Computer Physics Communications 276 (2022) 108364

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

PyCSP: A Python package for the analysis and simplification of chemically reacting systems based on Computational Singular Perturbation [☆], [☆☆]

Riccardo Malpica Galassi

Aero-Thermo-Mechanics Laboratory, Ecole Polytechnique de Bruxelles, Universite' Libre de Bruxelles, Avenue F.D Roosevelt 50, Brussels 1050, Belgium

ARTICLE INFO

Article history:
Received 25 January 2022
Received in revised form 28 March 2022
Accepted 30 March 2022
Available online 11 April 2022

Keywords:
Chemical kinetics
Combustion
CSP
Open source
Skeletal mechanism

ABSTRACT

`PyCSP` is a Python package for the analysis and simplification of chemically reacting systems, using algorithms based on the Computational Singular Perturbation (CSP) theory. It provides tools for the local characterization of the chemical dynamics, enabled by the recognition of a convenient projection basis which carries out a timescale-based uncoupling. The tools supplied within the package allow one to identify the rate-controlling chemical reactions, the intrinsic chemical timescales, the driving chemical timescale and indicators of the system's explosive or dissipative propensity. Possible applications are the analysis of numerical simulations of reacting flows, and the reduction of chemical kinetics models, based on the CSP information. This manuscript provides a brief overview of the foundations of CSP, a description of the libraries, and demonstrations of the features implemented in `PyCSP` with code examples, along with practical advices and guidelines for users.

Program summary
Program Title: `PyCSP`
CPC Library link to program files: https://doi.org/10.17632/59pw7pvkkb.1
Developer's repository link: https://github.com/rmalpica/PyCSP
Licensing provisions: MIT
Programming language: Python
Supplementary material: Code documentation and Python scripts employed to generate the figures.
Nature of problem: The evermore increasing availability of high-performance computing resources, and the compelling need for more advanced and sustainable energy conversion devices, based on unconventional combustion regimes and alternative fuels, are driving towards an unprecedented massive production of data in numerical simulations of reacting flows. The research questions behind the production of such huge datasets are typically related to (i) the fundamental understanding of combustion phenomena, and (ii) the development of reduced order models and/or turbulence-chemistry interaction sub-grid scale (closure) models, both with the aim of accelerating large scale simulations of real combustion devices.
Solution method: Both categories of research questions can widely benefit from the numerical tools available in `PyCSP`. The computational singular perturbation (CSP) framework allows one to extract concise information from chemically reacting systems, automatically and at reasonable cost. This is especially useful when the dataset is so massive and the number of degrees of freedom so large, i.e., hundreds of species/reactions per cell, that even a visual inspection becomes unmanageable. `PyCSP` offers a fast, user-friendly implementation of numerous analysis tools, enabling a more systematic data processing and, ultimately, providing the user with a deeper physical understanding of the problem under investigation. Moreover, the CSP theoretical framework can be exploited to generate reduced order models (ROMs), tailored to and to be employed in specific applications, in order to drastically reduce the computational cost of a numerical simulation, while retaining accuracy in global observables. The ROM is in the form of a skeletal kinetic mechanism of adjustable fidelity, or an adaptive chemistry integrator.

[☆] The review of this paper was arranged by Prof. N.S. Scott.
[☆☆] This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
E-mail addresses: riccardo.malpica.galassi@ulb.be, riccardo.malpica@gmail.com.

https://doi.org/10.1016/j.cpc.2022.108364

Additional comments including restrictions and unusual features: PyCSP relies on Cantera, an open-source suite of tools for problems involving chemical kinetics, thermodynamics, and transport processes, to efficiently incorporate detailed chemical thermo-kinetics models into the CSP calculations.

1. Introduction

The role of combustion in the ongoing energy transition is far from being secondary. Even though the main motivation driving such crucial process for mankind is the replacement of fossil fuels with renewable energy sources, the act of *burning* will remain a dominant energy conversion process. Hydrogen, hydrogen-based energy carriers (e.g. ammonia, synthetic methane, methanol, etc.), biomasses, and synthetic fuels, will play a central role in many large-scale applications, from energy-intensive industries to aviation, whose electrification does not appear to be a viable solution. The combustion of novel fuels requires fundamental studies to enable predictive modeling, and, in turn, to improve combustion efficiency, stability, thus safety and reliability of practical devices.

Flows in combustion chambers are characterized by strong interplays between turbulence, mixing and chemical reactions, that interact over a wide dynamic range of space and time scales. Detailed chemical kinetic mechanisms describe the conversion of reactants into products through thousands of elementary steps, often acting over a wide spectrum of disparate timescales which can range from nanoseconds to minutes. Since modeling the fluid dynamics of a practical system is a challenge of its own, the small-scale flowfield-chemistry interaction is doubtlessly the largest obstacle that keeps numerical simulations of practical combustion devices on the ambitious side. This is the reason why direct numerical simulations (DNS) of the reactive Navier-Stokes equations are practically limited to small and geometrically simple domains, being solely devoted at capturing small-scale phenomena in an attempt to improve their fundamental understanding. Indeed, when it comes to engineering applications, especially in the context of robust design of combustion devices, the DNS approach becomes impractical, due to the typically large spatial dimensions of the domains, the highly turbulent nature of the flow, and the number of simulations required to identify optimal configurations. In such cases, the only tractable computational approaches for turbulent combustion are those involving fully or partially modeled spatial frequencies, i.e., Reynolds-averaged Navier-Stokes (RANS) and large eddy simulations (LES), which however require closure rules for the subgrid-scale interactions. A universally accepted closure model, capable of describing turbulence-chemistry interaction in a wide range of combustion regimes, is still not available [1].

Fundamental understanding is at the heart of modeling. Commonly, DNS datasets of canonical configurations are employed to develop closure and reduced order models. Such datasets are huge and contain a plethora of information: each computational cell of a typically 3D mesh provides values for primitive state variables such as temperature, pressure, velocity, and composition ($\sim$10÷100 species), at each integration time-step. Fortunately, the true system's dynamics, that has to be understood, evolves with a much smaller number of degrees of freedom. In fact, the action of the components of the model that generate fast time-scales effectively constrains the evolution of the system within an embedding of low-dimensional manifolds [2,3]. The existence of manifolds paves the way to both (i) a more effective information retrieval and, in turn, physical understanding, and (ii) the development of reduced order models.

To this end, the computational singular perturbation (CSP) method seeks the *slow invariant manifolds* (SIMs) developing in complex systems of non-linear singularly-perturbed ODEs and PDEs by locally reformulating the underlying dynamics in terms of new state variables, which are linear combinations of the original ones, and whose time-evolutions are decoupled from each other. This projection operation allows to systematically identify features of the vector field which could not be investigated in the original state space, such as: the existence of fast/slow timescales, which in turn define fast/slow subspaces, the dimension and directions of the SIM, and the cause-effect relationships among the "causing" physical processes (convective or diffusive transport, chemical reactions) and the "effects" on the physical state variables. The CSP theory and the derived algorithmic tools are comprehensively described in [4,5].

PyCSP is an open-source Python package for the analysis and reduction of complex chemically reacting systems using CSP. It consolidates years of developments and experience with CSPTk, a Fortran 90 code devoted to CSP and employed in numerous works by the author's group [6–13] and other groups [14–18], which however was never publicly distributed. The typical investigation settings in which PyCSP is anticipated to be useful are (i) the post-process of massive reacting flows datasets, as done in [10,14,16,19,20] (ii) the generation of reduced order models (ROMs) in the form of skeletal kinetic mechanisms, as in [11,12,21], and (iii) the development of numerical techniques that exploit SIMs and ROMs for accelerating reacting flows computations on the fly, as in [22]. The basic features of PyCSP consist in the local timescale characterization, the identification of the slow manifold and the chemical processes, i.e., species and reactions, associated with the fast and slow systems, the identification of a representative of the driving chemical timescale based on the tangential stretching rate theory [23] and its associated chemical processes. On top of the basic features, additional modules offer the implementation of an adaptive stiff ODE solver and a mechanism simplification algorithm, both based on the CSP decomposition. The PyCSP package is aimed at researchers in the field of chemistry, combustion and engineering, providing them access to detailed but synthetic information about complex and otherwise unmanageable chemical systems. The package is designed to be accessible to inexperienced users, providing user-friendly examples of all the available features, while also offering the possibility of customization and expansion towards more advanced applications.

2. Software features

To leading order, the CSP decomposition is an eigen-decomposition of the $(N+1)$-by-$(N+1)$ Jacobian matrix of the local chemical source term $\dot{\omega}(T, Y_i)$, where T is temperature and Y_i are the N species mass fractions of a given state point. The aforementioned operation yields the CSP kernel, which is an object containing: the eigen-values, hence the chemical time-scales, i.e. the inverse of the eigen-values, the eigen-modes, i.e. the right eigen-vectors, and their amplitudes, computed as a projection of the vector field $\dot{\omega}$ onto the left eigen-vectors.[1] Once the CSP kernel is available, the application of a convenient criterion [6] determines how many of the $(N+1)$ modes are exhausted, i.e., have an approximately

[1] Note that in the partial differential equations (PDEs) version of CSP, the projected vector field is $(\dot{\omega}+L)$, where L is the transport term contribution (convection and/or diffusion).

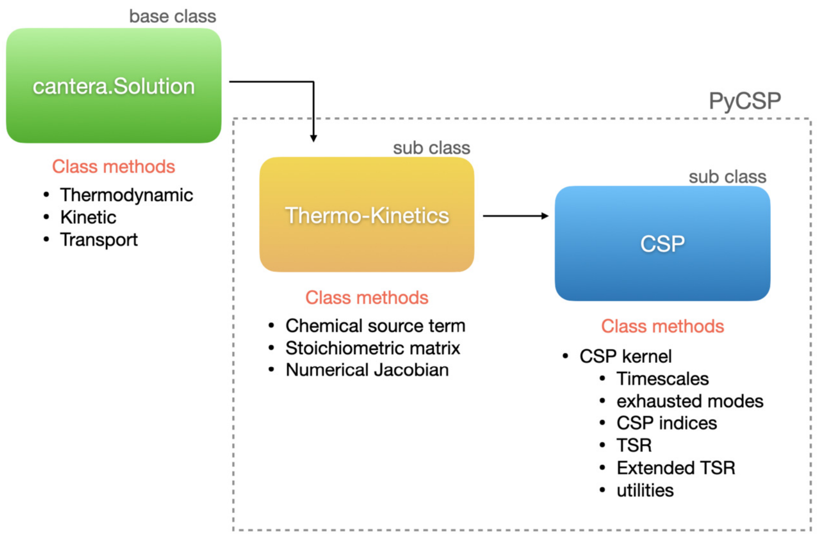


Fig. 1. PyCSP classes hierarchy: sub-classes of `cantera.Solution`.

zero amplitude. The number M of exhausted modes corresponds to the number of degrees of freedom lost by the trajectory, *i.e.* the trajectory has no components in the directions along which the exhausted (fast) time-scales act, and the dimension of the SIM is $(N+1)-M$. The availability of the CSP kernel and the number of exhausted modes, i.e., the fast/slow decomposition, allows one to compute the cause-effect relationships between the physical processes, e.g. chemical reactions, and the thermo-chemical state variables, distinguishing the processes that constrain the system to the SIM from those that evolve the dynamics along the SIM. To this end, the user may choose to retrieve various sets of importance/participation indices (of a process to a mode and/or a subspace), whose algorithmic description and interpretation is detailed in [4,5]. Also available in `PyCSP` is the Tangential Stretching Rate (TSR) index [23,8], in both the ODE and PDE versions. The latter requires the user to feed the computed diffusion and/or convection terms of the species and energy equations. The TSR is a weighted average of all the eigenvalues, with weights that depend on the mode amplitudes (with or without transport). Based on the value and the sign of the TSR, one can readily characterize the explosive/dissipative nature of the chemical (or chemical-transport) dynamics, identified by the positive/negative sign, and estimate the truly active chemical time scale, which is the inverse of the TSR.

2.1. *Software architecture*

`PyCSP` is written in Python 3. Most of the implemented methods depend on NumPy. The thermo-kinetic data management is performed with `Cantera` [24], which is an open source, object-oriented suite of tools for handling kinetics, thermodynamics and transport models and data. The core objects in the `PyCSP` package are the `CanteraThermoKinetics` and `CanteraCSP` classes. The `CanteraCSP` class inherits attributes and methods from the `CanteraThermoKinetics` class,[2] which is in turn a derived class of Cantera's `Solution` class [24] through multilevel inheritance, and extends them by adding CSP-related class methods. Hence, the constructor for this class is equivalent to the constructor of Cantera's `Solution` class. Indeed, Cantera is employed as a mean to efficiently incorporate detailed chemical thermo-kinetics models into the CSP calculations. This choice was made to ease the chemical model interpretation and to comply with a well-established standard. Moreover, this allows to inherit Cantera's wide availability of model reactors, such as zero-dimensional reactors, one-dimensional flames, reactor networks, multi-phase mixtures and catalytic combustion. More specifically, the two `PyCSP`'s base classes provide the following major features:

- `CanteraThermoKinetics`: computes the chemical source term, the generalized stoichiometric matrix and the numerical Jacobian matrix of the chemical source term, in both the constant pressure and constant volume versions, for a given solution object. In essence, this class assembles kinetics-related quantities which are not directly available in Cantera.
- `CanteraCSP`: computes the CSP kernel (timescales, CSP modes and mode amplitudes), the number of exhausted modes, the tangential stretching rate, the CSP indices (API, TPI, importance indices), the CSP pointers.

Two additional classes are provided, that work on instances of the `CanteraCSP` class, namely:

- `CSPsimplify`: analyzes a database of thermochemical states and generates simplified kinetic mechanisms as described in [21].
- `CSPsolver`: integrates in time the set of stiff chemical ODEs using a Python implementation of the CSP solver described in [25,22]. The CSP solver exploits the local CSP fast/slow decomposition to explicitly and adaptively integrate the slow time scales, while the contribution of the terms producing the fast time scales is taken into account at the end of each integration step as a correction.

Figs. 1 and 2 illustrate the software architecture.

2.2. *Software functionalities*

The `PyCSP` functionalities can be grouped into 3 categories: analysis, model reduction, and time integration. All of the functionalities rely on the CSP kernel, which is a state function, i.e., depends on and requires only a thermo-chemical state, e.g. temperature, pressure (or density), and chemical composition.[3] A Cantera-

[2] The names of the classes emphasize the fact of being sub-classes of `Cantera`'s classes.

[3] An exception is made for the analysis of reaction-transport problems, which requires additional information from the transport operators.

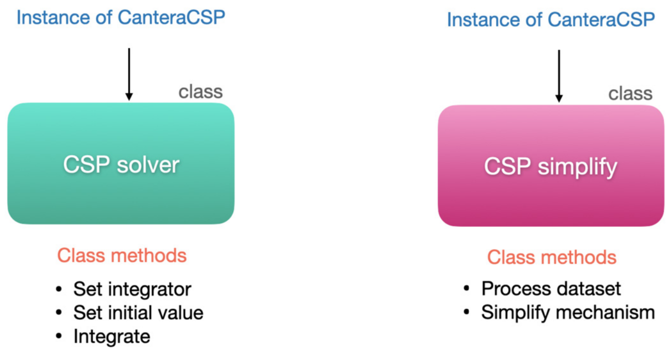


Fig. 2. PyCSP additional classes.

compliant input file, containing the thermo-kinetic model parameters, is required as well.[4] The property of the kernel of being a state function is prescribed in the logical structure of the code: once an instance of `CanteraCSP` is created (calling the constructor `gas = canteraCSP('input.yaml')`), a hard bound with the thermo-chemical state is created as well, and a CSP kernel is associated to a specific state.[5] In other words, the kernel computation is hidden: the user is only allowed to retrieve it, using the `get_kernel()` method, and the returned kernel always conforms to the present state (accessible and editable via the base class attributes, e.g., `gas.Y` and `gas.TP`). When the CSP kernel, or any derived object, is queried, the code computes the kernel only if the state (or any other class attributes, such as the Jacobian options) has been changed. This hard bound was created to avoid inconsistencies among subsequent calls to methods relying on the CSP kernel and to minimize the number of kernel computations. The following code snippet exemplifies the kernel query:

```
import PyCSP.Functions as csp
#instantiate CSP object
gas = csp.CanteraCSP('hydrogen.yaml')
#set the gas state (temperature, pressure, composition)
T = 1000
P = 101325
gas.TP = T, P
gas.set_equivalence_ratio(1.0, 'H2', 'O2:1, N2:3.76')
#set Jacobian options: constant pressure Jacobian
gas.constP = P
#retrieve the CSP kernel:
#eigen-values, right/left eigen-vectors, and amplitudes
lam,R,L,f = gas.get_kernel()
```

Now, suppose to modify the state temperature.

```
#set a new temperature
T = 1100
gas.TP = T, P
#retrieve the CSP kernel:
lam,R,L,f = gas.get_kernel()
```

The `get_kernel()` method will sense the change in an attribute of `gas`, and will recompute the CSP kernel before returning it. All the `PyCSP` functionalities automatically update the CSP kernel, if needed, and operate on the stored kernel directly, so that the user does not need to retrieve it.

The kernel calculation is the eigen-problem[6] of the Jacobian matrix of the chemical source term. The Jacobian matrix calculation requires the user to define the thermodynamic problem type, i.e., either a constant pressure or a constant volume problem (see the `gas.constP` attribute in the code snippet above). Both the Jacobian and the source term are attributes of the `CanteraThermoKinetics` class, which is the `CanteraCSP` parent class, hence they are always accessible as `gas.jacobian` and `gas.source`.

The main `PyCSP` functionalities are described here below, category wise.

2.3. Analysis

The purpose of the CSP analysis is to extract information from the local dynamics of a thermo-chemical state. The most basic information offered by the CSP analysis is the dimension M of the fast subspace, i.e., the number of exhausted modes. The method `calc_exhausted_modes()` returns `M`. This information is already remarkably valuable, because it reveals (i) the number $(N+1)-M$ of true degrees of freedom of the dynamics (the lower M, the higher-dimensional the trajectory, the faster the chemical activity), whose inspection helps to single out the dynamically significant regimes (minima of M); (ii) the fastest time scale of the slow dynamics τ^{M+1} (accessible as the M+1-th element of the timescales array, `gas.tau[M]`).

Many levels of information can be retrieved using the `calc_CSPindices()` method, namely: amplitude/timescale participation indices, fast/slow importance indices, species classification (fast/slow).[7] Another realm of the CSP analysis is the tangential stretching rate (TSR) analysis. The methods `calc_TSR()` and `calc_TSRindices()` are devoted to the computation of the TSR, i.e., a representative of the driving chemical time scale, and the amplitude/timescale participation indices to the TSR.[8]

2.4. Model reduction

The tools offered by the CSP theory, specifically the fast/slow importance indices, are exploited to provide a fully automated algorithm capable of producing families of skeletal mechanisms, tailored on a user-fed database of chemical states. The algorithm, presented in [21] and reported in Fig. 3, retains the most important species/reactions to the local active dynamics, whose kernel is identified by the TSR-active species. The user is only requested to provide a representative database of states and the original detailed mechanism. A series of thresholds can be applied sequentially to generate a family of skeletal mechanisms of higher to lower fidelity and size. The relevant methods of the `CSP-simplify` class are `process_dataset()`, which builds the set of importance indices over the chemical states database as a pre-process, and `simplify_mechanism(thr)`, which outputs species and reactions of the skeletal mechanism resulting from the threshold `thr`.

2.5. Time integration

The availability of the CSP objects makes it straightforward to build an adaptive time-integration scheme for stiff chemistry. The CSP solver, detailed in [22], exploits the projection basis defined by the Jacobian eigen-directions, and the partitioning into

[4] This data is imported from a YAML file. Such files are typically freely available or may be created by converting Chemkin-format files.

[5] Note that the thermo-chemical state is an attribute of the base class `Solution`.

[6] The `numpy.linal.eig` method is used, which is based on LAPACK's _geev.

[7] The mathematical definition and a detailed description of such indices is provided in [4,5]. In short, the participation index quantifies the contribution of a reaction to the amplitude of a mode or to the magnitude of a timescale; the importance index quantifies the contribution of a reaction to the rate of formation/consumption of a species, in either the fast or the slow subspace; the species classification associates a species to either the fast or the slow subspace based on the value of its CSP pointer.

[8] The TSR-API/TPI quantify the participation of a reaction to the driving mode/-timescale.

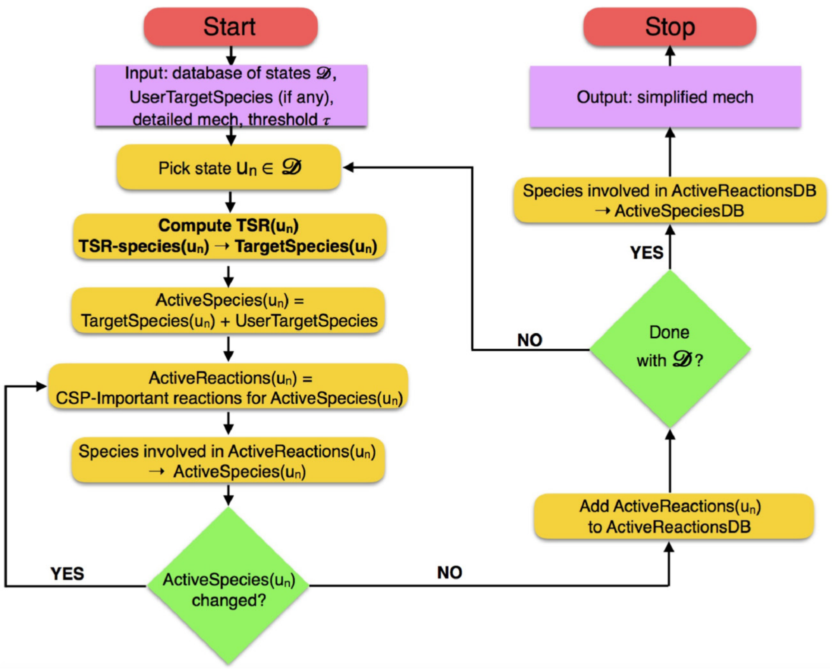

Start
Input: database of states $\mathscr{D}$, UserTargetSpecies (if any), detailed mech, threshold τ
Pick state $u_n \in \mathscr{D}$
Compute TSR(u_n) TSR-species(u_n) → TargetSpecies(u_n)
ActiveSpecies(u_n) = TargetSpecies(u_n) + UserTargetSpecies
ActiveReactions(u_n) = CSP-Important reactions for ActiveSpecies(u_n)
Species involved in ActiveReactions(u_n) → ActiveSpecies(u_n)
YES
ActiveSpecies(u_n) changed?
NO
Add ActiveReactions(u_n) to ActiveReactionsDB
NO
Done with $\mathscr{D}$?
YES
Species involved in ActiveReactionsDB → ActiveSpeciesDB
Output: simplified mech
Stop

Fig. 3. CSP-TSR simplification algorithm [21].

fast and slow components, to build a local non-stiff reduced order model, freed of the fast time-scales, which is explicitly integrated in time with a pace larger than that provided by implicit solvers. The `CSPsolver` class provides an implementation of the scheme for homogeneous problems.[9] The front-end methods are self-explanatory: `set_integrator()` and `set_initial_value()` are used to feed the relevant options and the initial condition, respectively, while `integrate()` advances the state in time by one time-step.

3. Illustrative examples

The package contains multiple usage examples of all the classes. Examples are limited to homogeneous reactors and laminar flames datasets, however one can CSP-analyze and/or simplify feeding any databases of thermochemical states (temperature, pressure, species composition) and flow properties (diffusion and/or convection fluxes) produced in any reacting flow simulations, e.g., laminar and well-resolved turbulent flames, i.e., fine-LES and DNS.[10]

3.1. Basic usage

To create a `CanteraCSP` object, a detailed chemical kinetic mechanism is required. In this example, the detailed chemistry is supplied by a 53-species, 325-reactions, kinetic scheme designed for methane oxidation [26], through the file `gri30.yaml`. The instantiation of a `CanteraCSP` object called `gas` is:

```
import PyCSP.Functions as csp
gas = csp.CanteraCSP('gri30.yaml')
```

Note that the whole set of Cantera's `Solution` class methods are also available.[11] The CSP analysis requires a thermochemical dataset. In the following, the Cantera's ideal gas reactor is employed to generate the data. The CSP analysis is performed on-the-fly, along with the time integration.

```
#set the initial mixture state (with Cantera's methods)
gas.TP = 1000, ct.one_atm
gas.set_equivalence_ratio(1.0, 'CH4', 'O2:1, N2:3.76')

#set  Jacobian  options: constant  volume  Jacobian
rho = gas.density
gas.constRho = rho

#set Cantera constant volume 0-D reactor model
r = ct.IdealGasReactor(gas)
sim = ct.ReactorNet([r])
states = ct.SolutionArray(gas, extra=['t'])

#create storage arrays
evals,Revec,Levec,fvec,M = ([],[],[],[],[])

#integrate ODE with Cantera
sim.set_initial_time(0.0)
while sim.time < 1000:
    sim.step()
    states.append(r.thermo.state, t=sim.time)
    #Perform CSP analysis on current state and append
    lam,R,L,f = gas.get_kernel()
    exhM = gas.calc_exhausted_modes(rtol=1.0e-3,atol=1.0e
     -10)
    evals.append(lam)
    Revec.append(R)
    Levec.append(L)
    fvec.append(f)
    M.append(exhM)
```

The choice of the relative and absolute tolerances for the exhausted modes calculation is up to the user. The lower the tolerances, the more sensitive is the criterion, the less modes will be declared as fast.[12] The reactor behavior in time is reported in Fig. 4. The fundamental CSP observables, namely eigenvalues (inverse of

[9] This implementation can be readily employed in transport/reactions operator-splitting schemes.

[10] The application of the CSP concepts to modeled or averaged simulations datasets requires the availability of the sub-grid contribution to the chemical source term in order to obtain meaningful results.

[11] A complete list and description of such methods is available at https://cantera.org/documentation/index.html, in the Python section.

[12] In the exhausted modes criterion [6], $\delta y^i_{fast} < y^i_{error} = \varepsilon^i_{rel} y^i + \varepsilon^i_{abs}, i = 1, \ldots, N_s + 1$ represent the contribution of the fast modes to the i-th variable rate-of-change. Practical experience suggests that relative tolerances in the range $10^{-2/-4}$ and absolute tolerances in $10^{-8/-12}$ are appropriate choices.

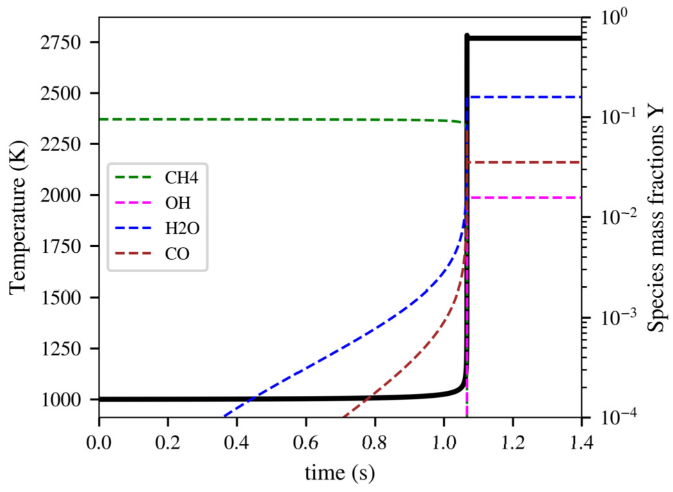


Fig. 4. Temperature (black) and selected species against time. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

timescales) and exhausted modes, are shown in Figs. 5 and 6. To aid the results interpretation, their evolution against the integration timestep counter[13] is also reported, in order to appreciate the dynamics in the rapid (in physical time) transients.

The number of exhausted modes M is an indicator of the local exhausted degrees of freedom of the dynamics. A low M indicates that a large number of modes is active and that chemistry is faster: this occurs during ignition, which is a transition from the initial manifold of dimension ~ 25 to the attractive equilibrium manifold. When approaching equilibrium, M grows to eventually include all modes, meaning that the dynamics is fully exhausted.

The eigenvalues in Fig. 5 are plotted in Λ formulation, following the definition $\Lambda := \text{Sign}(\lambda) * \text{Log}_{10}|1+\lambda|$, to appreciate their magnitude and sign simultaneously. Note that eigenvalues smaller than 10^0 (very slow and conserved modes) are collapsed onto the y=0 axis. Positive eigenvalues are associated to an explosive behavior of the mixture. The $(M+1)$-th eigenvalue evolution is reported in orange. It represents the fastest of the slow timescales. All the eigenvalues with larger magnitude (regardless of the sign[14]) are declared as fast/exhausted. In the plots of Fig. 5, these are the eigenvalues underneath the orange one. The magnitude of the $(M+1)$-th eigenvalue changes according to the value of M. A low value of M implies a faster dynamics, i.e., a larger $(M+1)$-th eigenvalue, and a larger number of slow scales.

The identification of a representative of the driving chemical timescale, i.e., the TSR, will be shown in the following section, which contains more detailed examples of code usage and results interpretation.

3.2. Analysis of a homogeneous reactor

The capabilities of the Tangential Stretching Rate and the related TSR Participation Indices are demonstrated in the analysis of a simple homogeneous problem, involving a hydrogen/air mixture. In particular, two test-cases are investigated: a low-temperature and a high-temperature oxidation. The detailed chemistry is supplied by a 12-species, 33-reactions, kinetic scheme designed for hydrogen and carbon monoxide oxidations [28]. The different initial conditions activate two distinct oxidation paths, known in literature as below- and above-crossover regimes [29–31]. In both cases, the slow initiation reaction, which generates the first radicals, is[15]:

$$H_2 + O_2 \longrightarrow HO_2 + H \qquad \text{(Rb 10)}$$

However, based on the mixture initial temperature, the ignition process is characterized by a competition between the chain branching reaction:

$$H + O_2 \longrightarrow O + OH \qquad \text{(Rf 1)}$$

and the chain terminating three-body collision:

$$H + O_2 + M \longrightarrow HO_2 + M \qquad \text{(Rf 9)}$$

which produces a rather inactive radical (HO_2) instead of two active radicals (O and OH).
Above the *crossover temperature*, which is pressure-dependent ($\sim$950 K at 1 atm for stoichiometric mixtures), the chain branching reaction dominates. However, below the crossover temperature, the two reactions compete and the chain branching process is inhibited. In this low-temperature regime, another oxidation process takes place, namely a thermal runaway caused by the very slow, slightly exothermic reactions:

$$H_2 + HO_2 \longrightarrow H_2O_2 + H \qquad \text{(Rb 18)}$$

$$2\,HO_2 \longrightarrow H_2O_2 + O_2 \qquad \text{(Rf 15)}$$

which generate H_2O_2 and in turn activate the alternative chain branching path:

$$H_2O_2 + M \longrightarrow 2\,OH + M \qquad \text{(Rf 16).}$$

The two analyzed test cases are isobaric (1 atm), stoichiometric H_2/air mixtures at the initial temperatures of 750 K and 1200 K. The datasets are generated using Cantera, which integrates in time the batch reactor model. Table 1 reports the reactions that are referred to in this section.

Table 1
Chemical reactions legend.

R - 1:	$H + O_2 \longleftrightarrow O + OH$
R - 2:	$O + H_2 \longleftrightarrow H + OH$
R - 3:	$H_2 + OH \longleftrightarrow H_2O + H$
R - 8:	$H + OH + M \longleftrightarrow H_2O + M$
R - 9:	$H + O_2 + M \longleftrightarrow HO_2 + M$
R - 10:	$H_2 + O_2 \longleftrightarrow HO_2 + H$
R - 15:	$2\,HO_2 \longleftrightarrow H_2O_2 + O_2$
R - 16:	$H_2O_2 + M \longleftrightarrow 2\,OH + M$
R - 18:	$H_2 + HO_2 \longleftrightarrow H_2O_2 + H$

The PyCSP code syntax for this example is reported in the following, supposing that a dataset of states $\{T, P, Y_i\}$ is available, e.g., has been produced using Cantera's `IdealGasConstPressureReactor`:

```
#create CanteraCSP object from mechanism file
gas = csp.CanteraCSP('hydrogen.yaml')

#set constant pressure Jacobian
gas.constP = P

#for each state in the dataset, compute CSP quantitites:
for state in dataset:
    gas.set_stateYT(state)
```

[13] Cantera's default stiff-ODE's time-integrator is CVODE [27]. The methods used in CVODE are variable-order, variable-step multistep methods, hence the counter shows a nonlinear mapping to the physical time. Note that smaller steps are taken in the fastest transients of the dynamics.
[14] It can be shown that such eigenvalues are always negative.

[15] The reactions numbering follows the kinetic mechanism ordering and is summarized in Table 1.

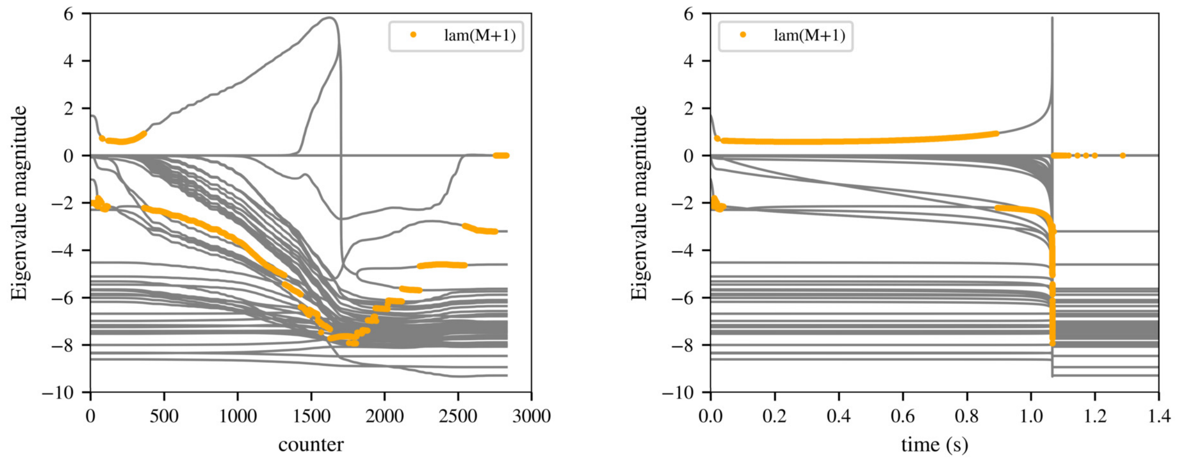


Fig. 5. Eigenvalues (grey, in Λ formulation), M+1-th eigenvalue (orange) against number of integration timestep (left) and time (right).

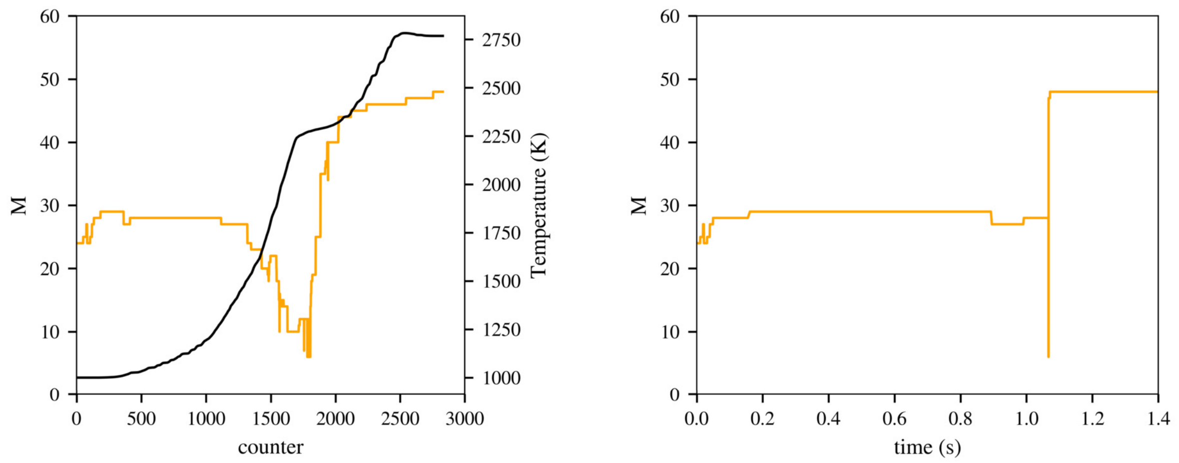


Fig. 6. Number of exhausted modes (orange) against number of integration timestep (left) and time (right). Temperature is also reported in (left).

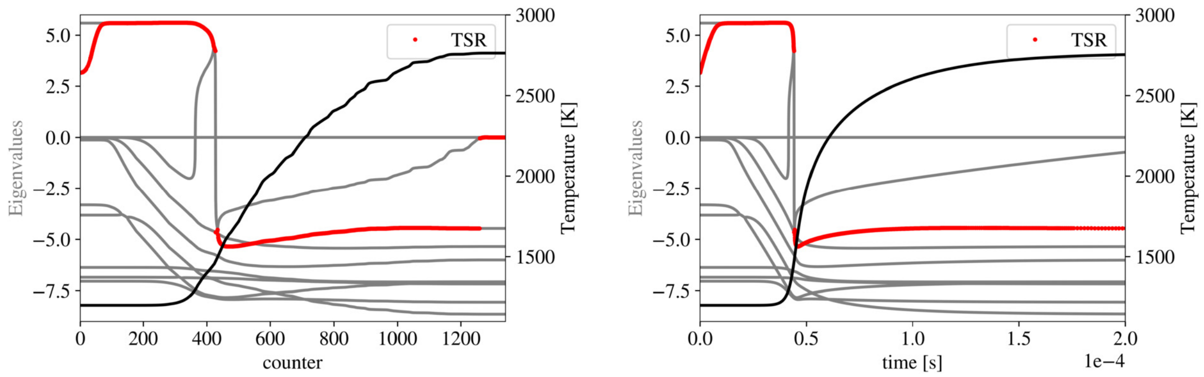


Fig. 7. Eigenvalues (grey, in Λ formulation), TSR (red, in Ω formulation) and temperature evolution against number of integration timestep (a) and time (b). Above crossover regime.

```
    lam, R, L, f = gas.get_kernel()
    NofDM = gas.calc_exhausted_modes(rtol=1.0e-4,atol=1.0e
     -12)
    tsr = gas.calc_TSR()
    tsrapi = gas.calc_TSRindices(type='amplitude')
```

3.2.1. *Above crossover auto-ignition*

Fig. 7 shows the systems' eigenvalues evolution against integration time-step (left) and time (right) for the T_0=1200 K case. The eigenvalues are plotted in Λ formulation, following the definition $\Lambda := \mathrm{Sign}(\lambda) * \mathrm{Log}_{10}|1+\lambda|$. Typical features of the eigenvalues can be observed: (i) their magnitude (and sign) changes in time because the problem is non-linear; (ii) a couple of positive eigenvalues exists, whose merging happens approximately in correspondence of the maximum temperature rate of increase, *i.e.* the ignition delay time, which is $\tau_{ign} \approx 4.5 \times 10^{-5}$; (iii) 4 dormant modes, associated to 4 zero eigenvalues[16] (collapsed onto

[16] The numerical values may slightly differ from an exact zero because a double precision representation can properly resolve eigenvalues only in a range that spans approximately fifteen orders of magnitude, hence the smallest (dormant) eigenval-

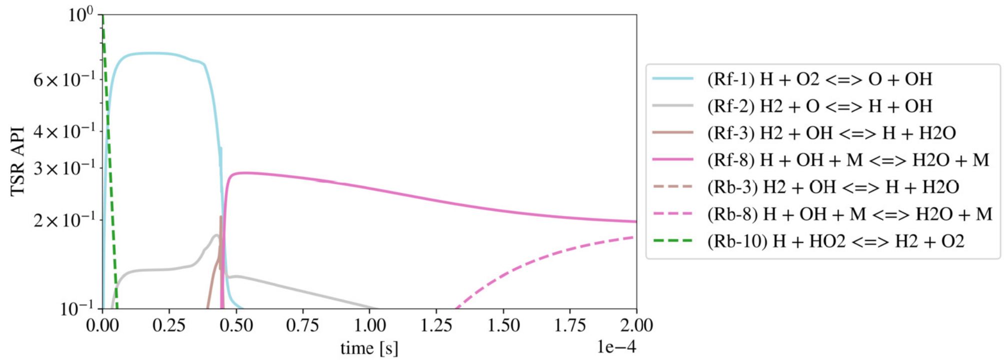


Fig. 8. TSR Participation Indices against time. Above crossover regime.

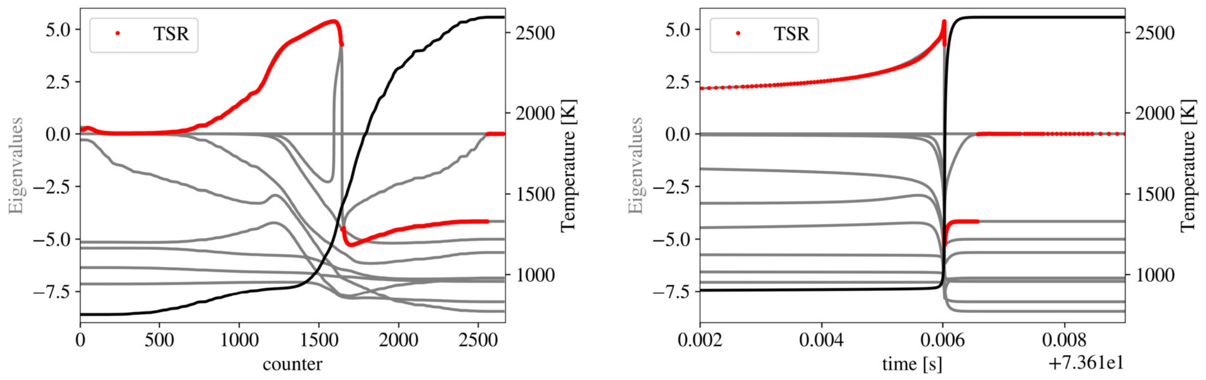


Fig. 9. Eigenvalues (grey, in Λ formulation), TSR (red, in Ω formulation) and temperature evolution against number of integration timestep (a) and time (b). Below crossover regime.

the zero axis in the plots), corresponding to the 4 conservation laws (3 elements plus enthalpy), make the total number of meaningful eigenvalues equal to 9. The conservation laws, in fact, induce a rank-deficiency in the Jacobian matrix, which translates into zero eigenvalues.

The TSR, in the same logarithmic formulation employed for the eigenvalues, is plotted as a red symbol on top of the eigenvalues. With the exception of the very first time-steps, the TSR tracks the positive eigenvalue, denoting the explosive propensity of the system, whose characteristic time scale is O($1/10^6$) s. After the merging, the TSR tracks the eigenvalues associated to mode #8 first, and mode #9 then, following it until equilibrium. This means that the system evolves and approaches equilibrium according to the dissipative scale equal to $1/\lambda_9$. Mode #9 is also the unique slow mode (M=8, not shown) from time-step #1300 to equilibrium.

The extremely short ignition delay time and the explosive nature of the system since the very beginning are clues of the above-crossover regime of this test-case. However, more insights on the physics of the problem may be obtained by resorting to the TSR participation indices, which identify the processes involved in the energy carrying, TSR-important, modes, *i.e.* the dominant reactions. Fig. 8 shows the TSR participation indices against time. As expected, in the very first stages Rb-10 is the dominant reaction, being the initiation step. Immediately after, the chain branching reaction Rf-1 takes the lead ($\sim$ 70 %), while the chain terminating reaction Rf-9 has a negligible participation index (not shown). This behavior is typical of the above-crossover regime. To a minor extent, chain branching reaction Rf-2 ($O + H_2 \longrightarrow H + OH$) and, very close to ignition (t $\approx$ 4e-5 *s*), chain carrying reaction Rf-3 ($H_2 + OH \longrightarrow H_2O + H$) share the remaining $\sim$ 30 % of importance. These three reactions are known as *shuffle* reactions, and constitute the submechanism that describes the rapid H-O-OH radical conversion in the radical pool.

After ignition, the recombination reaction Rf-8 ($H + OH + M \longrightarrow H_2O + M$) becomes the leading process, which becomes counterbalanced by its reverse reaction (Rb-8) when approaching equilibrium.

3.2.2. Below crossover auto-ignition

Fig. 9 shows the systems' eigenvalues evolution against integration time-step (left) and time (right) for the T_0=750 K case. Differently to the previous case, the positive eigenvalue becomes large with a long delay, and the ignition delay itself is extremely large ($\tau_{ign} \approx$ 73.6 s). However, the post-ignition behavior of the eigenvalues resembles the previous case one. Again, TSR tracks the positive eigenvalue until merging, then follows λ_9. More insights may be obtained from Fig. 10, that shows the time evolution of the TSR participation indices, plotted with 3 different degrees of magnification, so as to appreciate their evolution far and close to ignition. As expected, the initiation reaction Rb-10 is again the leading process in the first instants. Then, reaction Rb-18 takes the lead, followed by Rf-16 and Rf-15, perfectly resembling the alternative ignition path, typical of the below-crossover regime. Noteworthy, Rf-1 and Rf-9 have similar importance indices, since they compete, preventing the chain branching process typical of high-temperature mixtures. Then, after a long time delay, Rf-1 eventually becomes the most important process, in correspondence of the maximum temperature increase and positive eigenvalues merging. Fig. 11 shows the same indices plotted against the number of integration timestep, to allow a comparison with the eigenvalues/TSR evolution of Fig. 9 (left). Interestingly, the positive eigenvalue starts increasing around time-step #900, which corresponds to the couple Rf-1/Rf-

ues are too close to zero to be resolvable within machine accuracy [32]. However these are always negligibly small and their number is known a-priori, being equal to the number of elements.

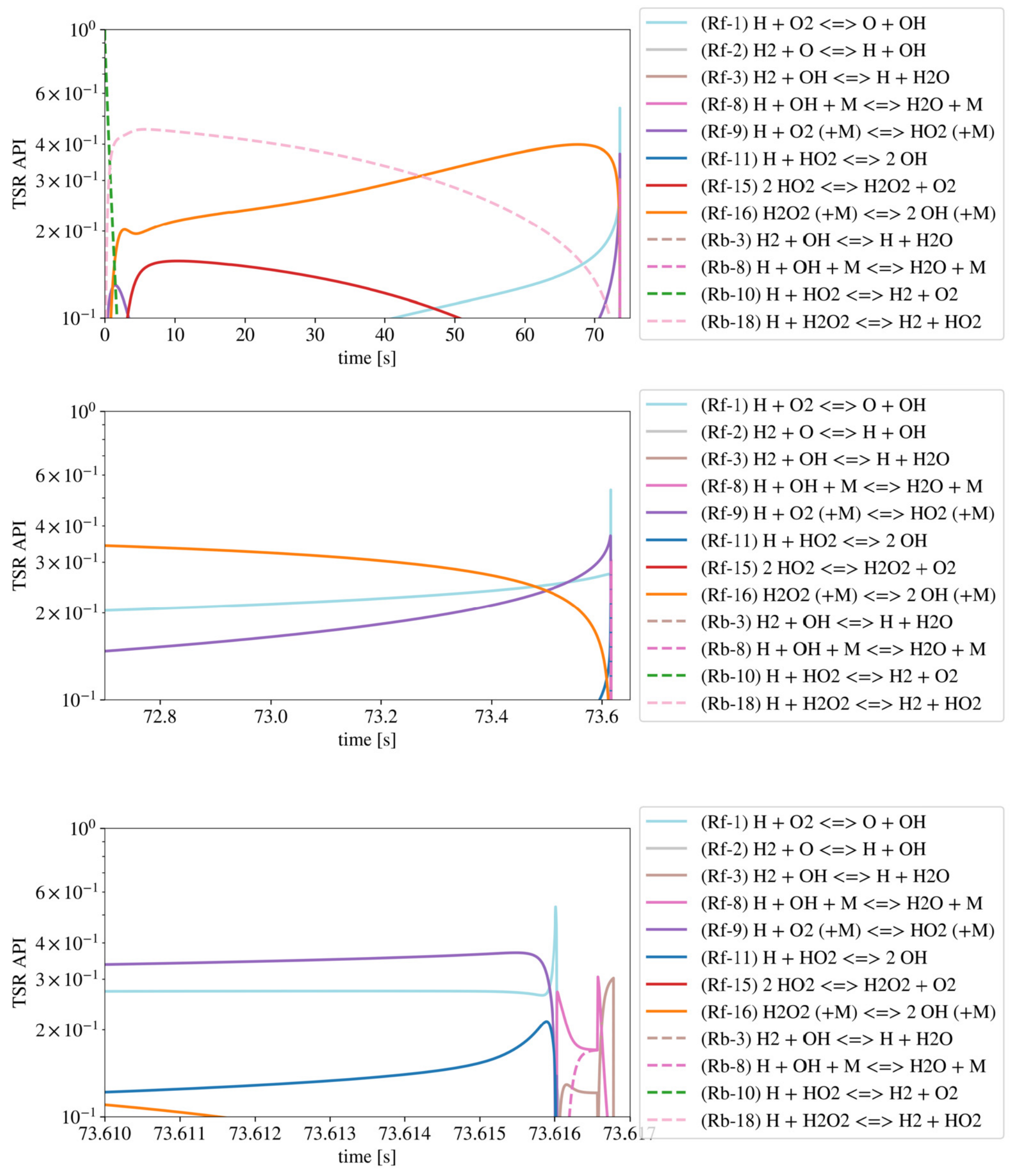


Fig. 10. TSR Participation Indices against time, with different degrees of magnification. Below crossover regime.

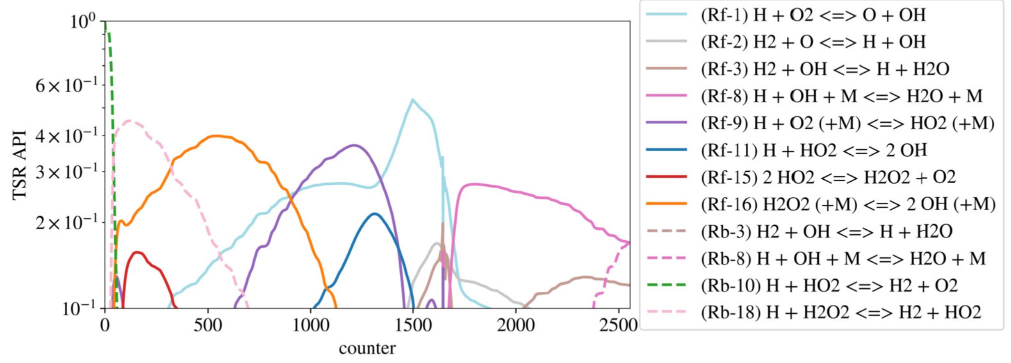


Fig. 11. TSR Participation Indices against number of time-step. Below crossover regime.

9 overtaking Rf-16 as the most important reactions. Interestingly, the sharpest positive eigenvalue growth happens between time-steps #1000 and #1400, where the chain terminating reaction Rf-9 is the leading process, with Rf-1 slightly less important. However, the maximum temperature rate and the eigenvalues merging correspond to Rf-1 being the reaction with the largest participation index.

In conclusion, the TSR analysis allowed to reveal the active chemical time scales and the reactions participating to the active modes. The latter are in complete agreement with the physical ex-

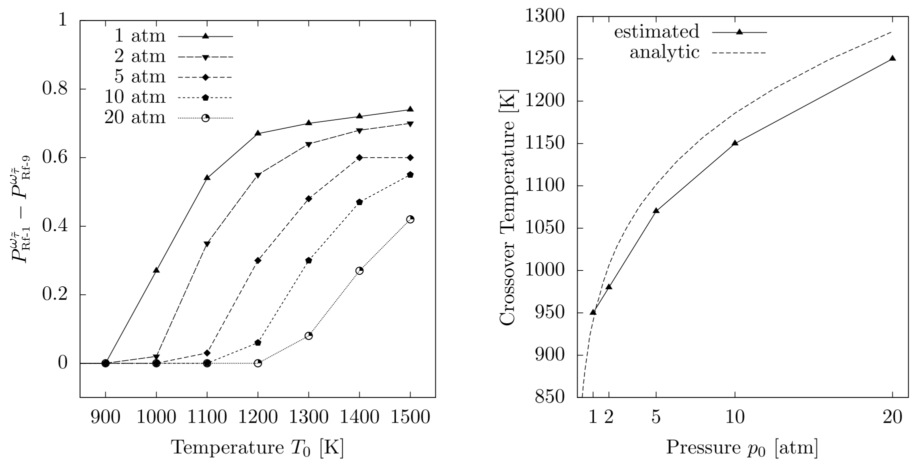


Fig. 12. Difference between $P^{\omega_{\bar{\tau}}}_{\text{Rf-1}}$ and $P^{\omega_{\bar{\tau}}}_{\text{Rf-9}}$ with increasing initial temperature T_0, for different pressure values p_0 (left); estimated crossover temperature against the analytical expression proposed in [33] (right).

pectations in both cases, *i.e.* above and below crossover, where substantially different chemical pathways are taken. In addition, the TSR participation indices may be even used to identify the regimes. For example, by inspecting the relative participation of reactions Rf-1 and Rf-9, one can infer whether the mixture experienced an above- or below-crossover ignition, and which is the crossover temperature, which is a function of pressure. Fig. 12 (left) shows the difference between $P^{\omega_{\bar{\tau}}}_{\text{Rf-1}}$ and $P^{\omega_{\bar{\tau}}}_{\text{Rf-9}}$, averaged over the branching region of the auto-ignition, computed for a set of stoichiometric batch reactor solutions obtained at different initial temperature and pressure. For a fixed pressure, it can be observed that the two indices are equal, *i.e.* their difference is zero, up to a certain initial temperature, where they start to differ in favor of a larger $P^{\omega_{\bar{\tau}}}_{\text{Rf-1}}$. When the two are equal, the regime may be labeled as below-crossover, while $P^{\omega_{\bar{\tau}}}_{\text{Rf-1}}$ larger than $P^{\omega_{\bar{\tau}}}_{\text{Rf-9}}$ is a typical behavior encountered in above-crossover regimes. The temperature at which the regime changes may be denoted as crossover temperature. Note that, increasing pressure, the initial temperature where reaction Rf-1 starts to participate more than Rf-9, *i.e.* the crossover temperature, shifts towards higher temperatures. This behavior is clearly visible in Fig. 12 (right), which resembles the analytic expression for the crossover temperature defined in [33].

3.3. Analysis of a laminar flame

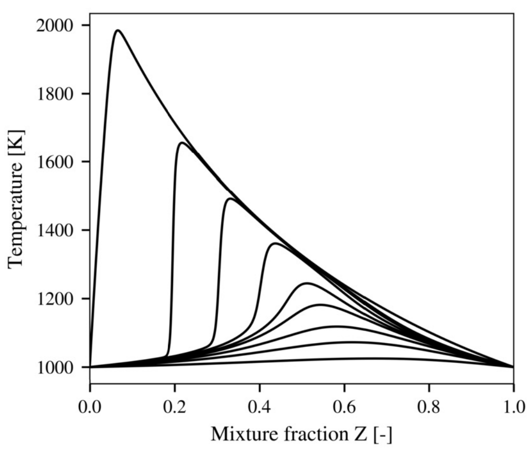
The recognition of patterns in multi-dimensional reacting flows, e.g. explosive layers and flame fronts, is typically done manually, by visual inspection of primitive state variables, such as Temperature or radical species concentrations. The CSP analysis, instead, is capable of returning a topology of the reacting field based on CSP observables, such as the number of exhausted modes or the TSR, which can be used for feature identification and tracking. It is hereby reported an example of how to perform a TSR analysis of pre-computed data of the ignition of an unsteady laminar flamelet, evolving in time in the mixture fraction space. The unsteady laminar flamelet model represents the competition between chemical kinetics and molecular diffusion processes in a flame structure which is locally one-dimensional, and depends only on time and on the coordinate normal to the flame front. This description is the basis of many models for turbulent combustion, under the hypothesis that the flame is thin compared to other flow and wrinkling scales. In place of the physical normal coordinate, the mixture fraction z is commonly employed, which measures the local fuel/oxidizer ratio. The mapping between the physical space and the mixture fraction space is embedded in the *scalar dissipation rate* $\chi := 2\mathcal{D}(\partial z/\partial x)^2$, which controls the diffusive terms in the flamelet evolution equations [1].

The dataset consists of thermochemical states (species and temperature, `stateYT`), and diffusive fluxes of species and temperature (`diffY` and `diffT`), the latter needed to compute the extended (to transport) TSR. In general, this analysis can be performed supplying either the diffusive fluxes, the convective fluxes, or both simultaneously. The kinetic mechanism [34], containing 12 species and 33 reversible reactions, is designed for syngas combustion. More details on the dataset and the results can be found in [8]. The time evolution of the flamelet temperature is reported in Fig. 13 (left): the initial condition (1000 K uniform temperature) ignites and evolves towards the steady solution with a peak temperature of 2000 K. The scalar field of the HCO mass fraction, which is a known marker of the flame front, is shown in the Z-time space in Fig. 13 (right). A visual inspection of the flame front does not highlight any peculiar role of transport in the ignition dynamics.

The relevant code lines for the TSR analysis of this dataset are:

```
#loop over data points
for i in range(begin,end):
    gas.constP = Pressure[i]
    stateYT = np.append(Y[i],Temp[i])
    rhsdiffYT = np.append(diffY[i],diffTemp[i])
    gas.set_stateYT(stateYT)
    TSR, NofDM = gas.calc_TSR(getM=True)
    TSRext, NofDMext = gas.calc_extended_TSR(getMext=True,
     diff=rhsdiffYT)
```

The main outcomes of the analysis are the scalar fields of M (exhausted modes) and TSR, in both the chemical and extended versions. As detailed in [8], the comparison of the CSP objects related to the chemical source term alone and to the chemical plus transport (i.e., extended) right hand side, sheds light on the role of transport on the ignition dynamics. Fig. 14 shows one possible visualization of the TSR fields. A positive TSR is a marker for chemical ignition, while a positive extended TSR is a marker for transport-supported ignition, i.e. where diffusion plays a significant role in the explosive regime (deflagration front). In the specific case presented in [8] and shown in Fig. 14, ignition starts at the most-ignitable mixture fraction ($Z_{ign} \simeq 0.5$) and is initially chemically-driven (green area in Fig. 14), then, as soon as kinetics creates spatial non-uniformities, diffusion steps in and propagates heat and mass outwards with respect to Z_{ign}. The reaction–diffusion wave (red area in Fig. 14) travels towards leaner mixtures and a peak temperature is reached at $Z \simeq 0.055$.



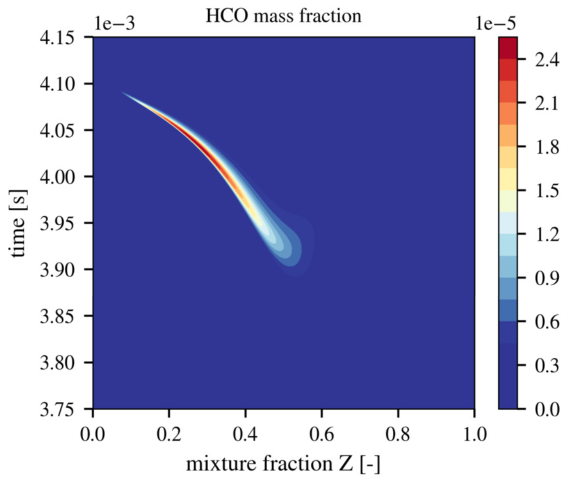


Fig. 13. Time evolution of Temperature in the mixture fraction space. The represented snapshots are taken at t = (3, 3.6, 3.8, 3.91, 3.95, 4, 4.06, 4.1, 5) $\times$ 10^{-3} s (left); scalar field of the HCO mass fraction in the Z-time space (right).

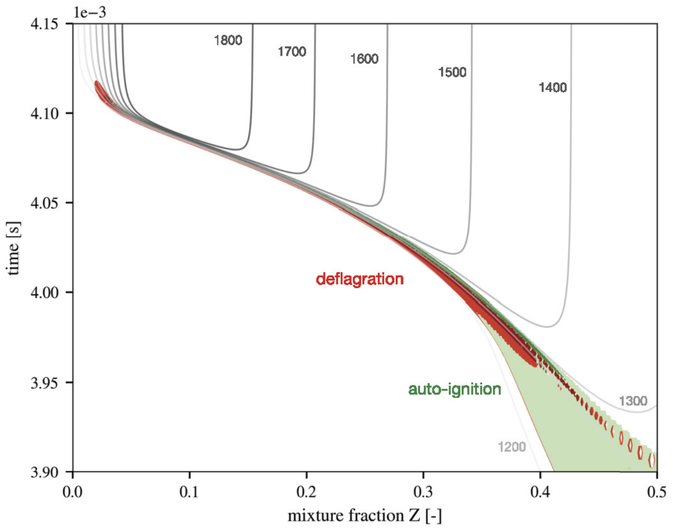


Fig. 14. Time evolution of a laminar flamelet in the Z-time space. Locus of positive chemical TSR (green), of positive extended TSR (red), contour-lines of Temperature (gray).

3.4. Simplification of a detailed kinetic mechanism

A typical mechanism simplification workflow using the `CSPsimplify` class is demonstrated in the size reduction of the 53-species GRI3.0 chemical reaction mechanism [26]. The simplification task is performed by feeding the algorithm with a database of thermochemical states describing the time evolution of a constant pressure homogeneous reactor initialized with a stoichiometric mixture of methane CH_4 and air at 1000 K at ambient pressure.

The dataset is generated with Cantera. The `CSPsimplify` class object `simplifier` is then created and associated to this dataset:

```
import PyCSP.Functions as csp
import PyCSP.Simplify as simp
dtl_mech = csp.CanteraCSP('gri30.yaml')
#init simplifier
simplifier = simp.CSPsimplify(dtl_mech,dataset)
```

The simplification settings are specified afterwards. Specifically, the TSR is in charge of automatically identifying the local target species. The inert species N_2 is manually added to this set. The detailed listing and description of all the settings is available in the user's manual.

```
#simplifier settings
simplifier.TSRtargetset = True
simplifier.TSRtol = 0.5
simplifier.targetset = {'N2'}
simplifier.problemtype = 'constP'
simplifier.scaled = False
simplifier.csprtol = 1.0e-2
simplifier.cspatol = 1.0e-8
```

The dataset processing consists in the calculation and storage (in dedicated class attributes) of the fast/slow importance indices:

```
simplifier.process_dataset()
```

The simplification algorithm is finally launched with a given threshold `thr` and the simplified mechanism is then created by instantiating a new `CanteraCSP` object:

```
species, reactions = simplifier.simplify_mechanism(thr)
simp = csp.CanteraCSP(thermo='IdealGas', kinetics='
    GasKinetics', species=species, reactions=reactions)
```

The execution of the `simplify_mechanism` method for a range of thresholds between 0 and 1 returns the simplified mechanisms whose errors on the ignition delay time with respect to the detailed mechanism (in the same ignition problem) are shown in Fig. 15 (left) as a function of the number of retained species. The ignition delay time is computed as the peak heat release rate. As the figure shows, the smaller mechanism obtained has 19 species and an error of 3% on the ignition delay time. Note that the error is non-monotonic, due to the effect of partially removing clusters of species. Comprehensive tests are advisable on homogeneous problems having different initial conditions than the training dataset one, such as the one shown in Fig. 15 (right), which depicts the ignition delay error of the 19-species skeletal mechanism over a range of conditions.

3.5. Time integration of a homogeneous reactor

This example shows how to integrate a homogeneous reactor with the `CSPsolver` class.

```
import PyCSP.Solver as cspS

#create an instance of CSPsolver (gas is an instance of
    CanteraCSP)
solver = cspS.CSPsolver(gas)
solver.set_integrator(cspRtol=1e-2,cspAtol=1e-8,factor=0.2)
solver.set_initial_value(y0,t0)

#advance in time with CSP solver
```

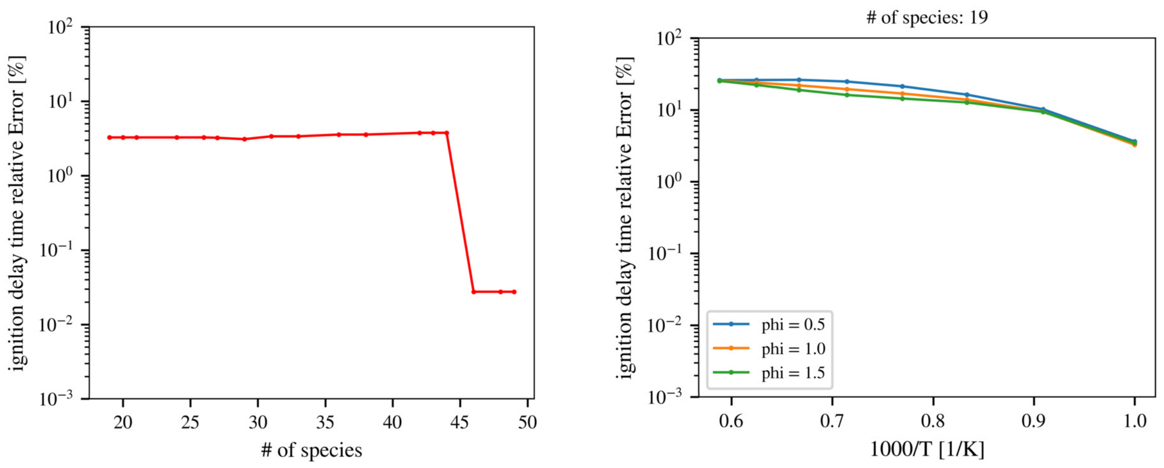


Fig. 15. Relative error on ignition delay time versus number of retained species (left); error of the 19-species mechanism in different temperature T and equivalence ratio ϕ conditions (right).

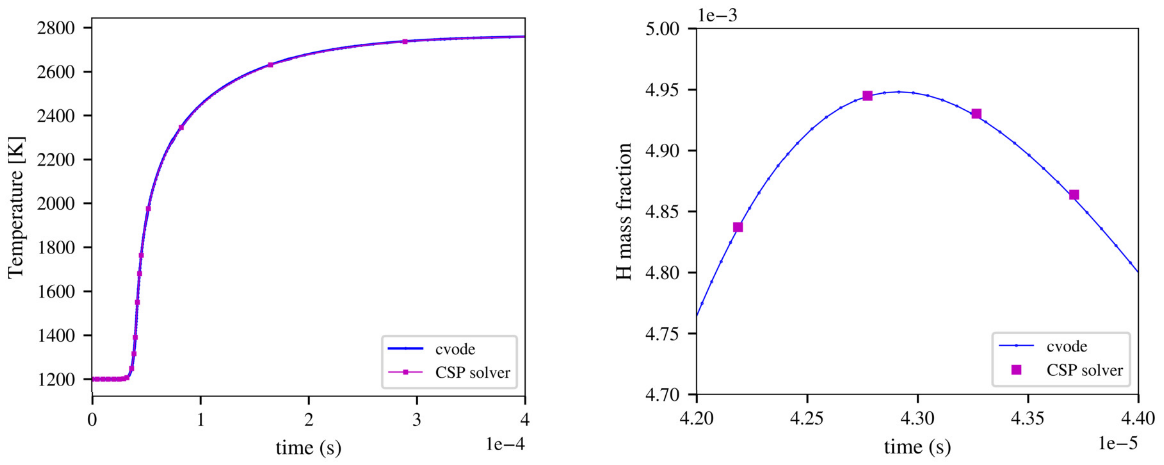


Fig. 16. Temperature and H mass fraction as solutions of a homogeneous reactor, obtained with CVODE (blue) and CSP solver (magenta).

```
while solver.t < t_end:
    solver.integrate()
```

The relevant difference with classic `Cantera`'s integration is the `set_integrator` method, which is used to set the CSP solver parameters, namely the exhausted modes tolerances and a safety factor γ on the time step magnitude, which is automatically estimated. Each call to `integrate()` performs the following series of operations (see [22] for more details): it computes the CSP kernel, it applies a first algebraic radical correction to approach the local manifold, it evaluates the number of exhausted modes, it builds the slow projection matrix, it integrates the slow dynamics with a 4-th order Runge-Kutta explicit scheme with a time step $dt = \gamma\, \tau^{M+1}$, it computes a second algebraic radical correction to approach the new manifold and obtain the state at time $t + dt$. Fig. 16 shows homogeneous reactor ODE solutions obtained with the CSP solver versus the `Cantera`'s implementation of CVODE. The initial mixture is ambient pressure, stoichiometric, hydrogen/air at 1200 K. The CSP solver is as accurate as CVODE, but advances in time with larger time steps.

4. Conclusions and future developments

`PyCSP` is a Python package which provides tools for the analysis and reduction of chemically reacting systems. It is aimed at researchers in the field of chemistry, combustion and engineering, who face the need to synthesize the information contained in massive simulation datasets and/or reduce the computational expense of such simulations using reduced order models. `PyCSP` exploits advanced algorithms based on computational singular perturbation to offer opportunities for feature identification and model reduction.

The `PyCSP` user is presented with several application examples, allowing rapid computation and investigation of typical cases, such as ideal reactors and multi-dimensional flames, facilitated by the inheritance of `Cantera`'s `Solution` class methods.

The CSP theory has already been employed to analyze other categories of multi-scale systems, e.g., biological kinetics models, pharmacokinetics [35–37], and catalytic systems [38], while other research fields might benefit in the future from the application of the CSP analysis, such as the phoretic self-propulsion in synthetic micro-swimmers [39,40]. Computational models for such systems share obvious similarities with chemical models for the oxidation of fuels. The `PyCSP` tools are readily extendable to the analysis of biological systems, provided that a kinetic mechanism description is available in a `Cantera`-compliant format. Moreover, the `PyCSP` package is potentially extendable in the following directions:

- The implementation of algorithms, e.g., based on Artificial Intelligence, devoted to the automatic classification/recognition of reacting flow patterns, supported by the outcomes of a CSP analysis, is advisable.

- The CSP observables are state functions. The training of an artificial neural-network (ANN) capable of predicting the TSR would allow, in example, a cheap on-the-fly calculation of the chemical driving timescale.
- The CSP analysis is intrinsically local, making it favorable to distribute the analysis of many state points to different processors. This practice would be of major interest when the analyzed dataset consists of millions or billions of data points, such as in reacting DNS.

Contributions in the aforementioned directions are warmly welcome.

Declaration of competing interest

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

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 801505. The author acknowledges the inestimable support of Prof. Mauro Valorani and Prof. Pietro Paolo Ciottoli. `PyCSP` has its roots in the Fortran code CSPTk, developed by M. Valorani, H. Najm, D. Goussis, S. Paolucci, F. Creta, E. Martelli, T. Grenga, P.P. Ciottoli, and the author of this manuscript.

References

[1] T. Poinsot, D. Veynante, Theoretical and Numerical Combustion, RT Edwards Inc., Philadelphia, PA, 2005.
[2] S.H. Lam, Combust. Sci. Technol. 89 (1993) 375, https://doi.org/10.1080/00102209308924120.
[3] U. Maas, S.B. Pope, Combust. Flame 88 (1992) 239, https://doi.org/10.1016/0010-2180(92)90034-M.
[4] M. Valorani, F. Creta, P.P. Ciottoli, R. Malpica Galassi, D.A. Goussis, H.N. Najm, S. Paolucci, H.G. Im, E.-A. Tingas, D.M. Manias, et al., Data Analysis for Direct Numerical Simulations of Turbulent Combustion: From Equation-Based Analysis to Machine Learning, Springer International Publishing, Cham, ISBN 978-3-030-44718-2, 2020, pp. 43–64.
[5] M. Valorani, F. Creta, P.P. Ciottoli, R. Malpica Galassi, D.A. Goussis, H.N. Najm, S. Paolucci, H.G. Im, E.-A. Tingas, D.M. Manias, et al., Data Analysis for Direct Numerical Simulations of Turbulent Combustion: From Equation-Based Analysis to Machine Learning, Springer International Publishing, Cham, ISBN 978-3-030-44718-2, 2020, pp. 65–88.
[6] M. Valorani, P.P. Ciottoli, R.M. Galassi, S. Paolucci, T. Grenga, E. Martelli, Flow Turbul. Combust. 101 (2018) 1023, https://doi.org/10.1007/s10494-018-9942-2.
[7] M. Valorani, S. Paolucci, P.P. Ciottoli, R. Malpica Galassi, Combust. Theory Model. 21 (2017) 137, https://doi.org/10.1080/13647830.2016.1243733.
[8] M. Valorani, P.P. Ciottoli, R. Malpica Galassi, Proc. Combust. Inst. 36 (2017) 1357, https://doi.org/10.1016/j.proci.2016.09.008.
[9] P. Pal, M. Valorani, P.G. Arias, H.G. Im, M.S. Wooldridge, P.P. Ciottoli, R. Malpica Galassi, Proc. Combust. Inst. 36 (2017) 3705, https://doi.org/10.1016/j.proci.2016.07.059.
[10] Z. Li, R. Malpica Galassi, P.P. Ciottoli, A. Parente, M. Valorani, Combust. Flame (ISSN 0010-2180) 208 (2019) 281, https://doi.org/10.1016/j.combustflame.2019.06.023.
[11] P.P. Ciottoli, R.M. Galassi, P.E. Lapenna, G. Leccese, D. Bianchi, F. Nasuti, F. Creta, M. Valorani 186 (2017) 83, https://doi.org/10.1016/j.combustflame.2017.07.035.
[12] R. Malpica, M. Valorani, H.N. Najm, C. Safta, M. Khalil, P.P. Ciottoli, Combust. Flame 179 (2017) 242, https://doi.org/10.1016/j.combustflame.2017.02.018.
[13] D.M. Manias, E.-A. Tingas, F.E. Hernández Pérez, R. Malpica Galassi, P. Paolo Ciottoli, M. Valorani, H.G. Im, Combust. Flame 200 (2019) 155, https://doi.org/10.1016/j.combustflame.2018.11.023, ISSN 00102180.
[14] W. Song, E.-a. Tingas, H.G. Im, Combust. Flame (ISSN 0010-2180) 195 (2018) 84, https://doi.org/10.1016/j.combustflame.2018.03.037.
[15] Z. Li, S. Tomasch, Z.X. Chen, A. Parente, I.S. Ertesvåg, N. Swaminathan, Proc. Combust. Inst. 000 (1) (2020), https://doi.org/10.1016/j.proci.2020.06.298, ISSN 15407489.
[16] D.M. Manias, E.-A. Tingas, Y. Minamoto, H.G. Im, Combust. Flame (ISSN 0010-2180) 208 (2019) 86, https://doi.org/10.1016/j.combustflame.2019.06.031.
[17] D.M. Manias, A.-E. Tingas, H.G. Im, Y. Minamoto, in: AIAA Scitech 2019 Forum, 2019.
[18] F.E.H. Perez, H.G. Im, A.E. Tingas, in: AIAA Scitech 2020 Forum, 2020.
[19] A.S. AlRamadan, R.M. Galassi, P.P. Ciottoli, M. Valorani, S.M. Sarathy, Combust. Flame 219 (2020) 242, https://doi.org/10.1016/j.combustflame.2020.05.026.
[20] E.A. Tingas, D.C. Kyritsis, D.A. Goussis, Combust. Flame 162 (2015) 3263, https://doi.org/10.1016/j.combustflame.2015.05.016.
[21] R. Malpica Galassi, P.P. Ciottoli, S.M. Sarathy, H.G. Im, S. Paolucci, M. Valorani, Combust. Flame 197 (2018) 439, https://doi.org/10.1016/j.combustflame.2018.08.007, ISSN 00102180.
[22] R. Malpica Galassi, P.P. Ciottoli, M. Valorani, H.G. Im, J. Comput. Phys. 451 (2022), https://doi.org/10.1016/j.jcp.2021.110875.
[23] M. Valorani, S. Paolucci, E. Martelli, T. Grenga, P.P. Ciottoli, Combust. Flame 162 (2015) 2963, https://doi.org/10.1016/j.combustflame.2015.05.015.
[24] D.G. Goodwin, R.L. Speth, H.K. Moffat, B.W. Weber, Cantera: an object-oriented software toolkit for chemical kinetics, thermodynamics, and transport processes, https://www.cantera.org, 2018, version 2.4.0, https://doi.org/10.5281/zenodo.1174508.
[25] M. Valorani, D.A. Goussis, J. Comput. Phys. 169 (2001) 44, https://doi.org/10.1006/jcph.2001.6709.
[26] G.P. Smith, D.M. Golden, M. Frenklach, N.W. Moriarty, B. Eiteneer, M. Goldenberg, C.T. Bowman, R.K. Hanson, S. Song, W.C. Gardiner, V. Lissianski, Z. Qin, http://combustion.berkeley.edu/gri-mech/, 2009.
[27] S.D. Cohen, A.C. Hindmarsh, Comput. Phys. (ISSN 0894-1866) 10 (1996) 138–143, https://doi.org/10.1063/1.4822377.
[28] J. Li, Z. Zhao, A. Kazakov, M. Chaos, F.L. Dryer, J.J.J. Scire, Int. J. Chem. Kinet. 39 (2007) 109, https://doi.org/10.1002/kin.20218.
[29] P. Boivin, C. Jimenez, A.L. Sànchez, F.A. Williams, Proc. Combust. Inst. 33 (2011) 517, https://doi.org/10.1016/j.proci.2010.05.002.
[30] P. Boivin, A.L. Sànchez, F.A. Williams, Combust. Flame 159 (2012) 748, https://doi.org/10.1016/j.combustflame.2011.08.019.
[31] C. Trevino, Prog. Astronaut. Aeronaut. 131 (1991) 19.
[32] J.C. Lee, H.N. Najm, S. Lefantzi, J. Ray, M. Frenklach, M. Valorani, D.A. Goussis, Combust. Theory Model. 11 (73) (2007), https://doi.org/10.1080/13647830600763595.
[33] P. Boivin, A.L. Sànchez, F.A. Williams, Combust. Flame 176 (2017) 489, https://doi.org/10.1016/j.combustflame.2016.11.008.
[34] J. Lu, Z. Zhao, A. Kazakov, M. Chaos, F.L. Dryer, J.J. Scire, Int. J. Chem. Kinet. 39 (2007) 109, https://doi.org/10.1002/kin.20218.
[35] D.G. Patsatzis, E.-A. Tingas, D.A. Goussis, S.M. Sarathy, PLoS ONE 14 (2019) 1, https://doi.org/10.1371/journal.pone.0226094.
[36] D.G. Patsatzis, D.T. Maris, D.A. Goussis, Bull. Math. Biol. 78 (1121) (2016), https://doi.org/10.1007/s11538-016-0176-y.
[37] I. Surovtsova, N. Simus, K. Hübner, S. Sahle, U. Kummer, BMC Syst. Biol. 6 (2012), https://doi.org/10.1186/1752-0509-6-14.
[38] O. Díaz-Ibarra, K. Kim, C. Safta, J. Zádor, H.N. Najm, Combust. Theory Model. 0 (2021) 1, https://doi.org/10.1080/13647830.2021.2002417.
[39] Y. Ibrahim, R. Golestanian, T.B. Liverpool, J. Fluid Mech. 828 (2017) 318–352, https://doi.org/10.1017/jfm.2017.502.
[40] J.L. Moran, J.D. Posner, Annu. Rev. Fluid Mech. 49 (2017) 511, https://doi.org/10.1146/annurev-fluid-122414-034456.