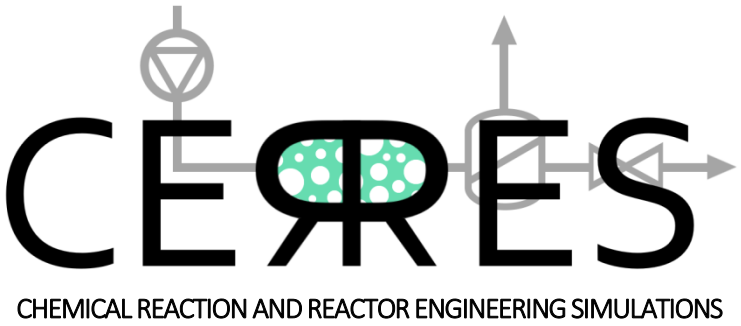


CHEMICAL REACTION AND REACTOR ENGINEERING SIMULATIONS

CERRES documentation

v0.02b

18. 4. 2021

Author: Damjan Lašič Jurković, National Institute of Chemistry, Ljubljana, Slovenia

About

CERRES (Chemical Reaction and Reactor Engineering Simulations) is a program designed for the simulation of various types of chemical reactors under different operating conditions with user-supplied chemistry, allowing for complex bulk and surface reaction (micro)kinetics. Furthermore, it provides additional modes of operation such as the comparison of the model results to experimentally measured values and reaction rate parameter optimization (fitting). The main goals of the software are computing efficiency, ease of use and wide functionality. CERRES is being developed by Damjan Lašič Jurković at the Department of Catalysis and Chemical Reaction Engineering at the National Institute of Chemistry in Ljubljana, Slovenia. Most of the software (including the UI) is written in Python while the computational backend is implemented in C.

The main features of the software are:

- Simulation of different types of chemical reactors
- Complex user-defined chemistry
- Model-experiment compare, parameter optimization, sensitivity analysis
- Efficient computation
- Ease of use
- Relevant examples from literature

The usage of the program is free of charge for any academic, educational or personal use; we only require the users to cite the CERRES webpage (www.cerres.org) and later the main CERRES paper (not yet released) in any scientific publications containing results obtained by the program. For commercial/for-profit use, please contact the developers. See the full license for more information.

Please note that this is the Beta version of the software, and thus some bugs are expected.

We are always happy to receive any feedback, bug reports or suggestions for additional features!

Contact:

Damjan Lašič Jurković
Department of Catalysis and Chemical Reaction Engineering
National Institute of Chemistry
Hajdrihova 19
1000 Ljubljana
Slovenia
e-mail: damjan.lasic@ki.si

Table of contents

1 License

The present version of CERRES is an open Beta version with the following license (licenses of the utilised libraries and bundled MinGW-w64 distribution of GNU GCC compiler are included in the "Used libraries and programs" chapter):

"

CERRES - Chemical Reaction and Reactor Engineering Simulations

Copyright (C) 2020 Damjan Lašič Jurković

All rights reserved.

By continuing to install or use this software (CERRES), you agree to be bound by the terms of this agreement.

CERRES is free (as in no cost) for academic, educational and personal use. In any case where the results of CERRES are used in a scientific publications, it is required that authors give the credit to the developers by citing the CERRES website (www.cerres.org). For any other cases, including but no limited to commercial or for-profit use (primarily intended for or directed towards commercial advantage or monetary compensation), the usage of CERRES is prohibited. In order to use CERRES for such applications, please contact the developers for a different licensing arrangement.

It is forbidden to distribute CERRES in any form. It is forbidden to modify CERRES binaries or attempt to reverse-engineer the source code.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NOTE:

The CERRES installer also installs the MinGW-w64 binaries (which include the GNU GCC compiler). These are a separate program invoked by CERRES via a subprocess call. The binaries are located in the <CERRES installation folder>/cvode_execution/mingw_libs/mingw. The program is covered by its own GNU GPLv3+ license, available in the root folder of CERRES installation. The source code is made available online in the public CERRES repository (https://github.com/DamjanLasicJurkovic/CERRES_public).

"

2 Installation procedure and requirements

2.1 Requirements

The current version is supported on x64 conforming CPUs (nearly all of Intel and AMD 64-bit processors). It is only supported for the Windows OS (tested on Windows 7, 8.1 and 10).

2.2 Installation

The installation process is designed to be as seamless as possible. Just run the CERRES installer (obtained from www.cerres.com). It should install CERRES and any other required dependencies. After the installer completes, the program is ready for use. It is recommended to first run the CERRES test (via the 'Help' menu in the UI), which runs some examples and thus checks that the software is working as intended.

2.3 Bugs/Troubleshooting/Feedback

Error messages in the CERRES output console are supposed to be as informative as possible in order to provide the user with most probable solutions. However, since this is the Beta version of the program, we expect some errors may be found by users which were not predicted and are not covered by the error handling procedures in CERRES. Therefore, a built-in bug report generation is provided, which produces a bug report file including the state of the output console, input parameters and copies of experiment and chemistry data – everything the developers need to reproduce the bug. If all else fails, we would be happy to receive also any user description of observed buggy behaviour. All bug reports, comments, feature requests and (constructive) criticism should be sent to damjan.lasic@ki.si.

3 Using CERRES

The usage of the program is designed to be easy and intuitive without any programming knowledge required. However, users should be familiar with general approaches and techniques of mathematical modelling of reactors and chemical processes. Currently, the usage is only supported through the graphical user interface. In later iterations, command line usage and python API may be implemented.

3.1 The graphical user interface (GUI)

The graphical user interface is used to open/edit/save the input parameter files and run them. It contains an output console where the prints and error message are printed. In addition, "Chemistry editor" and "Experimental data editor" are also provided for editing chemistry and experimental data files, respectively.

In order to run a file with CERRES, first load it via the "Load" or "Load example" menus. Then, clicking the "Run" button will run the current file, which contains all the required parameters. In most cases, this will also result in plotting the resulting data.

3.2 Input files

There are three main types of input files used by CERRES: input parameters files, chemistry files and experimental data files. All are described in more detail below.

3.2.1 Input parameters file

This is a text file containing all of the input parameters required for CERRES to run a simulation. The input parameter file also contains the names of chemistry and experimental data files to use. Can be loaded from or saved to any directory, but it's recommended to keep all input files in "~user\AppData\Local\cerres\input_files\parameter_files".

3.2.2 Chemistry file

The chemistry file contains the data about all of the species, reactions and reaction rate constants for a specific system. Must be located in "~user\AppData\Local\cerres\input_files\chemistry_files". The path can be opened in system viewer from the user interface as well for easier locating. It is recommended to edit the chemistry files via the provided "Chemistry editor". This will always assure correct formatting and file structure. The chemistry file is intentionally separate from the parameter file so different input parameters files can use the same chemistry data without duplication. The files are in CSV format, so users can also edit them in, for example, Microsoft Excel or LibreOffice Calc, but should be careful to retain the original formatting (commas as separators, utf-8 encoding). It is recommended to open the file in "Chemistry editor" and perform the "Check" procedure after any manual adjustments.

3.2.3 Experimental data file

Only used when experimental data is needed ("Exp.-Model compare" and "Optimization" operating modes). Contains the data about the number of experiments performed, experimental conditions and measured concentrations of different species. Can be edited with "Experimental data editor". Must be located in "~user\AppData\Local\cerres\input_files\exp_data". As with the chemistry files, these can also be edited in any CSV supporting editors.

3.3 Operation modes

CERRES supports five different operation modes, which dictate how the simulations are performed and ultimately the type of the results generated. The operating mode is set in the Input parameters file ("General" tab). The operating modes are:

3.3.1 Single parameter

Simulation using a single set of operating conditions. Result is the time-evolution of (position-dependent) concentrations, surface coverages, conversions and selectivities of all the chemical species in the reactor. No experimental data file needed for this simulation. The "WGS on Cu – plug flow reactor (PFR)" is an example of Single parameter simulation.

3.3.2 Multi parameter

Same as single parameter but performed multiple times at different operating conditions. Steady-state outlet concentrations/conversions/selectivities can be compared via the "Multi" plot. Furthermore, all of the sub-simulations can be plotted/exported in the same manner as the single parameter plot. No experimental data file needed for this simulation. The "WGS on Cu – surface coverages at diff P. and T." is an example of Multi parameter simulation.

3.3.3 Model-Exp. Compare

Comparison between experimental and model data. Experimental data file is needed for this mode. The model is computed, then the results are compared to experimentally measured data either in the form of "Parity" or "Outlet" plots. Whichever conditions available from the experimental data file are taken from there, while the rest are taken from the input file. "Methanol synthesis on CuZnAl – Exp.-model compare" is an example of such comparison.

3.3.4 Optimization

Kinetic parameter optimization using experimental data. Experimental data file is needed for this mode. Additional "Optimization" tab is available in the GUI for this mode, where it is possible to set which parameters to optimize. Optimized can be both/either the catalyst site concentrations (which are otherwise an input to the model), and/or the reaction rate constants parameters. The optimization algorithm recursively computes the model values at experimental operating conditions (like in "Model-Exp. Compare"), computes the sum of squared differences between the experimental and model points (for selected, optionally weighted species), then adjusts the parameters to minimize this sum, which achieves a better fit between the model and experiments. The optimization routines from the SciPy[1] library are used for this. More information can be found at the SciPy website (SciPy minimize). "Methanol synthesis on CuZnAl – A + Ea optimization" is an example of this operation mode, where the pre-exponential factor and activation energy of one of the reactions are optimized to improve the fit from the "Model-Exp. Compare" example above.

3.3.5 Sensitivity

Sensitivity analysis is performed by changing some parameters of different reactions and comparing its effect on model concentrations. No experimental data is required for this operation mode. The procedure is as follows. One of the reaction rate parameters is selected (via inputs) – for example, the forward rate activation energy (Ea_for). A "change factor" is also selected, for example, 1.1. The program will then iterate over all of the reactions in the chemistry file. For each reaction, the selected parameter is multiplied by the change factor, and the difference in selected value (for

example, one of the products conversions), is noted compared to unchanged constants. The same is done for dividing by the parameter value with said factor. Then, the results are plotted in a graph to compare the effects of the changes for each of the reactions. Additionally, a threshold can be applied to only display the reactions with some % of effect, which is handy for big systems. The sensitivity analysis can be used for example for model reduction, or for determining the most important reactions for optimization purposes. The example file "Butane dehydrogenation – sensitivity analysis" demonstrates this mode on a system with 107 reactions, where only handfuls are identified as significant. The selected parameter and the factor are set in the "Solver settings" tab.

3.4 Input parameters

CERRES performs each run on a set of input parameters. These parameters are provided by the user through the GUI, and can also be saved as a .txt file (Input parameters file). Generally, the inputs are either checkboxes (yes or no), drop-menus (selection among options) and entries (user input values). All individual inputs have information widgets which display the description, type, units, format, default value etc. of the specific input, so they are not individually explained here. Additionally, the entries are coloured red when the input is not of the correct format.

In the GUI, the inputs are logically separated into different subsets with tabs with the subset name. Each tab is further divided into short titled sections to logically group the parameters. Below, each subset of input parameters is explained in more detail.

3.4.1 General

The general tab contains the names of experimental (only used when needed) and chemistry files (always needed). The operation mode is selected in the general tab as well. The general tab also includes the file's title, author, description and references.

3.4.2 Reactor setup

In the reactor setup tab, the reactor type is selected. For the selected reactor, various other parameters are then available (while others being greyed out). These include information about the initial state of the bulk phase concentrations at simulation start, reactor dimensions, number of discretization finite volumes for multi-dimensional reactors and which of the mass transport limitation phenomena to use. Some of the values that would be expected here (like reactor volume) are rather included in the operating conditions tab, because different values can be used in multi parameter simulations.

3.4.3 Operating conditions

The operating conditions tab contains different parameters like temperature, pressure, flow rate etc. Two different types are used for the data, "Float arrays" and "Lists of float arrays". Each element of a float array corresponds to one of the many simulations to perform. So, for example, "Temperature" is of type float array. If we are doing 5 simulations at different temperatures, the input could look something like [100,200,300,400,500]. The first simulations will be at 100 °C, while the last at 500 °C. "Gas partial concentrations" is an example of a "List of float arrays". Each element must be the length of the amount of gas species in the chemistry file. So for example, if H_2 and CO_2 are the only gas species in the gas file, and we are doing 5 simulations as above, we may input something like [[1,1],[1,2],[1,3],[1,1],[1,1]]. In this case, equimolar H_2/CO_2 ratio would be used at simulations at

temperatures of 100, 400 and 500 °C, while ratios of 1:2 and 1:3 would be used for 200 and 300 °C, respectively. The entry will be coloured red if the format is incorrect.

There are certain rules for the lengths of the operating conditions arrays/lists. Generally, all of the lists that are not of length 1 (ex. [5] or [[1,2,3]]) should be of the same length (=the nr of performed simulations). Those with a length of 1 will be assumed to be the same value for all simulations. So, in the above case, if Temperature was [100], 100 °C would be taken for all five of the H_2/CO_2 ratios.

The exception to the above rule is the use of permutation. If any condition is selected to be permuted, it can have any length. For each element in the permuted conditions, ALL of the other conditions will be replicated. For the above example, if we set Pressure to be permuted with a value of [1e5, 2e5, 5e5], 15 total simulations would be performed, as the five temperatures and H_2/CO_2 ratios would be repeated for the pressures of 1, 2 and 5 bar. Permuting multiple values will recursively multiplicatively expand the number of simulations, which can reach very high numbers if not careful.

In cases where experimental data is used, the operating conditions values that are included in the experimental data are given precedence. For those conditions, the values in the input file are disregarded. Permutation does not work with operation modes which implement experimental data.

Additionally, user defined variables' values can be defined here. The user defined variables must match those defined in the chemistry file. User defined variables are explained in more detail in the "Chemistry" section.

3.4.4 Solver settings

The settings for the numerical ODE solving backend can be set here. The backend is implemented in C, utilizing the CVODE[2] solver from the SUNDIALS[3] library. The C code for the actual ODE functions is dynamically generated from the chemistry data and compiled using the GNU GCC compiler (https://gcc.gnu.org/) on the fly. The simulations can be multi-threaded on the simulation level. This means that for any run with multiple simulations (like "Multi parameter", "Exp.-Model compare" and "Optimization" operation modes), the number of simulations is divided amongst multiple threads. Dividing a single simulation is not supported, and in most cases, the overhead would not be worth it.

3.4.5 Data export

When "Export after computing" is checked, the results will be automatically exported after computation. Additionally, the export can be performed after computation by clicking the "Export now" button. Different kinds of exported data can be selected in this tab.

3.4.6 Results plotting

When "Plot after computing" and "Show image" are enabled, the results are automatically plotted and displayed as separate window figures after the run. They can also be manually plotted with the "Plot now" button after computation. Settings for the individual plot types can be set in this tab. Note that not all plot types are compatible with all operation modes – an error will be displayed in the case of incompatibility. The images can also be saved as PNG files with the chosen DPI. The plots are generated using the Matplotlib library.[4]

3.4.7 Optimization settings

This tab is only available when the selected operation mode is “Optimization”. The generation of this tab depends on the Chemistry file (set in the General tab), and a warning will be shown if the Chemistry file cannot be loaded or has errors. If the Chemistry file is legal, lists of species and reactions will be generated from which the user can select the appropriate species and parameters for regression. Boundaries and initial values (relative – default 1) can be set in this tab by the user, as well as additional optimization settings.

3.5 Calculations of various result types

In CERRES, different result types such as concentration, partial concentration, selectivity and conversion can be obtained, plotted and exported. For clarity and because all of them are not unambiguous for some of the reactor types the calculations are included below.

3.5.1 Concentration

The most straightforward one, all of the concentrations are in mol/L for gas and liquid species and in surface coverage fraction (unit-less) for adsorbed and site species.

3.5.2 Partial concentration

Partial concentration is generally computed as the molar ratios of all bulk phase (gas or liquid) species in the system, computed as:

$$y_i = \frac{c_i}{\sum_{j=0}^{nr\ bulk} c_j}$$

Where y_i is the partial concentration of species *i*, and c_i and c_j are concentrations of species *i* and *j*, respectively. The partial concentrations of all bulk species add up to 1.

In the cases of three-phase reactors, the partial concentration is calculated for the gas phase only. For membrane reactors, it is computed separately for retentate and permeate species, both summing to 1, making a total for all bulk species 2.

3.5.3 Conversion

Conversion is defined as the fraction of any inlet species that is converted, that is removed during the reaction process. It is only computed for the species that have non-zero inlet concentrations. Negative conversions indicate that more species exits than is consumed in the reactor, meaning that is additionally generated through some reactions. Conversions are calculated by the following formula:

$$x_i = \frac{c_{i,inlet}F_{inlet} - c_{i,outlet}F_{outlet}}{c_i F_{inlet}}$$

Where $c_{i,inlet}$ and $c_{i,outlet}$ are inlet and outlet concentrations of species *i* and F_{inlet} and F_{outlet} are the inlet and outlet total volumetric flow rates. In case of batch reactors, the concentrations without the flow rates are considered in the equation. In case of the three phase CSTR reactor, gas phase conversions (the phase with an inlet and an outlet) is based on flow rates, while liquid phase conversions are based on the initial concentration, if any.

For membrane reactors, the conversions are computed for the coupled species as if they were the same species (and as if the outlet flows from the retentate and permeate side were mixed). For the coupled species, the same value is reported for both "_ret" and "_perm" variants.

3.5.4 Selectivity

Selectivity is computed for each of the products, meaning each of the species with a zero inlet concentration. It calculated based on a specified element (in the "Solver settings"). It is defined as the amount of outlet moles of the element in some product divided by the total moles of the element in all products, as seen below.

$$sel_i = \frac{N_{E,i} c_i}{\sum_{j=0}^{nr\ prod} N_{E,j} c_j}$$

sel_i is the selectivity of product i and $N_{E,i}$ and $N_{E,j}$ are the number of moles of element E in the products i and j, respectively. The selectivities of all products add up to 1.

For batch reactors, products are considered as those with 0 initial concentrations (bulk only – gas and liquid). For three phase CSTR reactor, selectivities are meaningless as they would depend on the elapsed time of the process (gas is being removed while liquid is stationary), and are therefore set to 0, as is for the fixed gas reactor. For membrane reactors, the coupled species are again considered to be the same and mixed retentate and permeate flows are taken into account for selectivity calculations.

3.5.5 Site balance

Defined as the sum of all species' surface coverages, including free sites. Deviations from 1 can occur from incorrect reaction stoichiometry which can results in generation or consumption of total sites. Small deviations are otherwise expected due to numerical errors (very roughly speaking, less than 1e-8 in most cases, depending on solver tolerances). In some rare cases, numerical errors can lead to high deviations, especially with stiff systems with extremely long simulation times, where floating point rounding errors can accumulate over a large amount of solver steps.

3.5.6 Mass balance

The mass balance is computed as the fraction of outlet amount of each of the elements in the system (like C, H, O) compared to its inlet amount. Deviations can occur from incorrect reaction stoichiometry which can results in generation or consumption of various elements. Small deviations are otherwise expected due to numerical errors (very roughly speaking, less than 1e-8 in most cases). For batch reactors, it is instead computed as the sum of all moles of elements in the system compared to the initial values (including the catalytic sites). For membrane reactors, it is computed as if the inlet and outlet retentate and permeate flows were mixed. There are various variants of how mass balance is printed after computation (Solver settings). Mass balance can also be exported (Export settings).

average: (default) Defined as one minus the quotient of total inlet and outlet molar flow for each of the elements in all species. It therefore represents the deviation from ideal results (should be 0).

all: Prints the above average mass imbalance for each element separately.

full: Prints same as for All but does so for each problem rather than the average.

none: Doesn't print the mass balance. Also doesn't print the site balance.

3.5.7 Total concentration

The total concentration is computed for the bulk species, and is to be used for gas systems to check if the total pressure of the system equals the inlet pressure (for example, to see if “Velocity change” setting is required). It is calculated as a sum of all the bulk species and in the case of three phase systems as the sum of all gas species.

4 Chemistry

The models implemented in CERRES consider a specific chemistry set, which includes the phases, species, reactions, and reaction rate definitions for a certain system. In order to decouple this part from other parameters, the chemistry parameters are saved as separate chemistry files, which can be edited independently and used in multiple different models. This chapter covers the chemistry implementation and rules.

4.1 Species and phases

Each chemistry set contains a list of species related to a specific phase. Three different phases are distinguished: gas phase, liquid phase and solid phase. Most of the reactor models contain only 1-2 phases, for example, gas phase and solid (catalyst) phase.

4.1.1 Gas phase

Gas phase is considered as an ideal gas mixture, composed of the defined gas-phase species. The pressure of the gas phase is equal to the sum of partial pressures of all species and is in most cases constant and defined with the operating conditions for the certain model.

4.1.2 Liquid phase

Liquid phase is considered as an inert liquid carrier which contains certain concentrations of liquid-phase species. Contrary to the gas phase, the inert liquid carrier is implicitly assumed so that the liquid phase volume does not change.

4.1.3 Solid (catalyst) phase

The solid phase in CERRES implies a catalytic material. All of the solid-phase species are species that are adsorbed (chemisorbed) on the catalytic surface. Therefore, this is a not a solid-phase per se, but denotes the adsorbed species. The amount of the possible adsorption sites is defined in the input parameters as the concentration of active sites per volume of catalyst, and the amount of catalyst in the reactor is derived from the provided void fraction parameter. The surface amounts of different species are expressed in dimensionless surface coverages.

4.1.4 Catalytic sites

In CERRES, it is possible to express systems with multiple different catalytic sites. This allows user to compute models with simultaneous reactions on two or more different catalytic surfaces with different reaction sets, and even interactions between different catalytic sites (for example, both Zn and Cu sites in a CuZnAl catalyst). There are two different definitions of relations between different sites, which are either ‘mixed’ or ‘separated’, and are defined by the user. With separated sites, each site type is considered as a separate surface which cannot be coupled with another surface other than by desorption and re-adsorption of the species. Mixed sites imply a perfectly distributed mixture of sites, where reactions can occur between species adsorbed on different site types, and species can be adsorbed on two different sites at once. Furthermore, the dimensionless coverages of all adsorbed species and free sites sum to 1 in the case of mixed sites, and to N in the case of N separated sites.

4.1.5 Species definition

All of the species in a chemistry set are explicitly defined for each phase. It is required that all of the species’ names are unique. Some systems may only utilize one of the phases (for example,

homogenous kinetics problem in the gas phase), any combination of two phases or all three phases. In addition to the gas phase, liquid phase and solid phase species, the catalytic site species have to be explicitly defined as well. This is to improve the robustness of further stoichiometry and consistency checks, and is also used in equation generation to determine which site some reactions happen on.

4.1.6 Species naming

All of the species names must be unique. The allowed characters are uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9) and the special characters (*, #, %, &). The latter are conventionally used to denote catalytic surface sites. The species names should follow the standard chemical naming, so the species consists of elements, which is any group of letters starting with an uppercase or one of the special symbols. For example, He, H, O, $, *, *a, Cl are considered a single element and will be computed as such in the stoichiometry checker. Therefore, the species names cannot begin with a digit or a lowercase character, also a lowercase character cannot follow any digit in the species. All of the species may have a suffix that is ignored by the stoichiometry checker in order to make them unique. The suffix starts with an underscore (_) and can contain any letter but no numbers or other symbols. Empty suffixes are not allowed. For example, a user can denote gas phase and liquid phase hydrogen by having the species H2_g and H2_aq. The catalytic sites can only be of one element length and contain no numbers and no underscored suffix. Any whitespace anywhere in the species names is ignored.

Examples of **legal** spec names:

H2, CO2, COO, COO*, COO#_two, H2_gas, CO2_aq, CO**, CO*2, He, MeOH , H%, *o2

Examples of **illegal** spec names:

H2_g2, H2_, 2H, H!?, CH3(CH2)4CH3, methane, CH3CH2oH, hO2, Ho2o

Examples of **legal sites** spec names:

*, #, %, &, *a, *b, #eeee, S, Si

Examples of **illegal sites** spec names:

_a, a, #%, S2, s

The surface bound species' names are required to include the active surface site they are bound to. This is necessary for the parser to be able to determine on which of the sites the species is adsorbed to and how many sites it occupies. For example, one could define a hydrogen atom bound to a copper catalytic surface site (*) as H*, and then write a hydrogen dissociative adsorption reaction as H2 + 2* -> 2H*. For multiple-site bound species, an example reaction would be CO2_g + 2* -> COO** (identical to either COO*2 or CO2*2).

It is recommended to use the conventional molecule naming standards such as C2H5OH rather than Meth. The reason for this is that the program can then automatically check the stoichiometric consistency of all of the reactions and compute the reactor elemental mass balances as well. The standard for determining the elemental content for each of the species is the usual – an element is considered as any sequence of characters starting with a capital letter and followed by lower-case letters, such as H, Cu, Ar,… User can then leverage this to use customized parts such as 'Meth' above, but should be then careful to avoid reactions such as Meth -> C2H4 + H2O, as 'Meth' will be regarded as any other element, and the above reaction would have bad stoichiometry in the same

way as would, for example, Ar -> C2H4 + H2O, while C2H5OH -> C2H4 + H2O would be correct. The numbers in the species names denote the quantity of appearance of said elements, and therefore, from the stoichiometric point of view, OHH would contain the same elements as H2O.

It should be noted that CERRES does not support species such as CH3(CH2)4CH3. This functionality might be added later if needed.

Currently, CERRES does not support any special names often encountered in literature, such as M for any neutral molecule to be used for example in three body radical recombination reactions. Often in those cases, each of the neutral molecules is assigned a certain weight of contribution to the reaction. It was decided this functionality is excluded from CERRES due to the very high increase in complexity of the parser. However, users can emulate this behaviour by taking advantage of the user defined rate expression system, which is described further on.

4.2 Reactions

The reactions of a given chemistry set are described by a set of entries which contain information about the reagents, products, and reaction rate information.

4.2.1 Reagents and products

Each reaction contains a list of reagents and a list of products. The format by which these are input is a list of species, separated by the '+' sign. The species may be preceded by numerical constants, as is the case in usual chemical reaction representation.

Examples of reagent/product lists:

H2+2* (identical to H2 + * + *)

CO2 + CH4

OH+H+CH3

H*+H*

H*+CO2_g

CO2

The stoichiometry check will compute the number of each of the elements on both sides of the reaction equations and compare them, then report any discrepancy. In this way, the user gets warned if some typos have been made, and may avoid getting erroneous results and wrong mass balances from the model. Below is a list of example reactions and whether they succeed in the stoichiometry check:

H2 + 2* -> 2H* (SUCCESS)

CO2 + H2 -> CO + H2O (SUCCESS)

CO2+ H2 -> Co + H2O (FAIL)

CO2 +2H2 -> CO+H2O (FAIL)

H2 + * -> 2H* (FAIL)

Also, it should be noted what happens in cases of weird formatting. Any missing species are ignored, such as in "CO++ + +*". In addition, any isolated numbers are also ignored, such as in "CO+2*+3". Empty reactions assume no reagents/products are involved, and can be used, for example, to implement species removal terms.

4.2.2 Transfer between phases

Special consideration is given for "reactions" of transferring the species between phases.

For solid-gas and solid-liquid interactions, the adsorption and desorption reactions are written simply as a reaction between and empty catalytic site and a gas/liquid phases species, resulting in an adsorbed molecule. In addition, it is possible to use Eley–Rideal type of reactions where the adsorbed species directly react with gas or liquid phase species. When mass transport limitation from bulk gas to catalytic surface is enabled, only simple adsorption and desorption reactions are permitted, as discussed in the reactor section for mass transport limitations.

For transfers between the gas and liquid phase in three phase reactions, the only possibility is direct transfer by dissolution/evaporation. No reactions must be specified between the liquid and gas phase. Rather, it is implicitly assumed that the species with the same name but different suffix (for example, H2_gas and H2_liquid, each being in its own phase) can be transferred between the phases. The mass transfer coefficients and Henry's constants provided in operating conditions are used for the transfer rate – the order is determined by the order of the species in the gas phase. If there are more than one species with the same suffix in any of the phases, only the first one is considered.

Example:

Gas species: H2_g, CO2, CO_gas, CH4, H2_ion, O2_ooo

Liquid species: H, H3O, H2_l, O2_oo, O2_oooo, CO_aq, HCOO

Linked species will be: H2_g – H2_l, CO_gas – CO_aq, O2_ooo – O2_oo

4.2.3 Transfer through the membrane (membrane reactors)

The reactor types that include a membrane are "CSTR (membrane)", "PFR (membrane)", "PFR (membrane with diffusion)" and "PFR (membrane - fast)". When this reactor types are used, some special requirements have to be fulfilled in the chemistry file:

- Only gas or liquid reaction phase are allowed, not both
- All of the bulk species' suffixes have to be either "_ret" or "_perm"
- The bulk species have to be sorted, as in, "_ret" come first then "_perm"
- Only "_ret" species can interact with the catalyst
- No reactions are allowed between "_ret" and "_perm" species
- Bulk-bulk reactions between species on the same side (permeate or retentate) are allowed

In the system, "_ret" species are considered to be situated on the retentate side of the membrane, while "_perm" species are situated on the permeate side. The species with same names and different suffixes are coupled and can permeate through the membrane. The permeances are provided in the operating conditions, and are in the same order as the "_ret" coupled species. For example, if the species list is "H2O_ret, CO_ret, CO2_ret, H2_ret, CH3OH_ret, CHCOOH_ret, CO2_perm, H2_perm, CO_perm, H2O_perm", the species "H2O_ret, CO_ret, CO2_ret, H2_ret" will be coupled with "H2O_perm, CO_perm, CO2_perm, H2_perm", respectively. The order of

permeances in the operating conditions in this case would be "[[Permeance_H2O, Permeance_CO, Permeance_CO2, Permeance_H2]]". Note that in this case CH3OH_ret and CHCOOH_ret are only present on the retentate side and cannot permeate through the membrane.

See the "membrane reactor mass transfer" section in the "Reactors and mass transport phenomena" chapter for more details on the actual implementation.

4.2.4 Reaction rates

In CERRES, there are three possible default reaction rate modes to choose from, and they are the same for all of the reaction in a given chemistry set. The constant modes are 'direct constants', 'Arrhenius' and 'modified Arrhenius'. In addition, the user may define customized rate expressions for any desired reactions, which allows for a fine tuning of the kinetics. The user-defined expressions are covered in the next section.

The automatic reaction rate expression derivation is as following:

$$R = k_{for} {C_{reag,1}}^{ord_{reag,1}} {C_{reag,2}}^{ord_{reag,1}} \ldots - k_{back} {C_{prod,1}}^{ord_{prod,1}} {C_{prod,2}}^{ord_{prod,2}} \ldots$$

Where k_{for} and k_{back} are the rate constants of the forward and backward reaction, $C_{reag,i}$ and $C_{prod,i}$ are the concentrations of reagent and product *i*, and $ord_{reag,I}$ and $ord_{prog,i}$ are the orders for each species, which by default equals the stoichiometric amount of species *i* in the reaction. For surface species, the C is in dimensionless coverage, concentrations (mol/L) for the liquid phase species, and either pressure (Pa) or concentration (mol/L) for gas species, the latter only used in pure gas-gas bulk reactions (whereas pressure is used for adsorption and Eley-Rideal type reactions).

Examples:

Reaction 1:

H2 + 2Cu <-> 2HCu (where Cu is a catalytic site and H2 is in the gas phase)

Rate 1:

$$R = k_{for} p_{H2} {\theta_{Cu}}^2 - k_{back} {\theta_{HCu}}^2$$

Reaction 2:

H2+CO2 <-> H2O + CO (all in gas phase)

Rate 2:

$$R = k_{for} C_{H2} C_{CO2} - k_{back} C_{H2O} C_{CO}$$

C, *p* and θ in the equations above are concentration (mol/L), pressure (Pa) and the dimensionless surface coverage, respectively.

In addition, all of the reactions are considered reversible. If the reversible button is unchecked, the reverse rate is considered zero at all times. This check only works with automatic rate generation and not when user-defined expression for the specific reaction is active.

4.2.5 Reaction constant modes

The values of k_{for} and k_{back} are calculated from the provided user parameters in the following ways:

Direct constants: User provides the k_{for} and k_{back} values directly. Units are s^{-1} and whichever combination of $(mol/L)^{-X}$ and/or Pa^{-X} based on the reaction stoichiometry. **For gas-catalyst reactions, the unit is in Pa, for all other cases (gas-gas, liquid-catalyst, liquid-liquid) it's in mol/L.**

Arrhenius: User provides A_{for}, $E_{a,for}$, A_{back} and $E_{a,back}$. A is the pre-exponential factor with units of s^{-1} and whichever combination of $(mol/L)^{-X}$ and/or Pa^{-X} based on the reaction stoichiometry. E_a is activation energy with the units of kJ/mol. The reaction rate constants are computed as:

$$k = Ae^{-\frac{E_a}{RT}}$$

Where T is the temperature in K, R is the gas constant.

Modified Arrhenius: Same as Arrhenius, except that user provides the modified Arrhenius n-factor as well, and the constants are computed as (where n can be, for example, 1, 2, ½, -1, etc.:

$$k = T^n Ae^{-\frac{E_a}{RT}}$$

4.2.6 User defined rate expressions

User are given much more flexibility by being able to define their own reaction rate expression for any reaction by adjuring to the CERRES standard of doing so. The rate expression, in this case, is entirely replaced by the user provided expression.

The following operators are allowed in the rate expression:

* : multiplication

+ : addition

- : subtraction

^ : exponent

NOTE: division is not yet supported due to the difficulty of generation of analytical Jacobian matrices.

NOTE: for the exponents, only float/integer literals are allowed. **One should be careful with both negative and fractional exponential values to prevent solver convergence failures and unphysical results.** Since we want to allow negative and fractional exponents for concentrations ("reaction orders"), there are some errors due to numerical imprecision to take into account. If negative exponents are used (ex. c[x]^-1), division by zero can occur if the base is 0. In that case, the resulting error is prevented by evaluating to 0. If fractional exponentials (ex. c[x]^0.5) are used, negative numbers **may** result in error. In that case, to be sure to prevent crashes, the term is evaluated again to 0. The following "safe" power function as seen below is used at any time when doing risky exponents (negative and fractional exponents), so please keep this in mind when using them.

```
double pows(double b, double e) {
	if (b == 0 && e <= 0) { // check if divide by zero
		return 0;
	} else if (e-(int)e != 0 && b < 0) { // check if fraction and
negative base
		return 0;
	} else {
		return pow(d, e);
	}
}
```

The following expressions can be used as variables:

p/c[spec_name]: concentration, pressure or coverage of any species with spec_name. Pressure can only be used for gas species.

Examples: p[CH4], c[*], c[H*], c[CO2_aq],c[H2_gas]

T: temperature in Kelvin

k_for and k_back: kinetic constants, computed as defined above

user_var: any user defined variable, as explained above and in the operating conditions section

x: any number in integer, decimal, or floating point form (2, 0.1, 1.23e2) (has to be convertible to float – as seen below, no +/- signs though)

Additional rules:

Cannot chain exponentials (so c[O2]^2^2 is not allowed). Exponents always have to be a constant positive int/float number. For example, c[O2]^0.5 is allowed, while c[O2]^voltage is not. This is due to the usage of same Jacobian matrix orders for all operating conditions. The + and – signs are only used as operators and not as signs except if – is used right after ^ (but not +), so for example, -k_back*c[H2]+k_for*c[H]*c[*], k_for*10^-3 and k_for*c[H2]^-0.1 are legal while k_for*-1 and k_for^+2 are not. The operator precedence is as follows: ^,*, -/+. This means that, for example, 5/2^2 evaluates to 1.25.

Examples:

k_for*c[CH4]*c[CO2] - k_back*c[CO]*voltage^2

k_for*c[CO2]*c[*]^2 – k_back*c[CO2*] - k_for*c[CO2]*c[*]*poisoning_rate*10^-2

k_for*p[H2]*c[*]^1.1 – k_back*c[H*]^0.1 - k_for*c[CO2*]*0.01

4.2.7 User-defined variables

User also have to provide a list of any used user defined variables, to enable easier checking and reduce user input error. User variables can contain only lowercase and uppercase characters and the underscore sign ("_"). For the chemistry set to pass the checks, all of the unknown variables in the user defined expressions have to be found in the user defined variables list, and all of the variables in the list must be of correct format.

Examples of **legal** user defined variable names:

Voltage, current, DenSiTY, deact_speed, dampening_factor, reac_order, _, ___, _a, a_

Examples of **illegal** user defined variable names:

H2, _2, voltage_2, curr*, $, field(a)

4.3 Input chemistry files

The input chemistry file is a .csv (comma separate values) file, containing all the information of a particular chemistry set. All of the chemistry files are location in the folder "AppData/Local/cerres/input_files/kinetic_data". The folder can be directly opened from the GUI in a system viewer (Explorer). It is recommended to edit the chemistry files with the in-built CERRES

chemistry editor, for which no format and implementation knowledge are needed, however, the format is specified in a way that the user can edit the files in any popular csv editor, such as Microsoft Excel or LibreOffice Calc, with a low probability of errors occurring. See the “Input files” section for complete formatting guidelines (and investigate the chemistry files).

4.4 Chemistry editor (GUI)

It is recommended to edit chemistry files in provided Chemistry editor, which can be opened from the CERRES GUI. In the editor, reactions and species can be added, removed and edited. Furthermore, the chemistry file can be checked for errors, and the erroneous fields are handily coloured in the appropriate way to identify the error more easily. This can be seen after loading the “Example with errors (WGS on Cu)” and running the check. Error checks are also performed during execution of any input parameters file, and the simulation aborted if any critical errors occur. All of the checks are mentioned below, and all except stoichiometry are considered critical for the correct execution of the program.

4.5 Chemistry error checks in CERRES

CERRES performs three basic checks on running a set of input parameters for the chemistry.

4.5.1 Species consistency check

The species consistency check validates the following:

- → All of the species names are unique and of the correct format
- → All of the species contained in the reactions also appear in the list of species
- → There are no reactions between the gas and liquid phase
- → The reactions between the catalytic surface sites and bulk species are only with the liquid phase in three-phase systems (gas+liquid)
- → All of the surface bound species contain at least one catalytic site
- → If the system contains multiple catalytic sites and they are ‘separated’, no species can be simultaneously bound to two different catalytic sites
- → If the system contains multiple catalytic sites and they are ‘separated’, no reaction can occur between species bound on different catalytic sites

If any of these checks fails, the chemistry is invalid, and the program cannot proceed. An error is raised and an error message is printed.

4.5.2 Stoichiometry check

The stoichiometry check performs an evaluation of element balance for all reactions. If the check fails, the user is warned (and the violating reactions identified and displayed), but the program will still run. This is to ensure that non-physical disappearance of elements doesn’t occur by unintentional errors, but users can still make species disappearance or generation terms for their system if they are needed.

The stoichiometry check computes the quantity of each element on both product and reagent side of each reaction. If the elemental quantities aren’t equal for each of the elements, the checker reports the reaction as inconsistent.

4.5.3 User defined rate check

Checks whether the above requirements for user defined rate generation are compliant. Also checks whether all of the user variables used in the user defined rates are also defined in the user defined variables list (which is also checked for correct format). Additionally, the check is later performed when running the program and input data is known, if all the custom user defined variables needed are present in the inputs.

5 Models, reactors, equations and transport phenomena

This section covers the implementation of the models for different reactor types and mass transfer phenomena.

5.1 General about the models

The models in CERRES are always isothermal (constant temperature in time and space), and constant pressure is assumed in all points of the reactor (no pressure drops are computed by for example the Ergun equation). For the pressure, it can change in time with the changing concentrations. While this makes sense for example for batch type reactions, one should be careful when modelling continuous flow reactors such as CSTR. In CSTR, if "Velocity change" (see below) input is disabled, the inlet flowrate is assumed to be the same as outlet flow rate, which may lead to higher concentration than expected in the reactor when the gas expands during reaction (for example, H2->2H). In this case, the pressure in the reactor will be higher than the inlet pressure, although this may be unphysical. By using "Velocity change", the total concentration in such cases will always equal to the inlet total concentration, and the outlet flow rate will be changed accordingly.

5.2 Initial conditions

In most cases, the initial conditions in the reactor are that the bulk concentrations are equal to the inlet concentrations, and the catalytic surface is empty. These can be changed for some reactor types in the Reactor setup tab.

5.3 Species reaction mass balance terms

The reaction rate definition is already covered in the Chemistry section. The mass balances for the species depend on the species phase. For the surface adsorbed species, the mass balance is always just the sum of the reaction rates, multiplied by the stoichiometry coefficients, as shown in the equation below:

$$\frac{d\theta_i}{dt} = R_{i,cat} = \sum_{n=1}^{N} (-S_{i,n,f} + S_{i,n,b}) r_n$$

$R_{i,cat}$ is the reaction term of the mass balance (which equals to the total mass balance for surface species) through catalytic surface reactions. $S_{i,n,f}$ and $S_{i,n,b}$ are the forward and backward stoichiometry factors, e. g. how many times the species occurs in the reactions reagents or products, and r_n is the reaction rate. N is the number of reactions. The bulk species mass balances contain also the mass transport terms.

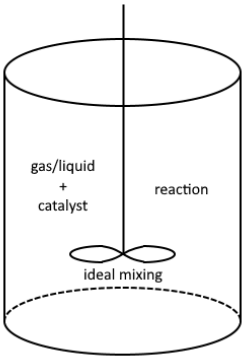
$$\frac{dC_i}{dt} = M_i + C^* \frac{1-\varepsilon}{\varepsilon} R_{i,cat} + R_{i,bulk}$$

M_i is the mass transport term which depends on the specific reactor type and $R_{i,bulk}$ is the reaction term for bulk-bulk type reaction. $R_{i,cat}$ in this case is multiplied by the expression where C^* is the concentrations of active sites per volume of the catalyst and ε is the void fraction, which links the changes in surface coverage to changes in bulk species concentration.

The mass balances for the reactor-specific mass transport phenomena are covered below. They only apply for the bulk (gas or liquid) species.

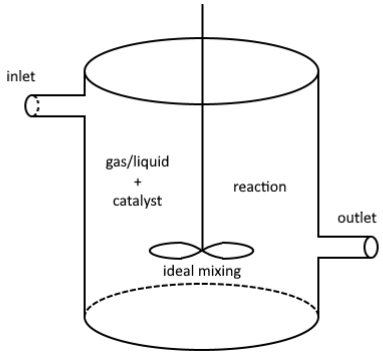
5.4 Reactor types and transport phenomena

5.4.1 Batch reactor



The simplest reactor type with no inlets or outlets. The mass balances are equal to the mass balances from reactions only, as defined above.

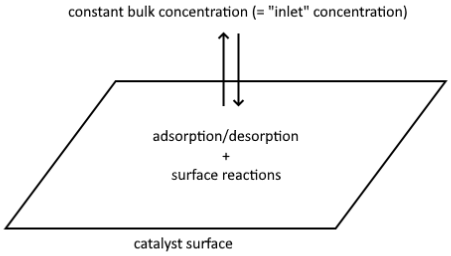
5.4.2 Continuous stirred-tank reactor (CSTR)



$$\frac{dC_i}{dt} = \frac{V\varepsilon}{F_{in}} C_{i,inlet} + C^* \frac{1-\varepsilon}{\varepsilon} R_{i,cat} + R_{i,bulk} - \frac{V\varepsilon}{F_{out}} C_i$$

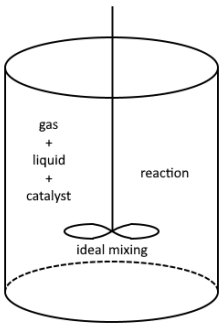
Simple one-point reactor with one inlet and one outlet. It is assumed that the reactor is ideally mixed, and concentration anywhere in the reactor equals C_i. V is the reactor volume, and F_{in} and F_{out} are the inlet and outlet gas/liquid flow rates. If the "Velocity change" parameter is disabled, they are equal, with it enabled, F_{out} is implicitly defined as such that the time derivative of the total concentration at all times is 0 for the gas phase.

5.4.3 Fixed bulk concentration reactor



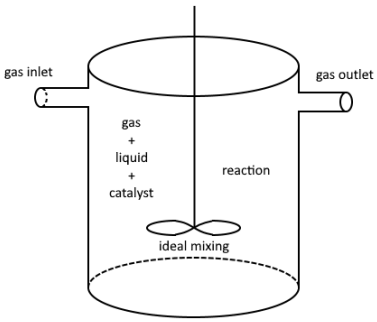
In this reactor type, only the surface species' time evolution is simulated, while the bulk concentrations are kept constant. Similarities can be drawn with usually implementations of kinetic Monte-Carlo approaches.

5.4.4 Batch (three-phase)



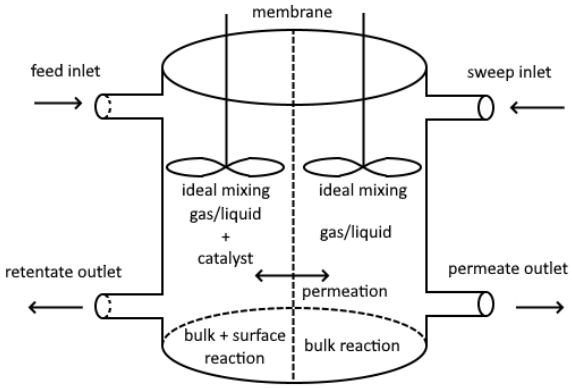
Same as the normal batch reactor, but allowing for three phase (gas+liquid+catalyst) chemistry.

5.4.5 CSTR (three-phase)



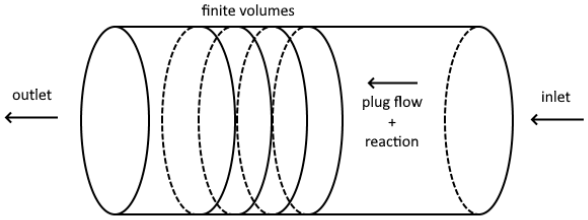
Same as the normal batch reactor, but allowing for three phase (gas+liquid+catalyst) chemistry. **The inlet and outlet, in this case, are for the gas phase only.**

5.4.6 CSTR (membrane)



Same as normal CSTR, but with a membrane separating the reactor into two compartments. Gas/liquid flows through each of them, each being at a different set pressure. Catalyst only on retentate side, while only bulk reactions possible on the (empty, void fraction = 1) permeate side. Different gas species can permeate through the membrane with a rate based on the values of their permeances. See the "membrane reactor mass transfer" section in this chapter for more details on the implementation.

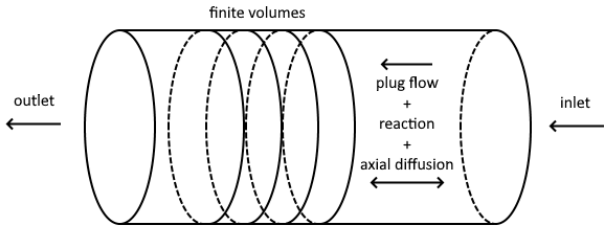
5.4.7 Plug flow reactor (PFR)



$$\frac{\partial C_i}{\partial t} = -v_x \frac{\partial C_i}{\partial x} + C^* \frac{1-\varepsilon}{\varepsilon} R_{i,cat} + R_{i,bulk}$$

A simple form of one-dimensional pipe reactor (catalytic packed bed). v_x is the gas velocity in the axial direction. For solving, the above equation is discretized to apply for a finite number of volumes (the number of which is set in the Reactor setup tab), forming a system of ordinary differential equations (ODEs). The velocity is assumed the same at any reactor point, and thus no radial changes in concentrations are observed, only axial ones. Velocity change can be applied here, as in the case of CSTR, in which case v_x is position dependent.

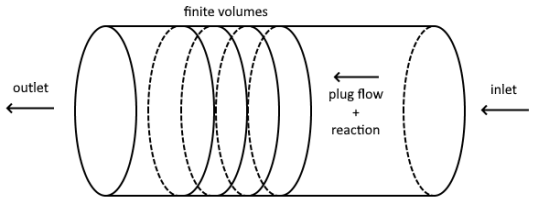
5.4.8 PFR (with diffusion)



$$\frac{\partial C_i}{\partial t} = -v_x \frac{\partial C_i}{\partial x} + \frac{D_i}{\tau}\frac{\partial^2 C_i}{\partial x^2} + C^* \frac{1-\varepsilon}{\varepsilon} R_{i,cat} + R_{i,bulk}$$

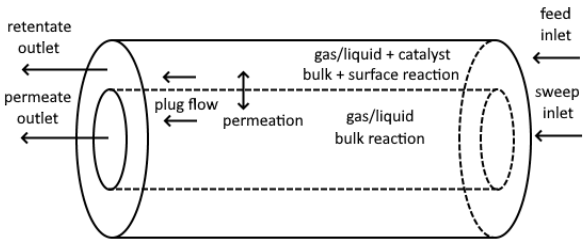
Exactly the same as above, but including also the diffusive/dispersive mass transport. *D* is the diffusion coefficient, which can be set in the Operating conditions tab. τ is the tortuosity factor due to a non-straight diffusion path through the catalytic bed.

5.4.9 PFR (fast)



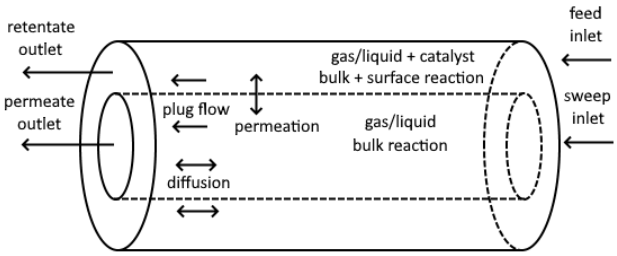
Same as normal PFR, with the difference that it is simulated literally by performing serial CSTR simulations for each reactor point. As such, it is best used only for the case where we are interested solely in the steady-state operation. It is generally faster than normal PFR simulation.

5.4.10 PFR (membrane)



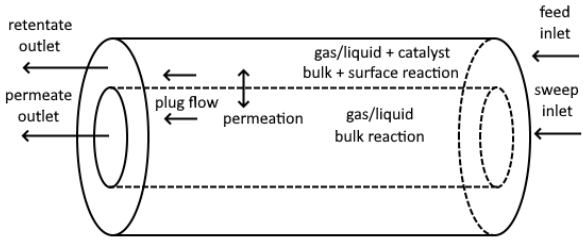
Same as PFR but with a membrane, implemented in a similar fashion as for CSTR (membrane). Please see the “membrane reactor mass transfer” section of this chapter for more details.

5.4.11 PFR (membrane with diffusion)



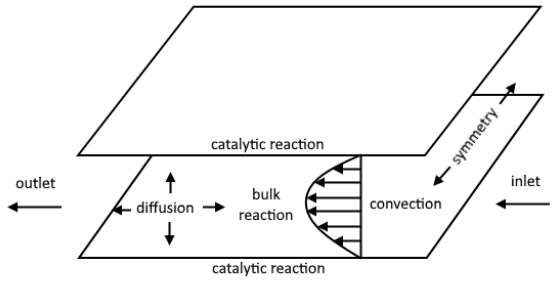
Same as PFR (membrane) but with additional axial dispersion/diffusion. The dispersion is taken into account on both sides of the membrane.

5.4.12 PFR (membrane - fast)



Same as PFR – membrane but implement as “fast” (see PFR – fast). This is especially useful for membrane systems where usually the “velocity change” input is required and analytical Jacobian generation with it enabled is no possible for normal PFR but is possible for PFR – fast. Please see the “membrane reactor mass transfer” section of this chapter for more details.

5.4.13 Parallel plates channel (2D)



$$bulk: \frac{\partial C_i}{\partial t} = -v_x(y)\frac{\partial C_i}{\partial x} + D_i\frac{\partial^2 C_i}{\partial x^2} + D_i\frac{\partial^2 C_i}{\partial y^2} + R_{i,bulk}$$

$$volumes\ next\ to\ walls: \frac{\partial C_i}{\partial t} = -v_x(y)\frac{\partial C_i}{\partial x} + D_i\frac{\partial^2 C_i}{\partial x^2} + D_i\frac{\partial^2 C_i}{\partial y^2} + C^*\frac{1-\varepsilon}{\varepsilon}R_{i,cat} + R_{i,bulk}$$

A slightly more complex reactor type, which is simulated in two dimensions, X (direction of flow) and Y (direction normal to the plates). The mass transport by the flow rate (convection) is in the X direction, while diffusion/dispersion can occur in either X, Y or both directions. The velocity

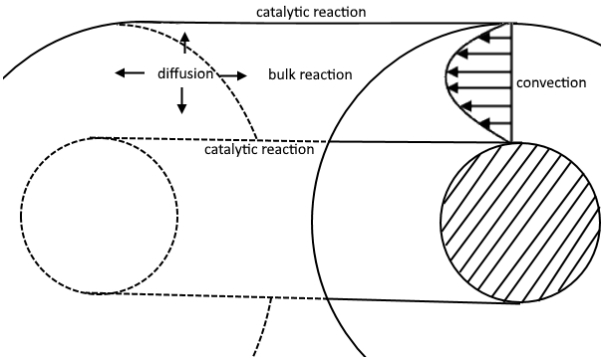
distribution between the plates is of convective shape, as is typical for laminar flow. In this reactor type, the **catalytic reaction is only possible on the walls** (one or both), while only bulk-bulk reactions can occur between the plates. The velocity profile is computed as (where $_{rel}$ denotes the relative values, and *S* is the cross section of each finite volume and *y* is the position in the reactor):

$$v_{x,rel} = 1 - (2y_{rel} - 1)^2$$

$$v_x = K v_{x,rel}$$

$$K = \frac{F_{tot}}{\frac{S}{n_y}\sum v_{x,rel}}$$

5.4.14 Round channel (2D)



Same as the parallel plates reactor, with the difference that the roundness is taken into account for the following aspects:

- Diffusion in radial (Y) direction considers the radial contribution
- Catalyst sites are distributed so that the outward (bigger) surface has more of them relative to the surface increase. In that regard, infinitely small layers of bulk gas/liquid by the surface should receive the same change in bulk due to reaction, but due to limited size of finite volumes the radial factor is taken into account.
- The gradually increasing finite volumes are taken into account for convective mass transport
- The velocity profile is different than a perfect concave due to the roundness (the maximum velocity is shifted towards the centre) – equation below

$$v_{x,rel} = \frac{ln\left(\frac{r}{R_I}\right)}{ln\left(\frac{R_O}{R_I}\right)}\left({R_O}^2 - {R_I}^2\right) - \left(r^2 - {R_I}^2\right)$$

5.5 Mass transport limitations and other phenomena

Normally, no mas transfer limitations are assumed, e. g. the concentration at the catalyst surface is the same as bulk concentration. Additional limitations and enhancements to the mass transfer rates can be incurred through the settings in the Reactor setup tab. They are described in this subchapter.

5.5.1 Mass transfer limitation from the bulk to the catalytic surface

When this is enabled, only simple adsorption/desorption reactions are permitted due to complexity issues of multiple Eley-Rideal type reactions. The requirements are thus that for each gas species, maximum one reaction with catalyst is permitted, and it needs to be of the form:

A + * <-> A*

Where A is the gas species in question. As such, we can derive a simple mass balances for such bulk species as follows (and as seen above):

$$\frac{dC}{dt} = R_{bulk} + R_{cat} + M$$

The general mass balance above describes the time derivative of concentration C of a gas species, and R_{bulk}, R_{cat} and M are bulk-bulk reaction, bulk-catalyst reaction and mass transport terms, respectively. For the mass transfer limitations of bulk to surface type, we are only interested in changing the R_{cat} term. Considering instant equilibrium, the molar flow rate to the surface via mass transfer must be equal to the surface reaction rate, leading to the following equation:

$$R_{cat} = -f_{site} rate_{f+b} = -f_{site} k_{for} C_s f_p \theta_* + f_{site} k_{back} \theta_{ads}$$

$$R_{cat} = -\frac{A_{cat} V_{cat} k_{MT}}{V_{void}} (C - C_s)$$

C_s is concentration of the species just by the surface of the catalyst, A_{cat} is the surface of the stationary film around the catalyst per catalyst volume, k_{MT} is the mass transfer coefficient and f_{site} is the $C^*(1-\varepsilon)/\varepsilon$ term described above.

$$-\frac{A_{cat} V_{cat} k_{MT}}{V_{void}} (C - C_s) = -f_{site} k_{for} C_s f_p \theta_* + f_{site} k_{back} \theta_{ads}$$

$$\frac{A_{cat} V_{cat} k_{MT}}{V_{void}} = E \quad ; \quad f_{site} k_{for} f_p = F \quad ; \quad f_{site} k_{back} = G$$

$$-E(C - C_s) = -F C_s \theta_* + G \theta_{ads}$$

$$-EC + C_s(F\theta_* + E) - G\theta_{ads} = 0$$

$$C_s = \frac{G\theta_{ads} + EC}{F\theta_* + E}$$

Therefore, we can write the final rate of the adsorption/desorption reaction as:

$$rate_{f+b} = k_{for} f_p \theta_* \frac{G\theta_{ads} + EC}{F\theta_* + E} - k_{back} \theta_{ads}$$

$$rate_{f+b} = k_{for} f_p \theta_* \frac{f_{site} k_{back} \theta_{ads} + EC}{f_{site} k_{for} f_p \theta_* + E} - k_{back} \theta_{ads}$$

When catalyst on walls option is used, the V_{cat} used to calculate area from A_{cat} is taken as reactor volume instead (instead of being ambiguously calculated from reactor dimensions).

$$E = \frac{A_{cat}V_{cat}k_{MT}}{V_{void}} = \frac{A_{cat}V_R(1-\varepsilon)k_{MT}}{V_R\varepsilon} = \frac{(1-\varepsilon)}{\varepsilon}A_{cat}k_{MT} \quad ; \quad E = A_{cat}k_{MT}$$

5.5.2 Internal mass transfer limitations

This phenomenon deals with the limitation of mass transfer into the catalytic particle, which can limit the conversions in case of fast reactions and relatively large particles. This is solved by a simple "Effectiveness factor" approach, which basically reduces the rates of bulk-catalytic reactions and is implementation-wise the same as reducing the parameter for the total available catalytic sites.

5.5.3 Velocity change

The following equations who how the model maintains constant pressure by implicitly assuming the velocity out of the reactor is such that the total concentration of gas species doesn't change, where R_i is the combined $R_{i,cat}$ and $R_{i,bulk}$. v_{factor} is basically equal to L/v_x or V/V'_x where L, V and V'_x are the finite element length, volume and volumetric flow rate, respectively.

$$\frac{dc_i}{dt} = R_i + v_{factor,inlet}C_{i,inlet} - v_{factor}c_i$$

$$v_{factor} = \frac{\sum_{j=1}^{n} R_j + v_{factor,inlet}C_{j,inlet}}{\sum_{j=1}^{n} c_j}$$

$$v_{factor} = \frac{\sum_{j=1}^{n} R_j + v_{factor,inlet}C_{j,inlet}}{\sum_{j=1}^{n} C_{j,inlet}} = \frac{\sum_{j=1}^{n} R_j + v_{factor,inlet}C_{j,inlet}}{C_{total,inlet}}$$

$$\frac{dc_i}{dt} = R_i + v_{factor,inlet}C_{i,inlet} - c_i\frac{\sum_{j=1}^{n} R_j + v_{factor,inlet}C_{j,inlet}}{C_{total,inlet}}$$

The assumption that the sum of all concentrations equals to the sum of inlet concentrations holds true for the model and the system of equations. The simplified equation with constant denominator term $C_{total,inlet}$ has an easy way to generate the analytical Jacobian of the system, and enables highly stable and efficient computing. The "Velocity change" parameter can highly increase the system's stiffness due to the coupling of all of the bulk species. Therefore, for systems with equimolar reactions or low conversions, consider disabling this option for faster computing. The comparison between the total concentration and expected total concentration, as well as inlet and outlet velocities can be plotted with the "Outlet" plot.

5.5.4 Three phase reactors mass transfer

The direction is considered from gas to liquid point of view, regarding the +/- signs.

Symbols and units:

He – Henry's constant [mol/(L bar)]

C – concentration [mol/L]

p – pressure [bar]

V – volume [L]

$\dot{N}$ – molar flow [mol/s]

t – time [s]

GLR – gas to liquid ratio [/]

A – interphase surface area [m^2]

k_{LTI} – mass transfer coefficient on the liquid side (bulk to interface) [m/s]

k_{GTI} – mass transfer coefficient on the gas side (bulk to interface) [m/s]

R – gas constant [L bar/(mol K)]

T – temperature [K]

i – interface

b – bulk

General equations:

$$He = \frac{C_{liquid,i}}{p_{gas,i}}$$

$$\frac{dC_{liquid,b}}{dt} = \frac{\dot{N}}{V_{liquid}} \quad ; \quad \frac{dC_{gas,b}}{dt} = -\frac{\dot{N}}{V_{gas}}$$

$$V_{liquid} = \frac{V_{reactor}\,\varepsilon}{1+GLR} \quad ; \quad V_{gas} = \frac{V_{reactor}\,\varepsilon}{1+\frac{1}{GLR}}$$

Gas to interface MT limitation

In this case, perfect mixing is assumed in the liquid phase.

$$\dot{N} = A\,k_{GTI}\left(C_{gas,b} - C_{gas,i}\right) = A\,k_{GTI}\left(C_{gas,b} - \frac{1}{He\,R\,T}C_{liquid,b}\right)$$

Liquid to interface MT limitation

In this case, perfect mixing is assumed in the gas phase.

$$\dot{N} = A\,k_{LTI}\left(C_{liquid,i} - C_{liquid,b}\right) = A\,k_{LTI}\left(He\,R\,T\,C_{gas,b} - C_{liquid,b}\right)$$

Both MT limitations

In this case, neither gas or liquid phase have perfect mixing.

$$\dot{N} = A\,k_{GTI}\left(C_{gas,b} - C_{gas,i}\right)$$

$$\dot{N} = -A\,k_{LTI}\left(C_{liquid,b} - C_{liquid,i}\right) = -A\,k_{LTI}\left(C_{liquid,b} - He\,R\,T\,C_{gas,i}\right)$$

$$\dot{N} = A\, k_{GTI} \left(C_{gas,b} \frac{He\, R\, T}{He\, R\, T + \frac{k_{GTI}}{k_{LTI}}} - C_{liquid,b} \frac{1}{He\, R\, T + \frac{k_{GTI}}{k_{LTI}}} \right)$$

No MT limitation

This variant is currently unsupported (instant equilibrium between the phases) due to implementation restrictions. Might be added later given enough requests. We suggest using a single MT limitation variant with high values in the meantime, or some other approach to reduce the unneeded coupling between different species.

5.5.5 Membrane reactor mass transfer

Generally, the membrane systems are comprised of two volumes, retentate and permeate sides, separated by a membrane. Both of the sides are treated as separate systems with their own pressure, flow rate as well as inlet and outlet gas concentrations. The systems are coupled only through mass transfer over the membrane, which is limited to a certain number of species (see chemistry section on membrane reactors). The mass transfer rate through the membrane for the coupled species is:

$$\dot{N}_{memb.} = A_{memb.} P_i \left(p_{i,ret} - p_{i,perm} \right) = A_{memb.} P_i RT \left(c_{i,ret} - c_{i,perm} \right)$$

where $N_{memb.}$ is the molar flow through the membrane in mol/s, $A_{memb.}$ is the membrane surface area in m^2 and P_i is the permeance of species *i* in mol/(m^2 Pa s) (units for the input file are mol/(m^2 Pa h)). In the case of liquid phase, concentration is used directly instead of pressure (in that case, input permeance is in units m/h)).

It is assumed for the CSTR type of the membrane reactor that both of the compartments are individually perfectly mixed, while for PFR types, each finite volume is perfectly mixed in its respective compartment. No model currently covers membrane transport in two dimensions (as in radial inhomogeneity) – partially due to much more complex flow profiles. The mass balances for bulk species for the CSTR reactor are shown below (while for PFR, a serial implementation can be conceptually assumed):

$$Retentate\ species: \frac{dC_i}{dt} = \frac{V_{ret}\varepsilon}{F_{ret,in}} C_{i,inlet} + C^* \frac{1-\varepsilon}{\varepsilon} R_{i,cat} + R_{i,bulk} - \frac{V_{ret}\varepsilon}{F_{ret,out}} C_i - \frac{\dot{N}_{memb.}}{V_{ret}\varepsilon}$$

$$Permeate\ species: \frac{dC_i}{dt} = \frac{V_{perm}}{F_{perm,in}} C_{i,inlet} + R_{i,bulk} - \frac{V_{perm}}{F_{perm,out}} C_i + \frac{\dot{N}_{memb.}}{V_{perm}}$$

The volumes for each side are computed from the input provided *RPR* (retentate/permeate volume ratio) as such:

$$V_{ret} = RPR\, V_{perm} = \frac{V_{reactor}}{\left(1 + \frac{1}{RPR}\right)}$$

The void fraction in this case only applies to retentate side, as it is assumed the catalyst is only present there while the permeate side has a void fraction of 1. Furthermore, it is assumed that the

available membrane surface area is not affected by the void fraction, that is, regardless of the void fraction; the input membrane surface area is taken with its unchanged value. If users wish to simulate coverage of the membrane surface with catalytic particles, it is suggested to just reduce the membrane surface area parameter accordingly. For the inlet gas concentration, the same operating conditions parameter is used, but the array is divided based on the membrane side and normalized separately. The liquid inlet concentrations are taken as they are.

Due to the nature of the membrane transport, it is highly recommended to select the “Velocity change” parameter. This will assure that the pressure always stays at its set value for both sides of the membrane, and only the gas flow rate changes. Otherwise, constant outlet flow rate is assumed, which will in most cases lead to unexpected pressures. The implementation of velocity change is the same as for reactor types – outlet flow rate is assumed such that the sum of the bulk species’ concentration time derivative is 0 (separately for each side of the membrane).

6 Used libraries and programs

CERRES uses a variety of different open-source libraries. We are highly thankful to the developers of these projects. The libraries are listed below, and their licenses are included in Appendix 1.

Python (https://www.python.org/)

Sundials (https://computing.llnl.gov/projects/sundials)

NumPy (https://numpy.org/)

SciPy (https://www.scipy.org/)

Matplotlib (https://matplotlib.org/)

OpenMP (https://www.openmp.org/)

PyInstaller (https://www.pyinstaller.org/index.html)

Inno Setup (https://jrsoftware.org/isinfo.php)

In addition, CERRES relies on the GNU GCC compiler (https://gcc.gnu.org/) as an external program for dynamically-generated code compilation. The compiler is included in the MinGW-w64 distribution (http://mingw-w64.org/doku.php), which is bundled in the CERRES installer and installed automatically under the "<CERRES installation folder>/cvode_execution/mingw_libs/mingw" path. CERRES invokes this program via command-line arguments, and as such the GNU GCC compiler is not a part of CERRES but a separate program, having its own GNU GNU GPLv3+ license, as displayed below. Due to the license requirements, we provide the MinGW-w64 source code in the public CERRES repository (https://github.com/DamjanLasicJurkovic/CERRES_public) via a zipped file ("mingw-w64-v7.0.0.zip"). Users that already have a GCC compiler installed on their system, can likely remove this folder from the CERRES installation folder to save space, the requirements being that the GCC **targets w64 architecture** and is on the **system PATH**. Note that this **has not been thoroughly tested**, so it is advised to back-up the bundled files, run the CERRES test from the GUI, and finally delete them only after confirming that everything works correctly.

Only to be used for automatic file generation or such tasks, most users should be better off using the UI editors. Before writing any scripts for file generation, we encourage the users to first open and check the format of the example files.

7 Input files format (advanced)

7.1 File format, parsing and sections

All of the CERRES files are comprised of sections. Each section is a part of text surrounded by token lines. The section begins on the first line after the token line, and contains all lines up to the section end line. The section start line begins with $_token_$, where _token_ represents any of the token words, as described below. The section end line begins with end.

The files are parsed in the following way. The file is first split into lines, then all token lines are identified. All sections are extracted, and the rest of the file is ignored. The sections are then parsed appropriately based on the section name and content. Any extra sections that are not used by the file format, for example $reactions$ in input parameters file, are ignored. Any missing sections result in an error.

7.2 List of the used section tokens

7.2.1 General

These are the tokens used by all input file types, but they can be omitted if not needed.

- $title$ - title of the file (limited to one line)
- $author$ - author of the file (limited to one line)
- $description$ - description of the file
- $references$ - any important literature references connected to the file
- $version$ - CERRES version, for back-compatibility implementation in cases of changing formats

7.2.2 Input parameters files

- $parameter_list$ - list of input parameters
    - In form of lines of “parameter_family.parameter_name = parameter_value”
    - EXAMPLE: operating_conditions.temperature_list_C = [100,200,300]
    - Can have comments on lines starting with # within the section

7.2.3 Chemistry files

- $species_gas$ - list of gas bulk species
- $species_liquid$ - list of liquid bulk species
- $species_adsorbed$ - list of species adsorbed on the surface of the catalyst
- $species_sites$ - list of catalytic sites
- $reactions$ - the list of reactions, constants, and potential user-defined rate expressions
- $user_variables$ - the list of any additional variables used in the user-defined rate expressions. This is used as an extra check to avoid user errors.
- k_mode - mode of kinetic constants (direct k, Arrhenius, modified Arrhenius)
- $sites_mode$ - separated or mixed

7.2.4 Experimental data files

- nr_exp - number of experiments, used for consistency check

- exp_data_type - type of data, either “steady_state” or “transient”
- $measured_species$ - list of all the species which were measured, used for consistency check
- $defined_parameters$ - which parameters are to be used from exp data file instead of the inputs, used for consistency check
    - Can only be the ones from “operating_conditions” list in inputs
    - Have to use internal names as in “all_parameter_file.csv” file (see below), for example, “temperature_list_C”
- $conditions$ - the operating conditions for each of the experiments is listed here
    - Nr_exp+1 rows, first is header
    - Nr_defined_parameters columns
    - Have to be the same as described above
    - Always ordered from exp 1 to exp N, even though diff order for example in transient data
    - Need to have the same format as inputs with one less nested array
    - (the inlet concentrations are therefore defined here)
    - (The simulation time can be adjusted here for each exp if transient and diff time scales)
- $measurements$ - the measured concentrations are listed here
    - Different whether transient or steady state
    - Units: partial concentrations for gas (dimensionless, won’t be normalized), mol/L for liquid, surface fraction for adsorbed/sites (dimensionless)
    - Steady state
        - Nr_exp+1 rows, first is header
        - Nr_measured_species columns
        - Concentrations as inputs with one less nested array level
    - Transient
        - Multiple subsections, starting line with exp_# (exp_1, exp_33)
        - Nr_t+1 rows, first is header
        - Can have different numbers of measurements and time scales for each experiment
        - Experiment numbers have to be from 1 to N, will be sorted on input
        - Nr_measured_species +1 columns, first is time
        - **Time is in seconds!!!**

7.3 Complete list of CERRES inputs

The complete list of CERRES inputs (if needed for programmed file generation purposes) can be found in the CERRES installation folder, under “input_files/par_descriptor_files/all_parameter_file.csv”. Note that this file is used by CERRES for various checks and GUI generation, so note that editing this file will most likely break installation.

8 References/literature

[1] E. Jones, E. Oliphant, P. Peterson, SciPy: Open Source Scientific Tools for Python, (2001). http://www.scipy.org/ (accessed January 1, 2019).

[2] S.D. Cohen, A.C. Hindmarsh, CVODE, a stiff/nonstiff ODE solver in C, Comput. Phys. 10 (1996) 138–143. doi:10.1063/1.4822377.

[3] A.C. Hindmarsh, P. Brown, K.E. Grant, S.L. Lee, R. Serban, D.E. Shumaker, C. Woodward, SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers, ACM Trans. Math. Softw. 31 (2005) 363–396. doi:10.1145/1089014.1089020.

[4] J.D. Hunter, Matplotlib: A 2D graphics environment, Comput. Sci. Eng. 9 (2007) 99–104. doi:10.1109/MCSE.2007.55.

9 Appendix 1: Licenses of the used libraries

CERRES uses a variety of different open-source libraries. We are highly thankful to the developers of these projects. The libraries are listed below, and their licenses are included in Appendix 1.

Python (https://www.python.org/)

License:

“

A. HISTORY OF THE SOFTWARE

==========

Python was created in the early 1990s by Guido van Rossum at Stichting Mathematisch Centrum (CWI, see http://www.cwi.nl) in the Netherlands as a successor of a language called ABC. Guido remains Python's principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for National Research Initiatives (CNRI, see http://www.cnri.reston.va.us) in Reston, Virginia where he released several versions of the software.

In May 2000, Guido and the Python core development team moved to BeOpen.com to form the BeOpen PythonLabs team. In October of the same year, the PythonLabs team moved to Digital Creations, which became Zope Corporation. In 2001, the Python Software Foundation (PSF, see https://www.python.org/psf/) was formed, a non-profit organization created specifically to own Python-related Intellectual Property. Zope Corporation was a sponsoring member of the PSF.

All Python releases are Open Source (see http://www.opensource.org for the Open Source Definition). Historically, most, but not all, Python releases have also been GPL-compatible; the table below summarizes the various releases.

Release	Derived from	Year	Owner	GPL-compatible? (1)

0.9.0 thru 1.2		1991-1995	CWI	yes
1.3 thru 1.5.2	1.2	1995-1999	CNRI	yes
1.6	1.5.2	2000	CNRI	no
2.0	1.6	2000	BeOpen.com	no
1.6.1	1.6	2001	CNRI	yes (2)
2.1	2.0+1.6.1	2001	PSF	no
2.0.1	2.0+1.6.1	2001	PSF	yes
2.1.1	2.1+2.0.1	2001	PSF	yes
2.1.2	2.1.1	2002	PSF	yes
2.1.3	2.1.2	2002	PSF	yes
2.2 and above	2.1.1	2001-now	PSF	yes

Footnotes:

(1) GPL-compatible doesn't mean that we're distributing Python under the GPL. All Python licenses, unlike the GPL, let you distribute a modified version without making your changes open source. The GPL-compatible licenses make it possible to combine Python with other software that is released under the GPL; the others don't.

(2) According to Richard Stallman, 1.6.1 is not GPL-compatible, because its license has a choice of law clause. According to CNRI, however, Stallman's lawyer has told CNRI's lawyer that 1.6.1 is "not incompatible" with the GPL.

Thanks to the many outside volunteers who have worked under Guido's direction to make these releases possible.

B. TERMS AND CONDITIONS FOR ACCESSING OR OTHERWISE USING PYTHON

PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using this software ("Python") in source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python alone or in any derivative version,

provided, however, that PSF's License Agreement and PSF's notice of copyright,
i.e., "Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020 Python Software Foundation;
All Rights Reserved" are retained in Python alone or in any derivative version
prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on
or incorporates Python or any part thereof, and wants to make
the derivative work available to others as provided herein, then
Licensee hereby agrees to include in any such work a brief summary of
the changes made to Python.

4. PSF is making Python available to Licensee on an "AS IS"
basis. PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON
FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS
A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON,
OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any
relationship of agency, partnership, or joint venture between PSF and
Licensee. This License Agreement does not grant permission to use PSF
trademarks or trade name in a trademark sense to endorse or promote
products or services of Licensee, or any third party.

8. By copying, installing or otherwise using Python, Licensee
agrees to be bound by the terms and conditions of this License
Agreement.

BEOPEN.COM LICENSE AGREEMENT FOR PYTHON 2.0

BEOPEN PYTHON OPEN SOURCE LICENSE AGREEMENT VERSION 1

1. This LICENSE AGREEMENT is between BeOpen.com ("BeOpen"), having an

office at 160 Saratoga Avenue, Santa Clara, CA 95051, and the
Individual or Organization ("Licensee") accessing and otherwise using
this software in source or binary form and its associated
documentation ("the Software").

2. Subject to the terms and conditions of this BeOpen Python License
Agreement, BeOpen hereby grants Licensee a non-exclusive,
royalty-free, world-wide license to reproduce, analyze, test, perform
and/or display publicly, prepare derivative works, distribute, and
otherwise use the Software alone or in any derivative version,
provided, however, that the BeOpen Python License is retained in the
Software, alone or in any derivative version prepared by Licensee.

3. BeOpen is making the Software available to Licensee on an "AS IS"
basis. BEOPEN MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, BEOPEN MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

4. BEOPEN SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE
SOFTWARE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS
AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE, OR ANY
DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

5. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

6. This License Agreement shall be governed by and interpreted in all
respects by the law of the State of California, excluding conflict of
law provisions. Nothing in this License Agreement shall be deemed to
create any relationship of agency, partnership, or joint venture
between BeOpen and Licensee. This License Agreement does not grant
permission to use BeOpen trademarks or trade names in a trademark
sense to endorse or promote products or services of Licensee, or any
third party. As an exception, the "BeOpen Python" logos available at
http://www.pythonlabs.com/logos.html may be used according to the
permissions granted on that web page.

7. By copying, installing or otherwise using the software, Licensee
agrees to be bound by the terms and conditions of this License
Agreement.

CNRI LICENSE AGREEMENT FOR PYTHON 1.6.1

---

1. This LICENSE AGREEMENT is between the Corporation for National Research Initiatives, having an office at 1895 Preston White Drive, Reston, VA 20191 ("CNRI"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 1.6.1 software in source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, CNRI hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 1.6.1 alone or in any derivative version, provided, however, that CNRI's License Agreement and CNRI's notice of copyright, i.e., "Copyright (c) 1995-2001 Corporation for National Research Initiatives; All Rights Reserved" are retained in Python 1.6.1 alone or in any derivative version prepared by Licensee. Alternately, in lieu of CNRI's License Agreement, Licensee may substitute the following text (omitting the quotes): "Python 1.6.1 is made available subject to the terms and conditions in CNRI's License Agreement. This Agreement together with Python 1.6.1 may be located on the Internet using the following unique, persistent identifier (known as a handle): 1895.22/1013. This Agreement may also be obtained from a proxy server on the Internet using the following URL: http://hdl.handle.net/1895.22/1013".

3. In the event Licensee prepares a derivative work that is based on or incorporates Python 1.6.1 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 1.6.1.

4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS" basis. CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

7. This License Agreement shall be governed by the federal intellectual property law of the United States, including without limitation the federal copyright law, and, to the extent such U.S. federal law does not apply, by the law of the Commonwealth of Virginia, excluding Virginia's conflict of law provisions. Notwithstanding the foregoing, with regard to derivative works based on Python 1.6.1 that incorporate non-separable material that was previously distributed under the GNU General Public License (GPL), the law of the Commonwealth of Virginia shall govern this License Agreement only as to issues arising under or with respect to Paragraphs 4, 5, and 7 of this License Agreement. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between CNRI and Licensee. This License Agreement does not grant permission to use CNRI trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

8. By clicking on the "ACCEPT" button where indicated, or by copying, installing or otherwise using Python 1.6.1, Licensee agrees to be bound by the terms and conditions of this License Agreement.

ACCEPT

CWI LICENSE AGREEMENT FOR PYTHON 0.9.0 THROUGH 1.2

Copyright (c) 1991 - 1995, Stichting Mathematisch Centrum Amsterdam, The Netherlands. All rights reserved.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Stichting Mathematisch Centrum or CWI not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

STICHTING MATHEMATISCH CENTRUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND

FITNESS, IN NO EVENT SHALL STICHTING MATHEMATISCH CENTRUM BE LIABLE

FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES

WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN

ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT

OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

“

Sundials (https://computing.llnl.gov/projects/sundials)

License:

“

All SUNDIALS packages are licensed under the BSD 3-Clause and are subject to the following Copyright notice. The text from this page is included as part of the SUNDIALS source code in the files LICENSE and NOTICE.

BSD 3-Clause License Copyright (c) 2002-2019, Lawrence Livermore National Security and Southern Methodist University. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. * Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This work was produced under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This work was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

“

NumPy (https://numpy.org/)

License:

“

Copyright (c) 2005, NumPy Developers

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the NumPy Developers nor the names of any contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

“

SciPy (https://www.scipy.org/)

License:

“

Copyright © 2001, 2002 Enthought, Inc.

All rights reserved.

Copyright © 2003-2019 SciPy Developers.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Enthought nor the names of the SciPy Developers may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

“

Matplotlib (https://matplotlib.org/)

License:

“

```
License agreement for matplotlib versions 1.3.0 and later
=========================================================

1. This LICENSE AGREEMENT is between the Matplotlib Development Team
("MDT"), and the Individual or Organization ("Licensee") accessing and
otherwise using matplotlib software in source or binary form and its
associated documentation.

2. Subject to the terms and conditions of this License Agreement, MDT
hereby grants Licensee a nonexclusive, royalty-free, world-wide license
to reproduce, analyze, test, perform and/or display publicly, prepare
derivative works, distribute, and otherwise use matplotlib
alone or in any derivative version, provided, however, that MDT's
License Agreement and MDT's notice of copyright, i.e., "Copyright (c)
2012- Matplotlib Development Team; All Rights Reserved" are retained in
matplotlib  alone or in any derivative version prepared by
Licensee.

3. In the event Licensee prepares a derivative work that is based on or
incorporates matplotlib or any part thereof, and wants to
make the derivative work available to others as provided herein, then
Licensee hereby agrees to include in any such work a brief summary of
the changes made to matplotlib .

4. MDT is making matplotlib available to Licensee on an "AS
IS" basis.  MDT MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED.  BY WAY OF EXAMPLE, BUT NOT LIMITATION, MDT MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF MATPLOTLIB
WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. MDT SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF MATPLOTLIB
 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR
LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING
MATPLOTLIB , OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF
THE POSSIBILITY THEREOF.
```

6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between MDT and Licensee. This License Agreement does not grant permission to use MDT trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

8. By copying, installing or otherwise using matplotlib , Licensee agrees to be bound by the terms and conditions of this License Agreement.

License agreement for matplotlib versions prior to 1.3.0

==========

1. This LICENSE AGREEMENT is between John D. Hunter ("JDH"), and the Individual or Organization ("Licensee") accessing and otherwise using matplotlib software in source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, JDH hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use matplotlib alone or in any derivative version, provided, however, that JDH's License Agreement and JDH's notice of copyright, i.e., "Copyright (c) 2002-2011 John D. Hunter; All Rights Reserved" are retained in matplotlib alone or in any derivative version prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on or incorporates matplotlib or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to matplotlib.

4. JDH is making matplotlib available to Licensee on an "AS IS" basis. JDH MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, JDH MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS

FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF MATPLOTLIB
WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. JDH SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF MATPLOTLIB
FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR
LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING
MATPLOTLIB , OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF
THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any
relationship of agency, partnership, or joint venture between JDH and
Licensee. This License Agreement does not grant permission to use JDH
trademarks or trade name in a trademark sense to endorse or promote
products or services of Licensee, or any third party.

8. By copying, installing or otherwise using matplotlib,
Licensee agrees to be bound by the terms and conditions of this License
Agreement.

“

OpenMP (https://www.openmp.org/)

(Implementation in the GNU GCC compiler - MinGW-w64 distribution, license covered below)

PyInstaller (https://www.pyinstaller.org/index.html)

Licensed under GPL, but only used for building (exception in the license allowing that).

See more: https://www.pyinstaller.org/license.html

Inno Setup (https://jrsoftware.org/isinfo.php)

License:

“

Inno Setup License

==========

Except where otherwise noted, all of the documentation and software included in the Inno Setup package is copyrighted by Jordan Russell.

Copyright (C) 1997-2020 Jordan Russell. All rights reserved.

Portions Copyright (C) 2000-2020 Martijn Laan. All rights reserved.

This software is provided "as-is," without any express or implied warranty. In no event shall the author be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter and redistribute it, provided that the following conditions are met:

1. All redistributions of source code files must retain all copyright notices that are currently in place, and this list of conditions without modification.

2. All redistributions in binary form must retain all occurrences of the above copyright notice and web site addresses that are currently in place (for example, in the About boxes).

3. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software to distribute a product, an acknowledgment in the product documentation would be appreciated but is not required.

4. Modified versions in source or binary form must be plainly marked as such, and must not be misrepresented as being the original software.

Jordan Russell

jr-2020 AT jrsoftware.org

https://jrsoftware.org/

“

In addition, CERRES relies on the GNU GCC compiler (https://gcc.gnu.org/) as an external program for dynamically-generated code compilation. The compiler is included in the MinGW-w64 distribution (http://mingw-w64.org/doku.php), which is bundled in the CERRES installer and installed automatically under the "<CERRES installation folder>/cvode_execution/mingw_libs/mingw" path. CERRES invokes this program via command-line arguments, and as such the GNU GCC compiler is not a part of CERRES but a separate program, having its own GNU GNU GPLv3+ license, as displayed below. Users that already have a GCC compiler installed on their system, can likely remove this folder from the CERRES installation folder to save space, the requirements being that the GCC **targets w64 architecture** and is on the **system PATH**. Note that this **has not been thoroughly tested**, so it is advised to back-up the bundled files, run the CERRES test from the GUI, and finally delete them only after confirming that everything works correctly. The MinGW-w64 license is included below:

"

```
MinGW-w64 licensing

******************

The copyright and license notices have been divided in two files:

The notices in COPYING.MinGW-w64.txt (this file) apply only to

MinGW-w64 itself. These don't apply to the binaries built with

MinGW-w64 unless you specifically tell MinGW-w64 to link against

these parts, for example, by enabling profiling code.

In addition to the notices in this file, also the notices in

COPYING.MinGW-w64-runtime.txt apply to MinGW-w64. Some (possibly

all) notices in that file may apply also to the binaries built with

this version of MinGW-w64. The idea is that if you create binary

packages of your software with MinGW-w64, you can simply copy

COPYING.MinGW-w64-runtime.txt into your package to fulfill the

license requirements of the MinGW runtime.

If you think that not all notices apply to your package and want to

remove some of them, note that, for example, the gdtoa files always

get linked in if you use any printf-like function. So usually it is

easiest and safest to just keep all the notices.

====================

GCC and GNU binutils

====================

Copyright (C) Free Software Foundation
```

License: GNU GPLv3+ (see the file COPYING.GPLv3)

==========

Profiling code

==========

Copyright 1998, 1999, 2000, 2001, 2002 Red Hat, Inc.

License: GNU GPLv2+ (see the file COPYING.GPLv2)

* * * * * * *

Copyright (c) 1982, 1983, 1986, 1992, 1993

The Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==========

DirectX and DDK headers

==========

DirectX and DDK headers are under GNU LGPLv2.1+ (see the file COPYING.LGPLv2.1) and copyrighted by various people. Using these headers doesn't make LGPLv2.1 apply to your code, because these headers files contain only data structure definitions, short macros, and short inline functions. Here is the relevant part from LGPLv2.1 section 5 paragraph 4:

If such an object file uses only numerical parameters, data
structure layouts and accessors, and small macros and small
inline functions (ten lines or less in length), then the use
of the object file is unrestricted, regardless of whether it
is legally a derivative work.

==========

libmangle and gendef

==========

Copyright (c) 2009 mingw-w64 project

Contributing authors: Kai Tietz, Jonathan Yong

Permission is hereby granted, free of charge, to any person obtaining a
copy of this software and associated documentation files (the "Software"),
to deal in the Software without restriction, including without limitation
the rights to use, copy, modify, merge, publish, distribute, sublicense,
and/or sell copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.

====

PSEH

====

Copyright (c) 2004-2008 KJK::Hyperion

Permission is hereby granted, free of charge, to any person obtaining a
copy of this software and associated documentation files (the "Software"),
to deal in the Software without restriction, including without limitation
the rights to use, copy, modify, merge, publish, distribute, sublicense,
and/or sell copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING

FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER

DEALINGS IN THE SOFTWARE.

“