

Reactive mixing code documentation

Jordi Petchamé, Jesús Carrera, Francisco Batlle

Description

- The chemistry library is a system of classes aimed at performing chemical calculations on a user-defined chemical system.
- A chemical system is a set of chemical species/phases and the reactions involving them
- The information is supposed to be read from chemical databases and input files

General structure

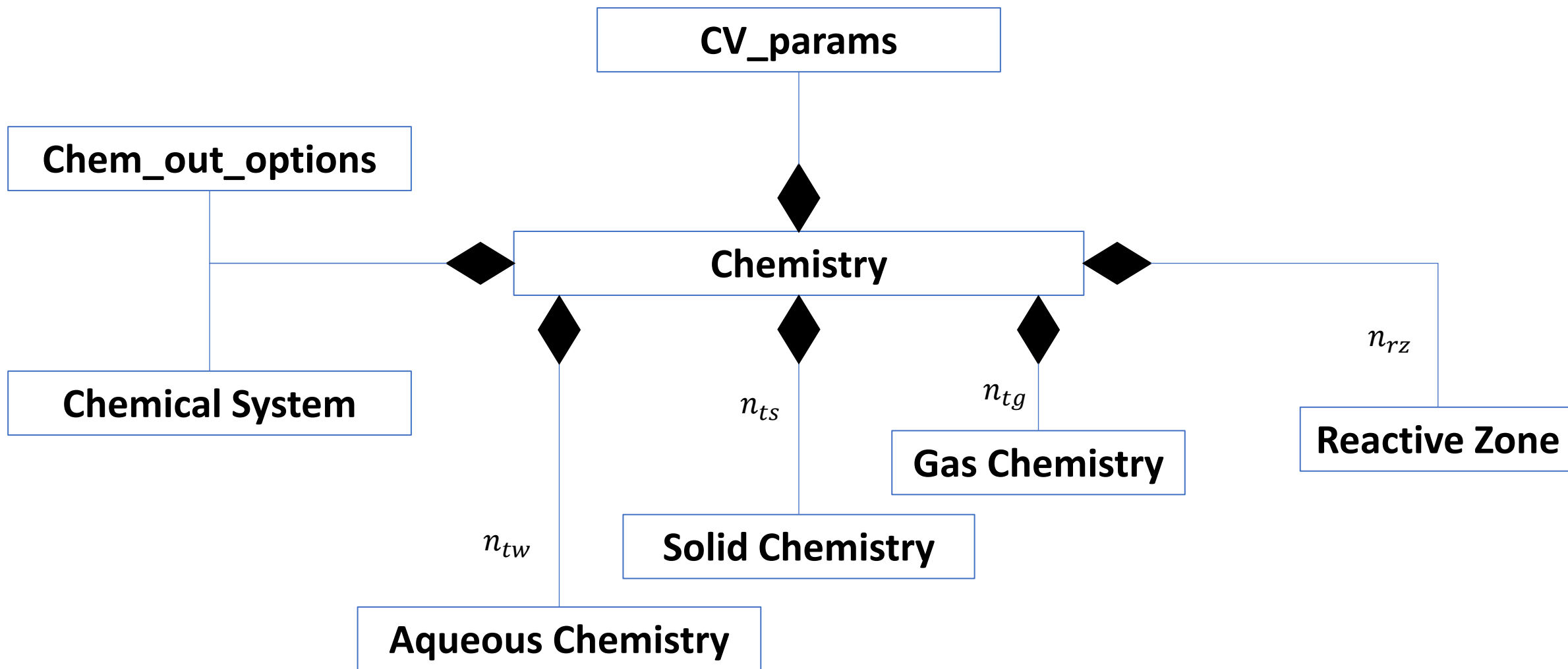
- The typical problem will involve a **Chemistry** object, created by the user, which contains a number of **Aqueous_chemistry** objects, each representing water masses of the problem (e.g., cells, or samples) and a number of **Solid_chemistry** objects (representing aggrupations of solid phases within a portion of medium) and **Gas_chemistry** objects (representing actual mixtures of gases within a portion of medium).

Expected use

- The client will create the **Chemistry** object, knowing that it contains a number of initial **Aqueous_chemistry**, **Solid_chemistry**, and **Gas_chemistry** objects, plus all the information related to chemistry (notably species and reactions), but the client knows little (or nothing) of chemistry. Still, the user will have to provide the path to read the **Chemical system, databases, etc.**
- Optionally, the client will assign these initial **Aqueous_chemistry**, **Solid_chemistry**, and **Gas_chemistry** objects, to the physical entities it is trying to model (objects **target_waters**, **target_solids**, and **target_gases**). The “target” prefix suggests that these objects are meant to represent discretized domains (i.e., cells or nodes for **target_waters**, cells or elements for **target_solids**). For reaction purposes, the client will have to assign also the solid and gas to which a given water belongs.
- The client can now operate with the target objects: speciation, reactive mixing, saturation indices, volumetric fractions, etc.

Chemistry Class

- This class is the **interface** with the client of all chemical calculations. Therefore, it contains:
 - Local information of the waters, solids and gases in the problem. This includes **concentrations, activities, activity coefficients, ionic activity, volumetric fractions of minerals, etc**
 - general **chemistry information** (species, phases, stoichiometry of reactions, equilibrium constants, kinetic parameters, etc.)
 - Main reactive mixing **solver**
 - Subroutines that perform chemical computations, such as speciation, reaction rates, conservative mixing, etc



Chemistry Class

Attributes

- Option for initialization (integer)
- Jacobian flag (integer)
- **Convergence parameters** (object of CV_params structure)
- N° target waters (n_{tw})
- N° external waters (n_{ew})
- N° domain waters (n_{dw})
- N° target solids ($n_{ts} \leq n_{tw}$)
- N° target gases (n_{tg})
- **Target waters** (vector dimension n_{tw} class **Aqueous chemistry**)
- **Initial Target waters** (vector dimension n_{tw} class **Aqueous chemistry**)
- **External waters indices** (integer vector dimension n_{ew})
- **Domain waters indices** (integer vector dimension n_{dw})
- **Target solids** (vector dimension n_{ts} class **solid chemistry**)
- **Initial Target solids** (vector dimension n_{ts} class **solid chemistry**)
- **Target gases** (vector dimension n_{tg} class **gas chemistry**)
- **Initial Target gases** (vector dimension n_{tg} class **gas chemistry**)
- N° reactive zones ($n_{rz} \leq n_{ts}$)
- **Reactive zones** (vector dim= n_{rz} **reactive zone** class)
- **Chemical System** (object of chemical system class)

$$n_{tw} = n_{ew} + n_{dw}$$

Attributes

Name	Type	Description
option	Integer	Option for reading chemical data 1. CHEPROO-based (only option currently ready) 2. PHREEQC 3. PFLOTRAN
Jac_flag	Integer	Flag for computing Jacobians 0: incremental coefficients 1: analytically
CV_params	Convergence parameters structure	Contains numerical parameters related to convergence of iterative methods

Methods

Name	Type	Arguments IN	Arguments OUT
Solve_reactive_mixing	Subroutine	<ul style="list-style-type: none">• Mixing ratios (real array class)• Mixing water indices (integer array class)• Storage matrix (real vector)• Time discretisation (object of time discretisation class)• Integration method chemical reactions (integer)	

- **Main solver**
- Computes concentrations of all species and reaction rates after reactive mixing during a given time interval

Solve_reactive_mixing

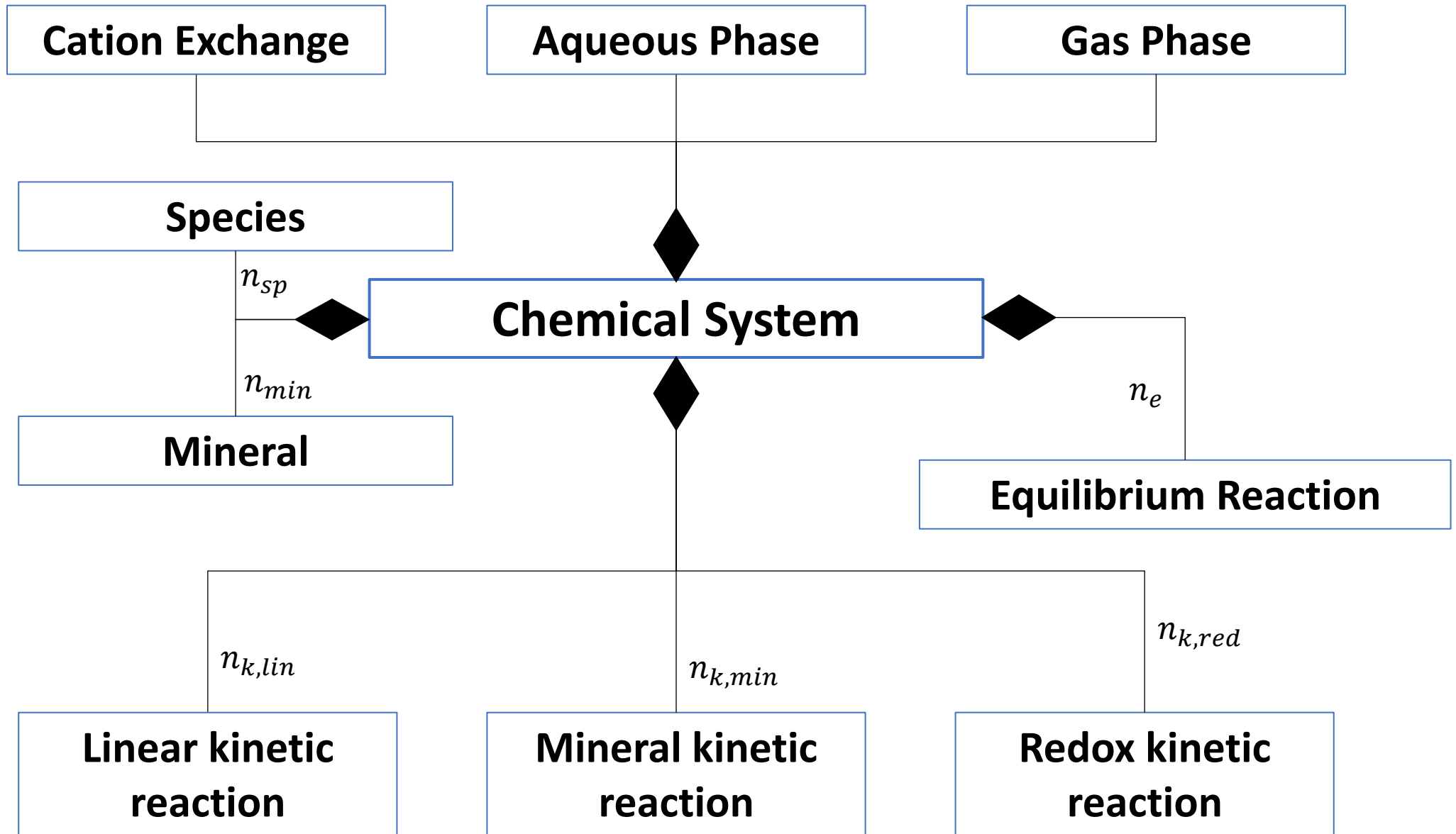
Local reactive mixing computations belong to **aqueous chemistry** class

1. Procedure pointer to choose time integration method: it will depend on the type of chemical system, the method to compute Jacobians and the integration method for chemical reactions
2. Do $k = 1, n^\circ$ time steps
 - I. Do $i = 1, n^\circ$ domain target waters
 - i. Solve mixing iteration with mixing ratios (*compute_c_tilde*)
 - ii. Solve reactive mixing iteration (with procedure pointer from 1) for i^{th} domain target water
 - iii. Compute equilibrium reaction rates (*compute_r_eq*)
 - iv. Compute state variables of minerals \longrightarrow Only if there are minerals in the reactive zone
 - v. Compute state variables of gases \longrightarrow Only if there are gases in the reactive zone

Chemical System Class

Chemical System

- It is the "book" of chemistry that contains all possible species, phases, and reactions.
- The *chemistry* class creates an object of this kind, which does the reading.
- Each *reactive zone* object will pick up the species, reactions, and stoichiometric matrices from this class.
- **We assume that all the waters contain the same aqueous species.**



Chemical System

Attributes

- N° species
- N° species constant activity
- Indices species constant activity (integer vector)
- N° species variable activity
- Indices species variable activity (integer vector)
- N° surfaces
- N° minerals
- N° minerals equilibrium
- N° Reactions
- N° equilibrium reactions
- N° homogeneous equilibrium reactions
- N° kinetic reactions
- N° linear kinetic reactions
- N° mineral kinetic reactions
- N° Monod kinetic reactions
- Squared charges (real vector)
- **Species** (Vector class Species)
- **Minerals** (vector class mineral)
- **Aqueous phase** (object aqueous phase class)
- **Gas phase** (object gas phase class)
- **Cation Exchange** (object cation exchange class)
- Stoichiometric Matrix (Real Matrix)
- Solid Stoichiometric Matrix (Real Matrix)
- Gas Stoichiometric Matrix (Real Matrix)
- **Equilibrium reactions** (vector equilibrium reaction class)
- **Kinetic reactions** (vector kinetic reaction class)
- **Linear kinetic reactions** (vector kinetic reaction class)
- **Mineral kinetic reactions** (vector mineral kinetic reaction class)
- **Redox reactions** (vector redox reaction class)
- **Speciation algebra** (object speciation algebra structure)

Species with constant activity:
Pure Minerals (1), Gases Constant Pressure, Ideal Water (1)

Local Chemistry Class

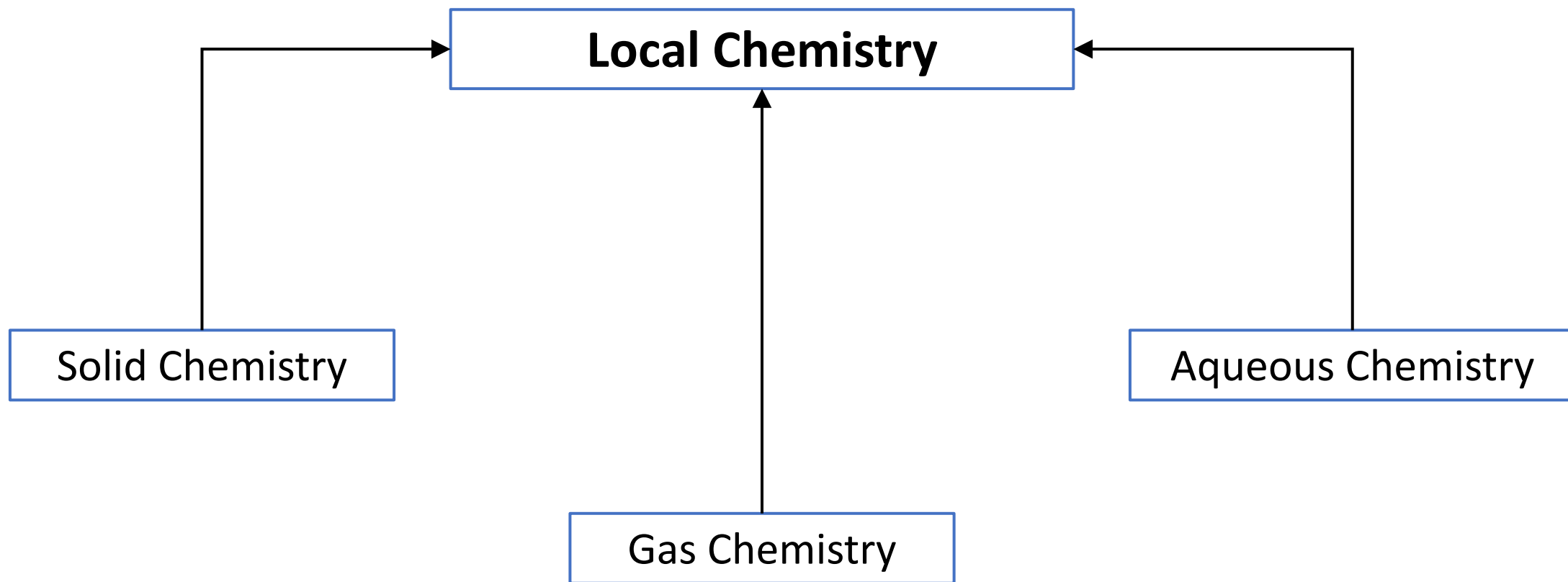
- Contains local information of a solution, such as temperature, density, pressure, concentrations, activity coefficients and their derivatives, reaction rates, etc
- **Abstract superclass**

Local Chemistry



Attributes

- Temperature (real scalar)
- density (real scalar)
- pressure (real scalar)
- Species concentrations (real vector)
- Log₁₀ activity coefficients (real vector)
- Jacobian log₁₀ activity coefficients (real matrix)
- Activities (real vector)
- equilibrium reaction rates (real vector)
- kinetic reaction rates (real vector)
- Volume (real)



Aqueous Chemistry Class

- **Subclass of local chemistry class**
- Contains concentrations & activities of aqueous species
- Contains properties of aqueous solution such as ionic activity, pH, pe and salinity
- It contains the logarithms of the activity coefficients of the aqueous species and their Jacobian.
- It is associated with a solid chemistry, a gas chemistry and an aqueous phase
- It performs the reactive mixing calculations with other objects of the same class
- It performs speciation calculations
- **Computes reaction rates**
- Reads initial water types

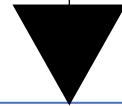
Aqueous Solution Parameters



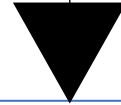
Aqueous Chemistry



Solid Chemistry



Gas Chemistry



Aqueous Phase

Aqueous Chemistry

Attributes

- **Aqueous solution parameters** (aqueous solution parameters structure object)
- Ionic activity (real scalar)
- pH (real scalar)
- pe (real scalar)
- Salinity (real scalar)
- Alkalinity (real scalar)
- **Solid chemistry** (solid chemistry class pointer)
- **Aqueous phase** (Aqueous Phase Class Pointer)
- **Gas chemistry** (gas chemistry class pointer)
- U_SkT_prod (real matrix)
- Indices aqueous phase (integer vector)

Attributes

Name	Type	Description
ionic_act	Real	Ionic activity of aqueous solution: $I = \frac{1}{2} \sum_{i=1}^{n_{aq}} m_i z_i^2$
salinity	Real	Salinity: $s = 1 + \frac{1}{TDS}$, $TDS = \sum_{i=1}^{n_{TDS}} m_i \omega_i$
alkalinity	Real	Alkalinity: $alk = [OH^-] + [HCO_3^-] + 2[CO_3^{2-}]$
U_SkT_prod	Real array	$\mathbf{US}_{K,nc}^T$

Aqueous Chemistry

Mixing Methods

- transport_iter_comp
- transport_iter_species
- transport_iter_aq_comp
- water_mixing_iter_EE_eq_kin
- water_mixing_iter_EE_kin
- water_mixing_iter_Efl_eq_kin_anal
- water_mixing_iter_Efl_kin_anal
- compute_c_tilde
- compute_u_tilde
- reaction_iteration_EE_eq_kin
- reaction_iteration_EE_kin
- compute_dfk_dc_aq_Efl
- compute_dfk_dc1_aq_Efl
- Newton_Efl_rk_eq_kin_aq_anal
- Newton_Efl_rk_kin_aq_anal

Mixing Methods

Name	Type	Arguments IN	Arguments OUT
water_mixing_iter_EE_eq_kin	Subroutine	<ul style="list-style-type: none">• \mathbf{c}_1^{k-1}• Initial guess $\mathbf{c}_{2,nc}^{k+1}$• $\tilde{\mathbf{c}}^k$• Time step Δt^k (optional)• Porosity ϕ (optional)	<ul style="list-style-type: none">• \mathbf{c}_{nc}^{k+1}

- It calculates concentrations of variable activity species and components in a reactive mixing iteration using Euler explicit.
- This subroutine is applied when there are equilibrium and kinetic reactions.

water_mixing_iter_EE_eq_kin

1. We calculate the concentration of components:

$$\mathbf{u}_j^{k+1} = \underbrace{\mathbf{U} \sum_{i \in I_j} \lambda_{ji}^k \hat{\mathbf{c}}_i^k}_{\text{Transport}} + \underbrace{\frac{\Delta t^k}{\phi_j} \mathbf{U} \mathbf{S}_{K,nc}^T \mathbf{r}_K(\mathbf{c}_j^k)}_{\text{Chemistry}} = \tilde{\mathbf{u}}_j^k + \frac{\Delta t^k}{\phi_j} \mathbf{U} \mathbf{S}_{K,nc}^T \mathbf{r}_K(\mathbf{c}_j^k)$$

Mixing ratios
Time step
Porosity

$$\hat{\mathbf{c}}^k = \langle \mathbf{c}^k | \mathbf{c}_{ext}^k | \mathbf{c}_{BC}^k \rangle$$

$$\hat{\mathbf{u}}^k = \mathbf{U} \hat{\mathbf{c}}^k$$

$\left\{ \begin{array}{l} j: \text{target (matrix column index)} \\ k: \text{time step} \end{array} \right.$

$$\left\{ \begin{array}{l} \tilde{\mathbf{c}}_j^k = \sum_{i \in I_j} \lambda_{ji}^k \hat{\mathbf{c}}_i^k \\ \tilde{\mathbf{u}}_j^k = \mathbf{U} \tilde{\mathbf{c}}_j^k \end{array} \right.$$

water_mixing_iter_EE_eq_kin

2. We speciate:

$$\left\{ \begin{array}{ll} \mathbf{f}(\mathbf{c}_{1j}^{k+1}) = \mathbf{U}_1 \mathbf{c}_{1j}^{k+1} + \mathbf{U}_{2,nc} \mathbf{c}_{2,ncj}^{k+1} - \mathbf{u}_j^{k+1} = \mathbf{0} & \longrightarrow \text{Residue} \\ \log_{10} \mathbf{c}_{2,ncj}^{k+1} = \mathbf{S}_{e,nc_1}^* \log_{10} \boldsymbol{\gamma}_{1j}^{k+1} \mathbf{c}_{1j}^{k+1} + \log_{10} \mathbf{K}^* - \log_{10} \boldsymbol{\gamma}_{2,ncj}^{k+1} & \longrightarrow \text{Mass action law} \end{array} \right.$$

$$\mathbf{U} = (\mathbf{U}_1 | \mathbf{U}_{2,nc}) \quad \mathbf{S}_{e,nc_1}^* = -\mathbf{S}_{e,nc_2}^{-1} \mathbf{S}_{e,nc_1} \quad \log_{10} \mathbf{K}^* = \mathbf{S}_{e,nc_2}^{-1} \log_{10} \mathbf{K}$$

$$\left\{ \begin{array}{l} \mathbf{c}_{1j}: \text{concentration of primary species in target } j \\ \mathbf{c}_{2,ncj}: \text{concentration of secondary species of variable activity in target } j \end{array} \right.$$

Mixing Methods

Name	Type	Arguments IN	Arguments OUT
water_mixing_iter_EE_kin	Subroutine	<ul style="list-style-type: none">• \mathbf{c}_1^{k-1}• Initial guess \mathbf{c}_2^{k+1}• $\tilde{\mathbf{c}}^k$• Time step Δt (optional)• Porosity ϕ (optional)	<ul style="list-style-type: none">• \mathbf{c}^{k+1}

- It calculates concentrations of aqueous species in a reactive mixing iteration using explicit Euler.
- This subroutine is applied when:
 - All species are aqueous
 - There are only kinetic reactions.

water_mixing_iter_EE_kin

1. We calculate the concentration of aqueous species:

$$\mathbf{c}_{aqj}^{k+1} = \underbrace{\sum_{i \in I_j} \lambda_{ji}^k \hat{\mathbf{c}}_{aqi}^k}_{\text{Transport}} + \underbrace{\frac{\Delta t^k}{\phi_j} \mathbf{s}_k^T \mathbf{r}_k(\mathbf{c}_{aqj}^k)}_{\text{Chemistry}} = \tilde{\mathbf{c}}_j^k + \frac{\Delta t^k}{\phi_j} \mathbf{s}_k^T \mathbf{r}_k(\mathbf{c}_{aqj}^k)$$

Mixing ratios
Time step

↓
↓

Porosity

$\left\{ \begin{array}{l} j: \text{target (matrix column index)} \\ k: \text{time step} \end{array} \right.$

$$\left\{ \begin{array}{l} \tilde{\mathbf{c}}_j^k = \sum_{i \in I_j} \lambda_{ji}^k \hat{\mathbf{c}}_{aqi}^k \end{array} \right.$$

Mixing Methods

Name	Type	Arguments IN	Arguments OUT
Newton_Efl_rk_eq_kin_aq_anal	Subroutine	<ul style="list-style-type: none"> • \mathbf{c}_1^{k-1} • Initial guess $\mathbf{c}_{2,nc}^{k+1}$ • $\tilde{\mathbf{c}}^k$ • Time step Δt^k • Porosity ϕ 	<ul style="list-style-type: none"> • \mathbf{c}_{nc}^{k+1} • N° iterations • Convergence flag

- Calculates variable activity species concentrations in a reactive mixing iteration using Newton's method for a time integration of Euler fully implicit in reaction rates.
- Applies Newton's method using fully implicit Euler in kinetic reactions to calculate concentrations at a target at time k+1
- This subroutine is used in the case of both equilibrium and kinetic reactions.
- The Jacobians are computed analytically.

Newton_Efl_rk_eq_kin_aq_anal

Discretisation:

$$\mathbf{u}_j^{k+1} = \tilde{\mathbf{u}}_j^k + \frac{\Delta t^k}{\phi_j} \mathbf{U} \mathbf{S}_{K,nc}^T \mathbf{r}_K(\mathbf{c}_j^{k+1})$$

$$\begin{cases} j: \text{target} \\ k: \text{time step} \end{cases}$$

Newton_Efl_rk_eq_kin_aq_anal

For each target j , in each time step $k + 1$, we use Newton to compute concentration variable activity species:

$$\left\{ \begin{array}{ll} \mathbf{f}_k \left(\mathbf{c}_{1j}^{k+1} \right) = \mathbf{U} \mathbf{c}_{ncj}^{k+1} - \tilde{\mathbf{u}}_j^k - \frac{\Delta t^k}{\phi_j} \mathbf{U} \mathbf{S}_{K,nc}^T \mathbf{r}_K \left(\mathbf{c}_j^{k+1} \right) = \mathbf{0} & \longrightarrow \text{Residue} \\ \log_{10} \mathbf{c}_{2,ncj}^{k+1} = \mathbf{S}_{e_1}^* \log_{10} \boldsymbol{\gamma}_{1j}^{k+1} \mathbf{c}_{1j}^{k+1} + \log_{10} \mathbf{K}^* - \log_{10} \boldsymbol{\gamma}_{2,ncj}^{k+1} & \longrightarrow \text{MAL} \end{array} \right.$$

Notation: subindex j
represents column j of
corresponding matrix

$$\mathbf{S}_{e_1}^* = -\mathbf{S}_{e,nc_2}^{-1} \mathbf{S}_{e_1}$$

$$\log_{10} \mathbf{K}^* = \mathbf{S}_{e,nc_2}^{-1} \log_{10} \mathbf{K}$$

$$\left\{ \begin{array}{l} \mathbf{c}_{1j}: \text{concentration primary species in target } j \\ \mathbf{c}_{2,ncj}: \text{concentration secondary variable activity species in target } j \end{array} \right.$$

Newton

- We linearise:

$$\forall i \geq 0, \quad \mathbf{0} = \mathbf{f}_k \left(\left(\mathbf{c}_{1j}^{k+1} \right)^{i+1} \right) \simeq \mathbf{f}_k \left(\left(\mathbf{c}_{1j}^{k+1} \right)^i \right) + \frac{\partial \mathbf{f}_k \left(\left(\mathbf{c}_{1j}^{k+1} \right)^i \right)}{\partial \mathbf{c}_{1j}^{k+1}} \underbrace{\left(\left(\mathbf{c}_{1j}^{k+1} \right)^{i+1} - \left(\mathbf{c}_{1j}^{k+1} \right)^i \right)}_{\left(\Delta \left(\mathbf{c}_{1j}^{k+1} \right) \right)^i}$$

Newton

- Initialisation:

$$\left(\mathbf{c}_{1j}^{k+1}\right)^0 = (1 + \mu)\mathbf{c}_{1j}^k - \mu\mathbf{c}_{1j}^{k-1}$$

parameter
↑
 $0 \leq \mu \leq 1$

- Actualisation in each iteration:

$$\left(\mathbf{c}_{1j}^{k+1}\right)^{i+1} = \left(\mathbf{c}_{1j}^{k+1}\right)^i + \left(\Delta\left(\mathbf{c}_{1j}^{k+1}\right)\right)^i$$

$\alpha \in (0,1)$: control factor

$$\left\{ \begin{array}{l} \left(\mathbf{c}_{1j}^{k+1}\right)^{i+1} \leq \alpha \left(\mathbf{c}_{1j}^{k+1}\right)^i \Rightarrow \left(\mathbf{c}_{1j}^{k+1}\right)^{i+1} = \alpha \left(\mathbf{c}_{1j}^{k+1}\right)^i \\ \left(\mathbf{c}_{1j}^{k+1}\right)^{i+1} \geq \frac{\left(\mathbf{c}_{1j}^{k+1}\right)^i}{\alpha} \Rightarrow \left(\mathbf{c}_{1j}^{k+1}\right)^{i+1} = \frac{\left(\mathbf{c}_{1j}^{k+1}\right)^i}{\alpha} \end{array} \right.$$

Newton

- Jacobian:

$$\frac{\partial \mathbf{f}_k(\mathbf{c}_{1j}^{k+1})}{\partial \mathbf{c}_{1j}^{k+1}} = \mathbf{U} \frac{\partial \mathbf{c}_{ncj}^{k+1}}{\partial \mathbf{c}_{1j}^{k+1}} - \frac{\Delta t^k}{\phi_j} \mathbf{U} \mathbf{S}_{K,nc}^T \left(\frac{\partial \mathbf{r}_K(\mathbf{c}_j^{k+1})}{\partial \mathbf{c}_{1j}^{k+1}} + \frac{\partial \mathbf{r}_K(\mathbf{c}_j^{k+1})}{\partial \mathbf{c}_{2,ncj}^{k+1}} \frac{\partial \mathbf{c}_{2,ncj}^{k+1}}{\partial \mathbf{c}_{1j}^{k+1}} \right)$$
$$\frac{\partial \mathbf{f}_k(\mathbf{c}_{1j}^{k+1})}{\partial \mathbf{c}_{1j}^{k+1}} \in \mathbb{R}^{n_p \times n_p}$$

Newton

- $\frac{\partial \mathbf{c}_{2,ncj}^{k+1}}{\partial \mathbf{c}_{1j}^{k+1}}$: solve linear system logarithms

$$\frac{\partial \log_{10} \mathbf{c}_{2,ncj}^{k+1}}{\partial \log_{10} \mathbf{c}_{1j}^{k+1}} = \mathbf{S}_{e,nc_1}^* \left(\frac{\partial \log_{10} \boldsymbol{\gamma}_{1j}^{k+1}}{\partial \log_{10} \mathbf{c}_{1j}^{k+1}} + \frac{\partial \log_{10} \boldsymbol{\gamma}_{1j}^{k+1}}{\partial \log_{10} \mathbf{c}_{2,ncj}^{k+1}} \frac{\partial \log_{10} \mathbf{c}_{2,ncj}^{k+1}}{\partial \log_{10} \mathbf{c}_{1j}^{k+1}} + \mathbf{I} \right) - \frac{\partial \log_{10} \boldsymbol{\gamma}_{2,ncj}^{k+1}}{\partial \log_{10} \mathbf{c}_{1j}^{k+1}} + \frac{\partial \log_{10} \boldsymbol{\gamma}_{2,ncj}^{k+1}}{\partial \log_{10} \mathbf{c}_{2,ncj}^{k+1}} \frac{\partial \log_{10} \mathbf{c}_{2,ncj}^{k+1}}{\partial \log_{10} \mathbf{c}_{1j}^{k+1}}$$

$$\left(\mathbf{I} - \mathbf{S}_{e,nc_1}^* \frac{\partial \log_{10} \boldsymbol{\gamma}_{1j}^{k+1}}{\partial \log_{10} \mathbf{c}_{2,ncj}^{k+1}} - \frac{\partial \log_{10} \boldsymbol{\gamma}_{2,ncj}^{k+1}}{\partial \log_{10} \mathbf{c}_{2,ncj}^{k+1}} \right) \frac{\partial \log_{10} \mathbf{c}_{2,ncj}^{k+1}}{\partial \log_{10} \mathbf{c}_{1j}^{k+1}} = \mathbf{S}_{e,nc_1}^* \left(\frac{\partial \log_{10} \boldsymbol{\gamma}_{1j}^{k+1}}{\partial \log_{10} \mathbf{c}_{1j}^{k+1}} + \mathbf{I} \right) - \frac{\partial \log_{10} \boldsymbol{\gamma}_{2,ncj}^{k+1}}{\partial \log_{10} \mathbf{c}_{1j}^{k+1}}$$

$$\frac{\partial \mathbf{c}_{2,ncj}^{k+1}}{\partial \mathbf{c}_{1j}^{k+1}} = \text{diag} \left(\mathbf{c}_{2,ncj}^{k+1} \right) \frac{\partial \log_{10} \mathbf{c}_{2,ncj}^{k+1}}{\partial \log_{10} \mathbf{c}_{1j}^{k+1}} \text{diag} \left(\frac{\mathbf{1}}{\mathbf{c}_{1j}^{k+1}} \right)$$

Algorithm for every time step

1. Initialisation concentration primary species:

$$i = 0, \quad \left(\mathbf{c}_{1j}^{k+1}\right)^i = (1 + \mu)\mathbf{c}_{1j}^k - \mu\mathbf{c}_{1j}^{k-1}, \quad 0 \leq \mu \leq 1$$

2. Given $\left(\mathbf{c}_{1j}^{k+1}\right)^i$, compute $\left(\mathbf{c}_{2,ncj}^{k+1}\right)^i$ from MAL

3. Given $\left(\mathbf{c}_{1j}^{k+1}\right)^i, \left(\mathbf{c}_{2,ncj}^{k+1}\right)^i$, compute $\frac{\partial \left(\mathbf{c}_{2,ncj}^{k+1}\right)^i}{\partial \left(\mathbf{c}_{1j}^{k+1}\right)^i}$

4. compute $\mathbf{r}_k \left(\left(\mathbf{c}_j^{k+1}\right)^i \right), \mathbf{f}_k \left(\left(\mathbf{c}_{1j}^{k+1}\right)^i \right)$.

If $\left\| \mathbf{f}_k \left(\left(\mathbf{c}_{1j}^{k+1}\right)^i \right) \right\| < \epsilon_{abs}$, stop (CV is attained).

7. compute $\frac{\partial \mathbf{r}_k \left(\mathbf{c}_j^{k+1} \right)}{\partial \mathbf{c}_{1j}^{k+1}}, \frac{\partial \mathbf{r}_k \left(\mathbf{c}_j^{k+1} \right)}{\partial \mathbf{c}_{2,ncj}^{k+1}}, \frac{\partial \mathbf{f}_k \left(\left(\mathbf{c}_{1j}^{k+1}\right)^i \right)}{\partial \mathbf{c}_{1j}^{k+1}}$

8. Solve linear system $\frac{\partial \mathbf{f}_k \left(\left(\mathbf{c}_{1j}^{k+1}\right)^i \right)}{\partial \mathbf{c}_{1j}^{k+1}} \left(\Delta \left(\mathbf{c}_{1j}^k \right) \right)^i = -\mathbf{f}_k \left(\left(\mathbf{c}_{1j}^{k+1}\right)^i \right)$

9. Check convergence: if $\left\| \frac{\left(\Delta \left(\mathbf{c}_{1j}^k \right) \right)^i}{\left(\mathbf{c}_{1j}^k \right)^i} \right\| < \epsilon_{abs}^2$, stop (CV is not attained).

Otherwise,

- a) compute $\left(\mathbf{c}_{1j}^{k+1}\right)^{i+1}$ using control factor
- b) $i = i + 1$
- c) Go to step 2

Convergence Newton

$$\left\| -\mathbf{f}_k \left(\left(\mathbf{c}_{1j}^{k+1} \right)^i \right) \right\| = \left\| \frac{\partial \mathbf{f}_k \left(\left(\mathbf{c}_{1j}^{k+1} \right)^i \right)}{\partial \mathbf{c}_{1j}^{k+1}} \Delta \left(\mathbf{c}_{1j}^{k+1} \right)^i \right\| \leq \left\| \frac{\partial \mathbf{f}_k \left(\left(\mathbf{c}_{1j}^{k+1} \right)^i \right)}{\partial \mathbf{c}_{1j}^{k+1}} \right\|_* \left\| \Delta \left(\mathbf{c}_{1j}^{k+1} \right)^i \right\| =: \eta_{k+1}^i$$

$$(\alpha - 1) \left(\mathbf{c}_{1j}^{k+1} \right)^i \leq \Delta \left(\mathbf{c}_{1j}^{k+1} \right)^i \leq \left(\frac{1}{\alpha} - 1 \right) \left(\mathbf{c}_{1j}^{k+1} \right)^i$$

$$\Rightarrow \left\| \Delta \left(\mathbf{c}_{1j}^{k+1} \right)^i \right\| \leq K_\alpha \left\| \left(\mathbf{c}_{1j}^{k+1} \right)^i \right\|, \quad K_\alpha = \max \left\{ 1 - \alpha, \frac{1}{\alpha} - 1 \right\}$$

Convergence Newton

We have to make sure that η_{k+1}^i tends to 0.

Mixing Methods

Name	Type	Arguments IN	Arguments OUT
water_mixing_iter_Efl_eq_kin_anal	Subroutine	<ul style="list-style-type: none">• \mathbf{c}_1^{k-1}• Initial guess $\mathbf{c}_{2,nc}^{k+1}$• $\tilde{\mathbf{c}}^k$• Time step Δt^k (optional)• Porosity ϕ (optional)	<ul style="list-style-type: none">• \mathbf{c}_{nc}^{k+1}

- Initialises aqueous primary species concentrations for Newton method
- Calls subroutine *Newton_Efl_rk_eq_kin_aq_anal* and changes initialisation parameter μ until Newton method converges
- Computes variable activity species concentrations

Mixing Methods

Name	Type	Arguments IN	Arguments OUT
Newton_Efl_rk_kin_aq_anal	Subroutine	<ul style="list-style-type: none">• $\tilde{\mathbf{c}}^k$• Time step Δt^k• Porosity ϕ	<ul style="list-style-type: none">• Nº iterations• Convergence flag

- Calculates aqueous species concentrations in a reactive mixing iteration using Newton's method for a time integration of Euler fully implicit in reaction rates.
- Applies Newton's method using fully implicit Euler in kinetic reactions to calculate concentrations at a target at time k+1, in a similar way as subroutine *Newton_Efl_rk_eq_kin_aq_anal*
- This subroutine is used in the case of only kinetic reactions.
- The Jacobians are computed analytically.

Mixing Methods

Name	Type	Arguments IN	Arguments OUT
water_mixing_iter_Efl_kin_anal	Subroutine	<ul style="list-style-type: none">• \mathbf{c}_1^{k-1}• Initial guess $\mathbf{c}_{2,nc}^{k+1}$• $\tilde{\mathbf{c}}^k$• Time step Δt^k (optional)• Porosity ϕ (optional)	<ul style="list-style-type: none">• \mathbf{c}_{nc}^{k+1}

- Initialises aqueous species concentrations for Newton method
- Calls subroutine *Newton_Efl_rk_kin_aq_anal* and changes initialisation parameter μ until Newton method converges

Mixing Methods

Method	Type	Description
compute_c_tilde	Function	Computes $\tilde{\mathbf{c}}_j^{k+1}$
compute_u_tilde	Function	Computes $\tilde{\mathbf{u}}_j^{k+1}$
reaction_iteration_EE_eq_kin	Subroutine	1) Calls compute_rk 2) Computes $\frac{\Delta t^k}{\phi_j} \mathbf{U} \mathbf{S}_{K,nc}^T \mathbf{r}_K(\mathbf{c}_j^k)$
reaction_iteration_EE_kin	Subroutine	1) Calls compute_rk 2) Computes $\frac{\Delta t^k}{\phi_j} \mathbf{S}_K^T \mathbf{r}_K(\mathbf{c}_j^k)$
compute_dfk_dc1_aq_Efl	Subroutine	Computes Jacobian of Newton residue $\mathbf{f}_k(\mathbf{c}_{1j}^{k+1})$
compute_dfk_dc_aq_Efl	Subroutine	Computes Jacobian of Newton residue $\mathbf{f}_k(\mathbf{c}_j^{k+1})$

Mixing Methods

Method	Type	Description
transport_iter_comp_EE	Subroutine	<ul style="list-style-type: none">• Computes component concentrations after mixing iteration and then speciates to compute variable activity concentrations.• This subroutine is used when there are aqueous and solid primary species.
transport_iter_species_EE	Subroutine	Sets aqueous concentrations after mixing iteration and computes the other aqueous state variables
transport_iter_aq_comp_EE	Subroutine	<ul style="list-style-type: none">• Computes aqueous component concentrations after mixing iteration and then speciates to compute variable activity concentrations.• This subroutine is used when there are only aqueous primary species.

Aqueous Chemistry

Speciation Methods

- `compute_c2nc_from_c1_aq_Picard`
- `compute_c2_from_c1_Picard`
- `compute_c2nc_from_c1_Picard`
- `compute_c2nc_from_c1_ideal`
- `compute_c2_from_c1_ideal`
- `compute_c_nc_from_u_aq_Newton`
- `compute_c_nc_from_u_Newton`
- `compute_c_nc_from_u_Newton_ideal`
- `compute_residual`
- `compute_dc2nc_dc1`
- `compute_dc2nc_dc1_ideal`
- `compute_dc2_dc1`
- `compute_dc2_dc1_ideal`

Speciation Methods

Name	Type	Arguments IN	Arguments OUT
compute_c2nc_from_c1_aq_Picard	Subroutine	<ul style="list-style-type: none">Initial guess $c_{2,nc}$	<ul style="list-style-type: none">$c_{2,nc}$Nº iterationsConvergence flag

- Calculates secondary variable activity species concentrations from primary species concentrations using Picard's method in the mass action law
- This subroutine is used only if all primary species are aqueous.
- This subroutine is used during the reactive mixing part

compute_c2nc_from_c1_aq_Picard

- Picard's Method:
 - Mass action law:

$$\log_{10} \mathbf{c}_{2,nc}^{i+1} = \mathbf{S}_{e,nc_1}^* \log_{10} \boldsymbol{\gamma}_1^i \mathbf{c}_1 + \log_{10} \mathbf{K}^* - \log_{10} \boldsymbol{\gamma}_{2,nc}^i, \quad i \geq 0$$

- Algorithm:
 - 1) $i \leftarrow 0$
 - 2) Assign initial guess $\mathbf{c}_{2,nc}^i$
 - 3) Compute $\log_{10} \boldsymbol{\gamma}_1^i$ y $\log_{10} \boldsymbol{\gamma}_{2,nc}^i$
 - 4) Compute $\log_{10} \mathbf{c}_{2,nc}^{i+1}$
 - 5) If $\left\| \frac{\mathbf{c}_{2,nc}^{i+1} - \mathbf{c}_{2,nc}^i}{\mathbf{c}_{2,nc}^i} \right\| < \epsilon_{rel}$, exit. The method has converged. Assign $\mathbf{c}_{2,nc} \leftarrow \mathbf{c}_{2,nc}^{i+1}$ and compute $\log_{10} \boldsymbol{\gamma}_1$ and $\log_{10} \boldsymbol{\gamma}_{2,nc}$
 - 6) $i \leftarrow i + 1$
 - 7) If $i > \text{max. n}^\circ \text{ iterations}$, exit. The method has not converged. Go to step 1 and assign another value to $\mathbf{c}_{2,nc}^{i=0}$
 - 8) Go to step 3

Speciation Methods

Name	Type	Arguments IN	Arguments OUT
compute_c2_from_c1_Picard	Subroutine	<ul style="list-style-type: none">• c_1• Initial guess c_2	<ul style="list-style-type: none">• c_2• Nº iterations• Convergence flag

- Calculates secondary species concentrations from primary species concentrations using Picard's method in the mass action law
- This subroutine is used during the initialisation part

compute_c2_from_c1_Picard

- Picard's Method:
 - Mass action law:

$$\log_{10} \mathbf{c}_2^{i+1} = \mathbf{S}_{e_1}^* \log_{10} \boldsymbol{\gamma}_1^i \mathbf{c}_1 + \log_{10} \mathbf{K}^* - \log_{10} \boldsymbol{\gamma}_2^i, \quad i \geq 0$$

- Algorithm:
 - 1) $i \leftarrow 0$
 - 2) Assign initial guess \mathbf{c}_2^i
 - 3) Compute $\log_{10} \boldsymbol{\gamma}_1^i$ y $\log_{10} \boldsymbol{\gamma}_2^i$
 - 4) Compute $\log_{10} \mathbf{c}_2^{i+1}$
 - 5) If $\left\| \frac{\mathbf{c}_2^{i+1} - \mathbf{c}_2^i}{\mathbf{c}_2^i} \right\| < \epsilon_{rel}$, exit. The method has converged. Assign $\mathbf{c}_2 \leftarrow \mathbf{c}_2^{i+1}$ and compute $\log_{10} \boldsymbol{\gamma}_1$ and $\log_{10} \boldsymbol{\gamma}_2$
 - 6) $i \leftarrow i + 1$
 - 7) If $i > \text{max. n}^\circ \text{ iterations}$, exit. The method has not converged. Go to step 1 and assign another value to $\mathbf{c}_2^{i=0}$
 - 8) Go to step 3

Speciation Methods

Name	Type	Arguments IN	Arguments OUT
compute_c_nc_from_u_aq_Newton	Subroutine	<ul style="list-style-type: none">Initial guess $\mathbf{c}_{2,nc}$\mathbf{u}	<ul style="list-style-type: none">\mathbf{c}_{nc}Nº iterationsConvergence flag

- Calculates variable activity species concentrations from component concentrations using Newton-Raphson's method.
- This subroutine is used only if all primary species are aqueous.
- This subroutine is used during the reactive mixing part
- The initial guess for the primary concentrations must be already set before calling this subroutine

compute_c_nc_from_u_aq_Newton

- Newton-Raphson:

- Residue:

$$f(\mathbf{c}_1) = \mathbf{U}_1 \mathbf{c}_1 + \mathbf{U}_{2,nc} \mathbf{c}_{2,nc} - \mathbf{u}$$

- Linealisation:

$$\forall i \geq 0, \quad \mathbf{0} = f(\mathbf{c}_1^{i+1}) \simeq f(\mathbf{c}_1^i) + \frac{\partial f(\mathbf{c}_1^i)}{\partial \mathbf{c}_1^i} \underbrace{(\mathbf{c}_1^{i+1} - \mathbf{c}_1^i)}_{\Delta \mathbf{c}_1^i}$$

- Linear system:

$$\frac{\partial f(\mathbf{c}_1^i)}{\partial \mathbf{c}_1^i} \Delta \mathbf{c}_1^i = -f(\mathbf{c}_1^i)$$

- Jacobian:

$$\frac{\partial f(\mathbf{c}_1^i)}{\partial \mathbf{c}_1^i} = \mathbf{U}_1 + \mathbf{U}_{2,nc} \frac{\partial \mathbf{c}_{2,nc}^i}{\partial \mathbf{c}_1^i}$$

compute_c_nc_from_u_aq_Newton

- Newton-Raphson:
 - Update primary concentrations:

$$\mathbf{c}_1^{i+1} = \mathbf{c}_1^i + \Delta \mathbf{c}_1^i$$

$$\alpha \in (0,1): \text{control factor} \quad \left\{ \begin{array}{l} \mathbf{c}_1^{i+1} \leq \alpha \mathbf{c}_1^i \Rightarrow \mathbf{c}_1^{i+1} = \alpha \mathbf{c}_1^i \\ \mathbf{c}_1^{i+1} \geq \frac{\mathbf{c}_1^i}{\alpha} \Rightarrow \mathbf{c}_1^{i+1} = \frac{\mathbf{c}_1^i}{\alpha} \end{array} \right.$$

compute_c_nc_from_u_aq_Newton

- Algorithm:

- 1) $i \leftarrow 0$
- 2) Compute $\mathbf{c}_{2,nc}^i$ using the Mass Action Law (**compute_c2nc_from_c1_aq**)
- 3) Compute residue $f(\mathbf{c}_1^i)$ (**compute_residual**)
- 4) If $\|f(\mathbf{c}_1^i)\| < \epsilon_{abs}$, exit. The method has converged.
- 5) Compute Jacobian secondary species $\frac{\partial \mathbf{c}_{2,nc}^i}{\partial \mathbf{c}_1^i}$ (**compute_dc2nc_dc1**)
- 6) Calculate Jacobian residue $\frac{\partial f(\mathbf{c}_1^i)}{\partial \mathbf{c}_1^i}$
- 7) Solve linear system $\frac{\partial f(\mathbf{c}_1^i)}{\partial \mathbf{c}_1^i} \Delta \mathbf{c}_1^i = -f(\mathbf{c}_1^i)$
- 8) If $\left\| \frac{\Delta \mathbf{c}_1^i}{\mathbf{c}_1^i} \right\| < \epsilon_{abs}^2$, exit. The method has NOT converged.
- 9) Compute \mathbf{c}_1^{i+1} with control factor (**update_conc_prim_species**)
- 10) $i \leftarrow i + 1$
- 11) Go to step 2

Aqueous Solution Parameters Structure

- It contains parameters related to properties of an aqueous solution

Aqueous Solution Parameters

```
graph TD; A["Aqueous Solution Parameters"] --- B["Attributes"]; A --- C["Methods"];
```

Attributes

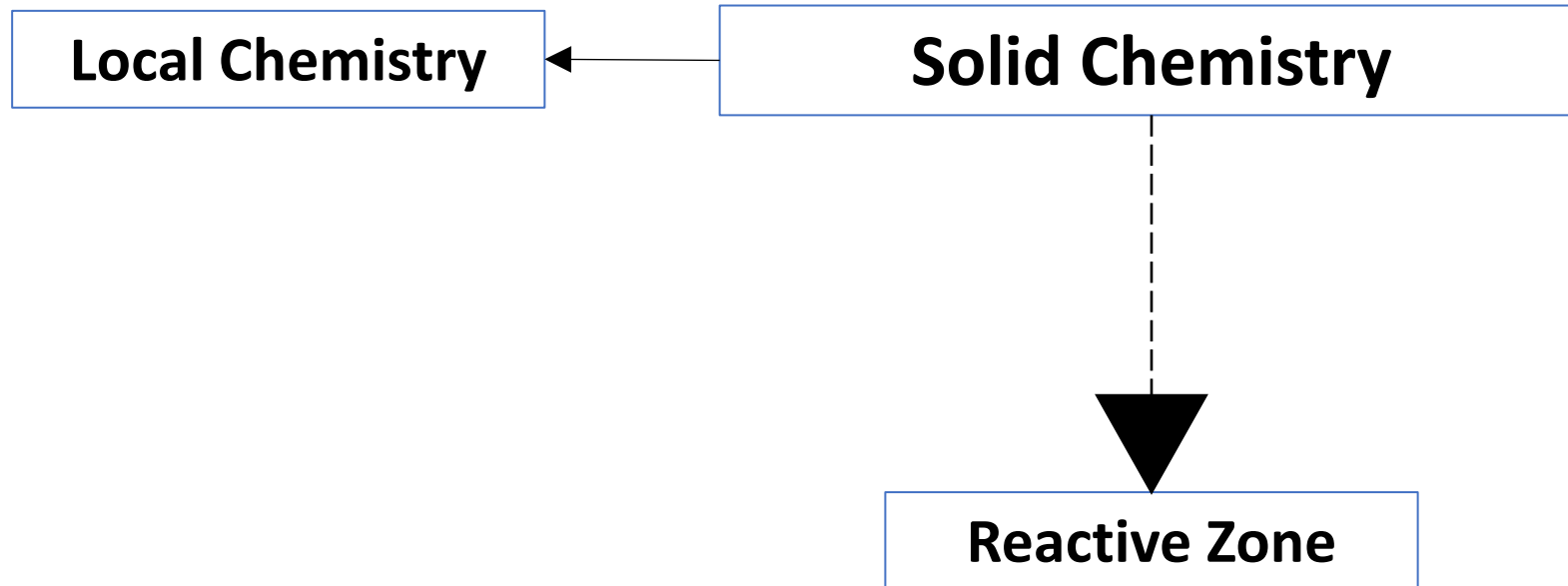
- A (=0.5092 a 25 °C)
- B (=0.3283 a 25 °C)

Methods

- Set

Class: Solid Chemistry

- **Subclass of local chemistry class**
- Contains state variables of a solid zone (equivalents of surface complexes, volumetric fractions and reactive surfaces of minerals, and the cation exchange capacity of a surface)
- It is associated with a **reactive zone** that contains the minerals and surface complexes in the solid zone
- Computes solid state variables
- Reads initial mineral zones



Solid Chemistry

```
graph TD; A[Solid Chemistry] --> B[Own Attributes]; A --> C[Computation Methods];
```

Own Attributes

- Equivalents (real vector)
- Volumetric fractions (real vector)
- Reactive surfaces (real vector)
- Cation exchange capacity (real scalar)
- **Reactive Zone** (pointer to reactive zone class)

Computation Methods

- compute_activities_solids
- compute_equivalents
- compute_conc_minerals_iter
- compute_mass_bal_mins

Methods

Subroutine	Description
compute_conc_minerals_iter	Computes concentration of minerals after a given time step

$\begin{cases} i: \text{mineral index} \\ k: \text{time step} \end{cases}$

$$c_i^{k+1} = c_i^k + \frac{\Delta t^k}{\phi_i} \langle \mathbf{s}_i, \mathbf{r}_{e,min} \rangle$$

Column of equilibrium stoichiometric matrix
corresponding to i^{th} mineral

Dot product

Mineral equilibrium
reaction rates

volumetric fraction

Methods

Subroutine	Description
compute_mass_bal_mins	Computes volumetric fractions of minerals after a given time step

$$\phi_i^{k+1} = \phi_i^k + \Delta t^k V_i r_i$$

volumetric fraction

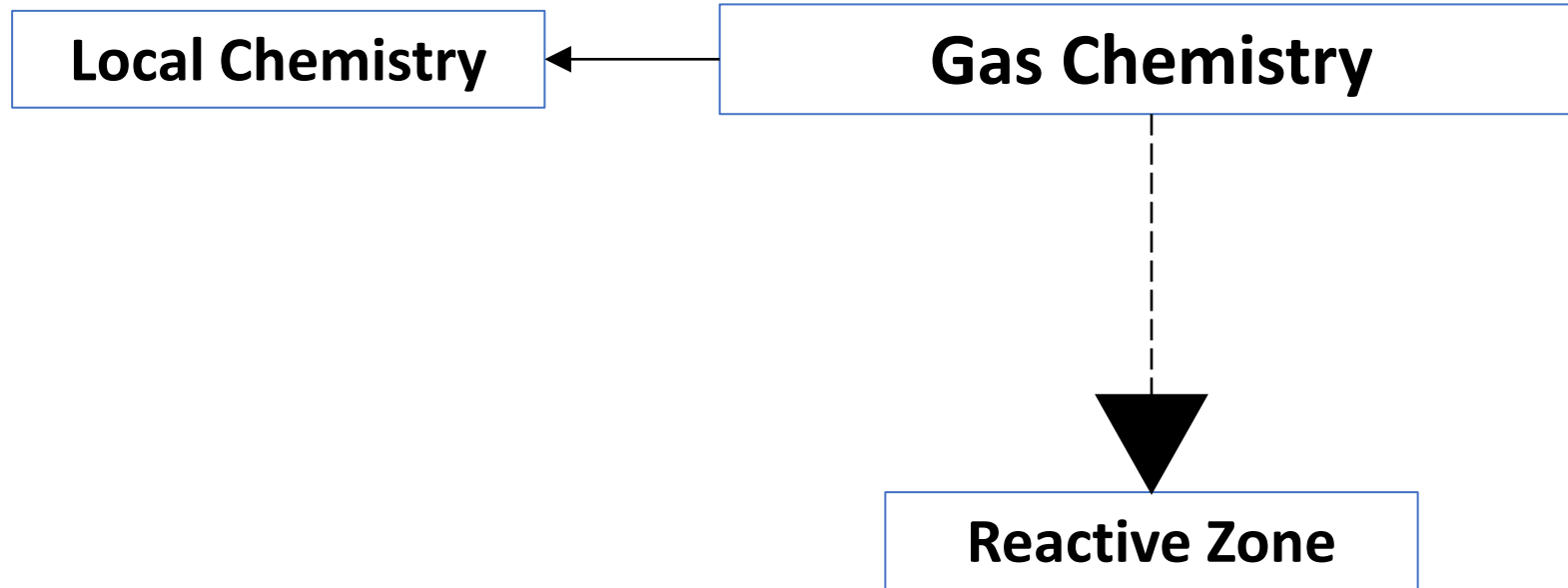
Molar volume

Mineral equilibrium reaction rate

$\begin{cases} i: \text{mineral index} \\ k: \text{time step} \end{cases}$

Class: Gas Chemistry

- **Subclass of local chemistry class**
- It is associated with a **reactive zone**, which contains the gases in the **gas phase**
- **Activities are partial pressures [atm]**
- **Concentrations are number of moles**
- Computes concentration, partial pressure and volume of gases
- Reads initial & boundary gas zones



Gas Chemistry

```
graph TD; GC[Gas Chemistry] --- OA[Own Attributes]; GC --- CM[Computation Methods];
```

Own Attributes

- **Reactive Zone** (pointer to reactive zone class)

Computation Methods

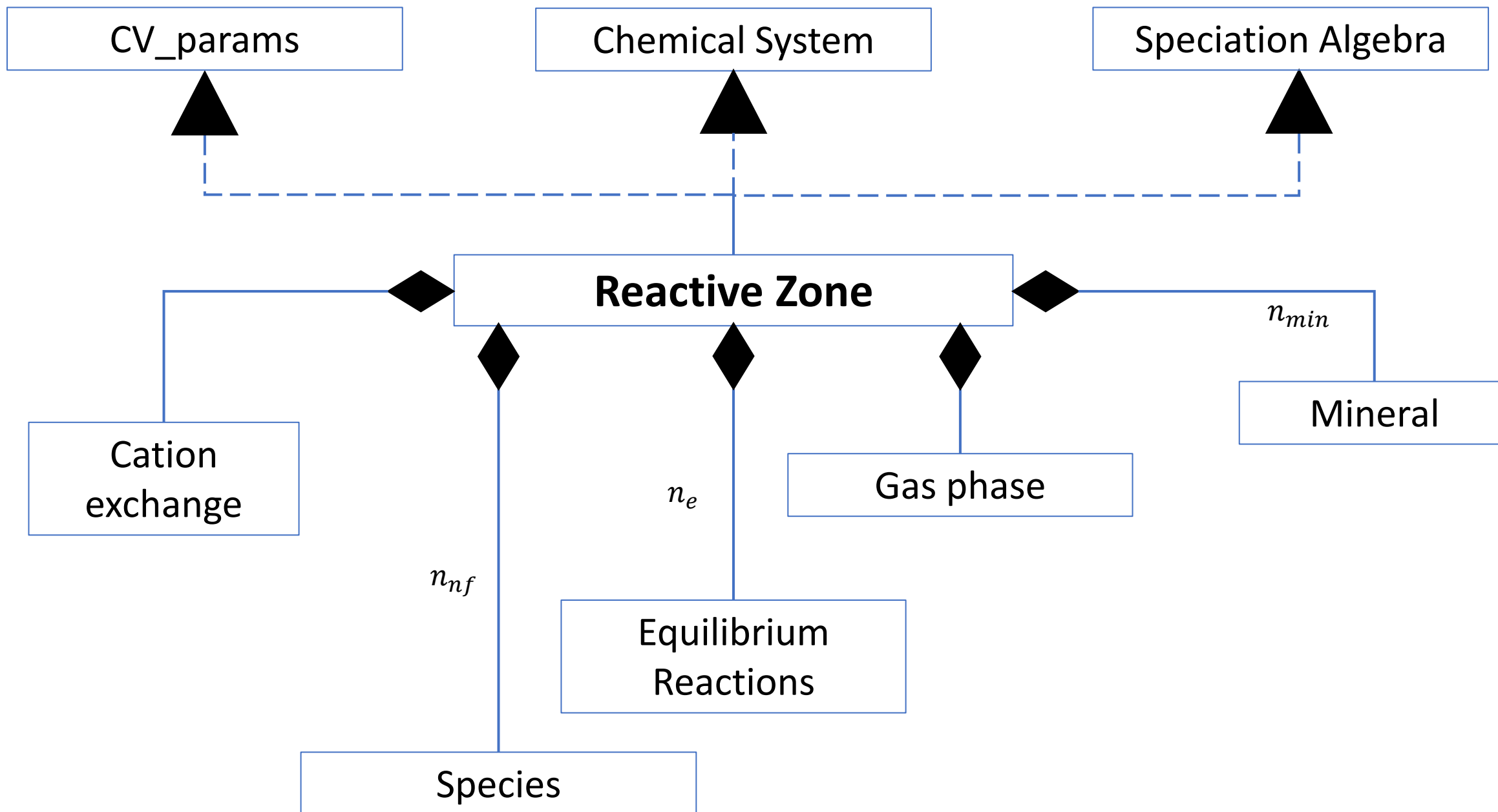
- Compute_conc_gases_ideal
- compute_conc_gases_iter
- compute_vol_gas
- compute_partial_pressures
- compute_pressure

Methods

Subroutine	Description
compute_conc_gases_ideal	<p>Computes number of moles n_i of each gas in gas phase using ideal gas equation</p> $p_i V_T = n_i R T$ <p>where i is the gas index</p>
compute_conc_gases_iter	<p>Computes number of moles of each gas in gas phase after an iteration of Euler explicit method using gas conservation equation:</p> $n_i^{k+1} = n_i^k + R_i V_w$ <p>where R_i is the i^{th} gas equilibrium reaction amount and V_w is the water volume</p>
compute_vol_gas	<p>Computes total volume of gas:</p> $V_T = \frac{n_T R T}{p_T}$ <p>where the subscript T means total</p>
compute_partial_pressures	<p>Computes partial pressure p_i of each gas in gas phase using ideal gas equation</p> $p_i = \frac{n_i R T}{V_T}$
compute_pressure	Computes sum of partial pressures

Class: Reactive Zone

- It contains the “**immobile**” **species** and **heterogeneous equilibrium reactions** that define a solid and/or gas chemistry.
- The *solid chemistry* and *gas chemistry* classes create objects of this class.
- It contains objects of the species class, the equilibrium reaction class, the gas phase class, the cation exchange class and the mineral class
- It contains a pointer to the *speciation algebra* structure, the *CV_params* structure and the *chemical system* class
- The *CV_params* pointer will depend on the type of equilibrium reactions involved
- It will obtain the species and reactions from the chemical system that it points to
- It will pass to the *speciation algebra* pointer the stoichiometric matrix and equilibrium constants



Reactive Zone

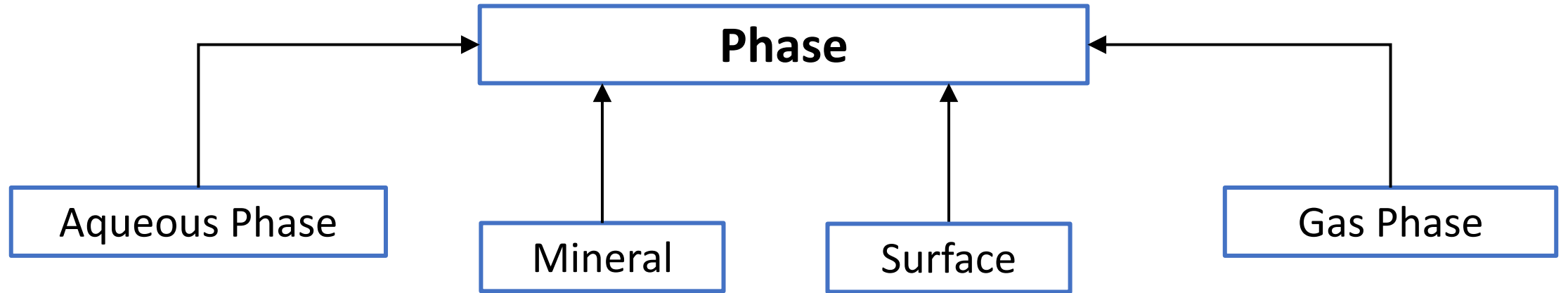
Attributes

- N° non-flowing species ($n_{nf} \in \mathbb{Z}_+$)
- **Non-flowing species** (vector class species)
- N° minerals ($n_{min} \in \mathbb{Z}_+$)
- **Minerals** (vector class mineral)
- N° solids (integer)
- Stoichiometric matrix (real matrix)
- Solid stoichiometric matrix (real matrix)
- **Chemical system** (pointer to chemical system class)
- **Speciation algebra** (pointer to speciation algebra structure)
- **CV_params** (pointer to CV_params class)
- **Cation Exchange zone** (object type cation exchange)
- **Gas phase** (object type gas phase)
- N° equilibrium reactions ($n_e \in \mathbb{Z}_+$)
- **Equilibrium reactions** (vector equilibrium reaction class)

Phase Class

- **Abstract superclass**
- Contains properties of a phase
- Contains inheritance depending on the type of phase

Phase Class



Phase



```
classDiagram
    class Phase {
        Attributes:
            Name (string)
            Nº species (integer)
            Nº variable activity species (integer)
            Nº constant activity species (integer)
        Methods:
            Set
    }
    class Attributes {
        Name (string)
        Nº species (integer)
        Nº variable activity species (integer)
        Nº constant activity species (integer)
    }
    class Methods {
        Set
    }
    Phase --> Attributes
    Phase --> Methods
```

The diagram shows a class named 'Phase' at the top center. Two lines extend downwards from the 'Phase' box to two separate boxes below it. The left box is titled 'Attributes' and contains a bulleted list of four items: 'Name (string)', 'Nº species (integer)', 'Nº variable activity species (integer)', and 'Nº constant activity species (integer)'. The right box is titled 'Methods' and contains a single bulleted item: 'Set'.

Attributes

- Name (string)
- Nº species (integer)
- Nº variable activity species (integer)
- Nº constant activity species (integer)

Methods

- Set

Aqueous Phase Class

- **Subclass of phase class.**
- It contains the aqueous species that are transported and their squared charges.
- **We assume the aqueous species are the same in all the waters.**
- Computes the base 10 logarithms of the activity coefficients of aqueous species and their Jacobian. The information is provided by the aqueous chemistry class.
- Contains indices of some aqueous species, such as H_2O , H^+ or HCO_3^- .

Aqueous Phase



n_{aq}

Aqueous Species

Aqueous Phase

Own Attributes

- **Aqueous species** (vector class aqueous species)
 - **Squared charges (real vector)**
 - Indices dissolved solids (integer vector)
 - N° aqueous complexes
 - Water flag
 - Index water
 - Index proton
 - Index OH^-
 - Index carbonate
 - Index bicarbonate
- } integers

Own Methods

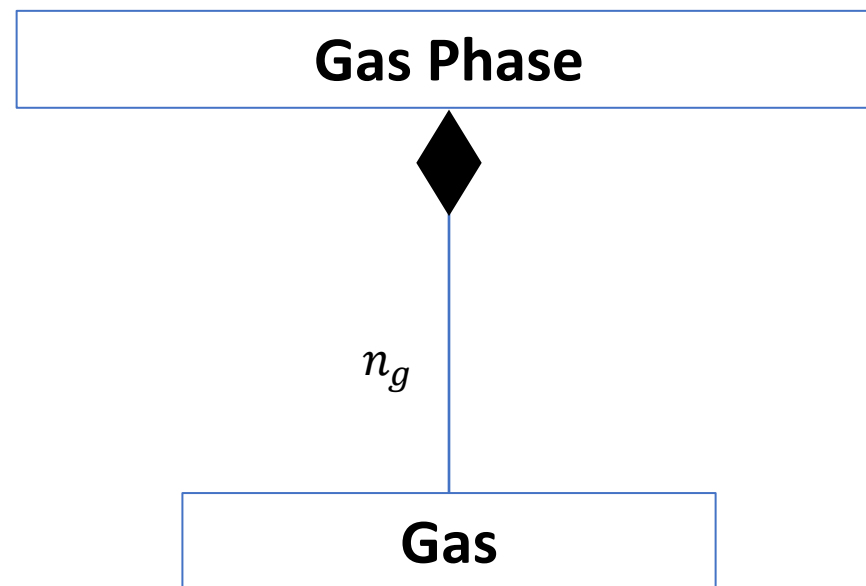
- Compute:
 - log₁₀ activity coefficients
 - log₁₀ Jacobian activity coefficients
 - **Squared charges**
- Rearrange:
 - Aqueous species

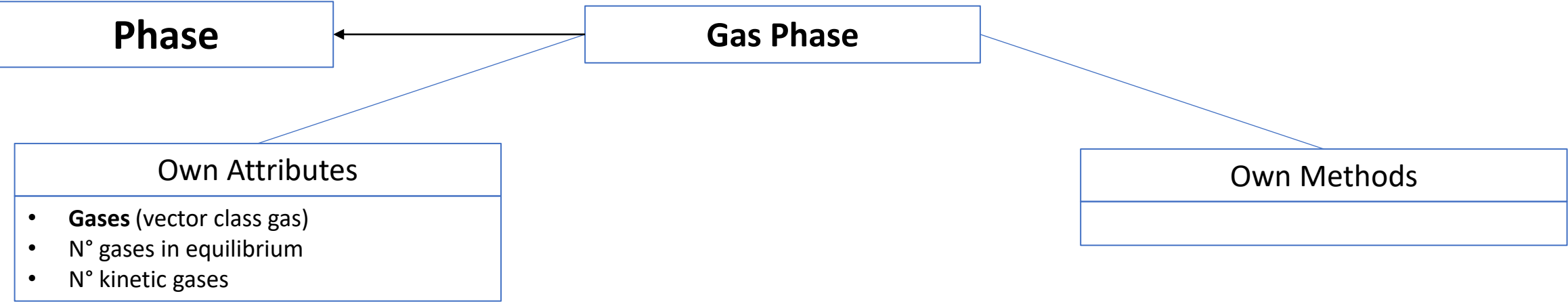
Methods

Subroutine	Arguments IN	Arguments OUT	Description
Compute_log_act_coeffs_aq_phase	<ul style="list-style-type: none"> Ionic activity Aqueous solution parameters object 	<ul style="list-style-type: none"> $\log_{10} \gamma_{aq,nc}$ 	It calculates the base 10 logarithms of the activity coefficients of the variable activity aqueous species.
Compute_log_Jacobian_act_coeffs_aq_phase	<ul style="list-style-type: none"> Outer product between $\frac{d \log_{10} \gamma_{aq,nc}}{dI}$ and $z_{aq,nc}^2$ Aqueous variable activity species concentration 	<ul style="list-style-type: none"> $\frac{\partial \log_{10} \gamma_{aq,nc}}{\partial \log_{10} c_{aq,nc}}$ 	Computes the partial derivatives of the base 10 logarithms of the activity coefficients with respect to the base 10 logarithms of the concentrations of the aqueous species of variable activity.

Gas Phase Class

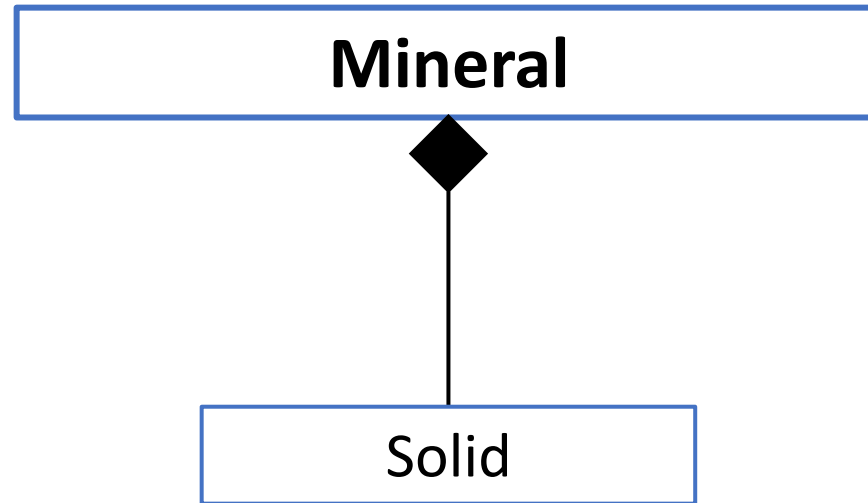
- **Subclass of phase class**
- Contains the gases in the chemical system, which we assume are the same in the whole domain.
- Contains the number of gases in equilibrium and in kinetic reactions.
- Computes log 10 activity coefficients and its log 10-Jacobian.
- **Transported gases must also be included**

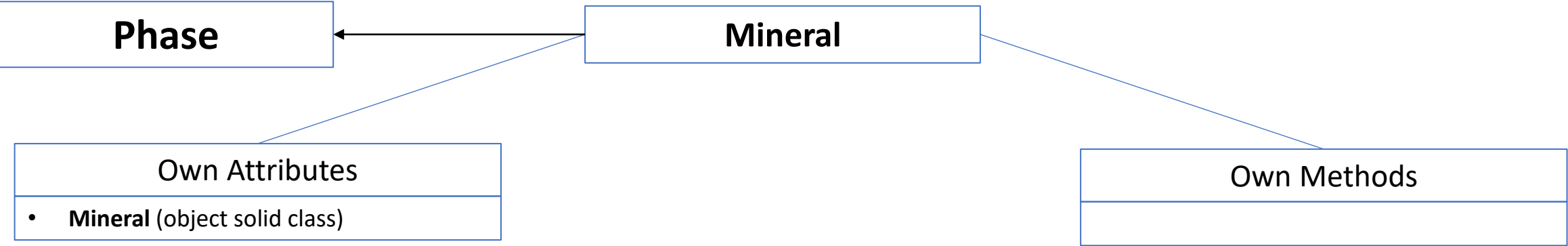




Mineral Class

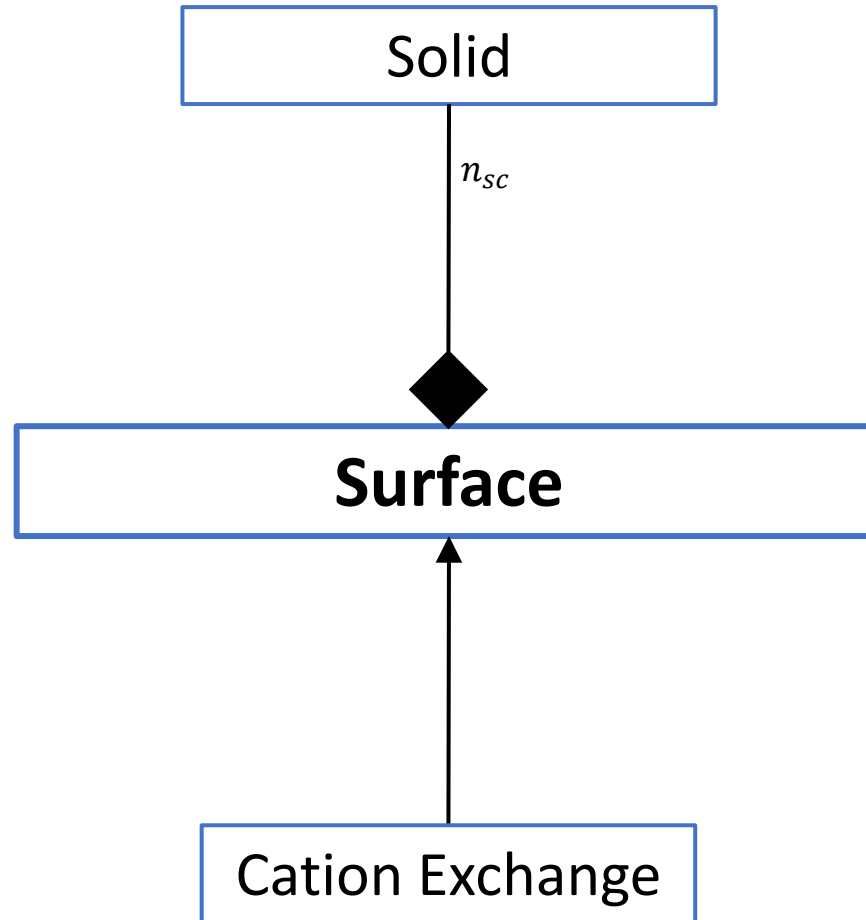
- **Subclass of phase class.**
- Contains the mineral that defines a mineral phase (es redundante, lo sé)





Class: Surface

- It encapsulates the attributes of an **exchange surface**
- **Subclass of phase class**
- Contains a free site and the surface complexes (all of them objects of solid class)



Surface



```
classDiagram
    class Surface {
        +Own Attributes
        +Own Methods
    }
    class "Own Attributes" {
        • N° surface complexes (n_sc, integer)
        • Surface complexes (vector solid class)
        • Free site (object solid class)
    }
    class "Own Methods" {
        • Set
        • Allocate
    }
```

The diagram shows a class named 'Surface' at the top center. Two lines extend downwards from the 'Surface' box to two separate boxes below it. The left box is titled 'Own Attributes' and contains a bulleted list of three items. The right box is titled 'Own Methods' and contains a bulleted list of two items.

Own Attributes

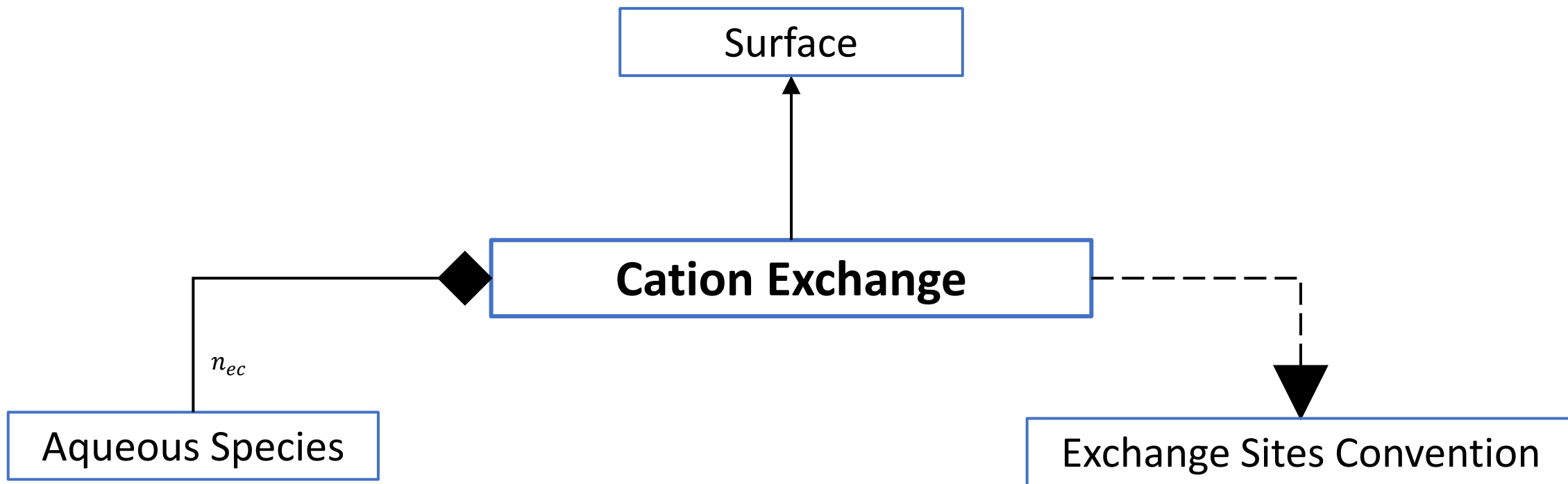
- N° surface complexes (n_{sc} , integer)
- **Surface complexes** (vector solid class)
- **Free site** (object solid class)

Own Methods

- Set
- Allocate

Class: Cation Exchange

- It encapsulates the attributes of a **cation exchange system**
- **Subclass of surface class**
- Contains the exchangeable cations (objects of *aqueous species* class)
- Points to the *exchange sites convention* superclass, which computes activities depending on the chosen convention (Gaines-Thomas, Gapon, Vanselow)



Cation Exchange

```
graph TD; CE[Cation Exchange] --- OA[Own Attributes]; CE --- OM[Own Methods];
```

Own Attributes

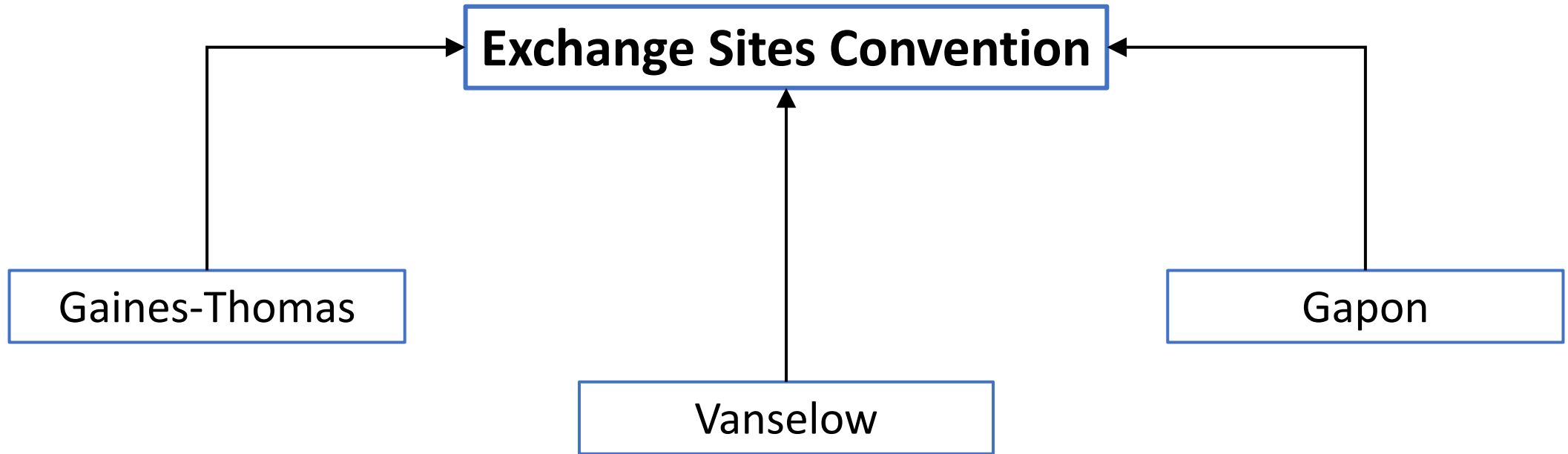
- Nº exchangeable cations (n_{ec} , integer)
- **Exchangeable cations** (vector class aqueous species) (**NO NECESARIO**)
- **Convention** (pointer class exchange sites convention)

Own Methods

- Set
- Allocate

Class: Exchange Sites Convention

- **Abstract superclass**
- Contains a deferred procedure which computes the log 10 activity coefficient of an adsorbed cation depending on the chosen convention (Gaines-Thomas, Vanselow, Gapon)
- **Each convention is a subclass of this superclass**
- This class has no attributes (it is a **library**)
- **Activity** of adsorbed cations: 3 conventions
 - Gaines-Thomas
 - Gapon
 - Vanselow



Exchange Sites Convention

```
graph TD; A[Exchange Sites Convention] --> B[Attributes]; A --> C[Methods]; B --> D[ ]; C --> E["• compute_log_act_coeff_ads_cat (deferred)"]
```

Attributes

Methods
<ul style="list-style-type: none">• <code>compute_log_act_coeff_ads_cat</code> (deferred)

Class: Gaines-Thomas

- **Subclass of exchange sites convention class**
- Contains a procedure which computes the log 10 activity coefficient of an adsorbed cation using the Gaines-Thomas convention
- **The activity is the equivalent fraction:** $a_i = \frac{z_i c_i \rho \phi}{\rho_s \theta_s CEC}$
 - z_i : charge solute
 - CEC : Cation Exchange Capacity [*equivalents*/ M_s]

$$a_{(X_{z_i}-I)} = \beta_I = \frac{[X_{z_i}-I]_{eq}}{\sum_{j=1}^{N_Y} [X_{z_j}-J]_{eq}} = \frac{z_i [X_{z_i}-I]_M}{CEC}$$

- This class has no attributes (it is a **library**)

Gaines-Thomas

```
graph TD; GT[Gaines-Thomas] --- A[Attributes]; GT --- M[Methods]; A --- A2[ ]; M --- M2["• compute_log_act_coeff_Gaines_Thomas"]
```

Attributes

Methods

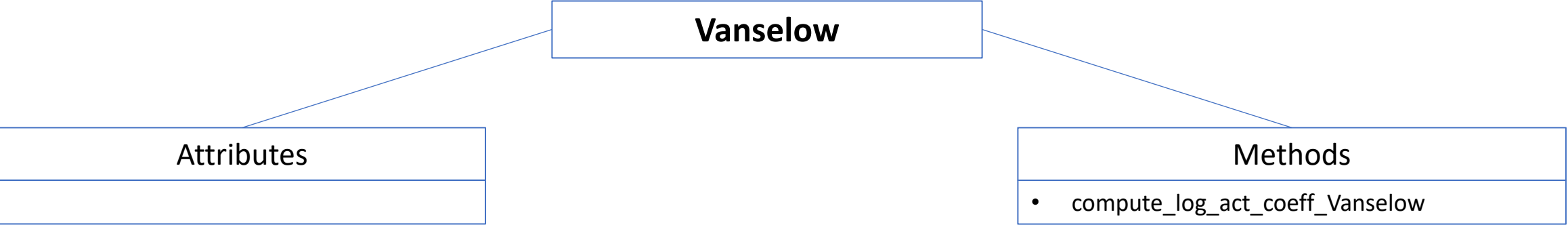
- `compute_log_act_coeff_Gaines_Thomas`

Class: Vanselow

- **Subclass of exchange sites convention class**
- Contains a procedure which computes the log 10 activity coefficient of an adsorbed cation using the Vanselow convention
- **The activity is the molar fraction**

$$\beta_I^M = \frac{\text{mmol I} - X_i \text{ per kg sediment}}{TEC} = \frac{(meq_{I-X_i}) / i}{\sum_{I,J,K...} (meq_{I-X_i}) / i}$$

- This class has no attributes (it is a **library**)



Class: Gapon

- **Subclass of exchange sites convention class**
- Contains a procedure which computes the log 10 activity coefficient of an adsorbed cation using the Gapon convention
- **The activity is the molar fraction of the total number of exchange sites**

$$K_{Ca\backslash K}^G = \frac{\beta_{Ca}}{\beta_K} \frac{[K^+]}{[Ca^{2+}]^{0.5}}$$

- This class has no attributes (it is a **library**)



Speciation Algebra Structure

Speciation Algebra

Attributes

- Flag components
- Flag cation exchange
- N° species (n_{sp})
- N° equilibrium reactions (n_e)
- N° components ($n_{comp} = n_{sp} - n_e$)
- N° constant activity species (n_c)
- N° variable activity species (n_{nc})
- N° primary species (n_p)
- N° aqueous primary species ($n_{p,aq}$)
- N° secondary aqueous species
- N° aqueous variable activity species
- N° aqueous secondary variable activity species
- $S_{e,2}^{-1}$
- $S_{e,1}^*$
- S_{e,nc_2}^{-1}
- S_{e,nc_1}^*
- $\log_{10} \mathbf{K}^*$
- $\log_{10} \tilde{\mathbf{K}}$
- \mathbf{U} (with constant activity species)
- \mathbf{U} (without constant activity species)

Flags (logicals)

dimensions

Arrays (real)

Methods

- Set:
 - **Flags**
 - **Dimensions**
- Compute:
 - **Arrays**

This type:

- contains the **flags**, **dimensions** and **arrays** relevant to **speciation** calculations.
- calculates them from the stoichiometric matrices and equilibrium constants that it receives from the reactive zone class.
- Stoichiometric matrices should be sorted into primary species, secondary species of variable activity, and secondary species of constant activity, respectively.
- The dimensions are obtained from the reactive zone class

Flags

Name	Type	Description
flag_comp	Logical	TRUE if component matrix has NO constant activity species FALSE otherwise
flag_cat_exch	Logical	TRUE if there are cation exchange reactions in the reactive zone that points to this structure FALSE otherwise

Dimensions

Name	Type	Description
num_prim_species	Integer	<p>If components include constant activity species (Saaltink et al, 1998), then</p> $n_p = n_{sp} - n_e$ <p>Otherwise (De Simoni et al, 2005), then</p> $n_p = n_{sp} - n_e - n_c$

Arrays

Name	Type	Description
$\mathbf{S}_{e,2}^{-1}$	Real matrix	Inverse of secondary species stoichiometric matrix $\mathbf{S}_{e,2}$
\mathbf{S}_{e,nc_2}^{-1}	Real matrix	Inverse of secondary variable activity species stoichiometric matrix \mathbf{S}_{e,nc_2}

Arrays

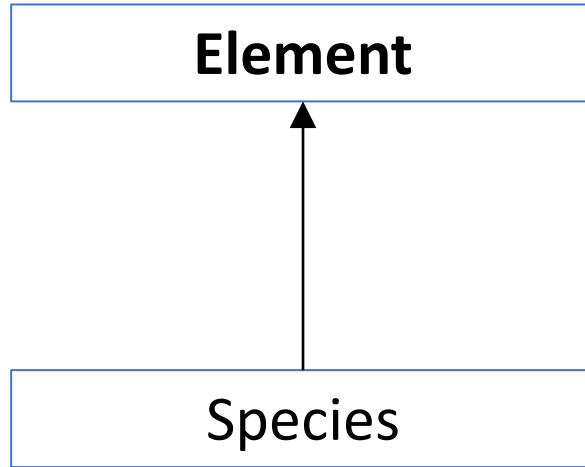
Name	Type	Formula	Description
$\mathbf{S}_{e,1}^*$	Real matrix	$\mathbf{S}_{e,1}^* = -\mathbf{S}_{e,2}^{-1}\mathbf{S}_{e,1}$	\mathbf{S}_{e_1} : Submatrix of $\mathbf{S}_{e,nc}$ which corresponds to primary species.
\mathbf{S}_{e,nc_1}^*	Real matrix	$\mathbf{S}_{e,nc_1}^* = -\mathbf{S}_{e,nc_2}^{-1}\mathbf{S}_{e,1}$	\mathbf{S}_{e_1} : Submatrix of $\mathbf{S}_{e,nc}$ which corresponds to primary species.

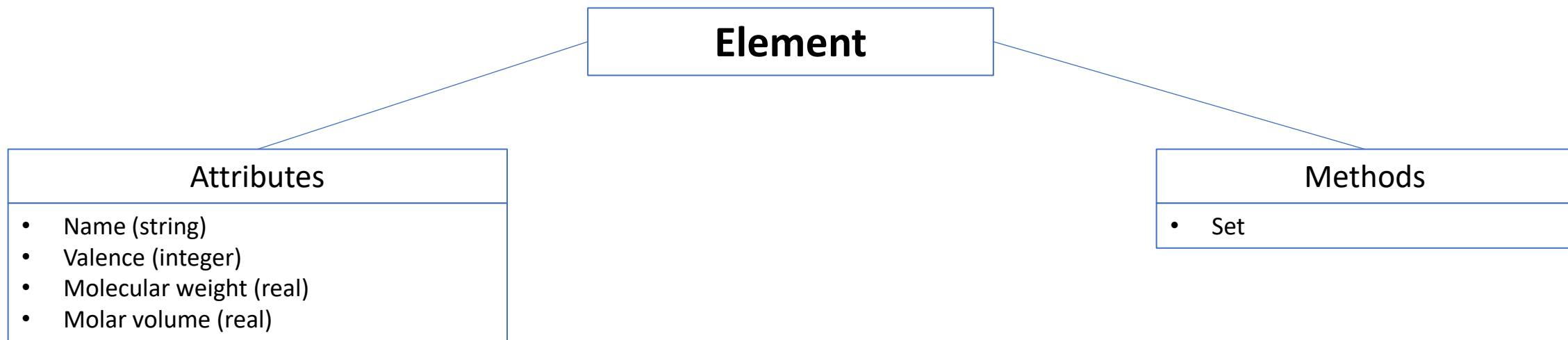
Arrays

Name	Type	Formula	Description
$\log_{10} \mathbf{K}^*$	Real vector	$\log_{10} \mathbf{K}^* = \mathbf{S}_{e,nc_2}^{-1} \log_{10} \mathbf{K}$	\mathbf{K} : equilibrium constants.
$\log_{10} \tilde{\mathbf{K}}$	Real vector	$\log_{10} \tilde{\mathbf{K}} = \mathbf{S}_{e,2}^{-1} \log_{10} \mathbf{K}$	\mathbf{K} : equilibrium constants.
\mathbf{U}	Real matrix	$\mathbf{U} = \left(\mathbf{I}_{n_p} \left \mathbf{S}_{e,nc_1}^{*T} \right. \right)$	Component matrix without constant activity species (De Simoni et al, 2005)
		$\mathbf{U} = \left(\mathbf{I}_{n_p} \left \mathbf{S}_{e,1}^{*T} \right. \right)$	Component matrix with constant activity species (Saaltink et al, 1998)

Element Class

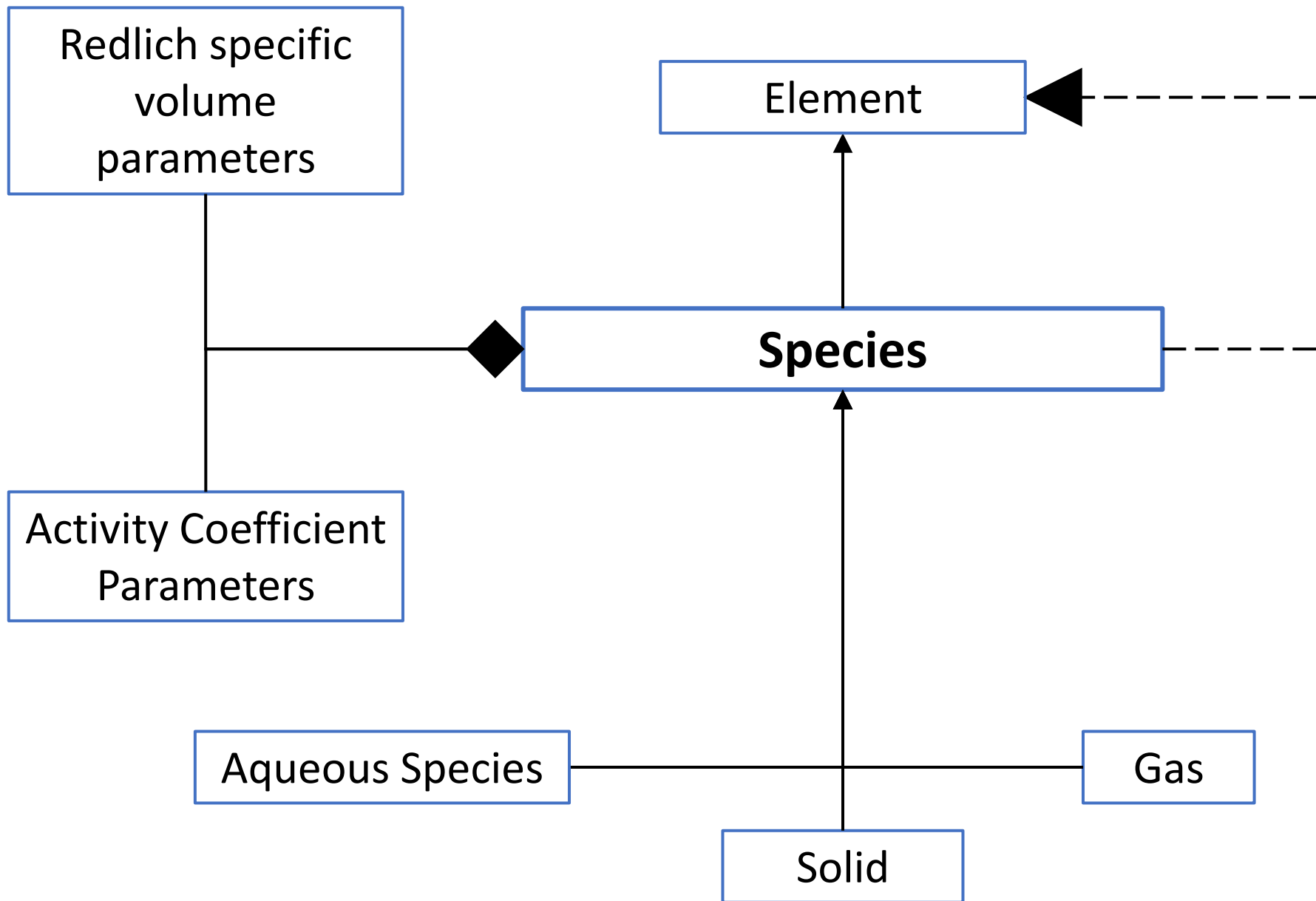
- It encapsulates the attributes of chemical elements.
- **Superclass**





Species Class

- It encapsulates the attributes of chemical species.
- It contains objects of the *activity coefficient parameters* & *Redlich specific volume parameters* structures
- It points to the *element* class (based on PHREEQC)
- **Subclass of element class**



Species

```
graph TD; Species[Species] --- Attributes[Own Attributes]; Species --- Methods[Own Methods];
```

Own Attributes

- **Element** (pointer to element class)
- Boolean constant activity (Logical)
- **Activity coefficient parameters** (object activity coefficient parameters structure)
- **Redlich specific volume parameters** (object Redlich specific volume parameters structure)

Own Methods

- Set
- Assign:
 - species
- Read:
 - species

Attributes

Name	Type	Description
element	Element class (pointer)	Element associated to species in PHREEQC database
cst_act_flag	logical	TRUE if species has constant activity FALSE otherwise
params_act_coeff	Activity coefficient parameters structure	Contains parameters to compute activity coefficients
params_spec_vol_Redlich	Redlich specific volume parameters structure	Contains parameters to compute conventional specific volume using Redlich equation (based on PHREEQC)

Activity Coefficient Parameters Structure

- It contains the parameters necessary to calculate the activity coefficient of a species.

Activity Coefficient Parameters

Attributes

- a_{TJ} (Truesdell-Jones)
 - b_{TJ} (Truesdell-Jones)
 - \dot{b}
 - α
 - β
 - Ion size parameter
- } Real scalars

Methods

- Set
- Compute:
 - \dot{b}
 - α
 - β

Redlich specific volume parameters structure

- It contains the parameters necessary to compute the conventional specific volume of a solute species with a Redlich-type equation (based on PHREEQC)

Redlich specific volume parameters

```
graph TD; Root[Redlich specific volume parameters] --- Attributes[Attributes]; Root --- Methods[Methods]; Attributes --- A1["• a1, a2, a3, a4"]; Attributes --- A2["• W"]; Attributes --- A3["• lon size parameter"]; Attributes --- A4["• i1, i2, i3, i4"]; A1 --- RS[Real scalars]; A2 --- RS; A3 --- RS; Methods --- M1["• Set"];
```

Attributes

- a_1, a_2, a_3, a_4
 - W
 - lon size parameter
 - i_1, i_2, i_3, i_4
- } Real scalars

Methods

- Set

Class: Aqueous Species

- **Subclass of species class**
- It encapsulates the attributes of aqueous chemical species.
- The activities of aqueous species is the molar fraction of species for ideal, low salinity solutions
- As salinity increases, “effective” concentration decreases $\Rightarrow a_i = \gamma_i m_i$
- For moderate salinities, $a_i \approx \gamma_i c_i$
- In general, $a_i \approx \gamma_i c_i / \omega_w$

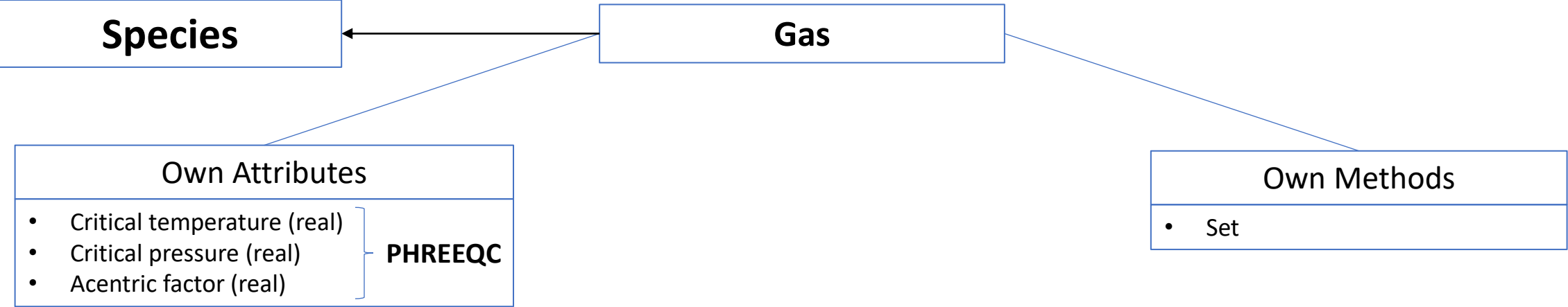


Methods to compute log 10 activity coefficient

Name	Type	Description
Debye_Huckel_restr	Subroutine	Calculates log 10 activity coefficient of species i with formula $\log_{10} \gamma_i = -Az_i^2 \sqrt{I}$
Debye_Huckel_ampl	Subroutine	Calculates log 10 activity coefficient of species i with formula $\log_{10} \gamma_i = -\frac{Az_i^2 \sqrt{I}}{1 + a_i B \sqrt{I}},$ where a_i is the ionic radius of species i
Davies	Subroutine	Calculates log 10 activity coefficient of species i with formula $\log_{10} \gamma_i = -Az_i^2 \left(\frac{\sqrt{I}}{1 + \sqrt{I}} - 0.3I \right)$
Truesdell_Jones	Subroutine	Calculates log 10 activity coefficient of species i with formula $\log_{10} \gamma_i = -\frac{Az_i^2 \sqrt{I}}{1 + Ba_i \sqrt{I}} + b_{TJ} I$

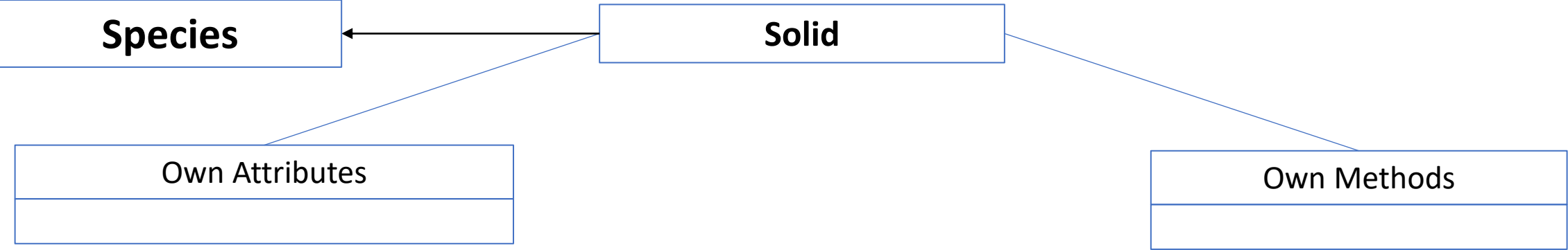
Class: Gas

- **Subclass of species class**
- It encapsulates the attributes of gases.
- **Activity of gas = partial pressure (for ideal conditions)**
- For high pressures and temperatures: $a_i = \Gamma_i p_i$



Class: Solid

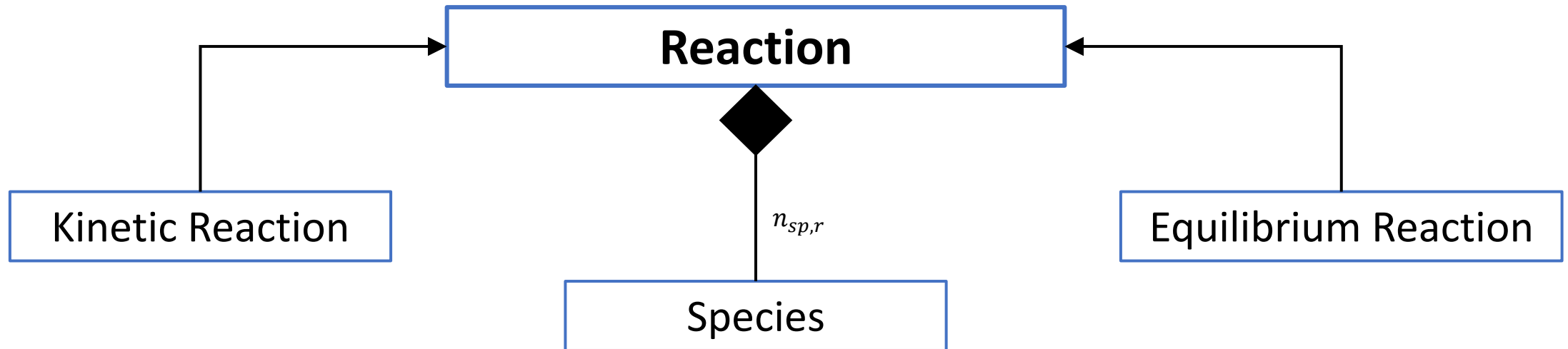
- **Subclass of species class**
- It encapsulates the attributes of a solid.
- **The class is empty at the moment**



Class: Reaction

- It has inheritance depending on the type of reaction.
- It encapsulates the information of a chemical reaction: number of species involved, species involved and their stoichiometric coefficients, name of reaction, reaction type, equilibrium constant, enthalpy, etc.
- **Superclass**

Class: Reaction



Reaction

```
graph TD; Reaction[Reaction] --- Attributes[Attributes]; Reaction --- Methods[Methods];
```

Attributes

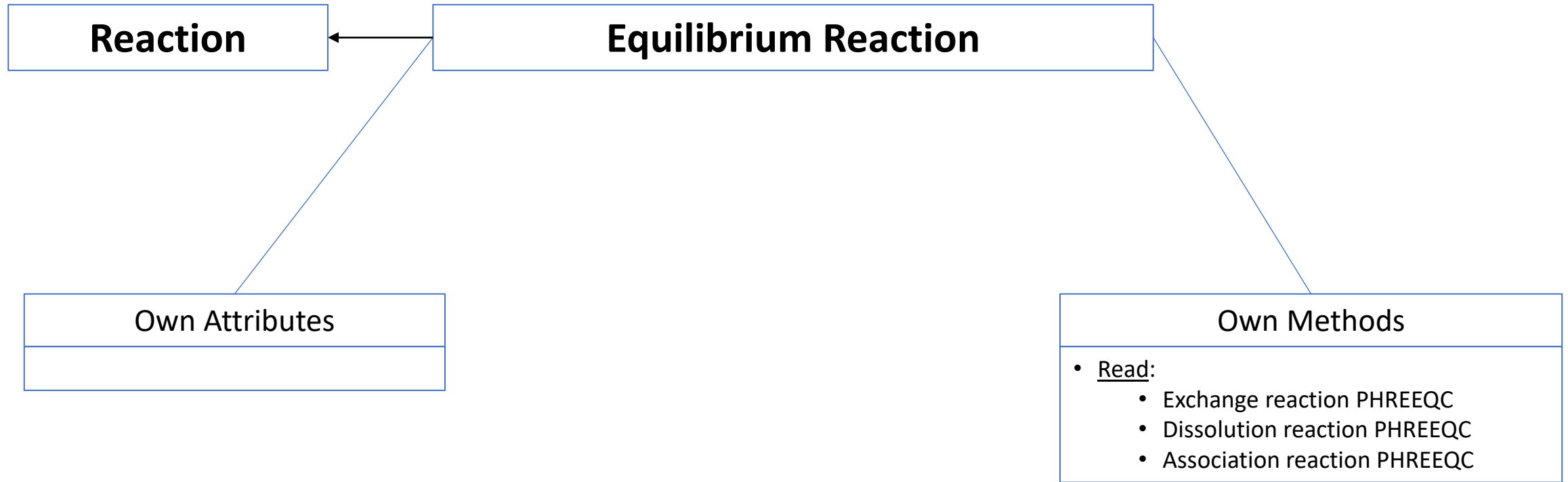
- Nº of species ($n_{sp,r}$)
- **Species** (vector class species)
- Stoichiometric coefficients (real vector)
- Name (string)
- Type (integer)
- Coefficients for dependence of equilibrium constant on temperature (real vector)
- Equilibrium constant (real scalar)
- Enthalpy (real scalar)

Methods

- Set
- Allocate
- Compute
 - $\log_{10} K(T)$

Equilibrium Reaction Class

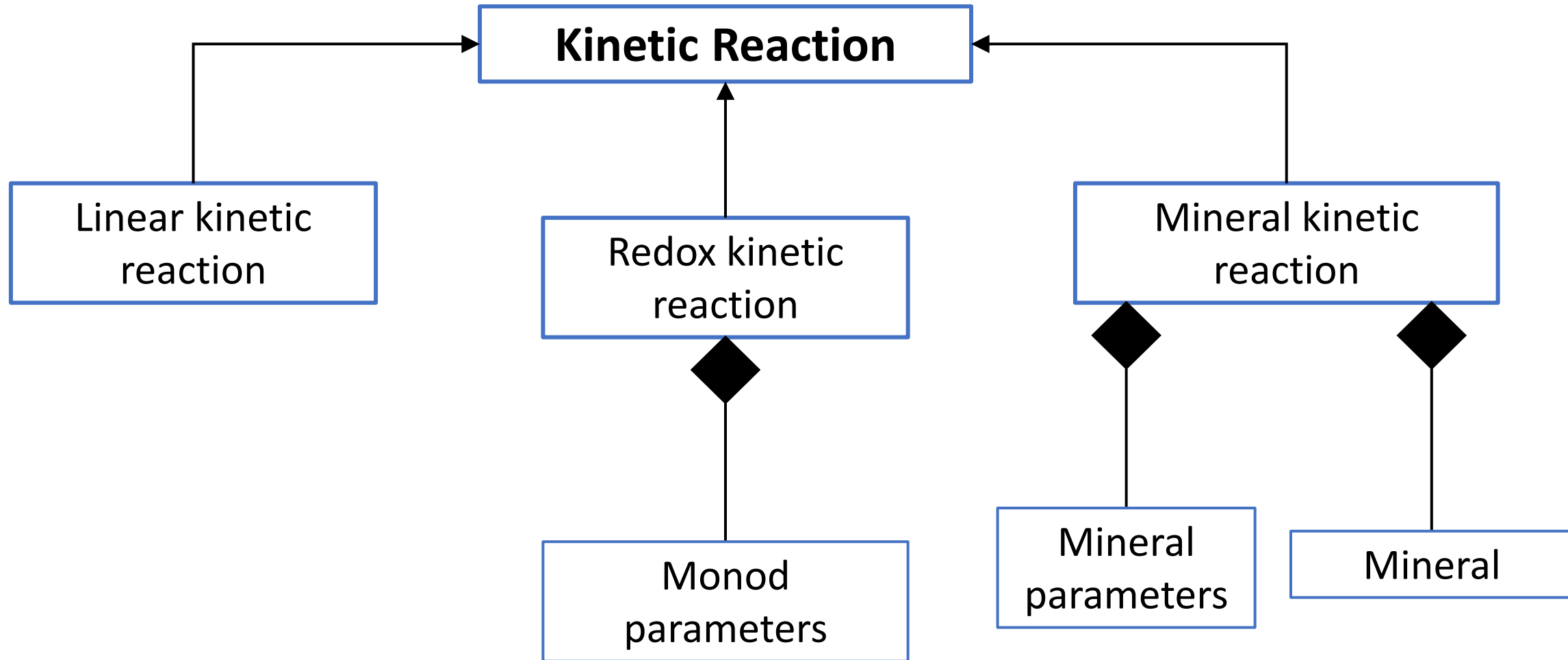
- **Subclass of reaction class.**
- Only contains procedures, not (own) attributes.
- Reads equilibrium reactions in PHREEQC database.

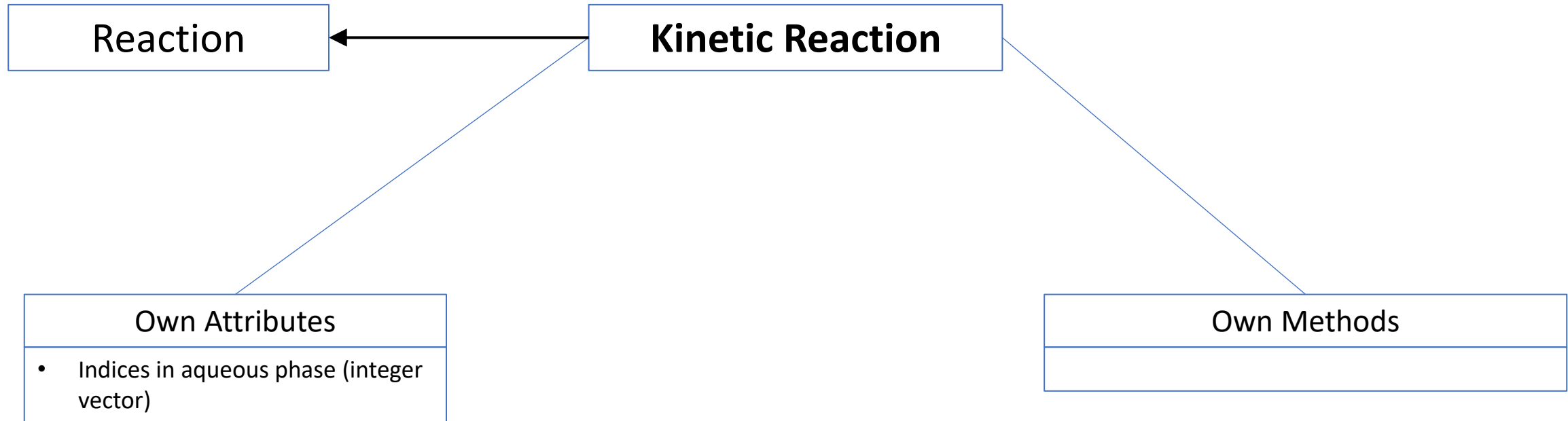


Kinetic Reaction Class

- **Subclass of reaction class.**
- Contains subclasses depending on the type of kinetic reaction
- **The subclasses compute the kinetic reaction rate and its gradient**
- Only has one own attribute (indices of aqueous species that control kinetic reaction rate) and no own methods.

Kinetic Reaction Class





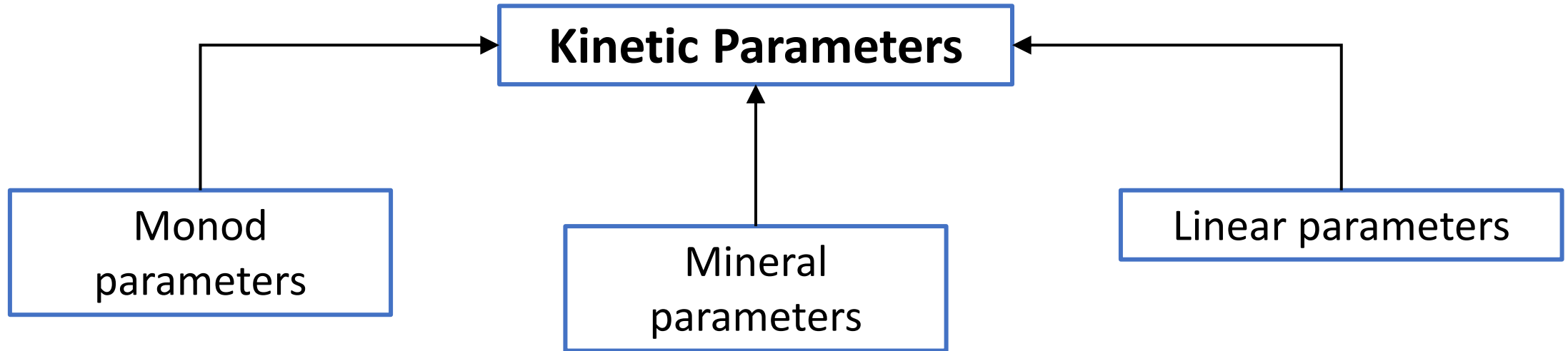
Attributes

Name	Type	Description
indices_aq_phase	Integer vector	Indices of species in aqueous phase class that control kinetic reaction rates.

Kinetic Parameters Class

- **Abstract superclass**
- Contains subclasses depending on the type of kinetic reaction
- The subclasses contain the parameters to compute each kinetic reaction
- Only has one own attribute (reaction rate constant) and no own methods.

Kinetic Parameters Class



Kinetic Parameters

```
graph TD; A[Kinetic Parameters] --> B[Attributes]; A --> C[Methods]; B --> D["• Reaction rate constant (real)"];
```

Attributes

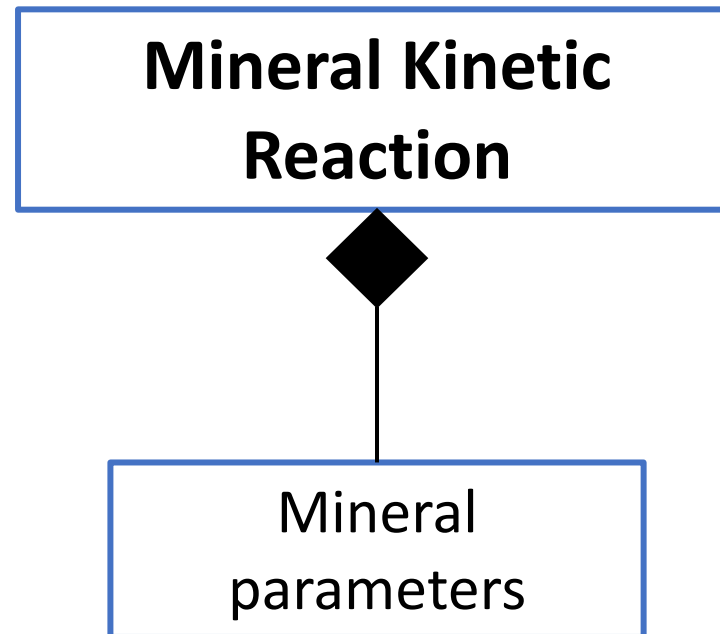
- Reaction rate constant (real)

Methods

Mineral Kinetic Reaction

- Subclass of kinetic reaction class
- Mineral dissolution/precipitation NOT in equilibrium

Mineral Kinetic Reaction



Kinetic Reaction

Mineral Kinetic Reaction

Attributes

- **params** (object mineral parameters structure)

Methods

- compute_drk_dc_mineral
- compute_rk_mineral
- set_mineral_params

Mineral Kinetic Reaction

Attributes	Type	Description
params	Mineral Parameters structure	Contains the parameters for calculating the mineral kinetic reaction rate

Mineral Kinetic Reaction

Methods	Type	Description
compute_drk_dc_mineral	subroutine	Calculates the mineral kinetic reaction rate gradient
compute_rk_mineral	subroutine	Calculates the mineral kinetic reaction rate
set_mineral_params	subroutine	Assigns attribute "params"

Mineral Parameters Structure

- **Kinetic parameters subclass**
- This structure contains the parameters needed to calculate the reaction rates using the transition state theory formula:

$$r_k = \sigma_k \zeta_k e^{-\frac{E_{a,k}}{RT}} \sum_{j=1}^{N_k} k_{kj} \prod_{i=1}^{N_c} a_{cat_i}^{p_{kji}} \left(\Omega_k^{\theta_{kj}} - 1 \right)^{\eta_{kj}} \quad E_{a,k} \in [30,80] \left(\frac{kJ}{mol} \right)$$

[moles per unit volume of rock per unit time]

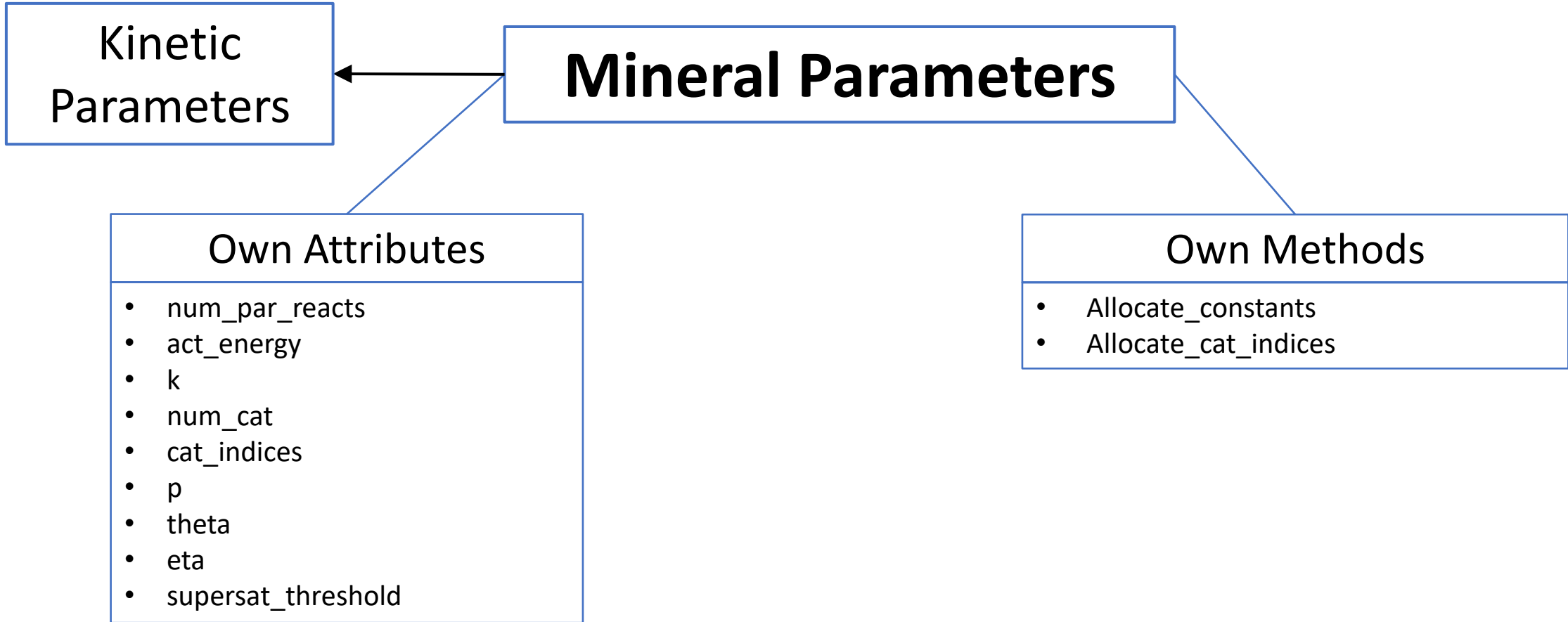
Gradient:

$$\frac{\partial r_k}{\partial c_l} = \sigma_k \zeta_k e^{-\frac{E_{a,k}}{RT}} \frac{v_l}{c_l} \sum_{j=1}^{N_k} k_{kj} \eta_{kj} \left(\Omega_k^{\theta_{kj}} - 1 \right)^{\eta_{kj}-1} \theta_{kj} \Omega_k^{\theta_{kj}} \prod_{i=1}^{N_c} a_{cat_i}^{p_{kji}}$$

Aqueous species index

If $\theta_k, \eta_k = 1$:

$$\frac{\partial r_k}{\partial c_l} = \sigma_k \zeta_k e^{-\frac{E_{a,k}}{RT}} \frac{v_l}{c_l} \Omega_k \sum_{j=1}^{N_k} k_{kj} \prod_{i=1}^{N_c} a_{cat_i}^{p_{kji}}$$



Mineral Parameters

Attributes	Type	Description
num_par_reacts	integer	No. of parallel reactions
act_energy	real	Apparent activation energy (J)
k	Real Vector	Experimental constants
num_cat	integer	No. of catalysts
cat_indices	Integer vector	Catalyst indices
p	Real matrix	Catalytic effect of some species
theta	Real Vector	Experimental parameters
eta	Real Vector	Experimental parameters
supersat_threshold	real	Supersaturation threshold

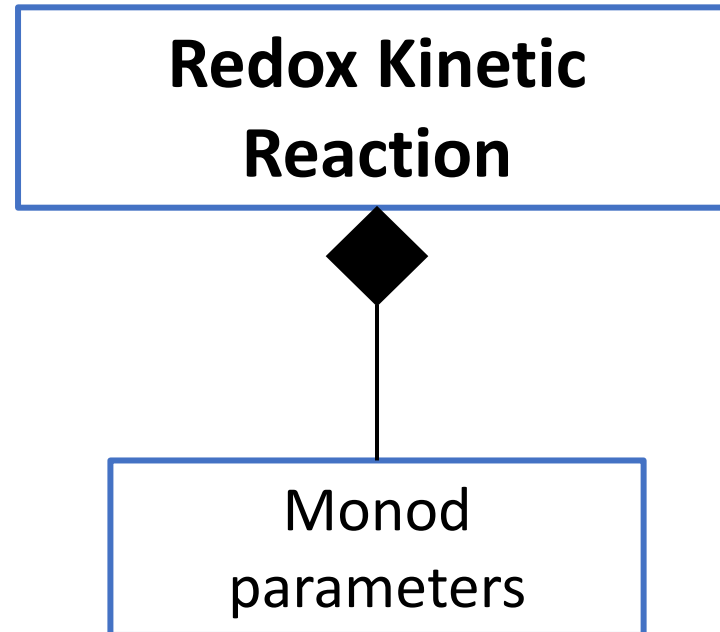
Mineral Parameters

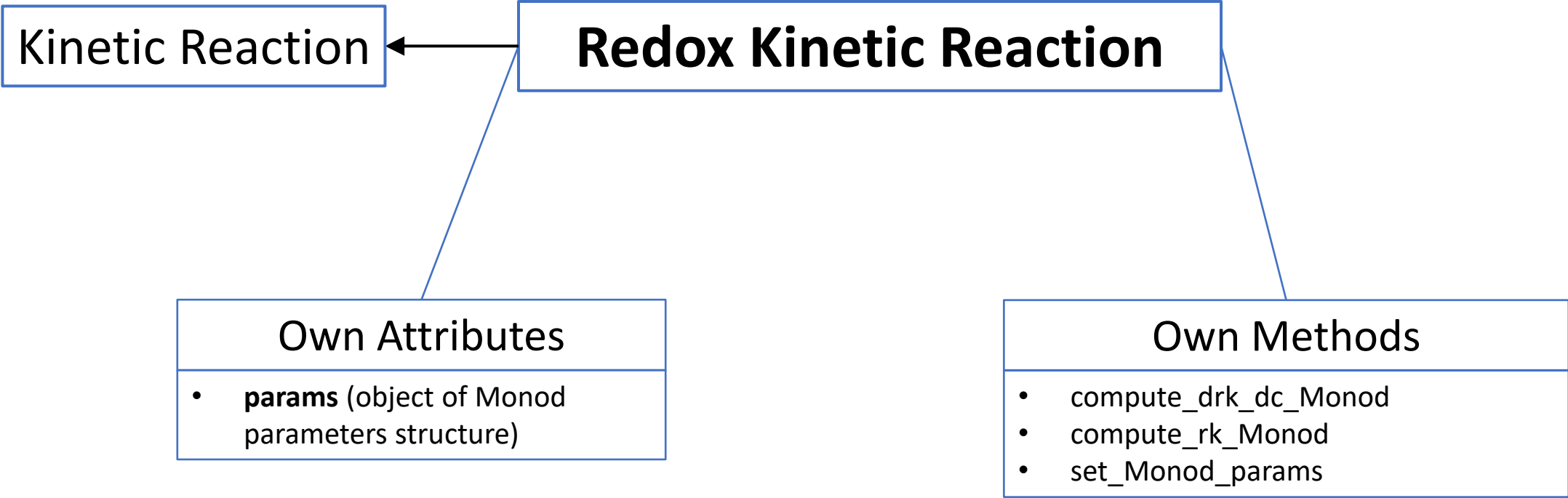
Methods	Type	Description
Allocate_constants	Subroutine	Allocates constants and parameters
Allocate_cat_indices	Subroutine	Allocates <i>cat_indices</i> attribute

Redox Kinetic Reaction

- Subclass of kinetic reaction class

Redox Kinetic Reaction





Redox Kinetic Reaction

Own Attributes	Type	Description
params	Monod Parameters structure	Contains the parameters for calculating the Monod reaction rate

Redox Kinetic Reaction

Methods	Type	Description
compute_drk_dc_Monod	subroutine	Calculates the reaction rate gradient
compute_rk_Monod	subroutine	Calculates the reaction rate
set_Monod_params	subroutine	Assigns object "params"

Monod Parameters Structure

- **Kinetic parameters subclass**
- This structure contains the parameters needed to calculate the reaction rate using the Monod expression
- **Microorganisms and logistical factor are not included yet**


$$r_k = \mu \prod_{i=1}^{n_T} T_i = \mu \prod_{i=1}^{n_{inh}} I_i \prod_{j=1}^{n_M} M_j$$

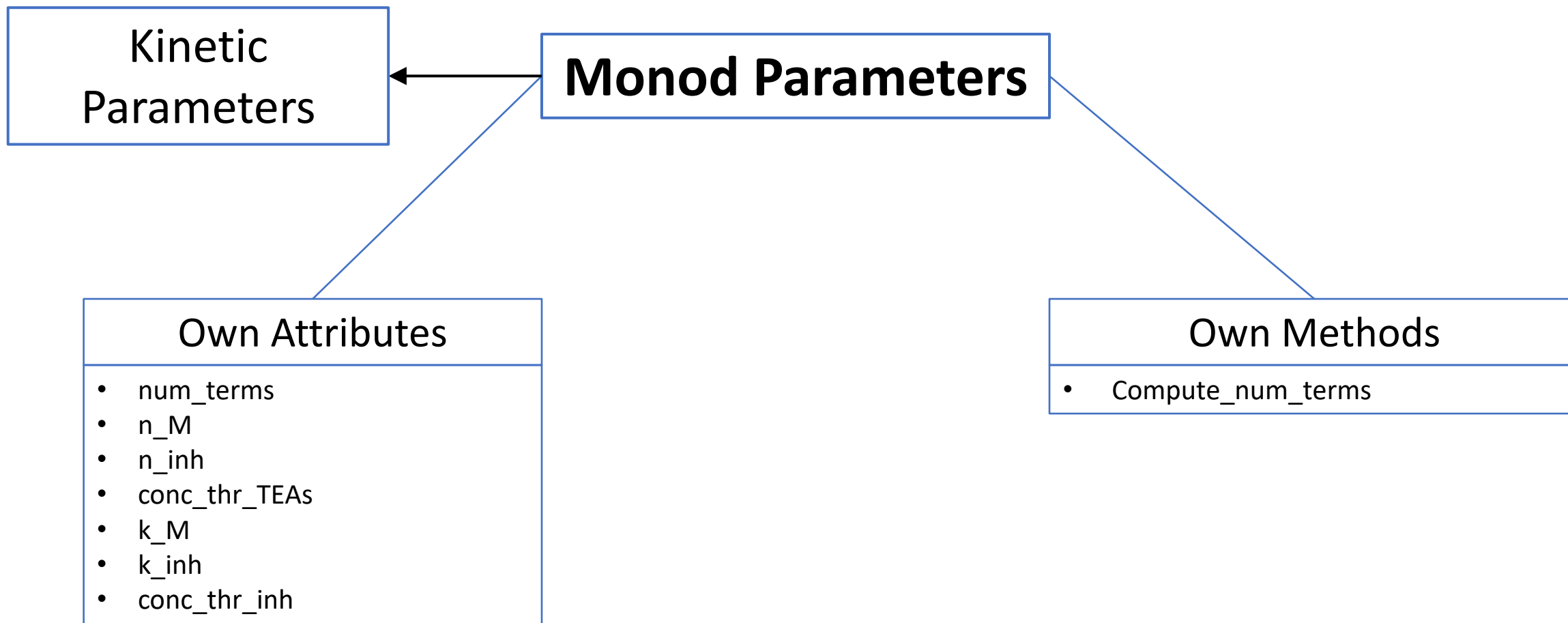
$$I = \frac{k_{inh}}{k_{inh} + c_{inh}}$$

$$M = \frac{c_M}{k_M + c_M}$$

Gradient:

$$\frac{\partial r_k}{\partial c_i} = r_k \frac{\partial \log T_i}{\partial c_i} = \begin{cases} r_k \frac{k_i}{c_i(k_i + c_i)}, & \text{e}^- \text{ acceptor/donor} \\ -\frac{r_k}{k_i + c_i}, & \text{inhibitors} \end{cases}$$


 Index species



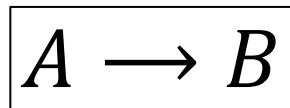
Monod Parameters

Own Attribute	Type	Description
num_terms	integer	Number of terms in the Monod expression
n_inh	integer	Number of inhibitors
conc_thr_TEAs	Real vector	TEA concentration threshold
k_M	Real vector	Half-saturation constants
k_inh	Real vector	Inhibition constants
conc_thr_inh	Real vector	Inhibitor concentration threshold

Monod Parameters

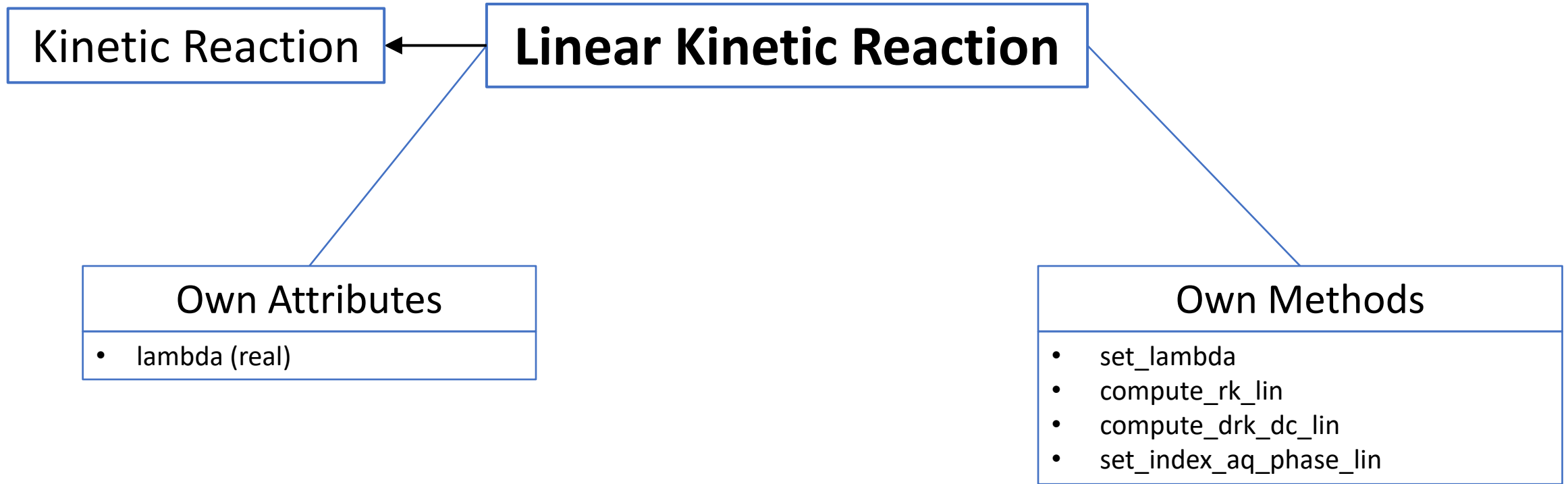
Own Method	Type	Description
Compute_num_terms	subroutine	Computes number of terms

Linear Kinetic Reaction Class



- **Subclass of kinetic reaction class.**
- Computes linear kinetic reaction rate and its gradient.
- Contains the reaction rate constant λ .

$$r_k = \lambda c_A$$



Structure: Chem_out_options

- Structure containing the number and indices of variables that will be written in output file
- Writing during the loop on the solver
- **This structure uses the aqueous chemistry class**

Chem_out_options

```
graph TD; A[Chem_out_options] --- B[Attributes]; A --- C[Methods];
```

Attributes

- Nº targets
- Targets
- Nº time steps
- Time steps
- Nº variables
- Variables (array of strings)
- Nº indices aqueous species
- Indices aqueous species
- Indices phases (minerals, gases, etc)
- Nº indices phases (minerals, gases, etc)
- Indices reactions
- Nº indices reactions

Methods

- Set
- Allocate
- Read

Structure: CV_params

- Structure containing the numerical parameters for iterative algorithms and linear system solvers

CV_params

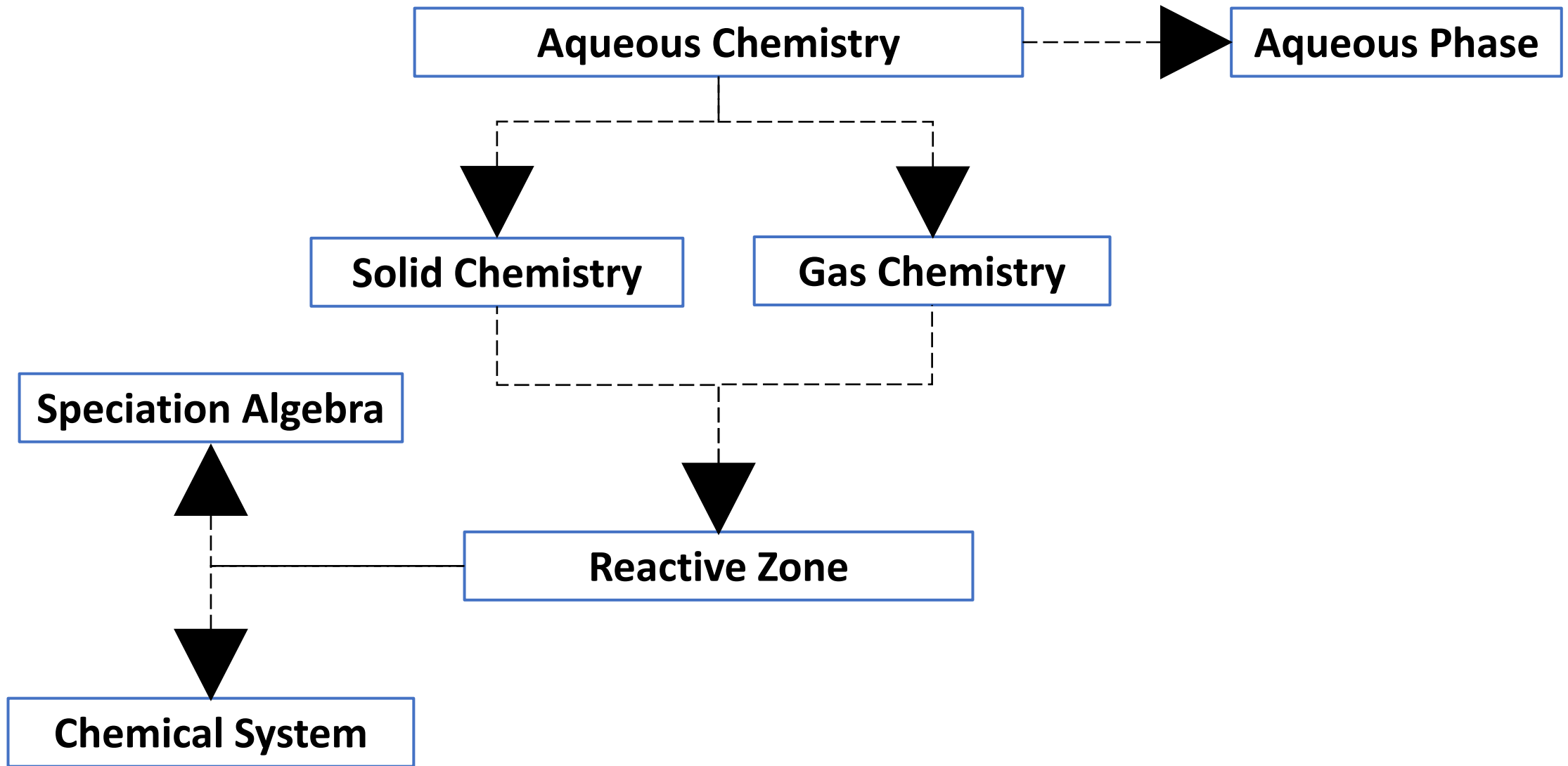


```
graph TD; CV_params[CV_params] --- Attributes[Attributes];
```

Attributes

- Absolute tolerance
- Relative tolerance
- Logarithmic absolute tolerance
- Max. no. iterations
- ϵ (for incremental coefficients)
- Control factor

Flow Diagrams



User guide

1. Define chemical system in “_sist_quim.dat” file
2. Define initial waters, solid and/or gas zones in “_quim_loc.dat” file
3. Define association between initial target waters, solids and/or gases in “_tar_wat.dat” file
4. Define time discretisation in “_WMA_discr.dat” file
5. Define output variables in “_out_options.dat” file
6. Compute mixing ratios and define indices of waters that will mix with each other in “_WMA_lambdas.dat”
7. Provide the following arguments to the **solve_reactive_mixing** interface:
 - Mixing ratios
 - Mixing waters indices
 - Time discretisation
 - Storage matrix

User guide

