	**2418.538**2574474097	**2418.538**4473019085	7.85×10^{-6}
$\ln(n_j)$: $(HCOOH)_2$	**-52.3748**90133250204	**-52.3748**88695317857	2.75×10^{-8}
$\ln(n_j)$: Ar	**-8.09402656**40663025	**-8.09402656**28475903	1.51×10^{-6}
$\ln(n_j)$: C	**-42.9570**51147438705	**-42.9570**47510708030	8.47×10^{-6}
$\ln(n_j)$: CO	**-9.75915**61204275976	**-9.75915**54029655168	7.35×10^{-6}

B. Rocket Example

Here we provide an example of the CEA rocket application. The problem details come from Example 8 in Gordon and McBride [2]. This rocket problem uses an infinite-area combustor (IAC), with `H2(L)` as the fuel at 20.27 K, and `O2(L)` as the oxidant at 90.17 K, and an oxidant-to-fuel ratio of 5.55157. With the exception of throat temperature, each variable shows an accuracy of $O(10^{-5})$, indicating good agreement. The error of the throat temperature is on the order of 10^{-3} which is acceptable, and can be explained by two factors: temperature is not used as the update variable in the equilibrium solver - log temperature is used instead, and the second factor is that, the rocket solver uses an outer convergence loop between each station, which may amplify any small numerical discrepancies in the equilibrium solver.

C. Shock Example

Table 3 gives values for an example shock problem using both CEA2 and CEA2022. Values provided for comparison include temperature, pressure, and the log concentration for a subset of species for both the incident and reflected shocks. The example comes from Example 7 in Gordon and McBride [2], which uses a mixture of H_2, O_2, and Ar, with 0.9 moles of Ar, and 0.05 moles of each of H_2 and O_2, with all gases initially at 300 K. In this example, we use a pressure of 0.1 bar, and an initial velocity of 1,400 m/s. The values computed using CEA2022 only differ from the CEA2 solution by between 10^{-4} and 10^{-5}, indicated good agreement between the two programs for shock tube problems.

Table 2 Rocket problem example using an infinite-area combustor.

Station	Variable	CEA2	CEA2022	Relative Error (%)
Combustor	T (K)	**3383.8446259451043**	**3383.8446259451043**	0.0
Combustor	$\ln(n_j)$: H_2	**-3.7643107135196541**	**-3.7643107135196558**	4.72×10^{-14}
Throat	T	**3185.67**31730807096	**3185.63**35035883212	1.25×10^{-3}
Throat	$\ln(n_j)$: H_2	**-3.77589**18668469459	**-3.77589**38050310880	5.13×10^{-5}
Exit (p_i/p_e)	T	**2567.34**01804441942	**2567.34**11236822476	3.67×10^{-5}
Exit (p_i/p_e)	$\ln(n_j)$: H_2	**-3.78804132**04534467	**-3.78804132**98752891	2.49×10^{-7}

Table 3 Shock tube problem example results.

Type	Variable	CEA2	CEA2022	Relative Error (%)
Incident	T (K)	**2268**.1839840465000	**2268**.4047056277263	9.30×10^{-5}
Incident	P (bar)	**1.923**5852279853596	**1.923**7641552369096	9.73×10^{-5}
Incident	$\ln(n_j)$: HO_2	**-17.22**3079678243373	**-17.22**2396748742256	3.97×10^{-5}
Incident	$\ln(n_j)$: OH	**-9.625**9080229365601	**-9.625**1244117341326	8.14×10^{-5}
Reflected	T (K)	**325**3.2881222920269	**325**4.3115063133464	3.15×10^{-4}
Reflected	P (bar)	**6.88**57588009600743	**6.88**63847357399406	9.09×10^{-5}
Reflected	$\ln(n_j)$: HO_2	**-15.14**9034932828400	**-15.14**8447374699135	3.88×10^{-5}
Reflected	$\ln(n_j)$: OH	**-7.68**11535944113283	**-7.68**01166285479265	1.35×10^{-4}

D. Detonation Example

For the validation example of the Champan-Jouget detonation problem, we compare results based on Example 6 from Gordon and McBride [2], which is a detonation with H_2 and O_2 as the fuel and oxidant, with an equivalence ratio $r = 1$, with the unburned gas at 1 bar and 298.15 K. The results are provided in Table 4, comparing temperature, pressure, and the log concentrations of a subset of species between CEA2 and CEA2022. Each of the values provided shows good agreement, with relative error on the order between 10^{-7} and 10^{-9} for each value.

Table 4 Chapman-Jouguet detonation problem example results.

	CEA2	CEA2022	Relative Error (%)
T (K)	**3674.2**808311051022	**3674.2**778635982045	8.08×10^{-7}
P (bar)	**18.7684**46693571065	**18.7684**59579609036	6.87×10^{-7}
$\ln(n_j)$: H_2	**-4.493612**3449492360	**-4.493612**4180425319	1.63×10^{-8}
$\ln(n_j)$: H_2O_2	**-13.4918960**07609208	**-13.4918960**98472500	6.73×10^{-9}
$\ln(n_j)$: OH	**-4.630083**6293723897	**-4.630083**9021590434	5.89×10^{-8}

V. Conclusion

In this work, we described the update to NASA's CEA software program, CEA2022. Like CEA2, CEA2022 can be used to solve equilibrium chemical composition analysis, as well as analyze theoretical rocket performance, shock tube characteristics, and Chapman-Jouguet detonation problems. Several features have been added to CEA2022 that were not previously available, including negative and inert reactants, and additions to the thermodynamic database to include green propellants. CEA2022 supports the legacy input and output file format, but emphasizes the use of a subroutine interface, and we have made APIs available in several programming languages, including Fortran, C, Python, and Matlab, with Excel expected very soon. Finally, we demonstrated consistent results with CEA2, providing validation examples for each application in CEA.

Appendix

A. Chemical Potential Equation

$$\mu_i = \begin{cases} \mu_j^\circ + RT \ln\left(\frac{n_j}{n}\right) + RT \ln(P), & j = 1, \ldots, NG \\ \mu_j^\circ, & j = NG+1, \ldots, NS \end{cases} \tag{13}$$

B. Reactants with an Inert Version

InertC10H8,naph	InertC2H4	InertCH4	InertH	InertH2	InertO
InertO2	InertAir	InertH2(L)	InertJP-10(L)	InertJP-10(g)	InertJP-4
InertJP-5	InertJet-A(L)	InertJet-A(g)	InertO2(L)	InertRP-1	

C. Thermodynamic Curve Fit Equations

$$\frac{C_p}{R} = a_1 T^{-2} + a_2 T^{-1} + a_3 + a_4 T + a_5 T^2 + a_6 T^3 + a_7 T^4 \tag{14}$$

$$\frac{H^\circ}{RT} = -a_1 T^{-2} + a_2 T^{-1} \ln T + a_3 + a_4 \frac{T}{2} + a_5 \frac{T^2}{3} + a_6 \frac{T^3}{4} + a_7 \frac{T^4}{5} + \frac{a_8}{T} \tag{15}$$

$$\frac{S^\circ}{R} = -a_1 \frac{T^{-2}}{2} - a_2 T^{-1} + a_3 \ln T + a_4 T + a_5 \frac{T^2}{2} + a_6 \frac{T^3}{3} + a_7 \frac{T^4}{4} + a_9 \tag{16}$$

Acknowledgments

The work presented in this paper was developed with support from the NASA Engineering and Safety Center (NESC), NASA's Hypersonic Technology Project (HTP), Transformational Tools and Technologies (TTT), and Revolutionary Vertical Lift Technologies (RVLT). The authors sincerely thank Christopher Snyder for his guidance and mentorship on this work.

References

[1] Gordon, S., and McBride, B. J., "Computer program for calculation of complex chemical equilibrium compositions and applications. Part 1: Analysis," Tech. rep., NASA, 1994. URL https://ntrs.nasa.gov/citations/19950013764.

[2] Gordon, S., and McBride, B. J., "Computer program for calculation of complex chemical equilibrium compositions and applications. Part 2: Users Manual and Program Description: Analysis," Tech. rep., NASA, 1994. URL https://ntrs.nasa.gov/citations/19960044559.