fO2calculate

Release 0.1.0

Kayla Iacovino

Mar 21, 2023

CONTENTS:

CHAPTER

ONE

INSTALLATION

fO2calculate can be installed with pip or pip3:

```
pip install fO2calculate
```

Always use the most up-to-date version of the code:

```
pip install fO2calculate --upgrade
```

1.1 fO2calculate Code Documentation

Contents

1.1.1 Modules

Calculate

class fO2calculate.calculate.**Calculate**(*sample*, ***kwargs*)

The Calculate object is a template for implementing user-friendly methods for running calculations using the various models implemented here. Results of the calculation are always returned by accessing self.result.

preprocess_sample(*sample*, ***kwargs*)

Creates sample object with all oxides and elements defined, even if they have 0 composition. Needed to avoid key errors when functions attempt to look up the concentration of some element or oxide.

fO2calculate.calculate.**calc_IW**(*pressure*, *temperature*)

Fe-FeO (Iron-Wustite)

Define IW buffer value at P and T

References

Campbell et al. (2009) High-pressure effects on the iron-iron oxide and nickel-nickel oxide oxygen fugacity buffers

Parameters

- **pressure** (*float*) – Pressure in bars
- **temperature** (*float*) – Temperature in degrees C

returns

- *float or numpy array* – log(fO2)
- *Polynomial coefficients*
- ———————
- *log fO2 = (a0+a1*P) + (b0+b1*P+b2*P^2+b3*P^3)/T*
- **a0** (*6.54106*)
- **a1** (*0.0012324*)
- **b0** (*-28163.6*)
- **b1** (*546.32*)
- **b2** (*-1.13412*)
- **b3** (*0.0019274*)

fO2calculate.calculate.**calc_SiSiO2**(*pressure*, *temperature*)

Si-SiO2

Define the silicon-silicon dioxide buffer value at P. Equation from Kathleen Vander Kaaden (pers. comm), with thermodynamic data taken from the JANAF tables in Chase (1998)

Parameters

- **pressure** (*float*) – Pressure in bars
- **temperature** (*float or numpy array*) – Temperature in degrees C

returns
log_fO2

rtype
float or numpy array

References

Chase, M. W. (1998). NIST-JANAF thermochemical tables. In Journal of Physical and Chemical Reference Data, 9 (1961 pp.). Gaithersburg, MD: National Institute of Standards and Technology.

Barin 1993, Phases of Silicon at High Pressure

Hu 1984, Thermochemical Data of Pure Substances (v.1 and V.2), CSEL QD 511.8 B369 1993

Fried, Howard, and Souers. EXP6: A new EOS library for HP Thermochemistry

Murnaghan parameters

Vo (ML/mol) - Fe:7.11, FeO:12.6, Si:12.06, SiO2:22.68 Bo (GPa) - Fe:139.0, FeO:142.5, Si:97.9, SiO2:27.0 Bo' - Fe:4.7, FeO:5.0, Si:4.16, SiO2:3.8

fO2calculate.calculate.**calc_activity**(*X*, *gamma*)

Returns the value of the activity of any species given the concentration of that species in mol fraction, X, and the activity coefficient gamma.

Parameters

- **X** (*float*) – Concentration of the given species in mol fraction
- **gamma** (*float*) – Activity coefficient of the given species

Returns
Activity of the given species

Return type
float

fO2calculate.calculate.**calc_dIW**(*silicate_comp*, *metal_comp*, *temperature=None*, *gammaFe='calculate'*, *gammaFeO='calculate'*, *interactions=['S', 'C', 'O', 'Ni', 'Cu', 'Si', 'Mn', 'Cr', 'Ga', 'Nb', 'Ta']*, *print_warnings=False*)

Returns fO2 in terms of delta Iron-Wustite. Calculation is performed using mole fractions and activity coefficients of Fe in the metal and FeO in the silicate.

Parameters

- **silicate_comp** (*dict*) – Dictionary of the composition of the silicate in wt% oxides.
- **metal_comp** (*dict*) – Dictionary of compositional information only for a metal, in terms of wt% elements
- **temperature** (*float*) – Temperature in degrees C.
- **gammaFe and gammaFeO** (*float*) – OPTIONAL. Default is "calculate" in which case the gammaFe or gammaFeO value will be calculated here. If the gammaFe or gammaFeO value is already known, it can be passed here to avoid having to calculate it again.
- **interactions** (*list*) – OPTIONAL. List of strings of element names. Elements are solutes in a metal Fe liquid alloy for which interaction parameters are known and for which the user wishes to calculate the effects of interaction within the alloy. Elements need not be infinitely dilute. Compositional and temperature ranges for which interaction parameters are known are given in the interaction_parameters script within this library.

Returns
logfO2 in terms of delta Iron-Wustite

Return type
float

fO2calculate.calculate.**calc_dIW_from_logfO2**(*pressure*, *temperature*, *logfO2*)

Calculates the fO2 relative to the IW buffer given the absolute fO2 value as logfO2, plus pressure and temperature

Parameters

- **pressure** (*float*) – Pressure in bars
- **temperature** (*float*) – Temperature in degrees C
- **logfO2** (*float*) – Absolute fO2 as logfO2

Returns
fO2 as deltaIW

Return type
float

fO2calculate.calculate.**calc_dSiSiO2**(*silicate_comp*, *metal_comp*, *aSiO2*, *temperature=None*, *gammaSi='calculate'*, *interactions=['S', 'C', 'O', 'Ni', 'Cu', 'Si', 'Mn', 'Cr', 'Ga', 'Nb', 'Ta']*, *print_warnings=False*)

Returns fO2 in terms of delta Si-SiO2. Calculation is performed using mole fractions and activity coefficients of Si in the metal and SiO2 in the silicate. At this time, the activity of SiO2 must be supplied. We recommend calculating this with MELTS. This is not yet implemented in this code to avoid making MELTS a dependency of this code.

Parameters

- **silicate_comp** (*dict*) – Dictionary of the composition of the silicate in wt% oxides.
- **metal_comp** (*dict*) – Dictionary of compositional information only for a metal, in terms of wt% elements
- **aSiO2** (*float*) – Activity of SiO2 in the melt.
- **temperature** (*float*) – Temperature in degrees C.
- **gammaSi** (*float*) – OPTIONAL. Default is "calculate" in which case the gammaSi value will be calculated here. If the gammaSi value is already known, it can be passed here to avoid having to calculate it again.
- **interactions** (*list*) – OPTIONAL. List of strings of element names. Elements are solutes in a metal Si liquid alloy for which interaction parameters are known and for which the user wishes to calculate the effects of interaction within the alloy. Elements need not be infinitely dilute. Compositional and temperature ranges for which interaction parameters are known are given in the interaction_parameters script within this library.

Returns
logfO2 in terms of delta Si-SiO2

Return type
float

fO2calculate.calculate.**calc_epsilon_at_temperature**(*i*, *j*, *temperature*)

Calculates the value for the interaction parameter epsilon between elements i and j at the given temperature based on a known value for e at a reference temperature.

Parameters

- **i** (*str*) – String of the name of the first of two elements for which to calculate epsilon
- **j** (*str*) – String of the name of the second of two elements for which to calculate epsilon

- **temperature** (*float*) – Temperature at which to calculate the reference gamma, in degrees C.

fO2calculate.calculate.**calc_gamma_FeO_silicate**(*silicate_comp*)

Returns a value for gammaFeO in the silicate. Parameterization is based on Holdzheid, where gammaFeO is taken as a constant value from 1.7-3, dependent only upon MgO content.

Parameters
silicate_comp (*dict*) – Dictionary of the composition of the silicate in wt% oxides.

Returns
gammaFeO in the silicate melt

Return type
float

fO2calculate.calculate.**calc_gamma_Fe_metal**(*metal_comp*, *temperature*, *interactions=['S', 'C', 'O', 'Ni', 'Cu', 'Si', 'Mn', 'Cr', 'Ga', 'Nb', 'Ta']*, *print_warnings=True*)

Calculates the activity coefficient, gamma, for iron. Interaction parameters epsilon are computed for all elements passed to interactions, so long as interaction parameter values are known.

Parameters

- **metal_comp** (*Sample object*) – Dictionary of compositional information only for a metal, in terms of wt% elements
- **temperature** (*float*) – Temperature at which to perform the calculation, in degrees C.
- **interactions** (*list*) – OPTIONAL. List of strings of element names. Elements are solutes in a metal Fe liquid alloy for which interaction parameters are known and for which the user wishes to calculate the effects of interaction within the alloy. Elements need not be infinitely dilute. Compositional and temperature ranges for which interaction parameters are known are given in the interaction_parameters script within this library.
- **print_warnings** (*bool*) – OPTIONAL. Default is True. If set to True, any warnings related to the lack of compositional data or interaction parameters will be printed.

fO2calculate.calculate.**calc_gamma_solute_metal**(*metal_comp*, *species*, *temperature*, *gammaFe='calculate'*, *interactions=['S', 'C', 'O', 'Ni', 'Cu', 'Si', 'Mn', 'Cr', 'Ga', 'Nb', 'Ta']*, *print_warnings=True*, ***kwargs*)

Calculates the activity coefficient, gamma, for any solutes in an Fe-rich metal alloy. Interaction parameters epsilon are computed for all elements passed to interactions, so long as interaction parameter values are known.

Parameters

- **metal_comp** (*dict*) – Dictionary of compositional information only for a metal, in terms of wt% elements
- **species** (*str*) – String of the name of the element for which to calculate gamma
- **temperature** (*float*) – Temperature at which to perform the calculation, in degrees C.
- **gammaFe** (*float*) – OPTIONAL. Default is "calculate" in which case the gammaFe value will be calculated here. If the gammaFe value is already known, it can be passed here to avoid having to calculate it again.
- **elements** (*list*) – OPTIONAL. List of elements for which to calculate gamma values, if that list is different than the list of interactions. Default value is None, in which case the elements list = interactions.
- **interactions** (*list*) – OPTIONAL.List of strings of element names. Elements are solutes in a metal Fe liquid alloy for which interaction parameters are known and for which the user

wishes to calculate the effects of interaction within the alloy. Elements need not be infinitely dilute. Compositional and temperature ranges for which interaction parameters are known are given in the interaction_parameters script within this library.

- **print_warnings** (*bool*) – OPTIONAL. Default is True. If set to True, any warnings related to the lack of compositional data or interaction parameters will be printed.

fO2calculate.calculate.**calc_ln_gamma_naught_at_temperature**(*species*, *temperature*)

Calculates the reference value for the activity coefficient gamma at the given temperature based on a known value for gamma at a reference temperature. NOTA BENE: if there is no tabulated value for the reference gamma, ideality will be assumed (the reference gamma will be set equal to 1).

Parameters

- **species** (*str*) – String of the name of the element for which to calculate gamma_naught
- **temperature** (*float*) – Temperature at which to calculate the reference gamma, in degrees C.

fO2calculate.calculate.**calc_logfO2_from_IW**(*pressure*, *temperature*, *dIW*)

Calculates the absolute fO2 value (as log(fO2)) based on deltaIW value, pressure, and temperature.

Parameters

- **pressure** (*float*) – Pressure in bars
- **temperature** (*float*) – Temperature in degrees C
- **dIW** (*float*) – fO2 in terms of deltaIW

Returns
log(fO2) absolute value

Return type
float

fO2calculate.calculate.**calc_logfO2_from_SiSiO2**(*pressure*, *temperature*, *dSiSiO2*)

Calculates the absolute fO2 value (as log(fO2)) based on deltaSiSiO2 value, pressure, and temperature.

Parameters

- **pressure** (*float*) – Pressure in bars
- **temperature** (*float*) – Temperature in degrees C
- **dIW** (*float*) – fO2 in terms of deltaSiSiO2

Returns
log(fO2) absolute value

Return type
float

class fO2calculate.calculate.**calculate_dIW**(*sample*, ***kwargs*)

Calculates the fO2 of a sample in terms of number of log units from the iron-wüstite buffer (dIW). Calculation is performed using mole fractions and activity coefficients of Fe in the metal and FeO in the silicate.

Parameters

- **sample** (*Sample class*) – Composition of silicate melt and metal phases as a sample object.
- **temperature** (*float*) – Temperature in degrees C.
- **interactions** (*list*) – OPTIONAL. List of strings of element names. Elements are solutes in a metal Fe liquid alloy for which interaction parameters are known and for which the user wishes to calculate the effects of interaction within the alloy. Elements need not be infinitely

dilute. Compositional and temperature ranges for which interaction parameters are known are given in the interaction_parameters script within this library.

Returns
dIW value

Return type
float

class fO2calculate.calculate.**calculate_dSiSiO2**(*sample*, ***kwargs*)

Calculates the fO2 of a sample in terms of number of log units from the Si-SiO2 buffer. Calculation is performed using mole fractions and activity coefficients of Si in the metal and SiO2 in the silicate.

Parameters

- **sample** (*Sample class*) – Composition of silicate melt and metal phases as a sample object.
- **temperature** (*float*) – Temperature in degrees C.
- **aSiO2** (*float*) – Activity of SiO2 in the melt.
- **interactions** (*list*) – OPTIONAL. List of strings of element names. Elements are solutes in a metal Fe liquid alloy for which interaction parameters are known and for which the user wishes to calculate the effects of interaction within the alloy. Elements need not be infinitely dilute. Compositional and temperature ranges for which interaction parameters are known are given in the interaction_parameters script within this library.

Returns
dSiSiO2 value

Return type
float

class fO2calculate.calculate.**calculate_gamma_Fe_metal**(*sample*, ***kwargs*)

Calculates the activity coefficient, gamma, for iron. Interaction parameters epsilon are computed for all elements passed to interactions, so long as interaction parameter values are known.

Parameters

- **sample** (*Sample class*) – Composition of silicate melt and metal phases as a sample object.
- **temperature** (*float*) – Temperature at which to perform the calculation, in degrees C.
- **interactions** (*list*) – OPTIONAL. List of strings of element names. Elements are solutes in a metal Fe liquid alloy for which interaction parameters are known and for which the user wishes to calculate the effects of interaction within the alloy. Elements need not be infinitely dilute. Compositional and temperature ranges for which interaction parameters are known are given in the interaction_parameters script within this library.
- **print_warnings** (*bool*) – OPTIONAL. Default is True. If set to True, any warnings related to the lack of compositional data or interaction parameters will be printed.

Returns
gammaFe in metal

Return type
float

class fO2calculate.calculate.**calculate_gamma_solute_metal**(*sample*, ***kwargs*)

Calculates the activity coefficient, gamma, for a solute in an Fe-rich metal alloy. Interaction parameters epsilon are computed for all elements passed to interactions, so long as interaction parameter values are known.

Parameters

- **sample** (*Sample class*) – Composition of silicate melt and metal phases as a sample object.
- **species** (*str*) – String of the name of the element for which to calculate gamma
- **temperature** (*float*) – Temperature at which to perform the calculation, in degrees C.
- **gammaFe** (*float*) – OPTIONAL. Default is "calculate" in which case the gammaFe value will be calculated here. If the gammaFe value is already known, it can be passed here to avoid having to calculate it again.
- **elements** (*list*) – OPTIONAL. List of elements for which to calculate gamma values, if that list is different than the list of interactions. Default value is None, in which case the elements list = interactions.
- **interactions** (*list*) – OPTIONAL.List of strings of element names. Elements are solutes in a metal Fe liquid alloy for which interaction parameters are known and for which the user wishes to calculate the effects of interaction within the alloy. Elements need not be infinitely dilute. Compositional and temperature ranges for which interaction parameters are known are given in the interaction_parameters script within this library.
- **print_warnings** (*bool*) – OPTIONAL. Default is True. If set to True, any warnings related to the lack of compositional data or interaction parameters will be printed.

Returns
gamma of given solute in metal. Will return 0 if gamma cannot be calculated.

Return type
float

fO2calculate.calculate.**metal_activity_from_composition**(*metal_comp*, *species*, *temperature*, *interactions=['S', 'C', 'O', 'Ni', 'Cu', 'Si', 'Mn', 'Cr', 'Ga', 'Nb', 'Ta']*)

Returns the activity of the given species in an Fe-rich metal, calculated as X times gamma.

Parameters

- **metal_comp** (*dict*) – Dictionary of compositional information only for a metal, in terms of wt% elements
- **species** (*string*) – Name of desired species for which to calculate the activity. Must match form of elements used in MetalSilicate (e.g., 'Fe', 'W', 'Ti')
- **temperature** (*float*) – Temperature at which to perform the calculation, in degrees C.
- **interactions** (*list*) – OPTIONAL. List of strings of element names. Elements are solutes in a metal Fe liquid alloy for which interaction parameters are known and for which the user wishes to calculate the effects of interaction within the alloy. Elements need not be infinitely dilute. Compositional and temperature ranges for which interaction parameters are known are given in the interaction_parameters script within this library.

Returns
Activity of the given species in an Fe-rich metal

Return type
float

sample_class

class fO2calculate.sample_class.**Sample**(*composition*, *units='wtpt'*, *default_normalization='none'*, *default_units='wtpt'*, *silence_warnings=False*)

Based on the sample_class module of VESIcal.

The sample class stores compositional information for samples, and contains methods for normalization and other compositional calculations. Designed to understand both silicate melt data and metal alloy data in a single sample. Silicate melt data must be in terms of oxides, and metal alloy data must be in terms of elements.

change_composition(*new_composition*, *units='wtpt'*, *inplace=True*)

Change the concentration of some component of the composition.

If the units are moles, they are read as moles relative to the present composition, i.e. if you wish to double the moles of MgO, if the present content is 0.1 moles, you should provide {'MgO':0.2}. The composition will then be re-normalized. If the original composition was provided in un-normalized wt%, the unnormalized total will be lost.

Parameters

- **new_composition** (*dict or pandas.Series*) – The components to be updated.
- **units** (*str*) – The units of new_composition. Should be one of: - wtpt (default) - mol
- **inplace** (*bool*) – If True the object will be modified in place. If False, a copy of the Sample object will be created, modified, and then returned.

Returns
Modified Sample class.

Return type
Sample class

check_cation(*cation*)

Check whether the sample composition contains the given cation.

Parameters
cation (*str*) – The element name to check the composition for.

Returns
Whether the composition contains the given element, or not.

Return type
bool

check_oxide(*oxide*)

Check whether the sample composition contains the given oxide.

Parameters
oxide (*str*) – Oxide name to check composition for.

Returns
Whether the composition contains the given oxide, or not.

Return type
bool

get_composition(*species=None*, *normalization=None*, *units=None*, *exclude_volatiles=False*, *asSampleClass=False*, *oxide_masses={}*, *how='combined'*, ***kwargs*)

Returns the silicate and metal composition in the format requested, normalized as requested.

Parameters

- **species** (*NoneType or str*) – The name of the oxide or cation to return the concentration of. If NoneType (default) the whole composition will be returned as a pandas.Series. If an oxide is passed, the value in wtpt will be returned unless units is set to 'mol', even if the default units for the sample object are mol. If an element is passed, the concentration will be returned as mol_cations, unless 'mol_singleO' is specified as units, even if the default units for the sample object are mol_singleO. Unless normalization is specified in the method call, none will be applied.
- **normalization** (*NoneType or str*) – The type of normalization to apply to the data. One of: - 'none' (no normalization) - 'standard' (default): Normalizes an input composition to 100%. - 'fixedvolatiles': Normalizes major element oxides to 100 wt%, including volatiles.

  The volatile wt% will remain fixed, whilst the other major element oxides are reduced proportionally so that the total is 100 wt%.

  – 'additionalvolatiles': Normalises major element oxide wt% to 100%, assuming it is volatile-free. If H2O or CO2 are passed to the function, their un-normalized values will be retained in addition to the normalized non-volatile oxides, summing to >100%.

  If NoneType is passed the default normalization option will be used (self.default_normalization).
- **units** (*NoneType or str*) – The units of composition to return, one of: - wtpt (default) - mol If NoneType is passed the default units option will be used (self.default_type).
- **exclude_volatiles bool** – If True, volatiles will be excluded from the returned composition, prior to normalization and conversion.
- **asSampleClass** (*bool*) – If True, the sample composition will be returned as a sample class, with default options. In this case any normalization instructions will be ignored.
- **oxide_masses** (*dict*) – Specify here any oxide masses that should be changed from the VESIcal default. This might be useful for recreating other implementations of models that use slightly different molecular masses. The default values in VESIcal are given to 3 dp.
- **how** (*str*) – Specify which composition to return. Either: 'combined' for both metal and silicate composition (default); 'metal' for only the metal composition; 'silicate' for only the silicate composition. Intended to be used by get_metal_composition() and get_silicate_composition() functions.

Returns
The sample composition, as specified.

Return type
pandas.Series, float, or Sample class

`get_metal_composition`(***kwargs*)

Returns only the metal composition. Inherits all arguments from get_composition()

`get_silicate_composition`(***kwargs*)

Returns only the silicate composition. Inherits all arguments from get_composition()

`set_default_normalization`(*default_normalization*)

Set the default type of normalization to use with the get_composition() method.

Parameters
default_normalization (*str*) – The type of normalization to apply to the data. One of: - 'none' (no normalization) - 'standard' (default): Normalizes an input composition to 100%. - 'fixedvolatiles': Normalizes major element oxides to 100 wt%, including volatiles.

The volatile wt% will remain fixed, whilst the other major element oxides are reduced proportionally so that the total is 100 wt%.

- 'additionalvolatiles': Normalises major element oxide wt% to 100%, assuming it is volatile-free. If H2O or CO2 are passed to the function, their un-normalized values will be retained in addition to the normalized non-volatile oxides, summing to >100%.

`set_default_units`(*default_units*)

Set the default units of composition to return when using the get_composition() method.

Parameters

default_units str – The type of composition to return, one of: - wtpt (default) - mol

batchfile

`class` fO2calculate.batchfile.**BatchFile**(*filename*, *sheet_name=0*, *file_type='excel'*, *units='wtpt'*, *label='Label'*, *default_units='wtpt'*, *default_normalization='none'*, *dataframe=None*, ***kwargs*)

Based on the batchfile class in VESIcal.

An excel file with sample names and variables. File must contain both silicate and metal data for each sample. Samples should be defined in rows, with silicates as wt% oxides and metals as wt% elements.

Variables

- **filename** (*str*) – Path to the excel file, e.g., "my_file.xlsx". This always needs to be passed, even if the user is passing a pandas DataFrame rather than an Excel file.
- **sheet_name** (*str*) – OPTIONAL. Default value is 0 which gets the first sheet in the excel spreadsheet file. This implements the pandas.read_excel() sheet_name parameter. But functionality to read in more than one sheet at a time (e.g., pandas.read_excel(sheet_name=None)) is not yet imlpemented in VESIcal. From the pandas 1.0.4 documentation:

  Available cases:

  - Defaults to 0: 1st sheet as a DataFrame
  - 1: 2nd sheet as a DataFrame
  - "Sheet1": Load sheet with name "Sheet1"
- **file_type** (*str*) – OPTIONAL. Default is 'excel', which denotes that passed file has extension .xlsx. Other option is 'csv', which denotes that the passed file has extension .csv.
- **units** (*str*) – OPTIONAL. Default is 'wtpt'. String defining whether the composition is given in wt percent ("wtpt", which is the default) or mole fraction (mol).
- **default_normalization** (*None or str*) – The type of normalization to apply to the data by default. One of: - None (no normalization) - 'standard' (default): Normalizes an input composition to 100%. - 'fixedvolatiles': Normalizes major element oxides to 100 wt%,

  including volatiles. The volatile wt% will remain fixed, whilst the other major element oxides are reduced proportionally so that the total is 100 wt%.

  - 'additionalvolatiles': Normalises major element oxide wt% to 100%, assuming it is volatile-free. If H2O or CO2 are passed to the function, their un-normalized values will be retained in addition to the normalized non-volatile oxides, summing to >100%.

- **str** (*default_units*) – The type of composition to return by default, one of: - wtpt (default) - mol
- **label** (*str*) – OPTIONAL. Default is 'Label'. Name of the column within the passed file referring to sample names.
- **dataframe** (*pandas DataFrame*) – OPTIONAL. Default is None in which case this argument is ignored. This argument is used when the user wishes to turn a pandas DataFrame into an BatchFile object, for example when user data is already in python rather than being imported from a file. In this case set *dataframe* equal to the dataframe object being passed in. If using this option, pass None to filename.

get_composition(*species=None*, *normalization=None*, *units=None*, *exclude_volatiles=False*, *asBatchFile=False*)

Returns a pandas DataFrame containing the compositional information for all samples in the BatchFile object

Parameters

- **species** (*NoneType or str*) – The name of the oxide or cation to return the concentration of. If NoneType (default) the whole composition of each sample will be returned. If an oxide is passed, the value in wtpt will be returned unless units is set to 'mol_oxides', even if the default units for the sample object are mol_oxides. If an element is passed, the concentration will be returned as mol_cations, unless 'mol_singleO' is specified as units, even if the default units for the sample object are mol_singleO. Unless normalization is specified in the method call, none will be applied.
- **normalization** (*NoneType or str*) – The type of normalization to apply to the data. One of: - 'none' (no normalization) - 'standard' (default): Normalizes an input composition to 100%. - 'fixedvolatiles': Normalizes major element oxides to 100 wt%,

  including volatiles. The volatile wt% will remain fixed, whilst the other major element oxides are reduced proportionally so that the total is 100 wt%.

  – 'additionalvolatiles': Normalises major element oxide wt% to 100%, assuming it is volatile-free. If H2O or CO2 are passed to the function, their un-normalized values will be retained in addition to the normalized non-volatile oxides, summing to >100%.

  If NoneType is passed the default normalization option will be used (self.default_normalization).
- **units** (*NoneType or str*) – The units of composition to return, one of: - wtpt (default) - mol If NoneType is passed the default units option will be used (self.default_type).
- **exclude_volatiles bool** – If True, volatiles will be excluded from the returned composition, prior to normalization and conversion.
- **asBatchFile** (*bool*) – If True, returns a BatchFile object. If False, returns a pandas.DataFrame object.

Returns
All sample information.

Return type
pandas.DataFrame or BatchFile object

get_data(*normalization=None*, *units=None*, *asBatchFile=False*)

Returns all data stored in a BatchFile object (both compositional and other data). To return only the compositional data, use get_composition().

Parameters

- **normalization** (*NoneType or str*) – The type of normalization to apply to the data. One of: - 'none' (no normalization) - 'standard' (default): Normalizes an input composition to 100%. - 'fixedvolatiles': Normalizes major element oxides to 100 wt%,

  including volatiles. The volatile wt% will remain fixed, whilst the other major element oxides are reduced proportionally so that the total is 100 wt%.

  – 'additionalvolatiles': Normalises major element oxide wt% to 100%, assuming it is volatile-free. If H2O or CO2 are passed to the function, their un-normalized values will be retained in addition to the normalized non-volatile oxides, summing to >100%.

  If NoneType is passed the default normalization option will be used (self.default_normalization).

- **units** (*NoneType or str*) – The units of composition to return, one of: - wtpt (default) - mol

  If NoneType is passed the default units option will be used (self.default_type).

- **asBatchFile** (*bool*) – If True, returns a BatchFile object. If False, returns a pandas.DataFrame object.

Returns
All sample information.

Return type
pandas.DataFrame or BatchFile object

`get_metal_composition`(***kwargs*)

Returns only the metal composition. Inherits all arguments from get_sample_composition()

`get_sample_composition`(*samplename*, *species=None*, *normalization=None*, *units=None*, *asSampleClass=False*, *how='combined'*, ***kwargs*)

Returns oxide composition of a single sample from a user-imported file as a dictionary

Parameters

- **samplename** (*string*) – Name of the desired sample
- **normalization** (*NoneType or str*) – The type of normalization to apply to the data. One of: - 'none' (no normalization) - 'standard' (default): Normalizes an input composition to 100%. - 'fixedvolatiles': Normalizes major element oxides to 100 wt%,

  including volatiles. The volatile wt% will remain fixed, whilst the other major element oxides are reduced proportionally so that the total is 100 wt%.

  – 'additionalvolatiles': Normalises major element oxide wt% to 100%, assuming it is volatile-free. If H2O or CO2 are passed to the function, their un-normalized values will be retained in addition to the normalized non-volatile oxides, summing to >100%.

  If NoneType is passed the default normalization option will be used (self.default_normalization).

- **units** (*NoneType or str*) – The units of composition to return, one of: - wtpt (default) - mol If NoneType is passed the default units option will be used (self.default_type).

- **asSampleClass** (*bool*) – If True, the sample composition will be returned as a sample class, with default options. In this case any normalization instructions will be ignored.
- **how** (*str*) – Specify which composition to return. Either: 'combined' for both metal and silicate composition (default); 'metal' for only the metal composition; 'silicate' for only the silicate composition. Intended to be used by get_metal_composition() and get_silicate_composition() functions.

Returns
Composition of the sample as oxides

Return type
dictionary, float, or sample_class.Sample object

`get_silicate_composition`(***kwargs*)

Returns only the silicate composition. Inherits all arguments from get_sample_composition()

`preprocess_sample`(*sample*)

Adds 0.0 values to any oxide data not passed.

Parameters
sample (*pandas DataFrame*) – self.data composition of samples in wt% oxides

Return type
pandas DataFrame

`save_csv`(*filenames*, *calculations*, ***kwargs*)

Saves data calculated by the user in batch processing mode to a comma-separated values (csv) file. Mirros the pandas.to_csv() method. Any argument that can be passed to pandas.csv() can be passed here. One csv file will be saved for each calculation passed.

Parameters

- **filenames** (*string or list of strings*) – Name of the file. Extension (.csv) should be passed along with the name itself, all in quotes (e.g., 'myfile.csv'). The number of calculations passed must match the number of filenames passed. If passing more than one, should be passed as a list.
- **calculations** (*pandas DataFrame or list of pandas DataFrames*) – A single variable or list of variables containing calculated outputs from any of the core BatchFile functions: calculate_dissolved_volatiles, calculate_equilibrium_fluid_comp, and calculate_saturation_pressure.

Returns

- *Creates and saves a CSV file or files with data from each*
- *calculation saved to its own file.*

`save_excel`(*filename*, *calculations*, *sheet_names=None*)

Saves data calculated by the user in batch processing mode (using the BatchFile class methods) to an organized Excel file, with the original user data plus any calculated data.

Parameters

- **filename** (*string*) – Name of the file. Extension (.xlsx) should be passed along with the name itself, all in quotes (e.g., 'myfile.xlsx').
- **calculations** (*pandas DataFrame or list of pandas DataFrames*) – A single DataFrame or list of DataFrames (e.g., calculated outputs from any of the core BatchFile functions: calculate_dissolved_volatiles, calculate_equilibrium_fluid_comp, and calculate_saturation_pressure). If None, only the original user data will be saved.

- **sheet_names** (*None, string, or list*) – OPTIONAL. Default value is None. Allows user to set the name of the sheet or sheets written to the Excel file.

Returns

- *Creates and saves an Excel file with data from each calculation*
- *saved to its own sheet.*

`set_default_normalization`(*default_normalization*)

Set the default type of normalization to use with the get_composition() method.

Parameters

default_normalization (*str*) – The type of normalization to apply to the data. One of: - 'none' (no normalization) - 'standard' (default): Normalizes an input composition to 100%. - 'fixedvolatiles': Normalizes major element oxides to 100 wt%,

including volatiles. The volatile wt% will remain fixed, whilst the other major element oxides are reduced proportionally so that the total is 100 wt%.

- 'additionalvolatiles': Normalises major element oxide wt% to 100%, assuming it is volatile-free. If H2O or CO2 are passed to the function, their un-normalized values will be retained in addition to the normalized non-volatile oxides, summing to >100%.

`set_default_units`(*default_units*)

Set the default units of composition to return when using the get_composition() method.

Parameters

default_units str – The type of composition to return, one of: - wtpt (default) - mol

`try_set_index`(*dataframe*, *label*)

Method to handle setting the index column in an BatchFile object. If no column is passed that matches the default index name, then this method will attempt to choose the 'best' column that the user might want to serve as an index column.

Parameters

- **dataframe** (*pandas DataFrame*)
- **label** (*str*) – Name of the column within the passed Excel file referring to sample names.

`fO2calculate.batchfile.`**`clean`**(*data*)

Takes a pandas dataframe (e.g. myfile.data, myfile.silicate_data) and removes any columns with all 0's, any non-numeric data.

Parameters

data (*pandas DataFrame*) – A pandas DataFrame object.

Return type

pandas DataFrame

`fO2calculate.batchfile.`**`from_DataFrame`**(*dataframe*, *units='wtpt'*, *label='Label'*)

Transforms any pandas DataFrame object into a VESIcal BatchFile object.

Parameters

- **dataframe** (*pd.DataFrame object*) – DataFrame object containing samples and oxide compositions.
- **units** (*str*) – OPTIONAL. Default is 'wtpt'. String defining whether the composition is given in wt percent ("wtpt", which is the default) or mole fraction ("mol").

- **label** (*str*) – OPTIONAL. Default is 'Label'. Name of the column within the passed file referring to sample names. This column will be set as the index column.

Return type
VESIcal.BatchFile object

class fO2calculate.batchfile.**status_bar**

Various styles of status bars that display the progress of a calculation within a loop

status_bar(*sample_name=None*, *btext=None*, *barLen=20*)

Prints an updating status bar to the terminal or jupyter notebook.

Parameters

- **percent** (*float*) – Percent value of progress from 0 to 1
- **sample_name** (*string*) – Name of the current sample being calculated
- **btext** (*string*) – Any extra text to display next to status bar
- **barLen** (*int*) – Length of bar to print

batch_calculate

class fO2calculate.batch_calculate.**BatchFile**(*filename*, *sheet_name=0*, *file_type='excel'*, *units='wtpt'*, *label='Label'*, *default_units='wtpt'*, *default_normalization='none'*, *dataframe=None*, ***kwargs*)

Performs model functions on a batchfile.BatchFile object

calculate_dIW(*temperature*, *pressure=None*, *returnlogfO2='automatic'*, *interactions=['S', 'C', 'O', 'Ni', 'Cu', 'Si', 'Mn', 'Cr', 'Ga', 'Nb', 'Ta']*, ***kwargs*)

Calculates the fO2 of all samples in a BatchFile in terms of number of log units from the iron-wüstite buffer (dIW).

Parameters

- **temperature** (*float, int, or str*) – Temperature in degrees C. Can be passed as float, in which case the passed value is used as the temperature for all samples. Alternatively, temperature information for each individual sample may already be present in the BatchFile object. If so, pass the str value corresponding to the column title in the BatchFile object.
- **pressure** (*float, int, or str*) – Pressure in bars. Only required if it is desired that the fO2 be returned both as dIW and as logfO2. Can be passed as float, in which case the passed value is used as the pressure for all samples. Alternatively, pressure information for each individual sample may already be present in the BatchFile object. If so, pass the str value corresponding to the column title in the BatchFile object.
- **returnlogfO2** (*str or bool*) – If set to True, function will return fO2 values both in terms of dIW and as logfO2. If True, a pressure must also be passed. If set to "automatic" (the default), will be set to True if a pressure is passed or set to False if no pressure is passed.
- **interactions** (*list*) – OPTIONAL. List of strings of element names. Elements are solutes in a metal Fe liquid alloy for which interaction parameters are known and for which the user wishes to calculate the effects of interaction within the alloy. Elements

need not be infinitely dilute. Compositional and temperature ranges for which interaction parameters are known are given in the interaction_parameters script within this library.

Returns
Original data passed plus newly calculated dIW values are returned.

Return type
pandas DataFrame

`calculate_dSiSiO2`(*temperature*, *aSiO2*, *pressure=None*, *returnlogfO2='automatic'*, *returndIW='automatic'*, *interactions=['S', 'C', 'O', 'Ni', 'Cu', 'Si', 'Mn', 'Cr', 'Ga', 'Nb', 'Ta']*, ***kwargs*)

Calculates the fO2 of all samples in a BatchFile in terms of number of log units from the Si-SiO2 buffer.

Parameters

- **temperature** (*float, int, or str*) – Temperature in degrees C. Can be passed as float, in which case the passed value is used as the temperature for all samples. Alternatively, temperature information for each individual sample may already be present in the BatchFile object. If so, pass the str value corresponding to the column title in the BatchFile object.
- **aSiO2** (*float, int, or str*) – Activity of SiO2 in the melt. Can be passed as float, in which case the passed value is used as the aSiO2 for all samples. Alternatively, aSiO2 information for each individual sample may already be present in the BatchFile object. If so, pass the str value corresponding to the column title in the BatchFile object.
- **pressure** (*float, int, or str*) – Pressure in bars. Only required if it is desired that the fO2 be returned both as dSiSiO2 and as logfO2. Can be passed as float, in which case the passed value is used as the pressure for all samples. Alternatively, pressure information for each individual sample may already be present in the BatchFile object. If so, pass the str value corresponding to the column title in the BatchFile object.
- **returnlogfO2** (*str or bool*) – If set to True, function will return fO2 values both in terms of dSiSiO2 and as logfO2. If True, a pressure must also be passed. If set to "automatic" (the default), will be set to True if a pressure is passed or set to False if no pressure is passed.
- **returndIW** (*str or bool*) – If set to True, function will return fO2 values both in terms of dSiSiO2 and as dIW. If True, a pressure must also be passed. If set to "automatic" (the default), will be set to True if a pressure is passed or set to False if no pressure is passed.
- **interactions** (*list*) – OPTIONAL. List of strings of element names. Elements are solutes in a metal Fe liquid alloy for which interaction parameters are known and for which the user wishes to calculate the effects of interaction within the alloy. Elements need not be infinitely dilute. Compositional and temperature ranges for which interaction parameters are known are given in the interaction_parameters script within this library.

Returns
Original data passed plus calculated dSiSiO2 values are returned.

Return type
pandas DataFrame

`calculate_gamma_Fe_metal`(*temperature*, *interactions=['S', 'C', 'O', 'Ni', 'Cu', 'Si', 'Mn', 'Cr', 'Ga', 'Nb', 'Ta']*, *print_warnings=False*, ***kwargs*)

Calculates the activity coefficient, gamma, for iron. Interaction parameters epsilon are computed for all elements passed to interactions, so long as interaction parameter values are known.

Parameters

- **temperature** (*float, int, or str*) – Temperature in degrees C. Can be passed as float, in which case the passed value is used as the temperature for all samples. Alternatively, temperature information for each individual sample may already be present in the BatchFile object. If so, pass the str value corresponding to the column title in the BatchFile object.
- **interactions** (*list*) – OPTIONAL. List of strings of element names. Elements are solutes in a metal Fe liquid alloy for which interaction parameters are known and for which the user wishes to calculate the effects of interaction within the alloy. Elements need not be infinitely dilute. Compositional and temperature ranges for which interaction parameters are known are given in the interaction_parameters script within this library.
- **print_warnings** (*bool*) – OPTIONAL. Default is True. If set to True, any warnings related to the lack of compositional data or interaction parameters will be printed.

Returns
Original data passed plus newly calculated values are returned.

Return type
pandas DataFrame

`calculate_gamma_solute_metal`(*temperature*, *species=['S', 'C', 'O', 'Ni', 'Cu', 'Si', 'Mn', 'Cr', 'Ga', 'Nb', 'Ta']*, *interactions=['S', 'C', 'O', 'Ni', 'Cu', 'Si', 'Mn', 'Cr', 'Ga', 'Nb', 'Ta']*, *gammaFe='calculate'*, *print_warnings=False*, ***kwargs*)

Calculates the activity coefficient, gamma, for multiple solutes in an Fe-rich metal alloy. Interaction parameters epsilon are computed for all elements passed to interactions, so long as interaction parameter values are known.

Parameters

- **sample** (*Sample class*) – Composition of silicate melt and metal phases as a sample object.
- **species** (*str*) – String of the name of the element for which to calculate gamma
- **temperature** (*float, int, or str*) – Temperature in degrees C. Can be passed as float, in which case the passed value is used as the temperature for all samples. Alternatively, temperature information for each individual sample may already be present in the BatchFile object. If so, pass the str value corresponding to the column title in the BatchFile object.
- **gammaFe** (*float, int, or str*) – OPTIONAL. Default is "calculate" in which case the gammaFe value will be calculated here. If the gammaFe value is already known, it can be passed here to avoid having to calculate it again. Can be passed as float or int, in which case the passed value is used as the gammaFe for all samples. Alternatively, gammaFe values for each individual sample may already be present in the BatchFile object. If so, pass the str value corresponding to the column title in the BatchFile object.
- **elements** (*list*) – OPTIONAL. List of elements for which to calculate gamma values, if that list is different than the list of interactions. Default value is None, in which case the elements list = interactions.

- **interactions** (*list*) – OPTIONAL. List of strings of element names. Elements are solutes in a metal Fe liquid alloy for which interaction parameters are known and for which the user wishes to calculate the effects of interaction within the alloy. Elements need not be infinitely dilute. Compositional and temperature ranges for which interaction parameters are known are given in the interaction_parameters script within this library.
- **print_warnings** (*bool*) – OPTIONAL. Default is True. If set to True, any warnings related to the lack of compositional data or interaction parameters will be printed.

Returns
Original data passed plus newly calculated values are returned.

Return type
pandas DataFrame

core

fO2calculate.core.**CationNum = {'Al2O3': 2, 'CO2': 1, 'CaO': 1, 'Cl': 1, 'CoO': 1, 'Cr2O3': 2, 'CuO': 1, 'F': 1, 'Fe2O3': 2, 'FeO': 1, 'Ga2O3': 2, 'H2O': 2, 'K2O': 2, 'MgO': 1, 'MnO': 1, 'MoO2': 1, 'Na2O': 2, 'Nb2O5': 2, 'NiO': 1, 'O': 0, 'P2O5': 2, 'PbO': 1, 'ReO3': 1, 'S': 1, 'SiO2': 1, 'Ta2O5': 2, 'TeO2': 1, 'ThO': 1, 'TiO2': 1, 'UO2': 1, 'V2O3': 2, 'WO3': 1, 'ZnO': 1}**

Names

exception fO2calculate.core.**Error**

Base class for exceptions in this module.

exception fO2calculate.core.**GeneralError**(*message*)

Exception raised for errors in the input.

Attributes:
expression – input expression in which the error occurred message – explanation of the error

exception fO2calculate.core.**InputError**(*message*)

Exception raised for errors in the input.

Attributes:
expression – input expression in which the error occurred message – explanation of the error

fO2calculate.core.**cations_to_oxides = {'Al': 'Al2O3', 'C': 'CO2', 'Ca': 'CaO', 'Cl': 'Cl', 'Co': 'CoO', 'Cr': 'Cr2O3', 'Cu': 'CuO', 'F': 'F', 'Fe': 'FeO', 'Fe3': 'Fe2O3', 'Ga': 'Ga2O3', 'H': 'H2O', 'K': 'K2O', 'Mg': 'MgO', 'Mn': 'MnO', 'Mo': 'MoO2', 'Na': 'Na2O', 'Nb': 'Nb2O5', 'Ni': 'NiO', 'O': 'O', 'P': 'P2O5', 'Pb': 'PbO', 'Re': 'ReO3', 'S': 'S', 'Si': 'SiO2', 'Ta': 'Ta2O5', 'Te': 'TeO2', 'Th': 'ThO', 'Ti': 'TiO2', 'U': 'UO2', 'V': 'V2O3', 'W': 'WO3', 'Zn': 'ZnO'}**

Masses

fO2calculate.core.**major_oxides = ['SiO2', 'TiO2', 'Al2O3', 'Fe2O3', 'FeO', 'MgO', 'CaO', 'Na2O', 'K2O', 'MnO', 'P2O5']**

Transformations

fO2calculate.core.**oxideMass = {'Al2O3': 101.960077, 'CO2': 44.009, 'CaO': 56.077000000000005, 'Cl': 35.45, 'CoO': 74.932194, 'Cr2O3': 151.98919999999998, 'CuO': 79.545, 'F': 18.998403163, 'Fe2O3': 159.687, 'FeO': 71.844, 'Ga2O3': 187.44299999999998, 'H2O': 18.015, 'K2O': 94.1956, 'MgO': 40.304, 'MnO': 70.937044, 'MoO2': 127.94800000000001, 'Na2O': 61.97853856000004, 'Nb2O5': 265.80773999999997, 'NiO': 74.69239999999999, 'O': 15.999, 'P2O5': 141.942523996, 'PbO': 223.19899999999998, 'ReO3': 234.204, 'S': 32.06, 'SiO2': 60.083, 'Ta2O5': 441.89076, 'TeO2': 159.59799999999998, 'ThO': 248.0367, 'TiO2': 79.865, 'UO2': 270.02691, 'V2O3': 149.88, 'WO3': 231.837, 'ZnO': 81.37899999999999}**

Other standard values

fO2calculate.core.**status_bar**(*percent*, *sample_name*, *barLen=20*)

Prints a status bar to the terminal.

percent: float
Percent value of progress from 0 to 1

barLen: int
Length of bar to print

1.2 MIT Licence

1.3 Indices and tables

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

INDEX

I

M

O

P

S

T