The OpenSMOKE++ Suite
Version 0.4

May 22, 2016

OpenSMOKE++ is a general framework for numerical simulations of reacting systems with detailed kinetic mechanisms, including thousands of chemical species and reactions. The framework is entirely written in object-oriented C++ and can be easily extended and customized by the user for specific systems, without having to modify the core functionality of the program. The OpenSMOKE++ framework can handle simulations of ideal chemical reactors (plug-flow, batch, and jet stirred reactors), shock-tubes, rapid compression machines, and can be easily incorporated into multi-dimensional CFD codes for the modeling of reacting flows. OpenSMOKE++ provides useful numerical tools such as the sensitivity and rate of production analyses, needed to recognize the main chemical paths and to interpret the numerical results from a kinetic point of view. Since simulations involving large kinetic mechanisms are very time consuming, OpenSMOKE++ adopts advanced numerical techniques able to reduce the computational cost, without sacrificing the accuracy and the robustness of the calculations.

Copyrights

License agreement

Using this software implies the following terms and conditions acceptance. Any publication written using the `OpenSMOKE++ Suite` must report the following references:

1. **Cuoci, A., Frassoldati, A., Faravelli, T., Ranzi E.,** *OpenSMOKE++: An object-oriented framework for the numerical modeling of reactive systems with detailed kinetic mechanisms*, Computer Physics Communications, 192, p. 237-264 (2015), DOI: 10.1016/j.cpc.2015.02.014
2. **Cuoci, A., Frassoldati, A., Faravelli, T., Ranzi E.,** *A computational tool for the detailed kinetic modeling of laminar flames: Application to C2H4/CH4 coflow flames*, Combustion and Flame, 160(5), p. 870-887 (2013), DOI: 10.1016/j.combustflame.2013.01.011

Limited warranty

Main developers

- **Alberto Cuoci**: kernel and core functions, ideal reactors, 1D solvers, ODE, DAE and NLS solvers, CHEMKIN interpreter, graphical post-processor
- **Mattia Bissoli**: interfaces with third-party ODE and DAE solvers, internal combustion enegines, sensitivity analysis
- **Alessandro Stagni**: analysis and reduction of kinetic mechansims, portability to Mac OSX and Linux, microgravity droplet

Acknowledgments

We thank the members of CRECK Modeling Group at Politecnico di Milano for their many suggestions, bug reports, valuable comments on the source code, functionality, and documentation.

Contents

Chapter 1

Introduction

The `OpenSMOKE++ Suite` is a collection of "standard solvers" for performing kinetic analyses with detailed kinetic mechanisms, with hundreds of species and thousands of reactions. The word "solver" has to be intended as an independent program, built with the aim to perform a specific task (for example to simulate a batch reactor, or to model a shock-wave, etc.). Thus, in the following, the `OpenSMOKE++ Suite` definition will be used to refer to the collection of `OpenSMOKE++` standard solvers. The list of available solvers (which is continuously growing) includes:

1. a kinetic pre-processor, a utility which is able to read, pre-process and analyze kinetic mechanisms written in the CHEMKIN format [?, ?, ?]. Its main purpose is to rewrite the kinetic scheme in a XML format which can be efficiently used by the `OpenSMOKE++ Suite` solvers;

2. a collection of solvers (i.e. independent executable files), one for each system to simulate. In other words, the `OpenSMOKE++ Suite` provides the solver dedicated to the simulation of plug flow reactors, the solver dedicated to the simulation of batch reactors, and so on. These solvers are completely independent from each other, but need the same pre-processed kinetic mechanism in XML format generated by the kinetic pre-processor described above;

3. a graphical post-processor, to easily post-process the simulation results. It is able not only to plot the usual profiles of temperature, pressure, composition, etc. along the time or space coordinate, but it is extremely useful to rapidly perform sensitivity analyses, rate of production analyses, and to draw flux diagrams. The figure below show a schematic diagram of the `OpenSMOKE++ Suite`.

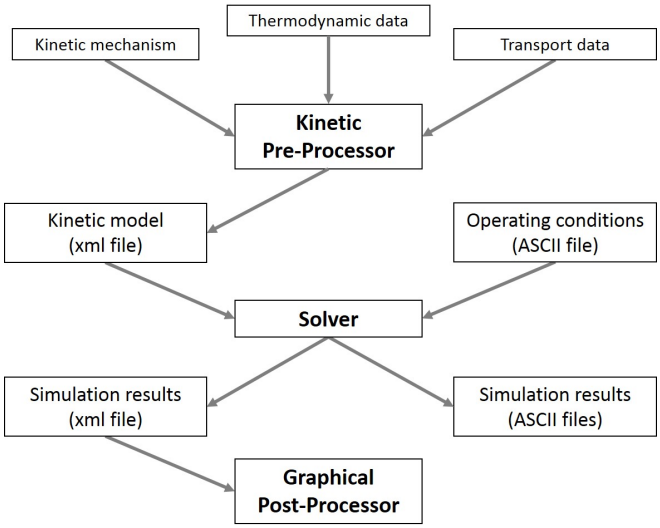

Figure 1.1: The `OpenSMOKE++ Suite` structure

1.1 Organization

The OpenSMOKE++ Suite consists of several executable files, libraries and utilities which are organized in several folders, as reported in the following:

- bin
- docs
- examples
- kinetic-mechanisms
- lib
- quick-start
- tutorials

bin

The bin folder contains the executable files to pre-process kinetic schemes in CHEMKIN format, to simulate ideal reactors (batch, perfectly stirred, plug-flow reactors, and shock tubes) and to graphically post-process the results.Together with the executables, the bin folder includes the needed dll libraries (for Microsoft Windows users) or the needed shared libraries (for Linux users) are available. We recommend to leave these libraries in the same bin folder containing the executable files.

docs

The docs folder contains documentation files.

examples

The examples folder, as suggested by the name, contains several examples to learn how to run the simulation for the ideal reactors available in the OpenSMOKE++ Suite. We strongly recommend to look at the examples reported in this folder to have more information about the most advanced options. All the examples can be simulated by simply double-clicking on the Run.bat file (for Microsoft Windows users) or running the Run.sh file (for Linux users).

kinetic-mechanisms

The kinetic-mechanisms folder contains several kinetic schemes in CHEMKIN format freely available from the web. Most of them are from the CRECK Modeling Group at Politecnico di Milano and can be downloaded at the following address: http://creckmodeling.chem.polimi.it/. For each kinetic scheme, at least the thermodynamic and the kinetic files are provided.

lib

The lib folder (available only for Linux platforms) contains the shared libraries needed to run the solvers and the graphical post-processor. In particular, the current version of OpenSMOKE++ Suite includes shared libraries for Boost C++, Intel MKL, and Sundials Suite.

quick-start

The quick-start folder contains the examples described in the Quick-Start Section (Chapter 3) of this User's Guide.

tutorials

The Tutorial folder contains the examples described in the Tutorials Section (Chapter 8)of this User's Guide.

1.2 How the OpenSMOKE++ Solvers work

All the `OpenSMOKE++ Suite` solvers, together with the kinetic pre-processor, do not have any graphical interface. They are based on input files in ASCII format. Only the graphical post-processor works with a GUI (Graphical User Interface), based on the QT libraries. This means that if you want to pre-process a kinetic mechanism (together with the thermodynamic and transport properties) or you want to simulate a reactor, your input conditions must be provided through one or more input files. These are simple ASCII files which can be written and modified using a generic text editor, like Notepad, Notepad++, Microsoft Word, gedit, etc. In the following lines, an example of perfectly stirred reactor is reported:

```
// Example of input file for OpenSMOKE++

Dictionary PerfectlyStirredReactor
{
        @KineticsFolder POLIMI_H2CO_NOX_1311;
        @Type           Adiabatic-ConstantPressure;
        @InletStatus    Inlet-Composition;
        @ResidenceTime  1 s;
        @Volume         100 cm3;
}

Dictionary Inlet-Composition
{
        @Temperature      1000. K ;
        @Pressure         101325. Pa ;
        @EquivalenceRatio 1.;
        @FuelMoles        H2 1.;
        @OxidizerMoles    O2 21 N2 79;
}
```

The main rules to write the input files are listed below:

1. The comments must be preceded by the `//` keyword. A comment can be inserted at the beginning or in the middle of a line. Every word or number reported after the `//` keyword is considered as a comment and ignored.

2. The input data must be organized in Dictionaries, i.e. homogeneous input data have to be collected together in the same structure (see the example above).

3. Each dictionary is defined by the `Dictionary` word, followed by the name of the dictionary. The options belonging to this dictionary have to be reported between the `{` and the `}` symbols.

4. Each keyword begins with the `@` symbol, usually requires one or more options (bool variables, number, strings, etc.)

5. Each input line must end with the `;` symbol.

In the example reported above, which refers to a steady-state, perfectly stirred reactor, two different dictionaries were defined, with names equal to `PerfectlyStirredReactor` and `Inlet-Composition`, respectively. In each of these two dictionaries, several input data are specified through different keywords. The available options depend on the solver you are using. Please, look at Chapters 6 and 7, where the complete list of options available for each kind of reactor are reported and explained.

Chapter 2

Installation

This section describes in details the procedure to install the `OpenSMOKE++ Suite` on your machine.

2.1 List of supported operating systems

Microsoft Windows family

- Microsoft Windows 10 (32/64 bit)
- Microsoft Windows 8 (32/64 bit)
- Microsoft Windows 7 (32/64 bit)
- Microsoft Windows XP (32/64 bit)

Linux family

- OpenSuse 13 (64 bit)
- Fedora 21 (64 bit)
- Ubuntu 12.04 (64 bit) and higher
- Xubuntu 12.04 (64 bit) and higher

Mac OS X

- Mac OS X 10.8 (64 bit) and higher

If your operating system does not appear in the list reported above, we suggest to install an Oracle VirtualBox (`https://www.virtualbox.org/`) running Ubuntu 14.04 or higher (`http://www.ubuntu.com/`).

2.2 Microsoft Windows

Pre-requisites for the graphical post-processor:

- Irfanview: `http://www.irfanview.com/`
- Graphiviz: `http://www.graphviz.org`

2.2.1 Installation of solvers

In order to install the `OpenSMOKE++ Suite`, double-click on the `msi` file. If you want to replace the existing version of `OpenSMOKE++ Suite` with a newer version, before proceeding with the new installation, we recommend to uninstall the previous version through the `Control Panel -> Add/Remove Programs` utility available in Microsoft Windows.

2.2.1.1 Manually setting the environment variables

Once you installed the software, the environment variables should be automatically updated. In case you experience problems in running the code, you have to manually update the environment variables according to the following procedure:

1. From the `Start` button, select `Computer`, right-click on it and select `Properties`
2. Click the `Advanced System Settings` link in the left column.
3. In the `System Properties` window, click on the `Environment Variables` button near the bottom of that tab.
4. In the `Environment Variables` window, create a new variable with name `OPENSMOKEPP_EXE_FOLDER` in the `User variables` section and click the `Edit` button. This variable has to point to the `bin` folder of your `OpenSMOKE++ Suite` installation. Of course the path depends on your system (i.e. on the location where you installed the `OpenSMOKE++ Suite`). As an example, if you installed the `OpenSMOKE++ Suite` in `C:\OpenSMOKE++ Suite` folder, the following string should be assigned to the `OPENSMOKEPP_EXE_FOLDER:`

```
C:\OpenSMOKE++ Suite\bin;
```

2.2.2 Installation of license file

The solvers belonging to `OpenSMOKE++ Suite` can be used only if a suitable license file (called `license.dat`) is available. The `license.dat` file must be present in the `bin` folder.

2.2.3 Notes about the OpenSMOKE++ Graphical Post-Processor

In order to use all the features of the graphical post-processor, the installation of Graphviz and Irfanview is strongly recommended. Without them, you cannot perform automatic generation of flux diagrams. However, all the other features of graphical post-processor work correctly even without Graphviz and Irfanview.

Instructions to install Graphviz

In order to create the flux analysis plots, Graphviz must be installed on your machine. You can download it from `http://www.graphviz.org/`. Once you installed the software, you need to update the environment variables using the following procedure (already described above):

1. From the `Start` button, select `Computer`, right-click on it and select `Properties`
2. Click the `Advanced System Settings` link in the left column.
3. In the `System Properties` window, click on the `Environment Variables` button near the bottom of that tab.
4. In the `Environment Variables` window, highlight the `PATH` variable in the `User variables` section and click the `Edit` button. Add or modify the path lines with the paths you want the computer to access. Each different directory is separated with a semicolon. In particular, modify the `PATH` variable by adding the path to the bin folder contained in the main Graphviz folder. Of course the path depends on your system (i.e. on the location where you installed the Graphviz library). As an example, if you installed the GraphViz library in "`C:\Program Files (x86)\Graphviz2.38`" folder, the following string should be appended to the PATH variable:

```
C:\Program Files (x86)\Graphviz2.38\bin;
```

Instructions to install Irfanview

In order to automatically visualize the flux analysis plots, the Irfanview viewer must be installed on your machine. You can download it from `http://www.irfanview.com/`. Once you installed the software, you need to update the environment variables using the following procedure (already described above):

1. From the `Start` button, select `Computer`, right-click on it and select `Properties`
2. Click the `Advanced System Settings` link in the left column.
3. In the `System Properties` window, click on the `Environment Variables` button near the bottom of that tab.
4. In the `Environment Variables` window, highlight the `PATH` variable in the `User variables` section and click the `Edit` button. Add or modify the path lines with the paths you want the computer to access. Each different directory is separated with a semicolon. In particular, modify the `PATH` variable by adding the path to the main Irfanview folder. Of course the path depends on your system (i.e. on the location where you installed IrfanView). As an example, if you installed Irfanview in "`C:\Program Files (x86)\IrfanView`" folder, the following string should be appended to the PATH variable:

```
C:\Program Files (x86)\IrfanView;
```

2.3 Linux

Pre-requisites for OpenSMOKE++ Suite Solvers:

- gfortran libraries: `https://gcc.gnu.org/wiki/GFortran`

Pre-requisites for the graphical post-processor:

- Qt4 libraries: `https://www.qt.io/download-open-source/`
- Graphiviz: `http://www.graphviz.org/`

2.3.1 Installation of solvers

1. Be sure that `gfortran` libraries are available on your machine. If not, you can easily find precompiled packages for the most common Linux distributions (see paragraph 2.3.5).
2. Unpack the `opensmoke++suite-0.4.tar.gz` file in the location you prefer using the following command:

```
tar -xzf opensmoke++suite-0.4.tar.gz
```

3. Update the environment variable settings:

   a) if running `bash` or `ksh` (if in doubt type: `echo $SHELL`):
   edit your `$HOME/.bashrc` file by adding the following lines:

```
export PATH=$PATH:/path_to_opensmoke_suite/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path_to_opensmoke_suite/lib
```

   b) if running `tcsh` or `csh`:
   edit your `$HOME/.cshrc` file by adding the following lines:

```
setenv PATH $PATH\:/path_to_opensmoke_suite/bin
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH\:/path_to_opensmoke_suite/lib
```

   As an example, if you unpacked the `opensmoke++suite-0.4.tar.gz` file directly in your home folder (`$HOME`), the following lines have to be added (in case of bash):

```
export PATH=$PATH:$HOME/opensmoke++suite-0.4/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/opensmoke++suite-0.4/lib
```

4. Close the terminal and open a new terminal (or source the updated `$HOME/.bashrc` or `$HOME/.cshrc`).

2.3.2 Installation of license file

The `OpenSMOKE++ Suite` solvers can be used only if a suitable license file (called `license.dat`) is available. The `license.dat` file must be present in the `bin` folder.

2.3.3 Notes about the OpenSMOKE++ Graphical Post-Processor

In order to use all the features of the graphical post-processor, the QT4 libraries are needed. Moreover, the installation of Graphviz (even if not strictly needed) is strongly recommended. Without it, you cannot perform automatic generation of flux diagrams. However, all the other features of graphical post-processor work correctly, even without Graphviz.

Instructions to install the Qt4 libraries

Precompiled Qt4 libraries are already available in most Linux distributions (e.g. Ubuntu, Xubuntu, Debian, Mandriva), which means that you do not have to install them. Thus, the instructions below refer to Linux distributions which are distributed without Qt4 libraries.

1. Installation of pre-compiled libraries (recommended)

   Even if not available in the basic distribution, precompiled Qt4 libraries available for most Linux distributions. Please, follow the instructions specific to the distribution you are using.

2. Compilation from source code

   If no pre-compiled libraries are available for your Linux distribution, you need to compile the source code. You can download it from `https://www.qt.io/download-open-source/`. Please, follow the instructions provided in the source code to compile and install the library. Be sure that the `LD_LIBRARY_PATH` environment variable includes the `lib` folder of installed Qt4 library.

Instructions to install the Graphviz library

In order to create the flux analysis plots, Graphviz must be installed on your machine. Two options are possible: installation of pre-compiled binaries (if available for your Linux distribution) or compilation from source code.

1. Installation of pre-compiled binaries (recommended)

   If available for your Linux distribution, you can simply install the Graphviz library directly from the pre-compiled binaries.

   Fedora

   The easiest way to install and maintain Graphviz on Fedora is to use `yum`. To set up `yum`, download the `graphviz-fedora.repo` `http://www.graphviz.org/graphviz-fedora.repo` . Save it (as `root`) in `/etc/yum.repos.d/` Then you can (as `root`) type:

```
yum list available 'graphviz*'
yum install 'graphviz*'
```

   OpenSuse

   Use the `Yast` utility to install GraphViz.

   Ubuntu and Xubuntu

   Type the following command in a terminal:

```
sudo apt-get install graphviz
```

2. Compilation from source code

   If no pre-compiled binaries are avialble for your Linux distribution, you need to compile the source code. You can download it from `http://www.graphviz.org/`. Please, follow the instructions provided in the source code to compile and install the library. Be sure that the `PATH` environment variable includes the `bin` folder of installed GraphViz.

2.3.4 Check your installation

1. Check if installation of solvers was done properly

   go to the `quick-start\01-adiabatic-batch-reactor` folder and run the simulation by typing

   ```
   ./Run.sh
   ```

2. Check if installation of graphical post-processor was done properly: from the same folder, type

   ```
   OpenSMOKE_PostProcessor.sh
   ```

If error messages about missing shared libraries appear, have a look at the Notes reported below.

2.3.5 Notes about shared libraries

You may experience some issues in running OpenSMOKE++ Suite solvers and/or graphical post processor, especially if you use are not using Ubuntu or Xubuntu. In most cases, the typical errors are about missing shared libraries:

`error while loading shared libraries: name_of_library.so`

In that cases you need to install the missing library/libraries. In particular, be sure to have the `Qt4` and `gfortran` libraries properly installed on your machine.

Here below we reported a list of known issues, together with possible solutions:

- **Fedora**

  Error message: `error while loading shared libraries: libgfortran.so.x`

  Solution: `su -c 'yum install gcc-gfortran'`

- **OpenSuse**

  Error message: `error while loading shared libraries: libQtGui.so.x`

  Solution: install `libqt4` using Yast

- **Ubuntu and Xubuntu**

  Error message: `error while loading shared libraries: libgfortran.so.x`

  Solution: `sudo apt-get install libgfortran`

- **Xubuntu**

  Error message: `error while loading shared libraries: libaudio.so.x`

  Solution: `sudo apt-get install libaudio2`

2.4 Mac OS X

Pre-requisites for OpenSMOKE++ Suite Solvers:

- gfortran libraries: `https://gcc.gnu.org/wiki/GFortranBinaries#MacOS`

Pre-requisites for the graphical post-processor:

- Graphiviz: `http://www.graphviz.org/Download_macos.php`

2.4.1 Installation of solvers

1. Be sure that `gfortran` libraries are available on your machine. If not, you can download and installed precompiled libraries for your Mac OS X at the following web-address:

   `https://gcc.gnu.org/wiki/GFortranBinaries#MacOS`

2. Unpack the `opensmoke++suite-0.4-macos.tar.gz` file in the location you prefer using the following command:

```
tar -xzf opensmoke++suite-0.4-macos.tar.gz
```

3. Update the environment variable settings:

   edit your `$HOME/.bashrc_profile` file (or your `$HOME/.profile` file, depending on your settings) by adding the following line:

```
export PATH=$PATH:/path_to_opensmoke_suite/bin
```

   As an example, if you unpacked the `opensmoke++suite-0.4-macos.tar.gz` file directly in your home folder (`$HOME`), the following line has to be added:

   `export PATH=$PATH:$HOME/opensmoke++suite-0.4-macos/bin`

4. Close the terminal and open a new terminal (or source the updated `$HOME/.bashrc_profile` or your `$HOME/.profile`, depending on your settings)

2.4.2 Installation of license file

The `OpenSMOKE++ Suite` solvers can be used only if a suitable license file (called `license.dat`) is available. The `license.dat` file must be present in the `bin` folder.

2.4.3 Notes about the OpenSMOKE++ Graphical Post-Processor

In order to use all the features of the graphical post-processor, the installation of Graphviz (even if not strictly needed) is strongly recommended. Without it, you cannot perform automatic generation of flux diagrams. However, all the remaining features of graphical post-processor work correctly even without Graphviz.

Instructions to install Graphviz

Two options are possible: installation of pre-compiled binaries (recommended) or compilation from source code.

1. Installation of pre-compiled binaries (recommended)

   Precompiled Graphviz is available at the following web address: `http://www.graphviz.org/Download_macos.php`. You can install it using the usual installation procedure for Mac OSX.

2. Compilation from source code

   The source code can be downloaded from `http://www.graphviz.org/`. Please, follow the instructions provided in the source code to compile and install the library. Be sure that the `PATH` environment variable includes the `bin` folder of installed GraphViz.

2.4.4 Check your installation

1. Check if installation of solvers was done properly

   go to the `quick-start\01-adiabatic-batch-reactor` folder and run the simulation by typing

```
./Run.sh
```

2. Check if installation of graphical post-processor was done properly

   from the same folder, type:

```
open /path_to_opensmoke_suite/bin/OpenSMOKE_PostProcessor.sh.app
```

As an example, if you installed OpenSMOKE++ in your home directory (`$HOME`), you can type:
`$HOME/opensmoke++suite-0.4-macos/bin/OpenSMOKE_PostProcessor.sh.app`

Chapter 3

Quick start

In this section we describe how to setup and run a typical OpenSMOKE++ simulation. As a first example, the attion will be focused on an adiabatic batch reactor, whose initial composition is a stoichiometric mixture of syngas (40% H2, 60% CO) and air (21% O2, 79% N2) at 10 atm and 600 K.

3.1 Selection of thermodynamic data and kinetic mechanism

In order to run simulations with OpenSMOKE++ Suite solvers, the user needs a kinetic scheme describing the homogeneous (gas-phase) (and, optionally, also the heterogeneous or surface reactions), and the thermodynamic properties for each species included in the kinetic mechanism. The current version of OpenSMOKE++ requires that the kinetic scheme, together with the thermodynamic and (optionally) transport properties, is provided according to the CHEMKIN rules [?]. Kinetic mechanisms and thermodynamic and transport data in CHEMKIN-compatible format are available from several research groups. In particular, a non exaustive list (adapted from[?]) is reported in the following:

- The CRECK Modeling group at Politecnico di Milano provides detailed kinetic mechanisms describing pyrolysis, combustion, and oxidation of several fuels: hydrogen, methane, propane, PRF, diesels, jet-fuels, biofuels, etc.

  http://creckmodeling.chem.polimi.it/

- Lawrence Livermore National Laboratory (LLNL) has posted combustion mechanisms, including thermodynamic data, transport data, and reaction sets for flame simulations with hydrogen and various hydrocarbon fuels.

  https://www-pls.llnl.gov/?url=science_and_technology-chemistry-combustion-mechanisms

- The Combustion Division of the Center for Energy Research at the University of California-San Diego has posted kinetic mechanisms for combustion qpplications, including nitrogen, JP10, and heptane chemistry.

  http://web.eng.ucsd.edu/mae/groups/combustion/mechanism.html

- The Combustion Group at Princeton University provides mechanisms for combustion of chlorinated and fluorinated compound mixtures with methane, and for high-pressure methane and propane flames.

  http://www.princeton.edu/mae/people/faculty/dryer/homepage/kinetic_models/

- Professor Alexander Burcat of Technion Israeli Institute of Technology has posted his database of thermodynamic properties. The data has been collected from many sources and is critically reviewed and frequently updated. The BURCAT.THR file is in CHEMKIN format, but contains comments and descriptive text about each species, so that file would require some user manipulation (commenting out text lines) in order to be used directly in OpenSMOKE++.

  http://garfield.chem.elte.hu/Burcat/burcat.html

- The Gas Research Institute (GRI) funded development of the GRI-mech, which is a CHEMKIN mechanism for natural gas combustion, including thermodynamic properties and rates. You can download the most recent version of GRI-mech from the GRI-mech website.

  http://combustion.berkeley.edu/gri-mech/

Please, look at the CHEMKIN user's guide [?]to find the syntax rules to correctly write and modify the files describing the kinetic mechanism and the thermodynamic and transport properties.

In this example we will use the `POLIMI_H2CO_NOX_1412` kinetic mechanism of CRECK Modeling group at Politecnico di Milano. This is a detailed kinetic mechanism describing the combustion of mixtures of hydrogen and carbon monoxide, consisting of 32 species involved in more than 170 reactions. It is available in the `kinetic-mechanisms\POLIMI_1412\Kinetics` folder or, alternatively, it can be freely downloaded from the following page: (`http://creckmodeling.chem.polimi.it/`). The thermodynamic data are available in the `kinetic-mechanisms\POLIMI_1412\Thermodynamics` folder. In particular, for the purposes of this first example, we need only the thermodynamic data (`POLIMI_TOT_NOX_1412.CKT` file) and the kinetic mechanism (`POLIMI_H2CO_NOX_1412.CKI` file). The transport data are not needed for the simulation of a batch reactor.

3.2 Pre-processing of kinetic mechanism

Before using it in `OpenSMOKE++ Suite`, the kinetic scheme has be be pre-processed and re-written in a new format. This operation can also be performed on-the-fly, i.e. directly when the simulation is started (see Section 7.6). However, in this first example, we prefer to perform this pre-processing operation as an independent step. The pre-processing of a kinetic mechanism (together with thermodynamic and transport data, if needed) must be performed only once, using a particular solver of `OpenSMOKE++ Suite`, which is called `OpenSMOKE_CHEMKIN_PreProcessor`.

1. Create a new folder and copy there the the thermodynamic and kinetic files mentioned above (`POLIMI_TOT_NOX_1412.CKT` and `POLIMI_H2CO_NOX_1412.CKI`, respectively).

2. Create a new input file (`kinetics.dic` in the following, but in principle the user is free to choose the name he prefers) in which you specify the the thermodynamic and kinetic files and the destination folder:

```
Dictionary CHEMKIN_PreProcessor
{
        @Thermodynamics POLIMI_TOT_NOX_1412.CKT;
        @Kinetics       POLIMI_H2CO_NOX_1412.CKI;
        @Output         POLIMI_H2CO_NOX_1412;
}
```

3. Run the kinetic pre-processor using the following command (in Microsoft Windows):

```
%OPENSMOKEPP_EXE_FOLDER%\OpenSMOKE_CHEMKIN_PreProcessor.exe --input kinetics.dic
```

   or (in Linux and Mac OSX):

```
OpenSMOKE_CHEMKIN_PreProcessor.sh --input kinetics.dic
```

4. If everything works properly, you will find the result of this pre-processing operation in the folder you specified through the `@Output` option in the `kinetics.dic` file.

3.3 Preparation of batch reactor simulation

We are now ready to write the input file (`batch.dic` in the following) reporting the input data for the batch reactor simulation.

1. In the same folder where the kinetic mechanism has been pre-processed, create a new text file, called `batch.dic`:

```
Dictionary BatchReactor
{
        @KineticsFolder         POLIMI_H2CO_NOX_1412;
        @Type                   NonIsothermal-ConstantVolume;
        @InitialStatus          initial-mixture;
        @EndTime                0.01 s;
        @Options                output-options;
```

```
}

Dictionary initial-mixture
{
        @Temperature        1000.   K ;
        @Pressure        101325. Pa ;
        @EquivalenceRatio   1.;
        @FuelMoles          H2 40. CO 60.;
        @OxidizerMoles      O2 21 N2 79;
}

Dictionary output-options
{
        @StepsFile              2;
}
```

For the meaning of each option and additional/alternative options, please look at Chapters 6 and 7.

2. Run the `OpenSMOKE++ Suite` solver for batch reactors (`OpenSMOKE_BatchReactor`) using the following command (in Microsoft Windows):

```
%OPENSMOKEPP_EXE_FOLDER%\OpenSMOKE_BatchReactor.exe --input batch.dic
```

or (in Linux and Mac OSX):

```
OpenSMOKE_BatchReactor.sh --input batch.dic
```

3. If everything works properly, you will find the result of this pre-processing operation in the `Output` folder.

3.4 Analysis of results (without graphical post-processor)

The results of the simulations are written in the `Output` folder. In this particular example three files are available:

- `FinalSummary.out`: this file reports the initial and final status of the reacting mixture (temperature, pressure, density, composition, etc.)
- `Output.out`: it is organized in columns and the first row reports the meaning of each column, together with a number which refers to the column number. Both the mole and mass fractions of species are reported: the mole fractions have the `x` suffix, while the mass fractions the `w` suffix. You can easily import this file in Microsoft Excel, Matlab, etc. or you can use Gnuplot to directly plot (also on the fly) the results.
- `Output.xml`: XML output file to be used by the `OpenSMOKE++` graphical post-processor (see next Section)

The profiles reported in the `Output.out` file can be easily plotted using Gnuplot, Microsoft Excel, Matlab, etc. In particular, we recommend to use Gnuplot. As an example, if you want to plot the mole fraction profiles of H2 versus time, in Gnuplot you can type:

```
p 'Output/Output.out' u 1:22 w l t 'H2'
```

Figure 3.1 shows additional profiles plotted with Gnuplot.

3.5 Analysis of results with OpenSMOKE++ graphical post-processor

In addition, you can analyze the results through the `OpenSMOKE++` Graphical PostProcessor.

1. In order to graphically post-process the results you have to run the `OpenSMOKE_PostProcessor` file (contained in the `bin` folder of `OpenSMOKE++ Suite`).

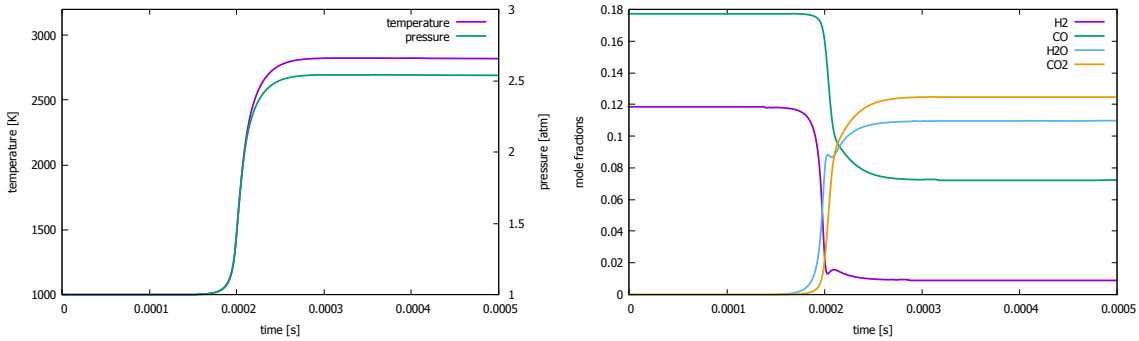

Figure 3.1: Adiabatic batch reactor: profiles of temperature and pressure (left) and main species (right). The plots have been obtained from the `Output.out` file, using gnuplot.

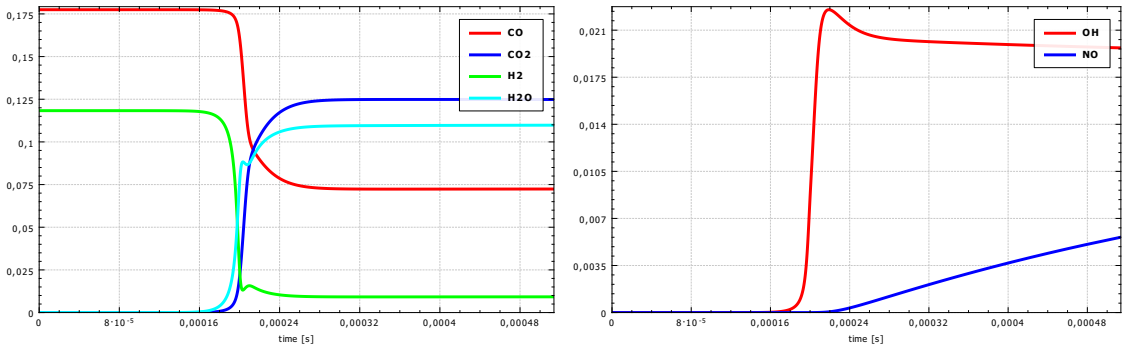

Figure 3.2: Adiabatic batch reactor: profiles of main species (left) and OH radical and NO (right). The profiles have been obtained directly using the `OpenSMOKE++ PostProcessor`.

2. Select the folder containing the `Output.xml` file generated by the simulation by clicking on the `Select Results...` button. The `Profiles` button becomes now available. If you click on it, you have the possibility to plot the calculated profiles of species, temperature, pressure, etc. As an example, in Figure 3.2 profiles of selected species are reported.

3. If you want to perform more interesting post-processing analyses, you have to load also the pre-processed kinetic scheme. In order to do this, you should close the `Profiles` window and click on the `Select Mechanism...` button. Then, select the folder containing the `kinetics.xml` file generated by the kinetic pre-processor. The `Rate of Production` button becomes available. If you click on this button, a new window will be available. Here you have the possibility to plot formation rates for all the species and reaction rates for all the reactions (see Figure 3.3). You can analyze the distribution of characteristic times by looking at the eigenvalues of the Jacobian matrix, by clicking on the `Analyze Characteristic Times` button.

4. More interestingly, you can perform rate of production analysis for each species in the kinetic scheme through the `Plot ROPA bars` button (see Figure 3.4).

5. Eventually, you can also draw path diagrams through the `Flux Analysis` button (this feature is only available if you correctly installed Graphviz and Irfanview, as described in the previous Chapter). As an example, in the `Type` panel select the `Local` radio button and write `0.022` in the corresponding window (this is the time in seconds at which you want to perform the flux analysis). Select `NO` from the `Species` list, `N` element from the `Element` list, and `Production` from the `Type` panel. If you click on the `Flux Analysis` button, you should get the flux diagram reported in Figure 3.5.

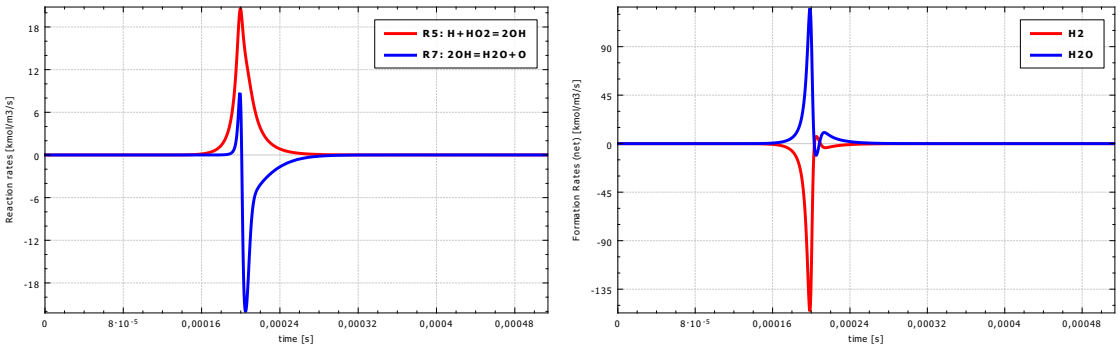

Figure 3.3: Adiabatic batch reactor: profiles of reaction rates (left) and formation rates of selected species (right). The profiles have been obtained directly using the `OpenSMOKE++ PostProcessor`.

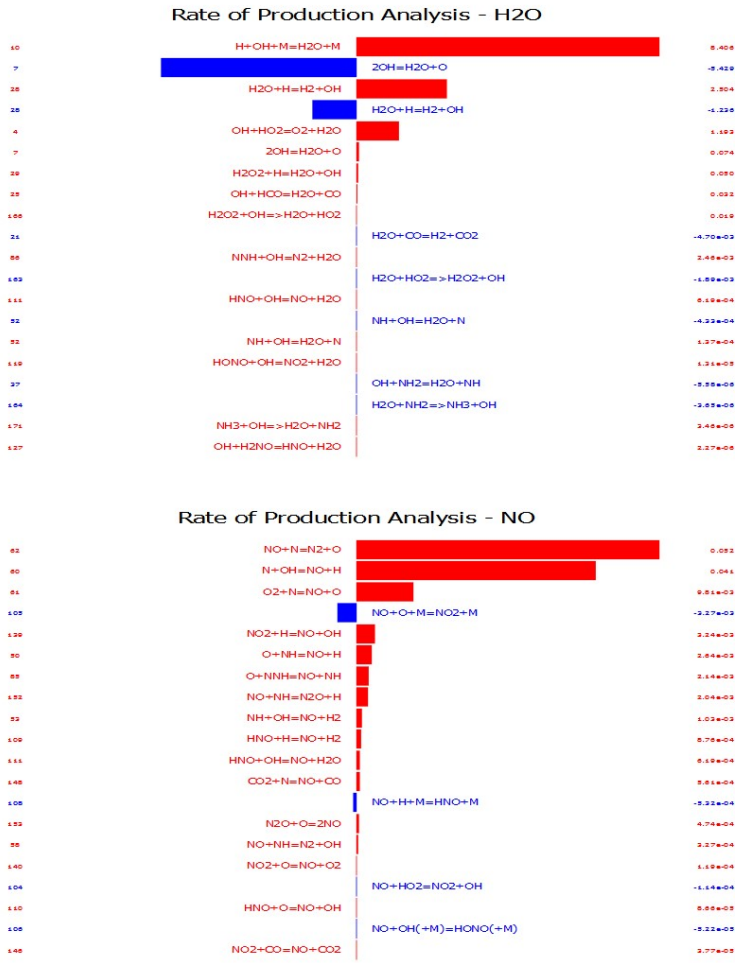

Figure 3.4: Adiabatic batch reactor: rate of production analyses for H2O and NO as calculated by suing the `OpenSMOKE++ PostProcessor`.

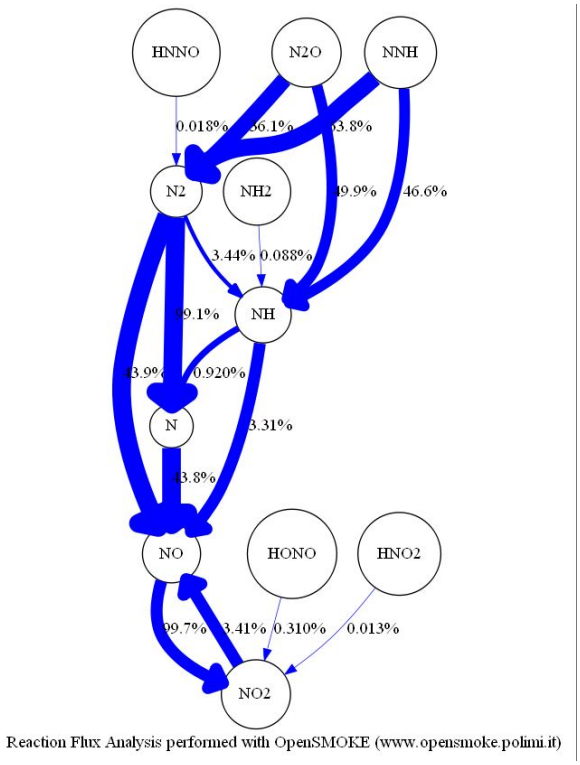

Figure 3.5: Adiabatic batch reactor: flux analysis (NO production, Element N, Depth 1, Width 4)

3.6 Sensitivity analysis

The `OpenSMOKE++ Suite` provides several useful tools to perform sensitivity analysis, both for unsteady and steady-state problems. Sensitivity analysis is very important for kinetic studies, since it allows the quantitative understanding of how the numerical solution of the governing equations depends on the various parameters contained in the model itself. Only the first-order sensitivity coefficients with respect to the reaction rate coefficients (pre-exponential factors, activation energy or kinetic constant) can be calculated.

Since the calculation of first-order sensitivity coefficient is very time consuming, the user has to request explicitly the sensitivity analysis before running the simulation. Thus, in order to perform the sensitivity analysis, we modify the `batch.dic` file by adding a new dictionary:

```
Dictionary sensitivity-options
{
        @Type                   arrhenius-parameters;
        @DenseSolver            Eigen;
        @DenseFullPivoting      false;
        @SubSteps               5;
        @Species                NO NO2 CO2;
}
```

and by enabling the sensitivity analysis calculation in the main `BatchReactor` dictionary, through the addition of the following command:

```
@SensitivityAnalysis    sensitivity-options;
```

Now you can run the batch reactor simulation using the following command (in Microsoft Windows):

```
%OPENSMOKEPP_EXE_FOLDER%\OpenSMOKE_BatchReactor.exe --input batch.dic
```

or (in Linux amd Mac OSX):

```
OpenSMOKE_BatchReactor.sh --input batch.dic
```

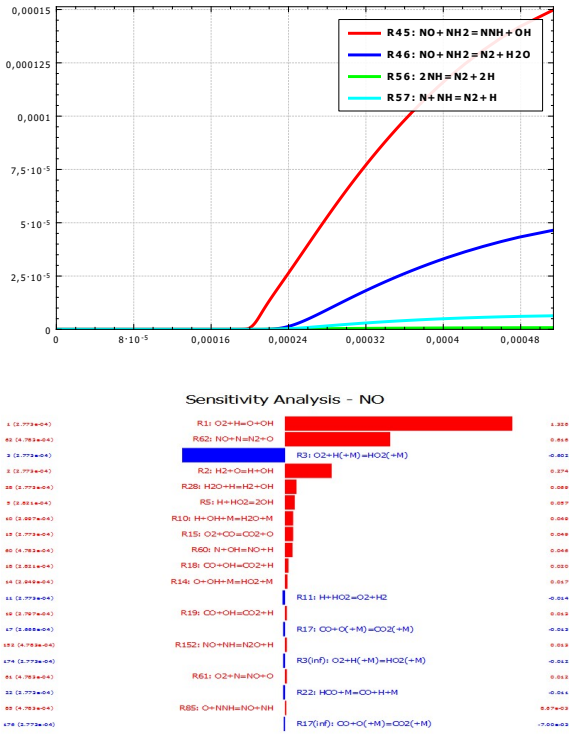

Figure 3.6: Adiabatic batch reactor: sensitivity analysis of NO.

If everything worked properly, you can now look at the results of sensitivity analysis using the `OpenSMOKE++ PostProcessor`. In particular, after selecting the folders containing the `Output.xml` file and the pre-processed kinetic mechanism, the third button, `Sensitivity Analysis`, will be available. You can now plot the specific sensitivity coefficient profiles for each species and reaction or a bar chart for every species. An example is reported in Figure 3.6.

Chapter 4

Kinetic pre-processor

In order to preprocess a kinetic scheme, the `OpenSMOKE++ Suite` provides the `OpenSMOKE_CHEMKIN_PreProcessor` utility. The user has to supply the files containing the thermodynamic data, the kinetic mechanism, and (optionally) the transport data. For the simulation of ideal reactors, usually the transport data are not needed and therefore the user could choose to preprocess only the thermodynamic and the kinetic data. This is useful, since in many cases the transport data are not available. In order to run the `OpenSMOKE_CHEMKIN_PreProcessor` utility, the user has to write an input file containing the instructions (i.e. the dictionary) to perform the preprocessing and/or additional useful operations (checking of the thermodynamic properties, post-processing of reaction rates, etc.). The rules to write the dictionary are reported in Section 6.1. Please look at the tutorials to have more details. The `OpenSMOKE_CHEMKIN_PreProcessor` utility can be run from the command line using the following instructions, where it is assumed that the file containing the dictionary is called `myinput.dic` and the dictionary `mydictionary`:

1. in Microsoft Windows

```
%OPENSMOKEPP_EXE_FOLDER%\OpenSMOKE_CHEMKIN_PreProcessor.exe --input myinput.dic
  --dictionary mydictionary
```

   Please, consider that the instruction reported above must be written in a single line. You can use multiple lines using the ^ symbol.

2. in Linux

```
OpenSMOKE_CHEMKIN_PreProcessor.sh --input myinput.dic --dictionary mydictionary
```

Notes

1. The `--input` option can be omitted. In this case the `OpenSMOKE_CHEMKIN_PreProcessor` assumes that the input file is called `input.dic`

2. The `--dictionary` option can be omitted. In this case the `OpenSMOKE_CHEMKIN_PreProcessor` assumes that the dictionary is called `CHEMKIN_PreProcessor`

3. In addition to several ASCII files (depending on the options specified in the dictionary), a `kinetics.xml` file will be generated by the `OpenSMOKE_CHEMKIN_PreProcessor`. This is the only file needed by the `OpenSMOKE++ Suite` solvers.

Chapter 5

Ideal reactors

The ideal reactor simulations can be performed using the same approach used for pre-processing the kinetic mechanism, as described in Chapter 4. Thus, the user has to supply a proper dictionary containing the instructions and the options for running the simulation under investigation. As usual, the solver can be run using the following instruction from the command line (where it is assumed that the dictionary is called `mydictionary`, contained in `myinput.dic` file):

1. in Microsoft Windows

```
%OPENSMOKEPP_EXE_FOLDER%\OpenSMOKE_SolverName.exe --input myinput.dic
  --dictionary mydictionary
```

   Please, consider that the instruction reported above must be written in a single line. You can use multiple lines using the ^ symbol.

2. in Linux

```
OpenSMOKE_SolverName.sh --input myinput.dic --dictionary mydictionary
```

Notes

1. The –input option can be omitted. In this case the `OpenSMOKE_SolverName` assumes that the input file is called `input.dic` The rules to write the dictionary for several ideal reactors are reported in Chapter 6. Please look at the tutorials to have more details.

2. The kinetic scheme can be provided in 2 different ways:

   a) If a pre-processed version of the kinetic scheme is already available (i.e. the `kinetics.xml` file was already generated as described in Chapter4), the user can use directly this file, by providing the folder where it is contained (through the `@KineticsFolder` option)

   b) The user can choose to pre-process the kinetic mechanism on the fly, by providing the paths to the kinetic files (usually through the `@KineticsPreProcessor` option)

3. At the end of the simulation, an `Output.xml` file will be written in the `Output` folder, together with additional files in ASCII format.

Chapter 6

Main dictionaries

As better explained in the previous Chapters, the input data have to be provided through proper dictionaries, depending on the type of solver the user wants to run. In the current version of `OpenSMOKE++ Suite` six different solvers are available:

1. CHEMKIN-PreProcessor
2. Batch-Reactor
3. Perfectly-Stirred-Reactor
4. Shock-Tube-Reactor
5. Plug-Flow-Reactor
6. PremixedLaminarFlame1D
7. Thermodynamic-Equilibrium
8. Laminar-Flamelet
9. LookUp-Tables
10. Microgravity-Droplet (under development)

Each of the dictionaries reported below usually need additional options, which are based on sub-dictionaries, described in Chapter 7.

6.1 Dictionary: CHEMKIN-PreProcessor

This dictionary is used by the `OpenSMOKE_CHEMKIN_Preprocessor` solver with the aim to preprocess (i.e. interpret) a kinetic mechanism written in the standard CHEMKIN format. Only the thermodynamic file is strictly necessary, but, obviously, a kinetic mechanism must be provided if the aim is to perform simulations of reacting systems. For most of the ideal reactors no transport data are required and therefore the user could choose to pre-process only the thermodynamic and the kinetic files. According to the specified options, additional analysis can be easily and rapidly performed. Please consider that the kinetic and thermodynamic data must be always provided in two different files (i.e. the thermodynamic data cannot be added to the same file where the kinetic scheme is written).

6.1.1 Additional comments

@Thermodynamics

This option requires the name of the file (ASCII) containing the thermodynamic data (in CHEMKIN format). Both local and global paths can be accepted. Several useful files are automatically generated in the `Output` folder. In particular, the `Thermodynamics_Coefficients.out` file reports the correlation coefficients for the thermodynamic properties in a readable format. The `Thermodynamics_Tables.out` file reports the thermodynamic properties (specific heat, enthalpy, entropy, etc.) for each species as function of temperature.

Option	Type	Meaning
@Thermodynamics	PATH	Name of the file (ASCII) containing the thermodynamic data (CHEMKIN® format)
@Output	PATH	Name of the folder where the pre-processed data are written
@Transport	PATH	Name of the file (ASCII) containing the transport data (CHEMKIN® format)
@Kinetics	PATH	Name of the file (ASCII) containing the kinetic mechanism (CHEMKIN® format)
@CheckThermodynamics	BOOL	The thermodynamic data are checked and additional files are written in output (together with a consistent reformulation of thermodynamic data)
@OutputOldStyle	BOOL	The output file are written also using the old format used by the OpenSMOKE framework (before 2013)
@TransportFittingCoefficients	BOOL	The fitting coefficients for the transport properties are written on a file. Please consider that this operation is very slow for large kinetic mechanisms (more than 1000 species) and produces huge files
@ReactionTables	BOOL	For each reaction detailed information is reported on a file (kinetic constants, change of moles, etc.)
@ReverseFitting	BOOL	For each reversible reaction the reverse kinetic constants are estimated assuming the Arrhenius' law
@Comments	SUBDICTIONARY [Comments]	Additional data (author name, comments, etc.) can be added to the pre-processed kinetic mechanism
@SparsityPatternAnalysis	BOOL	Additional analyses about the sparsity pattern of stoichiometric matrix and associated Jacobian matrix
@RewriteCHEMKIN	BOOL	The kinetic mechanism is rewritten in CHEMKIN format by accounting for the corrections on atomic balances carried out during the pre-processing phase (default: true)
@SpeciesBundling	BOOL	Enables the species bundling technique for the fast estimation of mass diffusion coefficients (default: false)

Table 6.1: CHEMKIN-PreProcessor dictionary

@Kinetics

This option requires the name of the file (ASCII) containing the kinetic mechanism (in CHEMKIN format). Both local and global paths can be accepted. The `Kinetics_Summary.out` file reports the whole kinetic mechanism written in a more readable format.

@Transport

This option requires the name of the file (ASCII) containing the transport data (in CHEMKIN format). Both local and global paths can be accepted.

@Output

This can specify the name of the folder where the results of the simulation will be written. Both local and global paths can be accepted.

@CheckThermodynamics

This option allows to perform a detailed check of the thermodynamic data, to find possible inconsistent or unphysical data. The `Thermodynamics_Status.out` file reports a summary of the analysis of the thermodynamic data. In addition, a new file, containing new thermodynamic data, called `thermo.CHEMKIN.CKT`, will be provided. This new thermodynamic file is generated on the basis of the original thermodynamic data, which are made perfectly consistent at the transition temperature between low and high temperature correlations.

@TransportFittingCoefficients

The fitting coefficients for the transport properties (viscosity, thermal conductivity, mixture-averaged mass diffusion, and mixture-averaged thermal diffusion) for each species in the kinetic file are written on a file. Please consider that this operation is very slow for large kinetic mechanisms (more than 1000 species) and produces huge files (dimensions of Gb).

@ReactionTables

For each reaction detailed information is reported on the `Reaction_Tables.out` file (kinetic constants, change of moles, etc.). The `Reaction_Tables.out` file is self-explanatory.

@ReverseFitting

For each reversible reaction the reverse kinetic constants are estimated assuming the Arrhenius' law. The kinetic parameters of the reverse reactions are then written on the `Reaction_FittedKinetics.out` file.

@SparsityPatternAnalysis

Additional analyses about the sparsity pattern of stoichiometric matrix and associated Jacobian matrix

@RewriteCHEMKIN

The kinetic mechanism is rewritten in CHEMKIN format by accounting for the corrections on atomic balances carried out during the pre-processing phase (default: true). At the end of this operation a new kinetic file (`kinetics.CHEMKIN.CKI`) is available in the Output folder.

@SpeciesBundling

Enables the species bundling technique for the fast estimation of mass diffusion coefficients (default: false). The technique is described in the following paper: **Lu and Law**, *Diffusion coefficient reduction through species bundling*, Combustion and Flame, 148(3), p. 117-126 (2007)

6.2 Dictionary: Batch Reactor

This dictionary is used for setting the simulation for a Batch Reactor. In the current version of `OpenSMOKE++ Suite` the user can simulate both isothermal and non-isothermal reactors. Both the constant volume and constant pressure constraints are available.

6.2.1 Additional comments

@KineticsFolder

This option is used to specify the name of the folder containing the kinetic scheme in XML Version (`kinetics.xml`). This file has to be generated using the `OpenSMOKE_CHEMKIN_Preprocessor`. Instead of using this option, the user has the possibility to pre-process the kinetic scheme on the fly using the `@KineticsPreProcessor` option (see below).

@KineticsPreProcessor

The user can pre-process a kinetic scheme available in CHEMKIN format on the fly, when the reactor simulation is performed (instead of pre-processing the kinetics in a previous step). This option is very useful when the kinetic mechanism is under construction and/or tuning and the user has the need to change often the reactions and the kinetic parameters. Please consider that the pre-processing operation can be quite long for very detailed kinetic schemes (thousands of species). The `@KineticsPreProcessor` option requires that the user specify the name of the sub-dictionary containing all the information needed to pre-process kinetic mechanisms on the fly. Please look at Section 7.6 to have more details.

@Type

This option is used to specify the type of reactor to simulate. Four different combinations are possible: `Isothermal-ConstantVolume`, `NonIsothermal-ConstantVolume`, `Isothermal-ConstantPressure`, `NonIsothermal-ConstantP`

@InitialStatus

This option is used to specify the features (temperature, pressure and composition) of the initial mixture (i.e. at time 0) in the batch reactor. The `@InitialStatus` option requires that the user specify the name of the corresponding sub-dictionary. Please look at Section 7.5 to have more details.

@EndTime

This is the residence time of the batch reactor.

@StartTime

Start time for transient simulation (default 0).

@Volume

This is the volume of the batch reactor. Please look at Section 7.7 to have more details.

@GlobalThermalExchangeCoefficient

Global thermal exchange coefficient U: $Q = UA(T - T_{env})$

@EnvironmentTemperature

EnvironmentTemperature T_{env}: $Q = UA(T - T_{env})$

@ExchangeArea

Exchange area A: $Q = UA(T - T_{env})$

Option	Type	Meaning
@KineticsFolder	PATH	Name of the folder containing the kinetic scheme (XML Version)
@KineticsPreProcessor	SUBDICTIONARY [Rapid-Kinetics]	Dictionary containing the list of kinetic files to be interpreted (see Section 7.6)
@Type	STRING	Batch reactor type: `Isothermal-ConstantVolume` | `NonIsothermal-ConstantVolume` | `Isothermal-ConstantPressure` | `NonIsothermal-ConstantPressure`
@InitialStatus	SUBDICTIONARY [Gas-Status]	Dictionary defining the initial gas composition, temperature and pressure (see Section 7.5)
@EndTime	MEASURE	Integration time for transient simulation
@StartTime	MEASURE	Start time for transient simulation (default 0)
@Volume	MEASURE	Volume of the reactor
@GlobalThermalExchangeCoefficient	MEASURE	Global thermal exchange coefficient U: $Q = UA(T - T_{env})$
@EnvironmentTemperature	MEASURE	EnvironmentTemperature T_{env}: $Q = UA(T - T_{env})$
@ExchangeArea	MEASURE	Exchange area A: $Q = UA(T - T_{env})$
@VolumeProfile	SUBDICT [XY-Profile]	Dictionary defining the volume profile (see Section 7.7)
@PressureCoefficient	MEASURE	Coefficient for pressure increase (example: 1e5 Pa/s)
@SensitivityAnalysis	SUBDICT [Sensitivity-Analysis]	Dictionary containing additional options for sensitivity analysis (see Section 7.4)
@Options	SUBDICT [Output-Options]	Dictionary containing additional options for solving the batch reactor (see Section 7.3)
@OdeParameters	SUBDICT [ODE-Solver]	Dictionary containing the numerical parameters for solving the ODE system (see Section 7.2)
@ParametricAnalysis	SUBDICT [Parametric-Analysis]	Dictionary containing additional options for performing a parametric analysis (see Section 7.8)

Table 6.2: Batch Reactor dictionary

@VolumeProfile

Dictionary defining the volume profile.

@PressureCoefficien

Coefficient for pressure increase (example: 1e5 Pa/s)

@SensitivityAnalysis

The user has to specify the name of the sub-dictionary in which the options for performing the sensitivity analysis are reported. Please look at Section 7.4 to have more details.

@Options

The user has to specify the name of the sub-dictionary in which the options devoted to the output operations are reported. Please look at Section 7.3 to have more details.

@OdeParameters

The user has to specify the name of the sub-dictionary in which the options about the ODE solver adopted for the solution of the reactor equations. Please look at Section 7.2 to have more details.

@ParametricAnalysis

Dictionary containing additional options for performing a parametric analysis. Please look at Section 7.8 to have more details.

6.3 Dictionary: Perfectly Stirred Reactor

This dictionary is used for setting the simulation for a Perfectly Stirred Reactor operating in unsteady or steady-state conditions. In the current version of `OpenSMOKE++ Suite` the user can simulate both isothermal and adiabatic reactors, only under the assumption of constant pressure.

6.3.1 Additional comments

@KineticsFolder

This option is used to specify the name of the folder containing the kinetic scheme in XML Version (`kinetics.xml`). This file has to be generated using the `OpenSMOKE_CHEMKIN_Preprocessor`. Instead of using this option, the user has the possibility to pre-process the kinetic scheme on the fly using the `@KineticsPreProcessor` option (see below).

@KineticsPreProcessor

The user can pre-process a kinetic scheme available in CHEMKIN format on the fly, when the reactor simulation is performed (instead of pre-processing the kinetics in a previous step). This option is very useful when the kinetic mechanism is under construction and/or tuning and the user has the need to change often the reactions and the kinetic parameters. Please consider that the pre-processing operation can be quite long for very detailed kinetic schemes (thousands of species). The `@KineticsPreProcessor` option requires that the user specify the name of the sub-dictionary containing all the information needed to pre-process kinetic mechanisms on the fly. Please look at Section 7.6 to have more details.

@Type

This option is used to specify the type of reactor to simulate. Two different combinations are possible: `Isothermal-ConstantPressure`, `NonIsothermal-ConstantPressure`.

Option	Type	Meaning
@KineticsFolder	PATH	Name of the folder containing the kinetic scheme (XML Version)
@KineticsPreProcessor	SUBDICT [Rapid-Kinetics]	Dictionary containing the list of kinetic files to be interpreted (see Section 7.6)
@Type	STRING	Perfectly Stirred Reactor type: `Isothermal-ConstantPressure` | `NonIsothermal-ConstantPressure`
@InletStatus	SUBDICT [Gas-Status]	Dictionary defining the inlet gas composition, temperature, and pressure (see Section 7.5)
@InitialStatus	SUBDICT [Gas-Status]	Dictionary defining the initial gas composition, temperature and pressure inside the reactor (see Section 7.5)
@ResidenceTime	MEASURE	Residence time
@EndTime	MEASURE	Integration time for the ODE solution
@Volume	MEASURE	Initial volume of reactor
@MassFlowRate	MEASURE	Inlet mass flow rate
@GlobalThermalExchangeCoefficient	MEASURE	Global thermal exchange coefficient U: $Q = UA(T - T_{env})$
@EnvironmentTemperature	MEASURE	EnvironmentTemperature T_{env}: $Q = UA(T - T_{env})$
@ExchangeArea	MEASURE	Exchange area A: $Q = UA(T - T_{env})$
@SensitivityAnalysis	SUBDICT [Sensitivity-Analysis]	Dictionary containing additional options for sensitivity analysis (see Section 7.4)
@Options	SUBDICT [Output-Options]	Dictionary containing additional options for solving the perfectly stirred reactor (see Section 7.3)
@OdeParameters	SUBDICT [ODE-Solver]	Dictionary containing the numerical parameters for solving the ODE system (see Section 7.2)
@ParametricAnalysis	SUBDICT [Parametric-Analysis]	Dictionary containing additional options for performing a parametric analysis (see Section 7.8)

Table 6.3: Perfectly Stirred Reactor options

@InletStatus

This option is used to specify the features (temperature, pressure and composition) of the inlet mixture in the perfectly stirred reactor. The `@InletStatus` option requires that the user specify the name of the corresponding sub-dictionary. Please look at Section 7.5 to have more details.

@InitialStatus

This option is used to specify the features (temperature, pressure and composition) of the initial mixture (i.e. at time 0) in the perfectly stirred reactor. This information is used only as a first guess solution to find the steady-state solution. The `@InitialStatus` option requires that the user specify the name of the corresponding sub-dictionary. Please look at Section 7.5 to have more details.

@ResidenceTime

This is the residence time of the perfectly stirred reactor.

@EndTime

This is the integration time for the solution of the ODE system. Since the user is interested in steady-state conditions, this number must be extremely large. The default value is 10^6 s.

@Volume

This is the volume of the perfectly stirred reactor.

@MassFlowRate

This is the mass flow rate of the inlet mixture.

@GlobalThermalExchangeCoefficient

Global thermal exchange coefficient U: $Q = UA(T - T_{env})$

@EnvironmentTemperature

EnvironmentTemperature T_{env}: $Q = UA(T - T_{env})$

@ExchangeArea

Exchange area A: $Q = UA(T - T_{env})$

@SensitivityAnalysis

The user has to specify the name of the sub-dictionary in which the options for performing the sensitivity analysis are reported. Please look at Section 7.4 to have more details.

@Options

The user has to specify the name of the sub-dictionary in which the options devoted to the output operations are reported. Please look at Section 7.3 to have more details.

@OdeParameters

The user has to specify the name of the sub-dictionary in which the options about the ODE solver adopted for the solution of the reactor equations. Please look at Section 7.2 to have more details.

@ParametricAnalysis

Dictionary containing additional options for performing a parametric analysis. Please look at Section 7.8 to have more details.

6.4 Dictionary: Shock Tube Reactor

This dictionary is used for setting the simulation for a Shock Tube Reactor. In the current version of `OpenSMOKE++ Suite` the user can simulate both incident and reflected shock conditions.

6.4.1 Additional comments

@KineticsFolder

This option is used to specify the name of the folder containing the kinetic scheme in XML Version (`kinetics.xml`). This file has to be generated using the `OpenSMOKE_CHEMKIN_Preprocessor`. Instead of using this option, the user has the possibility to pre-process the kinetic scheme on the fly using the `@KineticsPreProcessor` option (see below).

@KineticsPreProcessor

The user can pre-process a kinetic scheme available in CHEMKIN format on the fly, when the reactor simulation is performed (instead of pre-processing the kinetics in a previous step). This option is very useful when the kinetic mechanism is under construction and/or tuning and the user has the need to change often the reactions and the kinetic parameters. Please consider that the pre-processing operation can be quite long for very detailed kinetic schemes (thousands of species). The `@KineticsPreProcessor` option requires that the user specify the name of the sub-dictionary containing all the information needed to pre-process kinetic mechanisms on the fly. Please look at Section 7.6 to have more details.

@Type

This option is used to specify the type of reactor to simulate. Two different combinations are possible: `IncidentShock`, `ReflectedShock`.

@BeforeShockStatus @AfterShockStatus @ReflectedShockStatus

These options are used to specify the features (temperature, pressure and composition) of the gas mixture before or after the shock or the features of the reflected shock. These options require that the user specify the name of the corresponding sub-dictionary. Please look at Section 7.5 to have more details.

@IncidentShockVelocity @ReflectedShockVelocity

These options are used to specify the velocity of the incident or the reflected shocks.

@EndTime

Integration time.

@BoundaryLayerCorrection

Boundary layer correction.

@Diameter

Internal diameter of the shock-tube.

@Viscosity

Viscosity of the carrier gas at ambient temperature.

Option	Type	Meaning
@KineticsFolder	PATH	Name of the folder containing the kinetic scheme (XML Version)
@KineticsPreProcessor	SUBDICT [Rapid-Kinetics]	Dictionary containing the list of kinetic files to be interpreted (see Section 7.6)
@Type	STRING	Shock-tube reactor type: `IncidentShock`, `ReflectedShock`
@BeforeShockStatus	SUBDICT [Gas-Status]	Dictionary defining the gas composition, temperature and pressure before the shock (see Section 7.5)
@AfterShockStatus	SUBDICT [Gas-Status]	Dictionary defining the gas composition, temperature and pressure after the shock (see Section 7.5)
@ReflectedShockStatus	SUBDICT [Gas-Status]	Dictionary defining the gas composition, temperature and pressure of the reflected shock (see Section 7.5)
@IncidentShockVelocity	MEASURE	Incident shock velocity
@ReflectedShockVelocity	MEASURE	Reflected shock velocity
@EndTime	MEASURE	Time for transient simulation
@BoundaryLayerCorrection	BOOL	Boundary layer correction
@Diameter	MEASURE	Diameter of the shock tube
@Viscosity	MEASURE	Viscosity of the carrier gas at 300 K
@SensitivityAnalysis	SUBDICT [Sensitivity-Analysis]	Dictionary containing additional options for sensitivity analysis (see Section 7.4)
@Options	SUBDICT [Output-Options]	Dictionary containing additional options for solving the shock tube reactor (see Section 7.3)
@OdeParameters	SUBDICT [ODE-Solver]	Dictionary containing the numerical parameters for solving the ODE system (see Section 7.2)

Table 6.4: Shock Tube Reactor dictionary

@SensitivityAnalysis

The user has to specify the name of the sub-dictionary in which the options for performing the sensitivity analysis are reported. Please look at Section 7.4 to have more details.

@Options

The user has to specify the name of the sub-dictionary in which the options devoted to the output operations are reported. Please look at Section 7.3 to have more details.

@OdeParameters

The user has to specify the name of the sub-dictionary in which the options about the ODE solver adopted for the solution of the reactor equations. Please look at Section 7.2 to have more details.

6.5 Dictionary: Plug Flow Reactor

This dictionary is used for setting the simulation for a Plug Flow reactor. In the current version of `OpenSMOKE++ Suite` the user can simulate both isothermal and adiabatic reactors. Both constant pressure and constant velocity conditions are allowed.

6.5.1 Additional comments

@KineticsFolder

This option is used to specify the name of the folder containing the kinetic scheme in XML Version (`kinetics.xml`). This file has to be generated using the `OpenSMOKE_CHEMKIN_Preprocessor`. Instead of using this option, the user has the possibility to pre-process the kinetic scheme on the fly using the `@KineticsPreProcessor` option (see below).

@KineticsPreProcessor

The user can pre-process a kinetic scheme available in CHEMKIN format on the fly, when the reactor simulation is performed (instead of pre-processing the kinetics in a previous step). This option is very useful when the kinetic mechanism is under construction and/or tuning and the user has the need to change often the reactions and the kinetic parameters. Please consider that the pre-processing operation can be quite long for very detailed kinetic schemes (thousands of species). The `@KineticsPreProcessor` option requires that the user specify the name of the sub-dictionary containing all the information needed to pre-process kinetic mechanisms on the fly. Please look at Section 7.6 to have more details.

@Type

This option is used to specify the type of reactor to simulate. Two different types are possible: `Isothermal` or `NonIsothermal`.

@ConstantPressure

When this option is true, a constant pressure simulation will be performed. If this option is false, the simulation will be performed under the assumption of constant velocity.

@InletStatus

This option is used to specify the features (temperature, pressure and composition) of the inlet mixture in the plug flow reactor. The `@InletStatus` option requires that the user specify the name of the corresponding sub-dictionary. Please look at Section 7.5 to have more details.

@ResidenceTime

This is the residence time for the plug flow reactor.

Option	Type	Meaning
@KineticsFolder	PATH	Name of the folder containing the kinetic scheme (XML Version)
@KineticsPreProcessor	SUBDICT [Rapid-Kinetics]	Dictionary containing the list of kinetic files to be interpreted (see Section 7.6)
@Type	STRING	Plug flow reactor type: `Isothermal`, `NonIsothermal`
@ResidenceTime	MEASURE	Total residence time
@Length	MEASURE	Total length
@ConstantPressure	BOOL	Constant pressure vs Constant velocity simulation
@InletStatus	SUBDICT [Gas-Status]	Name of the dictionary defining the inlet gas composition, temperature and pressure (see Section 7.5)
@Velocity	MEASURE	Velocity of the inlet stream
@MassFlowRate	MEASURE	Mass flow rate of the inlet stream
@MoleFlowRate	MEASURE	Mole flow rate of the inlet stream
@VolumetricFlowRate	MEASURE	Volumetric flow rate of the inlet stream
@Diameter	MEASURE	Diameter of the tube
@TemperatureProfile	SUBDICT [XY-Profile]	Name of the dictionary defining the temperature profile (See Section 7.7)
@GlobalThermalExchangeCoefficient	MEASURE	Global thermal exchange coefficient U: $Q = UA(T - T_{env})$
@EnvironmentTemperature	MEASURE	EnvironmentTemperature T_{env}: $Q = UA(T - T_{env})$
@CrossSectionOverPerimeter	MEASURE	Ratio between the cross section and the perimeter (for a circular section it is equal to $D/4$)
@SensitivityAnalysis	SUBDICT [Sensitivity-Analysis]	Dictionary containing additional options for sensitivity analysis (see Section 7.4)
@Options	SUBDICT [Output-Options]	Dictionary containing additional options for solving the plug flow reactor (see Section 7.3)
@OdeParameters	SUBDICT [ODE-Solver]	Dictionary containing the numerical parameters for solving the ODE system (see Section 7.2)
@ParametricAnalysis	SUBDICT [Parametric-Analysis]	Dictionary containing additional options for performing a parametric analysis (see Section 7.8)

Table 6.5: Plug Flow Reactor dictionary

@Length

This is the length of the plug flow reactor.

@Velocity

This is the velocity of the inlet mixture.

@MassFlowRate

This is the mass flow rate of the inlet mixture.

@MoleFlowRate

This is the mole flow rate of the inlet mixture.

@VolumetricFlowRate

This is the volumetric flow rate of the inlet mixture.

@Diameter

This is the internal diameter of the plug flow reactor.

@TemperatureProfile

The user has the possibility to impose a temperature profile along the plug flow reactor. This temperature profile can be imposed by providing a proper sub-dictionary. Please look at Section 7.7 to have more details.

@GlobalThermalExchangeCoefficient

Global thermal exchange coefficient U: $Q = UA(T - T_{env})$

@EnvironmentTemperature

EnvironmentTemperature T_{env}: $Q = UA(T - T_{env})$

@CrossSectionOverPerimeter

Ratio between the cross section and the perimeter (for a circular section it is equal to $D/4$)

@SensitivityAnalysis

The user has to specify the name of the sub-dictionary in which the options for performing the sensitivity analysis are reported. Please look at Section 7.4 to have more details.

@Options

The user has to specify the name of the sub-dictionary in which the options devoted to the output operations are reported. Please look at Section 7.3 to have more details.

@OdeParameters

The user has to specify the name of the sub-dictionary in which the options about the ODE solver adopted for the solution of the reactor equations. Please look at Section 7.2 to have more details.

@ParametricAnalysis

Dictionary containing additional options for performing a parametric analysis. Please look at Section 7.8 to have more details.

6.6 Dictionary: PremixedLaminarFlame1D

This dictionary is used for setting the simulation for calculating laminar flame speeds.

6.6.1 Additional comments

@KineticsFolder

This option is used to specify the name of the folder containing the kinetic scheme in XML Version (`kinetics.xml`). This file has to be generated using the `OpenSMOKE_CHEMKIN_Preprocessor`. Instead of using this option, the user has the possibility to pre-process the kinetic scheme on the fly using the `@KineticsPreProcessor` option (see below).

@KineticsPreProcessor

The user can pre-process a kinetic scheme available in CHEMKIN format on the fly, when the reactor simulation is performed (instead of pre-processing the kinetics in a previous step). This option is very useful when the kinetic mechanism is under construction and/or tuning and the user has the need to change often the reactions and the kinetic parameters. Please consider that the pre-processing operation can be quite long for very detailed kinetic schemes (thousands of species). The `@KineticsPreProcessor` option requires that the user specify the name of the sub-dictionary containing all the information needed to pre-process kinetic mechanisms on the fly. Please look at Section 7.6 to have more details.

@Grid

Dictionary defining the computational 1D grid (see Section 7.9)

@InletStream

This option is used to specify the features (temperature, pressure and composition) of the inlet stream. The `@InletStatus` option requires that the user specify the name of the corresponding sub-dictionary. Please look at Section 7.5 to have more details.

@OutletStream

Dictionary defining the inlet gas composition, temperature, and pressure (see Section 7.5).

@OutletStream

Dictionary defining the outlet stream (gas composition, temperature and pressure). This information is used only as a first guess solution. See Section 7.5

@InletVelocity

First guess laminar flame speed. This value is used only to start the calculations. Obviously, if data are available, it is more convenient to provide a first-guess laminar flame speed close to the real value.

@Output

Folder where to write the results of the simulation

@SensitivityAnalysis

The user has to specify the name of the sub-dictionary in which the options for performing the sensitivity analysis are reported. Please look at Section 7.4 to have more details.

@DaeParameters

The user has to specify the name of the sub-dictionary in which the options about the tridiagonal-block DAE solver adopted for the solution of transport equations. Please look at Section 7.11 to have more details.

Option	Type	Meaning
@KineticsFolder	PATH	Name of the folder containing the kinetic scheme (XML Version)
@KineticsPreProcessor	SUBDICT [Rapid-Kinetics]	Dictionary containing the list of kinetic files to be interpreted (see Section 7.6)
@Grid	SUBDICT [AdaptiveGrid-1D]	Dictionary defining the computational 1D grid (see Section 7.9)
@InletStream	SUBDICT [Gas-Status]	Dictionary defining the inlet gas composition, temperature, and pressure (see Section 7.5)
@OutletStream	SUBDICT [Gas-Status]	Dictionary defining the outlet stream (gas composition, temperature and pressure). This information is used only as a first guess solution. See Section 7.5
@InletVelocity	MEASURE	Laminar flame speed: first guess value
@Output	PATH	Folder where to write the results of the simulation
@SensitivityAnalysis	SUBDICT [Sensitivity-Analysis]	Dictionary containing additional options for sensitivity analysis (see Section 7.4)
@DaeParameters	SUBDICT [DAE-TridiagonalBlock-Solver]	Dictionary containing the numerical parameters for solving the Triadiagonal Block DAE system (see Section 7.11)
@NlsParameters	SUBDICT [NLS-TridiagonalBlock-Solver]	Dictionary containing the numerical parameters for solving the Triadiagonal Block non-linear system (see Section 7.12)
@FalseTransientParameters	SUBDICT [FalseTransient-Solver]	Dictionary containing the numerical parameters governing the false-transient approach (see Section 7.13)
@CheckMassFractions	BOOL	Force non-negative mass fractions (default: false)
@UseDaeSolver	BOOL	Use DAE solver (instead of NL solver) to solve the steady-state problems (default: true)
@SpeciesBundling	DOUBLE	Enables the species bundling technique for the fast evaluation of mass diffusion coefficients. The double parameter is the maximum relative error which can be accepted (suggestion: 0.05)

Table 6.6: PremixedLaminarFlame1D options

@NlsParameters

The user has to specify the name of the sub-dictionary in which the options about the tridiagonal-block non-linear (NL) solver adopted for the solution of transport equations. Please look at Section 7.12 to have more details.

@FalseTransientParameters

The user has to specify the name of the sub-dictionary in which the options about the false-transient method adopted for the solution of transport equations. Please look at Section 7.13 to have more details.

@CheckMassFractions

Force non-negative mass fractions (default: false).

@UseDaeSolver

Use DAE solver (instead of NL solver) to solve the steady-state problems (default: true)

@SpeciesBundling

Enables the species bundling technique for the fast evaluation of mass diffusion coefficients. The double parameter is the maximum relative error which can be accepted (suggestion: 0.05). The technique is described in the following paper: **Lu and Law**, *Diffusion coefficient reduction through species bundling*, Combustion and Flame, 148(3), p. 117-126 (2007).

6.7 Dictionary: Thermodynamic Equilibrium

This dictionary is used for setting the simulation for thermodynamic equilibrium calculations. In the current version of `OpenSMOKE++ Suite` the user can calculating equilibrium conditions of a homogeneous mixture of gases, under two different constraints: fixed enthalpy and pressure or fixed temperature and pressure.

6.7.1 Additional comments

@KineticsFolder

This option is used to specify the name of the folder containing the kinetic scheme in XML Version (`kinetics.xml`). This file has to be generated using the `OpenSMOKE_CHEMKIN_Preprocessor`. Instead of using this option, the user has the possibility to pre-process the kinetic scheme on the fly using the `@KineticsPreProcessor` option (see below).

@KineticsPreProcessor

The user can pre-process a kinetic scheme available in CHEMKIN format on the fly, when the reactor simulation is performed (instead of pre-processing the kinetics in a previous step). This option is very useful when the kinetic mechanism is under construction and/or tuning and the user has the need to change often the reactions and the kinetic parameters. Please consider that the pre-processing operation can be quite long for very detailed kinetic schemes (thousands of species). The `@KineticsPreProcessor` option requires that the user specify the name of the sub-dictionary containing all the information needed to pre-process kinetic mechanisms on the fly. Please look at Section 7.6 to have more details.

@Type

In the current version of `OpenSMOKE++ Suite` the user can calculating equilibrium conditions of a homogeneous mixture of gases, under two different constraints: fixed enthalpy and pressure (`Fixed_H_P`) or fixed temperature and pressure (`Fixed_T_P`).

Option	Type	Meaning
@KineticsFolder	PATH	Name of the folder containing the kinetic scheme (XML Version)
@KineticsPreProcessor	SUBDICTIONARY [Rapid-Kinetics]	Dictionary containing the list of kinetic files to be interpreted (see Section 7.6)
@Type	STRING	Type of simulation to be carried out (fixed enthalpy and pressure or fixed temperature and pressure): `Fixed_H_P | Fixed_T_P`
@InitialStatus	SUBDICTIONARY [Gas-Status]	Dictionary defining the initial gas composition, temperature and pressure (see Section 7.5)
@Output	PATH	Path to the folder where the results have to be written
@OutputSpecies	STRINGS	List of species which will be written on ASCII file
@VerbosityLevel	MEASURE	Verbosity level: default is 0 (basically no details are written on the screen), maximum level is 5 (every detail about the calculations is reported on the screen)

Table 6.7: Thermodynamic Equilibrium dictionary

@InitialStatus

This option is used to specify the features (temperature, pressure and composition) of the initial mixture. The `@InitialStatus` option requires that the user specify the name of the corresponding sub-dictionary. Please look at Section 7.5 to have more details.

@Output

Path to the folder where the results have to be written.

@OutputSpecies

List of species which will be written on ASCII file. `ALL` means that every species contained in the kinetic mechanism is written on the file.

@Verbosity Level

Verbosity level: default is 0 (basically no details are written on the screen), maximum level is 5 (every detail about the calculations is reported on the screen).

6.8 Dictionary: LaminarFlamelet

This dictionary is used for setting the simulation for calculating laminar flame speeds.

6.8.1 Additional comments

@KineticsFolder

This option is used to specify the name of the folder containing the kinetic scheme in XML Version (`kinetics.xml`). This file has to be generated using the `OpenSMOKE_CHEMKIN_Preprocessor`. Instead of using this option, the user has the possibility to pre-process the kinetic scheme on the fly using the `@KineticsPreProcessor` option (see below).

@KineticsPreProcessor

The user can pre-process a kinetic scheme available in CHEMKIN format on the fly, when the reactor simulation is performed (instead of pre-processing the kinetics in a previous step). This option is very useful when the kinetic mechanism is under construction and/or tuning and the user has the need to change often the reactions and the kinetic parameters. Please consider that the pre-processing operation can be quite long for very detailed kinetic schemes (thousands of species). The `@KineticsPreProcessor` option requires that the user specify the name of the sub-dictionary containing all the information needed to pre-process kinetic mechanisms on the fly. Please look at Section 7.6 to have more details.

@Backup

Name (local or full path) of backup file (in xml format) to be used to restart the simulation.

@Grid

Dictionary defining the computational 1D grid (see Section 7.9)

@FuelStream

This option is used to specify the features (temperature, pressure and composition) of the inlet stream. The `@InletStatus` option requires that the user specify the name of the corresponding sub-dictionary. Please look at Section 7.5 to have more details.

@OxidizerStream

Dictionary defining the composition, temperature, and pressure of fuel stream (see Section 7.5).

Option	Type	Meaning
@KineticsFolder	PATH	Name of the folder containing the kinetic scheme (XML Version)
@KineticsPreProcessor	SUBDICT [Rapid-Kinetics]	Dictionary containing the list of kinetic files to be interpreted (see Section 7.6)
@Backup	PATH	Name (local or full path) of backup file (in xml format) to be used to restart the simulation
@Grid	SUBDICT [AdaptiveGrid-1D]	Dictionary defining the computational 1D grid (see Section 7.9)
@FuelSide	SUBDICT [Gas-Status]	Dictionary defining the composition, temperature, and pressure of fuel stream (see Section 7.5)
@OxidizerSide	SUBDICT [Gas-Status]	Dictionary defining the composition, temperature, and pressure of oxidizer stream. See Section 7.5
@StoichiometricScalarDissipationRates	VECTOR_MEASURE	List of stoichiometric scalar dissipation rates to be simulated. The equilibrium flamelet (i.e. at scalar dissipation rate equal to 0) is always automatically calculated. Example: `@StoichiometricScalarDissipationRates 1e-3 1e-1 10. Hz;`
@EnthalpyDefects	VECTOR_MEASURE	List of enthalpy defects to be simulated. The adiabatic flamelet (i.e. at enthalpy defect equal to 0) is always automatically calculated. Example: `@EnthalpyDefects 250 500 750 1000 kJ/kg;`
@SensitivityAnalysis	SUBDICT [Sensitivity-Analysis]	Dictionary containing additional options for sensitivity analysis (see Section 7.4) [Not yet available]
@OdeParameters	SUBDICT [ODE-TridiagonalBlock-Solver]	Dictionary containing the numerical parameters for solving the Triadiagonal Block ODE system (see Section 7.10)
@NlsParameters	SUBDICT [NLS-TridiagonalBlock-Solver]	Dictionary containing the numerical parameters for solving the Triadiagonal Block non-linear system (see Section 7.12)
@FalseTransientParameters	SUBDICT [FalseTransient-Solver]	Dictionary containing the numerical parameters governing the false-transient approach (see Section 7.13)
@CheckMassFractions	BOOL	Force non-negative mass fractions (default: false)
@UseOdeSolver	BOOL	Use ODE solver (instead of NLS solver) to solve the steady-state problems (default: true)
@UseNlsSolver	BOOL	Use NLS solver (in addition to the ODE solver) to solve the steady-state problems (default: true)
@DensityCorrectionOnScalarDissipationRate	BOOL	Applies a correction to the scalar dissipation rate to account for variations of densities along the flamelet (default: false)
@MinimumTemperature	MEASURE	Minimum temperature which can be accepted for non-adiabatic flamelets (default: 280 K)
@ExtinguishmentTemperature	MEASURE	Temperature below which the flame can be considered extinguished (default: 1000 K)

Table 6.8: LaminarFlamelet dictionary

@OutletStream

Dictionary defining the composition, temperature, and pressure of oxidizer stream (see Section 7.5).

@StoichiometricScalarDissipationRates

List of stoichiometric scalar dissipation rates to be simulated. The equilibrium flamelet (i.e. at scalar dissipation rate equal to 0) is always automatically calculated. Example: `@StoichiometricScalarDissipationRates 1e-3 1e-1 10. Hz;`

@EnthalpyDefects

List of enthalpy defects to be simulated. The adiabatic flamelet (i.e. at enthalpy defect equal to 0) is always automatically calculated. Example: `@EnthalpyDefects 250 500 750 1000 kJ/kg;`

@Output

Folder where to write the results of the simulation

@SensitivityAnalysis

The user has to specify the name of the sub-dictionary in which the options for performing the sensitivity analysis are reported. Please look at Section 7.4 to have more details.

@OdeParameters

The user has to specify the name of the sub-dictionary in which the options about the tridiagonal-block ODE solver adopted for the solution of transport equations. Please look at Section 7.10 to have more details.

@NlsParameters

The user has to specify the name of the sub-dictionary in which the options about the tridiagonal-block non-linear (NL) solver adopted for the solution of transport equations. Please look at Section 7.12 to have more details.

@FalseTransientParameters

The user has to specify the name of the sub-dictionary in which the options about the false-transient method adopted for the solution of transport equations. Please look at Section 7.13 to have more details.

@CheckMassFractions

Force non-negative mass fractions (default: false).

@UseOdeSolver

Use ODE solver (instead of NLS solver) to solve the steady-state problems (default: true)

@UseNlsSolver

Use NLS solver (in addition to the ODE solver) to solve the steady-state problems (default: true)

@DensityCorrectionOnScalarDissipationRate

Applies a correction to the scalar dissipation rate to account for variations of densities along the flamelet (default: false). The correction factor is: $\frac{3}{8}\frac{\left(\sqrt{\frac{\rho_{ox}}{\rho_{fuel}}}+1\right)^2}{\sqrt{\frac{\rho_{ox}}{\rho_{fuel}}}+1}$. Thus, the resulting scalar diffispation rate is given by: $\chi = \frac{a_s}{\pi} exp\left(-2\left(erfc^{-1}\left(2z_{st}\right)\right)^2\right)\frac{3}{8}\frac{\left(\sqrt{\frac{\rho_{ox}}{\rho_{fuel}}}+1\right)^2}{\sqrt{\frac{\rho_{ox}}{\rho_{fuel}}}+1}$

@MinimumTemperature

Minimum temperature which can be accepted for non-adiabatic flamelets (default: 280 K).

@ExtinguishmentTemperature

Temperature below which the flame can be considered extinguished (default: 1000 K).

6.9 Dictionary: LookUpTables

This dictionary is used for setting the simulation for calculating laminar flame speeds.

6.9.1 Additional comments

@KineticsFolder

This option is used to specify the name of the folder containing the kinetic scheme in XML Version (`kinetics.xml`). This file has to be generated using the `OpenSMOKE_CHEMKIN_Preprocessor`. Instead of using this option, the user has the possibility to pre-process the kinetic scheme on the fly using the `@KineticsPreProcessor` option (see below).

@KineticsPreProcessor

The user can pre-process a kinetic scheme available in CHEMKIN format on the fly, when the reactor simulation is performed (instead of pre-processing the kinetics in a previous step). This option is very useful when the kinetic mechanism is under construction and/or tuning and the user has the need to change often the reactions and the kinetic parameters. Please consider that the pre-processing operation can be quite long for very detailed kinetic schemes (thousands of species). The `@KineticsPreProcessor` option requires that the user specify the name of the sub-dictionary containing all the information needed to pre-process kinetic mechanisms on the fly. Please look at Section 7.6 to have more details.

@Input

Name (local or full path) of folder containing the XML files corresponding to the different laminar flamelets to be post-processed

@OutputASCII

Name of the folder where to write the output ASCII files

@OutputLookUpTable

Name of the folder where to write the pre-processed lookup table (in XML format)

@VarianceSlices

Number of slices in the mixture fraction variance space (default: 32)

@VarianceStretchingFactor

Stretching factor for building the mixture fraction variance space (default: 1.17)

@MaximumAllowedNormalizedVariance

Maximum allowed normalized variance (default: 0.98)

@PDFType

Type of PDF: `Beta` | `ClippedGaussian` (default: `Beta`)

Option	Type	Meaning
@KineticsFolder	PATH	Name of the folder containing the kinetic scheme (XML Version)
@KineticsPreProcessor	SUBDICT [Rapid-Kinetics]	Dictionary containing the list of kinetic files to be interpreted (see Section 7.6)
@Input	PATH	Name (local or full path) of folder containing the XML files corresponding to the different laminar flamelets to be post-processed
@OutputASCII	PATH	Name of the folder where to write the output ASCII files
@OutputLookUpTable	PATH	Name of the folder where to write the pre-processed lookup table (in XML format)
@VarianceSlices	INT	Number of slices in the mixture fraction variance space (default: 32)
@VarianceStretchingFactor	DOUBLE	Stretching factor for building the mixture fraction variance space (default: 1.17)
@MaximumAllowedNormalizedVariance	DOUBLE	Maximum allowed normalized variance (default: 0.98)
@PDFType	STRING	Type of PDF: `Beta` | `ClippedGaussian` (default: `Beta`)
@Overwrite	BOOL	Force overwriting of previous (if existing) output folder (default: false)

Table 6.9: LookUpTables dictionary

@Overwrite

Force overwriting of previous (if existing) output folder (default: false)

6.10 Dictionary: MicrogravityDroplet (under development)

This dictionary is used for setting the simulation for simulating evaporation and combustion of isolated fuel droplets in microgravity conditions. Please, consider that the current version of OpenSMOKE_MicrogravityDroplet tool is only preliminary. The solver is under development.

Option	Type	Meaning
@KineticsFolder	PATH	Name of the folder containing the kinetic scheme (XML Version)
@OutputFolder	PATH	Folder where to write the results of the simulation
@Type	STRING	Type of simulation: `Spark` (spark or hot-wire ignitions) | `NoSpark` (autoignitions)
@Fiber	SUBDICT [Fiber]	Name of the dictionary defining the fiber (see Section 7.14)
@DropletDiameter	MEASURE	Initial diameter of the droplet
@DropletTemperature	MEASURE	Initial temperature of the droplet (which is assumed perfectly uniform)
@FuelMassFractions	STRINGS + DOUBLES	Mass fractions of the liquid mixture (the sum must be equal to 1)
@FuelMoleFractions	STRINGS + DOUBLES	Mole fractions of the liquid mixture (the sum must be equal to 1)
@FuelMasses	STRINGS + DOUBLES	Masses of the liquid mixture components
@FuelMoles	STRINGS + DOUBLES	Moles of the liquid mixture components
@Inert	STRING	Name of inert species
@GasStatus	SUBDICT [Gas-Status]	Dictionary defining the gaseous atmosphere composition, temperature, and pressure (see Section 7.5)
@DropletPoints	INT	Number of points of the grid describing the liquid phase (i.e. the droplet)
@GasPoints	INT	Number of points of the grid describing the gaseous phase
@EndTime	MEASURE	Time to be simulated
@Soret	BOOL	Soret effect (default: `false`)
@SpeciesBundling	DOUBLE	Enables the species bundling technique for the fast evaluation of mass diffusion coefficients. The double parameter is the maximum relative error which can be accepted (suggestion: 0.05)
@EnhancingFactorLiquidThermalConductivity	DOUBLE	Enhancing factor for thermal conductivity of droplet (default 1.)
@EnhancingFactorLiquidMassDiffusionCoefficients	DOUBLE	Enhancing factor for mass diffusion coefficients of droplet (default 1.)
@OuterTemperatureProfile	SUBDICT [XY-Profile]	Dictionary defining the outer temperature profile in time (see Section 7.7)

Table 6.10: MicrogravityDroplet options (Part 1)

Option	Type	Meaning
@FlameRadiation	STRING	Model to be adopted for describing the radiative heat transfer: `none` | `analytical` | `optically-thin` | `discrete-ordinates-S2nonsym` | `discrete-ordinates-S2sym` | `P1` | `SP3`
@ExtinctionCoefficient	STRING	Model for calculating the extinction coefficients: `gray` (default) | `WSGG-Smith` | `WSGG-Truelove` | `WSGG-Yin`
@RadiationCorrectionCoefficient	DOUBLE	Correction coefficient for radiation (default 1.)
@DropletEmissivity	DOUBLE	Emissivity of the droplet surface (default 1.)
@ChamberEmissivity	DOUBLE	Emissivity of the chamber internal surface (default 1.)
@FirstGuessInterfaceTemperature	MEASURE	First guess temperature of the liquid/gas interface at time $t = 0$
@FirstGuessInterfaceMassFractions	STRINGS + DOUBLES	First guess mass fractions of the liquid/gas interface at time $t = 0$
@FirstGuessInterfaceVelocity	MEASURE	First guess velocity (gas side) of the liquid/gas interface at time $t = 0$
@SparkTemperature	MEASURE	Temperature of the spark (default 1800 K)
@SparkCoordinates	DOUBLES	Dimensionless coordinates (number of droplet radii) of spark (default: 1.01 2 5 6)
@SparkGeneratedHeatPeak	MEASURE	Thermal power per unit volume generated by the spark (default: 0 W/m3)
@SparkGeneratedHeatDuration	MEASURE	Spark duration (default: 100 ms)
@SparkGeneratedHeatTime	MEASURE	Spark time (default: 1 s)
@SparkGeneratedHeatCoordinates	DOUBLES	Dimensionless coordinates (number of droplet radii) of spark (default: 3 5)

Table 6.11: MicrogravityDroplet options (Part 2)

Option	Type	Meaning
@NLSAbsoluteTolerance	DOUBLE	Absolute tolerance for the solution of NLS
@NLSRelativeTolerance	DOUBLE	Relative tolerance for the solution of NLS
@DAEAbsoluteTolerance	DOUBLE	Absolute tolerance for the solution of DAE systems
@DAERelativeTolerance	DOUBLE	Relative tolerance for the solution of DAE systems
@NumberOfNLS	INT	Number of attempts to solve the NLS system describing the vapor-liquid equilibrium at interface
@IncreasingFactorOfNLS	DOUBLE	Coefficient measuring the increase of default tolerances to be applied to the successive attempts of solutions of NLS
@StepsVideo	INT	Number of steps for output on video
@StepsMap1D	INT	Number of steps for 1D map output
@StepsMap3D	INT	Number of steps for 3D map output
@SnapshotTimes	DOUBLES + MEASURE	Times for snapshots (example: `@SnapshotTimes 0.1 1.  2 s;`)
@SnapshotSpecies	STRINGS	Species for snapshots (example: `@SnapshotSpecies CO2 O2;`)
@VerboseContributions	STRINGS	List of species for which all the terms in the transport equations will be written on output files (example: `@VerboseContributions CO2 O2;`)
@Backup	PATH	Name of backup file (XML Version)
@BackupTime	MEASURE	Backup time (default: 1 h)
@PolimiSoot	SUBDICT [Polimi-Soot]	Name of the dictionary defining the rules for analyzing soot calculated using the Polimi mechanis (see Section 7.15)

Table 6.12: MicrogravityDroplet options (Part 3)

Option	Type	Meaning
@PrevaporizationCoefficient	DOUBLE	Prevaporization coefficient (default 0, max 1)
@VaporizationAccelerationCoefficient	DOUBLE	Correction coefficient for accelerating the droplet vaporization (default: 10)
@VaporizationAccelerationTime	MEASURE	Maximum time (i.e. duration) for accelerating the droplet vaporization (default: 1 ms)
@DistanceOfThermalWave	DOUBLE	Distance of thermal wave in number of radii
@TemperatureOfThermalWave	MEASURE	Temperature of thermal wave (e.g. 298 K)
@DerivativeGasTemperature	STRING	Derivative of gas temperature: `upwind` | `forward` | `backward` | `centered` (default: `upwind`)
@DerivativeGasVelocity	STRING	Derivative of gas velocity: `upwind` | `forward` | `backward` | `centered` (default: `backward`)
@DerivativeGasMassFractions	STRING	Derivative of gas mass fractions: `upwind` | `forward` | `backward` | `centered` (default: `upwind`)
@DerivativeGasMoleFractions	STRING	Derivative of gas mole fractions: `upwind` | `forward` | `backward` | `centered` (default: `backward`)
@DerivativeLiquidTemperature	STRING	Derivative of liquid temperature: `upwind` | `forward` | `backward` | `centered` (default: `upwind`)
@DerivativeLiquidVelocity	STRING	Derivative of liquid velocity: `upwind` | `forward` | `backward` | `centered` (default: `backward`)
@DerivativeLiquidMassFractions	STRING	Derivative of liquid mass fractions: `upwind` | `forward` | `backward` | `centered` (default: `upwind`)
@DerivativeLiquidMoleFractions	STRING	Derivative of liquid mass fractions: `upwind` | `forward` | `backward` | `centered` (default: `backward`)
@DerivativeLiquidDensity	STRING	Derivative of liquid density: `upwind` | `forward` | `backward` | `centered` (default: `upwind`)

Table 6.13: MicrogravityDroplet options (Part 4)

Chapter 7

Sub-Dictionaries

This Chapter reports the sub-dictionaries which are used by the different solvers (as explained in Chapter 6):

1. Comments
2. ODE-Solver
3. Output-Options
4. Sensitivity-Analysis
5. Gas-Status
6. Rapid-Kinetics
7. XY-Profile
8. Parametric Analysis
9. Adaptive-Grid1D
10. Ode-TridiagonalBlock-Parameters
11. Dae-TridiagonalBlock-Parameters
12. Nls-TridiagonalBlock-Parameters
13. FalseTransient-Parameters
14. Fiber
15. PolimiSoot

7.1 Sub-Dictionary: Comments

This dictionary can be used to add comments to the output files written by some of the `OpenSMOKE++ Suite` solvers (see Table 7.1)

7.1.1 Additional comments

@Author

Name of author/authors who developed the kinetic mechanism.

@Place

The place where the kinetic mechanism was developed.

Option	Type	Meaning
@Author	STRINGS	Name of the author
@Place	STRINGS	Name of the place
@Comments	STRINGS	Comments

Table 7.1: Comments dictionary

@Comments

Any useful comment to be attached to the kinetic mechanism (i.e. details about modified kinetic parameters, warnings about missing reactions, status of the mechanism, etc.)

7.1.2 Example

```
Dictionary CRECK_Signature
{
        @Author     CRECK Modeling Group;
        @Place      Politecnico di Milano (Italy);
        @Comments   This kinetic mechanism can be downloaded at:
                www.creckmodeling.chem.polimi.it;
}
```

7.2 Sub-Dictionary: ODE-Parameters

This dictionary is used for setting the options for the stiff ODE solvers available in `OpenSMOKE++ Suite` solvers (see Table 7.2).

7.2.1 Additional comments

@OdeSolver

The user can choose among differen ODE solvers. The OpenSMOKE++ native ODE solver (`OpenSMOKE`) is quite effective especially with very stiff kinetic mechanisms. We suggest to use it when the kinetic mechanisms from CRECK Modeling Group are adopted, or more in general for kinetic mechanisms based on the lumped procedures. The `CVODE` solver is the first option for very large kinetic mechanisms (thousands of species). More in general, we suggest to perform some preliminary tests in order to find the faster ODE solver for the kinetic mechanism under investigation.

@RelativeTolerance

Relative tolerance (default 1.2e-5).

@AbsoluteTolerance

Absolute tolerance (default 1.2e-5). For `CVODE` solver we suggest to use stricter bsolute tolerance (i.e. at least 1e-14).

@MaximumNumberOfSteps

Maximum number of steps (default 500000)

@MaximumStep

Maximum step (default: automatically chosen by the solver)

@MinimumStep

Minimum step (default: automatically chosen by the solver)

@InitialStep

Initial step (default: automatically chosen by the solver)

Option	Type	Meaning
@OdeSolver	STRING	ODE Solver: OpenSMOKE | CVODE | DASPK | DVODE | DLSODA | DLSODE | MEBDF | RADAU5
@RelativeTolerance	DOUBLE	Relative tolerance (default 1.2e-5)
@AbsoluteTolerance	DOUBLE	Absolute tolerance (default 1.e-10)
@MaximumNumberOfSteps	INT	Maximum number of steps (default 500000)
@MaximumStep	DOUBLE	Maximum step (default: automatically chosen by the solver)
@MinimumStep	DOUBLE	Minimum step (default: automatically chosen by the solver)
@InitialStep	DOUBLE	Initial step (default: automatically chosen by the solver)

Table 7.2: ODE-Parameters dictionary

7.2.2 Example

```
Dictionary ode-parameters
{
        @OdeSolver          OpenSMOKE;
        @AbsoluteTolerance  1e-14;
        @RelativeTolerance  1e-7;
}
```

7.3 Sub-Dictionary: Output-Options

This dictionary is used for setting additional options governing the output for many `OpenSMOKE++ Suite` solvers (see Table 7.3).

7.3.1 Additional comments

@OutputFolder

Name of the folder where to write the output data (default: `Output`).

@StepsVideo

Parameter governing the frequency of output on video (default: 50).

@StepsFile

Parameter governing the frequency of output on video (default: 5).

@OutputSpecies

List of species which will be written on ASCII file (default: all the species).

@Verbose

If false, it means that video info and output files will not be written (default: true)

@VerboseASCIIFile

If false, it means that output ASCII file will not be written (default: true).

@VerboseXMLFile

If false, it means that output XML file will not be written (default: true).

7.3.2 Example

```
Dictionary output-options
{
        @OutputFolder      MyOutput;
        @OutputSpecies     H2 O2 CH4;
        @StepsFile         1;
        @StepsVideo        10;
}
```

7.4 Sub-Dictionary: Sensitivity-Analysis

This dictionary is used for setting the options governing the sensitivity analysis.

Option	Type	Meaning
@OutputFolder	PATH	Name of the folder where to write the output data
@StepsVideo	INT	Parameter governing the frequency of output on video
@StepsFile	INT	Parameter governing the frequency of output on file
@OutputSpecies	STRINGS	List of species which will be written on ASCII file
@Verbose	BOOL	If set false means that video info and output files will not be written
@VerboseASCIIFile	BOOL	If set false means that output ASCII file will not be written
@VerboseXMLFile	BOOL	If set false means that output XML file will not be written

Table 7.3: Output-Options dictionary

Option	Type	Meaning
@Type	STRING	Type of sensitivity analysis: `arrhenius-parameters` | `kinetic-constants`
@DenseFullPivoting	BOOL	Full pivoting vs Partial pivoting (LU decomposition)
@DenseSolver	STRING	Linear algebra package: `Eigen`
@SubSteps	INT	Number of sub-steps when performing sensitivity analysis (default: 2)
@Species	STRINGS	List of species for which the sensitivity coefficients will be written

Table 7.4: Sensitivity-Analysis dictionary

7.4.1 Additional comments

@Type

Type of sensitivity analysis. In the current version of OpenSMOKE++ the sensitivity anlysis can be performed with respect to the frequency factor corresponding to each reaction (`arrhenius-parameters`) or with respct to the whole kinetic constant (`kinetic-constants`);

@DenseFullPivoting

Full pivoting vs Partial pivoting (LU decomposition). The solution of linear systems is the most expensive operation of sensitivity analysis. In order to speed-up the calculations, a partial pivoting strategy can be applied when the LU factorization is applied.

@DenseSolver

Linear algebra package. In the current version only the `Eigen` solver is available.

@SubSteps

Number of sub-steps when performing sensitivity analysis (default: 2)

@Species

List of species for which the sensitivity coefficients will be written. Please consider that for large kinetic mechanism the overall output resulting from the sensitivity analysis can be huge. We suggest to write senstiity coefficients only for species in which the user is really interested.

7.4.2 Example

```
Dictionary sensitivity-options
{
        @Type                arrhenius-parameters;
        @DenseSolver         Eigen;
        @DenseFullPivoting   false;
        @SubSteps            5;
        @Species             H2 O2 NO;
}
```

7.5 Sub-Dictionary: Gas-Status

This dictionary is used for setting the status of a mixture (composition, temperature and pressure).

7.5.1 Additional comments

@Temperature

Temperature of the mixture (must be used together with `@Pressure` or `@Density`).

@Pressure

Pressure of the mixture (must be used together with `@Temperature` or `@Density`).

@Density

Density of the mixture (must be used together with `@Temperature` or `@Pressure`).

Option	Type	Meaning
@Temperature	MEASURE	Temperature of the mixture
@Pressure	MEASURE	Pressure of the mixture
@Density	MEASURE	Density of the mixture
@MoleFractions	STRINGS + DOUBLES	Mole fractions of the mixture (the sum must be equal to 1)
@MassFractions	STRINGS + DOUBLES	Mass fractions of the mixture (the sum must be equal to 1)
@Moles	STRINGS + DOUBLES	Molar composition of the mixture (the values will be automatically normalized)
@Masses	STRINGS + DOUBLES	Mass composition of the mixture (the values will be automatically normalized)
@EquivalenceRatio	DOUBLE	Equivalence ratio
@FuelMoleFractions	STRINGS + DOUBLES	Mole fractions of the fuel mixture (the sum must be equal to 1)
@FuelMassFractions	STRINGS + DOUBLES	Mass fractions of the fuel mixture (the sum must be equal to 1)
@FuelMoles	STRINGS + DOUBLES	Molar composition of the fuel mixture (the values will be automatically normalized)
@FuelMasses	STRINGS + DOUBLES	Mass composition of the fuel mixture (the values will be automatically normalized)
@OxidizerMoleFractions	STRINGS + DOUBLES	Mole fractions of the oxidizer mixture (the sum must be equal to 1)
@OxidizerMassFractions	STRINGS + DOUBLES	Mass fractions of the oxidizer mixture (the sum must be equal to 1)
@OxidizerMoles	STRINGS + DOUBLES	Molar composition of the oxidizer mixture (the values will be automatically normalized)
@OxidizerMasses	STRINGS + DOUBLES	Mass composition of the oxidizer mixture (the values will be automatically normalized)

Table 7.5: Gas-Status dictionary

@MoleFractions

Mole fractions of the mixture (the sum must be exactly equal to 1). It is enough to completely characterize the mixture composition (i.e. no additional data are needed).

@MassFractions

Mass fractions of the mixture (the sum must be exactly equal to 1). It is enough to completely characterize the mixture composition (i.e. no additional data are needed).

@Moles

Molar composition of the mixture. The values will be automatically normalized, i.e. converted in mole fractions. It is enough to completely characterize the mixture composition (i.e. no additional data are needed).

@Masses

Mass composition of the mixture. The values will be automatically normalized, i.e. converted in mass fractions. It is enough to completely characterize the mixture composition (i.e. no additional data are needed).

@EquivalenceRatio

Equivalence ratio of the mixture. It requires the compostion of fuel mixture (defined through one of the following: `@FuelMoleFractions` | `@FuelMassFractions` | `@FuelMoles` | `@FuelMasses`) and the compostion of oxidizer mixture (defined through one of the following: `@OxidizerMoleFractions` | `@OxidizerMassFractions` | `@OxidizerMoles` | `@OxidizerMasses`).

@FuelMoleFractions

Mole fractions of the fuel mixture (the sum must be exactly equal to 1). It can be used only in conjuction with `@EquivalenceRatio`.

@FuelMassFractions

Mass fractions of the fuel mixture (the sum must be exactly equal to 1). It can be used only in conjuction with `@EquivalenceRatio`.

@FuelMoles

Molar composition of the fuel mixture. The values will be automatically normalized, i.e. converted in mole fractions. It can be used only in conjuction with `@EquivalenceRatio`.

@FuelMasses

Mass composition of the fuel mixture. The values will be automatically normalized, i.e. converted in mass fractions. It can be used only in conjuction with `@EquivalenceRatio`.

@OxidizerMoleFractions

Mole fractions of the oxidizer mixture (the sum must be exactly equal to 1). It can be used only in conjuction with `@EquivalenceRatio`.

@OxidizerMassFractions

Mass fractions of the oxidizer mixture (the sum must be exactly equal to 1). It can be used only in conjuction with `@EquivalenceRatio`.

@OxidizerMoles

Molar composition of the oxidizer mixture. The values will be automatically normalized, i.e. converted in mole fractions. It can be used only in conjuction with `@EquivalenceRatio`.

@OxidizerMasses

Mass composition of the oxidizer mixture. The values will be automatically normalized, i.e. converted in mass fractions. It can be used only in conjuction with `@EquivalenceRatio`.

7.5.2 Examples

```
Dictionary inlet-mixture
{
        @Temperature    1000.    K ;
        @Pressure       101325   Pa;
        @Moles          H2 2
                    O2 1
                    N2 3.76 ;
}
```

```
Dictionary inlet-mixture
{
        @Temperature        1000.   K ;
        @Pressure           101325. Pa ;
        @EquivalenceRatio   1.;
        @FuelMasses         H2 10.;
        @OxidizerMoles      O2 21
                        N2 79;
}
```

7.6 Sub-Dictionary: Rapid-Kinetics

This dictionary is used when the user wants to pre-process a kinetic scheme on the fly, i.e. just before running the simulation of a reactor.

7.6.1 Additional comments

@Thermodynamics

Name of the file containing the thermodynamic data (CHEMKIN format)

@Transport

Name of the file containing the transport data (CHEMKIN format)

@Kinetics

Name of the file containing the kinetic mechanism (CHEMKIN format)

@Output

Name of the folder where to write the interpreted kinetic mechanism (XML format) (default: `kinetics`)

Option	Type	Meaning
@Thermodynamics	PATH	Name of the file containing the thermodynamic data (CHEMKIN format)
@Transport	PATH	Name of the file containing the transport data (CHEMKIN format)
@Kinetics	PATH	Name of the file containing the kinetic mechanism (CHEMKIN format)
@Output	PATH	Name of the folder where to write the interpreted kinetic mechanism (XML format)

Table 7.6: Rapid-Kinetics dictionary

7.6.2 Example

```
Dictionary POLIMI_PRF_PAH_HT_1412
{
        @Kinetics          ../POLIMI_1412/Kinetics/POLIMI_PRF_PAH_HT_1412.CKI;
        @Thermodynamics    ../POLIMI_1412/Thermodynamics/POLIMI_TOT_NOX_1412.CKT;
        @Output            mykinetics;
}
```

7.7 Sub-Dictionary: XY-Profile

This dictionary is used to define a piece-wise linear profile.

7.7.1 Additional comments

@XVariable

Type of variable on the x axis: `length` | `time`.

@YVariable

Type of variable on the y axis: `temperature` | `pressure` | `volume`.

@XUnits

Units of measure of the x variable.

@YUnits

Units of measure of the y variable.

@Profile

XY Profile: pairs of x and y values (look at the example).

7.7.2 Example

```
Dictionary volume-profile
{
        @XVariable    time;
        @XUnits       s;
        @YVariable    volume;
        @YUnits       cm3;
        @Profile      0.0    1.00
                      0.1    1.15
                      0.2    1.35
                      0.4    1.60  ;
}
```

7.8 Sub-Dictionary: Parametric-Analysis

This dictionary is used to define the rules for parametric analysis.

7.8.1 Additional comments

@Type

Type of parameter to be analyzed: residence-time | temperature | pressure

Option	Type	Meaning
@XVariable	STRING	Type of variable on the x axis: `length` | `time`
@YVariable	STRING	Type of variable on the y axis: `temperature` | `pressure` | `volume`
@XUnits	STRING	Units of measure of the x variable
@YUnits	STRING	Units of measure of the y variable
@Profile	DOUBLES + DOUBLES	XY Profile (e.g. 0 100. 1 110. 2 200.)

Table 7.7: XY-Profile dictionary

Option	Type	Meaning
@Type	STRING	Type of parameter to be analyzed: `residence-time` | `temperature` | `pressure`
@ListOfValues	STRINGS	List of values (together with units of measure, if any)
@MinimumValue	MEASURE	Minimum value of parameter
@MaximumValue	MEASURE	Maximum value of parameter
@NumberOfPoints	INT	Number of points
@NumberOfThreads	INT	Number of threads

Table 7.8: Parametric-Analysis dictionary

@ListOfValues

List of values (together with units of measure, if any). Though this option the user defines explicitly the list of parameter values to be simulated. The alternative is to specify the interval through the following key-words: `@MinimumValue`, `@MaximumValue`, and `@NumberOfPoints`

@MinimumValue

Minimum value of parameter. It must be used in conjuction with `@MaximumValue` and `@NumberOfPoints` to specify the interval and the number of points for parametric analysis.

@MaximumValue

Maximum value of parameter. It must be used in conjuction with `@MinimumValue` and `@NumberOfPoints` to specify the interval and the number of points for parametric analysis.

@NumberOfPoints

Number of points. It must be used in conjuction with `@MinimumValue` and `@MaximumValue` to specify the interval and the number of points for parametric analysis.

@NumberOfThreads

Number of threads. If a multicore architecture is available, the user can distribute the parameteric analysis no more than one core, in order to speed-up the calculation.

7.8.2 Examples

```
Dictionary parametric-analysis
{
        @Type               temperature;
    @ListOfValues       1000 1100 1200 1300 1400 K;
    @NumberOfThreads    2;
}
```

```
Dictionary parametric-analysis
{
        @Type                temperature;
        @NumberOfPoints      10;
    @MinimumValue        1000 K;
    @MaximumValue        1200 K;
        @NumberOfThreads         5;
}
```

7.9 Sub-Dictionary: Adaptive-Grid1D

This dictionary is used for setting the options for the adaptive grid for 1D problems (see Table 7.9).

7.9.1 Additional comments

@Type

Type of grid: centered | database

@Lenght

Length of computational domain

Option	Type	Meaning
@Type	STRING	Type of grid: centered | database
@Length	MEASURE	Length of computational domain
@InitialPoints	INT	Initial number of points (must be between 7 and 15)
@Center	MEASURE	Center position (default: middle point)
@Width	MEASURE	Estimated (first guess) width of flame (default: 0.2*length)
@FixedPoint	MEASURE	Coordinate of fixed point (default: center of the grid)
@MaxPoints	INT	Maximum number of allowed points (default 300)
@MaxAdaptivePoints	INT	Maximum number of points that can be added per grid adaptation (default 10)
@GradientCoefficient	DOUBLE	Controls the maximum gradient allowed between two points (default 0.1)
@CurvatureCoefficient	DOUBLE	Controls the maximum curvature allowed between two points (default 0.5)
@Threshold	DOUBLE	Controls the minimum amount of each species to be considered active in the adaptive process (default: 1e-7)
@RegridPoints	INT	Points used for regrid operation (default: 20)

Table 7.9: Adaptive-Grid1D dictionary

@InitialPoints

Initial number of points (must be between 7 and 15)

@Center

Center position (default: middle point)

@Width

Estimated (first guess) width of flame (default: 0.2*length)

@FixedPoint

Coordinate of fixed point (default: center of the grid)

@MaxPoints

Maximum number of allowed points (default 300)

@MaxAdaptivePoints

Maximum number of points that can be added per grid adaptation (default 10)

@GradientCoefficient

Controls the maximum gradient allowed between two points (default 0.1)

@CurvatureCoefficient

Controls the maximum curvature allowed between two points (default 0.5)

@Threshold

Controls the minimum amount of each species to be considered active in the adaptive process (default: 1e-7)

@RegridPoints

Points used for regrid operation (default: 20)

7.9.2 Example

```
Dictionary grid
{
        @Length                 3 cm;
        @InitialPoints          12;
        @Type                   database;
        @MaxPoints              1000;
        @MaxAdaptivePoints      15;
        @GradientCoefficient    0.05;
        @CurvatureCoefficient   0.25;
        @Threshold              1e-5;
        @RegridPoints           20;
}
```

7.10 Sub-Dictionary: ODE-TridiagonalBlock-Parameters

This dictionary is used for setting the options for the stiff tridiagonal-block ODE solvers available in `OpenSMOKE++ Suite` (see Table 7.10).

Option	Type	Meaning
@OdeSolver	STRING	DAE Solver: OpenSMOKE++ | BzzOde | CVODE | DASPK (default: OpenSMOKE++)
@Jacobian	STRING	Jacobian sparsity pattern treatment: `TridiagonalBlock` | `Band` | `Sparse` (default: `TridiagonalBlock`)
@SparseSolver	STRING	Sparse solver: `EigenSparseLU` | `EigenBiCGSTAB` | `EigenGMRES` | `EigenDGMRES` | `Pardiso` | `SuperLUSerial` | `UMFPack` (default: `EigenSparseLU`)
@Preconditioner	STRING	Preconditioner to be used by iterative solvers: `ILUT` | `diagonal` (default: `ILUT`)
@RelativeTolerance	DOUBLE	Relative tolerance (default 1.e-6)
@AbsoluteTolerance	DOUBLE	Absolute tolerance (default 1.e-10)
@MaximumNumberOfSteps	INT	Maximum number of steps (default 500000)
@MaximumStep	DOUBLE	Maximum step (default: automatically chosen by the solver)
@MinimumStep	DOUBLE	Minimum step (default: automatically chosen by the solver)
@InitialStep	DOUBLE	Initial step (default: automatically chosen by the solver)
@MaximumOrder	INT	Maximum order of the method (default 5)
@MeanResidualThreshold	DOUBLE	Stop the integration when the mean residual is below this threshold (default: 0.)
@VerbosityLevel	INT	Verbosity level: 0=no output, 3 maximum level of output
@MinimumConstraints	BOOL	Constraints on minimum values (available only for OpenSMOKE++ and BzzOde solvers, default true)
@MaximumConstraints	BOOL	Constraints on maximum values (available only for OpenSMOKE++ and BzzOde solvers, default false)
@NonNegativeVariables	BOOL	Constraints on minimum values (available only for CVODE and DASPK solvers, default false)

Table 7.10: ODE-TridiagonalBlock-Parameters dictionary

7.10.1 Additional comments

@OdeSolver

The user can choose among four different DAE solvers: `OpenSMOKE++`, `BzzOde` (from the BzzMath library), `CVODE` (from the Sundials Suite) or `DASPK`

@Jacobian

The user to treat the Jacobian as a tridiagonal-block matrix (`TridiagonalBlock`), band matrix (`Band`) or as a sparse matrix (`Sparse`). The latter option is particular useful in case of limitations of available RAM.

@SparseSolver

Sparse solver to be used in case the `Sparse` option was chosen in `@Jacobian` option.

@Preconditioner

Preconditioner to be used in case the `Sparse` option was chosen in `@Jacobian` option.

@RelativeTolerance

Relative tolerance (default 1.e-6).

@AbsoluteTolerance

Absolute tolerance (default 1.e-10). For `IDA` solver we suggest to use stricter absolute tolerance (i.e. at least 1e-12).

@MaximumNumberOfSteps

Maximum number of steps (default 500000)

@MaximumStep

Maximum step (default: automatically chosen by the solver)

@MinimumStep

Minimum step (default: automatically chosen by the solver)

@InitialStep

Initial step (default: automatically chosen by the solver)

@MaximumOrder

Maximum order of the method (default 5)

@MeanResidualThreshold

Stop the integration when the mean residual is below this threshold (default: 0.)

@VerbosityLevel

Verbosity level: 0=no output, 3 maximum level of output

@MinimumConstraints

Constraints on minimum values (available only for OpenSMOKE++ and BzzOde solvers, default true)

@MaximumConstraints

Constraints on maximum values (available only for OpenSMOKE++ and BzzOde solvers, default false)

@NonNegativeVariables

Constraints on minimum values (available only for CVODE and DASPK solvers, default false)

7.10.2 Example

```
Dictionary dae-parameters
{
        @OdeSolver              OpenSMOKE++;
        @Jacobian                   TridiagonalBlock;
        @RelativeTolerance      1e-6;
        @AbsoluteTolerance      1e-10;
        @MaximumNumberOfSteps   100000;
        @MaximumStep            1e3;
        @MinimumStep            1e-9;
        @InitialStep            1e-9;
        @MaximumOrder           5;
        @MeanResidualThreshold  1e-5;
        @VerbosityLevel         0;
}
```

7.11 Sub-Dictionary: DAE-TridiagonalBlock-Parameters

This dictionary is used for setting the options for the stiff tridiagonal-block DAE solvers available in `OpenSMOKE++ Suite` (see Table 7.11).

7.11.1 Additional comments

@DaeSolver

The user can choose among four different DAE solvers: `OpenSMOKE++`, `BzzDae` (from the BzzMath library), `IDA` (from the Sundials Suite) or `DASPK`

@Jacobian

The user to treat the Jacobian as a tridiagonal-block matrix (`TridiagonalBlock`) or as a sparse matrix (`Sparse`). The latter option is particular useful in case of limitations of available RAM.

@SparseSolver

Sparse solver to be used in case the `Sparse` option was chosen in `@Jacobian` option.

@Preconditioner

Preconditioner to be used in case the `Sparse` option was chosen in `@Jacobian` option.

@RelativeTolerance

Relative tolerance (default 1.e-6).

@AbsoluteTolerance

Absolute tolerance (default 1.e-10). For `IDA` solver we suggest to use stricter absolute tolerance (i.e. at least 1e-12).

Option	Type	Meaning
@DaeSolver	STRING	DAE Solver: OpenSMOKE++ | BzzDae | IDA | DASPK (default: OpenSMOKE++)
@Jacobian	STRING	Jacobian sparsity pattern treatment: `TridiagonalBlock` | `Sparse` (default: `TridiagonalBlock`)
@SparseSolver	STRING	Sparse solver: `EigenSparseLU` | `EigenBiCGSTAB` | `EigenGMRES` | `EigenDGMRES` | `Pardiso` | `SuperLUSerial` | `UMFPack` (default: `EigenSparseLU`)
@Preconditioner	STRING	Preconditioner to be used by iterative solvers: `ILUT` | `diagonal` (default: `ILUT`)
@RelativeTolerance	DOUBLE	Relative tolerance (default 1.e-6)
@AbsoluteTolerance	DOUBLE	Absolute tolerance (default 1.e-10)
@MaximumNumberOfSteps	INT	Maximum number of steps (default 500000)
@MaximumStep	DOUBLE	Maximum step (default: automatically chosen by the solver)
@MinimumStep	DOUBLE	Minimum step (default: automatically chosen by the solver)
@InitialStep	DOUBLE	Initial step (default: automatically chosen by the solver)
@MaximumOrder	INT	Maximum order of the method (default 5)
@MeanResidualThreshold	DOUBLE	Stop the integration when the mean residual is below this threshold (default: 0.)
@VerbosityLevel	INT	Verbosity level: 0=no output, 3 maximum level of output

Table 7.11: DAE-TridiagonalBlock-Parameters dictionary

@MaximumNumberOfSteps

Maximum number of steps (default 500000)

@MaximumStep

Maximum step (default: automatically chosen by the solver)

@MinimumStep

Minimum step (default: automatically chosen by the solver)

@InitialStep

Initial step (default: automatically chosen by the solver)

@MaximumOrder

Maximum order of the method (default 5)

@MeanResidualThreshold

Stop the integration when the mean residual is below this threshold (default: 0.)

@VerbosityLevel

Verbosity level: 0=no output, 3 maximum level of output

7.11.2 Example

```
Dictionary dae-parameters
{
        @DaeSolver              OpenSMOKE++;
        @Jacobian                   TridiagonalBlock;
        @RelativeTolerance      1e-6;
        @AbsoluteTolerance      1e-10;
        @MaximumNumberOfSteps   100000;
        @MaximumStep            1e3;
        @MinimumStep            1e-9;
        @InitialStep            1e-9;
        @MaximumOrder           5;
        @MeanResidualThreshold  1e-5;
        @VerbosityLevel         0;
}
```

7.12 Sub-Dictionary: NLS-TridiagonalBlock-Parameters

This dictionary is used for setting the options for the tridiagonal-block non-linear (NL) solvers available in `OpenSMOKE++ Suite` (see Table 7.12).

7.12.1 Additional comments

@NlsSolver

The user can choose among three different NLS solvers: `OpenSMOKE++`, `BzzNls` (from the BzzMath library) or `KINSOL` (from the Sundials Suite)

Option	Type	Meaning
@NlsSolver	STRING	NLS Solver: `OpenSMOKE++` | `BzzNls` | `KINSOL` (default: `OpenSMOKE++`)
@Jacobian	STRING	Jacobian sparsity pattern treatment: `TridiagonalBlock` | `Sparse` (default: `TridiagonalBlock`)
@SparseSolver	STRING	Sparse solver: `EigenSparseLU` | `EigenBiCGSTAB` | `EigenGMRES` | `EigenDGMRES` | `Pardiso` | `SuperLUSerial` | `UMFPack` (default: `EigenSparseLU`)
@Preconditioner	STRING	Preconditioner to be used by iterative solvers: `ILUT` | `diagonal` (default: `ILUT`)
@FunctionTolerance	DOUBLE	Function tolerance (default 6.e-6)
@StepTolerance	DOUBLE	Step tolerance (default 3.6e-11)
@RelativeError	DOUBLE	Relative error (default 1.5e-8)
@MaximumNumberOfIterations	INT	Maximum number of iterations (default 200)
@MaximumSetupCalls	INT	Maximum number of setup calls (default 10)
@MaximumSubSetupCalls	INT	Maximum number of setup sub-calls (default 5)
@Scaling	INT	Scaling strategy for evaluation of residuals (default 1)
@Strategy	INT	Numerical methodology: NewtonBasic | NewtonGlobalization (default)
@VerbosityLevel	INT	Verbosity level: 0=no output, 3 maximum level of output

Table 7.12: NLS-TridiagonalBlock-Parameters dictionary

@Jacobian

The user to treat the Jacobian as a tridiagonal-block matrix (`TridiagonalBlock`) or as a sparse matrix (`Sparse`). The latter option is particular useful in case of limitations of available RAM.

@SparseSolver

Sparse solver to be used in case the `Sparse` option was chosen in `@Jacobian` option.

@Preconditioner

Preconditioner to be used in case the `Sparse` option was chosen in `@Jacobian` option.

@FunctionTolerance

Function tolerance (default 6.e-6)

@StepTolerance

Step tolerance (default 3.6e-11)

@RelativeError

Relative error (default 1.5e-8)

@MaximumNumberOfIterations

Maximum number of iterations (default 200)

@MaximumSetupCalls

Maximum number of setup calls (default 10)

@MinimumSubSetupCalls

Maximum number of setup sub-calls (default 5)

@Scaling

Scaling strategy for evaluation of residuals (default 1)

@Strategy

Numerical methodology: NewtonBasic | NewtonGlobalization (default)

@VerbosityLevel

Verbosity level: 0=no output, 3 maximum level of output

7.12.2 Example

```
Dictionary nls-parameters
{
        @NlsSolver                  OpenSMOKE++;
        @Jacobian                       TridiagonalBlock;
        @FunctionTolerance          1e-7;
        @StepTolerance              3e-11;
        @RelativeError              1.5e-8;
        @MaximumNumberOfIterations  200;
        @MaximumSetupCalls          10;
```

```
10          @MaximumSubSetupCalls       5;
11          @Scaling                    1;
12          @VerbosityLevel             0;
13          @Strategy                   NewtonGlobalization;
14  }
```

7.13 Sub-Dictionary: FalseTransient-Parameters

This dictionary is used for setting the options governing the false-transient methodology (see Table 7.13).

7.13.1 Additional comments

@FalseTransientSolver

The user can choose among three different NLS solvers: `OpenSMOKE++`, `BzzNls` (from the BzzMath library) or `KINSOL` (from the Sundials Suite)

@Jacobian

The user to treat the Jacobian as a tridiagonal-block matrix (`TridiagonalBlock`) or as a sparse matrix (`Sparse`). The latter option is particular useful in case of limitations of available RAM.

@SparseSolver

Sparse solver to be used in case the `Sparse` option was chosen in `@Jacobian` option.

@Preconditioner

Preconditioner to be used in case the `Sparse` option was chosen in `@Jacobian` option.

@FunctionTolerance

Function tolerance (default 6.e-6)

@StepTolerance

Step tolerance (default 3.6e-11)

@RelativeError

Relative error (default 1.5e-8)

@MaximumNumberOfIterations

Maximum number of iterations (default 200)

@MaximumSetupCalls

Maximum number of setup calls (default 10)

@MinimumSubSetupCalls

Maximum number of setup sub-calls (default 5)

@Scaling

Scaling strategy for evaluation of residuals (default 1)

Option	Type	Meaning
@FalseTransientSolver	STRING	NLS Solver: `OpenSMOKE++` | `BzzNls` | `KINSOL` (default: `OpenSMOKE++`)
@Jacobian	STRING	Jacobian sparsity pattern treatment: `TridiagonalBlock` | `Sparse` (default: `TridiagonalBlock`)
@SparseSolver	STRING	Sparse solver: `EigenSparseLU` | `EigenBiCGSTAB` | `EigenGMRES` | `EigenDGMRES` | `Pardiso` | `SuperLUSerial` | `UMFPack` (default: `EigenSparseLU`)
@Preconditioner	STRING	Preconditioner to be used by iterative solvers: `ILUT` | `diagonal` (default: `ILUT`)
@FunctionTolerance	DOUBLE	Function tolerance (default 6.e-6)
@StepTolerance	DOUBLE	Step tolerance (default 3.6e-11)
@RelativeError	DOUBLE	Relative error (default 1.5e-8)
@MaximumNumberOfIterations	INT	Maximum number of iterations (default 200)
@MaximumSetupCalls	INT	Maximum number of setup calls (default 10)
@MaximumSubSetupCalls	INT	Maximum number of setup sub-calls (default 5)
@Scaling	INT	Scaling strategy for evaluation of residuals (default 1)
@Strategy	INT	Numerical methodology: `NewtonBasic` | `NewtonGlobalization` (default: `NewtonGlobalization`)
@VerbosityLevel	INT	Verbosity level: 0=no output, 3 maximum level of output
@InitialStep	DOUBLE	Initial step (default: 1e-6)
@IncrementFactor	DOUBLE	Increment factor (default: 10.)
@MaximumStep	DOUBLE	Maximum step (default: 0.1)
@DecrementFactor	DOUBLE	Decrement factor (default: 2.)
@MinimumStep	DOUBLE	Minimum step (default: 1e-9)
@MinimumNumberSteps	INT	Minimum number of steps (default: 100)
@StepsBeforeIncreasing	INT	Steps to be performed before increasing the step (default: 20)
@StepsReusingJacobian	INT	Steps for which the Jacobian is kept fixed (default: 20)

Table 7.13: FalseTransient-Parameters dictionary

@Strategy

Numerical methodology: `NewtonBasic | NewtonGlobalization` (default: `NewtonGlobalization`)

@VerbosityLevel

Verbosity level: 0=no output, 3 maximum level of output

@InitialStep

Initial step (default: 1e-6)

@IncrementFactor

Increment factor (default: 10.)

@MaximumStep

Maximum step (default: 0.1)

@DecrementFactor

Decrement factor (default: 2.)

@MinimumStep

Minimum step (default: 1e-9)

@MinimumNumberSteps

Minimum number of steps (default: 100)

@StepsBeforeIncreasing

Steps to be performed before increasing the step (default: 20)

@StepsReusingJacobian

Steps for which the Jacobian is kept fixed (default: 20)

7.13.2 Example

```
Dictionary falsetransient-parameters
{
        @FalseTransientSolver         KinSol;
        @Jacobian                  TridiagonalBlock;
        @FunctionTolerance            1e-8;
        @StepTolerance                3e-11;
        @RelativeError                1.5e-8;
        @MaximumNumberOfIterations    200;
        @MaximumSetupCalls            10;
        @MaximumSubSetupCalls         5;
        @Scaling                      1;
        @VerbosityLevel               0;
        @Strategy                     NewtonGlobalization;
        @StepsReusingJacobian         20;
        @StepsBeforeIncreasing        25;
        @MinimumNumberSteps           80;
        @InitialStep                  1e-6;
        @IncrementFactor              10.;
```

```
19          @MaximumStep                0.01;
20          @DecrementFactor            2.;
21          @MinimumStep                1e-9;
22  }
```

7.14 Sub-Dictionary: Fiber

This dictionary can be used to describe fibers adopted for suspending isolated fuel droplets in microgravity atmospheres (see Table 7.14)

7.15 Sub-Dictionary: PolimiSoot

This dictionary can be used to manage/post-process the soot mechanism based on the discrete sectional method developed at Politecnico di Milano by the CRECK Modeling Group (see Table 7.15)

Option	Type	Meaning
@Material	STRINGS	Name of material: `quartz | SiC`
@Density	MEASURE	Fiber density
@ThermalConductivity	MEASURE	Fiber thermal conductivity
@SpecificHeat	MEASURE	Fiber constant pressure specific heat
@Emissivity	DOUBLE	Fiber emissivity (default: 0.)
@Diameter	MEASURE	Fiber diameter
@NusseltLaw	STRING	Law to be used for estimating the Nusselt's number: `stagnant | non-stagnant`
@Arrangement	STRING	Arrangement of fiber(s): `rod | single | cross`

Table 7.14: Fiber dictionary

Option	Type	Meaning
@SootLabel	STRING	Label indicating the soot particles (default: `BIN`)
@FractalDimension	DOUBLE	Fractal dimension of soot particles (default: 1.8)
@SootMinimumSection	INT	Index of minimum discrete section which is considered soot (default: 5)
@SootMinimumFractalDimension	INT	Index of minimum discrete section from which to apply the fractal dimension (default: 12)
@Density	DOUBLES	Linear density: BINstart DENSITYstart BINend DENSITYend (default: 10 1500. 20 1700.)
@PhysicalDiffusion	BOOL	Physical diffusion for soot particles (default: `true`)
@PhysicalDiffusionReductionCoefficient	DOUBLE	Physical diffusion reduction coefficient for soot particles(default: 1)
@ThermophoreticEffect	BOOL	Thermophoretic effect for soot sections
@ThermophoreticEffectInCorrectionVelocity	BOOL	Thermophoretic effect for soot sections in calculation of correction velocity (default: false)
@ThermophoreticEffectInEnthalpyFluxes	STRING	Thermophoretic effect for soot sections in calculation of enthalpy fluxes: `DoNotExclude | Exclude | ExcludeOnlyThermophoreticEffect` (default: `DoNotExclude`)
@ThermophoreticEffectSmoothingTime	MEASURE	Smoothing time for thermophoretic effect (default: 0. s)
@PlanckCoefficient	STRING	Calculation of Planck Coefficient: `Smooke| Kent | Sazhin` (default: `Smooke`)
@RadiativeHeatTransfer	BOOL	Radiative heat transfer from soot

Table 7.15: PolimiSoot dictionary

Chapter 8

Tutorials

8.1 Autoignition of a mixture of hydrogen and air

This tutorial presents a transient simulation of the spontaneous ignition of a stoichiometric hydrogen/air mixture at constant pressure. This tutorial uses the `POLIMI_H2CO_1412` kinetic mechanism developed by CRECK Modeling Group at Politecnico di Milano. The `POLIMI_H2CO_1412` mechanism is distributed with the current version of `OpenSMOKE++ Suite` or can be downloaded directly from the web site (`http://creckmodeling.chem.polimi.it/`). Here we are interested in determining the ignition time under a specified set of initial pressure and temperature conditions, assuming no heat loss to the environment (adiabatic conditions). In addition, we would like to determine which reactions contribute most to the autoignition process, using sensitivity analysis. The system is a closed or batch process, which means there is no flow of mass into or out of the reactor.

8.1.1 Setup

This tutorial project is located in the `examples\autoignition-h2-air` directory. A single, batch reactor at constant pressure will be simulated. The input data are provided though the `input.dic` file (see Figure 8.1). In paticular, the end time of the simulation is set to 0.25 ms. The initial temperature is equal to 1000 K and the pressure is kept equal to 1 atm (constant pressure). The volume is not important for the results of this simulation, so the default value of 1 cm3 will be used for the initial volume. Since this is a closed homogeneous system, the results in terms of species fraction and temperature will be the same, regardless of the volume value. If surface chemistry were included, the volume-to-surface ratio would be important, but in this case it is gas only.

The starting gas mixture is given in relative moles as 2.0 H2, 1.0 O2, and 3.76 N2. Writing it this way makes it easy to see that the O2/N2 ratio matches the composition of air, and that the fuel/air ratio is stoichiometric (two H2 per one O2). The normalization will occur automatically. The sensitivity analysis is enabled and this results in sensitivity coefficients being calculated for all species and all reactions and saved in the proper XML files. As already reported in the previous sections, this option should be used with care, as the computation time and solution-file sizes increase as a higher power of the size of the reaction mechanism. Except in the case of a very small reaction mechanism, such as the one being used in the sample problem, it is better to use the `@Species` option to request that sensitivity coefficients be output only for a few species of highest interest.

8.1.2 Results

Output results are located in the `Output` folder. The `OpenSMOKE++ PostProcessor` can be conveniently used to post process the results. Alternatively, the `Output.out` file can be post-processed using different tools, such as Gnuplot, Matlab, etc. Plots reported in the following were obtained using the `OpenSMOKE++ PostProcessor`.

Figure 8.2 (left) shows the temperature profile as a function of time for this problem. At the end of this simulation, the temperature is still rising; if it is run much longer, the temperature increases another ~300 K, nearing the adiabatic flame temperature. Although not shown, the volume in this constant-pressure system shows a corresponding increase at ignition. The calculated ignition time, defined as the time at which the gas reaches a temperature of 1400 K (i.e. corresponding to an increase of 400 K), is equal to 0.186 ms. Figure 8.2 (right) shows a close-up of species mole fractions as a function of time. Note that zooming in on the x-axis

```
Dictionary BatchReactor
{
        @KineticsPreProcessor  POLIMI_H2CO_1412;
        @Type                  NonIsothermal-ConstantPressure;
        @InitialStatus         initial-mixture;
        @EndTime               0.25 ms;
        @SensitivityAnalysis   sensitivity-options;
}

Dictionary POLIMI_H2CO_1412
{
        @Kinetics         ../../kinetic-mechanisms/POLIMI_1412/Kinetics/POLIMI_H2CO_1412.CKI;
        @Thermodynamics   ../../kinetic-mechanisms/POLIMI_1412/Thermodynamics/POLIMI_TOT_NOX_1412.CKT;
        @Output           kinetics;
}

Dictionary initial-mixture
{
        @Temperature    1000.   K ;
        @Pressure       101325. Pa ;
        @Moles             H2 2. O2 1. N2 3.76;
}

Dictionary sensitivity-options
{
        @Type               arrhenius-parameters;
        @DenseSolver        Eigen;
        @DenseFullPivoting  false;
        @SubSteps           2;
        @Species            H2 O2 H2O;
}
```

Figure 8.1: Input file for adiabatic, constant pressure batch reactor simulation

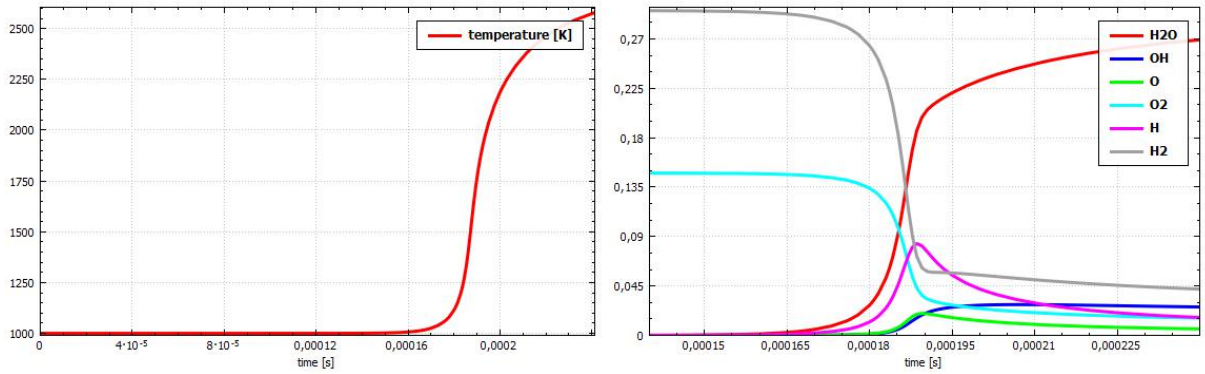

Figure 8.2: Temperature (left) and mole fraction profiles of main species (right) for the adiabatic, constant-pressure batch reactor

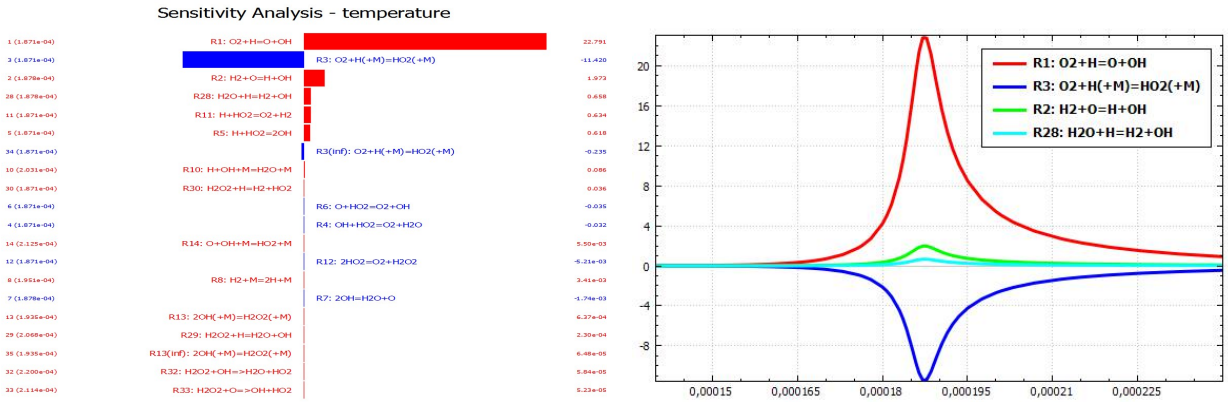

Figure 8.3: Sensitivity analysis for the adiabatic, constant-pressure batch reactor

shows the expected increase in radical species and the products at ignition, along with a decrease in hydrogen and oxygen reactants.

Figure 8.3 shows normalized sensitivity coefficients as a function of time for the four reactions that have the largest effect on the gas temperature. As one might expect, the largest sensitivity occurs near the time of ignition, when the most rapid change in temperature is taking place. The results also show that the dominant reaction for determining the temperature during ignition is the exothermic radical-recombination reaction #1: $O2 + H = O + OH$. The sensitivity coefficient for this reaction is positive, indicating that increasing the rate of this reaction will lead to a higher temperature (more heat production). In contrast, the sensitivity coefficient for reaction #3 is large and negative, indicating that increasing the rate of this reaction will lead to a lower temperature (less heat production).

8.1.3 Note

This tutorial was adapted from the Tutorials Manual (CK-TUT-15082-0809-UG-1) distributed with the CHEMKIN-PRO Software [**?**].

8.2 Ignition delay times for propane autoignition

This user tutorial presents an ignition-delay time calculation for the homogeneous, isobaric, adiabatic combustion of propane in air. There are various ways of defining the ignition time, experimentally as well as computationally, for combustion applications. For example, it is often defined as the time at which either the maximum or onset of certain species concentrations is reached, the time at which a specified rate of increase of temperature occurs, the time at which luminous radiant output from the system is first observed, etc. The

reported experimental data can vary greatly, depending on which definition was used in the experiments. Thus, it is often useful to select which ignition-delay time definition should be used in numerical computations. In this particular example, the ignition times are computed using two distinct definitions:

- as the time during which the maximum amount of heat is released (as indicated by the inflection point in the temperature profile)
- as the time corresponding to the maximum of a certain species (OH in this case) concentration

Discrepancies between results are presented.

8.2.1 Setup

This tutorial project is located in the `examples\autoignition_propane` directory. A single, batch reactor at constant pressure will be simulated. The input data are provided though the `input.dic` file (see Figure 8.4). The initial temperature is equal to 1200 K, while the pressure is kept constant and equal to 1 atm. The volume is not important for the results of this simulation, so the default value of 1 cm3 will be used for the initial volume. Since this is an isobaric closed homogeneous system, the results in terms of species fraction and temperature will be the same, regardless of the volume value. If surface chemistry were included, the volume-tosurface ratio would be important, but in this case only gas-phase chemistry is present. The starting gas mixture is given as 0.02 C3H8, 0.05 O2, and 0.93 Ar. These rather dilute conditions are representative of shock-tube experimental conditions. The end time is specified on through the `@EndTime` option. It is important to check the resulting output file after the run is complete to make sure that the `@EndTime` is large enough to allow for ignition to occur. If no ignition time is provided at the end of the output file, ignition has not yet occurred and the `@EndTime` of the simulation should be adjusted.

8.2.2 Results

Output results are located in the `Output` folder. The `OpenSMOKE++ PostProcessor` can be conveniently used to post process the results. Alternatively, the `Output.out` file can be post-processed using different tools, such as Gnuplot, Matlab, etc. Plots reported in the following were obtained using the `OpenSMOKE++ PostProcessor`.

Figure 8.5 shows the temperature (left) and OH mole fraction (right) profiles as a function of time. The calculated ignition times, according to the defintiions reported above, are very similar: in particular, the temperaturre inflection point is located at 13.29 ms, while the peak in the OH mole fraction profile is located at 13.34 ms.

However, this result, is not necessarily true for every operating condition. Some discrepancies can be observed between the two proposed criteria. As an example, we can repeat our calculations using a higher initial temperature, equal to 2600 K. The new results are reported in Figure 8.6. In this case, as evident from the Figure, the calculated ignition times are very different: in particular, the temperaturre inflection point is located at 0.8 μs, while the peak in the OH mole fraction profile is located at 4.1 μs.

8.2.3 Note

This tutorial was adapted from the Tutorials Manual (CK-TUT-15082-0809-UG-1) distributed with the CHEMKIN-PRO Software [**?**].

8.3 Simulating a Shock-Tube experiment

Shock tube experiments are often used to obtain chemical kinetic data at high temperatures, which is especially relevant to combustion modeling. In particular, they are commonly used to study reaction paths and to measure reaction rates at elevated temperatures. We can apply the `OpenSMOKE++` Shock Tube solver to validate the reaction mechanism or kinetic parameters derived from such experiments. In this tutorial, we want to reproduce one of the shock tube experiments done by Camac and Feinberg [**?**]. Camac and Feinberg measured the production rates of nitric oxide (NO) in shock-heated air over the temperature range of 2300 K to 6000 K. They also assembled a reaction mechanism with kinetic parameters derived from their experimental results. The reaction $N2+M = N+N+M$ in their mechanism has a different temperature dependency when the third body is a nitrogen atom (N). To properly incorporate different temperature dependencies for different third

```
Dictionary BatchReactor
{
        @KineticsPreProcessor  POLIMI_C1C3HT_1412;
        @Type                  NonIsothermal-ConstantPressure;
        @InitialStatus         initial-mixture;
        @EndTime               0.30 s;
}

Dictionary POLIMI_C1C3HT_1412
{
        @Kinetics         ../../kinetic-mechanisms/POLIMI_1412/Kinetics/POLIMI_C1C3HT_1412.CKI;
        @Thermodynamics   ../../kinetic-mechanisms/POLIMI_1412/Thermodynamics/POLIMI_TOT_NOX_1412.CKT;
        @Output           kinetics;
}

Dictionary initial-mixture
{
        @Temperature    1200.   K ;
        @Pressure       101325. Pa ;
        @Moles             C3H8 0.02  O2 0.05  AR 0.93;
}
```

Figure 8.4: Input file for adiabatic, constant pressure batch reactor simulation fed with propane and air

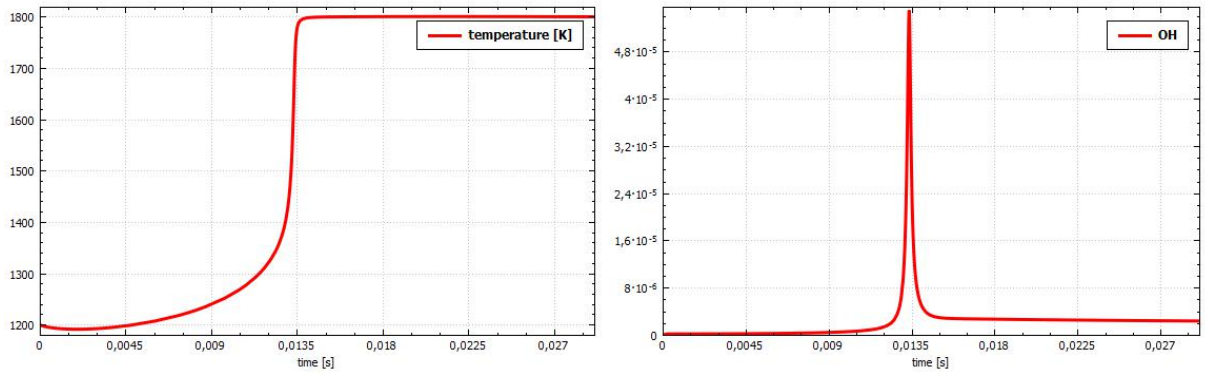

Figure 8.5: Temperature (left) and OH mole fraction (right) profiles for the adiabatic, constant-pressure batch reactor fed with propane and air (initial temperature equal to 1200 K)

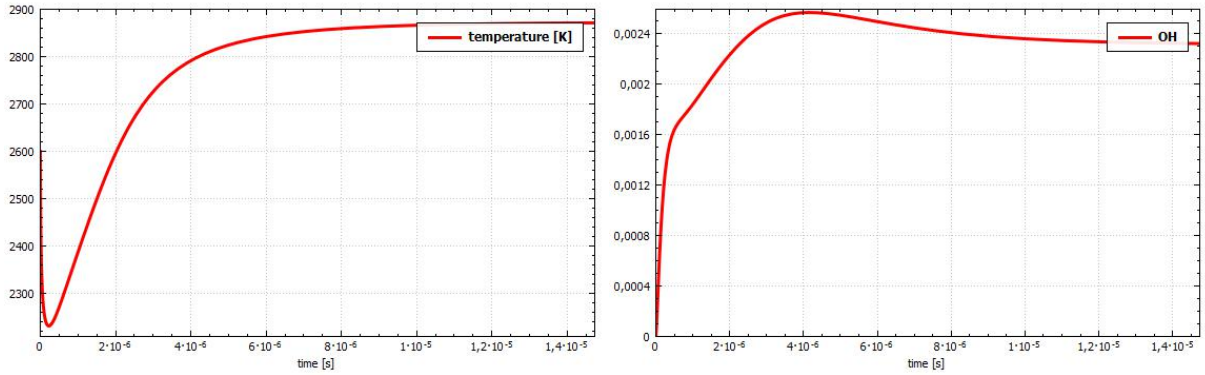

Figure 8.6: Temperature (left) and OH mole fraction (right) profiles for the adiabatic, constant-pressure batch reactor fed with propane and air (initial temperature equal to 2600 K)

bodies, we exclude N from participating as a third body in the original reaction, i.e., the effective third body efficiency for N is set to zero. And we add a new reaction $N2 + N = N + N$ to explicitly address the different temperature dependence of nitrogen atom as the third body.

8.3.1 Setup

Setting up a shock tube model requires information from the corresponding experiment. In addition to the conditions of the initial (unshocked) gas mixture, we will need to provide information on the diameter of the shock tube, the viscosity of the gas at 300 K, and the velocity of the incident shock. If we do not know the shock velocity from the experiment, we can estimate it by using the Chapmen-Jouguet detonation model. The shock tube diameter and the gas viscosity at 300 K are only required when the boundary-layer correction is used in the shock simulation.

This tutorial project is located in the `projects\shock_tube_experiment` directory. The input data are provided though the `input.dic` file (see Figure 8.7). In this example, we are considering an incident shock problem with boundary layer corrections. The unshocked temperature is 296K, the pressure is 6.58E-3 atm, and the shock velocity is 2.8E5 cm/sec. The initial mole fractions are 0.78118 N2, 0.20948 O2, and 0.00934 Ar. The problem will terminate at 200 microseconds. The shock tube diameter is 3.81 cm, and the viscosity of the gas at 300K is 1.777E-4 gm/cm/s; these parameters are used for the boundary layer corrections.

8.3.2 Results

The NO mole fraction behind the incident shock is shown in Figure 8.8 as a function of time. The NO mole fraction profile rapidly rises to a peak value then gradually falls back to its equilibrium level. Reasons for

```
Dictionary ShockTubeReactor
{
        @KineticsPreProcessor    Camac-Feinberg;
        @Type                    IncidentShock;
        @IncidentShockVelocity   2800 m/s;
        @BeforeShockStatus       initial-mixture;
        @EndTime                 2 ms;
        @Options                 output-options;

        @BoundaryLayerCorrection true;
        @Diameter                3.81 cm;
        @Viscosity               1.777e-5 kg/m/s;
}

Dictionary Camac-Feinberg
{
        @Kinetics        ../../kinetic-mechanisms/Camac-Feinberg/kinetics.CKI;
        @Thermodynamics  ../../kinetic-mechanisms/Camac-Feinberg/thermo.CKT;
        @Output          kinetics;
}

Dictionary initial-mixture
{
        @Temperature     296. K;
        @Pressure        6.58E-3 atm;
        @MoleFractions   AR 0.00934  N2 0.78118  O2 0.20948;
}

Dictionary output-options
{
        @StepsFile       1;
}
```

Figure 8.7: Input file for simulation of shock tube experiment

the greater-than-equilibrium peak NO concentration can be found in the paper by Camac and Feinberg and references therein. The predicted peak NO mole fraction is 0.046 and is in good agreement with the measured and the computed data by Camac and Feinberg.

8.3.3 Note

This tutorial was adapted from the Tutorials Manual (CK-TUT-15082-0809-UG-1) distributed with the CHEMKIN-PRO Software [?].

8.4 Perfectly Stirred Reactor

This tutorial presents the simulation of the steady-state combustion of a mixture of hydrogen, nitrogen, and oxygen in an adiabatic perfectly-stirred reactor. This tutorial uses the `POLIMI_H2CO_1412` kinetic mechanism developed by CRECK Modeling Group at Politecnico di Milano. This project demonstrates the use of parameteric analysis features of `OpenSMOKE++ Suite` for solving multiple reactors at different operating conditions.

8.4.1 Setup

This tutorial project is located in the `tutorials\perfectly-stirred-reactor` directory. The input data are provided though the `input.dic` file (see Figure 8.9). The residence time of the gas in the PSR is assumed to be 0.03 ms, the volume equal to 67.4 cm3, and the pressure is kept constant. The inlet temperature is equal to 298 K and the initial gas temperature in the PSR is equal to 1800 K. No heat loss is taken into account, so the

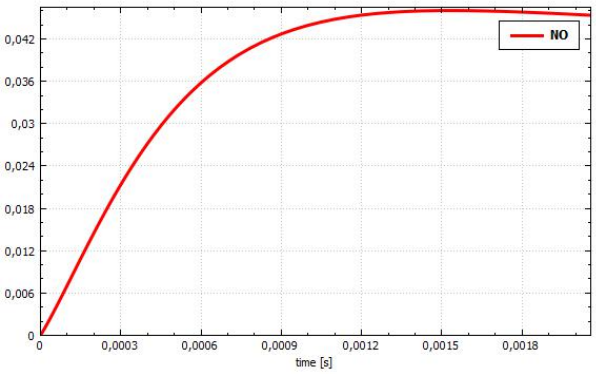

Figure 8.8: NO mole fraction profile from simulation of shock-tube experiment

system will be treated as adiabatic. No estimate of the gas composition is needed to help the solver converge on a solution. The fuel stream is composed of 60% H2 and 40% N2. The oxidizer is defined as 79% N2 and 21% O2 (air). An equivalence ratio of 1 is assumed.

A parametric analysis is performed, by studying the effects of pressure on the reactor's performances. In particular, the pressure was gradually increased from 1 atm to 10 atm.

8.4.2 Results

Output results are located in the `Output` folder. Since a parametric analysis was performed, the `Ouptut` folder contains several subfolders with names `Case0`, `Case1`, `Case2`, and so on, for each reactor which was simulated (10 in this specific example). In addition, the `ParametricAnalysis.out` file summarizes the output results for all the reactors simulated. Figure 8.10 shows the temperature (left) and H2O mole fraction (right) of outlet mixture as a function of the reactor pressure (according to the results reported in `ParametricAnalysis.out` file).

```
Dictionary POLIMI_H2CO_1412
{
 @Kinetics        ../../kinetic-mechanisms/POLIMI_1412/Kinetics/POLIMI_H2CO_1412.CKI;
 @Thermodynamics  ../../kinetic-mechanisms/POLIMI_1412/Thermodynamics/POLIMI_TOT_NOX_1412.CKT;
 @Output          kinetics;
}

Dictionary PerfectlyStirredReactor
{
        @KineticsPreProcessor  POLIMI_H2CO_1412;
        @Type                  NonIsothermal-ConstantPressure;
        @InletStatus           inlet-mixture;
        @InitialStatus         initial-mixture;
        @ResidenceTime         0.03 ms;
        @Volume                67.4 cm3;
        @Options               output-options;
        @ParametricAnalysis    parametric-analysis;
}

Dictionary inlet-mixture
{
        @Temperature       298. K ;
        @Pressure          1. atm ;
        @EquivalenceRatio  1.;
        @FuelMoles         H2 60 N2 40;
        @OxidizerMoles     O2 21 N2 79;
}

Dictionary parametric-analysis
{
        @Type               pressure;
        @ListOfValues       1. 2. 3. 4. 5. 6. 7. 8. 9. 10. atm;
}

Dictionary output-options
{
        @StepsFile         1000;
        @StepsVideo        5000;
        @VerboseASCIIFile  true;
        @VerboseVideo      false;
}
```

Figure 8.9: Input file for simulation of perfectly stirred reactors through parametric analysis

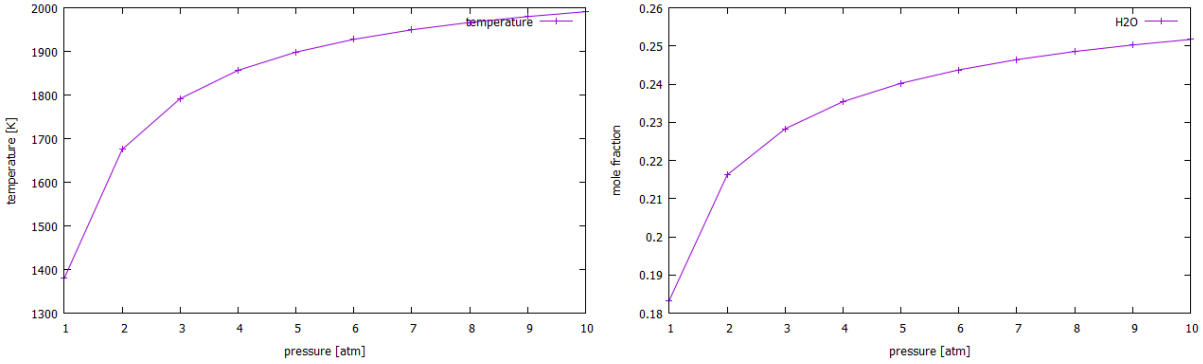

Figure 8.10: Temperature (left) and H2O mole fraction (right) of outlet mixture as a function of reactor pressure

Chapter 9

Publications using OpenSMOKE++

These are most recent publications in which the OpenSMOKE++ framework was adopted. Please send us information about your publications using OpenSMOKE++ (alberto.cuoci@polimi.it).

- **Jin, H., Cai, J., Wang, G., Wang, Y., Li, Y., Yang, J., Cheng, Z., Yuan, W., Qi, F.** A comprehensive experimental and kinetic modeling study of tert-butanol combustion (2016) Combustion and Flame, 169, pp. 154-170, DOI: 10.1016/j.combustflame.2016.03.026

- **Li, S., Yang, B., Qi, F.** Accelerate global sensitivity analysis using artificial neural network algorithm: Case studies for combustion kinetic model (2016) Combustion and Flame, 168, pp. 53-64, DOI: 10.1016/j.combustflame.2016.03.028

- **Bernardi, M.S., Pelucchi, M., Stagni, A., Sangalli, L.M., Cuoci, A., Frassoldati, A., Secchi, P., Faravelli, T.** Curve matching, a generalized framework for models/experiments comparison: An application to n-heptane combustion kinetic mechanisms (2016) Combustion and Flame, 168, pp. 186-203, DOI: 10.1016/j.combustflame.2016.03.019

- **Saggese, C., Cuoci, A., Frassoldati, A., Ferrario, S., Camacho, J., Wang, H., Faravelli, T.** Probe effects in soot sampling from a burner-stabilized stagnation flame (2016) Combustion and Flame, 167, pp. 184-197, DOI: 10.1016/j.combustflame.2016.02.013

- **Rebughini, S., Cuoci, A., Maestri, M.** Hierarchical analysis of the gas-to-particle heat and mass transfer in micro packed bed reactors (2016) Chemical Engineering Journal, 289, pp. 471-478, DOI: 10.1016/j.cej.2015.12.089

- **Nativel, D., Pelucchi, M., Frassoldati, A., Comandini, A., Cuoci, A., Ranzi, E., Chaumeix, N., Faravelli, T.** Laminar flame speeds of pentanol isomers: An experimental and modeling study (2016) Combustion and Flame, 166, pp. 1-18, DOI: 10.1016/j.combustflame.2015.11.012

- **Rebughini, S., Cuoci, A., Maestri, M.** Handling contact points in reactive CFD simulations of heterogeneous catalytic fixed bed reactors (2016) Chemical Engineering Science, 141, pp. 240-249, DOI: 10.1016/j.ces.2015.11.013

- **Rodriguez, A., Herbinet, O., Battin-Leclerc, F., Frassoldati, A., Faravelli, T., Ranzi, E.** Experimental and modeling investigation of the effect of the unsaturation degree on the gas-phase oxidation of fatty acid methyl esters found in biodiesel fuels (2016) Combustion and Flame, 164, pp. 346-362, DOI: 10.1016/j.combustflame.2015.11.032

- **Maffei, T., Gentile, G., Rebughini, S., Bracconi, M., Manelli, F., Lipp, S., Cuoci, A., Maestri, M.** A multiregion operator-splitting CFD approach for coupling microkinetic modeling with internal porous transport in heterogeneous catalytic reactors (2016) Chemical Engineering Journal, 283, pp. 1392-1404, DOI: 10.1016/j.cej.2015.08.080

- **Stagni, A., Frassoldati, A., Cuoci, A., Faravelli, T., Ranzi, E.** Skeletal mechanism reduction through species-targeted sensitivity analysis (2016) Combustion and Flame, 163, pp. 382-393, DOI: 10.1016/j.combustflame.2015.10.013

- **Saggese, C., Ferrario, S., Camacho, J., Cuoci, A., Frassoldati, A., Ranzi, E., Wang, H., Faravelli, T.** Kinetic modeling of particle size distribution of soot in a premixed burner-stabilized stagnation ethylene flame (2015) Combustion and Flame, 162 (9), pp. 3356-3369, DOI: 10.1016/j.combustflame.2015.06.002

- **Pelucchi, M., Frassoldati, A., Faravelli, T., Ruscic, B., Glarborg, P.** High-temperature chemistry of HCl and Cl<inf>2</inf> (2015) Combustion and Flame, 162 (6), pp. 2693-2704, DOI: 10.1016/j.combustflame.2015.04.0

- **Salenbauch, S., Cuoci, A., Frassoldati, A., Saggese, C., Faravelli, T., Hasse, C.** Modeling soot formation in premixed flames using an Extended Conditional Quadrature Method of Moments (2015) Combustion and Flame, 162 (6), pp. 2529-2543, DOI: 10.1016/j.combustflame.2015.03.002

- **Frassoldati, A., D'Errico, G., Lucchini, T., Stagni, A., Cuoci, A., Faravelli, T., Onorati, A., Ranzi, E.** Reduced kinetic mechanisms of diesel fuel surrogate for engine CFD simulations (2015) Combustion and Flame, 162 (10), pp. 3991-4007, DOI: 10.1016/j.combustflame.2015.07.039

- **De Bruycker, R., Pyl, S.P., Reyniers, M.-F., Van Geem, K.M., Marin, G.B.** Microkinetic model for the pyrolysis of methyl esters: From model compound to industrial biodiesel (2015) AIChE Journal, 61 (12), pp. 4309-4322, DOI: 10.1002/aic.14953

- **Evans, M.J., Medwell, P.R., Tian, Z.F., Frassoldati, A., Cuoci, A., Stagni, A.** Ignition characteristics in spatially zero-, one- and two-dimensional laminar ethylene flames (2015) 22nd AIAA Computational Fluid Dynamics Conference, 15 p.

- **Galletti, C., Ferrarotti, M., Parente, A., Tognotti, L.** Reduced NO formation models for CFD simulations of MILD combustion (2015) International Journal of Hydrogen Energy, 40 (14), pp. 4884-4897, DOI: 10.1016/j.ijhydene.2015.01.172