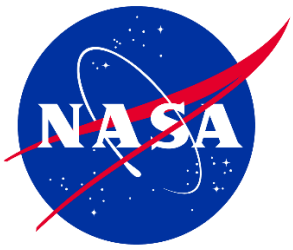
NASA/TM–20260007987
NESC-RP-22-01730



Updates and Modernization of the Chemical Equilibrium with Applications (CEA) Code

Mark K. Leader
Glenn Research Center, Cleveland, Ohio

Jonathan E. Jones/NESC
Langley Research Center, Hampton, Virginia

Kevin W. Dickens
Glenn Research Center, Cleveland, Ohio

August 2026

NASA STI Program Report Series

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

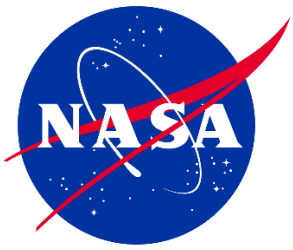
For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- Help desk contact information:

https://www.sti.nasa.gov/sti-contact-form/
and select the "General" help request type.

NASA/TM–20260007987
NESC-RP-22-01730



Updates and Modernization of the Chemical Equilibrium with Applications (CEA) Code

Mark K. Leader
Glenn Research Center, Cleveland, Ohio

Jonathan E. Jones/NESC
Langley Research Center, Hampton, Virginia

Kevin W. Dickens
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

August 2026

Acknowledgements

The assessment team would like to acknowledge the significant contributions made by Christopher Snyder (retired), Scott Jones (retired), Francisco Guzman, Martin Rabinowitz, Jeff Moder, Juanpablo Delgado, and Kerem Evrenosoglu. Special thanks for the significant contributions made to this work by the many members of the NASA Glenn Research Center (GRC), ARC, White Sands Test Facility (WSTF), Marshall Space Flight Center (MSFC), and Goddard Space Flight Center (GSFC) teams, in addition to Branko Ruscic of the Argonne National Laboratory (ANL), Adam Brand of the Air Force Research Laboratory (AFRL), and Eric Lemmon of the National Institute for Standards and Technology (NIST). Thanks to the NASA Engineering and Safety Center (NESC) for funding this work, and thanks to Hypersonic Technology Project (HTP), Revolutionary Vertical Lift Technology (RVLT), and Transformational Tools and Technologies (TTT) projects for their funding support.

The assessment team would also like to acknowledge the significant contributions made in the preparation of this final report by retired assessment team members Michael McTague and Jeffrey Hill and to thank Donald Parker, Brett Cruden, Joseph Schulz, and Jeffrey Moder for their peer reviews of the report.

Available from:

NASA STI Program / Mail Stop 050
NASA Langley Research Center
Hampton, VA 23681-2199

NASA Engineering and Safety Center
Technical Assessment Report

Updates and Modernization of the Chemical Equilibrium with Applications (CEA) Code

TI-22-01730

Jonathan Jones, NESC Lead

Kevin Dickens, Technical Lead

July 23, 2026

Report Approval and Revision History

NOTE: This document was approved at the July 23, 2026, NRB.

Approved: Michael Kirsch Digitally signed by Michael Kirsch Date: 2026.08.04 16:19:20 -04'00'

NESC Director

Version	Description of Revision	Office of Primary Responsibility	Effective Date
1.0	Initial Release	Jonathan E. Jones, NASA Technical Fellow for Propulsion, MSFC	7/23/2026

Table of Contents

List of Figures

List of Tables

Technical Assessment Report

1.0 Notification and Authorization

The Chemical Equilibrium with Applications (CEA) code is used extensively by government, industry, and academia for combustion analyses. It is used to calculate chemical equilibrium compositions and properties of complex mixtures from any set of reactants and determine thermodynamic and transport properties for the product mixture. With a history that dates back more than 50 years, the code was in need of database updates (e.g., addition of green propellants) and framework improvements for handling contemporary combustion problems on modern computational platforms. The NASA Engineering and Safety Center (NESC) was requested to facilitate updates to the thermochemical and transport databases in CEA and modernize the code framework to enhance its usability.

Dr. Jonathan Jones was selected to lead this assessment, and Mr. Kevin Dickens was selected as technical lead. Key stakeholders for this assessment include all NASA flight programs, including air breathing, hypersonic, and in-space; subject matter experts (SMEs) in propulsion, materials, chemistry, atmospheric sciences, and entry, descent, and landing (EDL); the NASA Ames Research Center (ARC) Aerothermodynamics Branch; and other government agencies, industry, and academia working on combustion-related technologies.

2.0 Signatures

Submitted by: NESC Lead

JONATHAN JONES

Digitally signed by JONATHAN JONES
Date: 2026.08.04 14:13:29 -05'00'

Dr. Jonathan E. Jones

Significant Contributors:

Mark Leader

Digitally signed by Mark Leader
Date: 2026.08.04 13:34:23 -04'00'

Dr. Mark K. Leader

Thomas Lavelle

Digitally signed by Thomas Lavelle
Date: 2026.08.04 14:44:12 -04'00'

Mr. Thomas M. Lavelle

Xiao-yen Wang

Digitally signed by Xiao-yen Wang
Date: 2026.08.04 15:06:17 -04'00'

Dr. Xiao-yen J. Wang

Kevin Dickens

Digitally signed by Kevin Dickens
Date: 2026.08.04 13:47:05 -04'00'

Mr. Kevin W. Dickens

Signatories declare the findings, observations, and NESC recommendations compiled in the report are factually based from data extracted from program/project documents, contractor reports, and open literature, and/or generated from independently conducted tests, analyses, and inspections.

3.0 Team Members

Name	Discipline	Organization/Host Center
Core Team		
Jonathan Jones	NESC Lead; NASA Technical Fellow for Propulsion	NESC/MSFC
Kevin Dickens	Technical Lead; NASA Deputy Technical Fellow for Propulsion	GRC
Mark Leader	Propulsion Engineer	GRC
Jeffrey Hill	Aerothermodynamics Branch Chief	ARC (retired)
Thomas Lavelle	Thermodynamics and Propulsion Systems Analyst	GRC
Michael McTague	Propulsion Engineer	GRC (retired)
Xiao-yen Wang	Branch Chief, Propulsion Systems Analysis	GRC
Anthony Iannetti	Chemical Engineer	GRC
Joy Hamilton	Early Career Engineer, Propulsion Chemistry	WSTF
Michael Martin	Propulsion Engineer	MSFC
Daniel Dorney	Former NASA Technical Fellow for Propulsion	MSFC (retired)
Consultants		
Laura Maynard-Nelson	Software	GRC
Business Management		
John LaNeave	Program Analyst	MTSO/LaRC
Pamela Stacy	Program Analyst	MTSO/LaRC
Assessment Support		
Melissa Strickland	Project Coordinator	AMA/LaRC
Linda Burgess	Planning and Control Analyst	AMA/LaRC
Jonay Campbell	Technical Editor	AS&M/LaRC

3.1 Acknowledgments

The assessment team would like to acknowledge the significant contributions made by Christopher Snyder (retired), Scott Jones (retired), Francisco Guzman, Martin Rabinowitz, Jeff Moder, Juanpablo Delgado, and Kerem Evrenosoglu. Special thanks for the significant contributions made to this work by the many members of the NASA Glenn Research Center (GRC), ARC, White Sands Test Facility (WSTF), Marshall Space Flight Center (MSFC), and Goddard Space Flight Center (GSFC) teams, in addition to Branko Ruscic of the Argonne National Laboratory (ANL), Adam Brand of the Air Force Research Laboratory (AFRL), and Eric Lemmon of the National Institute for Standards and Technology (NIST). Thanks to the NASA Engineering and Safety Center (NESC) for funding this work, and thanks to Hypersonic Technology Project (HTP), Revolutionary Vertical Lift Technology (RVLT), and Transformational Tools and Technologies (TTT) projects for their funding support.

The assessment team would also like to acknowledge the significant contributions made in the preparation of this final report by retired assessment team members Michael McTague and Jeffrey Hill and to thank Donald Parker, Brett Cruden, Joseph Schulz, and Jeffrey Moder for their peer reviews of the report.

4.0 Executive Summary

Chemical Equilibrium with Applications (CEA) is a NASA combustion analysis code, originally developed by Sanford Gordon and Bonnie McBride, which has been in continuous use for more than 50 years [ref. 1]. CEA supports a wide range of applications, including air-breathing propulsion; hypersonics; rocket propulsion; entry, descent, and landing (EDL); propellant formulation; and materials science. The code is widely used across NASA, other government agencies, industry, and academia. To remain relevant, the analysis capabilities needed modernization, including incorporation of thermochemical and transport properties for green propellants and sustainable fuels, as well as updates to the software framework to improve computational efficiency and enable interoperability with modern engineering tools. The timing of this effort was critical, as much of the institutional knowledge resides with a group of subject matter experts (SMEs) who are approaching retirement. In response, the NASA Engineering and Safety Center (NESC) was tasked with facilitating updates to the thermochemical and transport databases and modernizing the code framework to improve usability and long-term sustainability.

As part of this effort, the CEA code was successfully rewritten in the Fortran 2008 programming language, meeting the planned scope with two exceptions, which are noted in Section 10. Comprehensive beta testing was completed, and the software release plan was fully executed. The modernized code has been released as open-source software on GitHub and published in the NASA Technology Transfer Office Software Catalog, initially released as CEA 3.0.0. Validation testing showed agreement within 1% for 4,498 unique variables, with a single exception that can be attributed to differences in output formatting. Additionally, speed testing found CEA 3.0 to run 40% slower than CEA2 in standalone execution, within the factor of 2 mandated in the requirements (0.02685 seconds versus 0.01915 seconds). However, CEA 3.0 can perform much faster than CEA2 in some situations. For example, a parameter sweep requiring 108,500 equilibrium calculations can run in just 1.11 seconds using CEA 3.0 compared with 15 minutes and 3.96 seconds with CEA2.

4.1 Version Nomenclature

CEA2022 was the name initially given to this update to distinguish it from the previous version, CEA2. The updated version of the code has been released simply as “CEA,” starting at version 3.0, with the version number incrementing over time as appropriate. In this NESC report, “CEA 3.0” is used to refer to the current, updated version of the code, and the previous release is referred to as “CEA2” for clarity.

5.0 Assessment Plan

The NESC was requested to update the CEA thermochemical and transport databases, update the code structure (i.e., computer language and interfaces), and train the next generation of CEA SMEs to maintain and improve the analysis.

The scope of this effort included:

- Make the updated CEA code available to all NASA Centers, programs, and projects, as well as other government agencies, industry, and academia. This was completed by publishing the code open source on github.com/nasa/cea, with full documentation included online at nasa.github.io/cea.
- Train SMEs to update the CEA code, add chemical properties, and provide guidance on applications.
- Enter the updated CEA code into the NASA Technology Transfer Office Software Catalog.
- Deliver an NESC final report, NASA Technical Memorandum (TM), and NESC Technical Bulletin (TB).

6.0 Problem Description and Background

The CEA code has been used extensively by government, industry, and academia for a wide range of thermochemical and thermodynamic analyses. While commonly applied to combustion problems in air breathing propulsion, hypersonics, rocket propulsion, EDL, and propellant formulation, the CEA applications extend far beyond combustion analyses. The code is used throughout propulsion system analyses for analyzing compressors, turbines, burners, nozzles, and other components. Additionally, CEA finds application in diverse areas including fuel cell analyses, nuclear power working fluid calculations, materials research, and numerous other thermochemical problems. The code, however, needed both database updates (e.g., existing property updates and the addition of green propellants) and framework improvements for handling contemporary problems on modern computational platforms and interfacing with other programs. The NESC was requested to facilitate updates to the thermochemical and transport databases in CEA and modernize the code framework to enhance its usability.

The CEA code computes the chemical equilibrium compositions and properties of complex mixtures from any set of reactants and determines thermodynamic and transport properties for the resulting product mixture. The origins of the code began with work performed at NACA Lewis Flight Propulsion Laboratory (now NASA Glenn Research Center (GRC)) before Sputnik and before digital computers. Over the years, the work was expanded to include more combustibles and oxidizers and to solve more complex combustion conditions. A brief development history follows:

- **1947–1962, initial development:** The initial development of chemical equilibrium programs was performed by Vearl Huff, Virginia Morrell, and Frank Zeleznik, and was based on the work of Brinkley [ref. 2]. Their efforts were driven by the requirements of the Lewis Research Center to support various propulsion studies. The earliest versions of the program were relatively rudimentary, focusing on fundamental equilibrium calculations.
- **1965, contributions of Gordon and McBride:** Significant advancements were made through the contributions of Sanford Gordon and Bonnie McBride. Their work led to the creation of the NASA Lewis Chemical Equilibrium Program, which introduced more

sophisticated algorithms and expanded the program's capabilities to handle a broader range of chemical reactions and species [ref. 3].

- **1971, first comprehensive release:** The first comprehensive version of the software, known as the NASA Lewis Code, was released in 1971. This version was capable of calculating equilibrium compositions for a given set of reactants and conditions, and it provided essential thermodynamic properties such as enthalpy, entropy, and specific heat [refs. 4, 5].
- **1980s, enhancements and adaptations:** During the 1980s, the software underwent several enhancements to improve its accuracy and usability [refs. 6, 7, 8]. These updates incorporated more extensive thermodynamic data and refined algorithms to better handle complex mixtures and high-temperature conditions typical of aerospace applications.
- **1990s, transition to modern computing:** With the advent of more powerful computing resources, the 1990s saw the transition of CEA to modern computational platforms. This period marked the introduction of user-friendly interfaces and the integration of graphical output capabilities, making the software more accessible to a broader audience of engineers and researchers.
- **1994, CEA program release:** The culmination of these efforts was the release of the CEA program in 1994 [ref. 9]. This version, developed by Gordon and McBride, represented a significant leap forward in terms of computational efficiency and versatility. It featured comprehensive databases of thermodynamic properties and a robust framework for solving a wide array of chemical equilibrium problems.
- **2002, CEA2 program release:** CEA2 was written in Fortran 77. It is a robust chemical equilibrium solver with numerous applications spanning turbojet combustion and exhaust, rocket engines, shock waves, and detonation problems.
- **2025, CEA 3.0 Program Release:** The product of this NESC assessment was released as CEA version 3.0.0 (see Table 6.0-1). It is written in Fortran 2008 and was developed in compliance with NASA Procedural Requirement (NPR) 7150.2.

Table 6.0-1. Comparison of Legacy CEA (CEA2) versus Modernized CEA (CEA 3.0)

Feature	Legacy CEA (CEA2)	Modernized CEA (CEA 3.0)
Release Year	2002	2025
Programming Language	Fortran 77	Fortran 2008 (object-oriented, thread-safe)
Interface	Command-line only; input/output via text files	Multi-language application program interfaces (APIs): Fortran, C, Python, MATLAB, Excel planned; legacy command line interface (CLI) retained
Integration with Other Tools	Limited; required file input/output (I/O) for coupling	Direct API calls enable fast integration with engineering workflows
Execution Speed	Fast for single runs; very slow for parameter sweeps due to file I/O	Slightly slower for single runs (40% slower), but **~800x faster for large sweeps** via API (e.g., 108,500 cases in 1.11 sec vs. 15 min)
Database	Thermochemical and transport data for ~2000 species; no green propellants	Updated database; includes green propellants (i.e., ammonium dinitramide (ADN), hydroxylammonium nitrate (HAN), LMP-103S) and sustainable aviation fuels (SAFs)

Feature	Legacy CEA (CEA2)	Modernized CEA (CEA 3.0)
Supported Problem Types	Equilibrium, rocket, shock, detonation	Same as CEA2
New Capabilities	Baseline	Negative reactants, inert reactants, analytic derivatives for optimization and computational fluid dynamics (CFD) boundary conditions
Software Compliance	No formal compliance	Developed under NASA NPR 7150.2; version control on GitHub
Testing and Validation	Manual validation	Automated unit and integration tests; validated against 4,498 variables (agreement within 1%)
Performance on Modern Platforms	Limited portability	Cross-platform build support via CMake; hosted on GitHub
Distribution	Proprietary NASA software	Open-source on GitHub; listed in NASA Technology Transfer Catalog
Documentation	PDF user manual (RP-1311)	Online searchable documentation (nasa.github.io/cea)

Key takeaways are:

- **CEA 3.0 is a complete rewrite**, not just a translation, bringing modern software practices and interoperability.
- **APIs and thread safety** make CEA 3.0 dramatically faster for large-scale analyses.
- **Expanded database** supports green propellants and SAFs, aligning with sustainability goals.
- **Backward compatibility** is maintained via the legacy CLI, ensuring smooth transition for existing users.

The previous version of CEA, known as CEA2, was released in 2002. CEA includes methods to solve equilibrium, rocket, shock, and detonation problems. Of these, the equilibrium solver is the most fundamental in that it is called by the other problem types. The chemical equilibrium solution is based on the minimization of the Gibbs free energy of a multicomponent chemical mixture, where the Gibbs free energy is defined as

$$g = \sum_j^{NS} \mu_j n_j \quad \text{(Eq. 1)}$$

where NS is the number of species in the product mixture, μ_j is the chemical potential per kilogram-mole of the j-th species, and n_j is the amount (kilogram-moles per kilogram of mixture) of the species with index j. Importantly, this relies on the basic assumption of ideal gas, defined as

$$PV = n_g RT$$

where P, V, and T are the pressure, volume, and temperature of the mixture, respectively; R is the universal gas constant; and n_g is the number of gaseous moles in the mixture. The equilibrium solution permits condensed species to be included but assumes they have zero volume. Additionally, the minimization of free energy is done under the constraint that the amount of each element in the equilibrium product mixture is the same as in the initial reactant mixture, expressed as

$$\sum_j^{NS} a_{ij} n_j - b_i^\circ = 0, \; i = 1, \ldots, NE \quad \text{(Eq. 2)}$$

where $a_{ij}$is the stoichiometric coefficient matrix, which expresses the count of each element in each species such that the product $a_{ij}n_j = b_i$ (where b_i is the amount of element i of the product mixture); and b_i° is the initial amount of element i in the reactant mixture.

The solution of this constrained minimization problem starts by defining a Lagrangian expression

$$\mathcal{L} = g + \sum_i \lambda_i (b_i - b_i^\circ) \quad \text{(Eq. 3)}$$

and finding the stationary point by solving $\delta\mathcal{L} = 0$. The resulting expressions are derived from these principles and can be found in the original reference text [ref. 1].

The solution of a chemical equilibrium problem requires two fixed thermodynamic states to be defined by the user: temperature and pressure (TP), enthalpy and pressure (HP), entropy and pressure (SP), temperature and volume (TV), energy and volume (UV), or entropy and volume (SV). The equilibrium problem also takes as input the list of reactant names, the relative amount of each reactant, and optionally, each reactant's temperature. For HP, SP, UV, or SV problems, the reactant amounts and temperatures can be used by the program to compute the fixed value of enthalpy, energy, or entropy, because those are typically not known by the user in advance. The equilibrium problem computes the temperature of the product mixture, the amount of each product species, and the total moles of the resulting mixture. Based on these resulting state variables, the program outputs other thermodynamic states of interest, as well as transport properties if they are requested by the user.

These calculations require thermodynamic data for all species included in the mixture. To standardize input, CEA currently uses a nine-constant representation of the thermodynamic data. In this representation, $C_p^\circ(T)/R$ (heat capacity) is expressed as a seven-coefficient power series in T (in Kelvin (K)), with integration constants b_1 for $H°(T)/RT$ (enthalpy) and b_2 for $S^\circ(T)/R$ (entropy). The coefficients are used in Equations (1), (2), and (3) as

$$\frac{C_p^\circ}{R} = a_1 T^{-2} + a_2 T^{-1} + a_3 + a_4 T + a_5 T^2 + a_6 T^3 + a_7 T^4 \quad \text{(Eq. 4)}$$

$$\frac{H^\circ}{RT} = -a_1 T^{-2} + a_2 T^{-1} + a_3 + a_4 \frac{T}{2} + a_5 \frac{T^2}{3} + a_6 \frac{T^3}{4} + a_7 \frac{T^4}{5} + \frac{a_8}{T} \quad \text{(Eq. 5)}$$

$$\frac{S^\circ}{R} = -a_1 \frac{T^{-2}}{2} - a_2 T^{-1} + a_3 \ln T + a_4 T + a_5 \frac{T^2}{2} + a_6 \frac{T^3}{3} + a_7 \frac{T^4}{4} + a_9 \quad \text{(Eq. 6)}$$

This nine-constant form was adopted in 1994 and replaces an earlier seven-constant form [ref. 10]. Additional details about the method used by CEA can be found in Gordon and McBride [ref. 11].

GRC maintains a database with nine-constant empirical coefficients for over 2000 species. These coefficients were generated from least-squares curve fits to measured or calculated thermodynamic functions for condensed and gas-phase species [ref. 10]. The database is continually updated to reflect new species, improved measurements for current species, and newer physical constants.

The CEA development history underscores the importance of continuous improvement and adaptation to meet the evolving demands of aerospace technology. Its legacy of reliability and precision in chemical equilibrium calculations stands as a testament to NASA's commitment to advancing the frontiers of science and engineering.

7.0 Analysis

The objective of the current effort was to rewrite the CEA code in a modern, more serviceable computer language and place the program under configuration control. Fortran 2008 was selected as the primary programming language for the rewrite, preserving the original Fortran language of CEA2, maintaining fast execution speed, and adding object-oriented structures and other modern programming improvements. Version control is being maintained using best-practices on GitHub, and the program supports cross-platform build support using CMake[1]. Additionally, continuous testing was used during the software development cycle through unit testing. Integration testing was also implemented, testing a range of full example cases against CEA2 for all output values. This ensured not only consistency at the time of release but prevents future code modifications from having unintended consequences. The CEA thermodynamic property databases were reviewed and updated as needed; these changes are summarized in Section 7.2. A meeting that included existing users across government and industry was held to solicit suggestions for improving CEA. Finally, new SMEs were identified and trained to provide ongoing support for this critical capability.

The CEA governing equations and solution methodology are described by Gordon and McBride [ref. 1]. Some of the equations defining the equilibrium conditions are nonlinear and therefore require an iterative procedure to converge the system. CEA 3.0 uses the same Newton method and solution procedure used in CEA2 to iterate to convergence. In summary, this effort provides a modernized, validated, and backward-compatible rewrite of CEA2. The primary benefits of this effort can be summarized as follows:

- **APIs in multiple programming languages:** The Fortran source code was structured to use a subroutine interface, which enabled APIs for Fortran, C, Python, and MATLAB, with Excel planned as a priority update. This provides users with multiple ways to use the code, with the flexibility to select the programming language that best fits their existing toolchain. Providing APIs facilitates integration into other engineering tools—which is a common use-case for CEA—but this can also significantly reduce computational costs compared with CEA2. Speed is commonly cited by existing users as a bottleneck to using CEA, but this bottleneck only existed because of the lack of an API. Previously, as a workaround, users had to write input files, call the CEA executable, and parse output files each time CEA was used. This file I/O is extremely slow. By providing an API, the variables are passed through memory, resulting in much faster execution.
- **Modern software engineering best practices:** Hosting the code on GitHub allows both maintainers and users to track and compare code changes over time, and the maintainers will implement version control through release tags in the MAJOR.MINOR.PATCH format as the code is continually updated over time. Because CEA2 was not centrally stored in one place, over time there were multiple diverging versions of the code, making it difficult to compare results. In addition to maintaining the code on GitHub, cross-platform build support is implemented using CMake so the code can be easily installed on a diverse set of operating systems and compiler platforms. Finally, the use of software testing is another key to modern software development that has been implemented in a variety of ways. Unit tests were used extensively during the software rewrite so that results were “continuously” validated. These unit tests continue to be used any time the source code changes to ensure changes in one area

[1] https://cmake.org

do not have unintended consequences in another. Integration testing was also implemented, where the output of CEA 3.0 was compared against CEA2. This validates the output variables for each of the canonical examples in RP-1311 Part 2 [ref. 12], for a total of 4,498 variables.

- **Feature enhancements:** New features were added to the code based on feedback from the user community, including thread-safe solves (i.e., to allow the code to be run in parallel), negative reactant amounts (i.e., reactants being removed from the mixture), and inert reactants (i.e., those that are not allowed to react). Negative and inert reactants were unofficially included in certain copies of CEA2; these allowed users to approximate some nonequilibrium behaviors. An example of using negative reactants would be having water precipitate from a flow, and an example of using inert reactants would be having some fuel left uncombusted. These are nonequilibrium behaviors in an equilibrium code, so extreme caution is advised, but users have found these to be useful for specific purposes. Additionally, significant progress has been made in adding analytic derivatives to the equilibrium solver. This feature is not included in the initial release but is planned as a priority feature enhancement. Analytic derivatives for the code provide total derivatives of outputs (i.e., mixture temperature, species concentrations, and thermodynamic properties) with respect to inputs (i.e., reactant amounts and fixed thermodynamic properties corresponding to TP, HP, SP, TV, UV, or SV problem types). This enables gradient-based optimization, as well as being required for boundary conditions in CFD applications.
- **Thermodynamic library updates:** A select number of publicly available green propellants were added to the thermodynamic database as reactants so that users can model problems with green propellants and SAFs.

Each of these contributions is described in detail in the following sections.

7.1 Programing Language Updates

For efficiency and maintainability, it was necessary to implement CEA in a contemporary computer language. Fortran 2008 was selected as the language for source-code development. This provided consistent basic syntax with the legacy code, facilitating an easier transition while providing modern software paradigms (e.g., object-oriented typing) and maintaining consistent execution speed compared with CEA2. CEA2 operates exclusively through an input file interface, making it challenging to call it from other applications. For this reason, unlike CEA2, CEA 3.0 has emphasized the use of a subroutine interface in multiple programming languages. The Fortran 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, with Excel planned as a future update. 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 CEA 3.0 is shown in Figure 7.1-1. Each interface is 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, the analysis ultimately is solved in Fortran. This results in faster execution times for the interpreted languages (i.e., Python, Excel, and MATLAB).

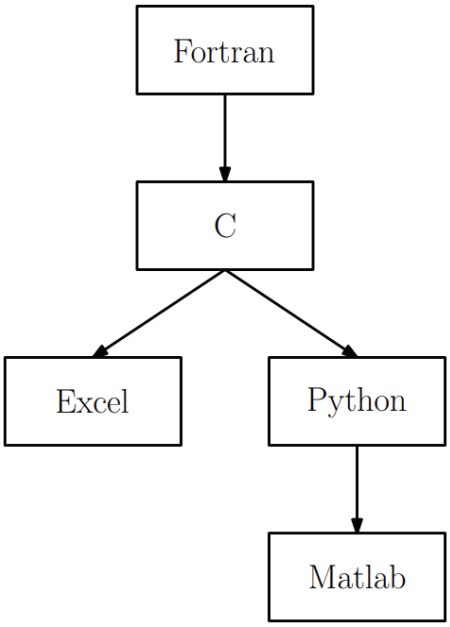


Figure 7.1-1. Hierarchy of Interface Language Calls

7.1.1 Fortran Interface

Fortran 2008 is used as the primary programming language for CEA 3.0. The program is broken into one module for each application (i.e., "equilibrium," "rocket," "shock," "detonation"), as well as other ancillary support modules. Each module contains several functions that make up the basis for the Fortran API.

The decision to use Fortran 2008 for CEA 3.0 was made due to its fast computational speed and similarity to the original language of CEA2. CEA 3.0 is a rewrite of CEA2, not simply a translation into a more modern version of Fortran. In doing this, the advantages of modern software development practices and the object-oriented nature of Fortran 2008 were used to improve the quality of the code for both developers and end users. The Fortran code was written in a thread-safe manner so users can run the code in parallel for even faster execution. Database initialization should be done on the root processor only, but then equilibrium or other problem solves can be run fully parallel since they are independent of one another.

7.1.2 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 Fortran's bind(c) attribute so they can 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 CEA 3.0. An example using the C interface is provided in Appendix C.

7.1.3 Python Interface

CEA 3.0 can also be called through Python. The Python wrapper imports the C-binding of CEA 3.0 through the Cython[2] package. To enable this, a definition file (.pxd) is created to declare the function calls and data types in the C interface. That definition file is then imported

[2] https://cython.org

into a Python wrapper file (.pyx extension). This allows a CEA 3.0 Python package to be written with an API that is more Pythonic while using the function-based C binding and maintaining the speed of the underlying Fortran solver. An example of the Python interface is provided in Appendix D.

Having access to Python makes plotting results more convenient as well. Below is the result of an example with plots created using the `matplotlib`[3] Python package. This example is an HP equilibrium problem run with varying initial temperatures and reactant equivalence ratios. Figure 7.1.3-1 shows contours of the final mixture temperature, and Figure 7.1.3-2 shows species concentrations of the equilibrium mixture. The reactants are H_2 and O_2, the initial mixture temperature varies between 300 and 2000 K, and the equivalence ratio ranges from 0.5 to 2.0. This example highlights the speed of the code, solving 108,500 equilibrium problems in 1.11 seconds on a desktop computer.

Figure 7.1.3-1. Contour Plot of Mixture Temperature for HP Problem with H_2 and O_2 Reactants, varying Initial Temperature and Equivalence Ratio

[3] https://matplotlib.org

Figure 7.1.3-2. Contour Plots of Product Species Concentrations for HP Problem with H_2 and O_2 Reactants, Varying Initial Temperature and Equivalence Ratio

7.1.4 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. For that reason, an additional function-based interface option has been added to the Python wrapper.

7.1.5 Excel Interface

The Excel interface was planned but delayed due to schedule constraints. It remains a priority and will be added as part of the first major revision of the software.

7.1.6 Legacy Command Line Interface

To support backward compatibility, CEA 3.0 supports the legacy text-based interface of CEA2. To use this interface, the problem details are specified in a text file with an .inp suffix. Then, the compiled program is called through a command-line interface using:

```
./cea <input_file_name>
```

where `<input_file_name>` is the name of the file without the .inp suffix. A result file with the same name and the extension .out is generated in the same folder.

The legacy interface is interpreted through the Fortran script named `main.f90`, which parses the .inp file, calls the required functions and subroutines in the Fortran API, and writes out the formatted .out file.

7.2 Thermochemical and Transport Database Updates

As part of this modernization effort, the enthalpies of formation, $\Delta H^\circ(T,P)$, of the following preexisting chemical species and SAFs from CEA2 were carefully reviewed and compared against those found in references 13 through 15:

Preexisting chemical species of interest:

- Nitric oxide, gaseous phase: NO (g)
- Dinitrogen tetroxide, gaseous phase: N_2O_4 (g)
- Water, gaseous and liquid phases: H_2O (g) and H_2O (l)
- Methane, gaseous phase: CH_4 (g)

Preexisting SAF of interest:

- Biodiesel: ethanol, gaseous and liquid phases: C_2H_5OH (g) and C_2H_5OH (l)

The enthalpies of formation, ΔH° (evaluated at $T = 298.15$ K, $P = 1.0$ bar, and at $T = 0.0$ K, $P = 0.0$ bar), of the aforementioned preexisting chemical species and SAF from CEA2, were found to be in excellent agreement with those in references 13 through 15, as summarized in Tables A-1 and A-2 in Appendix A. Consequently, it was determined that updating the enthalpies of formation of these species was unwarranted. Additionally, the reactants segment of the CEA thermodynamic database was appended to include the standard enthalpies of formation for the following non-sensitive green propellants and SAFs:

- Newly added non-sensitive green propellants:
  - Ammonium dinitramide (ADN): $H_4N_4O_4$
  - LMP-103S: 63% ADN ($N(NO_2)_2^-$ and NH_4^+), 14% H_2O, 18.4% methanol (CH_3OH), and 4.6% ammonia (NH_3). Sum formula: $CH_{11.66}N_{4.01}O_{5.89}$
  - Hydroxylammonium nitrate (HAN): $H_4N_2O_4$
- Newly added non-sensitive SAFs:
  - n-Butanol, gaseous and liquid phases: $C_4H_{10}O$ (g) and $C_4H_{10}O$ (l)

The addition of the green propellants and SAFs to the CEA thermo-chemical database expands the set of possible reactants that can be used for chemical equilibrium calculations.

7.3 Capability Improvements

CEA 3.0 includes new features that were requested by the existing user community. These features include negative reactants, inert reactants, and analytic derivatives (which are still under development). This section highlights special cases (e.g., condensed species) and problems (e.g., ionized products) that existed previously with CEA2.

7.3.1 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,” 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"[4] variable, the initial guess for the condensed species concentrations is assumed to be zero. CEA first converges a solution with only gas-phase species. At that point, CEA performs a test to determine whether any condensed species should be added. This is done by checking whether adding any would decrease the total Gibbs free energy of the mixture. If any meet this criterion, then the condensed species that decreases the Gibbs free energy the most is added to the mixture, and the iteration procedure is resumed. This cycle is repeated until convergence. Special considerations are given for phase transitions and for allowing multiple phases of the same species; these details are given in Gordon and McBride [ref. 1].

7.3.2 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. 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 details are given in Gordon and McBride [ref. 1].

7.3.3 Negative Reactants

CEA 3.0 allows reactant amounts to be specified as negative values, which can be useful in scenarios such as water precipitating from a flow. This is a non-equilibrium analysis, while CEA is, by definition, an equilibrium solver, so any results using this feature 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. First, the element constraint value is subtracted by the appropriate amount. The user should ensure 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. Second, if the case uses fuel and oxidant temperatures to compute the constraint value (e.g., HP, SP, UV, or SV problems where $H°/S°/U°$ are not specified by the user), then CEA will subtract the appropriate amount when computing the constraint value.

7.3.4 Inert Reactants

Inert reactants can be useful in certain cases (e.g., incomplete combustion) where a user may wish to have part of the fuel react and another portion not react. 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 Appendix B. When an inert element is used, internally in CEA a separate version of each element is used (e.g., "IC" instead of "C"). This causes inert elements to have their own element-amount constraints and not react with the active versions of the species.

[4] "Insert" is a legacy CEA keyword to include a species in the initial guess for the solution.

7.3.5 Analytic Total Derivatives

Analytic total derivatives were added to CEA, providing derivatives of outputs (i.e., product species concentrations, mixture temperature, and bulk thermodynamic properties) with respect to inputs (i.e., reactant amounts and fixed thermodynamic state inputs) corresponding to problem type (e.g., enthalpy and pressure for an HP problem). They are analytic in that they are defined analytically as opposed to through finite-difference, automatic differentiation, or complex-step methods. The choice of analytic derivatives is preferrable to finite difference for both accuracy and computational cost and preferrable to complex step in terms of computational cost, while the tradeoff against using automatic differentiation came down to ease of implementation for this codebase. Providing analytic total derivatives enables CEA to be used in optimization problems, making it even more appealing to integration with other codes focused on multidisciplinary design optimization (MDO) applications. Analytic gradients also provide additional benefits, including calculations for boundary conditions in CFD applications.

7.4 SME Support

Maintenance and user-community support for the CEA code will remain under the Propulsion Systems Analysis Branch at GRC. The project's GitHub page (https://github.com/nasa/cea) will be regularly monitored for issues and feature requests and will remain the preferred channel to request support.

7.5 Software Testing Approach

Unit testing was implemented concurrently with the software development process so that the new code was written to match the expected results from CEA2. Following initial completion of the code, validation tests were implemented to compare the results obtained using CEA 3.0 with those obtained using CEA2.

Each example in RP-1311 [ref. 12] was run in CEA2 and the output files stored for testing. Next, a Python script ran through each of the example problem input files, calling the CEA 3.0 legacy CLI, which generated output text files in the format of CEA2. At that point, the Python script read in the validation output files and the newly generated values from CEA 3.0 and stored them in a dictionary format. These values were then looped over and compared individually to ensure consistency across all output values, including product mixture species concentrations, bulk thermodynamic and transport properties (if applicable), and all other problem-specific variables including rocket, shock, and detonation parameters. This resulted in a total of 4,498 variables being tested, all of which agreed within 1%, with one exception due simply to rounding output (i.e., in Example 11, the concentration of F– is 0.00025 in CEA2 and 0.00026 in CEA 3.0). Additionally, all but nine variables agreed within 0.1% relative error *or* 1e-6 absolute error, and all but 74 agreed within 0.01% *or* 1e-6 absolute error[5]. These errors generally seem attributable to output formatting; because of the nature of CEA2, one can only compare as many digits as are output by CEA2, so in cases where CEA 3.0 outputs more digits, the extra digits are interpreted as error. For example, one of the nine terms failing the 0.1% threshold is a pressure value in Example 12 [ref. 11], which is 0.0212 in CEA2 and 0.02123 in CEA 3.0. The absolute difference of 0.0003 results in a relative error of 0.14%, even though the CEA 3.0 value would round to the

[5] Including the absolute error filter reduces terms that are already small in magnitude (e.g., the mole fraction of CN in Example 3 [ref. 11] is 1.066e-13 in CEA2 versus 1.067e-13 in CEA 3.0; the absolute error is 4e-16, but the relative error is 0.375%).

CEA2 value exactly. The lack of output precision from CEA2 makes it difficult to determine the true error between the codes; however, the consensus across multiple thresholds for a range of problem types indicates good agreement.

Additional unit testing is implemented at the Python API level using the pytest[6] library to ensure consistency and non-breaking changes when interface updates are made. In total, the use of routine unit testing at the Fortran and Python API levels, as well as automated validation testing of the legacy CLI, provide comprehensive testing coverage of the software, and more tests will continue to be added over time as needed.

7.6 Software Timing Comparison

Computational cost is a critical factor for CEA users, and a requirement of this update is that CEA 3.0 "be within a factor of 2" in runtime. In other words, it is acceptable for CEA 3.0 to be slower than CEA2, but no more than twice as slow. To confirm that CEA 3.0 is comparable to CEA2 in terms of computational cost, the NESC team ran the same case (Example #1 from RP-1311 Part 2 [ref. 2]) 20 times from the command line, timed the output using the GNU program "time"[7], and calculated the mean and standard deviation of the runs. The mean wall time for CEA2 was 0.01915 seconds with a standard deviation of 0.0097 seconds, while the mean wall time for CEA 3.0 was 0.02685 seconds with a standard deviation of 0.012 seconds. These results indicated that CEA 3.0 is 40.21% slower than CEA2, which satisfies the initial requirement. The reason for the slower computational cost is likely the added overhead of allocatable arrays and object-oriented typing. This example shows a direct comparison for a single input problem using the command line interface for both programs. However, CEA2 is often embedded in other programs and called repeatedly. This results in slower runtimes due to writing to and reading from file, which is the only way for other programs to interface with CEA2. To compare a case representative of looping over problem inputs, the team timed the example problem given in Section 7.1.3, which loops over input parameters and solves an HP problem 108,500 times. These timing comparisons were done on a 2023 MacBook Pro. Using CEA2, writing a new input file, running CEA2, and parsing the output file each time took 15 minutes and 3.964 seconds. Using the Python API in CEA 3.0, the same operation took 1.109 seconds. Neither of these times included the time used to generate plots. The difference in runtime is due solely to the lack of an API for CEA2, but this is how many existing users interact with the code today, so this comparison is representative of what many users will experience in their workflows involving CEA.

7.7 Software Compliance

The compliance matrix for Class D software was completed and successfully reviewed as part of the software development plan for CEA 3.0. As a part of the software assurance process, the NESC assessment team created a Fortran Coding Standards document. This document is available for future developers to ensure the code style, formatting, and standards are consistent across the project.

[6] https://docs.pytest.org/en/stable/#

[7] https://www.gnu.org/software/time/

8.0 Validation Testing Examples

This section presents results for a limited number of validation testing examples by comparing output from CEA 3.0 with results from CEA2. The included results are limited to one example per problem type, but the full testing scope of the software is more comprehensive. For each example, the problem setup is described and the results are discussed, to include a tabular comparison for a descriptive subset of outputs. Note that in these examples, relative error is computed as $\mathrm{abs}((CEA2 - CEA\ 3.0)/CEA2)$.

8.1 Equilibrium Example

This demonstration of the equilibrium application uses Example 4 from McBride et al. [ref. 12]. This is a UV problem with ρ = 14.428 kg/m^3 and u/R = −45.1343 (kg*mol)(K)/kg. The oxidant-to-fuel ratio is 17. Air was used as the only oxidant, with a temperature of 700 K. The fuel is a mixture of C_7H_8 (l) and C_8H_{19} (l) with a 40/60 weight fraction of each and a temperature of 298.15 K. The results for this problem are given in Table 8.1-1, which provides temperature and a subset of species concentrations for comparison. The log of the species concentration is provided because this is what CEA uses internally as state variables in the solver, which allows comparison of the trace species before they are truncated. This example shows that CEA 3.0 has a relative error on the order of 10^{-9} or better for each value shown, indicating very good agreement for the equilibrium solver.

Table 8.1-1. Equilibrium Example of a UV Problem

	CEA2	CEA 3.0	Relative Error
T (K)	**2418.53825**7447409	**2418.53825**58358501	6.66×10^{-10}
ln(nj): $(HCOOH)^2$	**–52.3748901**3325020	**–52.3748901**59103195	4.94×10^{-10}
ln(nj): Ar	**–8.0940265**64066302	**–8.0940265**628475956	1.51×10^{-10}
ln(nj): C	**–42.9570511**4743870	**–42.9570511**92012905	1.04×10^{-9}
ln(nj): CO	**–9.759156**120427597	**–9.759156**1333187880	1.32×10^{-9}

8.2 Rocket Example

This CEA rocket-application problem comes from Example 8 in McBride et al. [ref. 12]. This rocket problem uses an infinite-area combustor (IAC), with H_2 (l) as the fuel at 20.27 K, O_2 (l) as the oxidant at 90.17 K, and an oxidant-to-fuel ratio of 5.55157. Each variable shows a relative error on the order of 10^{-8} or better, indicating good agreement. The results are given in Table 8.2-1.

Table 8.2-1. Rocket Problem Example Using an Infinite-area Combustor

Station	Variable	CEA2	CEA 3.0	Relative Error
Combustor	T (K)	**3383.8446259451043**	**3383.8446259451043**	0.0
Combustor	ln(nj): H_2	**–3.7643107135196**541	**–3.7643107135196**558	4.72×10^{-16}
Throat	T	**3185.6731**730807096	**3185.6731**649632438	2.55×10^{-9}
Throat	ln(nj): H_2	**–3.7758918668**469459	**–3.7758918666**757006	4.54×10^{-11}
Exit (p_i/p_e)	T	**2567.340**1804441942	**2567.340**2122191874	1.24×10^{-8}
Exit (p_i/p_e)	ln(nj): H_2	**–3.7880413204**534467	**–3.7880413206**896888	6.24×10^{-11}

8.3 Shock Example

Table 8.3-1 gives values for an example shock problem using CEA2 and CEA 3.0. The values provided for comparison include temperature, pressure, and the log of the concentrations for a subset of species for both the incident and reflected shocks. The example comes from Example 7 in McBride et al. [ref. 12], 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, a pressure of 0.1 bar and an initial velocity of 1,400 m/s were used. The values computed using CEA 3.0 differ from the CEA2 solution on the order of between 10^{-8} and 10^{-10}, indicating good agreement between the two programs for shock tube problems.

Table 8.3-1. Shock Tube Problem Example Results

Type	Variable	CEA2	CEA 3.0	Relative Error
Incident	T (K)	**2268**.1839840465000	**2268**.1839777687906	2.77×10^{-9}
Incident	P (bar)	**1.923**5852279853596	**1.923**5852357512877	4.04×10^{-9}
Incident	ln(nj): HO_2	**–17.22**3079678243373	**–17.22**3079682138113	2.26×10^{-10}
Incident	ln(nj): OH	**–9.625**9080229365601	**–9.625**9080372154173	1.48×10^{-9}
Reflected	T (K)	**325**3.2881222920269	**325**4.2881272510235	1.52×10^{-9}
Reflected	P (bar)	**6.88**57588009600743	**6.88**57588828609000	1.19×10^{-8}
Reflected	ln(nj): HO_2	**–15.14**9034932828400	**–15.14**9034925313257	4.96×10^{-10}
Reflected	ln(nj): OH	**–7.68**11535944113283	**–7.68**11535907803705	4.73×10^{-10}

8.4 Detonation Example

For the validation example of the Chapman–Jouguet detonation problem, results based on Example 6 from McBride et al. [ref. 12] were compared. This example is based on a detonation with H_2 and O_2 as the fuel and oxidant, with an equivalence ratio $r = 1$, the unburned gas at 1 bar, and a temperature of 298.15 K. The results are provided in Table 8.4-1, comparing temperature, pressure, and the log of concentrations of a subset of species for CEA2 and

CEA 3.0. Each value provided shows good agreement, with relative error on the order of between 10^{-11} and 10^{-12} for each value.

Table 8.4-1. Chapman–Jouguet Detonation Problem Example Results

	CEA2	CEA 3.0	Relative Error
T (K)	**3674.2**808311051022	**3674.2**808310655046	1.08×10^{-11}
P (bar)	**18.7684**46693571065	**18.7684**46693128148	2.36×10^{-11}
$\ln(nj)$: H_2	**–4.49361**23449492360	**–4.49361**23449898409	9.04×10^{-12}
$\ln(nj)$: H_2O_2	**–13.491896**007609208	**–13.491896**007645531	2.69×10^{-12}
$\ln(nj)$: OH	**–4.630083**6293723897	**–4.630083**6294210113	1.05×10^{-11}

9.0 Conclusion

This work describes the updates to the NASA CEA software program, CEA 3.0. Like CEA2, CEA 3.0 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 CEA 3.0 that were not previously available, including negative and inert reactants and additions to the thermodynamic database, to include green propellants. CEA 3.0 supports the legacy input and output file format but emphasizes the use of a subroutine interface, and APIs now are available in several programming languages, including Fortran, C, Python, and MATLAB, with Excel as a planned addition. Finally, consistent results with CEA2 have been demonstrated by providing validation examples for each application in CEA.

The updated CEA code and suite of validation test cases were demonstrated and have been placed under configuration control using GitHub. The new SMEs at GRC will manage releases of future versions. The updated CEA version has been added to the Technology Transfer Office Software Catalog.

10.0 Findings

F-1. Through validation testing, with a single exception, CEA 3.0 matched the results of CEA2 for 4,498 unique variables within 1%, and the exception was the result of a rounding difference caused by output formatting.

F-2. Computational speed testing found that CEA 3.0 runs 40.21% slower than CEA2 for a single example run through the command line. This is based on running each program 20 times and taking the average of each, with CEA2 taking 0.01915 seconds and CEA 3.0 taking 0.02685 seconds. However, due to the addition of an API in 2022, running a parameter sweep of 108,500 cases takes 15 minutes and 3.96 seconds in CEA2, while taking just 1.11 seconds in CEA 3.0.

11.0 Alternate Technical Opinion(s)

No alternate technical opinions were identified during the course of this assessment by the NESC assessment team or the NESC Review Board (NRB).

12.0 Other Deliverables

The updated CEA analysis will be made available to all NASA Centers, including Headquarters, programs and projects, other government agencies, industry, and academia. The code has been made publicly available open source at github.com/nasa/cea. The updated CEA code will be entered into the NASA Technology Transfer Office Software Catalog. This NESC final report, a NASA TM, and an NESC TB were delivered. A user guide for the code is available in the form of searchable, online documentation at nasa.github.io/cea.

Mark Leader (mark.leader@nasa.gov) is the SME trained to maintain CEA and support the user community. As personnel change, the individual may change, but the position will remain under Propulsion Systems and Technologies at GRC (216-433-4000).

13.0 Recommendations for the NASA Lessons Learned Database

No recommendations for NASA lessons learned were identified as a result of this assessment.

14.0 Recommendations for NASA Standards, Specifications, Handbooks, and Procedures

No recommendations for NASA standards, specifications, or procedures were identified as a result of this assessment.

15.0 Definition of Terms

Finding	A relevant factual conclusion and/or issue that is within the assessment scope and that the team has rigorously based on data from their independent analyses, tests, inspections, and/or reviews of technical documentation.
Observation	A noteworthy fact, issue, and/or risk, which is not directly within the assessment scope, but could generate a separate issue or concern if not addressed. Alternatively, an observation can be a positive acknowledgement of a Center/Program/Project/Organization's operational structure, tools, and/or support.
Recommendation	A proposed measurable stakeholder action directly supported by specific Finding(s) and/or Observation(s) that will correct or mitigate an identified issue or risk.

16.0 Acronyms and Nomenclature List

ADN	Ammonium Dinitramide
AFRL	Air Force Research Laboratory
ANL	Argonne National Laboratory
API	Application Programming Interface
Ar	Argon
ARC	Ames Research Center
C	Carbon

CEA	Chemical Equilibrium with Applications
CFD	Computational Fluid Dynamics
CLI	Command Line Interface
EDL	Entry, Descent, and Landing
GRC	Glenn Research Center
GSFC	Goddard Space Flight Center
H	Hydrogen
HAN	Hydroxylammonium Nitrate
HP	Enthalpy and Pressure (problem type)
HTP	Hypersonic Technology Project
IAC	Infinite-area Combustor
I/O	Input/output
K	Kelvin
m	meter
MDO	Multidisciplinary Design Optimization
MSFC	Marshall Space Flight Center
N	Nitrogen
NESC	NASA Engineering and Safety Center
NIST	National Institute for Standards and Technology
NPR	NASA Procedural Requirement
O	Oxygen
RVLT	Revolutionary Vertical Lift Technology
s	second
SAF	Sustainable Aviation Fuel
SME	Subject Matter Expert
SP	Entropy and Volume (problem type)
SV	Entropy and Volume (problem type)
TB	Technical Bulletin
TM	Technical Memorandum
TP	Temperature and Pressure (problem type)
TTT	Transformational Tools and Technologies
TV	Temperature and Volume (problem type)
UV	Energy and Volume (problem type)
WSTF	White Sands Test Facility

17.0 References

1. Gordon, S. and McBride, B. J., "Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications: I. Analysis," NASA Reference Publication 1311, October 1994. URL: https://ntrs.nasa.gov/citations/19950013764
2. Brinkley, S. R., "Calculation of the Equilibrium Composition of Systems of Many Constituents," *Journal of Chemical Physics*, Vol. 15, pp. 107-110, 1947. URL: https://doi.org/10.1063/1.1746420
3. McBride, B. J. and Gordon, S., "FORTRAN IV Program for Calculation of Thermodynamic Data," NASA TN-4097, 1967.
4. Gordon, S. and McBride, B. J., "Computer Program for Calculation of Complex Chemical Equilibrium Compositions, Rocket Performance, Incident and Reflected Shocks, and Chapman-Jouguet Detonations," NASA SP-273, 1971.
5. Gordon, S. and McBride, B. J., "Computer Program for Calculation of Complex Chemical Equilibrium Compositions, Rocket Performance, Incident and Reflected Shocks, and Chapman-Jouguet Detonations," NASA SP-273, Interim Revision, 1976.
6. Gordon, S., "Thermodynamic and Transport Combustion Properties of Hydrocarbons with Air, I- Properties in SI Units," NASA TP-1906, 1982.
7. Gordon, S., McBride, B. J., and Zeleznik, F. J., "Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications, Supplement I-Transport Properties," NASA TM-86885, 1984.
8. Gordon, S. and McBride, B.J., "Finite Area Combustor Theoretical Rocket Performance," NASA TM-100785, 1988.
9. McBride, B. J., Gordon, S., and Reno, M. A., "Coefficients for Calculating Thermodynamic and Transport Properties of Individual Species," NASA TM-4513, 1993.
10. Leader, M. K., Lavelle, T. M., Wang, X. J., Dickens, K. W., McTague, M., and Hill, J. P., "CEA2022: Modernization of NASA Glenn's Software CEA (Chemical Equilibrium with Applications)," Thermal Fluids Analysis Workshop, Cleveland, Ohio, August 29, 2024. URL: https://ntrs.nasa.gov/api/citations/20240009728/downloads/TFAWS_2024_CEA.pdf
11. Gordon, S. and McBride, B. J., "Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications – II. User's Manual and Program Description," NASA Reference Publication 1311, 1996. URL: https://ntrs.nasa.gov/citations/19960044559
12. McBride, B. J., Zehe, M. J., and Gordan, S., "NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species," NASA/TP-2002-211556, September 2002.
13. Ruscic, B. and Bross, D. H., "Active Thermochemical Tables (ATcT) Values Based on Version 1.130 of the Thermochemical Network," Argonne National Laboratory, Lemont, Illinois 2023. URL: https://doi.org/10.17038/CSE/1997229
14. NIST Computational Chemistry Comparison and Benchmark Database, "NIST Standard Reference Database Number 101," Release 22, Editor: Russell D. Johnson III, May 2022. URL: https://doi.org/10.18434/T47C7Z.

15. Östmark, H., et al., “The Properties of Ammonium Dinitramide (ADN): Part 1, Basic Properties and Spectroscopic Data,” *Journal of Energetic Materials*, Vol. 18, No. 2-3, pp. 123-138, 2000. URL: https://doi.org/10.1080/07370650008216116.

16. Wingborg, N., “Heat of Formation of ADN-Based Liquid Monopropellants,” *Propellants, Explosives, Pyrotechnics*, Vol. 44, pp. 1090-1095, 2019.

17. Kounalakis, M. E. and Faeth, G. M., “Combustion of HAN-Based Liquid Monopropellants Near the Thermodynamic Critical Point,” *Combustion and Flame*, Vol. 74, pp. 184, 1988.

Appendix A. Thermodynamic Data for Green Propellants and Sustainable Aviation Fuels

Table A-1. Enthalpies of Formation, Evaluated at T = 0.0 K and P = 0.0 bar, for Nitric Oxide, Dinitrogen Tetroxide, Water Vapor, Methane, and Biodiesel

Species Name	Chemical Formula	CEA2 $\Delta_f H$ (T = 0.0 K, P = 0.0 bar) Values [kJ*mol^{-1}][12]	ATcT $\Delta_f H$ (T = 0.0 K, P = 0.0 bar) Values [kJ*mol^{-1}][13]	Calculated Relative Differences between CEA2 and ATcT $\Delta_f H$ (T = 0.0 K, P = 0.0 bar) Values [%]	NIST CCCBDB $\Delta_f H$ (T = 0.0 K, P = 0.0 bar) Values [kJ*mol^{-1}][14]	Calculated Relative Differences between CEA2 and NIST CCCBDB $\Delta_f H$ (T = 0.0 K, P = 0.0 bar) Values [%]
Gaseous nitric oxide	NO (g)	90.767	90.639 ± 0.066	0.14102%	90.54 ± 0.08	0.2501%
Gaseous dinitrogen tetraoxide	N_2O_4 (g)	20.400	20.19 ± 0.14	1.029%	20.40 ± 1.00	0.0000%
Gaseous water (i.e., water vapor or steam)	H_2O (g)	–238.922	–238.902 ± 0.022	-8.3709×10^{-2}%	–238.90 ± 0.03	9.2080×10^{-3}%
Gaseous methane	CH_4 (g)	–66.626	–66.549 ± 0.044	–0.11557%	–66.63 ± 0.30	-6.0037×10^{-3}%
Gaseous biodiesel, ethanol	C_2H_5OH (g)	–217.641	–217.31 ± 0.20	–0.15209%	–217.08 ± 0.50	0.25776%
Liquid biodiesel, ethanol	C_2H_5OH (l)	–269.741	–269.74 ± 0.20	-3.7073×10^{-4}%	Unavailable	N/A

Table A-2. Standard Enthalpies of Formation, ΔfH°(T, P), Evaluated at T = 298.15 K and P = 1.0 bar, for Nitric Oxide, Dinitrogen Tetroxide, Water, Methane, and Biodiesel

Species Name	Chemical Formula	CEA2 $\Delta_f H°$ (T = 298.15 K, P = 1.0 bar) Values [kJ*mol^{-1}][12]	ATcT $\Delta_f H°$ (T = 298.15 K, P = 1.0 bar) Values [kJ*mol^{-1}][13]	Calculated Relative Differences between CEA2 and ATcT $\Delta_f H°$ (T = 298.15 K, P = 1.0 bar) Values [%]	NIST CCCBDB $\Delta_f H°$ (T = 298.15 K, P = 1.0 bar) Values [kJ*mol^{-1}][14]	Calculated Relative Differences between CEA2 and NIST CCCBDB $\Delta_f H°$ (T = 298.15 K, P = 1.0 bar) Values [%]
Gaseous nitric oxide	NO (g)	91.271310	91.143 ± 0.066	0.14100%	91.04 ± 0.08	0.2534%
Gaseous dinitrogen tetroxide	N_2O_4 (g)	11.110919	10.90 ± 0.14	1.898%	11.11 ± 1.00	8.271×10^{-3}%
Gaseous water (i.e., water vapor or steam)	H_2O (g)	–241.826000	–241.805 ± 0.022	9.0000×10^{-2}%	–241.81 ± 0.03	6.6163×10^{-3}%
Liquid water	H_2O (l)	–285.830000	–285.800 ± 0.022	1.0000×10^{-2}%	Unavailable	N/A
Gaseous methane	CH_4 (g)	–74.600000	–74.518 ± 0.044	0.11000%	–74.60 ± 0.30	0.00000%
Gaseous biodiesel, ethanol	C_2H_5OH (g)	–234.950000	–235.03 ± 0.20	-3.4050×10^{-2}%	–234.80 ± 0.50	6.384×10^{-2}%
Liquid biodiesel, ethanol	C_2H_5OH (l)	–277.510000	–277.50 ± 0.20	-3.6035×10^{-3}%	Unavailable	N/A

Table A-3. Enthalpies of Formation, ΔfH°(T, P), Evaluated at T = 0.0 K and P = 0.0 bar, for n-Butanol [ref. 14]

Species Name	Chemical Formula	CEA 3.0's $\Delta_f H$(T = 0.0 K, P = 0.0 bar) Values [kJ*mol^{-1}]
Gaseous n-butanol	$C_4H_{10}O$ (g)	–223.74 ± 0.60[ref. 13]
Liquid n-butanol	$C_4H_{10}O$ (l)	–266.21 ± 0.60[ref. 13]

Table A-4. Standard Enthalpies of Formation, ΔfH°(T, P), Evaluated at T = 298.15 K and P = 1.0 bar, for Green Propellants ADN, LMP-103S, and HAN, and for SAF n-Butanol

Species Name	Chemical Formula	CEA 3.0 $\Delta_f H°$ (T = 298.15 K, P = 1.0 bar) Values [$kJ*mol^{-1}$]
Ammonium dinitramide (ADN)	$H_4N_4O_4$ [ref. 15]	148 ± 10 [ref. 15]
LMP-103S	63% ADN (i.e., $N(NO_2)_{2-}$ and NH_{4+}), 14% water (i.e., H_2O), 18.4% methanol (i.e., CH_3OH), and 4.6% ammonia (i.e., NH_3) (empirical formula: $CH_{11.66}N_{4.01}O_{5.89}$) [ref. 16]	–766.6 [ref. 16]
Hydroxylammonium nitrate (HAN)	$H_4N_2O_4$ [ref. 7]	–398.7352 [ref. 17]
Gaseous n-butanol	$C_4H_{10}O$ (g)	–251.16±0.60 [ref. 13]
Liquid n-butanol	$C_4H_{10}O$ (l)	–278.53±0.60 [ref. 13]

Appendix B. Included Inert Reactants

The list of included inert species is listed below in Table B-1. Inerted elements include only hydrogen, oxygen, and nitrogen. The name formats given in the table are how they are listed in the thermodynamic database and, therefore, how they should be called by the user.

Table B-1. List of Included Inert Reactants

InertCH4	InertC2H4	InertC10H8,naph	InertH	InertH2
InertO	InertO2	InertAir	InertH2(L)	InertJP-4
InertJP-5	InertJP-7	InertJP-10(L)	InertJP-10(g)	InertJet-A(L)
InertJet-A(g)	InertO2(L)	InertRP-1		

Appendix C. Sample Code using the C Interface

Sample code using the C interface is shown here.

```
#include "stdio.h"
#include "cea.h"

#define LEN(x)  (sizeof(x) / sizeof((x)[0]))
#define BAR     10000.00

int main(void) {

    //-------------------------------------------------------------
    // Problem Specification
    //-------------------------------------------------------------

    // Species
    const cea_string reactants[]     = { "Air", "C7H8(L)", "C8H18(L),n-octa" };
    const cea_real reactant_temps[]  = {700.00,    298.15,          298.15 };
    const cea_real fuel_weights[]    = {   0.0,       0.4,             0.6 };
    const cea_real oxidant_weights[] = {   1.0,       0.0,             0.0 };

    // Products
    const cea_string omitted_products[] = {
        "CCN",            "CNC",            "C2N2",           "C2O",
        "C3H4,allene",    "C3H4,propyne",   "C3H4,cyclo-",    "C3",
        "C3H5,allyl",     "C3H6,propylene", "C3H6,cyclo-",    "C3H3,propargyl",
        "C3H6O",          "C3H7,n-propyl",  "C3H7,i-propyl",  "Jet-A(g)",
        "C3O2",           "C4",             "C4H2",           "C3H8O,2propanol",
        "C4H4,1,3-cyclo-", "C4H6,butadiene", "C4H6,2-butyne",  "C3H8O,1propanol",
        "C4H8,tr2-butene", "C4H8,isobutene", "C4H8,cyclo-",    "C4H6,cyclo-",
        "(CH3COOH)2",     "C4H9,n-butyl",   "C4H9,i-butyl",   "C4H8,1-butene",
        "C4H9,s-butyl",   "C4H9,t-butyl",   "C4H10,isobutane", "C4H8,cis2-buten",
        "C4H10,n-butane", "C4N2",           "C5",             "C3H8",
        "C5H6,1,3cyclo-", "C5H8,cyclo-",    "C5H10,1-pentene", "C10H21,n-decyl",
        "C5H10,cyclo-",   "C5H11,pentyl",   "C5H11,t-pentyl", "C12H10,biphenyl",
        "C5H12,n-pentane", "C5H12,i-pentane", "CH3C(CH3)2CH3",  "C12H9,o-
bipheny",
        "C6H6",           "C6H5OH,phenol",  "C6H10,cyclo-",   "C6H2",
        "C6H12,1-hexene", "C6H12,cyclo-",   "C6H13,n-hexyl",  "C6H5,phenyl",
```

```
        "C7H7,benzyl",      "C7H8",            "C7H8O,cresol-mx",  "C6H5O,phenoxy",
        "C7H14,1-heptene",  "C7H15,n-heptyl",  "C7H16,n-heptane",  "C10H8,azulene",
        "C8H8,styrene",     "C8H10,ethylbenz", "C8H16,1-octene",   "C10H8,napthlene",
        "C8H17,n-octyl",    "C8H18,isooctane", "C8H18,n-octane",   "C9H19,n-nonyl",
        "Jet-A(L)",         "C6H6(L)",         "H2O(s)",           "H2O(L)"
    };

    // Thermo States
    const cea_real pressures[] = {100.0*BAR, 10.0*BAR, 1.0*BAR };
    const cea_real of_ratio    = 17.0;

    cea_set_log_level(CEA_LOG_DEBUG);
    cea_init();

    // Mixtures
    cea_mixture reac, prod;
    cea_mixture_create(&reac, LEN(reactants), reactants);
    cea_mixture_create_from_reactants(&prod,
        LEN(reactants), reactants,
        LEN(omitted_products), omitted_products
    );

    // Solver
    cea_eqsolver solver;
    cea_eqsolver_create_with_reactants(&solver, prod, reac);

    // Solution
    cea_eqsolution soln;
    cea_eqsolution_create(&soln, solver);

    printf(
        "%12s %12s %15s %15s %15s \n",
        "P (Pa)", "O/F Ratio", "T (K)", "H (J/kg)", "Cp (J/kg-K)"
    );

    cea_real weights[LEN(reactants)];
    cea_mixture_of_ratio_to_weights(reac, oxidant_weights, fuel_weights, of_ratio,
weights);

    cea_real enthalpy;
```

```
    cea_mixture_calc_property_multitemp(reac, CEA_ENTHALPY, weights,
reactant_temps, &enthalpy);

    for (int ip=0; ip < LEN(pressures); ++ip) {

        cea_eqsolver_solve(solver, CEA_HP, enthalpy, pressures[ip], weights, soln);

        cea_real temperature, heat_capacity;
        cea_eqsolution_get_property(soln, CEA_TEMPERATURE, &temperature);
        cea_eqsolution_get_property(soln, CEA_EQUILIBRIUM_CP, &heat_capacity);

        printf(
            "%12.2f%12.2f%15.6e %15.6e %15.6e \n",
            pressures[ip], of_ratio, temperature, enthalpy, heat_capacity
        );

    }

    cea_eqsolution_destroy(&soln);
    cea_eqsolver_destroy(&solver);
    cea_mixture_destroy(&prod);
    cea_mixture_destroy(&reac);

    return 0;
}
```

Appendix D. Sample Code using the Python API

Sample code using the Python API is shown here.

```
import numpy as np
import cea

atm_to_bar = 1.01325
J_to_cal = 1./4.184
R = 8314.51

# Options
transport = True

# Species
reac_names = ["H2", "Air"]
prod_names = ["Ar", "C", "CO", "CO2", "H", "H2", "H2O",
              "HNO", "HO2", "HNO2", "HNO3", "N", "NH",
              "NO", "N2", "N2O3", "O", "O2", "OH", "O3"]

# Thermo states
densities = 1.0e3*np.array([9.1864e-5, 8.0877e-6, 6.6054e-7])  # kg/m^3
temperatures = np.array([3000.0])
phis = np.array([1.0])
fuel_moles = np.array([1.0, 0.0])
oxidant_moles = np.array([0.0, 1.0])

# Mixtures
reac = cea.Mixture(reac_names)
prod = cea.Mixture(prod_names)

# Solver
solver = cea.EqSolver(prod, reactants=reac, transport=transport)
solution = cea.EqSolution(solver)

# Unit conversions
fuel_weights = reac.moles_to_weights(fuel_moles)
oxidant_weights = reac.moles_to_weights(oxidant_moles)
of_ratios = len(phis)*[0.0]
for i, phi in enumerate(phis):
```

```
    of_ratios[i] = reac.weight_eq_ratio_to_of_ratio(oxidant_weights,
                                                    fuel_weights,
                                                    phi)

# Equilibrium solve
for of_ratio in of_ratios:
    for density in densities:
        for t in temperatures:
            weights = reac.of_ratio_to_weights(oxidant_weights, fuel_weights, of_ratio)
            solver.solve(solution, cea.TV, t, 1.0/density, weights)
```