[![Build Status](https://travis-ci.org/cs207-g1/cs207-FinalProject.svg?branch=master)](https://travis-ci.org/cs207-g1/cs207-FinalProject)

[![Coverage Status](https://coveralls.io/repos/github/cs207-g1/cs207-FinalProject/badge.svg?branch=master)](https://coveralls.io/github/cs207-g1/cs207-FinalProject?branch=master)

# Chemical Kinetics Library

> Editors: Paul Blankley, Ryan Janssen, Boyuan Sun

> Date started: Oct 13, 2017

> Course: CS 207. System Development for Computational Science

> Organization: Harvard SEAS CS.207 Group 1

> Target User: Users who want to utilize the reation rate coefficients for various purposes

## I. Introduction

### 1.1 Preliminaries

Chemical reactions take place everywhere at any time. Computing the reaction rates of a particular reaction is the first and fundatmental step towards understanding the mechanics of the reaction. Various types of chemical reactions and reaction rates coefficients will be addressed in our library. The \*purpose\* of this chemical kinetics library is to calculate the reaction rates for species in by accepting users' input sets of chemical reactions in xml with corresponding coefficients. The library will calculate the progress rates, reaction rate coefficients including constant reaction rate coefficients, Arrhenius reaction rate coefficients, and modified Arrhenius reaction rate coefficients, and finally the reaction rates for each specie that the users have specified. Users will then choose how to use the rates given by the library afterwards.

### 1.2 Key Terminology

#### 1.2.1 Elementary reactions

[Elementary reactions](https://en.wikipedia.org/wiki/Elementary\_reaction), as indicated by the name, are the most fundamental chemical reactions in chemical kinetics. In short, if there is no intermediates found during the span of a chemical reaction, we define the reation to be elementary.

#### 1.2.2 Irreversible reactions vs. reversible reactions

[Irreversible reactions and reversible reactions](https://chem.libretexts.org/Core/Physical\_and\_Theoretical\_Chemistry/Equilibria/Reversibility\_and\_Equilibria/Reversible\_vs.\_Irreversible\_Reactions) are the reactions our library focus on. Both reactions are commonly found in nature. In short, irreverisble reactions occur when the products of the reactions cannot not be converted to the previous reactants. Reversible reactions are the opposite.

#### 1.2.3 Reaction rate coefficients

Chemical reaction rates are discussed [here](https://en.wikipedia.org/wiki/Reaction\_rate\_constant). In our chemical kinetics library we support three types of reaction rate coefficients:

- Constant reaction rate coefficients

\* Defined by the constant reaction rate coefficient k. No additional calculation is needed.

- Arrhenius reaction rate coefficients

\* Detailed description can be found and formula can be found [here](https://en.wikipedia.org/wiki/Arrhenius\_equation).

- Modified Arrhenius (a.k.a Kooij) reaction rate coefficients

\* Detailed description can be found and formula can be found [here](https://en.wikipedia.org/wiki/Arrhenius\_equation).

#### 1.2.4 Progress rate and Reaction rate

[Reaction rate](https://www.britannica.com/science/reaction-rate) in general defines the speed of a chemical reaction. Reaction rates are computed by the users' defined sets of inputs for each specie given its corresponding concentration and reaction rates. The definition of progress rate and reaction rate and formulas for computation can be found [here](https://github.com/IACS-CS-207/cs207-F17/blob/master/lectures/L8/L8.ipynb).

## II. Installation

We plan to migrate the project to the Anaconda cloud with PyPI for future easy install. For a beta version of the software package, download the chemkin.py file and place the file in your working directory. You can also run the following command to clone the git repo and then move the chemkin module to your working directory.

$ git clone https://github.com/cs207-g1/cs207-FinalProject.git

Once you move chemkin.py to your working directory you can just run:

- import chemkin

After that command you will have full access to the module.

## III. Basic Usage and Examples

### 3.1 General Usage

The ultimate output of the Chemical Kinetics library is to return a set of reaction rates \*f\* for a given set of reactions, which is calculated by calling the \*reaction\_rate\* method. Intermediate outputs can be called using the \*reaction\_coef\* method (returning the 'k' reaction coefficients for the system), or the \*progress\_rate\* method (returning the progress rate \*omega\*).

Data can be input in one of two ways:

1. From an .xml file via the built-in \*get\_reactions\* xml parser function, on class initiation. XML must be in the format specified in the attached sample .xml.

2. By inputting parameters directly to the class via the \*set\_params\* method.

All methods are implemented in the Reaction class, which will contain the necessary parameters to calculate outputs for a specified reaction temperature and species concentrations.

### 3.2 Method/Function details – ReactionSet class

#### 3.2.1 ReactionSet(xml\_doc) \*(class initialization)\*

This class represents the reaction tools for a set of elementary reactions. The class takes in an xml specifying the reaction data on initialization, of form specified in the “xml template” section below.

\_\_\*Args\*\_\_:

\* param\_dict; where param\_dict is the output from the parser function.

\_\_\*Returns\*\_\_:

\* None, but instantiates self

\_\_\*Raises\*\_\_:

\* None

</blockquote>

<br>

#### 3.2.2 reaction\_coefs(self, T)

Sets reaction coefficients for the given float temperature T. May be used externally but more commonly called by the class' own function progress\_rate.

<blockquote>

\_\_\*Args\*\_\_:

\* T; float; the temperature for all reactions

\_\_\*Returns\*\_\_:

\* None

\_\_\*Raises\*\_\_:

\* ValueError when T cannot be cast to a float or T is negative

Implementation Example:

```

>>> vp = np.array([[1.,2.],[2.,0.],[0.,2.]])

>>> vpp = np.array([[0.,0.],[0.,1.],[2.,1.]])

>>> pdict = {'vprime': vp, 'v2prime': vpp, 'A': [.00045,.00045], \

'b': [1.2,1.2], 'E': [1.7,1.7], \

'k': [float('nan'),float('nan')], 'coeftype': ['Arrhenius','modifiedArrhenius']}

>>> rrr = Reaction(pdict)

>>> rrr.reaction\_coef(900)

[0.00044989777442266471, 1.5783556022951033]

```

</blockquote>

<br>

#### 3.2.3 progress\_rate(self, x, T):

This function calculates the progress rate $\omega$ of a reaction of the following form:

V'11\*A + V'21\*B -> V''31\*C

V'12\*A + V'32\*C -> V''22\*B + V''32\*C

It takes in the vectors v', v'' and x in the order [[A],[B],[C]].

<blockquote>

\_\_\*Args\*\_\_:

\* v',v''; matrices, numpy arrays of form mxn where m is the number of reactants and n is number of equations.

\* x; vector, numpy array (or list of lists) of length equal to the number of reactants in the system of equations.

\_\_\*Returns\*\_\_:

\* list of floats; the progress rate of the reaction for each equation

\_\_\*Raises\*\_\_:

\* ValueError if the shapes of the v matrices are not equal

Implementation example:

```

>>> vp = np.array([[1.,2.],[2.,0.],[0.,2.]])

>>> vpp = np.array([[0.,0.],[0.,1.],[2.,1.]])

>>> pdict = {'vprime': vp, 'v2prime': vpp, 'A': [float('nan'),float('nan')], \

'b': [float('nan'),float('nan')], 'E': [float('nan'),float('nan')], \

'k': [10,10], 'coeftype': ['Constant','Constant']}

>>> rrr = Reaction(pdict)

>>> rrr.progress\_rate(np.array([[1.],[2.],[1.]]),10)

[40.0, 10.0]

```

</blockquote>

<br>

#### 3.2.4 reaction\_rate(self,x,T):

This function calculates the reaction rate of a reaction of the following form:

V'11\*A + V'21\*B -> V''31\*C

V'32\*C -> V'12\*A + V''22\*B

It takes in the vectors v', v'' from the class and x in the order [[A],[B],[C]].

<blockquote>

\_\_\*Args\*\_\_:

\* x; vector, numpy array (or list) of length equal to the number of reactants in the system of equations.

\* T; float, the strictly positive temperature

\_\_\*Returns\*\_\_:

\* vector of floats; the reaction rate for each equation

\_\_\*Raises\*\_\_:

\* ValueError when temp is less than 0

Implementation Example:

```

>>> vp = np.array([[1.,2.],[2.,0.],[0.,2.]])

>>> vpp = np.array([[0.,0.],[0.,1.],[2.,1.]])

>>> pdict = {'vprime': vp, 'v2prime': vpp, 'A': [float('nan'),float('nan')], \

'b': [float('nan'),float('nan')], 'E': [float('nan'),float('nan')], \

'k': [10,10], 'coeftype': ['Constant','Constant']}

>>> rrr = Reaction(pdict)

>>> rrr.reaction\_rate(np.array([[1.],[2.],[1.]]),10)

[-60.0, -70.0, 70.0]

```

</blockquote>

<br>

#### 3.2.5 set\_params(self,idx,A=None,b=None,E=None,R=None, k=None, coeftype=None):

This function takes inputs of the parameters you want to set for reaction coefficient calculations.

<blockquote>

\_\_\*Args\*\_\_:

\* A,b,E,T,R; all floats and optional arguments

\_\_\*Returns\*\_\_:

\* None (updates internal class parameters)

\_\_\*Raises\*\_\_:

\* ValueError ValueError when any input given a value other than None cannot be cast to a float

Implementation example:

```

>>> vp = np.array([[1.,2.],[2.,0.],[0.,2.]])

>>> vpp = np.array([[0.,0.],[0.,1.],[2.,1.]])

>>> pdict = {'vprime': vp, 'v2prime': vpp, 'A': [.00045,.00045], \

'b': [1.2,1.2], 'E': [1.7,1.7], \

'k': [float('nan'),float('nan')], 'coeftype': ['Arrhenius','modifiedArrhenius']}

>>> rrr = Reaction(pdict)

>>> w = rrr.reaction\_coef(900)

>>> ww = rrr.set\_params(1,k=10, coeftype='Constant')

>>> rrr.reaction\_coef(900)

[0.00044989777442266471, 10.0]

```

</blockquote>

<br>

#### 3.2.6 \_arrhenius(self, idx, T):

This internal function takes in the parameter T (kelvin temperature) from the class attributes, and it will return a value, k, that is the Arrhenius reaction rate coefficient.

<blockquote>

\_\_\*Args\*\_\_:

\* T, float; temperature, (gets args from class).

\_\_\*Returns\*\_\_:

\* The float k where k is the reaction rate coefficient.

\_\_\*Raises\*\_\_:

\* OverflowError after constant evaluation

\* FloatingPointError after constant evaluation for underflow

</blockquote>

<br>

#### 3.2.7 \_mod\_arrhenius(self, idx, T):

This internal function takes in the parameter T (kelvin temperature) from the class attributes, and it will return a value, k, that is the modified Arrhenius reaction rate coefficient.

<blockquote>

\_\_\*Args\*\_\_:

\* T, float; temperature (gets args from class).

\_\_\*Returns\*\_\_:

\* The float k where k is the reaction rate coefficient.

\_\_\*Raises\*\_\_:

\* OverflowError after constant evaluation

\* FloatingPointError after constant evaluation for underflow

</blockquote>

<br>

#### 3.2.8 get\_reactions(name):

This function takes in the name of the input xml file, and returns a dictionary of relevant information for a set of chemical reactions. Note this is a \*\*function\*\* within Chemkin and not a \*\*method\*\* of Reaction.

<blockquote>

\_\_\*Args\*\_\_:

\* name; name of the input .xml file

\_\_\*Returns\*\_\_:

\* reaction\_dict, dictionary of data for a reaction. Contains the following keys:

\* reaction\_dict['species']; list of strings, species of the reaction

\* reaction\_dict['As']; list of floats, corresponding to reaction parameter A for each equation (= NaN for any equations that don't use A.

\* reaction\_dict['bs']; list of floats, corresponding to reaction parameter b for each equation (= NaN for any equations that don't use b.

\* reaction\_dict['Es']; list of floats, corresponding to reaction parameter E for each equation (= NaN for any equations that don't use E.

\* reaction\_dict['ks']; list of floats, corresponding to reaction parameter k for each equation (= NaN for any equations that don't use k (ie, non-constant equations)).

\* reaction\_dict['rxn\_types']; List of strings. Elements Correspond to same reactions as reaction\_parameters. Each string is one of { 'Arrhenius', 'modifiedArrhenius', 'Constant' }

\* reaction\_dict['vprime']; np array, full vprime matrix of all reactions in the xml file

\* reaction\_dict['v2prime']; np array, full v2prime matrix of all reactions in the xml file

\_\_\*Raises\*\_\_:

\* FileNotFoundError if name is not a valid .xml path

\* ValueError if xml is not in specified data format

Implementation example:

```

>>> print(ck.get\_reactions("demo\_xmls/rxns.xml"))

{'species': array(['H', 'O', 'OH', 'H2', 'H2O', 'O2'], dtype='<U3'),

'A': array([ 3.52000000e+10, 5.06000000e-02, nan]),

'b': array([ nan, 2.7, nan]),

'E': array([ 71400., 26300., nan]),

'k': array([ nan, nan, 1000.]),

'coeftype': array(['Arrhenius', 'modifiedArrhenius', 'Constant'], dtype='<U17'),

'vprime': array([[ 1., 0., 0.],

[ 0., 1., 0.],

[ 0., 0., 1.],

[ 0., 1., 1.],

[ 0., 0., 0.],

[ 1., 0., 0.]]),

'v2prime': array([[ 0., 1., 1.],

[ 1., 0., 0.],

[ 1., 1., 0.],

[ 0., 0., 0.],

[ 0., 0., 1.],

[ 0., 0., 0.]])}

```

</blockquote>

<br>

### 3.3 Method/Function details – Reaction/ReversibleReaction/IrreversibleReaction class family

## 4.0 Sample .xml format

All .xml reaction files should follow the sample format used below. Source and Designer of this format is David Sondak, Harvard University CS207:

```

<?xml version="1.0"?>

<ctml>

<phase>

<speciesArray> H O OH H2 H2O O2 </speciesArray>

</phase>

<reactionData id="test\_mechanism">

<!-- reaction 01 -->

<reaction reversible="no" type="Elementary" id="reaction01">

<equation>H + O2 =] OH + O</equation>

<rateCoeff>

<Arrhenius>

<A>3.52e+10</A>

<E>7.14e+04</E>

</Arrhenius>

</rateCoeff>

<reactants>H:1 O2:1</reactants>

<products>OH:1 O:1</products>

</reaction>

<!-- reaction 02 -->

<reaction reversible="no" type="Elementary" id="reaction02">

<equation>H2 + O =] OH + H</equation>

<rateCoeff>

<modifiedArrhenius>

<A>5.06e-2</A>

<b>2.7</b>

<E>2.63e+04</E>

</modifiedArrhenius>

</rateCoeff>

<reactants>H2:1 O:1</reactants>

<products>OH:1 H:1</products>

</reaction>

<!-- reaction 03 -->

<reaction reversible="no" type="Elementary" id="reaction03">

<equation>H2 + OH =] H2O + H</equation>

<rateCoeff>

<Constant>

<k>1.0e+03</k>

</Constant>

</rateCoeff>

<reactants>H2:1 OH:1</reactants>

<products>H2O:1 H:1</products>

</reaction>

</reactionData>

</ctml>

```