Chemical Kinetics Solver - Documentation

Detlev Conrad Mielczarek

December 3, 2017

Contents

1 Compiling the Code

The code is developed using Eclipse CDT. Initially, it was developed on both Windows (MinGW) and Linux (GCC), however with the introduction of ODEPACK, Windows is no longer tested. I.e. it should work if you can mirror the Linux compilation steps, however you, the user, are on your own.

1.1 Dependencies

To compile the code, the following is required:

i) a C++ compiler with at least C++11 support

ii) a Fortran compiler (to build the ODEPACK libraries/archives)

iii) a copy of the Intel ODE library

iv) an IDE to build, though one could write makefiles/compile commands etc. by hand.

odepack In October 2017, the code was expanded to support an additional ODE solver, namely ODEPACK from the LLNL. The specific solver that was implemented is LSODA which has been used to solve problems in chemical kinetics in the past.

For the sake of simplicity, ODEPACK can be compiled into an archive from which the desired functions are then called. While different optimisation flags can be set, it appears that performance is similar, at least within my limited testing. One suggestion for compiling ODEPACK into an archive is given below:

```
gfortran -O3 -march=native -c opkda1.f opkda2.f opkdmain.f
ar -rcs libodepack.a opkda1.o opkda2.o opkdmain.o
```

The archive can then be added to the linker like any standard library.

1.2 Compilation

The recommended method of compiling the code is to import the project into eclipse CDT. Tell the compiler to look for the headers in the headers folder and add the relevant libraries in the linked. It is (or was) important that `-march=native` is selected in the optimisation field to ensure the best possible performance. This flag will enable platform specific compiler optimisations and will therefore impact portability of the code.

2 Running the Code

Running the code is very simple and straightforward. The binary is called from the command line, followed by the name of the chemkin-style mechanism file and the name of the file containing the initial conditions. A word of warning: The code was developed for work with liquid phase mechanisms developed with RMG and is tested with these. Thus third bodies or pressure dependent reactions are currently **NOT** supported and undefined behaviour may occur. The code will by default write a logfile in which information on the run is reported. This output can be displayed in the terminal by placing an empty file called `debug` in the folder from which the code is called.

2.1 input file layout

The input file takes some inspiration from html in that specific settings are grouped together in begin and end tags. This helps maintaining shorter, clearer keywords while also writing clearer and more robust code.

Technically, the only required input are the initial conditions as well as species concentrations. All other settings can be considered to be optional. Assuming the mechanism to be solved contains oxygen and dodecane, whose behaviour is of interest for 100 s in 1 s time intervals at 298.15 K, a minimum input may look as follows:

```
<Initial Conditions>
Temperature 298.15    ! in Kelvin
EndTime 1e2 1e0
</Initial Conditions>

<Species>
O2 0.002
Dodecane 4.7
</Species>
```

Please note that comments within the individual input blocks are not officially supported and may lead to undefined behaviour.

Default parameters are chosen for tolerances, while time stepping is automated in LSODA from odepack. While the code was initially developed using the Intel ODE solver, odepack has now been chosen as the default solver due to its speed. This does however include a stability penalty, e.g. it may be required to tweak both the time step as well as tolerance for LSODA. However, this reduction in stability is worth the benefit of being able to handle large mechanisms. In addition, with LSODA, the full analytical Jacobian should not be used, as an internally generated numerical Jacobian is more efficient. This is most likely due to the sparse nature of the Jacobian which is not taken into consideration in the analytical form.

3 List of All Possible Input Options

The following is a list of all input commands - with no guarantees regarding completeness or accuracy (as the code still develops). The list is compiled with the best intention, however the final authority on possible options is the code itself.

For reasons of ease of coding, extendibility, flexibility and robustness, the input has been organised in thematic sections. These are reproduced here in the documentation.

Solver Parameters The following solver parameters are valid input options. For all parameters present therein, default parameters are available. These should be suitable for most applications and thus the entire block can be omitted if the default parameters are to be used.

```
<Solver Parameters>
odepack   ! default
intelode   ! default false
Jacobian   ! default false
Force Stability   ! default false
Use General Solver   ! default false
hm   ! Intel ODE only
initialh   ! Intel ODE only
RTOL   ! relative tolerance
ATOL   ! absolute tolerance
IRREV   ! default false, auto true when required
Print Rates   ! default false
Stoichiometry Matrix For Opt   ! default false
</Solver Parameters>
```

Initial Conditions Initial conditions are essential and **must** be given by the user a default temperature of 298.15 K is set in the code if none is given, but this should not be relied on.

```
<Initial Conditions>
Temperature 298.15   ! in Kelvin
EndTime 1e3 1e1   ! end time and time step
</Initial Conditions>
```

(Initial) Species Initial species concentrations are essential and **must** be given by the user in mol/L units.

```
<Species>
!Species Name - followed by concentration
O2 0.002
Optional:
!Species Name - followed by concentration - followed by
    ConstantConcentration
O2 0.002 ConstantConcentration
</Species>
```

Mechanism Reduction The code offers three methods for the reduction of the reaction mechanism.
A first step is the lumping or grouping of species. This is enabled automatically when a file detailing the species lumping is present. In this case, the user can chose which lumping method is used.
The second step is using the rates to determine which reactions are important and removing unneeded reactions. This method required the mechanism to be solved in full once. The user is required to specify how severe the reaction removal is to be. The keyword for lumping consists of a slow and fast flag, as well as the lumping type. Slow and fast as well as the type may be omitted for defaults.

```
<Mechanism Reduction>
Slow Lumping Type 1
Slow Lumping Type 2
Lumping Type 3
Fast Lumping Type 1
Fast Lumping Type 2
ReduceReactions
</Mechanism Reduction>
```

Analysis The code has some built in analysis functions. These should work - future work may be required here...

```
<Analysis>
RatesMaxAnalysis
StreamRatesAnalysis
RatesSpeciesAllAnalysis
RatesAnalysisAtTime - followed by times (sorted)
RatesOfSpecies - follwoed by species names
</Analysis>
```

Pressure Vessel The code contains functionality to include an oxygen reservoir as found in the PetroOxy. This section may need re-evaluation/configuration.

```
<PressureVessel>
Sample Size=
Vessel Size=
Initial Pressure=
Maximum Pressure=
Gas Solubility=
Gas Species=
PetroOxy Temperature Rise=

Experimental: Henry Law Diffusion Limit
</PressureVessel>\\
```