Chemical Kinetics Library

1. Introduction

`chemkin` stands for chemical kinetics and is an objected-oriented library for modeling kinetics of chemical reactions.

1.1 Key Chemical Kinetics Concepts

Chemical kinetics is the study of rates of chemical processes such as reaction rates, reaction mechanisms, etc. as well as the construction of mathematical models that can describe the characteristics of a chemical reaction (wikipedia). Typically, amounts of molecular species reacted (consumed)/formed and the rates of their consumption/formation are of interest.

Chemical reactions can be categorized as elementary or non-elementary and reversible or irreversible. Each class of chemical reactions can be modeled by ODEs (ordinary differential equations) or PDEs (partial differntial equations) using different strategies.

Irreversible Elementary Reaction

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

$$\sum_{i=1}^{N} \nu'_{ij} s_i \longrightarrow \sum_{i=1}^{N} \nu''_{ij} s_i, \qquad \text{for } j = 1, \ldots, M$$

where
s_i = Chemical symbol of specie i
ν'_{ij} = Stoichiometric coefficients of reactants
ν''_{ij} = Stoichiometric coefficients of products

The **rate of change of species** i (i.e. the **reaction rate of species** i) can be written as

$$f_i = \frac{d[i]}{dt} = \sum_{j=1}^{M} \nu_{ij} \omega_j \qquad i = 1, \ldots, N$$

where
$\nu_{ij} = \nu''_{ij} - \nu'_{ij}$
ω_j = Progress rate of reaction j

The **progress rate** ω_j for reaction j is given by

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

where
k_j = Forward reaction rate coefficient for reaction j
x_i = Concentration of specie i

There are several types of **forward reaction rate coefficient** k_j, including -

Constant coefficient: coefficient is constant for all reaction temperatures

$$k_j = k_j$$

Arrhenius coefficient:

$$k_j = A \cdot exp^{-E_a/(RT)}$$

Modified Arrhenius coefficient:

$$k_j = AT^b exp^{-E_a/(RT)}$$

where
A = Arrhenius prefactor
b = Modified Arrhenius paramter
E_a = Activation energy
R = Ideal gas constant
T = Temperature

Note: A complete table of notation of **irreversible elementary** reacion

Symbol	Meaning
s_i	Chemical symbol of specie i
ν'_{ij}	Stoichiometric coefficients of reactants
ν''_{ij}	Stoichiometric coefficients of products
N	Number of species in system
M	Number of elementary reactions
f_i	Rate of consumption or formation of specie i (reaction rate)
ω_j	Progress rate of reaction j
x_i	Concentration of specie i
k_j	Reaction rate coefficient for reaction j
A	Arrhenius prefactor
b	Modified Arrhenius paramter
R	Ideal gas constant
E_a	Activation energy
T	Temperature

Reversible Elementary Reaction

For a system consisting of N species undergoing M **reversible**, **elementary** reactions of the form:

$$\sum_{i=1}^{N} \nu'_{ij} s_i \rightleftharpoons \sum_{i=1}^{N} \nu''_{ij} s_i \qquad j = 1, \ldots, M$$

where

s_i = Chemical symbol of specie i
ν'_{ij} = Stoichiometric coefficients of reactants
ν''_{ij} = Stoichiometric coefficients of products

The **rate of change of species** i (i.e. the **reaction rate of species** i) can be written as

$$f_i = \frac{d[i]}{dt} = \sum_{j=1}^{M} \nu_{ij} r_j \qquad i = 1, \ldots, N$$

where
$\nu_{ij} = \nu''_{ij} - \nu'_{ij}$
r_j = Total progress rate of reaction j

The **total progress rate** r_j of reaction j is given by

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

where
$k_j^{(f)}$ = Forward reaction rate coefficient for reaction j
$k_j^{(b)}$ = Backward reaction rate coefficient for reaction j
x_i = Concentration of specie i

The **backward reaction rate coefficient** $k_j^{(b)}$ is given by

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

where
k_j^e = Equilibrium constant for reaction j

The **equilibrium constant** k_j^e is related to the equilibrium thermochemistry of the elementary reactions, and it is given by

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

where
$\gamma_j = \sum_{i=1}^{N} \nu_{ij}$
p_0 = Pressure of the reactor (e.g. 10^5 Pa)
ΔS_j = Entropy change of reaction j
ΔH_j = Enthalpy change of reaction j

Read more about Equilibrium constant

The **entropy change** ΔS_j and the **enthalpy change** ΔH_j of reaction j is given by

$$\Delta S_j = \sum_{i=1}^{N} \nu_{ij} S_i = \sum_{i=1}^{N} \nu_{ij} \int_{T_0}^{T} \frac{C_{p,i}(T)}{T} \, \mathrm{d}T \qquad j = 1, \ldots, M$$

$$\Delta H_j = \sum_{i=1}^{N} \nu_{ij} H_i = \sum_{i=1}^{N} \nu_{ij} \int_{T_0}^{T} C_{p,i}(T) \, \mathrm{d}T \qquad j = 1, \ldots, M$$

where
$C_{p,i}$ = Specific heat at constant pressure of species i

The **specific heat at constant pressure** $C_{p,i}$ is given by a polynomial in T (called the NASA polynomial),

$$C_{p,i} = \left(\sum_{k=1}^{5} a_{ik} T^{k-1} \right) R, \qquad i = 1, \ldots, N.$$

where
T = Temperature
R = Ideal gas constant

The integrated forms of ΔS_j, ΔH_j, $C_{p,i}$, using 7^{th} order NASA polynomials are:

$$\frac{C_{p,i}}{R} = a_{i1} + a_{i2} T + a_{i3} T^2 + a_{i4} T^3 + a_{i5} T^4$$

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

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

for $i = 1, \ldots, N$.

Note: A complete table of notation of **reversible elementary** reacion

Symbol	Meaning
s_i	Chemical symbol of specie i
ν'_{ij}	Stoichiometric coefficients of reactants
ν''_{ij}	Stoichiometric coefficients of products
N	Number of species in system
M	Number of elementary reactions
f_i	Rate of consumption or formation of specie i (reaction rate)
r_j	Total progress rate of reaction j
x_i	Concentration of specie i
$k_j^{(f)}$	Forward reaction rate coefficient for reaction j
$k_j^{(b)}$	Backward reaction rate coefficient for reaction j
k_j^e	Equilibrium constant for reaction j
p_0	Pressure of the reactor
ΔS_j	Entropy change of reaction j
ΔH_j	Enthalpy change of reaction j
S_i	Entropy of species i
H_i	Enthalpy of species i

Symbol	Meaning
$C_{p,i}$	Specific heat at constant pressure of species i
T	Temperature
R	Ideal gas constant

1.2 The `chemkin` Library

The high level functionality of the **chemkin** module is to take an XML file with reaction data as input and outputs the RHS (right-hand-side) of the ODE describing the rate of change of all molecular species involved in the chemical reaction(s) of interest (i.e. for **irreversible elementary** reaction: $\sum_{j=1}^{M} \nu_{ij}\omega_j, \text{ for } i = 1, \ldots, N$, and for **reversible elementary** reaction: $\sum_{j=1}^{M} \nu_{ij}r_j, \text{ for } i = 1, \ldots, N$).

Features of the **chemkin** module include:

- Parsing XML file with chemical reaction data to extract relevant paramters
- Handling the calculation of 3+ classes of reaction rate coefficients (e.g. constant, Arrhenius and modified Arrhenius) given the appropriate parameters
- Handling the calculation of progress rates (ω_j or r_j) and reaction rates (f_i) for a system consisting of N species undergoing M **irreversible** or **reversible elementary** reactions

Overall structure of the `chemkin` library

```
chemkin/
    __init__.py
    chemkin_errors.py
    preprocessing/
        __init__.py
        parse_xml.py
        tests/
            test_parse_xml.py
    reaction/
        __init__.py
        base_rxn.py
        elementary_rxn.py
        non_elementary_rxn.py
        reaction_coefficients.py
        tests/
            test_base_rxn.py
            test_elementary_rxn.py
            test_non_elementary_rxn.py
            test_reaction_coefficients.py
    thermodynamics/
        __init__.py
        thermo.py
        NASA_coef.sqlite
        tests/
            test_thermo.py
    viz/
        __init__.py
        summary.py
```

```
        tests/
            test_summary.py
    xml-files/
```

A brief description of each subdirectory:

- `chemkin_errors` module hosts functions to detect library-related errors.
- `preprocessing` package contains modules to parse input files, extracts and returns relevant reaction parameters into a python dictionary. Currently, the library only parses .xml input files.
- `reaction` package contains modules to handle different reaction types as well as calculating the forward reaction coefficients
- `thermodynamics` package contains modules to handle all thermodynamics related calculations. For now, it handles the processing of backward reaction coefficients in reversible reactions using the NASA_coef SQL database.
- `viz` package contains modules that allow the user to visualize reaction kinetics (e.g. printing reaction rates in a prettified, tabular format)

Proposed Feature

Our proposed future feature is to install a differential equation solver to calculate species concentrations as a function of time.

Motivation

We motivate our feature by the following:

- Visualize the change in species concentration over time.
- Identify time to reach equilibrium (i.e., reaction completion).
- Get the concentrations at some end time (t_{end}).

For the end-user, it may be useful to learn about the concentration of the various species at any given point in time (as opposed to just the reaction rates).

Implementation

There will be a solver.py module containing a class that will be the workhorse to solve concentration of species over time. User will be able to call this class and use a solve() method for the desired outputs.

The current `summary.py` module will call this method `plot_reaction_rates()` in order to produce a plot of the species concentrations over time.

The feature will require an external ODE solver, most likely the `ODEint` from `scipy.integrate` as well as the `matplotlib`.

We envision at least 3 additional library functions that follows:

1. Given an end-time (t_{end}) and reaction data, function outputs the concentrations of each species at t_{end}.
2. Given reaction data, function outputs the time to reach equilibrium (in the case of reversible elementary reactions) or the time for the reaction to reach completion (in the case of irreversible elementary reactions).
3. Given an end-time (t_{end}), function plots the time evolution of species concentrations from t_0 to t_{end}.

Additional packaging

Below is the structure of the add-on to the package.

```
chemkin/
    __init__.py
    solver/
        __init__.py
        ODEint_solver.py
        test/
            __init__.py
            test_ODEint_solver()
```

2. Installation

The neccessary code can be found at and downloaded from here.

It is also pip-installable using the following command:

```
$ pip install chemkin
```

You can specify the directory of installation using the following command:

```
$ pip install -t [installation directory] chemkin
```

You can run the test suite on your local machine by typing the following command in your command line when in the directory of the installed module.

```
$ cd [installation directory]
$ pytest
```

3. Basic Usage

3.1 Structure of the input file

Chemical reaction data should be stored in XML format with the following specifications:

1. a `<phase>` element with a `<speciesArray>` child element which lists the molecular species invovled in the reaction
2. a `<reactionData>` element that stores relevant parameters of the reaction:
    - `<reaction reversible>` tag indicates whether a reaction is reversible or irreversible, possible values = [“yes”, “no”]
    - `<type>` tag indicates whether a reaction is elementary or non-elementary, possible values = [“Elementary”,

"Non-Elementary]
- `<equation>` tag specifies the chemical reaction
- `<rateCoeff>` tag stores parameters relevant to calculate the rate coefficient. It has a child tag indicating the rate coefficient class, which can be one of three acceptable types, each of which dictates what coefficient values are parsed by the `XmlParser` class:
    1. `<Arrhenius>` : Coefficients [A, E] will be retrieved.
    2. `<modifiedArrhenius>` : [A, b, E] will be retrieved.
    3. `<Constant>` : k will be retrieved.

Note There should be only 1 system of reactions per XML file (i.e. 1 `<reactionData>` element). If more than 1 `<reactionData>` element resides within the XML, only the first `<reactionData>` element (i.e. first system of reactions) will be considered

Note For a system of chemical reactions, there should only be 1 type of reaction (i.e. all elementary reversible or all elementary irreversible). If more than 1 type of reaction is specified in the XML file, then a ChemkinError will be raised.

Note If no recognized child tag of `<rateCoeff>` is encountered, then `XmlParser` will raise a `ChemKinError`. Also, its retrieval of elements is case-sensitive. So, for example, `<A>`, `<b>`, `<E>`, and `<k>` must be used to store the appropriate coefficients; elements named `<a>`, `<B>`, `<e>` or `<K>` would not be recognized and would lead to an error.

Example 3.1. XML file for the following chemical reaction:

$$2H_2 + O_2 \quad \longrightarrow \quad 2OH + H_2$$

```
<?xml version="1.0"?>
<ctml>

    <phase>
        <speciesArray> H2 O2 OH HO2 H2O </speciesArray>
    </phase>

    <reactionData id="test_mechanism">

        <!-- reaction 01  -->
        <reaction reversible="no" type="Elementary" id="reaction01">
            <equation>2H2 + O2 [=] 2OH + H2</equation>
            <rateCoeff>
                <modifiedArrhenius>
                    <A units="m3/mol/s">1e+8</A>
                    <b>0.5</b>
                    <E units="J/mol">5e+04</E>
                </modifiedArrhenius>
            </rateCoeff>
            <reactants>H2:2 O2:1</reactants>
            <products>OH:2 H2:1</products>
        </reaction>
    </reactionData>

</ctml>
```

3.2 Reading and parsing the input file

Reading and parsing the input XML file is handled in the `preprocessing` package, where there are two related classes that allow the user to work with reaction data stored in XML files.

- `XmlParser`
- `RxnData`

3.2.1 `XmlParser`

The `XmlParser` class pulls reaction data from XML files and preprocesses the data (e.g. calculates the reaction rate coefficients from given paramters) with its `parsed_data_list(Ti)` method. `parsed_data_list(Ti)` takes a list of temperatures as input and returns a `list` of `dictionaries` that contain relevant reaction parameters at each of the temperatures.

Each dictionary in the returned list contains the following attributes:

- `species` : a list of moleucular species invovled in the reaction
- `ki` : a list of forward reaction rate coefficients, the i^{th} entry in the list is the coefficient for the i^{th} reaction in the system
- `sys_vi_p` : a list of stoichemotetric coefficients of the reactants, the i^{th} entry in the list is the coefficients for the i^{th} reaction in the system
- `sys_vi_dp` : a list of stoichemotetric coefficients of the products, the i^{th} entry in the list is the coefficients for the i^{th} reaction in the system
- `is_reversible` : an indicator of whether the reaction is reversible (takes on values of True or False)
- `T` : a float of reaction temperature
- `b_ki` : a list of backward reaction rate coefficients, the i^{th} entry in the list is the coefficient for the i^{th} reaction in the system (only dictionary for reversible reactions has this attribute)

Example 3.2. Read in, parse and preprocess an input XML file

```
from chemkin import pckg_xml_path
from chemkin.preprocessing.parse_xml import XmlParser

Ti = [100, 750, 1500, 2500, 5000]
xi = [2., 1., .5, 1., 1., 1., .5, 1.]

xml_parser = XmlParser(pckg_xml_path('rxns_reversible'))
parsed_data_list = xml_parser.parsed_data_list(Ti)
```

The `parsed_data_list(Ti)` method abstracts much of the preprocssing stage, including calculating the reaction rate coefficients. However, there are instances where the user might want to simply extract XML elements (e.g. if the user wishes to calculate the reaction rate coefficients in an user-specified manner). The `XmlParser` module also allows the user to work directly with XML elements with its `load()` method. `load` returns a `tuple` of two lists:

- the species invovled in all the reactions in the file
- list of `RxnData` objects, with each object containing the data for an individual reaction (discussed in next section)

Example 3.3. Read in and parse XML file to work with XML elements directly

```
from chemkin import pckg_xml_path
from chemkin.preprocessing.parse_xml import XmlParser

xml_parser = XmlParser(pckg_xml_path('rxns_reversible'))
species, reaction_data = xml_parser.load()
```

3.2.2 `RxnData`

The reaction data parsed by the XmlParser `load()` function from XML files is returned as a list of `RxnData` objects. This class encapsulates relevant information from the XML file in a way that allows the caller to easily process reactions differently according to their features.

`RxnData` objects have the following attributes:

- `rxn_id` : a string for reaction ID
- `reversible` : a boolean indicating whether the reaction is reversible
- `type` : a RxnType object from the Enum class indicating whether a reaction is elementary or non-elementary
- `rate_coef` : a list of parameters for reaction rate coefficient
- `reactants` : a dictionary with molecular species of the reactants as keys and their respecitve stoicheometric coefficient as values
- `products` : a dictionary with moelcular speicies of the products as keys and their respective stoichemotetric coefficient as values

Example 3.4. Working with `RxnData` objects

```
from chemkin import pckg_xml_path
from chemkin.preprocessing.parse_xml import XmlParser

xml_parser = XmlParser(pckg_xml_path('rxns_reversible'))
_, reaction_data = xml_parser.load()

for rxn in reaction_data:
    if rxn.reversible:
        # Handle special reversible reaction logic
```

3.3. Calculating kinetic parameters of interest

Two families of modules in the `reaction` package allow you to compute kinetic paramters such as reaction rate coefficients, progress rates and reaction rates.

- `reaction_coefficients` module handles calculations of reaction rate coefficients
- `base_rxn` , `elementary_rxn` and `non_elementary_rxn` modules handle calculations of progress rates and reaction rates.

3.3.1. `reaction_coefficients` module

The `reaction_coefficients` module contains a base class `RxnCoefficientBase` from which then the three subclasses (`ConstantCoefficient` , `ArrheniusCoefficient` , and `ModifiedArrheniusCoefficient`) inherit its basic properties (such as `init` , `__repr__` and `__eq__`). When creating the instances of these classes, their parameters are based on inputs extracted using the `load` method from the `XmlParser` class and stored in `RxnData` object. As documented above, `RxnData` objects have a `rate_coef` attribute, which is a list of parameters for reaction rate coefficient. The list can have the following entires: temperature T, Arrhenius constant A (where applicable), modified

constant b (where applicable), ideal gas constant R, and Activation energy E.

Since different classes of rate coefficients have different number of input parameters, we can utilize the length of the `rate_coef` to determine the class to use to calculate the reaction rate coefficients. The `get_coeff()` method handls the calculation of the the rate coefficients ki for reaction i.

Example 3.5. Calculating reaction rate coefficients

```
from chemkin import pckg_xml_path
from chemkin.preprocessing.parse_xml import XmlParser

xml_parser = XmlParser(pckg_xml_path('rxns_reversible'))

_, reaction_data = xml_parser.load()

coef_params = reaction_data.rate_coeff

        if isinstance(coef_params, list):
            if len(coef_params) == 3: # modified arrhenius coef
                A = coef_params[0]
                b = coef_params[1]
                E = coef_params[2]
                ki.append(ModifiedArrheniusCoefficient(A, b, E, T).get_coef())
            else: # arrhenius coef
                A = coef_params[0]
                E = coef_params[1]
                ki.append(ArrheniusCoefficient(A, E, T).get_coef())
        else: # const coef
            ki.append(ConstantCoefficient(coef_params).get_coef())
```

3.3.2 `base_rxn` module

The `base_rxn` module contains a single `RxnBase` class, from which the subclasses in `elementary_rxn` and `non_elementary_rxn` modules inherit. The `RxnBase` base class and its subclasses have methods to calculate the progress rates and reaction rates given data on a system of chemical reactions and associated parameters.

`RxnBase` objects have the following attributes:

- `ki` : a list of reaction rate coefficients
- `xi` : a list of concentrations of molecular species
- `vi_p` : a list of stoichiometric coefficients of the reactants
- `vi_dp` : a list of stoichiometric coefficients of the product
- `wi` : a list of progress rates
- `rates` : a list of reaction rates

Note These attributes are initialized when `RxnBase` objects are created. For example,

```
reaction1 = RxnBase(ki=[10, 10], xi=[1.0, 2.0, 1.0], vi_p=[[1.0, 2.0, 0.0], [2.0, 0.0,
2.0]], vi_dp=[[0.0, 0.0, 2.0], [0.0, 1.0, 1.0]]
```

`RxnBase` objects have the following methods:

- `progress_rate()` : Returns a list of progress rates for the system (Not implemented in the base class)
- `reaction_rate()` : Returns a list of reaction rates for the system (Not implemented in the base class)

3.3.3 `elementary_rxn` module

The `elementary_rxn` module deals with elementary chemical reactions. It contains an `ElementaryRxn` base class which inhertis from `RxnBase` and two subclasses `IrreversibleElementaryRxn` and `ReversibleElementaryRxn`, which handle **irreversible** and **reversible** elementary reactions, respectively.

The `IrreversibleElementaryRxn` class

This class handles a system consisting of N species undergoing M **irreversible**, **elementary** reactions of the form:

$$\sum_{i=1}^{N} \nu'_{ij} s_i \longrightarrow \sum_{i=1}^{N} \nu''_{ij} s_i, \qquad \text{for } j = 1, \dots, M$$

The progress rate ω_j is given by

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

The reaction rate $f_i = \frac{d[i]}{dt}$ is given by

$$f_i = \frac{d[i]}{dt} = \sum_{j=1}^{M} \nu_{ij} \omega_j, \qquad \text{for } i = 1, \dots, N$$

`IrreversibleElementaryRxn` shares the same class attributes as the base class and implements the two methods in the following manner:

- `progress_rate()` : Returns a list of $k_j \prod_{i=1}^{N} x_i^{\nu'_{ij}}$
- `reaction_rate()` : Returns a list of $\sum_{j=1}^{M} \nu_{ij} \omega_j$

The `ReversibleElementaryRxn` class

This class handles a system consisting of N species undergoing M **reversible**, **elementary** reactions of the form:

$$\sum_{i=1}^{N} \nu'_{ij} s_i \rightleftharpoons \sum_{i=1}^{N} \nu''_{ij} s_i \qquad j = 1, \dots, M$$

The total progress rate r_j is given by

$$r_j = k_j^{(f)} \prod_{i=1}^{N} x_i^{\nu'_{ij}} - k_j^{(b)} \prod_{i=1}^{N} x_i^{\nu''_{ij}}, \qquad j = 1, \dots, M$$

The reaction rate $f_i = \frac{d[i]}{dt}$ is given by

$$f_i = \frac{d[i]}{dt} = \sum_{j=1}^{M} \nu_{ij} r_j, \qquad \text{for } i = 1, \dots, N$$

`ReversibleElementaryRxn` shares the same class attributes as the base class and implements the two methods in the following manner:

- `progress_rate()` : Returns a list of $k_j^{(f)} \prod_{i=1}^{N} x_i^{\nu'_{ij}} - k_j^{(b)} \prod_{i=1}^{N} x_i^{\nu''_{ij}}$
- `reaction_rate()` : Returns a list of $\sum_{j=1}^{M} \nu_{ij} r_j$

3.3.4 `non_elementary_rxn` module

The implementation for non-elementary reactions is TBD.

4. Examples

4.1. A system of irreversible, elementary chemical reactions

- The system of chemical reactions of interest:

$$
\begin{aligned}
2H_2 + O_2 &\longrightarrow 2OH + H_2 \\
OH + HO_2 &\longrightarrow H_2O + O_2 \\
H_2O + O_2 &\longrightarrow HO_2 + OH
\end{aligned}
$$

- Input XML file:

```
<?xml version="1.0"?>

<ctml>
    <phase>
        <speciesArray> H2 O2 OH HO2 H2O </speciesArray>
    </phase>

    <reactionData id="test_mechanism">
        <!-- reaction 01  -->
        <reaction reversible="no" type="Elementary" id="reaction01">
            <equation>2H2 + O2 [=] 2OH + H2</equation>
            <rateCoeff>
                <modifiedArrhenius>
                    <A units="m3/mol/s">1e+8</A>
                    <b>0.5</b>
                    <E units="J/mol">5e+04</E>
                </modifiedArrhenius>
            </rateCoeff>
            <reactants>H2:2 O2:1</reactants>
            <products>OH:2 H2:1</products>
        </reaction>

        <!-- reaction 02 -->
        <reaction reversible="no" type="Elementary" id="reaction02">
            <equation>OH + HO2 [=] H2O + O2</equation>
            <rateCoeff>
                <Constant>
                    <k>1e+4</k>
                </Constant>
            </rateCoeff>
            <reactants>OH:1 HO2:1</reactants>
            <products>H2O:1 O2:1</products>
        </reaction>

        <!-- reaction 03 -->
        <reaction reversible="no" type="Elementary" id="reaction03">
            <equation>H2O + O2 [=] HO2 + OH</equation>
            <rateCoeff>
                <Arrhenius>
```

```
                    <A units="m3/mol/s">1e+7</A>
                    <E units="J/mol">1e+04</E>
                </Arrhenius>
            </rateCoeff>
            <reactants>H2O:1 O2:1</reactants>
            <products>HO2:1 OH:1</products>
        </reaction>
    </reactionData>
</ctml>
```

4.1.1 Printing reaction rates

This example demonstrates the highest-level and most-abstracted use of the library. The user simply specifies the temperatures and initial species concentrations with the corresponding input XML file which contains the reaction data, and the following code prints reaction rates in a prettified, tabular format.

- Given the the species concentration $xi = [2.0, 1.0, 0.5, 1.0, 1.0]$ in units of $m^3/(mol \cdot s)$, calculate the reaction rate of each species at $Ti = [750, 1500, 2500]$ in units of K.

```
from chemkin import pckg_xml_path
from chemkin.preprocessing.parse_xml import XmlParser
from chemkin.viz import summary

Ti = [100, 750, 1500, 2500, 5000]
xi = [2., 1., .5, 1., 1., 1., .5, 1.]

xml_parser = XmlParser(pckg_xml_path('rxns_reversible'))
parsed_data_list = xml_parser.parsed_data_list(Ti)
summary.print_reaction_rate(parsed_data_list, xi)
```

- The expected result:

```
------At Temperature 100 K------
Backward reaction coefficients not defined: T=100 is not in some specie's temperature
range.
--------------------------------
------At Temperature 750 K------
     H : 3.42304562795e+16
     O : -3.38308977852e+16
     OH : -3.52979102957e+16
     H2 : 4.07078984572e+13
     H2O : 5.86479724945e+14
     O2 : 3.44028050967e+16
     HO2 : -7.63606068937e+13
     H2O2 : -5.52803118677e+13
--------------------------------
------At Temperature 1500 K------
     H : 1.32279654839e+14
     O : -3.12877268431e+14
     OH : -1.37621783394e+14
     H2 : 6.07049182e+13
     H2O : 6.45299057466e+13
     O2 : 3.36486898686e+14
     HO2 : -4.18771319555e+13
     H2O2 : -1.01625193691e+14
```

```
--------------------------------
------At Temperature 2500 K------
     H : -2.4081673376e+14
     O : -1.5885953972e+13
     OH : 3.22268792909e+14
     H2 : 6.41189812893e+13
     H2O : 4.70126660046e+13
     O2 : -2.74831634318e+13
     HO2 : 5.28617565897e+12
     H2O2 : -1.54500764698e+14
--------------------------------
------At Temperature 5000 K------
Backward reaction coefficients not defined: T=5000 is not in some specie's temperature
range.
--------------------------------
```