MINICHEM

Release 0.0.1

Parthkumar Patel, A. John Arul

Jan 11, 2020

CONTENTS

The MINICHEM is a code to determine the mole quantity released in the cover gas, which is primarily the instantaneous in-containment source. Although, the present code only depicts the chemical aspects of the release behaviour of RN, which we consider it as a first step towards mechanistic model development for oxide fuelled SFRs. For the purpose of this analysis, ULOF event resulting in whole core melt is considered. The in-vessel source term is determined using chemical equilibrium approach with no mixture assumption. No mixture assumption essentially means that during chemical equilibrium, the mixing properties of the species are not considered; The estimated equilibrium species mole numbers corresponds to the vapour pressure of the species at specified temperature. This assumption usually leads to conservative estimates. With the help of this equilibrium species, distribution of RN in three phases (solid, gas and liquid) can be determined; From this information, release fractions of RNs to the cover gas are evaluated. The code is capable to calculate the equilibrium species at both (T, P) and (T, V) case.

Apart from source term calculations, the code can be used for the general purpose chemical equilibrium calculations. The code is developed at the Reactor Shielding and Data Division (RSDD) at Indira Gandhi Center for Atomic Research, India.

Recommended publication for citing

Patel, P.R., Arul, A.J., In-vessel source term calculation for a hypothetical core disruptive accident using chemical equilibrium approach for a medium sized sodium cooled fast reactor. Submitted to Nuclear Engg. & Design 25.

CHAPTER

ONE

REQUIREMENTS

MINICHEM is compatible for Python >=3.7.1. The following packages are required to run MINICHEM:

- re
- numpy
- bokeh
- scipy
- sympy
- pandas
- warning
- networkx
- selenium
- phantomjs
- holoviews
- itertools
- matplotlib

CHAPTER

TWO

THEORY

The problem of determining the equilibrium concentration of various species given a set of input elements can be formulated in terms of either entropy, Gibbs or Helmholtz function. For example, if the system is defined in terms of temperature and pressure, then minimisation of Gibbs function is appropriate since temperature and pressure are independent variables of Gibbs function [Kenneth1956] . If the system is defined in terms of temperature and volume, then minimisation of Helmholtz function is appropriate.

For a multi-phase system with $(\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_{\mathrm{N_g}}, \ldots \mathrm{x}_{\mathrm{N_g}+\mathrm{N_s}})$ moles of $(\mathrm{N_g} + \mathrm{N_s})$ species containing $\mathrm{N_e}$ elements, the Helmholtz function to be minimized is as follows:

$$\begin{aligned} \mathrm{A_{system}}(\mathrm{T}, \mathrm{V}) &= \mathrm{G} - \mathrm{PV} \\ &= \sum_{i=1}^{\mathrm{N_s}+\mathrm{N_g}} \mu_i x_i - \mathrm{PV} \end{aligned} \tag{2.1}$$

Where G is the Gibbs function. The system pressure P is in bars. Dividing Eq. (2.1) by RT, we get,

$$\tilde{\mathrm{A}}(\mathrm{T}, \mathrm{V}) = \frac{\mathrm{A_{system}}}{\mathrm{RT}} = \underbrace{\sum_{i=1}^{\mathrm{N_s}+\mathrm{N_g}} \tilde{\mu}_i x_i - \bar{x}^g}_{\mathrm{F(x)}} \tag{2.2}$$

Where $\bar{x}^g$ is the total number of moles of gas in the system. $\tilde{\mu}_i$ is the reduced (dimension less) chemical potential and can be given in terms of their standard chemical potential functions as,

$$\tilde{\mu}_i = \begin{cases} (\tilde{\mu}^o)_i^g + \ln \frac{\mathrm{RT}}{\mathrm{V}} + \ln x_i^g & \text{for } i = 1, 2 \ldots, \mathrm{N_g} \\ (\tilde{\mu}^o)_i^c & \text{for } i = 1, 2 \ldots, \mathrm{N_s} \end{cases} \tag{2.3}$$

The superscript **g** and **c** are used for gaseous and condensed phase species. For example, x_i^g is the number of moles for i^{th} chemical species in gas phase. $(\tilde{\mu}^o)_i^g$ is chemical potential of i^{th} gaseous chemical species in gas phase at standard conditions. $\mathrm{N_s}$ is the total number of the condensed species, $\mathrm{N_g}$ is the total number of the species in the gaseous phase.

While the free energy Eq. (2.2) is minimized to solve for equilibrium, the species have to satisfy non-negativity constraint $x_i \geq 0$ and the constraint for element conservation given as,

$$\underbrace{\sum_{i=1}^{\mathrm{N_g}} a_{ij}^g x_i^g + \sum_{i=1}^{\mathrm{N_s}} a_{ij}^c x_i^c = b_j}_{\mathrm{C_j(x)}} \qquad j = 1, 2, 3 \ldots, \mathrm{N_e} \tag{2.4}$$

Where a_{ij}^g is the number of atoms of j^{th} element in specie i in the gas phase and a_{ij}^c is the number of atoms of j^{th} element in i^{th} species with condensed phase. b_j is the total number of moles of element j, originally present in system mixture. The above free energy functions are minimised with two methods, viz., (1) Quadratic gradient descent method (2) sequential least square minimisation (SLSQP).

2.1 Quadratic gradient descent method:

To minimise the free energy (Eq. (2.2)) of the system containing $(\mathrm{x}_1, \mathrm{x}_2, \ldots \mathrm{x}_{\mathrm{N_g}}, \ldots, \mathrm{x}_{\mathrm{N_g}+\mathrm{N_s}})$ moles of $\mathrm{N_g}$ + $\mathrm{N_s}$ species, with constraints described in Eq. (2.4), the method of Lagrange $'$ s underdetermined multipliers is used. Here, the formulation is given for constant temperature and constant volume problem. Formulation for constant pressure and constant temperature can be found in literature. The Lagrangian function to be minimized can be written as,

$$\mathrm{L} = \tilde{\mathrm{A}} - \sum_{j=1}^{N_e} \pi_j C_j(x) \tag{2.4}$$

Where, π_j are the undetermined multipliers. The partial derivatives of L with respect to i^{th} chemical species can be given after regrouping in terms of the gas species part and condensed species part as,

$$\frac{\partial L}{\partial x_i} = \frac{\partial \mathrm{L^g}}{\partial \mathrm{x_i}} + \frac{\partial \mathrm{L^c}}{\partial \mathrm{x_i}} \tag{2.5}$$

Where, the $\frac{\partial \mathrm{L^g}}{\partial \mathrm{x_i}}$ and $\frac{\partial \mathrm{L^c}}{\partial \mathrm{x_i}}$ are given by,

$$\frac{\partial \mathrm{L^g}}{\partial \mathrm{x_i}} = (c_i + \ln x_i^g) - \sum_{j=1}^{\mathrm{N_e}} a_{ij}^g \pi_j = 0 \quad i = 1, 2 \ldots, \mathrm{N_g} \tag{2.6}$$

$$\frac{\partial \mathrm{L^c}}{\partial \mathrm{x_i}} = (\tilde{\mu}^o)_i^c - \sum_{j=1}^{\mathrm{N_e}} a_{ij}^c \pi_j = 0 \quad i = 1, 2 \ldots, \mathrm{N_s} \tag{2.7}$$

Where,

$$c_i = (\tilde{\mu}^o)_i^g + \ln \frac{\mathrm{RT}}{\mathrm{V}} \tag{2.8}$$

To linearize Eq. (2.6), Taylor series expansion about y_i^g is carried out, which results,

$$f_i + \frac{x_i^g}{y_i^g} - 1 - \sum_{j=1}^{\mathrm{N_e}} a_{ij}^g \pi_j = 0 \qquad i = 1, 2 \ldots, \mathrm{N_g} \tag{2.9}$$

Where, f_i can be given as,

$$f_i = c_i + \ln y_i^g \tag{2.9}$$

From the Taylor expanded form, the improved mole numbers for the next iteration can be obtained from the rearranged Eq. (2.9)

$$x_i^g = -f_i y_i^g + y_i^g \left(\sum_{i=1}^{\mathrm{N_g}} \pi_j a_{ij}^g + 1 \right) \tag{2.10}$$

Substituting Eq. (2.10) in Eq. (2.4) we have,

$$\sum_{k=1}^{\mathrm{N_e}} r_{jk} \pi_k + \sum_{i=1}^{\mathrm{N_s}} a_{ij}^c x_i^c = b_j + \sum_{i=1}^{\mathrm{N_g}} a_{ij}^g f_i y_i^g - \sum_{i=1}^{\mathrm{N_g}} a_{ij}^g y_i^g \qquad j = 1, 2 \ldots, \mathrm{N_e} \tag{2.11}$$

Where,

$$r_{jk} = r_{kj} = \sum_{i=1}^{\mathrm{N_g}} (a_{ij}^g a_{ik}^g) y_i^g \; j = 1, 2 \ldots, \mathrm{N_e} \tag{2.11}$$

The Eqs. (2.7) and (2.11) are solved simultaneously to get π_j and :math:x_i^c. Using π_j, updated values of x^g are obtained using Eq. (2.10). Here, it should be noted that the formulation of condensed species is such that, the moles of each condensed phase species is directly obtained from the solution of Eq. (2.7) and (2.11), without applying any correction. If all x^g values are positive, they are considered as the guessed value for the next iteration. If not, then guessed values are corrected with the following Eq.as,

$$y^p_{i,improved} = y^p_i + \lambda(x^p_i - y^p_i) \tag{2.11}$$

Where, p is the phase of chemical species (gaseous species **g** or condensed species **c**). The λ is the correction factor, and can be given according to citet{gunnar_eriksson_thermodynamic_1971},

$$\lambda = 0.99\lambda'(1 - 0.5\lambda') \tag{2.12}$$

Where, λ' is the value required for the next step to remain positive as given by,

$$\lambda' = \min_i \left(\frac{y^{g,c}_i}{(y^{g,c}_i - x^{g,c}_i)} \right) \tag{2.13}$$

The corrected values of y^g are considered for the next iteration. Since, in the above formulation, the set of condensed species is not known beforehand, for the first iteration, only gaseous species equilibrium is obtained. In subsequent iterations, species are added such that they reduce overall system $'$ s free energy. This can be assured by

$$(\tilde{\mu}^o)^c_i - \sum_{j=1}^{\mathrm{N_e}} \pi_j b_j \leq 0 \tag{2.14}$$

Sometimes with particular species, the combination can lead to a set of the dependent Eqs.in the formulation. Set of dependent species are identified with the help of reduced row echelon form. Subsequently, dependent species, as well as the matching combination of dependent species present in species set, are removed temporarily, and only species which decrease overall free energy of the system is included in species set. For example, species set containing, U (L), $\mathrm{UO_2(L)}$ and $\mathrm{U_4O_9(III)}$, leads to linear dependence and all three species are removed, and species which lower free energy of the system are included in species set.

The above Helmholtz function minimization subjected to mole number conservation constraint is implemented in python and the code is hereafter referred to as **MINICHEM** (MINImisation of CHEMical potentials). Once the equilibrium species moles in various phases are determined, the release fraction of the $\mathrm{j^{th}}$ chemical species are determined as follows:

$$\text{Elemental release fraction} = \mathrm{RF(e)_j} = \frac{\sum_i \text{Number of moles of } j^{th} \text{ element in } i^{th} \text{ gaseous species}}{\text{Total mole inventory of } j^{th} \text{ element}} \tag{2.15}$$

The isotopic release fraction is defined as,

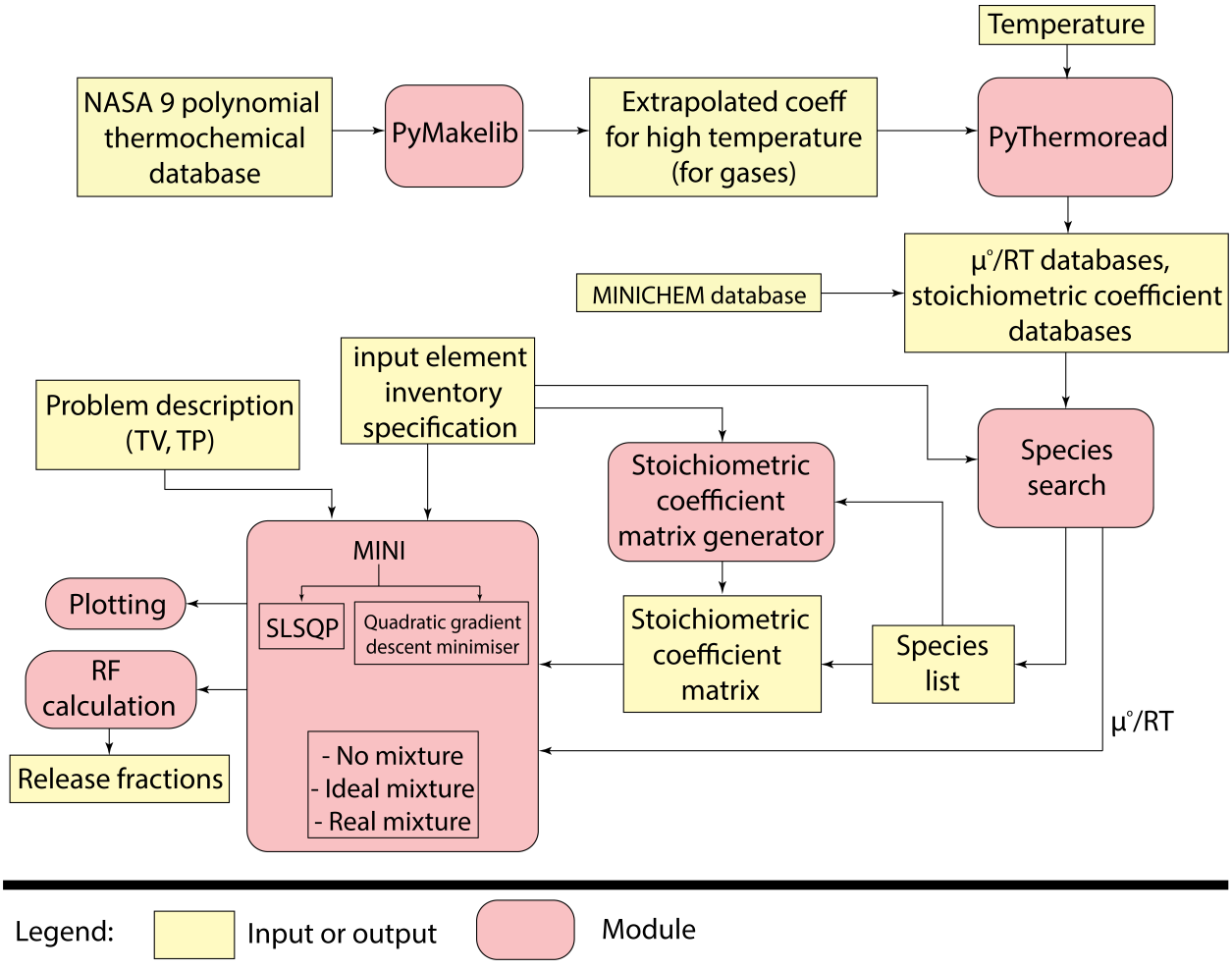
$$\text{Isotopic release fraction } RF(iso)_k = \mathrm{RF(e)}_j \times \text{isotopic fraction of } k^{th} \text{ isotope}$$

CHAPTER

THREE

CODE STRUCTURE

The MINICHEM is written in Python 3.7.1. However it is also tested with Python >= 3.7.1. Main modules of the MINICHEM are given in the Fig. **??**. Each of module is explained in the subsequent section.



3.1 Modules

3.1.1 PyMakelib

`Filename: pymakelib.py`

This module reads `NASA's CEA thermochemical database` file named `thermo.inp`. This module is translated version of the original Fortran `makelib.f` module (available in CEA code package). The data for the gas phase is extrapolated for the higher temperature. The restructured data is stored in `thermochemical_database.txt`.

3.1.2 PyThermoread

`Filename: pythermoread.py`

This module reads extrapolated thermochemical databases in `thermochemical_database.txt` and calculates the chemical potentials and stoichiometric values at the specified temperature. These calculated chemical potentials for the respective species are stored in the dictionary named as: `grt_dict` and `stoichiometric_dict`. Following are the functions are part of the pythermoread.

HRT(a1, a2, a3, a4, a5, a6, a7, b1, b2, t):
Finds the value of $\frac{H}{RT}$

Parameters

- **a1** – C_p coefficient
- **a2** – C_p coefficient
- **a3** – C_p coefficient
- **a4** – C_p coefficient
- **a5** – C_p coefficient
- **a6** – C_p coefficient
- **a7** – C_p coefficient
- **b1** – Integration coefficient
- **b2** – Integration coefficient
- **t** – Temperature (K)

Returns Returns the value of $\frac{H}{RT}$

SR(a1, a2, a3, a4, a5, a6, a7, b1, b2, t):
Finds the value of $\frac{S}{R}$

Parameters

- **a1** – C_p coefficient
- **a2** – C_p coefficient
- **a3** – C_p coefficient
- **a4** – C_p coefficient
- **a5** – C_p coefficient
- **a6** – C_p coefficient

- **a7** – C_p coefficient
- **b1** – Integration coefficient
- **b2** – Integration coefficient
- **t** – Temperature (K)

Returns Returns the value of $\frac{S}{R}$

GRT1(a1, a2, a3, a4, a5, a6, a7, b1, b2, temp):
Calculates the thermochemical potential from the specified 9 polynomial coefficients and the temperature information.

Parameters

- **a1** – C_p coefficient
- **a2** – C_p coefficient
- **a3** – C_p coefficient
- **a4** – C_p coefficient
- **a5** – C_p coefficient
- **a6** – C_p coefficient
- **a7** – C_p coefficient
- **b1** – Integration coefficient
- **b2** – Integration coefficient
- **temp** – Temperature (K)

Returns Returns thermochemical potential from the specified 9 polynomial coefficient and temperature.

thermoread():
From the `thermochemical_database.txt`, this function reads all NASA 9 polynomial thermochemical potentials for the all the chemical species and converts this database into the dictionary. This function also returns the stoichiometric data for all the thermochemical species.

Returns `thermo_dict`, `stiochemitric_dict`. Dictionary containing all NASA 9 polynomial coefficients and stoichiometric coefficient information for all chemical species specified in `thermochemical_database.txt`.

calculate_grt(grt_dict, input_temp, thermo_dict):
The function calculates the chemical potential using `thermo_dict` at specified input temperature and returns in the form of dictionary.

Parameters

- **grt_dict** – Dictionary to store the thermochemical potentials at specified temperature
- **input_temp** – input temperature at which the chemical potential to be calculated.
- **thermo_dict** – dictionary containing NASA 9 polynomial thermochemical database.

Returns `grt_dict`. Dictionary containing the chemical potentials at the specified temperature.

only_grt(grt_dict, strlist):
This function provides functionality to calculate the chemical equilibrium for the desired chemical species only. The function takes the input of the complete combination of the input element as dictionary and the list of desired species which we want to calculate the thermochemical equilibrium. The function will delete other species combination.

Parameters

- **grt_dict** – all combination of input1 from thermochem lib
- **strlist** – list of the desired elements

Returns grt_dict, updated grt_dict, which only contains $\frac{g}{RT}$ data of the desired elements which are in strlist.

3.1.3 Species search

Filename: species_search.py

In order to calculate the chemical equilibrium using the given input chemical elements/species, list of possible species which are combination of the input chemical elements/species. This module finds the combination of the input chemical elements/species from the grt_dict species list.

combination_search(species, grt_dict, combination_sp):
Searches the combination of the input elements in the grt_dict

Parameters

- **species** – List of species for which combination search will be taken out
- **grt_dict** – Dictionary of the chemical potentials
- **combination_sp** – list of combination of species

Returns combination_sp, list of species containing combination of species list.

Example If species is H2O, Then the combinations in grt_dict might be: OH, H2O2, H, O2 etc.

el(species):
The function takes the input species (list form), and convert the element of the list which can be compound/element to the element. INPUT: species (list)

3.1.4 Stoichiometric coefficient matrix generator

Filename: stoichiometric_coeff_matrix_generator.py

This module generates the stoichiometric coefficient. The module contains following functions:

stoi(species, input1, stoichiometric_dict):
Makes one row of the stoichiometry coefficient.

:param species:List of species (species containing combination of the input elements) :param input1: list of the elements provided as input. :param stoichiometric_dict: dictionary of the species with the information :returns:list of the row of the stoichiometric matrix

make_ac(input1, b, considered_sp_c, stoichiometric_dict):
Makes stoichiometry matrix for the condensed species.

Parameters

- **input1** – list of the input elements (provided by user)
- **b** – input element inventory value (provided by user)
- **considered_sp_c** – set of considered species in the condensed phase

Returns a_c condensed stoichiometry matrix

test_for_dependence(a_c, inds, input1, b, stoichiometric_dict,

sp_c, dict_of_all_sp_grt, a, pis, initial_sp_c, total_sp_c):

Sometimes the two or more dependent species in the condensed stoichiometric coefficient matrix can occurs, this can make the condensed stoichiometric coefficient matrix singular. This function checks the `a_c` matrix for the dependent row, and returns the list of the dependent rows as well as the list of the dependent species.

Parameters

- **a_c** – condensed part of stoichiometric coefficient matrix
- **inds** – Indices which are found to be dependent (calculated from the reduced row echelon form)
- **input1** – list of the input elements (provided by user)
- **b** – input element inventory value (provided by user)
- **stoichiometric_dict** – dictionary of the species with the information
- **sp_c** – list of the condensed chemical species
- **dict_of_all_sp_grt** – dictionary containing chemical potential of the all chemical species at the specified temperature
- **a** – gas part of the stoichiometric coefficient matrix
- **pis** – list of the π_i
- **initial_sp_c** – list of the all condensed species in the dict_of_all_sp_grt
- **total_sp_c** – list of the all condensed chemical species which are being considered for the equilibrium calculation

Returns updated `a_c`, updated list `sp_c`, updated list `total_sp_c`, returns the list of dependent species in the a_c matrix

3.1.5 MINI

`Filename: mini.py`

This module contains necessary functions to calculate the thermochemical equilibrium using the `Quadratic gradient descent minimisation method` and `SLSQP` method. The calculation using SLSQP method is performed using built in `scipy` module named `scipy.optimize.optimize`. The major functions `MINI` module are described below:

mini_solver(input1, b, sp_g, INSERT, total_sp_c, a, a_g, trace, dict_of_all_sp_grt, initial_sp_c, grt_dict, stoichiometric_dict, switch, temperature, v=0, pressure=1):

Determines the equilibrium species in from given input elment/species list. The function contain `sd_tv` and `sd_tp` sub-functions, which basically calculates the equilibrium species for the (T, V) and (T, P) cases respectively.

Parameters

- **input1** – list of the element in the inventory
- **b** – inventory of the elements specified as input
- **sp_g** – list of the gaseous species considered
- **INSERT** – initial list of the condensed species, from which the iteration starts. This speeds up the convergence if the several equilibrium species are known before hand.

- **total_sp_c** – list of the all condensed species considered for the equilibrium calculation
- **a** – condensed part of the stoichiometric matrix
- **a_g** – gaseous part of the stoichiometric matrix
- **trace** – min. amount of the allowed mole number
- **dict_of_all_sp_grt** – chemical potential dictionary of the all chemical species at the specified temperature.
- **initial_sp_c** – initial list of the considered condensed chemical species
- **grt_dict** – chemical potential dictionary of the all chemical specis at the specified temperature.
- **stoichiometric_dict** – dictionary consisting the stoimetric data for the all the chemical species
- **temperature** – specified temperature
- **v** – system volume

Returns y, Equilibrium mole number species wise. sp_g, list of the gaseous phase species. sp_c, list of the condensed phase species.

min_fun_helmholtz(x, species, grt_dict, temperature, v):

This function calculates helmholtz function for the guessed array x containing mole numbers at each iteration

Parameters

- **x** – array containing mole numbers
- **species** – list of species
- **grt_dict** – dictionary of chemical potential for all the chemical species
- **temperature** – specified temperature
- **v** – system volume

Returns helmholtz function value for x

gibbs_calculate(x, species, grt_dict, temperature, P):

This function calculates Gibbs function for the guessed array x containing mole numbers at each iteration

Parameters

- **x** – array containing mole numbers
- **species** – list of species
- **grt_dict** – dictionary of chemical potential for all the chemical species
- **temperature** – specified temperature
- **P** – system pressure

Returns Gibbs function value

3.1.6 RF

`Filename: rf.py`

This module takes the output equilibrium mole number array as input and calculates the release fractions in the cover gas.

rf(y, species, input1, stoichiometric_dict, el_inventory):

Takes the output mole number array and returns the dictionary with the release fraction in the cover gas.

Parameters

- **y** – output array containing mole number
- **species** – list of the species considered
- **input1** – list of the element initially considered.
- **stoichiometric_dict** – dictionary containing the stoichiometric information for the all the chemical species.
- **el_inventory** – dictionary containing the information about the input inventory specified.

Returns Prints the cover gas release fractions and writes the output in the iom.txt, released_mole_el.txt, released_sp.txt

3.1.7 Plotting

`Filename: plot_hv.py`

This module plots the sankey charts using `holoviews` module.

plot_hv(input1, stoichiometric_dict, include_el, opfilename, Min=0, Max=1e9):

Plotting module

Parameters

- **input1** – list of input elements
- **include_el** – list of element for which sankey chart is drawn
- **Min** – min mole number species to be included in chart
- **Max** – max mole number species to be included in chart
- **opfilename** – opfilename

Returns saves sankey chart

CHAPTER

FOUR

RUNNING THE CODE

Before running the code, please check *Requirements*, for the necessary python packages required to run the code. To run this program following files are required:

- chem_parse.py
- data_process.py
- mini.py
- plot_hv.py
- pythormoread.py
- rf.py
- species_search.py
- stoichiometric_coeff_matrix_generator.py

MINICHEM uses following input files.

Input files:

- thermo_chemical_database.txt
- thermo_python.in

The code generates following output files.

Output files:

- **iom.txt: Gives the released mole, RF, and details about considered species** and released species.
- released_sp.txt: Gives the released species mole
- released_mole_el.txt: el wise result of the released mole.

4.1 Defining input inventory

The input inventory can be specified by modifying the input1 list. Each list input element will be string starting with the mole number followed by the element name.

For example,

```
input1 = ['2H', '1N', '1O']
```

4.2 Defining temperature, pressure, volume parameters

For each case ((T, P) or (T, V)), the temperature (in K), P (in bars) and V (in m^3) needs to specified, even if one of these parameter is redundant.

4.3 Method

MINICHEM can calculate (T, P) or (T, V) chemical equilibrium using two method: 1. Quadratic gradient descent minimising method 2. Sequential Quadratic Programming

For this, there are two switch provided named `method` and `switch` - The valid value for the `method` is *SLSQP`or `MINI* (which calculates chemical equilibrium using *Quadratic gradient descent minimising method*) - The valid value for the `switch` is *TP`or `TV*.

4.4 Trace

By default the value of `trace` is set to 10^{-25}.

CHAPTER

FIVE

EXTENDING DATABASE

Since, the thermochemical database is fed via dictionary (`dict_of_all_sp_grt`), the thermochemical database can be easily modified by either providing NASA 9 polynomial in `thermo_dict` or directly providing chemical potentials to `dict_of_all_sp_grt` at specified temperature.

```
thermo_dict, stoichiometric_dict = pythermoread.thermoread()
dict_of_all_sp_grt = pythermoread.calculate_grt(grt_dict, temperature, thermo_dict)
```

CHAPTER

SIX

REFERENCES

BIBLIOGRAPHY

[Kenneth1956] Kenneth Denbigh, 1956. The Principles of Chemical Equilibrium, with Applications in Chemistry and Chemical Engineering., 4^{th} Edition. Cambridge University Press, Cambridge.