The PYCHEMKIN User Manual

Jane Huang, Kimia Mavon, Weidong Xu, Zeyu Zhao

1 Introduction

`pychemkin` is a Python 3 library that computes the reaction rates of species participating in a system of elementary reactions.

1.1 Key chemical concepts and terminology

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

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

for **irreversible reactions** (i.e., the reaction only proceeds in the forward direction) and

$$\sum_{i=1}^{N} \nu'_{ij} \mathcal{S}_i \rightleftharpoons \sum_{i=1}^{N} \nu''_{ij} \mathcal{S}_i, \qquad j = 1, \ldots, M. \tag{2}$$

for **reversible reactions** (i.e., the reaction can proceed in either the forward or backward directions).

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

$$f_i = \sum_{j=1}^{M} (\nu''_{ij} - \nu'_{ij})\omega_j = \sum_{j=1}^{M} \nu_{ij}\omega_j, \qquad i = 1, \ldots, N. \tag{3}$$

The **progress rate** of the jth reaction is given by

$$\omega_j = k_j^{(f)} \prod_{i=1}^{N} x_i^{\nu'_{ij}} - k_j^{(b)} \prod_{i=1}^{N} x_i^{\nu''_{ij}}, \qquad j = 1, \ldots, M. \tag{4}$$

The **forward reaction rate coefficient** $k_j^{(f)}$ is assumed to take one of three possible forms:

1. $k = \text{constant}$
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.

The forward and backward reaction rate coefficients are related by

$$k_j^{(b)} = \frac{k_j^{(f)}}{k_j^e}, \qquad j = 1, \ldots, M, \tag{5}$$

where the **equilibrium coefficient** k_j^e is given by

$$k_j^e = \left(\frac{p_0}{RT}\right)^{\gamma_j} \exp\left(\frac{\Delta S_j}{R} - \frac{\Delta H_j}{RT}\right), \qquad j = 1, \ldots, M. \tag{6}$$

The pressure p_0 is fixed at 10^5 Pa in this package. $\gamma_j = \sum_{i=1}^{N} \nu_{ij}$. The **entropy change** of reaction j is

$$\Delta S_j = \sum_{i=1}^{N} \nu_{ij} S_i, \qquad j = 1, \ldots, M, \tag{7}$$

where S_i is the entropy of specie i. Likewise, the **enthalpy change** of reaction j is

$$\Delta H_j = \sum_{i=1}^{N} \nu_{ij} H_i, \qquad j = 1, \ldots, M. \tag{8}$$

An irreversible reaction can be thought of as the limiting case where k_j^e approaches ∞, in which case the backwards reaction rate coefficient $k_j^{(b)}$ approaches 0. The progress rate expression then simplifies to

$$\omega_j = k_j^{(f)} \prod_{i=1}^{N} x_i^{\nu'_{ij}}, \qquad j = 1, \ldots, M. \tag{9}$$

1.2 Features

The package can solve for the reaction rates of a system of elementary 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.

1.2.1 Calculation of thermodynamic quantities

Traditionally for combustion chemistry, the entropy and enthalpy of each species are approximated by polynomial fits to numerical calculations from Gordon and McBride's 1963 report, *The Thermodynamic Properties of Chemical Substances to 6000 K, NASA Report SP-3001*:

$$\frac{H_i}{RT} = a_{i1} + \frac{1}{2}a_{i2}T + \frac{1}{3}a_{i3}T^2 + \frac{1}{4}a_{i4}T^3 + \frac{1}{5}a_{i5}T^4 + \frac{a_{i6}}{T} \tag{10}$$

and

$$\frac{S_i}{R} = a_{i1}\ln(T) + a_{i2}T + \frac{1}{2}a_{i3}T^2 + \frac{1}{3}a_{i4}T^3 + \frac{1}{4}a_{i5}T^4 + a_{i7}. \tag{11}$$

These are known as the **NASA polynomials**. For each specie, there are two sets of coefficients a_i, the first of which is applicable at low temperatures and a second that is applicable at high temperatures. `pychemkin` stores these coefficients (taken from `http://burcat.technion.ac.il/dir/`) in an SQL database and retrieves values for species requested by the user.

2 Installation

2.1 Where to find and download the code

2.1.1 Using pip

`pychemkin` v. 0.1.1 is hosted on PyPi at `https://pypi.python.org/pypi/pychemkin`. To download and install the package, simply type `pip install pychemkin` into your terminal.

2.1.2 Installing from source

To get the most up-to-date version, you can go to `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. Once you download the contents of the repository from GitHub, enter the directory and type `python setup.py install` In order to run the test suite, you need to have pytest v. 3.00+ and pytest-cov v. 2.5+ installed. When you're in the top level of the package directory, type `pytest` into the terminal. The results of the test code will be printed out to the terminal.

2.2 Dependencies

This package has dependencies that usually come standard with the Anaconda distribution, but will otherwise automatically be installed for you if you use pip.

- NumPy v. 1.11+
- sqlite3 v. 3.13.0+

Earlier versions of these packages may work, but the code has only been validated on the listed versions.

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 User-required input

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 `tests/test_xml` 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 pychemkin import chemkin
>>>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 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])
```

3.3 Obtaining intermediate calculations

Rate coefficients and progress rates are calculated in the course of computing the reaction rates. While these methods do not have to be called by the user to obtain the reaction rates, they are accessible if the user wishes to obtain these values.

3.3.1 Obtaining forward 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:

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

3.3.2 Obtaining 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.progress_rate(x,T)
array([ 6.55925729e+06, 2.69357993e+05, 1.00000000e+03])
```