12th U.S. National Combustion Meeting
Organized by the Central States Section of the Combustion Institute
May 24–26, 2021 (Virtual)
College Station, Texas

# Implementing a Steady-State Solver for Zero-Dimensional Reactors in Cantera

*Paul Blum*[1], *Bryan Weber*[1*], *and Raymond Speth*[2]

[1]*Department of Mechanical Engineering, University of Connecticut, Storrs, CT USA*
[2]*Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA USA*
[*]*Corresponding author:* bryan.weber@uconn.edu

**Abstract:** Zero-dimensional reactors are well understood numerical systems, often used to model simplified combustion systems. The time-dependent system of equations representing the evolution of species and energy in zero-dimensional reactors is implemented by several software packages. In particular, the free and open-source Cantera software includes a number of zero-dimensional reactors which can be solved as a function of time for specified initial conditions.

In Cantera, zero-dimensional reactors are modeled using the total reactor mass, volume, energy or temperature, and species mass fractions. These state variables are jointly applied in a manner that satisfies the laws of conservation for total mass and energy while maintaining chemical consistency in accordance with reaction rate formulas. Cantera includes a transient solver mode for zero-dimensional reactors where the governing equations are solved using the CVODES library from the SUNDIALS package, given initial values of the state variables.

However, several important classes of combustion problems involve the steady-state solution of zero-dimensional reactors. Currently, steady-state solutions are computed in Cantera by integrating the time-dependent system of ODEs until the relative change in the system state between subsequent timesteps is below a specified threshold. This is inefficient for large chemical systems and using a solver dedicated to solving the steady-state problem can provide significant speed improvements.

Cantera has implemented a damped-Newton/time-stepping hybrid solver, designed for differential algebraic equation (DAE) solutions in steady-state one-dimensional combustion problems with multiple domains, each containing multiple spatial points. However, this solver has not previously been applied to zero-dimensional reactors.

In this presentation, we will demonstrate the novel application of Cantera's hybrid steady/unsteady solver for zero-dimensional reactors. The relevant equations will be reviewed, and their combination to form a system of differential-algebraic governing equations will be presented. Several modifications to Cantera's source code were required to achieve this functionality, which will also be presented.

An example of a steady-state reactor network solution will be presented in the context of the classic zero-dimensional well-stirred reactor problem. Results will be compared to transient simulations to verify the solver's accuracy and to determine possible speed-up factors.

***Keywords: Cantera, zero-dimensional reactor, steady-state, kinetics***

## 1. Introduction

In combustion and many other fields, experiments are commonly modeled with computer simulations. The purposes of these simulations are manyfold, from developing a deeper understanding of the physical processes driving a global phenomenon, to developing and predicting the performance of new engineering designs.

Performing accurate computer simulations requires both knowledge of the overall governing equations (for example, mass and energy) as well as the specific parameters relevant to the problem at hand (for example, reaction rate

coefficients). An appropriate computer program will implement the relevant governing equations and provide a facility to input parameters of the simulation.

In general, the governing equations of a system will depend on time and spatial coordinates. In some cases, it is appropriate to simplify the governing equations to remove the spatial dependence. Such systems are typically called homogeneous, or batch, reactors. This simplification of the governing equations results in a system of ordinary differential equations (ODEs). The numerical solution of systems of ODEs is a well-studied problem and many algorithms are available to perform these calculations. The interested reader is referred to one of the many textbooks available, for example, the work of Butcher [1].

On the other hand, there are many systems for which neglecting the spatial variation of properties is not appropriate. For these systems, the governing equations result in a set of coupled partial differential equations (PDEs) in time and space. The solution of this type of system is substantially more difficult than a system of ODEs, resulting in both more complicated code and longer computation times.

Therefore, systems with spatial dependence are often assumed to be at steady state. Under the assumption of steady state, and after spatial discretization, the governing equations are modified to remove the dependence on time derivatives, resulting in a system of nonlinear algebraic equations. Moreover, it is often useful to solve homogeneous systems under the assumption of steady state, especially open systems that allow the transfer of matter across the system boundary. The solution of such systems is also well studied; the interested reader is referred to the work of Brenan et al. [2].

### 1.1 Cantera

Beginning with the CHEMKIN library [3] and the related application codes, many computer programs have been written to solve the ODE and algebraic systems relevant for combustion. Often, the development of a new framework is motivated by the availability of new programming languages and paradigms that promise to simplify development, maintenance, use, or all three. Around the year 2000, Prof. Dave Goodwin at the California Institute of Technology identified that the C++ programming language and an object-oriented approach to development would be useful to modularize many common combustion simulations. Thus, he began development on Cantera.

Some 20 years later, Cantera [4] has grown as a free, open-source computational toolbox to solve problems involving thermodynamics, chemical kinetics, and transport. Cantera is written primarily in the C++ programming language, with user interfaces in Python, C++, MATLAB®, and FORTRAN. The core of Cantera is a hierarchical collection of objects designed to represent thermodynamic phases and chemical reactions, along with their associated properties.

Cantera also includes objects that implement the solution of the mass, energy, and momentum conservation equations for several systems. For zero-dimensional (0-D) systems, that is, those dependent on time but not space, the relevant classes are the `Reactor` and `ConstPressureReactor`. As discussed in Section 2, the difference between these classes are the state variables used in the solution of the governing equations.

Cantera can also solve problems in one spatial dimension (1-D), for example, the canonical laminar flame speed calculation. These problems are constructed assuming that the system is in steady state, such that the governing equations can be reduced to a set of algebraic equations. Although the steady-state functionality is built-in to Cantera, it is possible to use the thermochemical building blocks within Cantera to construct a transient 1-D solver; interested readers are directed to the work of Long et al. [5].

By contrast, for the 0-D reactors, only the transient solution is possible at present. Cantera constructs the set of ordinary differential equations that represent the system and passes them to the CVODES solver in the SUNDIALS package [6]. However, as discussed previously, having the ability to directly solve the steady-state problem for 0-D reactors would be advantageous.

Thus, in this work, we present the development and application of a steady-state solver for Cantera 0-D reactors. The code is based on the existing hybrid steady-transient solver used for 1-D problems in Cantera, but generalized to solve problems with no spatial dependency. In the following sections we describe the relevant conservation equations for 0-D reactors, the existing implementation in Cantera, and the necessary modifications for direct steady state solution. This is followed by a demonstration of the accuracy and utility of the new solver from this work.

## 2. Overview of Conservation Equations

As discussed previously, the two primary 0-D objects in Cantera are the `Reactor` and `ConstPressureReactor`. These reactors are general purpose objects, permitting phases with any implemented equation of state to be used, and

allowing various inlets and outlets to connect an arbitrary number of reactors into a network.

The state in a reactor at any instant of time is determined by the set of state variables. For the `Reactor`, these include:

- $m$, the mass of the reactor's contents (kg)
- $V$, the total volume of the reactor ($m^3$)
- $U$, the total internal energy of the reactor's contents (J)
- $Y_k$, the mass fraction of species $k$ for $k \in 1,2,\ldots,N$, where $N$ is the total number of species

For the `ConstPressureReactor`, it is more convenient to work with the enthalpy, rather than the internal energy and total volume. Thus, the number of state variables for a `ConstPressureReactor` is one fewer than the number for a `Reactor`:

- $m$, the mass of the reactor's contents (kg)
- $H$, the total enthalpy of the reactor's contents (J)
- $Y_k$, the mass fraction of species $k$ for $k \in 1,2,\ldots,N$

In the present work, we focus on the `Reactor` class for convenience, but the implementation of the steady-state solver in Cantera is general and can be used for either reactor object type. We also note that open-system, constant-pressure, homogeneous reactors with a fixed residence time (usually called the perfectly stirred reactor (PSR)) are usually modeled using constant volume reactors where the mass flow rate is adjusted to achieve the steady state solution.

### 2.1 Transient Governing Equations

To advance the state of a `Reactor` instance in time, it is necessary to simultaneously solve the mass, energy, and species conservation equations. Cantera `Reactor` instances can also account for varying volume, heat transfer to or from the surroundings, or surface reactions, but these effects are neglected here for simplicity. The following derivation follows the Cantera documentation for a single reactor [7].

The transient mass conservation equation is given by:

$$\frac{dm}{dt} = \sum_{\text{in}} \dot{m}_{\text{in}} - \sum_{\text{out}} \dot{m}_{\text{out}} \tag{1}$$

where $m$ is the mass of the reactor at any instant and the subscripts in and out indicate inlet and outlet mass flow rates, respectively. Transient conservation of energy is given by:

$$\frac{dU}{dt} = \sum_{\text{in}} \dot{m}_{\text{in}} h_{\text{in}} - h \sum_{\text{out}} \dot{m}_{\text{out}} \tag{2}$$

where $h$ is the mass-specific enthalpy, given by:

$$h = \sum_k h_k Y_k \,. \tag{3}$$

Note that properties at the outlet of the reactor are equal to the properties inside the reactor and do not have a subscript.

Finally, the species conservation equations are given by:

$$m \frac{dY_k}{dt} = \sum_{\text{in}} \dot{m}_{\text{in}} Y_{k,\text{in}} - Y_k \sum_{\text{out}} \dot{m}_{\text{out}} + \dot{\omega}_k W_k V \tag{4}$$

where $\dot{\omega}_k$ and $W_k$ are the volumetric rate of production and molecular weight, respectively, of species $k$. There are $N$ species conservation equations that must be evaluated.

In addition to these conservation equations, a `Reactor` must solve for the time-rate change of the reactor volume ($dV/dt$), but we assume this term is zero and neglect it here. This set of $N+3$ ordinary differential equations is evaluated in Cantera's source code and the resulting set of time derivatives is passed to CVODES for time integration.

### 2.2 Steady State Governing Equations

A reactor is in steady state when all state variables no longer change as a function of time, as different internal processes that would normally change these variables are perfectly balanced with each other. That is, the time derivative of all state variables is zero in steady state, by definition. The governing equations previously discussed will still dictate the physical properties of the system, but the left-hand side of Eqs. (1), (2), and (4) will be equal to zero. The steady-state assumption reduces the governing equations to a system of nonlinear algebraic equations.

For a single `Reactor`, the mass conservation equation becomes:

$$\sum_{\text{in}} \dot{m}_{\text{in}} - \sum_{\text{out}} \dot{m}_{\text{out}} = 0 \tag{5}$$

implying that the sum of the inlet mass flow rates is equal to the sum of the outlet mass flow rates. Next, the steady-state energy conservation equation becomes:

$$\sum_{\text{in}} \dot{m}_{\text{in}} h_{\text{in}} - h \sum_{\text{out}} \dot{m}_{\text{out}} = 0\,. \tag{6}$$

Note that, in steady state, the time rate change of the reactor volume ($dV/dt$) is zero by definition, so there is no possibility for moving boundary work in the energy equation. Finally, the set of species conservation equations become:

$$\sum_{\text{in}} \dot{m}_{\text{in}} (Y_{k,\text{in}} - Y_k) + \dot{\omega}_k W_k V = 0 \tag{7}$$

where we have used the implication of Eq. (5) to simplify the first term of Eq. (7). Including the trivial equation for the constant volume, there are again $N+3$ equations to be solved.

### 2.3 Existing Transient Integration Method

To conduct a transient integration of Eqs. (1), (2), and (4), Cantera requires a set of initial values for the state variables and access to an integrator. An object known in Cantera as a `ReactorNet` coordinates the evaluation of the governing equations for a set of reactor instances, and passes the resulting time derivative vector to the CVODES integrator.

The general solution procedure for 0-D simulations in Cantera is:

1. Create instances of classes to calculate thermodynamic, chemical kinetic, and transport properties. This can be accomplished in one step by creating a `Solution` container class, usually by importing data from an input file.
2. Set the intensive state of the `Solution` by fixing two independent thermodynamic variables and the composition.
3. Create an instance of a `Reactor` class and insert the `Solution` instance. The `Reactor` has an associated extensive volume, which determines two of the state variables discussed in Section 2.
4. Create instances of any devices necessary to connect multiple `Reactor` instances together, such as `MassFlowController` or `PressureController` instances.
5. Create an instance of a `ReactorNet` and insert all of the `Reactor` instances that will be integrated together.
6. Advance the simulation in time using `ReactorNet.step()` or `ReactorNet.advance(end_time)` until the desired termination criteria are reached.

This procedure is demonstrated, in Python, in the example below. This example is based on the longer example with more exposition available on the Cantera example website.

In Listing 1, line 1 imports Cantera into the Python namespace, using the standard abbreviation `ct`. Line 2 accomplishes Step 1 of the list above, by loading the `h2o2.yaml` input file distributed with Cantera. New input files can be created by converting existing mechanisms from other formats; interested readers are directed to the Cantera online tutorials. Line 3 then sets the intensive state using the temperature, pressure, and mole fractions. Note that Cantera uses SI units by default.

Line 4 creates the `Reactor` instance and sets the extensive volume. Note that the `IdealGasReactor` is used in Listing 1. The `IdealGas*` reactor classes use a slightly modified set of state variables, replacing $U$ with $T$ (or $H$ with

```
1 import cantera as ct
2 gas = ct.Solution("h2o2.yaml")
3 gas.TPX = 1000.0, 101325.0, "H2:2,O2:1,N2:4"
4 reactor = ct.IdealGasReactor(gas, volume=1.0E-5)
5 sim = ct.ReactorNet([reactor])
6 sim.advance(1.0)
```

Listing 1: Cantera `Reactor` example in Python

$T$ in the case of the constant pressure reactor) and transforming the energy conservation equation by using the specific heats. This improves the numerical stability and performance of the integration, but is obviously only applicable to ideal gases.

Finally, lines 5 and 6 create the `ReactorNet` containing the single `Reactor` in this example and use `ReactorNet.advance()` to simulate to an end time of 1 second. CVODES takes multiple internal time steps to reach this end time, to ensure accuracy of the final solution.

CVODES uses a variable-order BDF method to perform time integration. The first-order BDF method is equivalent to the backward Euler method, which is shown here for simplicity. Starting from the initial state, future states are calculated implicitly using an iterative formula:

$$y_{n+1} = y_n + \Delta t f(y_{n+1}, t_{n+1}) \tag{8}$$

where $y$ is the state vector:

$$y = \begin{bmatrix} m & V & U & Y_1 & Y_2 & \dots & Y_N \end{bmatrix}^T, \tag{9}$$

$n$ indicates the time step, and $f(y,t)$ is defined as the time derivative of the state vector, given by Eqs. (1), (2), and (4):

$$f(y,t) \equiv \frac{dy}{dt} \tag{10}$$

Rearranging terms in Equation (8) allows the solution to be performed as a root-finding problem:

$$y_{n+1} - y_n - \Delta t f(y_{n+1}, t_{n+1}) = 0 \tag{11}$$

$$\implies g(y_{n+1}) = 0 \tag{12}$$

Higher-order BDF methods incorporate the values of $y_{n-i}$ and $f(y_{n-i}, t_{n-i})$ where $i+1$ is the order of the BDF method to define a similar root-finding problem. $f(y,t)$ is implemented in the source code by the `Reactor::evalEqs` method. Given a state vector $y$, `evalEqs` computes the time derivative vector $dy/dt$. In the case of an adiabatic constant volume reactor, `evalEqs` returns the derivatives as shown in Eqs. (1), (2), and (4).

### 2.4 Modifications for the Steady-State Solver

In contrast to the transient case, the steady-state governing equations presented in Eqs. (5) to (7) are formulated as a system of nonlinear equations, and can be solved directly as a root-finding problem. Cantera already includes infrastructure for solving problems of this type in its `oneD` solution module, where the steady-state assumption is utilized in order to simplify modeling and computation. The 1-D solver was built for problems with spatial dependence, and supports a hierarchy of linked domains and their contained solution points. A domain defines a 1-D system of equations, and different domains in the same problem can have varying numbers of state variables or solution points. Beneath the classes that implement the spatial logic for Cantera's 1-D solver in the source code, there is a custom C++ implementation for a bounded and damped quasi-Newton solver with adjustable time stepping. This numerical solution algorithm is highly optimized to solve stiff nonlinear initial-value problems, and works especially well for computing gas state formulations where component bounding and damped stepping are of critical importance towards achieving solution convergence. With the incorporation of adjustable time stepping, the Cantera 1-D solver provides a highly-capable method to improve convergence in complex systems. If the Newton method fails, the 1-D solver can take a series of transient time-integration steps from the initial state, thereby advancing the simulation to a future

state where nonlinear solution is more likely to converge. For stiff systems, this cycle is repeated until a solution is identified.

Zero-dimensional transient solution in Cantera is implemented in `ReactorNet`, numerically characterized by a single system of ordinary differential equations that encompasses the governing equations of all contained `Reactor` instances. With some interfacing, the existing Cantera 1-D solver can be used to directly solve for `ReactorNet` steady state, configuring the problem as a single-domain and single-point one-dimensional system. However, because of identified potential for significant simplification and generalization only achievable by re-implementing the 1-D solver, a new `Newton` solver class was developed to be used for 0-D solution.

The `Newton` class is based on the 1-D solver, but eliminates support for calculating spatial dependence with domains and solution points. The logic for this capability consisted of mostly loop-based iterations of Newton solutions over layered 1-D components. Additionally, the new `Newton` implementation simplifies the Jacobian matrix, minimizing entries by switching from banded to dense structure and removing an external class dependence by storing the matrix locally as an `Array2D`. The new Jacobian matrix is generated explicitly in the solver with finite differences, where a residual vector is evaluated for small perturbations of each member of the solution vector. Solution of the Jacobian is performed by LAPACK using LU decomposition. For computational efficiency, the Jacobian is reused in several subsequent Newton steps because the Newton solver is not very sensitive to the Jacobian.

Cantera's transient formulation of the ODE equations includes a dynamic component to describe reactor volume; however, in steady-state, this property is a defined constant. Thus, the algebraic constraint on the volume will result in an unsolvable singular Jacobian if used in simulation. To fix this problem, the Jacobian row and column corresponding to any constant component (volume in the cases shown here) should be filled with 0, with a 1 in the main-diagonal position.

## 3. Results and Discussion

The hybrid Newton solver implemented here is a powerful tool for steady-state solution and for nonlinear solution in general. The adaptability of this solver offers usefulness in a wide range of applications, while its convergence capability enables new solution possibilities in complex reactor networks. In this section, the `Newton` solver class is numerically verified, and its application to the simulation of a methane-fueled combustor is demonstrated and compared with Cantera's existing time integration approach. All code references in this section, as well as additional Cantera steady-state examples, are available on GitHub.

### 3.1 Verification of Newton Solver

After re-implementing the Cantera 1-D solver, functionality was verified by using the new implementation to find the solutions to two systems of nonlinear equations that have analytical solutions. The systems used are:

$$\begin{cases} x - y &= 0 \\ x^2 - y &= 0 \end{cases} \tag{13}$$

$$\begin{cases} x^2 - y &= 0 \\ 4x^2 - 2x + 3y &= 0 \end{cases} \tag{14}$$

The analytical solutions $(x, y)$ to the systems in Eqs. (13) and (14) are shown in Table 1, along with the computed results from the `Newton` solver class. The tolerance for the `Newton` solver was set to $1 \times 10^{-14}$ and the precision of the `double` type numbers used in the solution is approximately 15 decimal digits.

These results confirm the ability of the `Newton` solver to produce accurate results, independent of the starting guess, although the number of iterations increases for a worse initial guess.

### 3.2 Accuracy and Performance in Steady-State Simulation

The steady-state solver implementation was tested and compared to the time-integration solution method in a series of simulations of a methane-fueled combustor, written in Python. The simulation consists of a $1\,\mathrm{m}^3$ combustor modeled as a standard `Reactor`, with a single inlet and a single outlet. An air/fuel mixture flows into the combustor from an infinitely large upstream tank modeled as a `Reservoir`. The inlet mass flow rate is controlled by a `MassFlowController` to hold the residence time of the gas in the `Reactor` constant.

Table 1: Verification results of the re-implemented `Newton` solver class.

| System | Analytical Solution | Computed Solution | Initial Guess | Newton Iterations |
|---|---|---|---|---|
| Eq. (13) | $(1,\ 1)$ | $(1,\ 1)$ | $(0,\ 2)$ | 8 |
| Eq. (13) | $(1,\ 1)$ | $(1,\ 1)$ | $(50,\ 99)$ | 17 |
| Eq. (13) | $(0,\ 0)$ | $(-3.29\times10^{-22},\ -3.29\times10^{-22})$ | $(0.5,\ -0.5)$ | 10 |
| Eq. (14) | $(2/7,\ 4/49)$ | $(0.285714,\ 0.0816327)$ | $(10,\ 20)$ | 14 |
| Eq. (14) | $(2/7,\ 4/49)$ | $(0.285714,\ 0.0816327)$ | $(500,\ -999)$ | 22 |
| Eq. (14) | $(0,\ 0)$ | $(-5.46\times10^{-23},\ -1.56\times10^{-23})$ | $(-50,\ 99)$ | 20 |

Exhaust leaves the `Reactor` and flows into an infinitely large downstream capture tank, which is modeled as a `Reservoir`. The outlet mass flow rate is controlled by a `PressureController`, which continuously computes the outlet mass flow rate that will maintain constant `Reactor` pressure.

The fuel is pure methane ($CH_4$), and the equivalence ratio, $\phi$, in the upstream tank is specified using air as the oxidizer. Air is modeled as a mixture of $O_2$ and $N_2$ in molar proportions of $1:3.76$. The thermochemical data are taken from GRI-Mech 3.0 for simplicity [8]. Equivalence ratios of $\phi = 0.5$, 1.0, and 1.5, with initial temperature of $T_0 = 300\,\mathrm{K}$ and initial pressure of $p_0 = 1\,\mathrm{atm}$ were used for the simulations.

In this example, many steady-state solutions are to be computed for a range of combustor residence times. A combustor's steady state changes with its residence time, related to the corresponding completeness of its combustion reaction. This characteristic is reflected in the steady-state reactor temperature, which decreases as the residence time decreases, as shown in Figs. 1 to 4.

In order to investigate the accuracy of the new solver, results from the steady-state solver are verified against solutions computed by Cantera's existing time-integration solver. The steady-state temperature agrees within $5\times10^{-7}\,\%$ between the time-integration and new steady-state solver, as illustrated in Fig. 1.

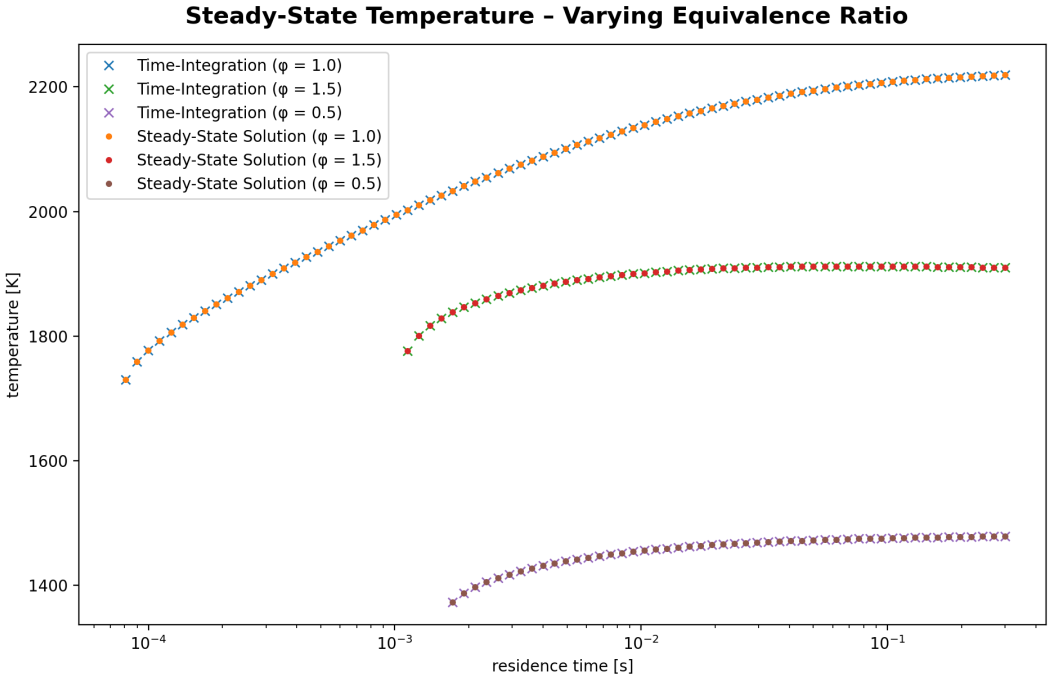


Figure 1: Accuracy verification of the steady-state solver against the existing Cantera time-integration solver.

Finding a combustor's steady state is an initial value problem, and as such it requires an initial guess from which

to start numerical iterations towards the solution. Numerical efficiency is strongly affected by choice of initial guess, as demonstrated in Table 1, and an insufficient guess can cause solution divergence.

For this simulation, one reasonable initial guess is the equilibrium state of the inlet air/fuel mixture, as this is the steady-state solution at an infinite residence time. Alternately, solutions at smaller residence times can be found using the solution from a larger residence time as an initial guess. With this method, solutions can be reliably traced backwards along the reaction S-curve to the extinction point. Note that initializing from the equilibrium state is a more computationally expensive solution method, but solutions are independent of the previous solution.

These differing initial guess formulations provide differing steady-state solutions in computations by the new steady-state solver implementation. As shown in Fig. 2, the new solver can find solutions on the physically unstable middle branch of the S-curve when using the equilibrium solution as the initial guess. Cantera's time-integration solution finds only stable burning solutions. At present, it is not clear why using equilibrium as the initial guess sometimes finds the middle-branch solution. Further investigation will be conducted as future work.

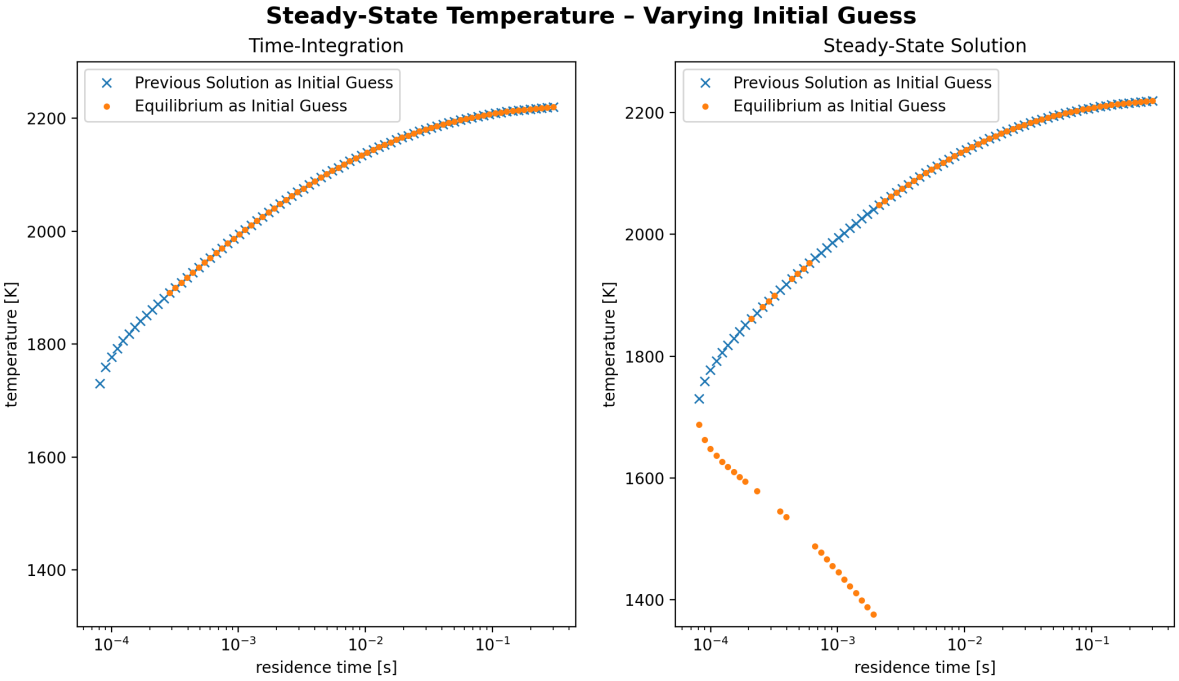


Figure 2: Differing solution paths between solver implementations. $\phi = 1.0$, $T_0 = 300\,\mathrm{K}$, $p_0 = 1\,\mathrm{atm}$.

Utilization of the new steady-state solver implementation can allow a significant reduction in the number of time steps and the amount of computation time needed for solution convergence. Figures 3 and 4 present performance comparisons between Cantera's existing time-integration solver and the new steady-state solver. The total computation times are measured on a 2017 MacBook Pro with a 2.3 GHz Dual-Core Intel Core i5 processor and 8 GB RAM. Nonetheless, we expect similar differences between steady-state and transient solutions on other machines.

On average, the steady-state solver found solutions more than twice as fast as the time-integration solver. As discussed previously, the new solver combines direct nonlinear solution with transient time integration to provide a fast and robust method for solving root-finding problems. Time steps are computationally expensive due to the implicit formulation and requisite Jacobian evaluation, but can be used to improve conditioning of the initial guess in cases of divergence with the `Newton` solver. The total run time can be minimized by optimizing the number of time-integration steps to be taken after divergence with Newton's method. For this example, a series of 17 time steps provided optimal computation time.

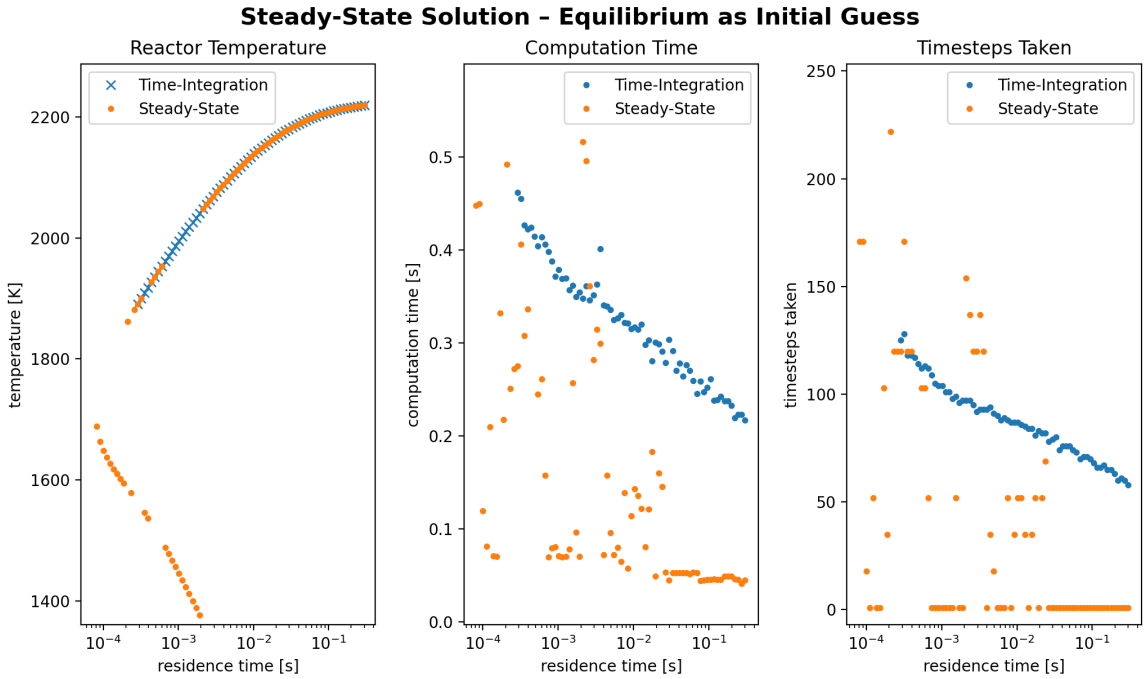


Figure 3: A performance comparison between solvers running a computationally expensive solution method. $\phi = 1.0$, $T_0 = 300\,\mathrm{K}$, $p_0 = 1\,\mathrm{atm}$.

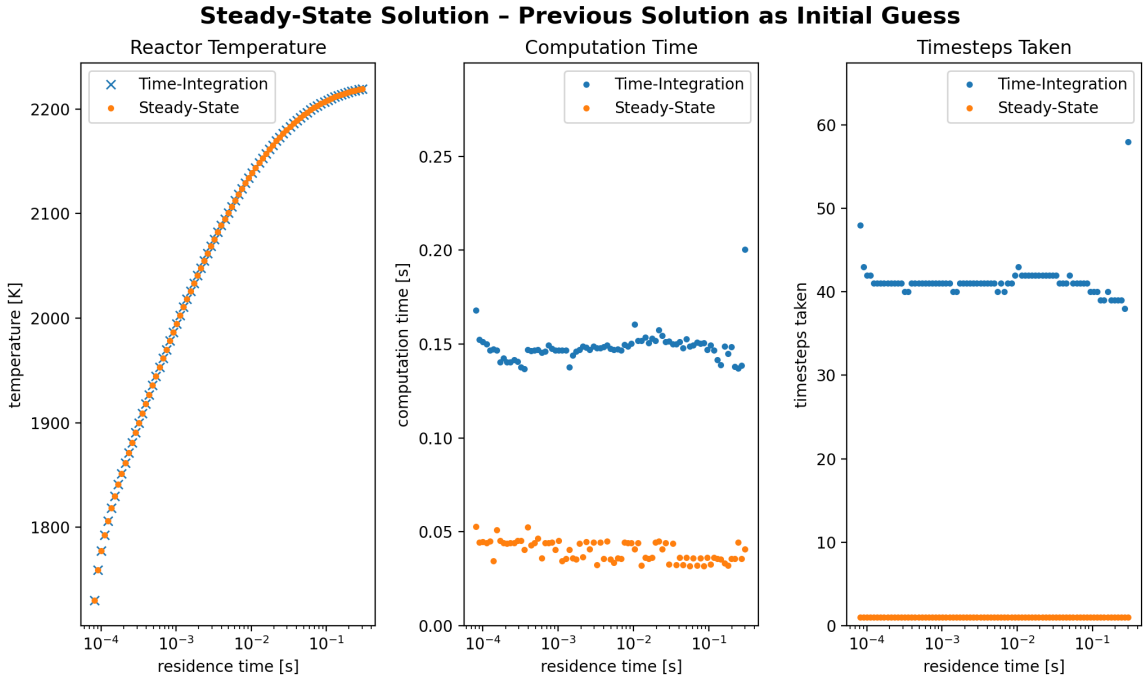


Figure 4: A performance comparison between solvers running an optimized solution method. $\phi = 1.0$, $T_0 = 300\,\mathrm{K}$, $p_0 = 1\,\mathrm{atm}$.

## 4. Conclusions

In this work, we demonstrated a new implementation of a solver for the nonlinear algebraic system associated with the steady-state governing equations for combustion. The solver is integrated into Cantera, the free and open-source toolbox for thermochemical simulations.

The new solver implementation is demonstrated to produce accurate solutions for systems of equations with known analytical solutions. In addition, the new implementation is shown to agree very well with the existing time-integration solver built-in to Cantera. Finally, the new implementation provides at least 2X speedup on a common combustion problem.

### 4.1 Future Work

This presentation details an early implementation of the steady-state solver for 0-D systems in Cantera. The new solver is expected to be included in the next version of Cantera. Remaining tasks include coupling the time-integration code directly into the `Newton` solver class, adding documentation, and providing more examples demonstrating the use and convenience of the steady state solver in Cantera.

### Acknowledgements

This material is based upon work at the University of Connecticut supported by the National Science Foundation under Grant No. 1931539. This material is also based upon work at the Massachusetts Institute of Technology supported by the National Science Foundation under Grant No. 1931391. P.B. was supported during the Summer of 2020 by a Google Summer of Code award.

### References

[1] J. C. Butcher, Numerical Methods for Ordinary Differential Equations, J. Wiley, Chichester, West Sussex, England ; Hoboken, NJ, 2003.

[2] K. E. Brenan, S. L. Campbell, and L. R. Petzold, Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations, Society for Industrial and Applied Mathematics, 1995, DOI: 10.1137/1.9781611971224.

[3] R. J. Kee, F. M. Rupley, E. Meeks, and J. A. Miller, CHEMKIN-III: A FORTRAN Chemical Kinetics Package for the Analysis of Gas-Phase Chemical and Plasma Kinetics, Report No. SAND96-8216, Sandia National Laboratories, 1996.

[4] D. G. Goodwin, R. L. Speth, H. K. Moffat, and B. W. Weber, *Cantera: An Object-oriented Software Toolkit for Chemical Kinetics, Thermodynamics, and Transport Processes*, `https://www.cantera.org`, Version 2.5.1, 2021, DOI: 10.5281/zenodo.4527812.

[5] A. E. Long, R. L. Speth, and W. H. Green, Ember: An Open-Source, Transient Solver for 1D Reacting Flow Using Large Kinetic Models, Applied to Strained Extinction, Combustion and Flame 195 (2018) 105–116. DOI: 10.1016/j.combustflame.2018.05.001.

[6] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers, ACM Transactions on Mathematical Software 31 (2005) 363–396. DOI: 10.1145/1089014.1089020.

[7] C. Developers, *Reactor Models in Cantera*, Cantera, 2021, URL: `https://cantera.org/science/reactors.html` (visited on 04/05/2021).

[8] G. P. Smith, D. M. Golden, M. Frenklach, N. W. Moriarty, B. Eiteneer, M. Goldenberg, C. T. Bowman, R. K. Hanson, S. Song, J. William C. Gardiner, V. V. Lissianski, and Z. Qin, *GRI-Mech 3.0*, URL: `http://combustion.berkeley.edu/gri-mech/version30/text30.html`.