C207 Final Project
Report/Documentation

Chemical Kinetics Library

Submitted in fulfillment as per the requirements of the course

CS207 : Systems Development for Computational Science

Submitted by
Group 11

Team Members

Karan R. Motwani
Hannah Sim
Shiyu Huang
Haixing Yin

Under the guidance of
Prof. David Sondak

Harvard University
Cambridge, Massachusetts – 02138

Fall Semester, 2017

Abstract

This repository is collection of Python algorithms and classes intended for research in Reaction Rate quantification, including the calculation of Reaction Rate Coefficients and Progress Rate for a system of chemical reactions. The library is capable of receiving an XML file as input and extracting the chemical species and quantities of interest for further manipulation. The library is designed for flexibility, extensibility and ease of use. Its software design follows an object-oriented approach and its code is written on Python over an Anaconda Platform. The library consists of three levels of documentation: a model document available in PDF format, comprehensive code documentation within individual Python files and a lower-level user's guide in the README file available in the repository. This is the model document: it gives a comprehensive overview of the library's capabilities, provides procedures for software execution, and includes example for ease of use.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Definition

It is accepted that computation has emerged as the third pillar of science alongside the pillars of theory and experiment. Computational science is maturing rapidly and has found considerable and significant use in supporting scientists from various disciplines. The Chemical Kinetics Library available in this repository aims to ease the process of calculation of Reaction Rate, Reaction Rate Coefficients and Progress Rate of a system of chemical reactions. The tool simplifies the process by orders of magnitude by simply taking a standard XML file as input and returning the desired parameters in suitable data structures. This library has great application in the field of Ordinary Differential Equation (ODE) solving for chemical reactions and can be further extended to more complex applications such as learning new reaction pathways in an artificial neural net based code library.

1.2 Chemical Kinetics : Brief Overview

In the follow subsections, the essential Chemical Kinetics concepts behind the algorithms and their implementation have been discussed.

1.2.1 Reaction Rate and Coefficient

In chemical kinetics a reaction rate constant or reaction rate coefficient, k, quantifies the rate of a chemical reaction. For a reaction between reactants A and B to form product C :

$$\nu_A A + \nu_B B \longrightarrow \nu_C C.$$

The reaction rate is often found to have the form :

$$r = k(T)[A]^m[B]^n$$

Here k(T) is the reaction rate constant that depends on temperature. [A] and [B] are the molar concentrations of substances A and B in moles per unit volume of solution, assuming the reaction is taking place throughout the volume of the solution. (For a reaction taking place at a boundary one would use instead moles of A or B per unit area).

The exponents **m** and **n** are called partial orders of reaction and are not generally equal to the stoichiometric coefficients a and b. Instead they depend on the reaction mechanism and can be determined experimentally.

The Reaction Rate Coefficient has multiple forms :

$$k_{\text{const}} = k \qquad \textbf{Constant}$$

$$k_{\text{arr}} = A \exp\left(-\tfrac{E}{RT}\right) \qquad \textbf{Arrhenius}$$

$$k_{\text{mod-arr}} = AT^b \exp\left(-\tfrac{E}{RT}\right) \qquad \textbf{Modified Arrhenius}$$

The symbols used above are :

- The Arrhenius prefactor : A
- The Modified Arrhenius parameter : b
- The Temperature : T in Kelvin.
- Activation Energy : E in Joule/mol
- $R = 8.314$ is the ideal gas constant (in Joules/ mol K).

1.2.2 System of Reactions

Consider a system consisting of N species undergoing M irreversible, elementary reactions of the form :

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

The Rate of Change of Specie i (the Reaction Rate) can be written as :

$$f_i = \sum_{j=1}^{M} \nu_{ij}\omega_j, \qquad i = 1, \ldots, N$$

And the Progress Rate for each reaction is given by :

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

and k_j is the forward reaction rate coefficient. The symbols used above are :

- Chemical symbol of the Specie i : $\mathcal{S}_i$
- Stoichiometric Coefficients of the Reactants : ν'_{ij}
- Stoichiometric coefficients of Products : ν''_{ij}
- Rate of consumption or formation of specie i (Reaction Rate) : f_i
- Progress Rate of reaction j : ω_j
- Concentration of Specie i : x_i
- Reaction Rate Coefficient for reaction j : k_j

1.3 Code Features

1.3.1 Flow of Control

From the above expressions, it is clear that the terms/parameters of a system of reactions need to be calculated in an specified order. The Reaction and Progress Rate (ω_j) depend on the Rate Coefficient (k_j) and the Concentrations of each Specie (ν_{ij}) depends on the Reaction Rate (f_i) and Time (t) for which the reaction is allowed to progress.

The Initial Concentrations of the Reactants (ν'_{ij}) and the Temperature (T) need to be input by the user dynamically for a specific reaction to observe the variance of the results for different initial conditions. The Species involved in the system and the type of reaction, however, is pre-defined.

Abiding by this model, the usage of the library needs to follow an ordered set of operations :

- The **Parser** extracts the Parameters and Species of a chemical reaction from XML file.

- In the **ChemKin** module, the Reaction Rate Coefficient needs to be calculated for an input Temperature and Concentrations using the information about the extracted Parameters.
- Finally, the respective Reaction Rate/Progress Rate can be calculated based on Stoichiometric Coefficients, provided as input through the **Terminal**.

1.3.2 Data Structures

Classes

- **class 'ReactionParser'** : The user instantiates an object of this class with the XML filename as an argument, it consists of methods that facilitate the extraction of Type of Reaction, Rate Parameters, Species and Stoichiometric Coefficients from the file.
- **class 'Reaction'** : The class 'ReactionParser' instantiates an object of this class with the parameters obtained from methods of the class 'ReactionParser'. It consists of methods that allow the user definition of Temperature, Concentrations and calculation Progress/Reaction Rate.
- **class 'ReactionCoeff'** : An object of this class is called by method 'compute_reaction_rate_coeff' of class 'Reaction' with the parameters for the calculation of Rate Coefficient in the form of a dictionary and Temperature as arguments. It consists of methods that calculate the Reaction rate Coefficent based on the parameters passed.

Dictionaries

- **Reaction Rate Parameters** : Parameters such as Activation Energy, Arrhenius pre-factor and Rate Constant are passed as a single dictionary for ease of calculation of Rate Coefficients.
- **Stoichiometric Reactant Coefficients** : The values of the Stoichiometric Reactant Coefficients are stored in a dictionary with the unique species as an index for ease of calculation of Reaction/Progress Rate.
- **Stoichiometric Product Coefficients** : The values of the Stoichiometric Product Coefficients are stored in a dictionary with the unique species as an index for ease of calculation of Reaction/Progress Rate.

Other Data Structures

- **Lists** : are used to store a collection of the unique species of the system of reactions.
- **Sets** : are used for easy unique-ness check operations at the reaction level.

Chapter 2

Installation

2.1 Downloading the Repository

The repository is available on **github** through URL : `https://github.com/cs207-group11/cs207-FinalProject.git`. To download :

- Below 'Contributors', a green button to 'Clone or download' the repository is present.
- Use the 'Download ZIP' option to begin the download.
- Once completed, the ZIP file will be present in 'Downloads' folder of your computer (default).
- Unzip this file into a directory of your choice using WinRAR or equivalent software.
- Now, follow instructions in Chapter 3 : Procedure for usage.

This link can also be used to clone the repository to your machine through the Mac Terminal/GitBash.

- Enter directory of your choice using appropriate 'cd' commands.

```
$ git clone <repository URL mentioned above>
$ cd cs207-FinalProject
```

You are now in the repository and can follow Chapter 3 : Procedure instructions for usage.

2.2 Test Suite

The testing is an important phase of development of any project and a variety of tests are necessary before the design is released. We have defined a test module called 'test_chemkin.py' within this repository.

2.2.1 Types of Errors

- ValueError: check the user's input values for incorrect variable or parameter definitions. Ex : Negative Temperature/Molar Concentration.
- AttributeError: check if the execution of the program is in the right order. Ex : If the Rate Coefficient has been calculated before attempting to calculate the Reaction Rate.
- TypeError : checks if the entered parameters are real and belong to a certain data type. Ex : Modified Arrhenius factor 'b' is real and a float.
- NotImplementedError : checks if a certain feature/function has been designed or is due for implementation.

The above types of errors are tested for within the module 'test_chemkin.py' for erroneous input to test the robustness of the 'chemkin.py' module.

2.2.2 Running the Test Suite

The test suite can be run from the **Terminal** to view the results of the testing.

```
#Enter repository directory
>>> python -m pytest test_chemkin.py
```

Chapter 3

Usage and Examples

3.1 Procedure

- Clone the repository from **github** onto your local machine.
- Copy your target XML file to the directory of the cloned repository and instantiate an object of **class ReactionParser** with the filename.
- The '__call__' method defined for your convenience allows you to begin the extraction of parameters from the XML file.
- In addition to the parameters, information about the Stoichiometric Coefficients and Type of Reaction is derived from the XML File for the reactants and products separately.
- The object of **class Reaction** accepts the parameters extracted from the XML file directly through the **class ReactionParser**.
- The Temperature needs to be provided by the user at run-time through a method 'set_temperature' of the **class Reaction**.
- The Reaction Rate Coefficient is calculated by calling a method 'compute_reaction_rate_coeff' of **class Reaction** which instantiates an object of **class ReactionCoeff** and passes a dictionary of parameters necessary for calculation.
- The **class ReactionCoeff** consists of methods for individual forms of the Reaction Rate Coefficient and returns the coefficient according to the parameters passed as input.

- Finally, the concentrations of the unique species needs to be provided by the user at run-time through a method 'set_concentrations' of the **class Reaction**.
- Using the Reaction Rate Coefficent, Unique Specie Concentrations and Stochiometric Coefficients, the Reaction Rate and Progress Rate can be obtained using methods 'compute_reaction_rate' and 'compute_progress_rate' defined in the **class Reaction**.

3.2 Examples

3.2.1 Case 1 : Single Reaction System

Problem :
Calculate the **Progress Rate** for a reaction of the following form :

$$\nu_A A + \nu_B B \longrightarrow \nu_C C.$$

Order your concentration vector so that x = [[A] [B] [C]]^T
Test your function with $\nu_i' =$ [2.0 1.0 0.0]^T
And concentrations $x =$ [1.0 2.0 3.0]^T
Reaction Rate Coefficent : $k = 10$

Solution Code :

```
#Copy target XML file into repository
>>> parser(xml_filename)
>>> parser()

#Fetch desired reaction index, in this case 0 (zero)
>>> reac1 = parser.reaction_list[0]

#Print Reaction Equation
>>> print(reac1)

#Set Temperature (although not needed for this)
#reac1.set_temperature(1000)
>>> reac1.set_concentrations({'A':1,'C':3,'B':2})

#Order doesn't matter as long as all species exist
>>> k = reac1.compute_reaction_rate_coef()
```

```
>>> omega = reac1.compute_progress_rate()

#Reaction Rate can be calculated similarly
>>> print ("Reaction Rate Coefficient =",k)
>>> print ("Progress Rate =",omega)
```

Result :

```
Hannahs-MacBook-Pro-5:cs207-FinalProject hannahsim$ python crude_interface.py
Reaction : A + B =] C
Reaction rate coefficient (k) : 10.0
Reaction progress rate: [ 20.]
Hannahs-MacBook-Pro-5:cs207-FinalProject hannahsim$
```

Chapter 4

Future Work

In the future, we intend to add a non-trivial feature for simplifying calculations in Chemical Kinetics.

References

[1] Chemical Kinetics : Wikipedia, https://en.wikipedia.org/wiki/Chemical_kinetics