POLITECNICO
MILANO 1863

NetSMOKE

User Guide

CRECK Modeling

DEVELOPER
Mensi Matteo
matteo.mensi@mail.polimi.it
github.com/Snatched

Summary

Introduction

NetSMOKE is an integrated framework associated with the OpenSMOKE++ libraries and software developed within the CRECK Modelling group of Politecnico di Milano.

It provides tools and interfaces to solve equivalent reactor network models using the core OpenSMOKE++ functionalities as a back-end.

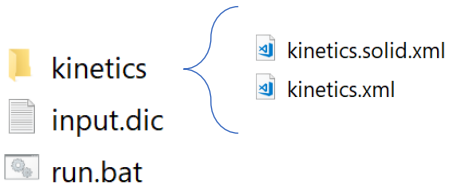
This user guide explains how to use the software and how to compile correctly input file and kinetics mechanism for usage within NetSMOKE.

Generic usage & setup

Place the NetSMOKE executable and associated dlls in any location you prefer. It will be called using batch files pointing at it from your simulation directory.

A simulation directory should contain:

- “input.dic” text file
- “kinetics” folder
- “run.bat” batch file



The input is explained later. The folder called “kinetics” should contain a kinetic scheme in *.xml format. This requires preprocessing a CHEMKIN scheme using the OpenSMOKE++ Preprocessor (part of the OpenSMOKE++ suite).

For gas phase only – one kinetic scheme. Name it “kinetics.xml”.

For solid phase – add a second solid phase kinetic scheme and name it “kinetics.solid.xml”.

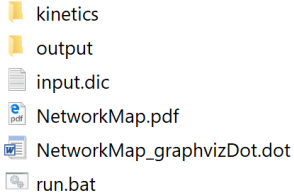
The run.bat can be created following this procedure: create a *.txt file and open it. Inside write the path to the NetSMOKE executable in the format `"..\..\..\exe"` including the `""`. In a new line write `pause`.

Save the file and rename it as “run.bat”. Double clicking this file will launch the program setting the simulation directory as the working directory.

The simulation will produce a folder called “ouput”. Inside this folder one can find dedicated subfolders for each reactor in the network (provided the options was set by the user) and a `StreamSummary.out` file which contains the most important parameters for each stream in the network.

Graphviz usage

The program is capable of producing a functional *.pdf file containing a simple schematic of the network to help with navigating results and double checking that the input file is what the user meant to say.



To do so the opensource free software Graphviz is used. After installing it you need to expand the PATH environmental variable on your machine by adding the path to the Graphviz binaries folder.

In the NetworkMap.pdf:

Reactor

- is bigger than the other units.
- changes shape according to its type: a circle is a PSR; a rectangle is a PFR.
- changes colour according to its energy features: red is Adiabatic; dashed red is HeatExchanger; blue is Isothermal.

Mixer

- is an inverse triangle.
- changes colour according to its energy features: red is Adiabatic; blue is Isothermal.

Splitter and Phase-splitter

- are triangles.
- streams exiting a splitter have, next to their identification number, the splitting ratio.

Stream

- changes colour according to its phase: black is Mix, orange is Gas, green is Solid.

Input file

The input file consists of all the inlet streams and units of the reactor network model. It requires a UTF-8 encoding plain text file in the format *.dic. You can achieve this by simply creating a *.txt file and edit it with any free text editor (Windows Notepad, Notepad++, PSpad, VScode, Atom and the likes are recommended. WordPad, Word or any document editor is discouraged) and then renaming the file as “input.dic”. This will change the extension to the one recognized by the software preventing conflicts with other files.

Currently, there are four main units:

- Reactors
- Splitters
- Mixers
- Phase splitters

The input file consists of three main types of declaration, which are General Options, Units and Streams. No particular order is needed for these declarations.

This gives us the following main keywords. Please note that each main keyword starts with a “@”.

- @GeneralOptions
- @Reactor
- @Mixer
- @Splitter
- @Phase-splitter
- @Stream

The general rule is to use “//” to terminate a declaration. Do so by adding a “//” after each declaration, making sure that the file ends with “//”.

General options

The General options declaration contains all the generic instructions for the program to run. All future updates on more user-controllable options like tolerances and desired solving methods will be added in this declaration.

Currently, seven options are settable.

SystemPressure – Mandatory. This sets the pressure of the network. Accepts bar, atm and Pa as units.

Usage example: SystemPressure 1 atm

NoRecycles – Writing this line enables the sequential solver. Greatly increases speed when solving "trivial" networks where there are no recycles.

SetLegacyOSppPrint – Enables dedicated reactor result printing. In the "output" folder you will find folders named like the reactors in the input file. Each of these contains detailed information on the reactor.

SetOutletsVideoPrint – Enables printing of the outlet streams in the console at the end of the simulation.

SolidDensity – Mandatory for simulations containing solid species. Accepts a number in kg/m3.

Usage example: SolidDensity 850 kg/m3

Restart – resume the simulation from the backup contained in the backup folder.

RTD – perform RTD step experiment tracking the reported species.

Usage example: RTD N2

Reactor

Reactor declaration starts with:

```
@Reactor    (Name)
```

(Name) consists of the letter "R" followed by a series of numbers that identifies the reactor.

The declaration continues with some lines that must start, as already stated, with a Reactor keyword.

(the order by which the keywords are placed is not important).

Reactor keywords are:

1. Phase(compulsory)
2. Type(compulsory)
3. Energy(compulsory)
4. Temperature
5. UA
6. ResidenceTime/Volume/Diameter+Length (compulsory)
7. Inlet_stream(compulsory)
8. Outlet_stream(compulsory)

1) Phase

This line requires one word to indicate the kinetic scheme necessary for the reactor resolution.

Currently, Phase can only be “Mix”, “Gas” or “Solid”. Mix or Solid phase trigger a two-phase reactor and currently only PSRs can be solved as two-phase reactors. Don’t use non-gas PFRs.

Example:

```
Phase Mix
```

2) Type

This line requires one word to indicate the reactor type.

Currently, type can only be “PFR”(Plug Flow Reactor) or “PSR”(Perfectly Strirred Reactor).

Example:

```
Type  PSR
```

3) Energy

This line requires one word to indicate the energetic behavior of the reactor.

Currently, Energy can only be “HeatExchanger”, “Adiabatic” or “Isothermal”.

Example:

```
Energy      Isothermal
```

4) Temperature

This keyword is required only for Isothermal reactors. It is not needed for HeatExchanger or Adiabatic reactors.

This line indicates the reactor temperature and requires two elements: a number (the Temperature value) and the Temperature unit of measurement. Only °C and K are recognized as units.

Example:

```
Temperature  700.23  °C
```

5) UA

This keyword is required only for HeatExchanger reactors. It is not needed for Isothermal or Adiabatic reactors.

This line indicates the product between the heat transfer coefficient and the exchange surface. It requires two elements: a number (the UA value) and the UA unit of measurement. Supported units are W/K.

Example:

```
UA    52.679 W/K
```

6) ResidenceTime/Volume/Diameter and length

This line gives an idea of the dimensions of the reactor.

This line requires two elements: a number (the definition value) and the unit of measurement. Residence time can be s, ms, h, min. Volume can be either m^3 or cm^3. Diameter and length can be either m or cm.

Examples:

Define by residence time only:

```
ResidenceTime  25.78 s
```

Define by volume only (Only PSR are supported):

```
Volume  5.0 m3
```

Define by length and diameter (Only PFR are supported):

```
Length  30.0 m
Diameter 3.0 m
```

7) Inlet_stream

This line indicates the stream that enters the reactor (only one stream can enter a reactor).

It requires one element: the number (an integer) that identifies the entering stream.

Example:

```
Inlet_stream        6
```

8) Outlet_stream

This line indicates the stream that exits the reactor (only one stream can exit a reactor).

It requires one element: the number (an integer) that identifies the exiting stream.

Example:

```
Outlet_stream
```

Complete example of a reactor declaration(remember to end the declaration with a "//"):

```
@Reactor                    R1
Phase                   Gas
Type                    PFR
Temperature             257.98 K
Energy                  Isothermal
Inlet_stream            4
Outlet_stream           12
ResidenceTime           36.8 s
//
```

Mixer

Mixer declaration starts with:

```
@Mixer    (Name)
```

(Name) consists of the letter “M” followed by a series of numbers that identifies the mixer.

The declaration continues with some lines that must start, as already stated, with a Mixer keyword.

(the order by which the keywords are placed is not important).

Mixer keywords are:

1. Energy(compulsory)
2. Temperature
3. Inlet_stream (compulsory)
4. Outlet_stream (compulsory)

1) Energy

This line requires one word to indicate energetic behavior of the mixer.

Currently, Energy can only be “Adiabatic” or “Isothermal”.

Example:

```
Energy      Adiabatic
```

2) Temperature

This keyword is required only for Isothermal mixers and. It is not needed for Adiabatic mixers.

This line indicates the temperature of the exiting stream and requires two elements: a number (the Temperature value) and the Temperature unit of measurement.

Example:

```
Temperature  259.497   K
```

3) Inlet_stream

This line indicates the streams that enters the mixer (it is compulsory to have more than one entering stream).

It requires a series of numbers (integers) that identify the entering streams, separated by a comma or a space. List of streams ends with a”;”.

Example:

```
Inlet_stream        6,7,1,9;
```

4) Outlet_stream

This line indicates the stream that exits the mixer (only one stream can exit a mixer).

It requires one element: the number (an integer) that identifies the exiting stream.

Example:

```
Outlet_stream       10
```

Complete example of a mixer declaration (remember to end the declaration with a "//"):

```
@Mixer      M2
Inlet_stream            3,4,25;
Energy                  Isothermal
Temperature       200 K
Outlet_stream           18
//
```

Splitter

Splitter declaration starts with:

```
@Splitter    (Name)
```

(Name) consists of the letter "S" followed by a series of numbers that identifies the splitter.

The declaration continues with some lines that must start, as already stated, with a Splitter keyword.

(the order by which the keywords are placed is not important).

Splitter keywords are:

1. Inlet_stream (compulsory)
2. Outlet_stream (compulsory)

1) Inlet_stream

This line indicates the stream that enters the splitter (only one stream can enter a splitter).

It requires one element: the number (an integer) that identifies the entering stream.

Example:

```
Inlet_stream        10
```

2) Outlet_stream

This line starts a list of the number (an integer) that identifies the exiting stream followed by its splitting ratio (a double) (remember to separate the two elements with a space or a comma).

End the list by introducing a ";" in the last line interesting the Outlet_stream.

Of course, the sum of the splitting ratios must be 1.

Example:

```
Outlet_stream           2,0.6
                        4,0.4 ;
```

Remember that splitters are mono-phase, therefore an Outlet_stream cannot have a different phase with respect to another Outlet_stream or the Inlet_stream. That said, error message might be uncorrect due to the default assignment of the phases of streams inside the network.

Complete example of a splitter declaration (remember to end the declaration with a "//"):

```
@Splitter S2
Outlet_stream           11,0.62
                    12,0.28
                    14,0.1;
Inlet_stream            20
//
```

Phase splitter

Phase-spitter declaration starts with:

```
@Phase-splitter    (Name)
```

(Name) consists of the letters "PS" followed by a series of numbers that identifies the phase-splitter.

The declaration continues with some lines that must start, as already stated, with a Phase-splitter keyword.

(the order by which the keywords are placed is not important).

Phase-splitter keywords are:

1. Inlet_stream (compulsory)
2. Outlet_stream (compulsory)

1) Inlet_stream

This line indicates the stream that enters the phase-splitter (only one stream can enter a phase-splitter).

It requires one element: the number (an integer) that identifies the entering stream.

Remember that this stream phase must be Mix.

Example:

```
Inlet_stream        3
```

2) Outlet_stream

This line starts a list of the number (an integer) that identifies the exiting stream followed by its splitting phase (a string) (remember to separate the two elements with a space or a comma).

End the list by introducing a ";" in the last line interesting the Outlet_stream.

Currently, the splitting phase can only be Gas or Solid.

Example:

```
Outlet_stream           2,Solid
                  4,Gas;
```

Complete example of a phase-splitter declaration (remember to end the declaration with a "//"):

```
@Phase-splitter PS2
Inlet_stream            19
Outlet_stream           23,Gas
                  21,Solid;
//
```

Stream

Only system input streams declarations are needed.

Stream declaration starts with:

```
@Stream      (Name)
```

(Name) consists of a number (an integer) that identifies the stream.

The declaration continues with some lines that must start, as already stated, with a Stream keyword.

(the order by which the keywords are placed is not important).

Stream keywords are:

1. Phase
2. SolidType
3. MassFlowRate
4. Temperature
5. MassFraction

1)Phase

This line requires one word to indicate the stream phase.

Currently, Phase can be only Gas, Solid or Mix. For now, to define a solid stream, just use a Mix stream with only solid species.

Example:

```
Phase       Solid
```

If the keyword “Phase” is absent, the default stream phase is Mix.

2)SolidType

This keyword is required only for system input solid streams. This is not yet implemented due to the function to convert specific solid types into proper CHEMKIN species being WIP.

3)MassFlowRate

This keyword is compulsory only for system input streams.

This line requires two elements: a number (the MassFlowRate value) and the MassFlowRate unit of measurement. Supported units are kg/s, g/s, kg/h, kg/min.

Example:

```
MassFlowRate       50.35       kg/s
```

4)Temperature

This keyword is compulsory only for system input streams.

This line requires two elements: a number (the Temperature value) and the Temperature unit of measurement.

Example:

```
Temperature  693.56  °C
```

5)MassFraction

This keyword is compulsory.

This line starts a list of Compounds (a string) followed by its mass fraction (a double) (remember to separate the two elements with a space or a comma).

End the list by introducing a ";" in the last line interesting MassFraction.

If the stream is solid and is a system input, then the mass fraction compounds will be given (future planning) in terms of C, H, O, N, S, Ash and H2O (Elemental or ultimate analysis). For now, this is not possible yet, so use specific species of the mechanism.

Mass fractions must sum up to 1.

Complete example of a system input solid stream declaration (remember to end the declaration with a "//"):

```
Stream 6

Phase             Gas

MassFlowRate              45.7 kg/s

Temperature       333.8 K

MassFraction            CO2 0.54

                  H2 0.26

                  O2 0.1

                  N2 0.1;

//
```

Full example

With this user guide, you can find an example "input.dic" to see a complete, simple network definition example.

Advanced features

Resuming a simulation

NetSMOKE performs backups of the current simulation status, which are put in the "backup" folder automatically created by the program. Togheter with a "Backup.bin" file, a copy of the input file associated with that backup is present with a corresponding timestamp and date of its generation.

In case of crash or simulation abort due to external events (like a power outage), the user can set the "Restart" keyword in the general options section to use the last system status to resume calculations.

Performing an RTD analysis

It is possible to simulate step experiments when the reactor network is composed of only perfectly stirred reactors by perturbating an inlet stream and monitoring the concentration of the tracer in all the outlet streams.

First, run a version of the network in absence of the tracer. Once the simulation is in steady state and completed, open the input file and:

1. Add the “RTD” keyword together with the tracker species in the general options section (i.e.: RTD Ar)
2. Add the “Resume” keyword to the general options section.
3. Modify the composition and flow rate of one or more inlet stream to include the tracer.

The program will run the transient version of the network until steady state. In the folder “RTD”, automatically generated, a file named X_track.out, where X stands for the species used in the RTD keyword, can be found. This file shows the outlet concentration as a function of time in each stream leaving the network.

Contacts

If needed, you can contact the developer using the following informations:

Matteo Mensi

matteo.mensi@mail.polimi.it

Or use the GitHub issue tracker at this repository: github.com/Snatched/NetSMOKE