The chemkin User Manual

Jane Huang, Kimia Mavon, Weidong Xu, Zeyu Zhao

1 Introduction

`chemkin` is a Python 3 library that computes the reaction rates of all species participating in a system of elementary, irreversible reactions.

1.1 Key chemical concepts and terminology

A system consisting of M elementary reactions involving N species has the general form

$$\sum_{i=1}^{N} \nu'_{ij} \mathcal{S}_i \longrightarrow \sum_{i=1}^{N} \nu''_{ij} \mathcal{S}_i, \qquad j = 1, \ldots, M. \tag{1}$$

S_i is the ith specie in the system, ν'_{ij} is its stoichiometric coefficient (dimensionless) on the reactants side of the jth reaction, and ν''_{ij} is its stoichiometric coefficient (dimensionless) on the product side for the jth reaction.

Each specie is characterized by a concentration x_i, in units of [mol/vol]. The reaction rate of each specie is the time rate of change of its concentration, $\frac{dx_i}{dt}$. The reaction rate is usually represented by the symbol f_i, such that

$$f_i = \sum_{j=1}^{M} (\nu''_{ij} - \nu'_{ij}) \omega_j = \sum_{j=1}^{M} \nu_{ij} \omega_j, \qquad i = 1, \ldots, N. \tag{2}$$

ω_j is the progress rate of the jth reaction,

$$\omega_j = k_j \prod_{i=1}^{N} x_i^{\nu'_{ij}}, \qquad j = 1, \ldots, M. \tag{3}$$

The forward reaction rate coefficient k_j for each reaction is assumed to take one of three possible forms:

1. $k =$ constant
2. Arrhenius: $k = A \exp(-\frac{E}{RT})$, where A is the pre-factor, E is the activation energy, R is the universal gas constant, and T is the temperature.
3. Modified Arrhenius: $k = AT^b \exp(-\frac{E}{RT})$, where A is the pre-factor, E is the activation energy, R is the universal gas constant, T is the temperature, and b is the temperature scaling parameter.

1.2 Features

The package can solve for the reaction rates of a system of elementary, irreversible reactions. The number of reactions and species is arbitrary. For each system of reactions, the user supplies the species participating in the reactions, the chemical equations, the stoichiometric coefficients for the reactants and products, and the rate coefficient parameters (e.g., E and A for Arrhenius rates). For a given system, the user can then specify a temperature and a vector of species concentrations in order to return the reaction rates in the form of a NumPy array. Rate coefficients and reaction progress rates can also be retrieved.

2 Installation

2.1 Where to find and download the code

The package is freely available at `https://github.com/cs207group4/cs207-FinalProject`. If you have a GitHub account, you can simply open your terminal and type
`git clone git@github.com:cs207group4/cs207-FinalProject.git`

Otherwise, you can download the package by clicking on the green button in the upper right corner of the page that says "Clone or download," then click "Download ZIP" to download the entire repository as a ZIP file.

If you just want to use the library, you can download the single library file at `https://raw.githubusercontent.com/cs207group4/cs207-FinalProject/master/chemkin.py`

2.2 Dependencies

This package has dependencies that usually come standard with the Anaconda distribution, but are straightforward to install if you do not have them in your Python distribution. (Click on the individual package names to be directed to their website and installation instructions).

- NumPy v.1.11+

2.3 Running the test suite

In order to run the test suite, you need to have pytest v. 3.00+ and pytest-cov v.2.5+ installed. Once you download the contents of the repository from GitHub, enter the directory and type `pytest` into the terminal. The results of the test code will be printed out to the terminal.

3 Basic Usage and Examples

As an example of basic code usage, we consider the following system of elementary, irreversible reactions:

1. $H + O_2 \xrightarrow{k_1} OH + O$
2. $H_2 + O \xrightarrow{k_2} OH + H$
3. $H_2 + OH \xrightarrow{k_3} H_2O + H$

Reaction 1 has an Arrhenius rate coefficient with $A = 3.52 \times 10^{10}$ and $E = 7.14 \times 10^4$. Reaction 2 has a modified Arrhenius rate coefficient with $A = 5.06 \times 10^{-2}$, $b = 2.7$, and $E = 2.63 \times 10^4$. Finally, reaction 3 has a constant rate coefficient of $k_3 = 10^3$.

3.1 Input format

In an xml input file, the user provides the species participating in the reactions, the chemical equations, the stoichiometric coefficients, and the rate coefficient parameters. See `rxns.xml` in the repository folder for an example of how to format the input file.

The xml file will be processed and stored in a `chemkin` object as follows:

```
>>>from chemkin import *
>>>rxn_system = chemkin.from_xml('rxns.xml')
Finished reading xml input file
```

We can print out information about the reaction system as follows:

```
>>>print(rxn_system)
chemical equations:
[
H + O2 =] OH + O
H2 + O =] OH + H
H2 + OH =] H2O + H
]
species: [`H', `O', `OH', `H2', `H2O', `O2']
nu_react:
[[ 1. 0. 0.]
 [ 0. 1. 0.]
 [ 0. 0. 1.]
 [ 0. 1. 1.]
 [ 0. 0. 0.]
 [ 1. 0. 0.]]
nu_prod:
[[ 0. 1. 1.]
```

```
 [ 1. 0. 0.]
 [ 1. 1. 0.]
 [ 0. 0. 0.]
 [ 0. 0. 1.]
 [ 0. 0. 0.]]
reaction coefficients:
[
Arrhenius Reaction Coeffs: {`A': 35200000000.0, `E': 71400.0, `R': 8.314}
modifiedArrhenius Reaction Coeffs: {`A': 0.0506, `b': 2.7, `E': 26300.0, `R': 8.314}
Constant Reaction Coeffs: {`k': 1000.0, `R': 8.314}
]
reaction types: [`Elementary', `Elementary', `Elementary']
reversible: [`no', `no', `no']
```

3.2 Computing rate coefficients

While the xml file provides the parameters for the functional form of the rate coefficient expression, a temperature (usually) has to be specified to compute the rate coefficient. Our package does this in the following manner:

```
>>> rc = ReactionCoeffs('Arrhenius', A = 1e7, E=1e3)
>>> rc.set_params(T=1e2)
>>> rc.kval()
   3003549.0889639612
```

3.3 Computing progress rates

The progress rate values w_i can be computed in the following manner:

```
>>>T = 1000 #K
>>>x = np.array([1,1,1,1,1,1])
>>> rxn_system.set_rc_params(T=1000)
>>>rxn_system.progress_rate(x)
array([ 6.55925729e+06, 2.69357993e+05, 1.00000000e+03])
```

3.4 Computing reaction rates

Given the reaction data from a user-provided input file, the reaction rates can be computed for an arbitrary temperature and set of species concentrations.

```
>>>T = 1000 #K
>>>x = np.array([1,1,1,1,1,1])
>>>rxn_system.reaction_rate_T(x,T)
```

```
array([ -6.28889929e+06, 6.28989929e+06, 6.82761528e+06,
          -2.70357993e+05, 1.00000000e+03, -6.55925729e+06])
```

There are more use cases of our library, please refer to `demo.ipynb`