Package **AquaEnv**: an Aquatic modelling Environment in **R**

Andreas F. Hofmann, Karline Soetaert and Mathilde Hagens

Abstract

`AquaEnv` (Hofmann *et al.* 2010b) is an integrated development toolbox for aquatic chemical model generation focused on (ocean) acidification and CO_2 air-water exchange. It contains

- All elements necessary to model the pH, the related CO_2 air-water exchange, as well as aquatic acid-base chemistry in general for an arbitrary marine, estuarine or freshwater system.
- A suite of tools to visualize this information.
- It can be used to build dynamic models of aquatic systems that include acid-base chemistry.
- The sensitivity of the system to variations in the input variables can be visualized.
- A number of example "applications" that make use of `AquaEnv` are:
  - a theoretical titration simulator
  - a routine to determine total alkalinity ([TA]), the total dissolved inorganic carbon concentration ($[\sum CO_2]$), as well as additionally the electrode standard potential (E_0) and the first dissociation constant of the carbonate system ($K^*_{CO_2}$) from titration data.

Keywords: aquatic modelling, pH, pH scales, dissolved inorganic carbon, total alkalinity, total alkalinity curve fitting, theoretical titration, revelle factor, omega, solubility products, CO_2, ocean acidification, estuaries, carbonate system, seawater, buffer factors, R.

Contents

1. Introduction

`AquaEnv` can be used in three ways

- It provides separate functions to calculate the stoichiometric equilibrium constants (K^*) for key acid base systems in natural seawater, the Henry's constants (K_0), as well as the solubility products (K_{sp}) for calcite and aragonite.

  This functionality is provided via the functions `K_CO2, K_HCO3, K_BOH3, K_W, K_HSO4, K_HF, K_NH4, K_H2S, K_H3PO4, K_H2PO4, K_HPO4, K_SiOH4`, `K_SiOOH3, K0_CO2, K0_O2, Ksp_aragonite` and `Ksp_calcite`.

- It is also possible to estimate all the acid-base chemistry using one single function: `aquaenv`. This function returs a list of class `aquaenv` that contains, in addition to the input parameters:
  - the clorinity, the ionic strength, $[\sum B(OH)_3]$, $[\sum H_2SO_4]$, $[\sum HF]$, $[Cl^-]$, $[Cl^-]$, $[\sum Br]$, $[Na^+]$, $[Mg^{2+}]$, $[Ca^{2+}]$, $[K^+]$, $[Sr^{2+}]$ calculated from salinity as given in DOE (1994) [1]
  - the gauge pressure p (total pressure minus atmospheric pressure, Feistel 2008) either given as input variable, or calculated from depth (according to Fofonoff and Millard 1983), or calculated from the total pressure P and the atmospheric pressure Pa, both of which can be given as input variables and are also stored in an object of class `aquaenv`
  - the seawater density calculated from temperature and salinity as given by Millero and Poisson (1981)
  - a set of conversion factors to convert between different pH scales (Dickson 1984; Zeebe and Wolf-Gladrow 2001) and between mol/kg-H_2O and mol/kg-solution (inferred from Roy *et al.* (1993b) and DOE (1994))
  - the Henry's constants for CO_2 (Weiss 1974) and for O_2 (inferred from Weiss 1970) calculated from temperature and salinity as well as the associated saturation concentrations of CO_2 and O_2.
  - the ion product of water (Millero 1995), the stoichiometric equilibrium constants of HSO_4^- (Dickson 1990a), HF (Dickson and Riley 1979a) , CO_2 (Roy *et al.* 1993b), HCO_3^- (Roy *et al.* 1993b), $B(OH)_3$ (Dickson 1990a), NH_4^+(Millero *et al.* 1995), H_2S (Millero 1995), H_3PO_4(Millero 1995), $H_2PO_4^-$ (Millero 1995), HPO_4^{2-} (Millero 1995), $Si(OH)_4$ (Millero *et al.* 1988), $SiO(OH)_3^-$ (Wischmeyer *et al.* 2003), HNO_2 (Riordan *et al.* 2005), HNO_3, H_2SO_4 (Atkins 1996), HS (Atkins 1996) mostly calculated as functions of temperature and salinity and pressure corrected according to Millero (1995).
  - the solubility products of calcite and aragonite (Mucci 1983) as well as the associated Ω's if a full speciation is calculated (see below)
  - the fugacity of CO_2 - if a full speciation is calculated (see below)
  - if $[\sum CO_2]$ and pH are given [TA] is calculated, if $[\sum CO_2]$ and [TA] are given pH is calculated, if $[\sum CO_2]$ and $[CO_2]$ or fCO_2 are given, pH and [TA] are calculated.

---

[1] Please note that if values for $[\sum B(OH)_3]$, $[\sum H_2SO_4]$, and/or $[\sum HF]$ are given as input parameters, these parameters are used and not the ones calculated from salinity.

  - if either one of the pairs pH and [CO_2] or fCO_2, pH and [TA], or [TA] and [CO_2] or fCO_2 is given, [$\sum CO_2$] is calculated
  - if sufficient information is given and the flag `speciation=TRUE` is set, a full speciation of [$\sum$ CO2], [$\sum$ NH4], [$\sum H_2S$], [$\sum$ HNO3], [$\sum$ HNO2], [$\sum$ H3PO4], [$\sum Si(OH)_4$], [$\sum B(OH)_3$], [$\sum H_2SO_4$], [$\sum$ HF], as well as water itself is calculated
  - if the flag `dsa = TRUE` is set, all necessary quantities for the explicit "direct substitution approach" (DSA) to pH modelling as given in Hofmann *et al.* (2008) are calculated. These are the buffer factor (the partial derivative of [TA] with respect to [H^+]) and the partial derivatives of [TA] with respect to the other total quantities. Furthermore, the partial derivatives of [TA] with respect to changes in the equilibrium constants (K^*), multiplied with the partial derivatives of the equilibrium constants with respect to their variables needed for the DSA with time variable equilibrium constants as described in Hofmann *et al.* (2009) are calculated. Finally, the ionization fractions as defined by Stumm and Morgan (1996) and used in Hofmann *et al.* (2010a) are calculated for the full speciation.
  - Thirdly, a generic function `BufferFactors` is provided, based on Hagens and Middelburg (2016). This function internally calls the function `aquaenv` and uses its output to analytically calculate the sensitivity of pH and concentrations of CO_2 and other acid-base species to a change in ocean chemistry, as well as the Revelle factor.

- Input for `aquaenv` and `BufferFactors` has to be supplied in standard SI units, the free proton pH scale and in molinity (mol/kg-solution)[2]. Conversion of input parameters to these necessary units and pH scale can be done with the generic function `convert`.

- The information created with `aquaenv` is also supplied in standard SI units and in molinity. All elements of an object of class `aquaenv` of a certain unit or pH scale can be converted into other units or pH scales with the function `convert` as well.

- One can use input vectors of salinity S, temperature t, or gauge pressure p (as well as total pressure P and depth d) for `aquaenv` to obtain vectors of all calculated information as function of the input vector. This can be visualized in a large variety of ways using the `plot` function specially defined for objects of class `aquaenv`.

- Objects of class `aquaenv` can be used in dynamic models to define the state of the system in each timestep of the numerical integration (done with e.g. **deSolve**). With the function `aquaenv` and the flag `from.data.frame=TRUE` it is possible to convert output of those dynamic models into objects of class `aquaenv` which allows the user to use the whole suite of visualisation tools that is provided by the function `plot` in **AquaEnv**.

- Hofmann *et al.* (2008), Hofmann *et al.* (2009), and Hofmann *et al.* (2010a) describe methods for an "explicit" pH modelling which allows for the quantification of the influences of kinetically modelled processes on pH. Objects of class `aquaenv` provide all needed quantities (partial derivatives of [TA], ionization fractions, etc.) to employ both

---

[2] Note that it is not sufficient to give a gravimetric concentration in mol/kg since there is a substantial difference between mol/kg-H_2O (molality) and mol/kg-solution (molinity).

of those methods in dynamic models. **AquaEnv** also provides the functionality to cumulatively plot the obtained influences on pH.

- As an example of how to use the aquatic chemical toolbox that is provided by **AquaEnv**, two applications are provided:
  - The function `titration`: creates theoretical titrations which can be used to e.g. create Bjerrum plots with the function `plot.aquaenv` in **AquaEnv**.
  - The function `TAfit`: a routine based on a method in DOE (1994) that makes use of that theoretical titration function and allows for determining total alkalinity ([TA]), the total dissolved inorganic carbon concentration ($[\sum CO2]$), as well as additionally the electrode standard potential (E_0) and the first dissociation constant of the carbonate system ($K^*_{CO_2}$) using the Levenberg-Marquart algorithm (least squares optimization procedure) as provided in the R package **minpack.lm**.

2. The elements of an object of class `aquaenv`

The function `aquaenv`, the central function of **AquaEnv**, returns an object of class `aquaenv`. This object is a list of different elements which can be accesses with the $ character or with the [[]] operator

```
> test <- aquaenv(S = 35, t = 10)
> test$t

[1] 10
attr(,"unit")
[1] "deg C"
```

If enough input data is supplied to define the pH of the system and the flags `speciation` and `dsa` are `TRUE` while the flag `skeleton` is `FALSE`, an object of class `aquaenv` contains the following elements[3]

element	unit	explanation
`S`	"psu" (no unit)	salinity
`t`	°C	temperature
`p`	bar	gauge pressure (total pressure minus atmospheric pressure, Feistel 2008)
`T`	K	absolute temperature
`Cl`	‰	chlorinity
`I`	mol/kg-H_2O	ionic strength
`P`	bar	total pressure
`Pa`	bar	atmospheric pressure
`d`	m	depth
`density`	kg/m^3	(seawater) density

---

[3] Literature references are given in Appendix B.

`SumCO2`	mol/kg-soln	$[\sum CO_2]$, total dissolved inorganic carbon concentration
`SumNH4`	mol/kg-soln	$[\sum NH_4^+]$, total ammonium concentration
`SumH2S`	mol/kg-soln	$[\sum H_2S]$, total sulfide concentration
`SumHNO3`	mol/kg-soln	$[\sum HNO_3]$, total nitrate concentration
`SumHNO2`	mol/kg-soln	$[\sum HNO_2]$, total nitrite concentration
`SumH3PO4`	mol/kg-soln	$[\sum H_3PO_4]$, total phosphate concentration
`SumSiOH4`	mol/kg-soln	$[\sum Si(OH)_4]$, total silicate concentration
`SumBOH3`	mol/kg-soln	$[\sum B(OH)_3]$, total borates concentration
`SumH2SO4`	mol/kg-soln	$[\sum H_2SO_4]$, total sulfate concentration
`SumHF`	mol/kg-soln	$[\sum HF]$, total fluoride concentration
`Br`	mol/kg-soln	$[Br^-]$, bromide concentration
`ClConc`	mol/kg-soln	$[Cl^-]$, chloride concentration
`Na`	mol/kg-soln	$[Na^+]$, sodium concentration
`Mg`	mol/kg-soln	$[Mg^{2+}]$, magnesium concentration
`Ca`	mol/kg-soln	$[Ca^{2+}]$, calcium concentration
`K`	mol/kg-soln	$[K^+]$, potassium concentration
`Sr`	mol/kg-soln	$[Sr^{2+}]$, strontium concentration
`molal2molin`	(mol/kg-soln)/(mol/kg-H2O)	concentration conversion factor: from molality to molinity
`free2tot`	-	pH conversion factor: free scale to total scale
`free2sws`	-	pH conversion factor: free scale to sawater scale
`tot2free`	-	pH conversion factor: total scale to free scale
`tot2sws`	-	pH conversion factor: total scale to seawater scale
`sws2free`	-	pH conversion factor: seawater scale to free scale
`sws2tot`	-	pH conversion factor: seawater scale to total scale
`K0_CO2`	mol/(kg-soln*atm)	Henry's constant for CO_2
`K0_O2`	mol/(kg-soln*atm)	Henry's constant for O_2
`fCO2atm`	atm	atmospheric fugacity of CO_2
`fO2atm`	atm	atmospheric fugacity of O_2
`CO2_sat`	mol/kg-soln	CO_2 saturation concentration at an atmospheric fugacity of `fCO2atm`
`O2_sat`	mol/kg-soln	O_2 saturation concentration at an atmospheric fugacity of `fO2atm`
`K_W`	$(mol/kg\text{-}soln)^2$, free pH scale	stoichiometric equilibrium ion product of H_2O: $K^*_W = [H^+][OH^-]$
`K_HSO4`	mol/kg-soln, free pH scale	stoichiometric equilibrium constant $K^*_{HSO_4^-} = [H^+][SO_4^{2-}]/[HSO_4^-]$
`K_HF`	mol/kg-soln, free pH scale	stoichiometric equilibrium constant $K^*_{HF} = [H^+][F^-]/[HF]$
`K_CO2`	mol/kg-soln, free pH scale	stoichiometric equilibrium constant $K^*_{CO_2} = [H^+][HCO_3^-]/[CO_2]$
`K_HCO3`	mol/kg-soln, free pH scale	stoichiometric equilibrium constant $K^*_{HCO_3^-} = [H^+][CO_3^{2-}]/[HCO_3^-]$
`K_BOH3`	mol/kg-soln, free pH scale	stoichiometric equilibrium constant $K^*_{B(OH)_3} = [H^+][B(OH)_4^-]/[B(OH)_3]$

`K_NH4`	mol/kg-soln, free pH scale	stoichiometric equilibrium constant $K^*_{NH_4^+} = [H^+][NH_3]/[NH_4^+]$
`K_H2S`	mol/kg-soln, free pH scale	stoichiometric equilibrium constant $K^*_{H_2S} = [H^+][HS^-]/[H_2S]$
`K_H3PO4`	mol/kg-soln, free pH scale	stoichiometric equilibrium constant $K^*_{H_3PO_4} = [H^+][H_2PO_4^-]/[H_3PO_4]$
`K_H2PO4`	mol/kg-soln, free pH scale	stoichiometric equilibrium constant $K^*_{H_2PO_4^-} = [H^+][HPO_4^{2-}]/[H_2PO_4^-]$
`K_HPO4`	mol/kg-soln, free pH scale	stoichiometric equilibrium constant $K^*_{HPO_4^{2-}} = [H^+][PO_4^{3-}]/[HPO_4^{2-}]$
`K_SiOH4`	mol/kg-soln, free pH scale	stoichiometric equilibrium constant $K^*_{Si(OH)_4} = [H^+][SiO(OH)_3^-]/[Si(OH)_4]$
`K_SiOOH3`	mol/kg-soln, free pH scale	stoichiometric equilibrium constant $K^*_{SiO(OH)_3^-} = [H^+][SiO_2(OH)_2^{2-}]/[SiO(OH)_3^-]$
`K_HNO2`	mol/kg-soln; mol/kg-H2O; mol/l	approximate value for equilibrium constant $K^*_{HNO_2} = [H^+][NO_2^-]/[HNO_2]$
`K_HNO3`	mol/kg-soln; mol/kg-H2O; mol/l	approximate value for equilibrium constant $K^*_{HNO_3} = [H^+][NO_3^-]/[HNO_3]$
`K_H2SO4`	mol/kg-soln; mol/kg-H2O; mol/l	approximate value for equilibrium constant $K^*_{H_2SO_4} = [H^+][HSO_4^-]/[H_2SO_4]$
`K_HS`	mol/kg-soln; mol/kg-H2O; mol/l	approximate value for equilibrium constant $K^*_{HS^-} = [H^+][S^{2-}]/[HS^-]$
`Ksp_calcite`	$(mol/kg\text{-}soln)^2$	stoichiometric equilibrium solubility product of calcite: $Ksp^*_{cal} = [Ca^{2+}][CO_3^{2-}]$
`Ksp_aragonite`	$(mol/kg\text{-}soln)^2$	stoichiometric equilibrium solubility product of aragonite: $Ksp^*_{ara} = [Ca^{2+}][CO_3^{2-}]$
`TA`	mol/kg-soln	[TA], total alkalinity
`pH`	-, free scale	pH
`fCO2`	atm	fugacity of CO_2 in the water (i.e. in a small volume of air equilibrated with the water)
`CO2`	mol/kg-soln	$[CO_2]$
`HCO3`	mol/kg-soln	$[HCO_3^-]$
`CO3`	mol/kg-soln	$[CO_3^{2-}]$
`BOH3`	mol/kg-soln	$[B(OH)_3]$
`BOH4`	mol/kg-soln	$[B(OH)_4^-]$
`OH`	mol/kg-soln	$[OH^-]$
`H3PO4`	mol/kg-soln	$[H_3PO_4]$
`H2PO4`	mol/kg-soln	$[H_2PO_4^-]$
`HPO4`	mol/kg-soln	$[HPO_4^{2-}]$
`PO4`	mol/kg-soln	$[PO_4^{3-}]$
`SiOH4`	mol/kg-soln	$[Si(OH)_4]$
`SiOOH3`	mol/kg-soln	$[SiO(OH)_3^-]$
`SiO2OH2`	mol/kg-soln	$[SiO_2(OH)_2^{2-}]$
`H2S`	mol/kg-soln	$[H_2S]$
`HS`	mol/kg-soln	$[HS^-]$

`S2min`	mol/kg-soln	$[S^{2-}]$
`NH4`	mol/kg-soln	$[NH_4^+]$
`NH3`	mol/kg-soln	$[NH_3]$
`H2SO4`	mol/kg-soln	$[H_2SO_4]$
`HSO4`	mol/kg-soln	$[HSO_4^-]$
`SO4`	mol/kg-soln	$[SO_4^{2-}]$
`HF`	mol/kg-soln	$[HF]$
`F`	mol/kg-soln	$[F^-]$
`HNO3`	mol/kg-soln	$[HNO_3]$
`NO3`	mol/kg-soln	$[NO_3^-]$
`HNO2`	mol/kg-soln	$[HNO_2]$
`NO2`	mol/kg-soln	$[NO_2^-]$
`omega_calcite`	-	saturation state Ω with respect to calcite
`omega_aragonite`	-	saturation state Ω with respect to aragonite
`revelle`	-	Revelle factor (redundant in current version of **AquaEnv**)
`c1`	-	ionization fraction $c_1 = [CO_2]/[\sum CO_2]$
`c2`	-	ionization fraction $c_2 = [HCO_3^-]/[\sum CO_2]$
`c3`	-	ionization fraction $c_3 = [CO_3^{2-}]/[\sum CO_2]$
`dTAdSumCO2`	-	$\frac{\partial[TA]}{[\partial \sum CO_2]}$ with $[TA] = f([H^+], [\sum CO_2], ...)$
`b1`	-	ionization fraction $b_1 = [B(OH)_3]/[\sum B(OH)_3]$
`b2`	-	ionization fraction $b_2 = [B(OH)_4^-]/[\sum B(OH)_3]$
`dTAdSumBOH3`	-	$\frac{\partial[TA]}{[\partial \sum B(OH)_3]}$ with $[TA] = f([H^+], [\sum CO_2], ...)$
`so1`	-	ionization fraction $so_1 = [H_2SO_4]/[\sum H_2SO_4]$
`so2`	-	ionization fraction $so_2 = [HSO_4^-]/[\sum H_2SO_4]$
`so3`	-	ionization fraction $so_3 = [SO_4^{2-}]/[\sum H_2SO_4]$
`dTAdSumH2SO4`	-	$\frac{\partial[TA]}{[\partial \sum H_2SO_4]}$ with $[TA] = f([H^+], [\sum CO_2], ...)$
`f1`	-	ionization fraction $f_1 = [HF]/[\sum HF]$
`f2`	-	ionization fraction $f_1 = [F^-]/[\sum HF]$
`dTAdSumHF`	-	$\frac{\partial[TA]}{[\partial \sum HF]}$ with $[TA] = f([H^+], [\sum CO_2], ...)$
`p1`	-	ionization fraction $p_1[H_3PO_4]/[\sum H_3PO_4]$
`p2`	-	ionization fraction $p_2[H_2PO_4^-]/[\sum H_3PO_4]$
`p3`	-	ionization fraction $p_3[HPO_4^{2-}]/[\sum H_3PO_4]$
`p4`	-	ionization fraction $p_4[PO_4^{3-}]/[\sum H_3PO_4]$
`dTAdSumH3PO4`	-	$\frac{\partial[TA]}{[\partial \sum H_3PO_4]}$ with $[TA] = f([H^+], [\sum CO_2], ...)$
`si1`	-	ionization fraction $si_1 = [Si(OH)_4]/[\sum Si(OH)_4]$
`si2`	-	ionization fraction $si_2 = [SiO(OH)_3]/[\sum Si(OH)_4]$
`si3`	-	ionization fraction $si_3 = [SiO_2(OH)_2]/[\sum Si(OH)_4]$

`dTAdSumSumSiOH4`	-	$\frac{\partial[\mathrm{TA}]}{[\partial\sum\mathrm{Si(OH)_4}]}$ with $[\mathrm{TA}] = \mathrm{f}([\mathrm{H}^+], [\sum\mathrm{CO_2}], ...)$
`s1`	-	ionization fraction $s_1 = [\mathrm{H_2S}]/[\sum\mathrm{H_2S}]$
`s2`	-	ionization fraction $s_2 = [\mathrm{HS^-}]/[\sum\mathrm{H_2S}]$
`s3`	-	ionization fraction $s_3 = [\mathrm{S^{2-}}]/[\sum\mathrm{H_2S}]$ Note that we do assume that $\mathrm{S^{2-}}$ exists. However, s_3 is very small.
`dTAdSumH2S`	-	$\frac{\partial[\mathrm{TA}]}{[\partial\sum\mathrm{H_2S}]}$ with $[\mathrm{TA}] = \mathrm{f}([\mathrm{H}^+], [\sum\mathrm{CO_2}], ...)$
`n1`	-	ionization fraction $n_1 = [\mathrm{NH_4^+}]/[\sum\mathrm{NH_4^+}]$
`n2`	-	ionization fraction $n_2 = [\mathrm{NH_3}]/[\sum\mathrm{NH_4^+}]$
`dTAdSumNH4`	-	$\frac{\partial[\mathrm{TA}]}{[\partial\sum\mathrm{NH_4^+}]}$ with $[\mathrm{TA}] = \mathrm{f}([\mathrm{H}^+], [\sum\mathrm{CO_2}], ...)$
`na1`	-	ionization fraction $na_1 = [\mathrm{HNO_3}]/[\sum\mathrm{HNO_3}]$
`na2`	-	ionization fraction $na_2 = [\mathrm{NO_3^-}]/[\sum\mathrm{HNO_3}]$
`dTAdSumHNO3`	-	$\frac{\partial[\mathrm{TA}]}{[\partial\sum\mathrm{HNO_3}]}$ with $[\mathrm{TA}] = \mathrm{f}([\mathrm{H}^+], [\sum\mathrm{CO_2}], ...)$
`ni1`	-	ionization fraction $ni_1 = [[\mathrm{HNO_2}]/[\sum\mathrm{HNO_2}]$
`ni2`	-	ionization fraction $ni_2 = [[\mathrm{NO_2^-}]/[\sum\mathrm{HNO_2}]$
`dTAdSumHNO2`	-	$\frac{\partial[\mathrm{TA}]}{[\partial\sum\mathrm{HNO_2}]}$ with $[\mathrm{TA}] = \mathrm{f}([\mathrm{H}^+], [\sum\mathrm{CO_2}], ...)$
`dTAdH`	-	$\frac{\partial[\mathrm{TA}]}{[\partial[\mathrm{H}^+]}$: buffer factor with $[\mathrm{TA}] = \mathrm{f}([\mathrm{H}^+], [\sum\mathrm{CO_2}], ...)$
`dTAdKdKdS`	-	$\sum_\mathrm{i}\frac{\partial[\mathrm{TA}]}{\partial\mathrm{K_i^*}}\frac{\partial\mathrm{K_i^*}}{\partial\mathrm{S}}$ with $[\mathrm{TA}] = \mathrm{f}([\mathrm{H}^+], [\sum\mathrm{CO_2}], ..., \mathrm{K_i^*})$
`dTAdKdKdT`	-	$\sum_\mathrm{i}\frac{\partial[\mathrm{TA}]}{\partial\mathrm{K_i^*}}\frac{\partial\mathrm{K_i^*}}{\partial\mathrm{T}}$ with $[\mathrm{TA}] = \mathrm{f}([\mathrm{H}^+], [\sum\mathrm{CO_2}], ..., \mathrm{K_i^*})$
`dTAdKdKdp`	-	$\sum_\mathrm{i}\frac{\partial[\mathrm{TA}]}{\partial\mathrm{K_i^*}}\frac{\partial\mathrm{K_i^*}}{\partial\mathrm{p}}$ with $[\mathrm{TA}] = \mathrm{f}([\mathrm{H}^+], [\sum\mathrm{CO_2}], ..., \mathrm{K_i^*})$
`dTAdKdKdSumH2SO4`	-	$\sum_\mathrm{i}\frac{\partial[\mathrm{TA}]}{\partial\mathrm{K_i^*}}\frac{\partial\mathrm{K_i^*}}{\partial[\sum\mathrm{H_2SO_4}]}$ with $[\mathrm{TA}] = \mathrm{f}([\mathrm{H}^+], [\sum\mathrm{CO_2}], ..., \mathrm{K_i^*})$
`dTAdKdKdSumHF`	-	$\sum_\mathrm{i}\frac{\partial[\mathrm{TA}]}{\partial\mathrm{K_i^*}}\frac{\partial\mathrm{K_i^*}}{\partial[\sum\mathrm{HF}]}$ with $[\mathrm{TA}] = \mathrm{f}([\mathrm{H}^+], [\sum\mathrm{CO_2}], ..., \mathrm{K_i^*})$

3. Using **AquaEnv**

3.1. Calling the “K” functions directly

The elements `K_CO2`, `K_HCO3`, `K_BOH3`, `K_W`, `K_HSO4`, `K_HF`, `K_NH4`, `K_H2S`, `K_H3PO4`, `K_H2PO4`, `K_HPO4`, `K_SiOH4`, `K_SiOOH3`, `K0_CO2`, `K0_O2`, `Ksp_aragonite`, and `Ksp_calcite` can be calculated directly. This is done via functions that bear the same name as those elements.

```
> K_CO2(S = 15, t = 30)

[1] 9.926218e-07
attr(,"unit")
[1] "mol/kg-soln"
attr(,"pH scale")
[1] "free"

> K0_CO2(S = 15, t = 30)

[1] 0.02780196
attr(,"unit")
[1] "mol/(kg-soln*atm)"

> Ksp_calcite(S = 15, t = 30, p = 100)

[1] 1.721545e-07
attr(,"unit")
[1] "(mol/kg-soln)^2"
```

One of the input arguments can also be a vector:

```
> K_CO2(S = 10:15, t = 30)

[1] 9.280129e-07 9.413916e-07 9.545927e-07 9.675593e-07 9.802471e-07
[6] 9.926218e-07
attr(,"unit")
[1] "mol/kg-soln"
attr(,"pH scale")
[1] "free"
```

3.2. The function `aquaenv`

Minimal `aquaenv` *definition*

Minimally, an object of class `aquaenv` can be defined with just a temperature (`t`) and salinity (`S`) value

```
> ae <- aquaenv(S = 30, t = 15)
> ae$K_CO2

[1] 9.274931e-07
attr(,"unit")
[1] "mol/kg-soln"
attr(,"pH scale")
[1] "free"
```

Optionally the pressure (here the gauge pressure p) can be given. As in the above case, the returned object of class `aquaenv` then contains a standard set of elements as shown by the `names` command.

```
> ae <- aquaenv(S = 30, t = 15, p = 10)
> names(ae)
```

```
 [1] "S"            "t"            "p"            "T"
 [5] "Cl"           "I"            "P"            "Pa"
 [9] "d"            "density"      "SumCO2"       "SumNH4"
[13] "SumH2S"       "SumHNO3"      "SumHNO2"      "SumH3PO4"
[17] "SumSiOH4"     "SumBOH3"      "SumH2SO4"     "SumHF"
[21] "Br"           "ClConc"       "Na"           "Mg"
[25] "Ca"           "K"            "Sr"           "molal2molin"
[29] "free2tot"     "free2sws"     "tot2free"     "tot2sws"
[33] "sws2free"     "sws2tot"      "K0_CO2"       "K0_O2"
[37] "fCO2atm"      "fO2atm"       "CO2_sat"      "O2_sat"
[41] "K_W"          "K_HSO4"       "K_HF"         "K_CO2"
[45] "K_HCO3"       "K_BOH3"       "K_NH4"        "K_H2S"
[49] "K_H3PO4"      "K_H2PO4"      "K_HPO4"       "K_SiOH4"
[53] "K_SiOOH3"     "K_HNO2"       "K_HNO3"       "K_H2SO4"
[57] "K_HS"         "Ksp_calcite"  "Ksp_aragonite"
```

```
> ae$Ksp_calcite
```

```
[1] 3.643728e-07
attr(,"unit")
[1] "(mol/kg-soln)^2"
```

The pressure can also be given via the total pressure (`P`) or the water depth (`d`). The atmospheric pressure can be given as well (`Pa`, e.g. for the case of a mountain lake). Furthermore, if the depth is given, the latitude (`lat`) can also be specified (default is 0 degrees).

```
> ae <- aquaenv(S = 30, t = 15, p = 10)
> ae[c("p", "P")]
```

```
$p
[1] 10
attr(,"unit")
[1] "bar"

$P
[1] 11.01325
attr(,"unit")
[1] "bar"
```

```
> ae <- aquaenv(S = 30, t = 15, P = 10)
> unlist(ae[c("p", "P")])

       p        P
 8.98675 10.00000

> ae <- aquaenv(S = 30, t = 15, P = 10, Pa = 0.5)
> unlist(ae[c("p", "P")])

   p    P
 9.5 10.0

> ae <- aquaenv(S = 30, t = 15, d = 100)
> unlist(ae[c("p", "P")])

       p        P
10.05769 11.07094

> ae <- aquaenv(S = 30, t = 15, d = 100, lat = 51)
> unlist(ae[c("p", "P")])

       p        P
10.08985 11.10310
```

A minimal set of elements in an object of class `aquaenv` can be obtained by setting the flag `skeleton` to `TRUE`.

```
> ae <- aquaenv(S = 30, t = 15, p = 10, skeleton = TRUE)
> names(ae)

 [1] "S"         "t"         "p"         "T"         "Cl"        "I"        
 [7] "P"         "Pa"        "d"         "density"   "SumCO2"    "SumNH4"   
[13] "SumH2S"    "SumHNO3"   "SumHNO2"   "SumH3PO4"  "SumSiOH4"  "SumBOH3"  
[19] "SumH2SO4"  "SumHF"     "K_W"       "K_HSO4"    "K_HF"      "K_CO2"    
[25] "K_HCO3"    "K_BOH3"    "K_NH4"     "K_H2S"     "K_H3PO4"   "K_H2PO4"  
[31] "K_HPO4"    "K_SiOH4"   "K_SiOOH3"  "K_HNO2"    "K_HNO3"    "K_H2SO4"  
[37] "K_HS"     
```

Defining the complete `aquaenv` *system in different ways*

If enough information is given to define a complete speciation, i.e. either one of the pairs $[\sum CO_2]$ and pH, $[\sum CO_2]$ and [TA], $[\sum CO_2]$ and $[CO_2]$, or $[\sum CO_2]$ and fCO_2, a full `aquaenv` system can be defined.

```
> S      <- 30
> t      <- 15
> p      <- 10
> SumCO2 <- 0.0020
> pH     <- 8
> TA     <- 0.002142233
> fCO2   <- 0.0005272996
> CO2    <- 2.031241e-05
> ae <- aquaenv(S, t, p, SumCO2 = SumCO2, pH = pH)
> ae$TA

[1] 0.002134693
attr(,"unit")
[1] "mol/kg-soln"

> ae <- aquaenv(S, t, p, SumCO2 = SumCO2, TA = TA)
> ae$pH

[1] 8.019601
attr(,"pH scale")
[1] "free"

> ae <- aquaenv(S, t, p, SumCO2 = SumCO2, CO2 = CO2)
> ae$pH

[1] 7.993364
attr(,"pH scale")
[1] "free"

> names(ae)

 [1] "S"               "t"               "p"               "T"
 [5] "Cl"              "I"               "P"               "Pa"
 [9] "d"               "density"         "SumCO2"          "SumNH4"
[13] "SumH2S"          "SumHNO3"         "SumHNO2"         "SumH3PO4"
[17] "SumSiOH4"        "SumBOH3"         "SumH2SO4"        "SumHF"
[21] "Br"              "ClConc"          "Na"              "Mg"
[25] "Ca"              "K"               "Sr"              "molal2molin"
[29] "free2tot"        "free2sws"        "tot2free"        "tot2sws"
[33] "sws2free"        "sws2tot"         "K0_CO2"          "K0_O2"
[37] "fCO2atm"         "fO2atm"          "CO2_sat"         "O2_sat"
[41] "K_W"             "K_HSO4"          "K_HF"            "K_CO2"
[45] "K_HCO3"          "K_BOH3"          "K_NH4"           "K_H2S"
[49] "K_H3PO4"         "K_H2PO4"         "K_HPO4"          "K_SiOH4"
[53] "K_SiOOH3"        "K_HNO2"          "K_HNO3"          "K_H2SO4"
[57] "K_HS"            "Ksp_calcite"     "Ksp_aragonite"   "TA"
```

```
[61] "pH"                "fCO2"              "CO2"               "HCO3"
[65] "CO3"               "BOH3"              "BOH4"              "OH"
[69] "H3PO4"             "H2PO4"             "HPO4"              "PO4"
[73] "SiOH4"             "SiOOH3"            "SiO2OH2"           "H2S"
[77] "HS"                "S2min"             "NH4"               "NH3"
[81] "H2SO4"             "HSO4"              "SO4"               "HF"
[85] "F"                 "HNO3"              "NO3"               "HNO2"
[89] "NO2"               "omega_calcite"     "omega_aragonite"
```

As seen above, a full speciation is calculated along with the pH or total alkalinity, respectively. If only pH or total alkalinity is needed, the calculation of the full speciation can be toggled off. Furthermore, the flag `skeleton` also works for a full system.

```
> ae <- aquaenv(S, t, p, SumCO2 = SumCO2, pH = pH, speciation = FALSE)
> names(ae)

 [1] "S"               "t"               "p"               "T"
 [5] "Cl"              "I"               "P"               "Pa"
 [9] "d"               "density"         "SumCO2"          "SumNH4"
[13] "SumH2S"          "SumHNO3"         "SumHNO2"         "SumH3PO4"
[17] "SumSiOH4"        "SumBOH3"         "SumH2SO4"        "SumHF"
[21] "Br"              "ClConc"          "Na"              "Mg"
[25] "Ca"              "K"               "Sr"              "molal2molin"
[29] "free2tot"        "free2sws"        "tot2free"        "tot2sws"
[33] "sws2free"        "sws2tot"         "K0_CO2"          "K0_O2"
[37] "fCO2atm"         "fO2atm"          "CO2_sat"         "O2_sat"
[41] "K_W"             "K_HSO4"          "K_HF"            "K_CO2"
[45] "K_HCO3"          "K_BOH3"          "K_NH4"           "K_H2S"
[49] "K_H3PO4"         "K_H2PO4"         "K_HPO4"          "K_SiOH4"
[53] "K_SiOOH3"        "K_HNO2"          "K_HNO3"          "K_H2SO4"
[57] "K_HS"            "Ksp_calcite"     "Ksp_aragonite"   "TA"
[61] "pH"              "fCO2"            "CO2"
```

```
> ae <- aquaenv(S, t, p, SumCO2 = SumCO2, pH = pH, speciation = FALSE,
+               skeleton = TRUE)
> names(ae)

 [1] "S"          "t"          "p"          "T"          "Cl"         "I"
 [7] "P"          "Pa"         "d"          "density"    "SumCO2"     "SumNH4"
[13] "SumH2S"     "SumHNO3"    "SumHNO2"    "SumH3PO4"   "SumSiOH4"   "SumBOH3"
[19] "SumH2SO4"   "SumHF"      "K_W"        "K_HSO4"     "K_HF"       "K_CO2"
[25] "K_HCO3"     "K_BOH3"     "K_NH4"      "K_H2S"      "K_H3PO4"    "K_H2PO4"
[31] "K_HPO4"     "K_SiOH4"    "K_SiOOH3"   "K_HNO2"     "K_HNO3"     "K_H2SO4"
[37] "K_HS"       "TA"         "pH"         "fCO2"       "CO2"
```

All the quantities needed for the explicit pH modelling approaches as given in Hofmann *et al.* (2008) and Hofmann *et al.* (2010a) can be calculated by setting the flag `dsa` to `TRUE`.The flag `revelle` is redundant; the Revelle factor can instead be calculated analytically using `BufferFactors$RF` following Hagens and Middelburg (2016). This is further detailed in section XX.

```
> ae <- aquaenv(S, t, p, SumCO2 = SumCO2, fCO2 = fCO2, dsa = TRUE)
> ae$dTAdH
```

```
[1] -16070.66
attr(,"unit")
[1] "(mol-TA/kg-soln)/(mol-H/kg-soln)"
attr(,"pH scale")
[1] "free"
```

```
> ae$revelle
```

```
NULL
```

If an ambivalent situation is created because the user enters too much information, an error message is displayed

```
> ae <- aquaenv(S, t, p, SumCO2 = SumCO2, CO2 = CO2, fCO2 = fCO2)
```

```
[1] "Error! Overdetermined system: entered fCO2: 0.0005272996 , calculated fCO2: 0.00052729949660769"
[1] "Please enter only one of: pH, TA, CO2, or fCO2."
```

```
> ae <- aquaenv(S, t, p, SumCO2 = SumCO2, pH = pH, TA = TA)
```

```
[1] "Error! Overdetermined system: entered TA: 0.002142233 , calculated TA: 0.0021346934487761"
[1] "Please enter only one of: pH, TA, CO2, or fCO2."
```

```
> ae <- aquaenv(S, t, p, SumCO2 = SumCO2, pH = pH, CO2 = CO2)
```

```
[1] "Error! Overdetermined system: entered pH: 8 , calculated pH: 7.99336441966162"
[1] "Please enter only one of: pH, TA, CO2, or fCO2."
```

```
> ae <- aquaenv(S, t, p, SumCO2 = SumCO2, pH = pH, fCO2 = fCO2)
```

```
[1] "Error! Overdetermined system: entered pH: 8 , calculated pH: 7.99336433791644"
[1] "Please enter only one of: pH, TA, CO2, or fCO2."
```

```
> ae <- aquaenv(S, t, p, SumCO2 = SumCO2, TA = TA, CO2 = CO2)
```

```
[1] "Error! Overdetermined system: entered TA: 0.002142233 , calculated TA: 0.00213218846905773"
[1] "Please enter only one of: pH, TA, CO2, or fCO2."
```

```
> ae <- aquaenv(S, t, p, SumCO2 = SumCO2, TA = TA, fCO2 = fCO2)
```

```
[1] "Error! Overdetermined system: entered TA: 0.002142233 , calculated TA: 0.00213218843834296"
[1] "Please enter only one of: pH, TA, CO2, or fCO2."
```

Calculating $[\sum \mathrm{CO_2}]$

$[\sum \mathrm{CO_2}]$ can be calculated by giving a constant pair of either pH and $[\mathrm{CO_2}]$, pH and $\mathrm{fCO_2}$, pH and [TA], [TA] and $[\mathrm{CO_2}]$, or [TA] and $\mathrm{fCO_2}$

```
> fCO2    <- 0.0006943363
> CO2     <- 2.674693e-05
> pH      <- 7.884892
> TA      <- 0.0021
> S <- 30
> t <- 15
> p <- 10
> ae <- aquaenv(S, t, p, SumCO2 = NULL, pH = pH, CO2 = CO2)
> ae$SumCO2

[1] 0.002033881
attr(,"unit")
[1] "mol/kg-soln"

> ae <- aquaenv(S, t, p, SumCO2 = NULL, pH = pH, fCO2 = fCO2)
> ae$SumCO2

[1] 0.002033881
attr(,"unit")
[1] "mol/kg-soln"

> ae <- aquaenv(S, t, p, SumCO2 = NULL, pH = pH, TA = TA)
> ae$SumCO2

[1] 0.002005564
attr(,"unit")
[1] "mol/kg-soln"

> ae <- aquaenv(S, t, p, SumCO2 = NULL, TA = TA, CO2 = CO2)
> ae$SumCO2

[1] 0.002007323
attr(,"unit")
[1] "mol/kg-soln"

> ae <- aquaenv(S, t, p, SumCO2 = NULL, TA = TA, fCO2 = fCO2)
> ae$SumCO2

[1] 0.002007323
attr(,"unit")
[1] "mol/kg-soln"
```

Cloning an object of class `aquaenv`

It is possible to clone an object of class `aquaenv`, either 1 to 1 or with different pH, [TA], or $K^{*}_{CO_2}$.

```
> S      <- 30
> t      <- 15
> SumCO2 <- 0.0020
> TA     <- 0.00214
> ae <- aquaenv(S, t, SumCO2 = SumCO2, TA = TA)
> ae$pH

[1] 8.017927
attr(,"pH scale")
[1] "free"

> ae1 <- aquaenv(ae = ae)      # this is the same
> ae1$pH

[1] 8.017927
attr(,"pH scale")
[1] "free"

> ae2 <- aquaenv(ae = ae, pH = 9)
> c(ae$TA, ae2$TA)

[1] 0.002140000 0.002947263

> ae3 <- aquaenv(ae = ae, TA = 0.002)
> c(ae$pH, ae3$pH)

[1] 8.017927 7.555188

> K_CO2 <- 1e-6
> ae4 <- aquaenv(ae = ae, k_co2 = 1e-6)
> c(ae$TA, ae4$TA)

[1] 0.00214 0.00214
```

Note that `k_co2` as an input argument is in lower case (in constrast to the element `K_CO2` of class `aquaenv`!

Preparing input arguments

Input arguments for the function `aquaenv` need to be in mol/kg-solution and on the free pH scale. Data in other concentration units or pH scales can be converted using the function `convert`. This is a rather complex function, but its typical use is:

```
convert(x, vartype, what, S, t, p, SumH2SO4, SumHF, khf)
convert(x, from, to, factor, convattr)
```

```
> S <- 10
> t <- 15
> pH_NBS      <- 8.142777
> SumCO2molar <- 0.002016803
> (pH_free     <- convert(pH_NBS,      "pHscale", "nbs2free",    S = S, t = t))

[1] 8.050993
attr(,"pH scale")
[1] "free"

> (SumCO2molin <- convert(SumCO2molar, "conc",    "molar2molin", S = S, t = t))

[1] 0.002003213
attr(,"unit")
[1] "mol/kg-soln"

> ae <- aquaenv(S, t, SumCO2 = SumCO2molin, pH = pH_free)
> ae$pH

[1] 8.050993
attr(,"pH scale")
[1] "free"

> ae$SumCO2

[1] 0.002003213
attr(,"unit")
[1] "mol/kg-soln"
```

Vectors as input variables

One of the input variables for the function `aquaenv` may be a vector. All the other input variables are then assumed to be constant. The elements of the resulting two dimensional object of class `aquaenv` are then vectors containing the elements as a function of the input variable for which a vector is given.

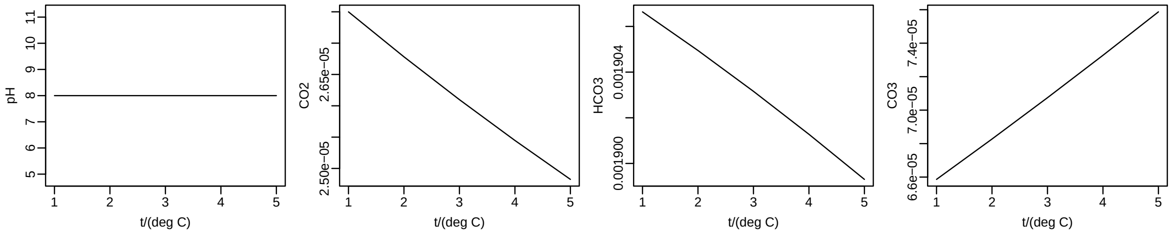
```
> SumCO2 <- 0.0020
> pH     <- 8
> S      <- 30
> t      <- 1:5
> p      <- 10
> ae <- aquaenv(S, t, p, SumCO2 = SumCO2, pH = pH)
> rbind(t = ae$t, TA = ae$TA)
```

```
           [,1]        [,2]        [,3]        [,4]        [,5]
t  1.000000000 2.000000000 3.000000000 4.000000000 5.000000000
TA 0.002073831 0.002077889 0.002081984 0.002086118 0.002090292
```

A two dimensional object of class `aquaenv` can be visualized using the `plot` function. For convenience of the user, the default setting for the `plot` function for an object of class `aquaenv` results in a new plotting device being opened. Setting the flag `newdevice` to `FALSE` prevents that.

The `plot` function plots all elements of the respective object of class `aquaenv`. This, however, might not be what the user wants, especially if a larger plotting device cannot properly displayed like in the case above. In this case the parameter `what` can be used. Note, however, that the default setting for calling `plot` with the parameter `what` is that `mfrow=c(1,1)`. So if one wants to plot several elements, `mfrow` needs to be set to a suitable value.

```
> plot(ae, xval = t, xlab = "t/(deg C)",
+   what = c("pH", "CO2", "HCO3", "CO3"),
+   mfrow = c(1, 4))
```



Other input arguments can also be vectors, e.g. the obvious S, t, or p:

```
> ae <- aquaenv(S=20:24, t=15, p=10, SumCO2 = SumCO2, pH = pH, dsa = TRUE)
> rbind(ae$S, ae$TA)
```

But also [$\sum \mathrm{CO_2}$], [TA], pH and [$\sum \mathrm{NH_4^+}$] can be vectors, e.g.

```
> ae <- aquaenv(20, 10, SumCO2=seq(0.001, 0.002, 0.00025), TA = 0.002)
> rbind(ae$SumCO2, ae$pH, ae$HCO3)
```

```
             [,1]         [,2]        [,3]        [,4]        [,5]
[1,] 0.0010000000 0.0012500000 0.001500000 0.001750000 0.002000000
[2,] 9.8479526195 9.3438747892 8.954638606 8.516985645 7.707874204
[3,] 0.0002798428 0.0006918592 0.001127442 0.001556787 0.001914737
```

Calculating [$\sum \mathrm{CO_2}$] *from input vectors*

The functionality of calculating [$\sum \mathrm{CO_2}$] can also be used together with vectors as input variables.

```
> ae <- aquaenv(S = 30, t = 11:15, SumCO2 = NULL, pH = pH, CO2 = CO2,
+               dsa = TRUE)
> ae$SumCO2

[1] 0.002441625 0.002493119 0.002544334 0.002595224 0.002645739
attr(,"unit")
[1] "mol/kg-soln"
```

Conversion from and to a dataframe

Objects of class `aquaenv` can be converted to an R `data.frame` to further post-process them with standard R means. Similarly, R `data.frames` can be converted to objects of class `aquaenv` to use the plotting facilities that exist for objects of class `aquaenv`. This can be helpful for plotting output of a dynamic model run, e.g. from R package **deSolve**, as will be shown later in this document.

```
> ae <- aquaenv(S = 30, t = 11:15, SumCO2 = NULL, pH = 8, CO2 = 2e-5)
> aedataframe <- as.data.frame(ae)
> dim(aedataframe)

[1]  5 91

> aedataframe[, 1:3]

   S  t p
1 30 11 0
2 30 12 0
3 30 13 0
4 30 14 0
5 30 15 0

> aetest      <- aquaenv(ae = aedataframe, from.data.frame = TRUE)
```

Converting elements in an object of class `aquaenv`

Elements of an object of class `aquaenv` are calculated in predefined units, e.g., the concentrations are in unit mol/kg-solution (molinity). The function `convert` can be used to convert all elements in an object of class `aquaenv` that share a common attribute, e.g. the unit.

```
> ae <- aquaenv(S = 30, t = 10)
> ae$SumBOH3

[1] 0.0003563636
attr(,"unit")
[1] "mol/kg-soln"
```

```
> ae <- convert(ae, "mol/kg-soln", "umol/kg-H2O", 1e6/ae$molal2molin, "unit")
> ae$SumBOH3
```

```
[1] 367.442
attr(,"unit")
[1] "umol/kg-H2O"
```

Quantities needed for explicit pH modelling

As mentioned above, the quantities needed for the explicit pH modelling approach (direct substitution approach - DSA) as presented by Hofmann *et al.* (2008) can be calculated with the function `aquaenv` by setting the flag `dsa` to `TRUE`.

```
> ae <- aquaenv(S = 30, t = 15, d = 10, SumCO2 = 0.002, pH = 8, dsa = TRUE)
```

This command calculates the buffer factor and the partial derivatives of `[TA]` with respect to other summed quantities referred to in Hofmann *et al.* (2008). This functionality is expanded with the function `BufferFactors`, which is discussed in a separate subsection.

```
> ae$dTAdH
```

```
[1] -16393.9
attr(,"unit")
[1] "(mol-TA/kg-soln)/(mol-H/kg-soln)"
attr(,"pH scale")
[1] "free"
```

```
> ae$dTAdSumCO2
```

```
[1] 1.042189
attr(,"unit")
[1] "(mol-TA/kg-soln)/(mol-SumCO2/kg-soln)"
```

Moreover, setting the flag `dsa` to `TRUE` calculates the sums of the partial derivatives of [TA] with respect to the equilibrium constants (K^*'s) multiplied with the partial derivatives of the respective equilibrium constant with one of their variables (i.e., S, T, d, $[\sum H_2SO_4]$, or $[\sum HF]$) as introduced in Hofmann *et al.* (2009).

```
> ae$dTAdKdKdS
```

```
[1] 3.85e-06
attr(,"unit")
[1] "(mol-TA/kg-soln)/\"psu\""
```

```
> ae$dTAdKdKdSumH2SO4
```

```
[1] -0.001039435
attr(,"unit")
[1] "(mol-TA/kg-soln)/(mol-SumH2SO4/kg-soln)"
```

Furthermore, the ionization fractions used for the pH dependent fractional stoichiometric pH modelling approach described in Hofmann *et al.* (2010a) are calculated as well

```
> ae$c1

[1] 0.01009794
```

3.3. The use of `BufferFactors`

An introduction to `BufferFactors`

As described above, by setting the flag `dsa` to `TRUE`, the function `aquaenv` calculates the total buffer factor and the partial derivatives of `[TA]` with respect to other summed quantities referred to in Hofmann *et al.* (2008). The function `BufferFactors` allows for a more generic, analytical calculation of partial derivatives related to carbonate system calculations, as presented in Hagens and Middelburg (2016). This includes the sensitivity of pH and concentrations of [CO_2] and other acid-base species to a change in ocean chemistry, i.e. a change in [TA] or other summed quantities.

`BufferFactors` internally calls the `aquaenv` function and uses its output to calculate the sensitivities. Its output is a list of elements, the first of which is of class `aquaenv` and is the output of the internal call to `aquaenv`. `BufferFactors` runs without specifying any input variables; in this case, the global contemporary surface ocean values given in Table 4 of Hagens and Middelburg (2016) are used as input.

```
> BF <- BufferFactors()
> names(BF)

 [1] "ae"         "dTA.dH"     "dtotX.dH"   "dTA.dX"     "dtotX.dX"   "dTA.dpH"
 [7] "dtotX.dpH"  "dH.dTA"     "dH.dtotX"   "dX.dTA"     "dX.dtotX"   "dpH.dTA"
[13] "dpH.dtotX"  "beta.H"     "RF"

> BF$dtotX.dpH

     SumCO2
0.005080433
```

Specifying input parameters for `BufferFactors`

An object of class `aquaenv` where a complete speciation is calculated can be used as input for `BufferFactors`:

```
> ae <- aquaenv(S = 30, t = 15, d = 10, SumCO2 = 0.002, pH = 8.1,
+               skeleton = TRUE)
> BF <- BufferFactors(ae = ae)
> BF$RF

[1] 12.76839
```

Note that `ae$dTAdH` and `BF$beta.H` both represent the total buffer factor and are thus the same:

```
> ae <- aquaenv(S = 30, t = 15, d = 10, SumCO2 = 0.002, pH = 8.1, dsa = TRUE)
> BF <- BufferFactors(ae = ae)
> cbind(ae$dTAdH,BF$beta.H)

         [,1]     [,2]
[1,] -23910.4 -23910.4
```

As an alternative to an object of class `aquaenv`, input parameters can be given by specifying the argument `parameters`. This argument is a vector which can contain the following inputs: `DIC`, `TotNH3`, `TotP`, `TotNO3`, `TotNO2`, `TotS`, `TotSi`, `TB`, `TotF`, `TotSO4`, `sal`, `temp`, `pres`, `Alk`. All of these should be supplied in the same units as for `aquaenv`, i.e. mol/kg-soln.

```
> parameters <- c(DIC = 0.002, Alk = 0.0022)
> BF <- BufferFactors(parameters = parameters)
> BF$RF

[1] 11.64887
```

Note that if temperature, salinity, pressure or a specific total concentration is not given in `parameters`, the default value of the contemporary global ocean is used. However, if an object of class `aquaenv` is provided, all of its total concentrations are used, even if they are not specified in the calculation of the object, as is the case for e.g. $[\sum \mathrm{NH}_4^+]$ in the example above. However, one or more output variables of `aquenv` can be overruled by specifying them in the `parameters` argument, and specifying both `aquaenv` and `parameters` as input parameters for `BufferFactors`:

```
> ae <- aquaenv(S = 30, t = 15, d = 10, SumCO2 = 0.002, pH = 8.1,
+               skeleton = TRUE)
> BF <- BufferFactors(ae = ae)
> BF$RF

[1] 12.76839

> parameters <- c(Alk = 0.0022)
> BF_2 <- BufferFactors(ae = ae, parameters = parameters)
> BF_2$RF
```

```
[1] 11.98122
```

Specifying which output is produced by `BufferFactors`

By default, `BufferFactors` only displays the sensitivities related to [$\sum \mathrm{CO_2}$]. This is indicated by the argument `species`, which defaults to `species = c("SumCO2")`. However, sensitivities can be calculkated for any total concentration contributing to [TA], or any acid-base species contributing to this total concentration:

```
> BF <- BufferFactors(species = c("CO2", "HCO3", "CO3", "SumNH4"))
> BF$dtotX.dX
```

```
       CO2       HCO3        CO3     SumNH4
183.151849   1.117004  10.071717   1.042292
```

In the case when species are defined which corresponding total concentration equals zero, the corresponding output produces `NaN`:

```
> ae <- aquaenv(S = 30, t = 15, d = 10, SumCO2 = 0.002, pH = 8.1,
+               skeleton = TRUE)
> BF <- BufferFactors(ae = ae, species = c("SumCO2", "SumNH4"))
> BF$dtotX.dX
```

```
  SumCO2   SumNH4
126.1004      NaN
```

Other arguments of `BufferFactors`

`BufferFactors` allows using different pre-defined sets of equilibrium constants, as well as specifying specific values for them, similarly to how this is done in the function `aquaenv`:

```
> BF <- BufferFactors(k1k2 = "roy")
> BF$RF
```

```
[1] 9.847236
```

```
> BF <- BufferFactors(k_co2 = 1e-6)
> BF$RF
```

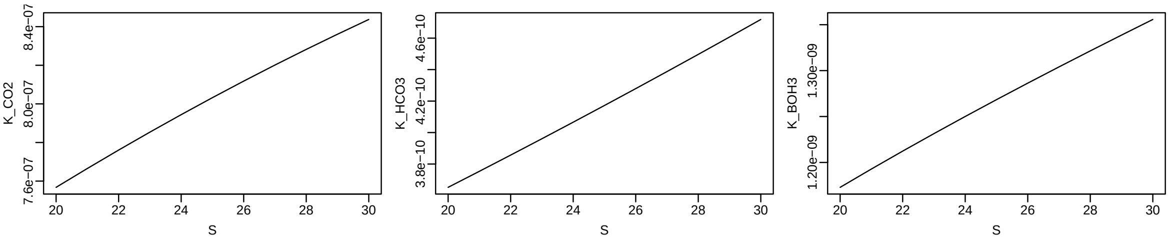
```
[1] 9.891083
```

3.4. The `plot.aquaenv` function

In the previous sections, the `plot` function has been introduced. When the first element of the arguments list of `plot` is an object of class `aquaenv`, the function `plot.aquaenv` is called. This is a multifunctional tool to visualize information contained in an object of class `aquaenv`. For the convenience of the users, `plot.aquaenv` combines the call of standard R plotting functions and the previous call of the function `par` to set parameters like `mfrow`, `mar`, etc. as well as the opening of a plotting device with a certain size. As already shown above, setting the flag `newdevice` to `FALSE` suppresses the opening of a new plotting device.

For example

```
> ae <- aquaenv(20:30, 10)
> plot(ae, xval = 20:30, xlab = "S", what = c("K_CO2", "K_HCO3", "K_BOH3"),
+   size = c(10, 2), mfrow = c(1,3))
```



Furthermore the parameter `device` can be specified which allows the user to write the plots to .eps and .pdf files. The parameter `filename` can be used to specify a filename other than the default filename "aquaenv".

Whereas these features make the function `plot.aquaenv` different from standard R plotting functions, when setting the flags `newdevice` and `setpar` to `FALSE`, `plot.aquaenv` behaves like a "normal" R plotting function.

Furthermore, the function `plot.aquaenv` can be used to create "cumulative" plots and "Bjerrum" plots. This will be explained in upcoming sections.

3.5. Using objects of class `aquaenv` in dynamic models

Ordinary dynamic equation models

It is convenient to use objects of class `aquaenv` in a dynamic model, e.g. solved using the R package **deSolve**.

This is illustrated with an example[4].

First, the input parameters are specified:

```
> parameters <- list(
+       S         = 25        , # psu
+       t_min     = 5         , # degrees C
+       t_max     = 25        , # degrees C
+
+       k         = 0.4       , # 1/d            proportionality factor for air-water exchange
+       rOx       = 0.0000003 , # mol-N/(kg*d)  maximal rate of oxic mineralisation
+       rNitri    = 0.0000002 , # mol-N/(kg*d)  maximal rate of nitrification
+       rPP       = 0.000006  , # mol-N/(kg*d)  maximal rate of primary production
+
```

---

[4] For information about how to set up a dynamic model with **deSolve**, consult the documentation of **deSolve**

```
+        ksDINPP    = 0.000001  , # mol-N/kg
+        ksNH4PP    = 0.000001  , # mol-N/kg
+
+        D          = 0.1       , # 1/d           (dispersive) transport coefficient
+
+        O2_io      = 0.000296  , # mol/kg-soln
+        NO3_io     = 0.000035  , # mol/kg-soln
+        SumNH4_io  = 0.000008  , # mol/kg-soln
+        SumCO2_io  = 0.002320  , # mol/kg-soln
+        TA_io      = 0.002435  , # mol/kg-soln
+
+        C_Nratio   = 8         , # mol C/mol N   C:N ratio of organic matter
+
+        a           = 30       , # time at which PP begins
+        b           = 50       , # time at which PP stops again
+
+        modeltime   = 100        # duration of the model
+                   )
```

Next the model function that calculates the derivatives is defined. In this function, an object of class `aquaenv` is created in each timestep, some of its elements are used to calculated kinetic rate expressions and part of the object is returned as output. A function that returns the temperature is defined first.

```
> temperature <- with (parameters,
+     approxfun(x = 0:101,
+                y = c(seq(t_min, t_max, (t_max-t_min)/50),
+                      seq(t_max, t_min, -(t_max-t_min)/50)))
+      )
> boxmodel <- function(time, state, parameters) {
+   with (
+        as.list(c(state, parameters)),     {
+          t       <- temperature(time)
+          ae      <- aquaenv(S = S, t = t, SumCO2 = SumCO2,
+                             SumNH4 = SumNH4, TA = TA)
+
+          ECO2    <- k * (ae$CO2_sat - ae$CO2)
+          EO2     <- k * (ae$O2_sat  - O2)
+
+          # dilution
+          TO2     <- D*(O2_io     - O2)
+          TNO3    <- D*(NO3_io    - NO3)
+          TSumNH4 <- D*(SumNH4_io - SumNH4)
+          TTA     <- D*(TA_io     - TA)
+          TSumCO2 <- D*(SumCO2_io - SumCO2)
+
+          RNit      <- rNitri * SumNH4/(SumNH4+1e-8)
+
+          ROx       <- rOx
+          ROxCarbon <- ROx * C_Nratio
+
+          pNH4PP <- 0
+          RPP <- 0
+
+          if ((time > a) && (time < b)) {
+              RPP    <- rPP * ((SumNH4+NO3)/(ksDINPP + (SumNH4+NO3)))
+              pNH4PP <- SumNH4/(SumNH4+NO3)
+          }
+
+          RPPCarbon <- RPP * C_Nratio
+
+          dO2     <- TO2     + EO2 - ROxCarbon - 2*RNit  + (2-2*pNH4PP)*RPP + RPPCarbon
+          dNO3    <- TNO3    + RNit -(1-pNH4PP)*RPP
+
```

```
+           dSumCO2 <- TSumCO2 + ECO2 + ROxCarbon - RPPCarbon
+           dSumNH4 <- TSumNH4 + ROx  - RNit - pNH4PP*RPP
+
+           dTA     <- TTA     + ROx - 2*RNit -(2*pNH4PP-1)*RPP
+
+           ratesofchanges <- c(dO2, dNO3, dSumNH4, dSumCO2, dTA)
+
+           return(list(ratesofchanges, ae[c("t", "NH4", "NH3", "pH")]))
+         } )
+ }
```

The model is solved, and the output can be plotted in the same way as any `deSolve` object. The package **deSolve** (Soetaert *et al.* 2010) has to be loaded first.

```
>  initialstate <-   with (parameters,
+      c(O2=O2_io, NO3=NO3_io, SumNH4=SumNH4_io, SumCO2=SumCO2_io, TA=TA_io))
>  times        <- 1:100
>  output       <- vode(initialstate,times,boxmodel,parameters, hmax = 1)
```

```
> plot(output)
```

Models using the explicit pH modelling approach

3.5.2.1 In one single model

Since an object of class `aquaenv` can contain all quantities necessary to employ the explicit pH modelling approaches as introduced by Hofmann *et al.* (2008, 2009, 2010a), they can readily be used in an explicit pH model.

As an example, we give a model that calculates the pH in the “classical” way in every timestep using `aquaenv`, also employs the explicit pH modelling approach (direct substitution approach - DSA) given in Hofmann *et al.* (2008) and additionally employs fractional stoichiometry as given in Hofmann *et al.* (2010a). The pH evolution is thus calculated in three different ways which allows comparing the three values for consistency.

Again, a list of parameters is defined

```
> parameters <- list(
+      S          = 25    , # psu
+      t          = 15    , # degrees C
+
+      k          = 0.4       , # 1/d            proportionality factor for air-water exchange
+      rOx        = 0.0000003 , # mol-N/(kg*d)  maximal rate of oxic mineralisation
+      rNitri     = 0.0000002 , # mol-N/(kg*d)  maximal rate of nitrification
+      rPP        = 0.0000006 , # mol-N/(kg*d)  maximal rate of primary production
+
+      ksSumNH4   = 0.000001  , # mol-N/kg
+
+      D          = 0.1       , #   1/d            (dispersive) transport coefficient
+
+      SumNH4_io  = 0.000008  , # mol/kg-soln
+      SumCO2_io  = 0.002320  , # mol/kg-soln
+      TA_io      = 0.002435  , # mol/kg-soln
+
```

```
+       C_Nratio   = 8       , # mol C/mol N     C:N ratio of organic matter
+
+       a          = 5       , # time from which PP begins
+       b          = 20       , # time where PP shuts off again
+
+       modeltime  = 30        # duration of the model
+                   )
```

and a model function is defined. Again, an object of class `aquaenv` is created in each timestep and the respective elements are used.

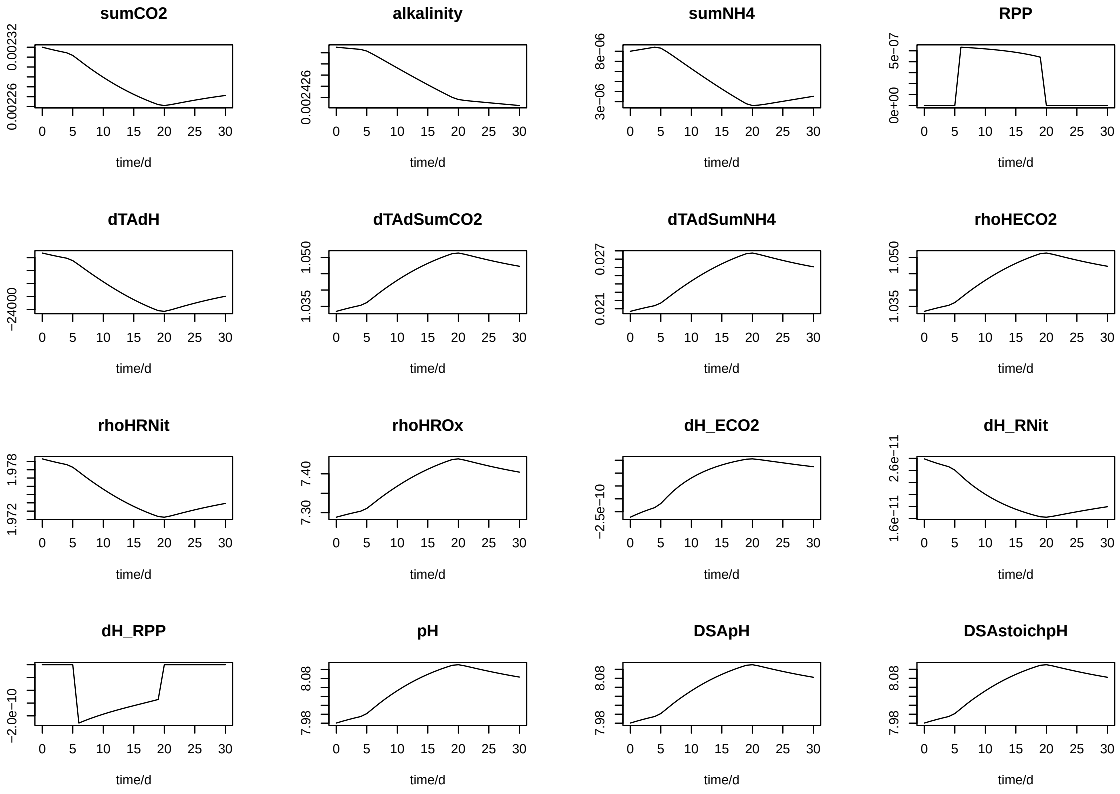
```
> boxmodel <- function(timestep, currentstate, parameters)
+ {
+   with (
+         as.list(c(currentstate,parameters)),
+         {
+           # the "classical" implicit pH calculation method is applied in aquaenv
+           ae <- aquaenv(S=S, t=t, SumCO2=sumCO2,
+             SumNH4=sumNH4, TA=alkalinity, dsa=TRUE)
+
+           ECO2    <- k * (ae$CO2_sat - ae$CO2)
+
+           RNit      <- rNitri
+           ROx       <- rOx
+
+           if ((timestep > a) && (timestep < b))
+               RPP <- rPP * (sumNH4/(ksSumNH4 + sumNH4))
+           else
+               RPP <- 0
+
+           dsumCO2 <- ECO2 + C_Nratio*ROx - C_Nratio*RPP
+           dsumNH4 <- ROx  - RNit - RPP
+           dalkalinity <- ROx - 2*RNit - RPP
+
+           # The DSA pH
+           dH    <- (dalkalinity - (dsumCO2*ae$dTAdSumCO2 + dsumNH4*ae$dTAdSumNH4))/ae$dTAdH
+           DSApH <- -log10(H)
+
+           # The DSA pH using pH dependent fractional stoichiometry (= using partitioning coefficients)
+           rhoHECO2 <- ae$c2 + 2*ae$c3
+           rhoHRNit <- 1 + ae$n1
+           rhoHROx  <- C_Nratio * (ae$c2 + 2*ae$c3) - ae$n1
+           rhoHRPP  <- -(C_Nratio * (ae$c2 + 2*ae$c3)) + ae$n1
+
+           dH_ECO2  <- rhoHECO2*ECO2/(-ae$dTAdH)
+           dH_RNit  <- rhoHRNit*RNit/(-ae$dTAdH)
+           dH_ROx   <- rhoHROx*ROx  /(-ae$dTAdH)
+           dH_RPP   <- rhoHRPP*RPP  /(-ae$dTAdH)
+
+           dH_stoich   <- dH_ECO2 + dH_RNit + dH_ROx + dH_RPP
+           DSAstoichpH <- -log10(H_stoich)
+
+           ratesofchanges <- c(dsumNH4, dsumCO2, dalkalinity, dH, dH_stoich)
+           processrates   <- c(ECO2=ECO2, RNit=RNit, ROx=ROx, RPP=RPP)
+           DSA            <- c(DSApH=DSApH, rhoHECO2=rhoHECO2, rhoHRNit=rhoHRNit, rhoHROx=rhoHROx,
+                               rhoHRPP=rhoHRPP, dH_ECO2=dH_ECO2, dH_RNit=dH_RNit, dH_ROx=dH_ROx,
+                               dH_RPP=dH_RPP, DSAstoichpH=DSAstoichpH)
+
+           return(list(ratesofchanges, processrates, DSA, ae))
+         }
+         )
+ }
```

The model is solved

```
> with (as.list(parameters),
+       {
+         H_init       <<- 10^(-(aquaenv(S=S, t=t, SumCO2=SumCO2_io, SumNH4=SumNH4_io, TA=TA_io,
+                                    speciation=FALSE)$pH))
+         initialstate <<- c(sumNH4=SumNH4_io, sumCO2=SumCO2_io, alkalinity =TA_io, H=H_init,
+                              H_stoich=H_init)
+         times        <<- c(0:modeltime)
+       })
> output       <- vode(initialstate, times, boxmodel, parameters, hmax=1)
```

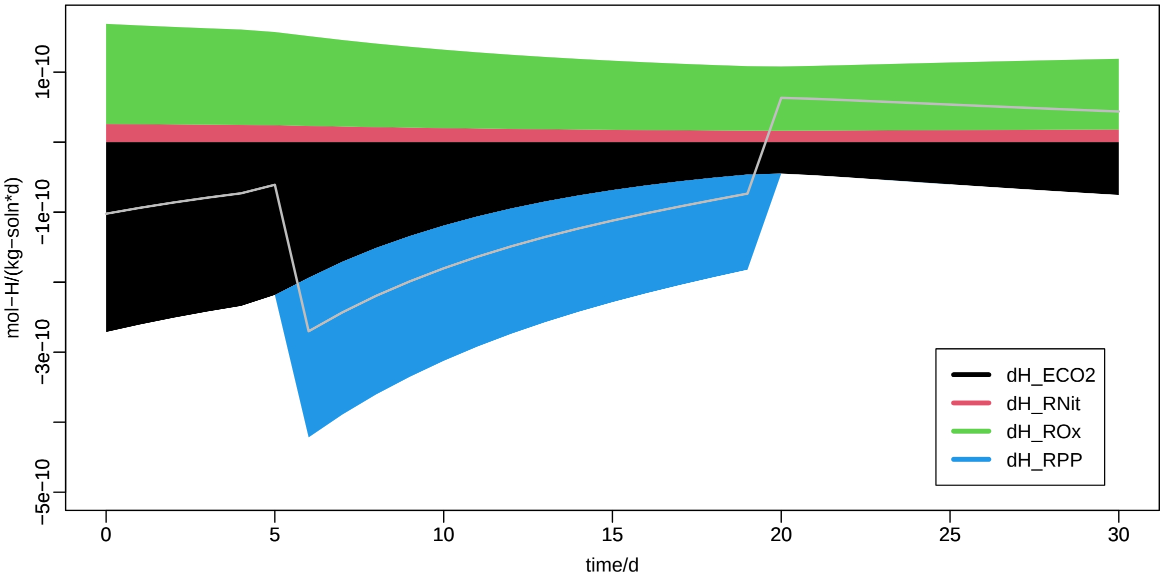
and output can be plotted, again using `plot.aquaenv`. Note that here the parameter `what` is used.

```
> select <- c("sumCO2", "alkalinity", "sumNH4", "RPP", "dTAdH", "dTAdSumCO2",
+            "dTAdSumNH4","rhoHECO2", "rhoHRNit", "rhoHROx","dH_ECO2","dH_RNit",
+            "dH_RPP","pH", "DSApH", "DSAstoichpH")
> plot(output, which = select, xlab="time/d", mfrow=c(4,4))
```



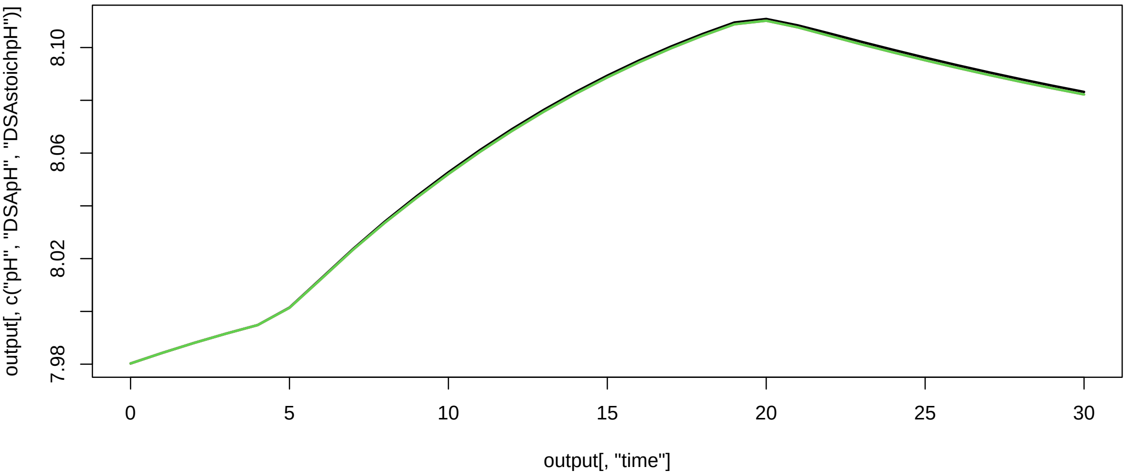
Here, the cumulative plotting functionality of `plot.aquaenv` can be employed as well to visualize the influences of the different kinetically modelled processes on [H^+].

```
> what <- c("dH_ECO2", "dH_RNit", "dH_ROx", "dH_RPP")
> plot(aquaenv(ae=as.data.frame(output), from.data.frame=TRUE),
+   xval = times, what = what, xlab = "time/d", size = c(7,5),
+   ylab = "mol-H/(kg-soln*d)", legendposition = "bottomright", cumulative = TRUE)
```



Finally, the pH values calculated with the three different methods can be plotted in one single graph to see that they are identical, i.e. the three methods of pH calculation are consistent with each other

```
> matplot(output[, "time"],
+   output[, c("pH", "DSApH", "DSAstoichpH")], type = "l", lty = 1, lwd = 2)
```



3.5.2.2 In three separate models

3.5.2.2.1 The implicit pH modelling approach

A list of parameters:

```
> parameters <- list(
+     t           = 15        , # degrees C
+     S           = 35        , # psu
+
+     SumCO2_t0   = 0.002     , # mol/kg-soln  (comparable to Wang2005)
+     TA_t0       = 0.0022    , # mol/kg-soln  (comparable to Millero1998)
+
+     kc          = 0.5       , # 1/d                proportionality factor for air-water exchange
+     kp          = 0.000001  , # mol/(kg-soln*d)          max rate of calcium carbonate precipitation
+     n           = 2.0       , # -                 exponent for kinetic rate law of precipitation
+
+     modeltime   = 20        , # d              duration of the model
+     outputsteps = 100         #                number of outputsteps
+                   )
```

The model function:

```
> boxmodel <- function(timestep, currentstate, parameters)
+ {
+   with (
+         as.list(c(currentstate,parameters)),
+         {
+           ae    <- aquaenv(S=S, t=t, SumCO2=SumCO2, TA=TA, SumSiOH4=0,
+                            SumBOH3=0, SumH2SO4=0, SumHF=0)
+
+           Rc    <- kc * ((ae$CO2_sat) - (ae$CO2))
+           Rp    <- kp * (1-ae$omega_calcite)^n
+
+           dSumCO2 <- Rc - Rp
+           dTA     <- -2*Rp
+
+           ratesofchanges <- c(dSumCO2, dTA)
+
+           processrates   <- c(Rc=Rc, Rp=Rp)
+
+           return(list(ratesofchanges, list(processrates, ae)))
+         }
+         )
+ }
```

Solving the model:

```
> with (as.list(parameters),
+       {
+         initialstate <<- c(SumCO2=SumCO2_t0, TA=TA_t0)
+         times        <<- seq(0,modeltime,(modeltime/outputsteps))
+       })
> output       <<- vode(initialstate,times,boxmodel,parameters, hmax=1)
```

Visualisation of the results can be done similarly as above.

3.5.2.2.2 The explicit pH modelling approach

A list of parameters:

```
> parameters <- list(
+     S           = 35        , # psu
+     t           = 15        , # degrees C
+
+     SumCO2_t0   = 0.002     , # mol/kg-soln  (comparable to Wang2005)
```

```
+     TA_t0       = 0.0022    , # mol/kg-soln  (comparable to Millero1998)
+
+     kc          = 0.5       , # 1/d                 proportionality factor for air-water exchange
+     kp          = 0.000001  , # mol/(kg-soln*d)         max rate of calcium carbonate precipitation
+     n           = 2.0       , # -                 exponent for kinetic rate law of precipitation
+
+     modeltime   = 20        , # d              duration of the model
+     outputsteps = 100         #                number of outputsteps
+                    )
```

The model function:

```
> boxmodel <- function(timestep, currentstate, parameters)
+ {
+   with (
+         as.list(c(currentstate,parameters)),
+         {
+           ae    <- aquaenv(S=S, t=t, SumCO2=SumCO2, pH=-log10(H), SumSiOH4=0,
+                            SumBOH3=0, SumH2SO4=0, SumHF=0, dsa=TRUE)
+
+           Rc    <- kc * ((ae$CO2_sat) - (ae$CO2))
+           Rp    <- kp * (1-ae$omega_calcite)^n
+
+           dSumCO2 <- Rc - Rp
+
+           dHRc    <- (      -(ae$dTAdSumCO2*Rc   ))/ae$dTAdH
+           dHRp    <- (-2*Rp -(ae$dTAdSumCO2*(-Rp)))/ae$dTAdH
+           dH      <- dHRc + dHRp
+
+           ratesofchanges <- c(dSumCO2, dH)
+
+           processrates   <- c(Rc=Rc, Rp=Rp)
+           outputvars     <- c(dHRc=dHRc, dHRp=dHRp)
+
+           return(list(ratesofchanges, list(processrates, outputvars, ae)))
+         }
+         )
+ }
```

Solving the model:

```
> with (as.list(parameters),
+       {
+         aetmp <- aquaenv(S=S, t=t, SumCO2=SumCO2_t0, TA=TA_t0, SumSiOH4=0, SumBOH3=0, SumH2SO4=0, SumHF=0)
+         H_t0  <- 10^(-aetmp$pH)
+
+         initialstate <<- c(SumCO2=SumCO2_t0, H=H_t0)
+         times        <<- seq(0,modeltime,(modeltime/outputsteps))
+       })
> output       <- vode(initialstate,times,boxmodel,parameters, hmax=1)
```

Visualisation of the results can again be done similarly as above.

3.5.2.2.3 The fractional stoichiometric approach

A list of parameters:

```
> parameters <- list(
+      S           = 35        , # psu
+      t           = 15        , # degrees C
+
+      SumCO2_t0   = 0.002     , # mol/kg-soln  (comparable to Wang2005)
+      TA_t0       = 0.0022    , # mol/kg-soln  (comparable to Millero1998)
```

```
+
+      kc          = 0.5        , # 1/d                 proportionality factor for air-water exchange
+      kp          = 0.000001   , # mol/(kg-soln*d)         max rate of calcium carbonate precipitation
+      n           = 2.0        , # -                 exponent for kinetic rate law of precipitation
+
+      modeltime   = 20         , # d              duration of the model
+      outputsteps = 100          #                number of outputsteps
+                    )
```

The model function:

```
> boxmodel <- function(timestep, currentstate, parameters)
+ {
+   with (
+         as.list(c(currentstate,parameters)),
+         {
+           ae    <- aquaenv(S=S, t=t, SumCO2=SumCO2, pH=-log10(H), SumSiOH4=0,
+                            SumBOH3=0, SumH2SO4=0, SumHF=0, dsa=TRUE)
+
+           Rc    <- kc * ((ae$CO2_sat) - (ae$CO2))
+           Rp    <- kp * (1-ae$omega_calcite)^n
+
+           dSumCO2 <- Rc - Rp
+
+           rhoc    <- ae$c2 + 2*ae$c3
+           rhop    <- 2*ae$c1 + ae$c2
+
+           dHRc    <- rhoc*Rc/(-ae$dTAdH)
+           dHRp    <- rhop*Rp/(-ae$dTAdH)
+           dH      <- dHRc + dHRp
+
+           ratesofchanges <- c(dSumCO2, dH)
+
+           processrates   <- c(Rc=Rc, Rp=Rp)
+           outputvars     <- c(dHRc=dHRc, dHRp=dHRp, rhoc=rhoc, rhop=rhop)
+
+           return(list(ratesofchanges, list(processrates, outputvars, ae)))
+         }
+         )
+ }
```

Solving the model:

```
> with (as.list(parameters),
+       {
+         aetmp <- aquaenv(S=S, t=t, SumCO2=SumCO2_t0, TA=TA_t0, SumSiOH4=0,
+                          SumBOH3=0, SumH2SO4=0, SumHF=0)
+         H_t0  <- 10^(-aetmp$pH)
+
+         initialstate <<- c(SumCO2=SumCO2_t0, H=H_t0)
+         times        <<- seq(0,modeltime,(modeltime/outputsteps))
+         output       <<- as.data.frame(vode(initialstate,times,boxmodel,parameters, hmax=1))
+       })
```

Visualisation of the results can again be done similarly as above.

3.6. Titration simulation: the function `titration`

With the function `titration` **AquaEnv** provides a powerful tool to simulate titrations. A two dimensional object of class `aquaenv` will be created where the second dimension is the amount of titrant added. For this purpose, three extra elements are added to the `aquaenv` object that will be created:

element	unit	explanation
delta_conc_titrant	mol/kg-solution	the offset in concentration of the titrant that is caused by adding the titrant to the sample
delta_mass_titrant	kg	the amount of mass of titrant solution added
delta_moles_titrant	mol	the amount of moles of titrant added

Each one of these elements is a suitable `xval` for plotting an `aquaenv` object generated by `titration`.

Titration with HCl

The standard titration type is titration with hydrochloric acid (HCl). A simple example will illustrate this.

An object of class `aquaenv` needs to be created to define the initial conditions of the titration. That is temperature, salinity, depth, the concentrations of all summed quantities and the initial pH (or [TA]).

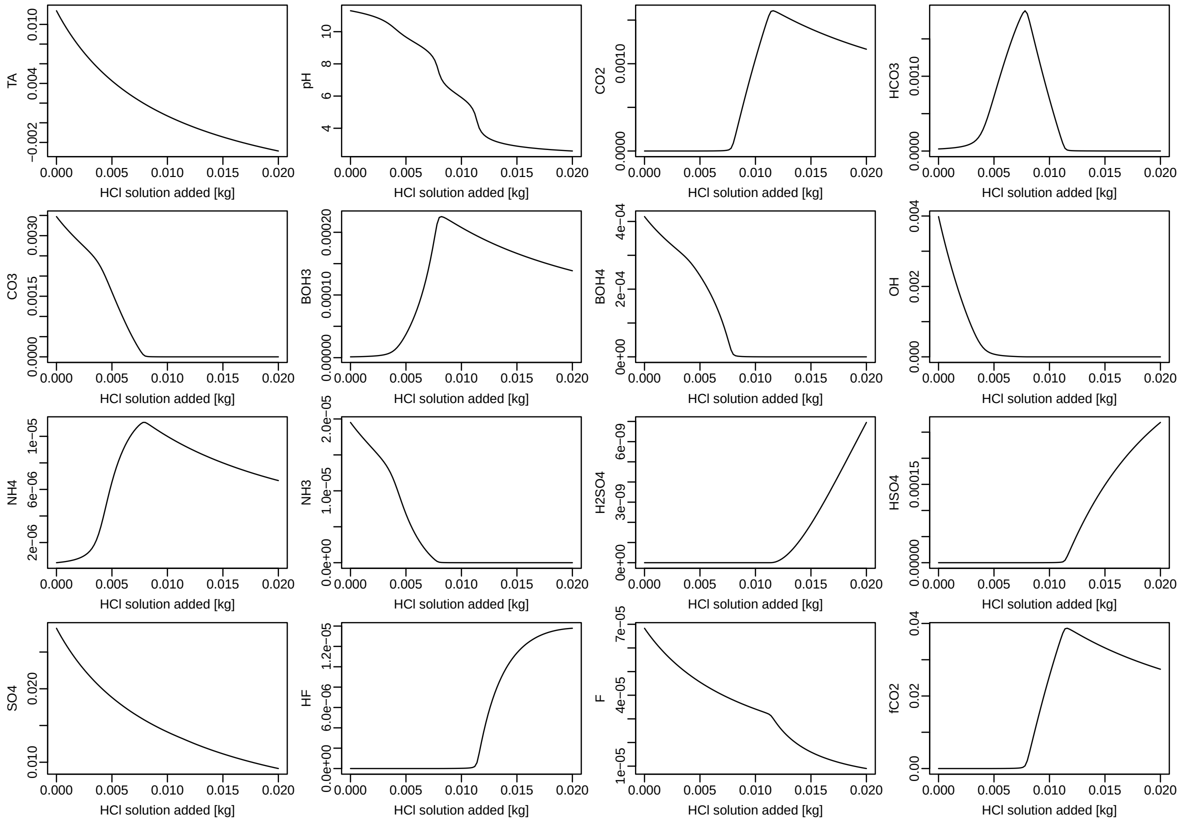
```
> ae_init <- aquaenv(S = 35, t = 15, SumCO2 = 0.0035, SumNH4 = 0.00002, pH = 11.3)
```

Then `titration` can be run to create the object describing the simulated titration. In this example the titrant is HCl of the relatively low concentration of 0.01 mol/kg-solution. The sample solution amounts to 10 g. To sweep a considerable pH range quite a lot of titrant needs to be added: 20 g. This means the salinity of the solution in the titration vessel will change due to dilution with the titrant solution. For this reason, the salinity of the titrant solution needs to be given via the parameter `S_titrant`. However, we assume the titrant does not contain borate, sulfate or fluoride, that is why we do not set the flag `seawater_titrant` to `TRUE`.

```
> ae <- titration(ae_init, mass_sample = 0.01, mass_titrant = 0.02,
+    conc_titrant = 0.01, S_titrant = 0.5, steps = 100)
```

To get a quick overview, all elements of the obtained `aquaenv` object can be plotted (not shown) but also a selection of elements can be plotted as a function of the added titrant mass.

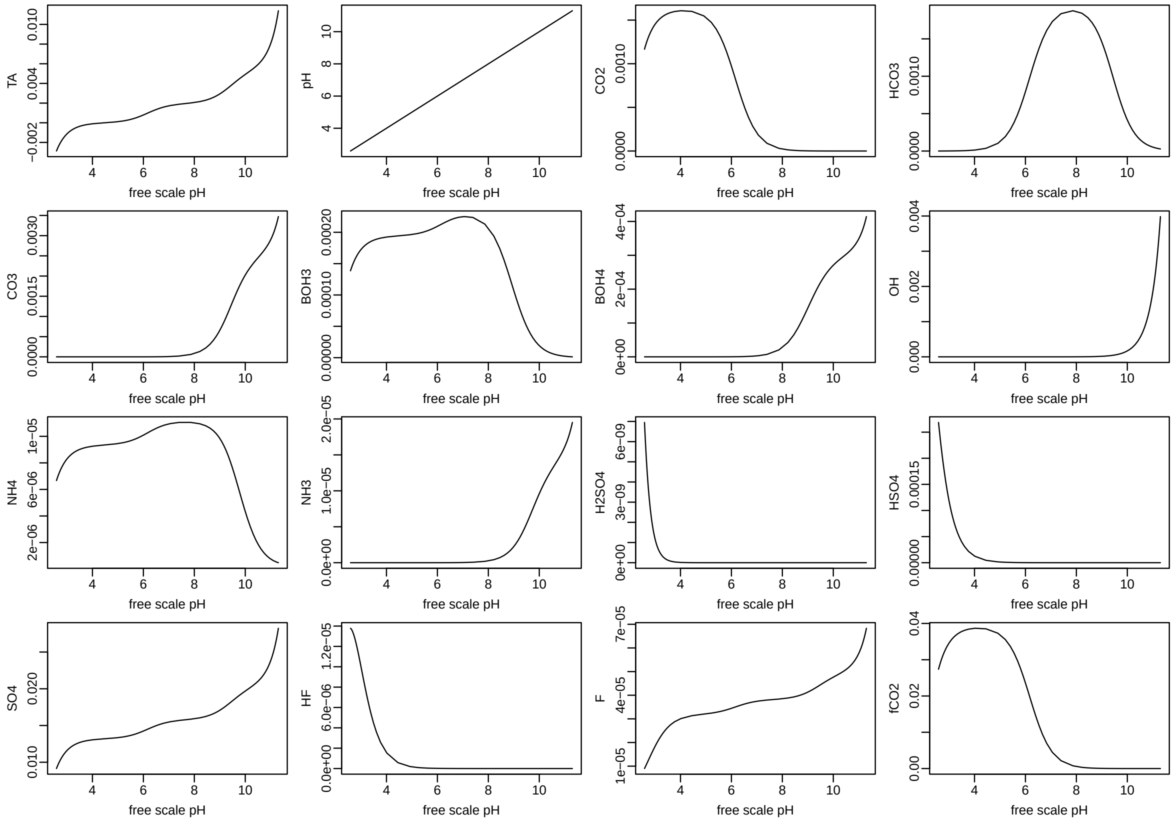
```
> what  <- c("TA", "pH", "CO2", "HCO3", "CO3", "BOH3", "BOH4", "OH",
+             "NH4", "NH3", "H2SO4", "HSO4", "SO4", "HF", "F", "fCO2")
> plot(ae, xval = ae$delta_mass_titrant, xlab = "HCl solution added [kg]",
+      what = what, size = c(12,8), mfrow = c(4,4))
```



It can also be plotted against titrant concentration, the moles of added titrant (not shown), or against other variables such as pH (figure)

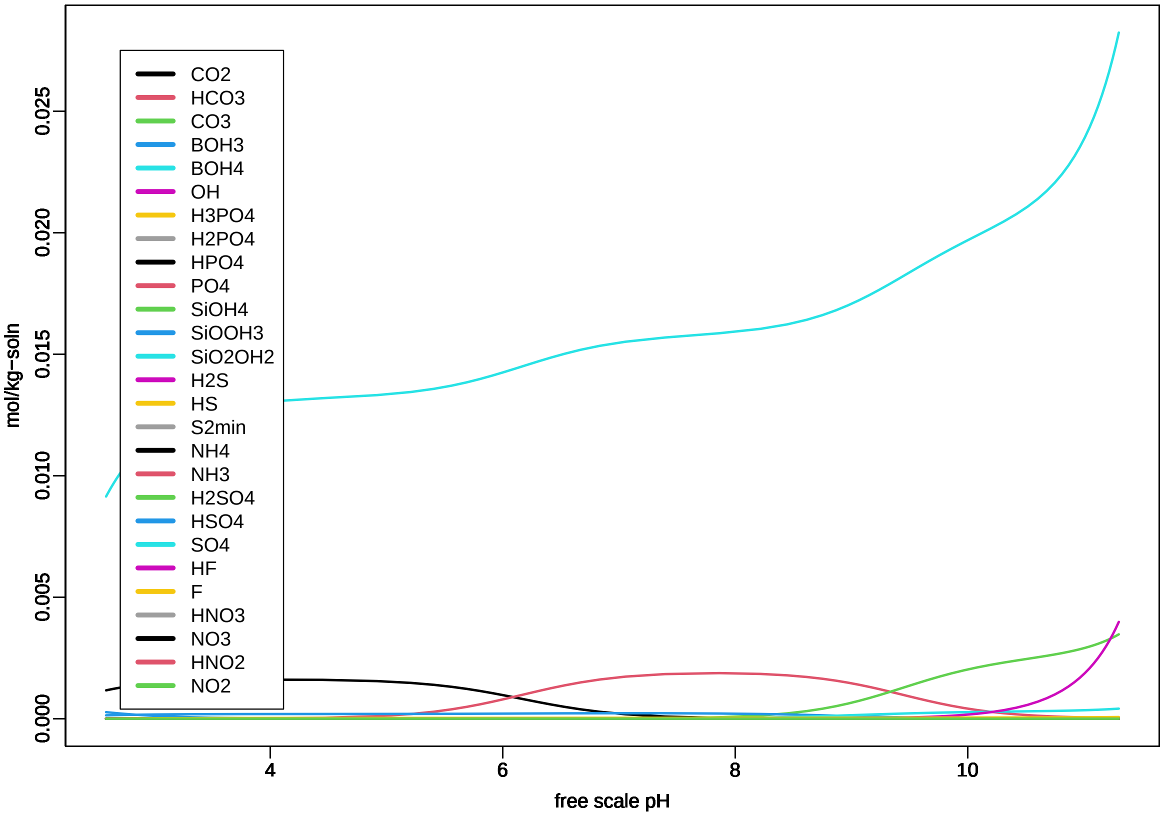
```
> plot(ae, xval = ae$delta_conc_titrant, what = what,
+      xlab = "[HCl] offset added [mol/kg-soln]",
+      size = c(14,10), mfrow = c(4,4))
> plot(ae, xval=ae$delta_moles_titrant, xlab = "HCl added [mol]",
+      what = what, size = c(14,10), mfrow = c(4,4))
```

```
> plot(ae, xval = ae$pH, xlab = "free scale pH", what = what,
+      size = c(12,8), mfrow = c(4,4))
```



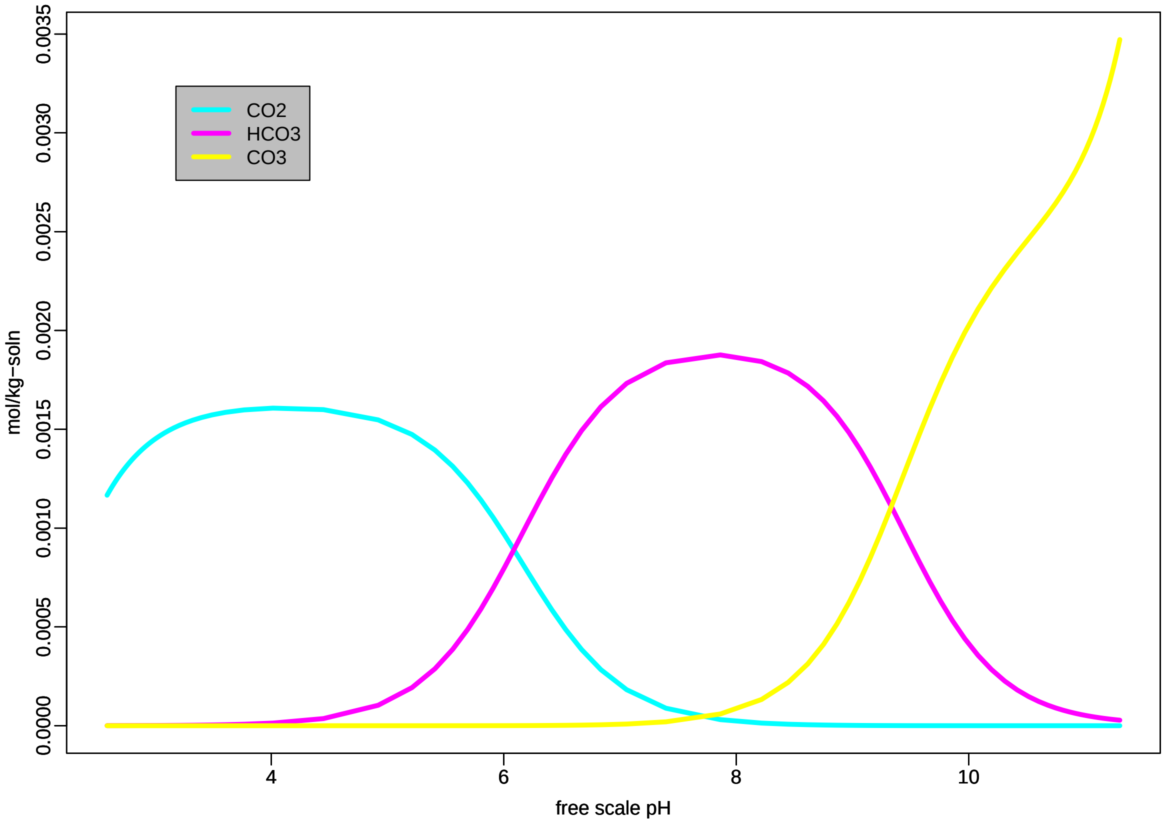
The function `plot.aquaenv` also offers the possibility of creating Bjerrum plots from objects obtained with `titration`. The simplest way to do that is:

```
> plot(ae, bjerrum = TRUE)
```



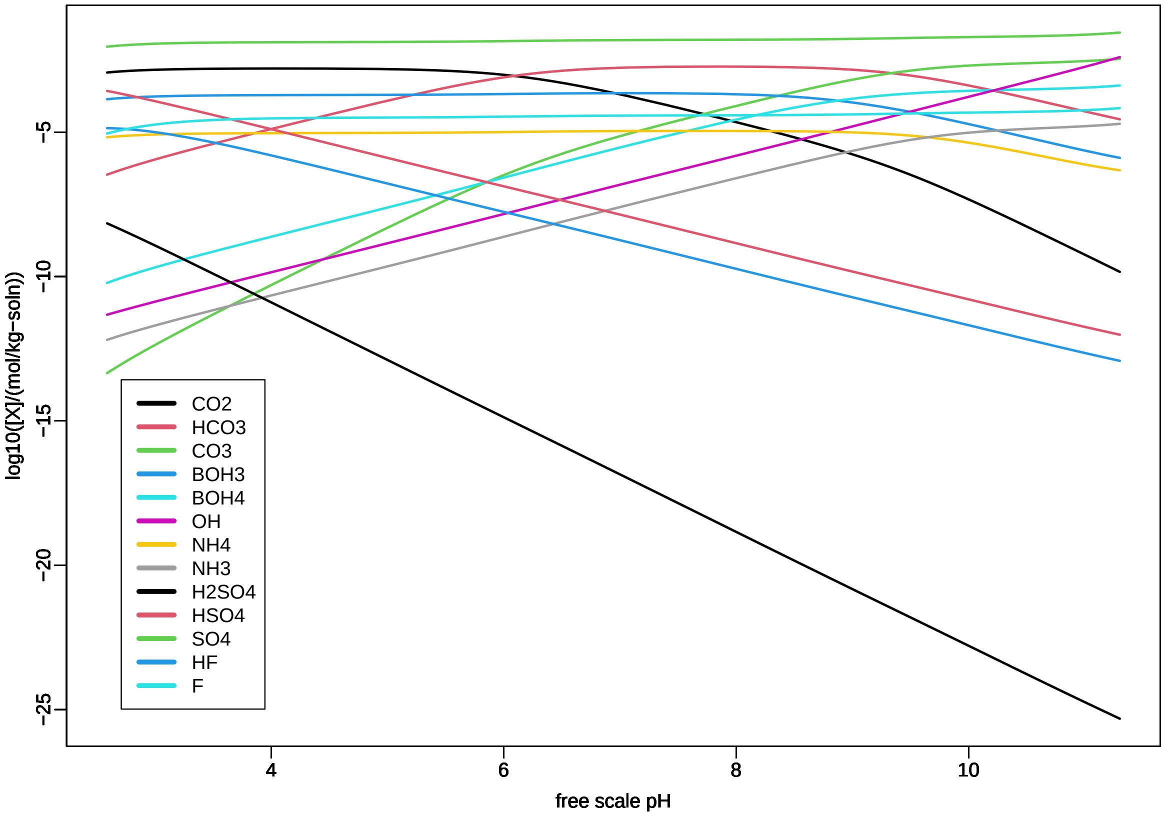
Or just select a few elements and customize the plotting style

```
> what  <- c("CO2", "HCO3", "CO3")
> plot(ae, what = what, bjerrum = TRUE, lwd = 4,
+   palette = c("cyan", "magenta", "yellow"),
+   bg = "gray", legendinset = 0.1, legendposition = "topleft")
```



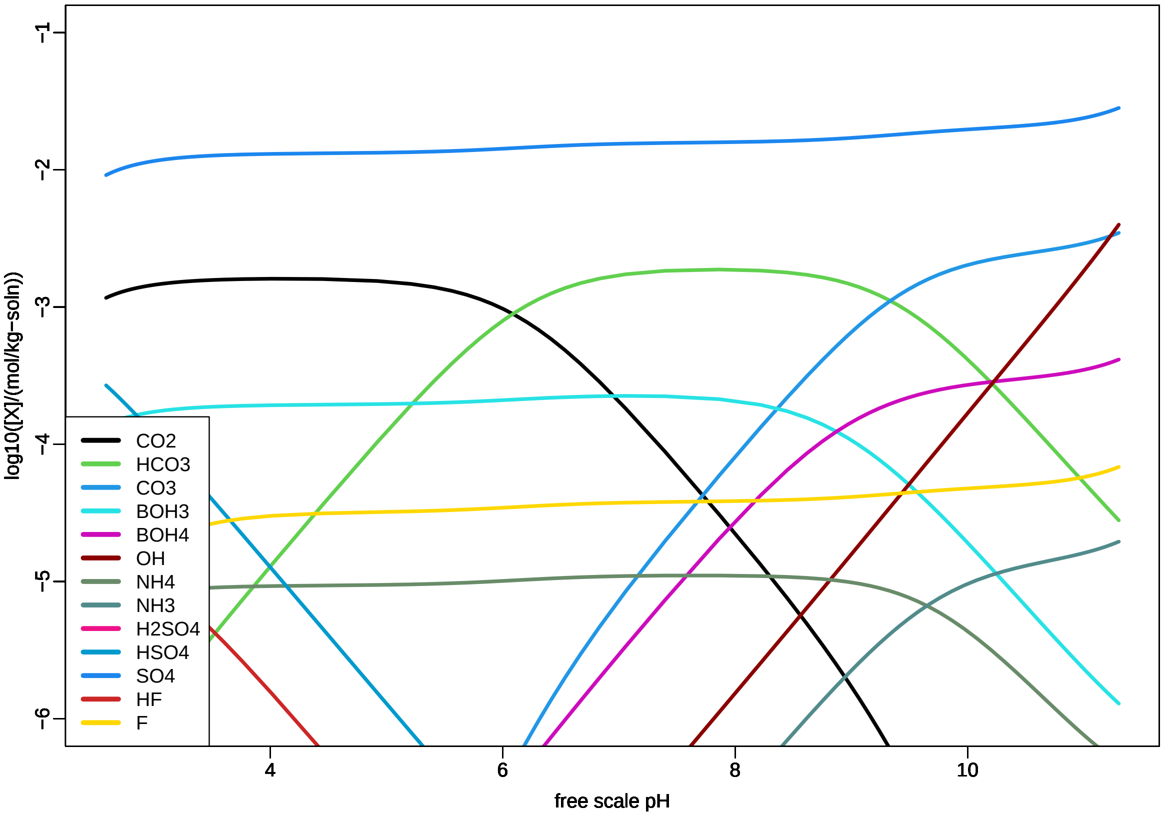
Generally Bjerrum plots are done on the log scale. This can be accomplished using the flag `log`

```
> what  <- c("CO2", "HCO3", "CO3", "BOH3", "BOH4", "OH",
+            "NH4", "NH3", "H2SO4", "HSO4", "SO4", "HF", "F")
> plot(ae, what = what, bjerrum = TRUE, log = TRUE)
```



Furthermore, we can zoom in to the region of most interest to marine scientists

```
> plot(ae, what = what, bjerrum = TRUE, log = TRUE, ylim = c(-6,-1),
+      legendinset = 0, lwd = 3,
+      palette = c(1, 3, 4, 5, 6, colors()[seq(100,250,6)]))
```



Titration with NaOH

Similar to the titration with HCl, also a titration with NaOH can be simulated (Please note that due to runtime constraints of the vignette for this package, this titration simulation is not run during construction of the vignette and, therefore, no resulting plots of the following code chunks is included. The user can generate the plots by extracting and executing the relevant code-chunks.)

```
> ae <- titration(aquaenv(S = 35, t = 15, SumCO2 = 0.0035, SumNH4 = 0.00002, pH = 2),
+        mass_sample = 0.01, mass_titrant = 0.02, conc_titrant = 0.01,
+        S_titrant = 0.5, steps = 50, type = "NaOH")
```

Plotting everything

```
> plot(ae, xval = ae$delta_mass_titrant, xlab = "NaOH solution added [kg]",
+      mfrow = c(10,10))
```

Plotting selectively

```
> what  <- c("TA", "pH", "CO2", "HCO3", "CO3", "BOH3", "BOH4", "OH",
+             "NH4", "NH3", "H2SO4", "HSO4", "SO4", "HF", "F", "fCO2")
> plot(ae, xval = ae$delta_mass_titrant, xlab = "NaOH solution added [kg]",
+      what = what, size = c(12,8), mfrow = c(4,4))
> plot(ae, xval = ae$pH, xlab = "free scale pH", what = what,
+      size = c(12,8), mfrow = c(4,4))
```

Bjerrum plots

```
> what  <- c("CO2", "HCO3", "CO3")
> plot(ae, what=what, bjerrum=TRUE)
```

```
> what  <- c("CO2", "HCO3", "CO3", "BOH3", "BOH4", "OH",
+            "NH4", "NH3", "H2SO4", "HSO4", "SO4", "HF", "F")
> plot(ae, what = what, bjerrum = TRUE, log = TRUE, ylim = c(-6,-1),
+      legendinset = 0, lwd = 3, palette = c(1,3,4,5,6,colors()[seq(100,250,6)]))
```

Titration with a titrant with high concentrations and a large sample volume - classical Bjerrum plots

The Bjerrum plots created in the previous two sections do not really look like the classical textbook ones. This is because we simulated a titration with a small sample volume and a titrant with low concentrations that is most representative to e.g. pore water titrations. As a result the total concentrations like, e.g., total carbonate decreased due to dilution. In simulating a titration with a rather large volume and a titrant with high concentrations the volume and salinity corrections do not matter any more and graphs known from textbooks (e.g. Zeebe and Wolf-Gladrow 2001) are produced.

```
> ae <- titration(aquaenv(S = 35, t = 15, SumCO2 = 0.0035, SumNH4 = 0.00002, pH = 11.3),
+       mass_sample = 100, mass_titrant = 0.5, conc_titrant = 3,
+       S_titrant = 0.5, steps = 100)
```

Plotting everything (not shown)

```
> plot(ae, xval = ae$delta_mass_titrant,
+    xlab = "HCl solution added [kg]", mfrow = c(10, 10))
```

Plotting selectively and with different elements for `xval` (not shown)

```
> what  <- c("TA", "pH", "CO2", "HCO3", "CO3", "BOH3", "BOH4", "OH",
+            "NH4", "NH3", "H2SO4", "HSO4", "SO4", "HF", "F", "fCO2")
> plot(ae, xval = ae$delta_mass_titrant, xlab = "HCl solution added [kg]",
+      what = what, size = c(12, 8), mfrow = c(4,4))
> plot(ae, xval = ae$pH, xlab = "free scale pH", what = what,
+      size = c(12,8), mfrow = c(4,4))
> plot(ae, xval = ae$delta_conc_titrant, xlab = "[HCl] offset added [mol/kg-soln]",
+      what = what, size = c(12,8), mfrow = c(4,4))
> plot(ae, xval = ae$delta_moles_titrant, xlab = "HCl added [mol]",
+      what = what, size = c(12,8), mfrow = c(4,4))
```

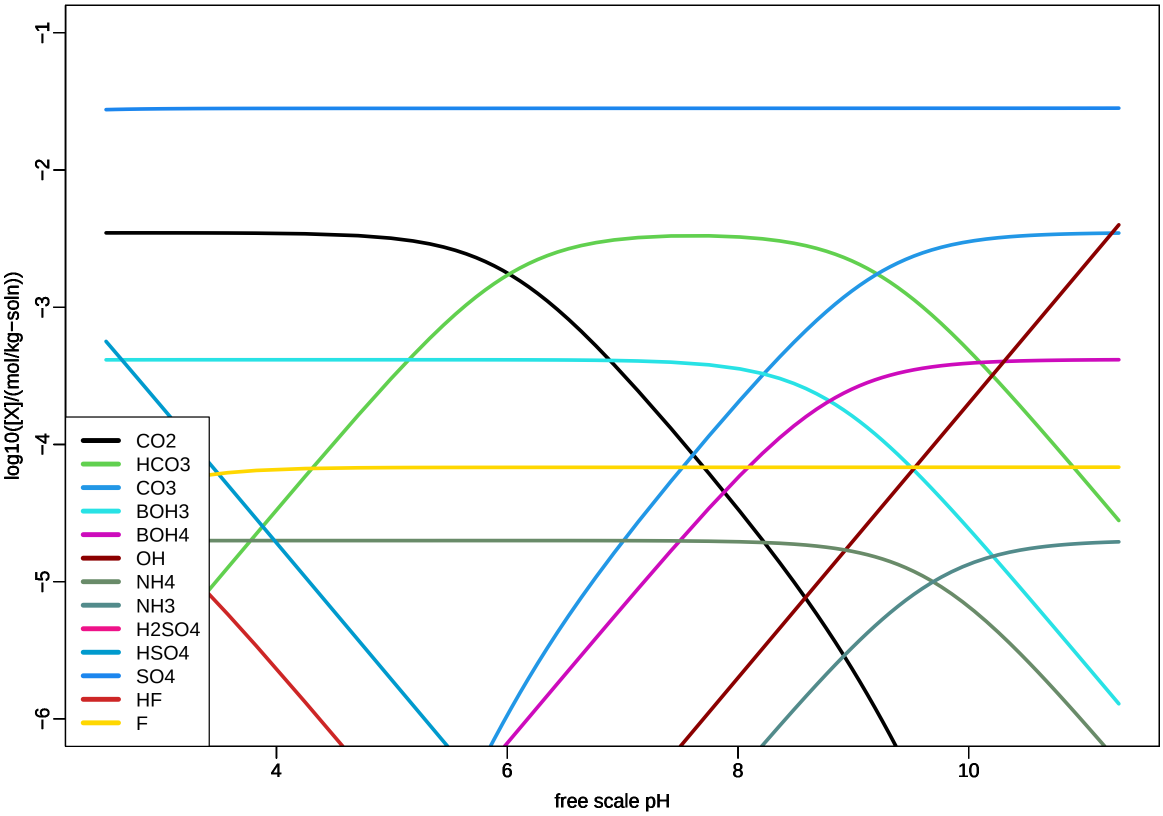
Creating different kinds of Bjerrum plots (not shown)

```
> plot(ae, bjerrum=TRUE)
> what  <- c("CO2", "HCO3", "CO3")
```

```
> plot(ae, what = what, bjerrum = TRUE)
> plot(ae, what = what, bjerrum = TRUE, lwd = 4,
+      palette=c("cyan", "magenta", "yellow"), bg = "gray",
+      legendinset = 0.1, legendposition = "topleft")
> what  <- c("CO2", "HCO3", "CO3", "BOH3", "BOH4", "OH",
+      "NH4", "NH3", "H2SO4", "HSO4", "SO4", "HF", "F")
> plot(ae, what = what, bjerrum = TRUE, log = TRUE)
```

and the classical textbook one

```
> what  <- c("CO2", "HCO3", "CO3", "BOH3", "BOH4", "OH",
+             "NH4", "NH3", "H2SO4", "HSO4", "SO4", "HF", "F")
> plot(ae, what = what, bjerrum = TRUE, log = TRUE, ylim = c(-6,-1),
+      legendinset = 0, lwd = 3, palette = c(1,3,4,5,6,colors()[seq(100,250,6)]))
```



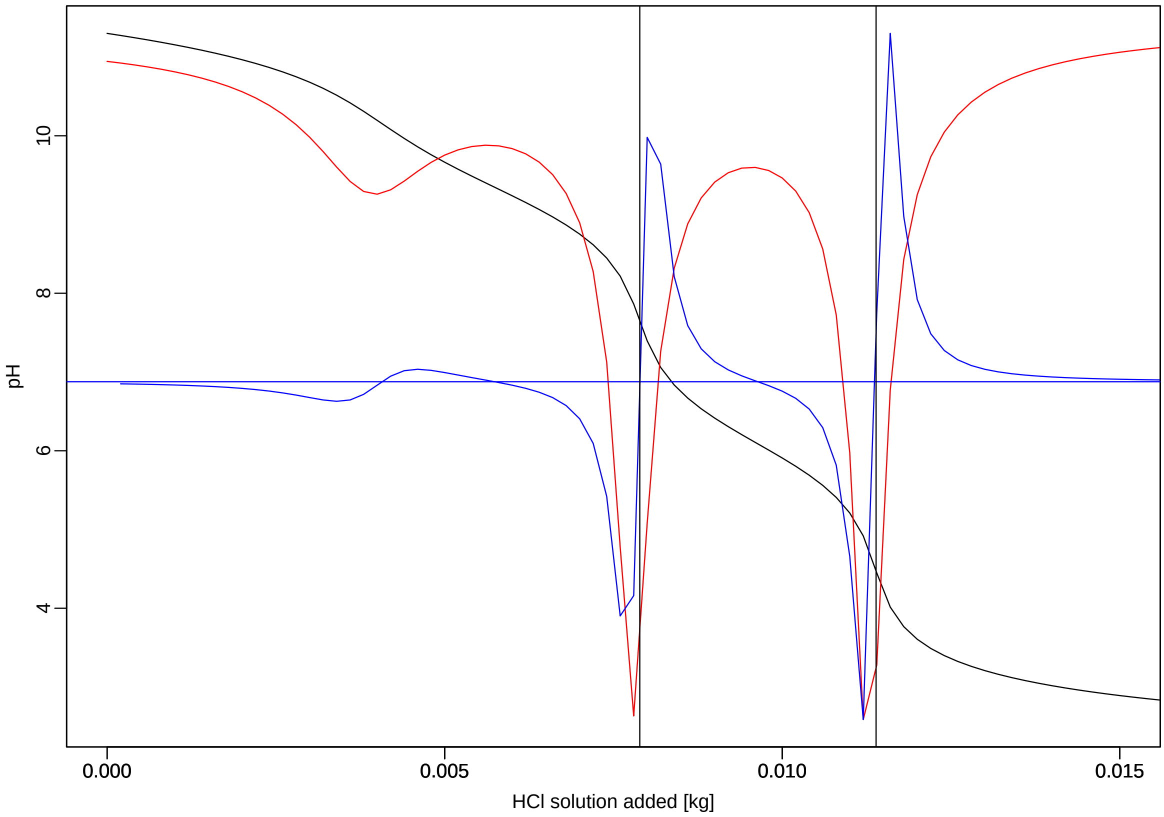
3.7. Calculating information from titration curves: the function `TAfit`

A little theory

While titrating a sample of natural seawater with HCl one sees two clear equivalence points (Dickson 1981). The second equivalence point is the equivalence point of total alkalinity and the difference beween the second and the first equivalence point signifies the total amount of $\sum CO_2$ of the sample Hansson and Jagner (1973).

This can be illustrated with **AquaEnv**. The respective titration curve can be plotted, together with its first and second derivative. Furthermore, the equivalence points can be marked with vertical lines (Please note that for a titrant concentration of 0.01 mol/kg-solution and 0.01 kg of sample, the value of the concentration (in mol/kg-solution) of total alkalinity and total carbonate equals the value of the total amount (in mol)).

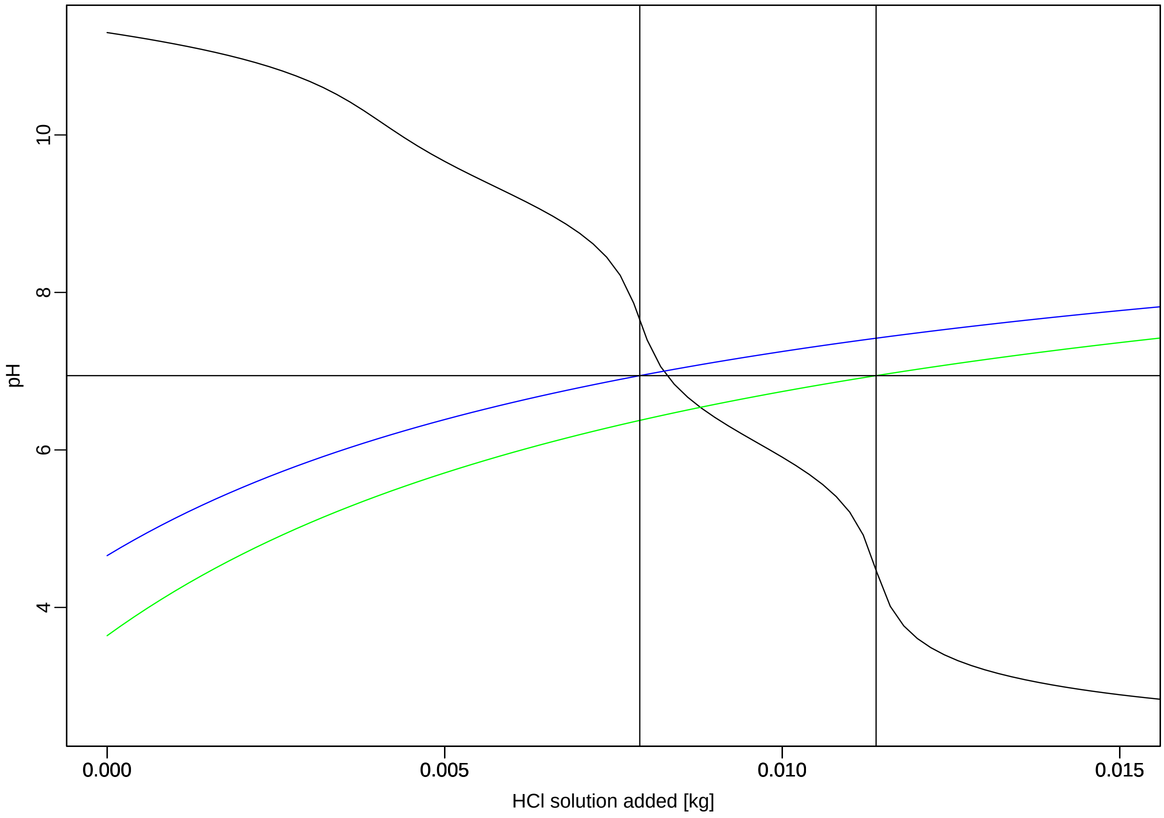
```
> ae_init <- aquaenv(S = 35, t = 15, SumCO2 = 0.0035, SumNH4 = 0.00002, pH = 11.3)
> ae <- titration(ae_init, mass_sample = 0.01, mass_titrant = 0.02,
+       conc_titrant = 0.01, S_titrant = 0.5, steps = 100)
> plot(ae, xval = ae$delta_mass_titrant, xlab = "HCl solution added [kg]",
+      what = "pH", xlim = c(0,0.015))
> par(new = TRUE)
> plot(ae$delta_mass_titrant[1:100], diff(ae$pH), type = "l",
+      col = "red", xlim = c(0,0.015), ylab = "", xlab = "", yaxt = "n")
> par(new = TRUE)
> plot(ae$delta_mass_titrant[2:100], diff(diff(ae$pH)), type = "l",
+      col = "blue", xlim = c(0,0.015), ylab = "", xlab = "", yaxt = "n")
> abline(h =0, col = "blue")
> abline(v = ae$TA[[1]])
> abline(v = ae$TA[[1]] - ae$SumCO2[[1]])
```



Following classical chemical textbooks (e.g. Skoog and West 1982), one can determine [TA] and [$\sum CO_2$] of a sample by graphically determining those equivalence points. However, there is no mechanistic understanding of the contents of the solution involved in doing so.

Other methods, called "Gran evaluations" (Gran 1952; Hansson and Jagner 1973; Dickson 1981; Haraldsson *et al.* 1997; Anderson *et al.* 1999), try to linearize the mechanistic model of what is going on in the solution during titration. They define the so called linear "Gran functions" and try to find their roots to determine the equivalence points. We will illustrate that by plotting the Gran functions F0 (blue) and F2 (-F1, green) and again mark the equivalence points with vertical lines. The y=zero line for the Gran functions is indicated by a horizontal line.

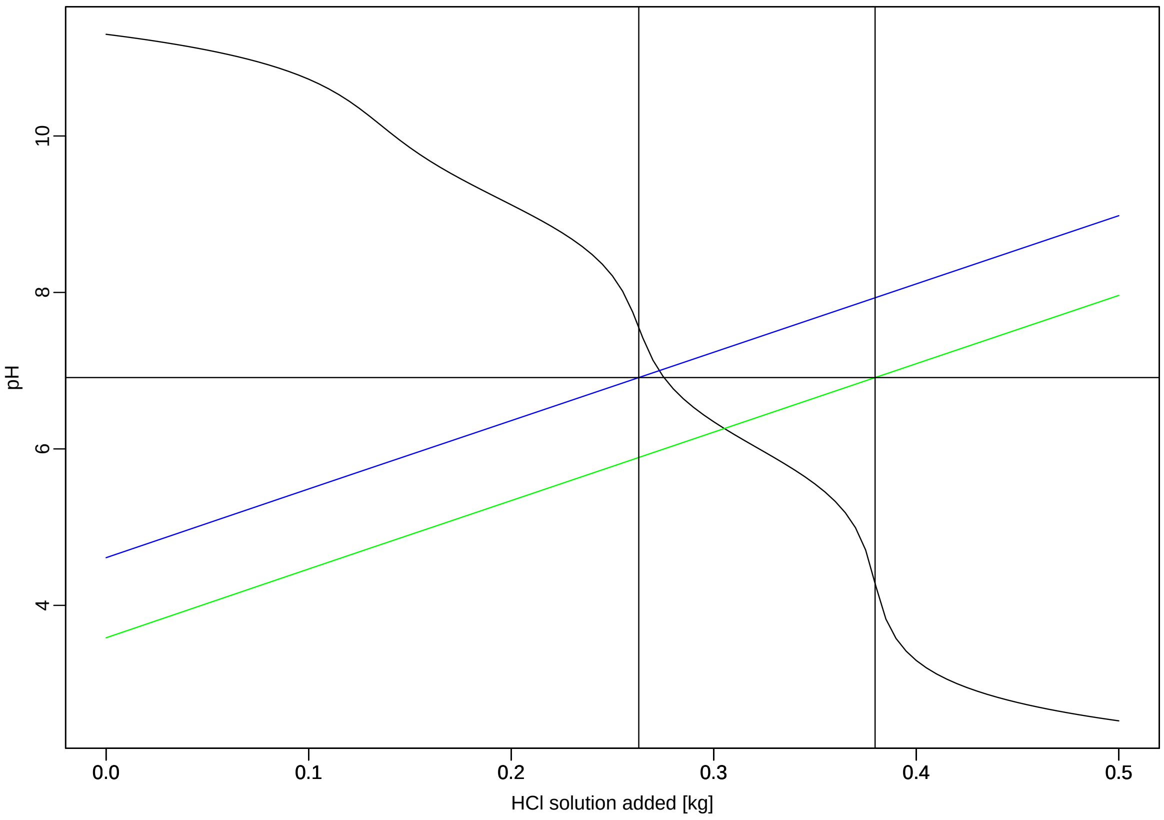
```
> plot(ae, xval = ae$delta_mass_titrant, xlab = "HCl solution added [kg]",
+      what = "pH", xlim = c(0,0.015))
> prot1 <- c()
> for (i in 1:length(ae$pH))
+     prot1 <- c(prot1, (10^-(ae$pH[[i]])+ae$HSO4[[i]]+ae$HF[[i]]+
+               ae$CO2[[i]]-ae$CO3[[i]]-ae$BOH4[[i]]-ae$OH[[i]]))
> par(new = TRUE)
> plot(ae$delta_mass_titrant, prot1, type = "l", col = "blue", xlim = c(0,0.015),
+      ylab = "", xlab = "", yaxt = "n", ylim = c(-0.015,0.015))
> prot2 <- c()
> for (i in 1:length(ae$pH))
+     prot2 <- c(prot2, (10^-(ae$pH[[i]])+ae$HSO4[[i]]+ae$HF[[i]]-
+               ae$HCO3[[i]]-2*ae$CO3[[i]]-ae$BOH4[[i]]-ae$OH[[i]]))
> par(new = TRUE)
> plot(ae$delta_mass_titrant, prot2, type = "l", col = "green", xlim = c(0,0.015),
+      ylab = "", xlab = "", yaxt = "n", ylim = c(-0.015,0.015))
> abline(v = ae$TA[[1]])
> abline(v = ae$TA[[1]] - ae$SumCO2[[1]])
> abline(h = 0)
```



One can see that the Gran functions are not linear. This is due to volume and salinity change effects during the titration. This can be overcome by either employing "modified Gran functions" (see Haraldsson *et al.* 1997) that correct for the volume changes or by using a titration with a titrant with high concentrations and a large sample volume (Please note that here the value of the concentration of total alkalinity and total carbonate does not equal their total amount and need to be converted with the factor 100/3)

```
> ae <- titration(aquaenv(S = 35, t = 15, SumCO2 = 0.0035, SumNH4 = 0.00002,
+                          pH = 11.3),
+                  mass_sample = 100, mass_titrant = 0.5, conc_titrant = 3,
+                  S_titrant = 0.5, steps = 100)
> plot(ae, xval = ae$delta_mass_titrant, xlab = "HCl solution added [kg]",
+      what = "pH", xlim = c(0, 0.5))
> prot1 <- c()
> for (i in 1:length(ae$pH))
+     prot1 <- c(prot1, (10^-(ae$pH[[i]])+ae$HSO4[[i]]+ae$HF[[i]]+
+                 ae$CO2[[i]]-ae$CO3[[i]]-ae$BOH4[[i]]-ae$OH[[i]]))
> par(new = TRUE)
> plot(ae$delta_mass_titrant, prot1, type = "l", col = "blue",
+      xlim = c(0,0.5),  ylab = "", xlab = "", yaxt = "n",
+      ylim = c(-0.015, 0.015))
> prot2 <- c()
> for (i in 1:length(ae$pH))
+     prot2 <- c(prot2, (10^-(ae$pH[[i]])+ae$HSO4[[i]]+ae$HF[[i]]-
```

```
+                 ae$HCO3[[i]]-2*ae$CO3[[i]]-ae$BOH4[[i]]-ae$OH[[i]]))
> par(new = TRUE)
> plot(ae$delta_mass_titrant, prot2, type = "l", col = "green",
+      xlim = c(0,0.5), ylab = "", xlab = "", yaxt = "n",
+      ylim = c(-0.015,0.015))
> abline(v = (ae$TA[[1]]*100/3))
> abline(v = ((ae$TA[[1]] - ae$SumCO2[[1]])*100/3))
> abline(h = 0)
```



Another proposed method of determining [TA] and $[\sum \mathrm{CO_2}]$ is to not only determine the two equivalence points, but to fit the whole titration curve with a theoretical titration curve based on a mechanistic model of what is going on in the solution during the titration (Dickson 1981; DOE 1994; Anderson *et al.* 1999). The function `titration` of **AquaEnv** provides exactly such a theoretical titration curve and the function `TAfit` makes use of this fact to determine [TA] and $[\sum \mathrm{CO_2}]$ of a sample by non linear curve fitting.

Determining TA and $[\sum \mathrm{CO_2}]$ *by non linear curve fitting*

3.7.2.1 Proof of concept

First, a proof of concept will show that the function `TAfit` is implemented consistently. Some "data" can be generated with the `titration` function.

```
> initial_ae <- aquaenv(S = 35, t = 15, SumCO2 = 0.002, TA = 0.0022)
> ae <- titration(initial_ae, mass_sample = 0.01, mass_titrant = 0.003,
```

```
+                    conc_titrant = 0.01, S_titrant = 0.5, steps = 20)
```

Now, the input data for the `TAfit` routine can be generated: a table with the added mass of the titrant and the resulting free scale pH:

```
> titcurve <- cbind(ae$delta_mass_titrant, ae$pH)
```

Note that For the `TAfit` all total quantities except $\sum CO_2$ ($\sum NH_4^+$, $\sum H_2S$, $\sum H_3PO4$, $\sum Si(OH)_4$, $\sum HNO3$, $\sum HNO2$, $\sum B(OH)_3$, $\sum H_2SO_4$, $\sum HF$) need to be known. However, the latter three can be calculated from salinity as it is done in this example.

```
> fit1 <- TAfit(initial_ae, titcurve, conc_titrant = 0.01,
+         mass_sample = 0.01, S_titrant = 0.5)
> fit1

$TA
[1] 0.002197372
attr(,"unit")
[1] "mol/kg-soln"

$SumCO2
[1] 0.001996944
attr(,"unit")
[1] "mol/kg-soln"

$sumofsquares
[1] 0.001195296
```

Thus, we see that `TAfit` calculates the correct $\sum CO_2$ and TA values.

Trying the Roy *et al.* (1993b) (K_CO2 and K_HCO3) and Perez and Fraga (1987b) (K_HF) values. (Please note that due to runtime constraints of the vignette for this package, this calculation is not run during construction of the vignette. The user can perform the calculation by extracting and executing the relevant code-chunks.)

```
> initial_ae_ <- aquaenv(S = 35, t = 15, SumCO2 = 0.002, TA = 0.0022,
+                        k1k2 = "lueker", khf = "perez")
> ae_         <- titration(initial_ae_, mass_sample = 0.01, mass_titrant = 0.003,
+                          conc_titrant = 0.01,
+                          S_titrant = 0.5, steps = 20, k1k2 = "roy",
+                          khf = "perez")
> titcurve_   <- cbind(ae_$delta_mass_titrant, ae_$pH)
> fit1_       <- TAfit(initial_ae_, titcurve_, conc_titrant = 0.01,
+                      mass_sample = 0.01, S_titrant = 0.5, k1k2 = "roy",
+                      khf = "perez", verbose = TRUE)
> fit1_
```

`TAfit` can also take E (V) values as input variables, so we generate E values using E0 = 0.4 V and the Nernst equation. (But before that, we calculate a titration with a titrant with the same salinity as seawater such that S does not change during the titration. Otherwise we would need to calculate the S profile for the titration extra to use it to convert to the total scale in the following step.) However, to do so we first need to convert our pH curve to the seawater pH scale. According to DOE (p.7, ch.4, sop.3 1994) and Dickson *et al.* (2007), the Nernst equation relates E to the total proton concentration.

```
> ae <- titration(initial_ae, mass_sample = 0.01, mass_titrant = 0.003,
+                   conc_titrant = 0.01, steps = 20, seawater_titrant = TRUE)
> titcurve    <- cbind(ae$delta_mass_titrant, ae$pH)
> tottitcurve <- cbind(ae$pH, convert(ae$pH, "pHscale", "free2tot", S = 35, t = 15))
> Etitcurve   <- cbind(ae$delta_mass_titrant,
+   (0.4 - ((PhysChemConst$R/10)*initial_ae$T/PhysChemConst$F)*
+      log(10^-tottitcurve[,2])))
```

Again, `TAfit` can be executed, this time also calculating E0. Note that the flag `verbose = TRUE` causes `TAfit` to show the progress of the fitting procedure in a plot window. (Please note that due to runtime constraints of the vignette for this package, this calculation is not run during construction of the vignette. The user can perform the calculation by extracting and executing the relevant code-chunks.)

```
> fit2 <- TAfit(initial_ae, Etitcurve, conc_titrant = 0.01, mass_sample = 0.01,
+   Evals = TRUE, verbose = TRUE, seawater_titrant = TRUE)
> fit2
```

Furthermore, `TAfit` can fit K_CO2 as well, however, one single value for the whole titration curve is fitted, i.e. there is no correction for K_CO2 changes due to changing S due to mixing with the titrant

```
> fit3 <- TAfit(initial_ae, titcurve, conc_titrant = 0.01, mass_sample = 0.01,
+      S_titrant = 0.5, K_CO2fit = TRUE)
> fit3
```

```
$TA
[1] 0.002201834
attr(,"unit")
[1] "mol/kg-soln"

$SumCO2
[1] 0.001999162
attr(,"unit")
[1] "mol/kg-soln"

$K_CO2
[1] 9.436631e-07
attr(,"unit")
```

```
[1] "mol/kg-soln"
attr(,"pH scale")
[1] "free"

$sumofsquares
[1] 0.0005029526
```

```
> initial_ae$K_CO2
```

```
[1] 9.630811e-07
attr(,"unit")
[1] "mol/kg-soln"
attr(,"pH scale")
[1] "free"
```

One can see that the fitted value for K_CO2 is not the same as the value in the initial `aquaenv` object, which is the "correct" value. That is, because during data creation K_CO2 changed along the course of the titration due to changes in salinity. Assuming that the titrant has the same salinity as the sample (and is made up of natural seawater, i.e. containing SumBOH4, SumH2SO4 and SumHF as functions of S), then the "correct" K_CO2 should be fitted. This can be accomplished in `TAfit` by not giving the argument `S_titrant` (i.e. assuming the titrant has the same salinity as the sample) and setting the flag `seawater_titrant` to `TRUE`

```
> ae       <- titration(initial_ae, mass_sample = 0.01, mass_titrant = 0.003,
+                     conc_titrant = 0.01, steps = 20, seawater_titrant = TRUE)
> titcurve <- cbind(ae$delta_mass_titrant, ae$pH)
> fit4 <- TAfit(initial_ae, titcurve, conc_titrant = 0.01, mass_sample = 0.01,
+              K_CO2fit = TRUE, seawater_titrant = TRUE)
> fit4
```

```
$TA
[1] 0.002200386
attr(,"unit")
[1] "mol/kg-soln"

$SumCO2
[1] 0.00199587
attr(,"unit")
[1] "mol/kg-soln"

$K_CO2
[1] 9.69964e-07
attr(,"unit")
[1] "mol/kg-soln"
attr(,"pH scale")
[1] "free"
```

```
$sumofsquares
[1] 0.0001623836
```

Furthermore, TA, SumCO2, K_CO2 and E0 can be fitted at the same time. (Please note that due to runtime constraints of the vignette for this package, this calculation is not run during construction of the vignette. The user can perform the calculation by extracting and executing the relevant code-chunks.)

```
> Etitcurve <- cbind(titcurve[,1], (0.4 - ((PhysChemConst$R/10)*initial_ae$T/
+            PhysChemConst$F)*log(10^-titcurve[,2])))
> fit5 <- TAfit(initial_ae, Etitcurve, conc_titrant = 0.01, mass_sample = 0.01,
+                K_CO2fit = TRUE, seawater_titrant = TRUE, Evals = TRUE)
> fit5
```

Sometimes, the obtained titration curve is not equally spaced on the x axis. `TAfit` can deal with such curves if the flag `equalspaced` is set to `FALSE`. (Please note that due to runtime constraints of the vignette for this package, this calculation is not run during construction of the vignette. The user can perform the calculation by extracting and executing the relevant code-chunks.)

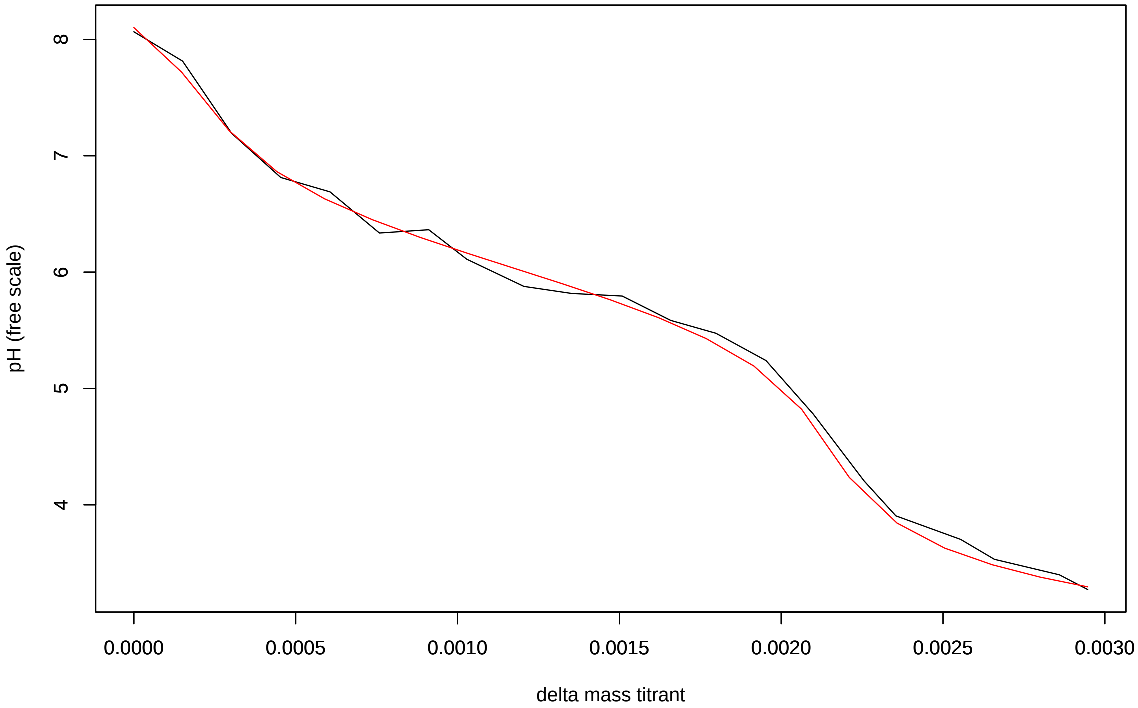
```
> neqsptitcurve <- rbind(titcurve[1:9,], titcurve[11:20,])
> fit6 <- TAfit(initial_ae, neqsptitcurve, conc_titrant = 0.01,
+                mass_sample = 0.01, seawater_titrant = TRUE, equalspaced = FALSE,
+                verbose = TRUE, debug = TRUE)
> fit6
```

Finally, some "noise" is added to the generated data

```
> noisetitcurve <- titcurve * rnorm(length(titcurve), mean = 1, sd = 0.01) #one percent er
> fit7 <- TAfit(initial_ae, noisetitcurve, conc_titrant = 0.01, mass_sample = 0.01,
+                seawater_titrant = TRUE, verbose = FALSE, debug = TRUE)
> fit7


$TA
[1] 0.002155043
attr(,"unit")
[1] "mol/kg-soln"

$SumCO2
[1] 0.001946753
attr(,"unit")
[1] "mol/kg-soln"

$sumofsquares
[1] 0.08767393
```

The flag `verbose=TRUE` (default is `FALSE`) prompts to show the traject of the fitting procedure in a plot window. However, each new fit is plotted over the first one and Sweave includes only the first plot in each code chunk in the resulting LaTeXfile. Therefore, we use the flag `debug=TRUE` to visualize the final fit

```
> ylim <- range(noisetitcurve[,2], calc)
> xlim <- range(tit$delta_mass_titrant, noisetitcurve[,1])
> plot(noisetitcurve[,1], noisetitcurve[,2], xlim = xlim, ylim = ylim,
+      type = "l", xlab = "delta mass titrant", ylab = "pH (free scale)")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim, type = "l",
+      col = "red", xlab = "", ylab = "")
```



3.7.2.2 Test with generated data from literature

Dickson (1981) provided a synthetic dataset to test total alkalinity fitting programs. This dataset is included in **AquaEnv** as `sample_dickson1981`. Following quantities are given

```
> conc_titrant <- 0.3     # mol/kg-soln
> mass_sample  <- 0.2     # kg
> S_titrant    <- 14.835  # is aequivalent to the ionic strength of 0.3 mol/kg-soln
> SumBOH3  <- 0.00042 # mol/kg-soln
```

```
> SumH2SO4 <- 0.02824 # mol/kg-soln
> SumHF    <- 0.00007 # mol/kg-soln
```

Note that all concentrations are in mol/kg-solution and the mass of the sample is in kg. Note further that the salinity of the titrant has been calculated from its ionic strenght of 0.3 mol/kg-soln.

In the original dataset as represented in `sample_dickson1981`, the mass of titrant is given in g which needs to be converted to kg

```
> sam <- cbind(sample_dickson1981[,1]/1000, sample_dickson1981[,2]) # convert mass of titr
```

Then an attempt to recalculate the [TA] and [$\sum CO_2$] values given in Dickson (1981) ([TA]=0.00245 mol/kg-soln and [$\sum CO_2$] 0.00220 mol/kg-soln) can be done

```
> dicksonfit <- TAfit(aquaenv(S = 35, t = 25, SumBOH3 = SumBOH3,
+              SumH2SO4 = SumH2SO4, SumHF=SumHF), sam, conc_titrant,
+              mass_sample, S_titrant = S_titrant, debug = TRUE)
> dicksonfit

$TA
[1] 0.002464256
attr(,"unit")
[1] "mol/kg-soln"

$SumCO2
[1] 0.002191958
attr(,"unit")
[1] "mol/kg-soln"

$sumofsquares
[1] 0.01287255
```

This shows the fit is not accurate. Why is that so?

3.7.2.2.1 Does the salinity correction (`S_titrant`) matter?

Let us calculate a theoretical titration without salinity correction

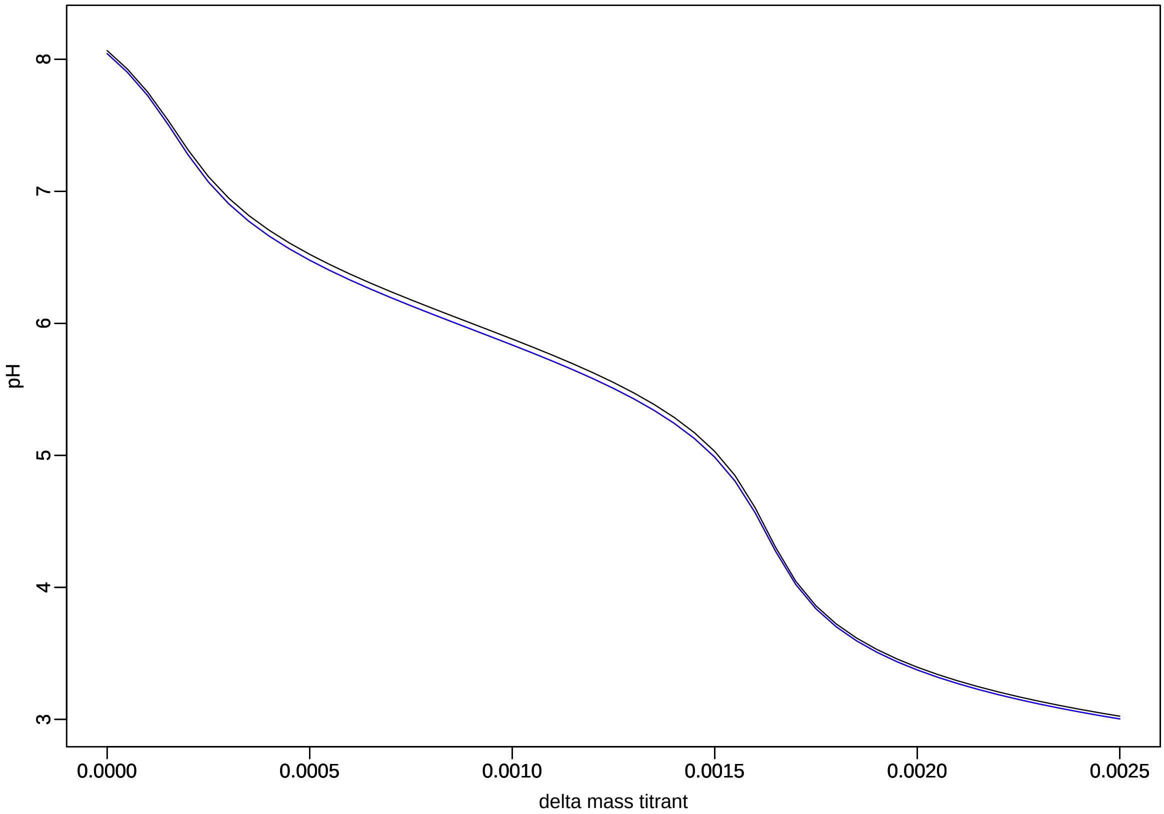
```
> dicksontitration1 <- titration(aquaenv(S = 35, t = 25, SumCO2 = 0.00220,
+    SumBOH3 = SumBOH3, SumH2SO4 = SumH2SO4, SumHF = SumHF, TA = 0.00245),
+    mass_sample = mass_sample, mass_titrant = 0.0025,
+    conc_titrant = conc_titrant, steps = 50, type = "HCl")
```

and one with salinity correction

```
> dicksontitration2 <- titration(aquaenv(S = 35, t = 25, SumCO2 = 0.00220,
+     SumBOH3 = SumBOH3, SumH2SO4 = SumH2SO4, SumHF = SumHF, TA = 0.00245),
+     mass_sample = mass_sample, mass_titrant = 0.0025,
+     conc_titrant = conc_titrant, S_titrant = S_titrant, steps = 50, type = "HCl")
```

Now the difference between both curves (in red and blue) and the “Dickson” curve (in black) can be visualized

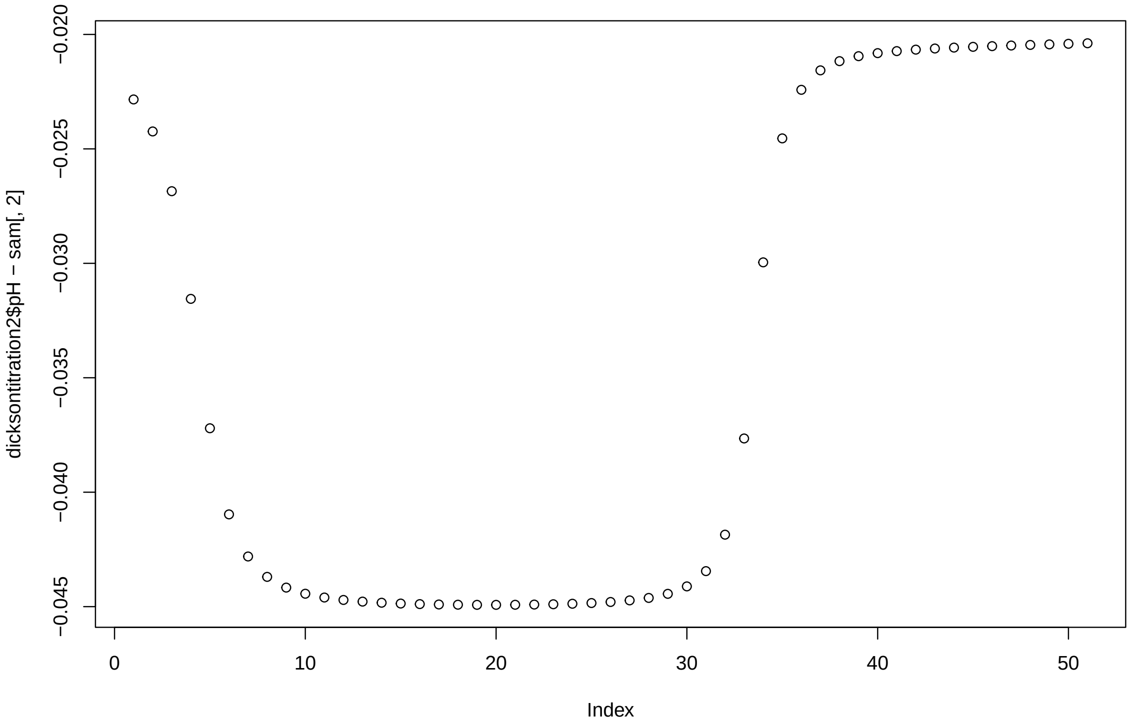
```
> plot(dicksontitration1, xval = dicksontitration1$delta_mass_titrant,
+      what = "pH", xlim = c(0,0.0025), ylim = c(3,8.2), col = "red",
+      xlab = "delta mass titrant")
> par(new = TRUE)
> plot(dicksontitration2, xval = dicksontitration2$delta_mass_titrant,
+      what = "pH", xlim = c(0,0.0025), ylim = c(3,8.2), col = "blue", xlab = "")
> par(new = TRUE)
> plot(sam[,1], sam[,2], type = "l", xlim = c(0,0.0025),
+      ylim = c(3,8.2), xlab = "", ylab = "")
```



That means, the salinity correction makes no significant difference (the red and the blue curve cannot be discerned), because the relation between the total amount of sample and the added amount of titrant is very large: salinity only drops from 35 to 34.75105.

But there is an offset between the "Dickson" curve and our curve

```
> plot(dicksontitration2$pH - sam[,2])
```



3.7.2.2.2 Does fitting `K_CO2` as well improve the fit?

```
> dicksonfit2 <- TAfit(aquaenv(S = 35, t = 25, SumBOH3 = SumBOH3,
+        SumH2SO4 = SumH2SO4, SumHF = SumHF), sam, conc_titrant,
+        mass_sample, S_titrant = S_titrant, debug = TRUE, K_CO2fit = TRUE)
> dicksonfit2

$TA
[1] 0.002458081
attr(,"unit")
[1] "mol/kg-soln"

$SumCO2
[1] 0.002194006
attr(,"unit")
[1] "mol/kg-soln"

$K_CO2
[1] 1.03096e-06
attr(,"unit")
[1] "mol/kg-soln"
attr(,"pH scale")
```

```
[1] "free"

$sumofsquares
[1] 0.005724457
```

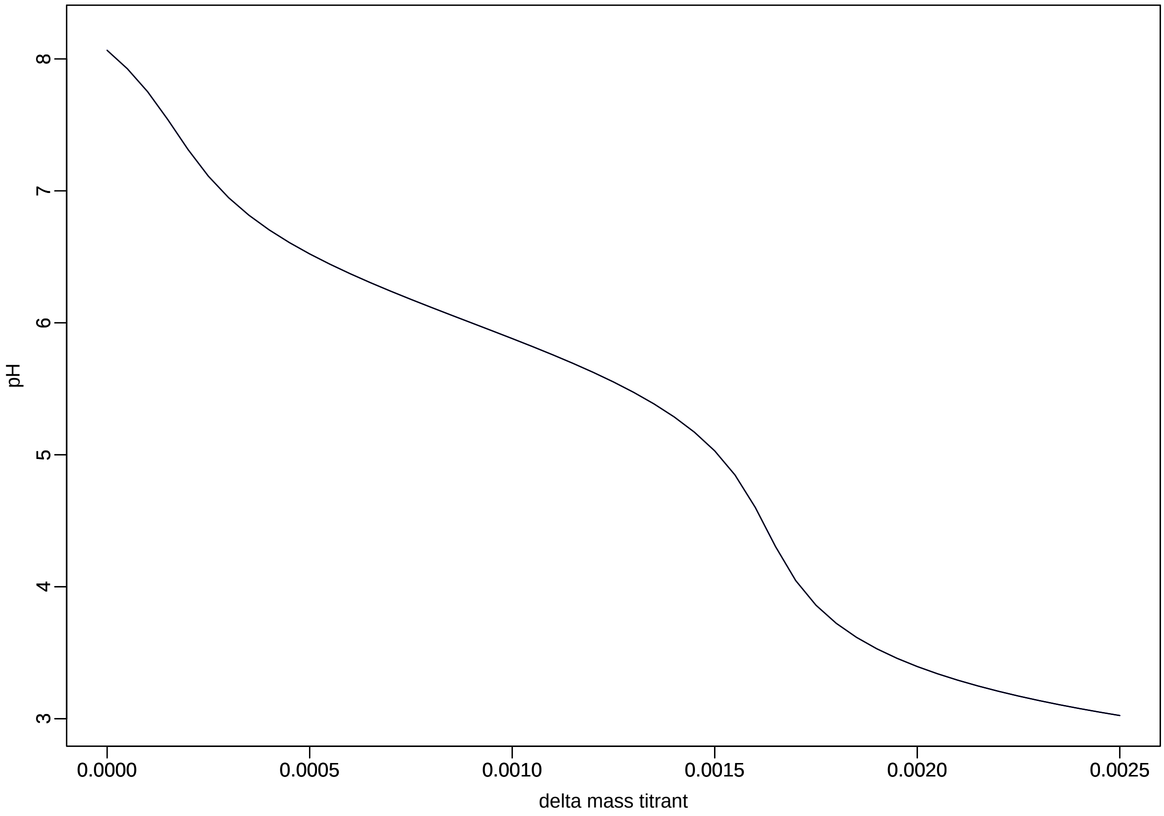
Yes it does, but it is not optimal yet.

```
>
```

There still remains one major difference between the calculations carried out in Dickson (1981) and the calculations in **AquaEnv**: Dickson (1981) uses fixed values for the equilibrium constants and does not calculate them as functions of temperature and salinity. Furthermore, the values that are used in Dickson (1981) are not exactly the same as are obtained in **AquaEnv** for the same salinity and temperature.

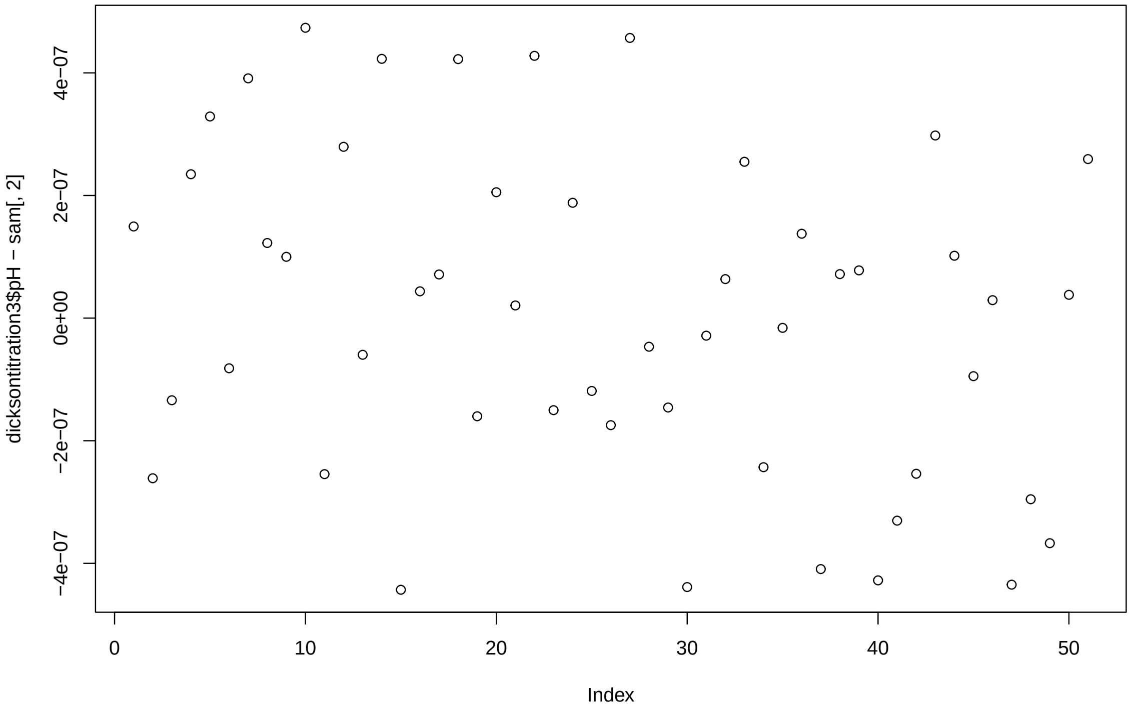
Let us calculate a theoretical titration curve employing exactly the same equilibrium constant values as used in Dickson (1981) and plot the result together with the "Dickson" curve

```
> dicksontitration3 <- titration(aquaenv(S = 35, t = 25, SumCO2 = 0.00220,
+    SumBOH3 = SumBOH3, SumH2SO4 = SumH2SO4, SumHF = SumHF, TA = 0.00245,
+    k_w = 4.32e-14, k_co2 = 1e-6, k_hco3 = 8.20e-10, k_boh3 = 1.78e-9,
+    k_hso4 = (1/1.23e1), k_hf = (1/4.08e2)),
+    mass_sample = mass_sample, mass_titrant = 0.0025, conc_titrant = conc_titrant,
+    steps = 50, type = "HCl", S_titrant = S_titrant,
+    k_w = 4.32e-14, k_co2 = 1e-6, k_hco3 = 8.20e-10, k_boh3 = 1.78e-9,
+    k_hso4 = (1/1.23e1), k_hf = (1/4.08e2))
> plot(dicksontitration3, xval = dicksontitration3$delta_mass_titrant,
+   what = "pH", xlim = c(0,0.0025), ylim = c(3,8.2), col = "blue",
+   xlab = "delta mass titrant")
> par(new = TRUE)
> plot(sam[,1], sam[,2], type = "l", xlim = c(0,0.0025), ylim = c(3,8.2),
+   xlab = "", ylab = "")
```



Plotting the differences between both curves reveals that they are the same down to 1 umol/kg-soln.

```
> plot(dicksontitration3$pH - sam[,2])
```



Calculating [TA] and [$\sum CO_2$] using `TAfit` and exactly the same equilibrium constant values as used in Dickson (1981)

```
> dicksonfit3 <- TAfit(aquaenv(S = 35, t = 25, SumBOH3 = SumBOH3, SumH2SO4 = SumH2SO4,
+        SumHF = SumHF, k_w = 4.32e-14, k_co2 = 1e-6, k_hco3 = 8.20e-10,
+        k_boh3 = 1.78e-9, k_hso4 = (1/1.23e1), k_hf = (1/4.08e2)),
+        sam, conc_titrant, mass_sample, S_titrant = S_titrant, debug = TRUE,
+        k_w = 4.32e-14, k_co2 = 1e-6, k_hco3 = 8.20e-10, k_boh3 = 1.78e-9,
+        k_hso4 = (1/1.23e1), k_hf = (1/4.08e2))
> dicksonfit3

$TA
[1] 0.00245
attr(,"unit")
[1] "mol/kg-soln"

$SumCO2
[1] 0.0022
attr(,"unit")
[1] "mol/kg-soln"

$sumofsquares
[1] 3.279302e-12
```

reveals that now exactly the same values are calculated as are given in Dickson (1981).

4. Extending **AquaEnv**

It is simple for the user to create own functions that use **AquaEnv** and extend its functionality. We will demonstrate that by creating simple analogons for the **AquaEnv** functions `titration` and `TAfit`.

The function `simpletitration` will take the following arguments

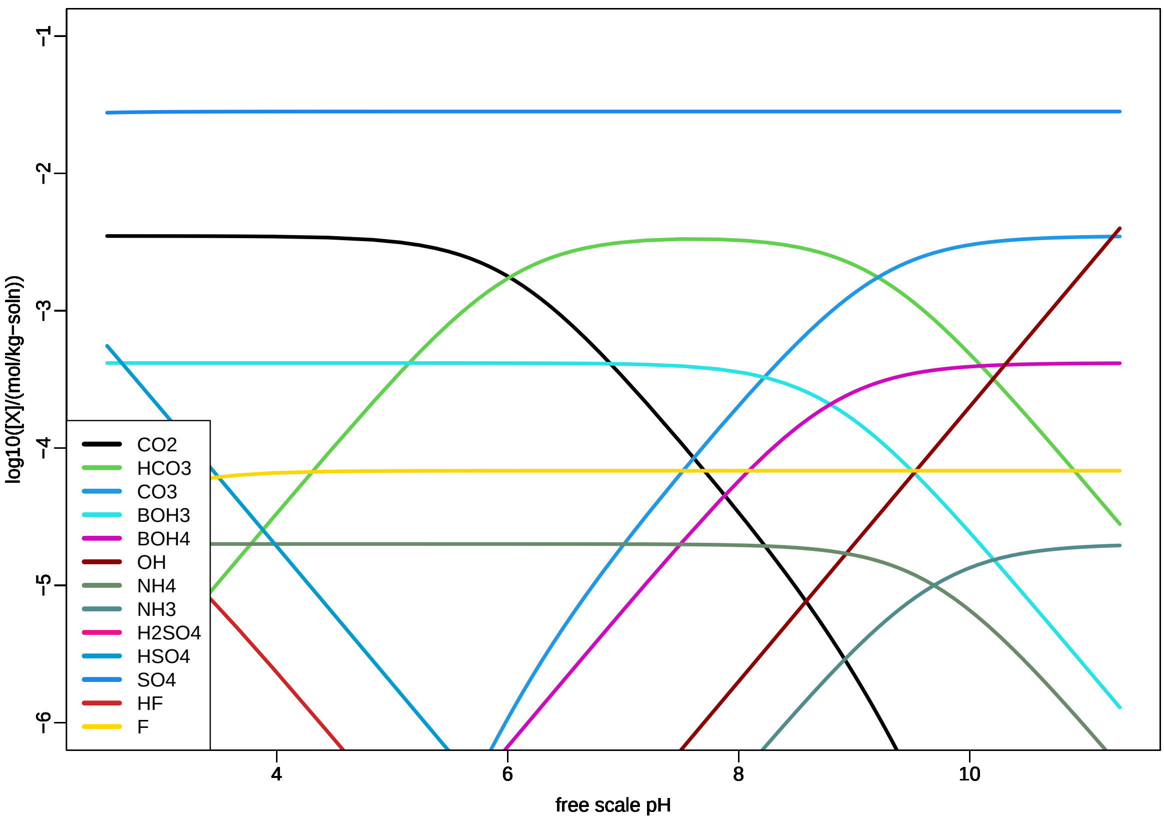
`aquaenv`	an object of class `aquaenv`: minimal definition, contains all information about the system: S, t, p, total concentrations of nutrients etc
`volume`	the volume of the (theoretical) titration vessel in l
`amount`	the amount of titrant added in mol
`steps`	the amount of steps the amount of titrant is added in
`type`	the type of titrant: either "HCl" or "NaOH"

The function is defined as

```
> simpletitration <- function(aquaenv,                # an object of class aquaenv: minimal definition,
+                                                     # contains all information about the system:
+                                                     # T, S, d, total concentrations of nutrients etc
+                             volume,                 # the volume of the (theoretical) titration vessel in l
+                             amount,                 # the amount of titrant added in mol
+                             steps,                  # the amount of steps the amount of titrant is added in
+                             type)                   # the type of titrant: either "HCl" or "NaOH"
+   {
+     directionTAchange   <- switch(type, HCl  = -1, NaOH = +1)
+     TAconcchangeperstep <- convert(((amount/steps)/volume), "conc", "molar2molin", aquaenv$t, aquaenv$S)
+
+     aquaenvtemp <- aquaenv
+
+     for (i in 1:steps)
+       {
+         TA          <- aquaenvtemp$TA + (directionTAchange * TAconcchangeperstep)
+         aquaenvtemp <- aquaenv(ae=aquaenvtemp, TA=TA)
+         aquaenv     <- merge(aquaenv, aquaenvtemp)
+       }
+
+     aquaenv[["DeltaCTitrant"]] <- convert((amount/volume)/steps*(1:(steps+1)),
+                                           "conc", "molar2molin", aquaenv$t, aquaenv$S)
+     return(aquaenv)  # object of class aquaenv which contains a titration simulation
+   }
```

and can be used to create a bjerrum plot

```
> ae <- simpletitration(aquaenv(S = 35, t = 15, SumCO2 = 0.003500,
+                               SumNH4 = 0.000020, pH = 11.3),
+            volume =100, amount = 1.5, steps = 100, type = "HCl")
> what  <- c("CO2", "HCO3", "CO3", "BOH3", "BOH4", "OH", "NH4", "NH3",
+             "H2SO4", "HSO4", "SO4", "HF", "F")
> plot(ae, what = what, bjerrum = TRUE, log = TRUE, ylim = c(-6,-1),
+      legendinset = 0, lwd = 3, palette = c(1,3,4,5,6,colors()[seq(100,250,6)]))
```



The function **`simpletitration`** in turn can be used to create a simple analogon to **`TAfit`** with the arguments

`ae`	an object of class `aquaenv`: minimal definition, contains all information about the system: S, t, p, total concentrations of nutrients etc
`pHmeasurements`	a table containing the titration curve: basically a series of pH values (pH on free proton scale)
`volume`	the volume of the titration vessel
`amount`	the total amount of the titrant added
`TAguess`=0.0025	a first guess for [TA] and [SumCO2] to be used as initial values for the optimization procedure
`type`="HCl"	the type of titrant: either "HCl" or "NaOH"

defined as

```
> simpleTAfit <- function(ae,                    # an object of class aquaenv: minimal definition,
+                                                # contains all information about the system:
+                                                # T, S, d, total concentrations of nutrients etc
+                         pHmeasurements,        # a table containing the titration curve:
+                                                # basically a series of pH values (pH on free proton scale)
+                         volume,                # the volume of the titration vessel
+                         amount,                # the total amount of the titrant added
+                         TAguess=0.0025,        # a first guess for [TA] and [SumCO2] to be used as
+                                                # initial values for the optimization procedure
+                         type="HCl")            # the type of titrant: either "HCl" or "NaOH"
+   {
```

```
+     ae$Na <- NULL   # make sure ae gets cloned as "skeleton": cloneaquaenv determines "skeleton"
+                     # TRUE or FALSE from the presence of a value for Na
+     residuals <- function(state)
+       {
+         ae$SumCO2  <- state[[1]]
+         pHcalc     <- simpletitration(aquaenv(ae=ae, TA=state[[2]]), volume=volume,
+                                       amount=amount, steps=(length(pHmeasurements)-1), type=type)$pH
+         residuals <- pHmeasurements-pHcalc
+
+         return(residuals)
+       }
+
+     require(minpack.lm)
+     out <- nls.lm(fn=residuals, par=c(TAguess, TAguess))  #guess for TA is also used as guess for SumCO2
+
+     result                     <- list(out$par[[2]], out$par[[1]], out$deviance)
+     attr(result[[1]], "unit") <- "mol/kg-soln"
+     attr(result[[2]], "unit") <- "mol/kg-soln"
+     names(result)              <- c("TA", "SumCO2", "sumofsquares")
+     return(result)  # a list of three values
+                     # ([TA] in mol/kg-solution, [SumCO2] in mol/kg-solution, sum of the squared residuals)
+   }
```

The function `simpleTAfit` can be used to calculate TA and SumCO2

```
> pHmeasurements <- ae$pH
> fit <- simpleTAfit(aquaenv(S = 35, t = 15, SumNH4 = 0.00002),
+         pHmeasurements, volume = 100, amount = 1.5)
> fit
```

```
$TA
[1] 0.01139004
attr(,"unit")
[1] "mol/kg-soln"

$SumCO2
[1] 0.0035
attr(,"unit")
[1] "mol/kg-soln"

$sumofsquares
[1] 1.680385e-20
```

A. Abbreviations for references used throughout the code and in the helpfiles

Atkins1996	Atkins (1996)
Boudreau1996	Boudreau (1996)
DOE1994	DOE (1994)
Dickson1979a	Dickson and Riley (1979a)
Dickson1981	Dickson (1981)
Dickson1984	Dickson (1984)
Dickson1987	Dickson and Millero (1987)
Dickson1990	Dickson (1990a)
Dickson2007	Dickson *et al.* (2007)
Emerson2008	Emerson and Hedges (2008)
Feistel2008	Feistel (2008)
Fofonoff1983	Fofonoff and Millard (1983)
Follows2006	Follows *et al.* (2006)
Hofmann2008	Hofmann *et al.* (2008)
Khoo1977	Khoo *et al.* (1977)
Lewis1998	Lewis and Wallace (1998)
Lueker2000	Lueker *et al.* (2000)
Millero1981	Millero and Poisson (1981)
Millero1988	Millero *et al.* (1988)
Millero1995	Millero (1995)
Millero1995a	Millero *et al.* (1995)
Millero2006	Millero *et al.* (2006)
Mucci1983	Mucci (1983)
Perez1987a	Perez and Fraga (1987a)
Riordan2005	Riordan *et al.* (2005)
Roy1993b	Roy *et al.* (1993b)
Sundquist1979	Sundquist *et al.* (1979)
Weiss1970	Weiss (1970)
Weiss1974	Weiss (1974)
Wischmeyer2003	Wischmeyer *et al.* (2003)
Zeebe2001	Zeebe and Wolf-Gladrow (2001)

B. References for the elements of an object of class `aquaenv`

element	references
`p, P, Pa, p`	The relation between pressure and depth given in Fofonoff and Millard (1983) is used. The standard value for atmospheric pressure `Pa` at sea level as well as the definition of total pressure and gauge pressure is taken from Feistel (2008).
`Cl`	DOE (1994, chapter 5, p. 11), and Zeebe and Wolf-Gladrow (2001, p. 100, footnote 3)
`I`	DOE (1994, chapter 5, p. 13, 15), Zeebe and Wolf-Gladrow (2001, p.12), and Roy *et al.* (1993b, p.257). Note that the approximation I/(mol/kg-solution) $\approx$ 0.0199201 `S` is given in Millero (1982, p. 428.). This relationship converted into mol/kg-H_2O and the last digits adjusted (from 0.0199201 to 0.019924) results in the formula used here.
density	Millero and Poisson (1981) and DOE (1994, chapter 5, p. 6f).
`Br, ClConc, Na, Mg, Ca, K, Sr`	DOE (1994, chapter 5, p.11)
`molal2molin`	Roy *et al.* (1993b, p.257), and DOE (1994, chapter 5, p. 15)
`free2tot, tot2free`	Dickson (1984, p.2302), DOE (1994, chapter 4, p.16), Zeebe and Wolf-Gladrow (2001, p.57, 261)
`free2sws, tot2sws, sws2free, sws2tot`	Dickson (1984, p.2303), Zeebe and Wolf-Gladrow (2001, p.57)
`K0_CO2`	Weiss (1974), DOE (1994, chapter 5, p. 13) (here it is stated that the unit is mol/(kg-solution*atm)), Millero (1995, p.663), Zeebe and Wolf-Gladrow (2001, p.257)
`K0_O2`	derived from a formula for the oxygen saturation concentration in ml-O_2/kg-solution by Weiss (1970) using the first virial coefficient of oxygen (Atkins 1996, p. 41, 1029) and the atmospheric oxygen fugacity (Williams 2004)
`K_W`	Millero (1995, p.670) (**original reference**, but slightly different formula for seawater pH), DOE (1994, chapter 5, p. 18) (NOT the original reference! DOE (1994) cites in an update from 1997 Millero (1995)! However the version of the formula used here is the one converted to total pH scale given in DOE (1994)), and Zeebe and Wolf-Gladrow (2001, p. 258). Constant type (stoichiometric), pH scale (total, converted to free here) , and concentration unit (mol/kg-solution squared): DOE (1994, chapter 5, p. 12,18), pH scale also in Zeebe and Wolf-Gladrow (2001, p. 258).
`K_HSO4`	DOE (1994, chapter 5 page 13), Zeebe and Wolf-Gladrow (2001, p. 260), Dickson (1990b) (original reference). Constant type (stoichiometric), pH scale (free) , and concentration unit (mol/kg-H_2O converted to mol/kg-solution here): DOE (1994, chapter 5, p. 13). Note that it is also possible to use the constant according to Khoo *et al.* (1977), as cited in, e.g., Roy *et al.* (1993b), Millero (1995), and Lewis and Wallace (1998). In Lewis and Wallace (1998) it is stated that the constant resulting from this equation is in mol/kg-H_2O and on the free pH scale.
`K_HF`	Dickson and Riley (1979b, p. 91) (original reference), DOE (1994, c. 5, p. 15), Roy *et al.* (1993b, p. 257), Dickson and Millero (1987, p. 1783), Millero (1995, p. 664), Zeebe and Wolf-Gladrow (2001, p. 260) (converted to molinty and total scale). Constant type (stoichiometric), pH scale (free) , and concentration unit (mol/kg-H_2O converted to mol/kg-solution here): DOE (1994, chapter 5, p. 15, 16). In AquaEnv, it is also possible to use the constant according to Perez and Fraga (1987a).

`K_CO2`, `K_HCO3`	Roy *et al.* (1993b, p. 254) (original reference), DOE (1994, chapter 5, p.14) (in a version converted to mol/kg-H_2O), Millero (1995, p. 664), Zeebe and Wolf-Gladrow (2001, p. 255). Constant type (stoichiometric) and concentration unit (mol/kg-H_2O converted to mol/kg-solution here): DOE (1994, chapter 5, p. 14, 15), pH scale (total, converted to free here): In DOE (1994, chapter 5, p. 12) the total scale is stated for the formula for high salinities and thus can be inferred for the formula for low salinities. The scale is also indirectly stated for both formulations in the original reference Roy *et al.* (1993b). Note that in Roy *et al.* (1993b) a function for fresh water (based on Millero (1979) which in turn is on a temperature relationship from Harned and Davis (1943) and Harned and Scholes (1941) respectively) and a function for seawater is derived. In Millero (1995) it is stated that for S<5 the fresh water formula (based on Millero (1979)) should be used and for S>=5 the seawater formula derived in Roy *et al.* (1993b). However, both formulations do not always intersect at `S`=5. The true intersection with respect to salinity `S` is a function of temperature `t`. Here, we first calculate this intersection by numerical root finding and then decide which formulation to use. This practise results in a continuous function with respect to `S`. (Note that there is a typesetting error in Roy *et al.* (1993b): one of the numerical values for the function for $K^*_{CO_2}$ is given as 310.48919, but correct is 2310.48919. However, in Millero (1995) this value is stated correctly.) In `AquaEnv`, it is also possible to use the constants according to Lueker *et al.* (2000) and Millero *et al.* (2006).
`K_BOH3`	Dickson (1990a, p. 763) (original, but mol/kg-H_2O version), DOE (1994, ch. 5, p. 14), Zeebe and Wolf-Gladrow (2001, p. 262), Millero (1995, p.669) (mol/kg-H_2O version) , agrees with data in Roy *et al.* (1993a). Constant type (stoichiometric) and concentration unit (mol/kg-solution): DOE (1994, chapter 5, p. 14), pH scale (total): DOE (1994, chapter 5, p. 12) and Zeebe and Wolf-Gladrow (2001, p.263).
`K_NH4`	Millero *et al.* (1995) (original reference), Millero (1995, p.671). Constant type (stoichiometric) and concentration unit (mol/kg-solution): Millero (1995, p.671), pH scale (seawater, converted to free here): Lewis and Wallace (1998) (in corrections of Millero (1995)).
`K_H2S`	Millero *et al.* (1988) (original reference), Millero (1995, p.671). Constant type (stoichiometric) and concentration unit (mol/kg-solution): Millero (1995, p.671), pH scale (seawater, converted to free here): Lewis and Wallace (1998) (in corrections of Millero (1995)).
`K_H3PO4`, `K_H2PO4`, `K_HPO4`	Millero (1995, p.670) (original reference, but formula for seawater scale pH), DOE (1994, ch. 5, p 16,17), agrees with data in Dickson and Riley (1979a). Constant type (stoichiometric), concentration unit (mol/kg-solution), and pH scale (total, converted to free here): DOE (1994, chapter 5, p. 12, 16, 17).
`K_SiOH4`	Millero *et al.* (1988) (original reference), DOE (1994, chapter 5, p 17), Millero (1995, p.671) (formula for seawater scale pH) Constant type (stoichiometric), concentration unit (mol/kg-H_2O converted to mol/kg-solution here by omitting the conversion summand ln(1-0.001005 S)), and pH scale (total, converted to free here): DOE (1994, chapter 5, p. 12, 17).
`K_SiOOH3`	Wischmeyer *et al.* (2003) (original reference), corrected due to personal communication with Dieter Wolf-Gladrow (one of the authors). The corrected version can be obtained from either Dieter Wolf-Gladrow or Andreas F Hofmann (a.hofmann@nioo.knaw.nl). Constant type (stoichiometric), concentration unit (mol/kg-solution), and pH scale (total, converted to free here): Wischmeyer *et al.* (2003).
`K_HNO2`	Constant value, not a function of temperature and salinity! Obtained as a hybrid pk value (featuring the activity of the proton but the concentration of other species (see Zeebe and Wolf-Gladrow (2001) for a treatment of different types of equilibrium constants) in molar concentration (mol/l) on the NBS pH scale (Durst 1975) from Riordan *et al.* (2005). Used as an approximation for the stoichiometric $K^*_{HNO_2}$ in mol/kg-solution on the free proton pH scale here.
`K_H2SO4`	Constant value, not a function of temperature and salinity! Obtained as a standard pK value from Atkins (1996, p. 1045). Used as an approximation for the stoichiometric $K^*_{H_2SO_4}$ in mol/kg-solution on the free proton pH scale here.
`K_HS`	Constant value, not a function of temperature and salinity! Obtained as a standard pK value from Atkins (1996, p. 1045). Used as an approximation for the stoichiometric $K^*_{Hs^-}$ in mol/kg-solution on the free proton pH scale here.
`Ksp_calcite`, `Ksp_aragonite`	Mucci (1983) (original reference), Boudreau (1996). Note that in there are errors in Boudreau (1996): b_0 for calcite is not 0.7712 but 0.77712 and b_1 for aragonite is not 0.001727 but 0.0017276.

`pH`	As given in Dickson (1984), p. 2303 (use of "m") and Dickson and Riley (1979a), p. 91f all concentrations appearing in the definition of the total and the seawater pH scale are **molal** (mol/kg-H_2O) concentrations. But in Roy *et al.* (1993b), p. 257 and in DOE (1994), chapter 4, SOP 6, p. 1 it is stated, that concentrations for the seawater and total pH scale are in mol/kg-solution. To be consistent with DOE (1994) **molin** concentrations (mol/kg-solution) are chosen for calculating the pH.
`revelle`	N/A (function redundant in current version of **AquaEnv**)
`dTAdH`, `dTAdSumCO2`, `dTAdSumBOH3`, `dTAdSumH2SO4`, `dTAdSumHF`, `dTAdSumH3PO4`, `dTAdSumSumSiOH4`, `dTAdSumH2S`, `dTAdSumNH4`, `dTAdSumHNO3`, `dTAdSumHNO2`	Hofmann *et al.* (2008)
`c1`, `c2`, `c3`, `b1`, `b2`, `so1`, `so2`, `so3`, `f1`, `f2`, `p1`, `p2`, `p3`, `p4` `si1`, `si2`, `si3`, `s1`, `s2`, `s3`, `n1`, `n2`, `na1`, `na2`, `ni1`, `ni2`	Skoog and West (1982), Stumm and Morgan (1996), Hofmann *et al.* (2010a)
`dTAdKdKdS`, `dTAdKdKdT`, `dTAdKdKdp`, `dTAdKdKdSumH2SO4`, `dTAdKdKdSumHF`	Hofmann *et al.* (2009)

The values for `K_W`,`K_HSO4`, `K_HF`, `K_CO2`, `K_HCO3`, `K_BOH3`, `K_NH4`, `K_H2S`, `K_H3PO4`, `K_H2PO4`, `K_HPO4`, `K_SiOH4`, `K_SiOOH3`, `Ksp_calcite`, `Ksp_aragonite` obtained as functions of salinity `S` and temperature `t` from the above references are pressure corrected using the gauge pressure `p` according to Millero (1995) with corrections by Lewis and Wallace (1998).

In general it is to be said that all corrections from Lewis and Wallace (1998) have been applied.

References

Anderson LG, Turner DR, Wedborg M, Dyrssen D (1999). “Determination of total alkalinity and total dissolved inorganic carbon.” In K Grasshoff, K Gremling, M Ehrhardt (eds.), *Methods of Seawater Analysis*. Wiley-VCH.

Atkins PW (1996). *Physikalische Chemie*. 2nd edition. VCH Weinheim.

Boudreau BP (1996). “A method-of-lines code for carbon and nutrient diagenesis in aquatic sediments.” *Computers & Geosciences*, **22**(5), 479–496. `doi:10.1016/0098-3004(95)00115-8`.

Dickson AG (1981). "An Exact Definition of Total Alkalinity and a Procedure for the Estimation of Alkalinity and Total Inorganic Carbon from Titration Data." *Deep-Sea Research Part a-Oceanographic Research Papers*, **28**(6), 609–623. `doi:10.1016/0198-0149(81)90121-7`.

Dickson AG (1984). "pH Scales and Proton-Transfer Reactions in Saline Media Such as Sea-Water." *Geochimica et Cosmochimica Acta*, **48**(11), 2299–2308. `doi:10.1016/0016-7037(84)90225-4`.

Dickson AG (1990a). "Standard Potential of the Reaction - AgCl(S)+1/2H$_2$(G)=Ag(S)+HCl(Aq) and the Standard Acidity Constant of the Ion HSO$_4^-$ in Synthetic Sea-Water from 273.15-K to 318.15-K." *Journal of Chemical Thermodynamics*, **22**(2), 113–127. `doi:10.1016/0021-9614(90)90074-Z`.

Dickson AG (1990b). "Thermodynamics of the Dissociation of Boric-Acid in Synthetic Seawater from 273.15-K to 318.15-K." *Deep-Sea Research Part a-Oceanographic Research Papers*, **37**(5), 755–766. `doi:10.1016/0198-0149(90)90004-F`.

Dickson AG, Millero FJ (1987). "A Comparison of the Equilibrium-Constants for the Dissociation of Carbonic-Acid in Seawater Media." *Deep-Sea Research Part a-Oceanographic Research Papers*, **34**(10), 1733–1743. `doi:10.1016/0198-0149(87)90021-5`.

Dickson AG, Riley JP (1979a). "Estimation of Acid Dissociation-Constants in Seawater Media from Potentiometric Titrations with Strong Base .1. Ionic Product of Water - Kw." *Marine Chemistry*, **7**(2), 89–99. `doi:10.1016/0304-4203(79)90001-X`.

Dickson AG, Riley JP (1979b). "Estimation of Acid Dissociation-Constants in Seawater Media from Potentiometric Titrations with Strong Base .2. Dissociation of Phosphoric-Acid." *Marine Chemistry*, **7**(2), 101–109. `doi:10.1016/0304-4203(79)90002-1`.

Dickson AG, Sabine C, Christian JR (eds.) (2007). *Guide to best practices for ocean CO2 measurements.* 3. PICES special publications.

DOE (1994). *Handbook of Methods for the Analysis of the Various Parameters of the Carbon Dioxide System in Sea Water.* ORNL/CDIAC-74.

Durst A (1975). *Standard Reference Materials: Standardization of pH Measurements*, volume 260-53 of *NBS Spec. Publ.* National Bur. Standards, Washington, D.C.

Emerson S, Hedges JI (2008). "Carbonate Chemistry." In *Chemical Oceanography and the Marine Carbon Cycle*, pp. 101–133. Cambridge University Press, Cambridge.

Feistel R (2008). "A Gibbs function for seawater thermodynamics for-6 to 80 degrees C and salinity up to 120 g kg(-1)." *Deep-Sea Research Part I-Oceanographic Research Papers*, **55**(12), 1639–1671. `doi:10.1016/j.dsr.2008.07.004`.

Fofonoff NP, Millard RCJ (1983). "Algorithms for computation of fundamental properties of seawater." *Unesco Technical Papers in Marine Science*, **44**, 55 pp.

Follows MJ, Ito T, Dutkiewicz S (2006). "On the solution of the carbonate chemistry system in ocean biogeochemistry models." *Ocean Modelling*, **12**(3-4), 290–301. `doi:10.1016/j.ocemod.2005.05.004`.

Gran G (1952). "Determination of the Equivalence Point in Potentiometric Titrations .2." *Analyst*, **77**(920), 661–671. doi:10.1039/AN9527700661.

Hagens M, Middelburg JJ (2016). "Generalised expressions for the response of pH to changes in ocean chemistry." *Geochimica et Cosmochimica Acta*, **187**, 334–349. doi:10.1016/j.gca.2016.04.012.

Hansson I, Jagner D (1973). "Evaluation of Accuracy of Gran Plots by Means of Computer Calculations - Application to Potentiometric Titration of Total Alkalinity and Carbonate Content in Sea-Water." *Analytica Chimica Acta*, **65**(2), 363–373. doi:10.1016/S0003-2670(01)82503-4.

Haraldsson C, Anderson LG, Hassellov M, Hulth S, Olsson K (1997). "Rapid, high-precision potentiometric titration of alkalinity in ocean and sediment pore waters." *Deep-Sea Research Part I-Oceanographic Research Papers*, **44**(12), 2031–2044. doi:10.1016/S0967-0637(97)00088-5.

Harned HS, Davis R (1943). "The Ionization Constant of Carbonic Acid in Water and the Solubility of Carbon Dioxide in Water and Aqueous Salt Solutions from 0 to 50 deg C." *Journal of the American Chemical Society*, **65**(10), 2030–2037. doi:10.1021/ja01250a059.

Harned HS, Scholes SR (1941). "The Ionization Constant of HCO3- from 0 to 50 deg C." *Journal of the American Chemical Society*, **63**(6), 1706–1709. doi:10.1021/ja01851a058.

Hofmann AF, Meysman FJR, Soetaert K, Middelburg JJ (2008). "A step-by-step procedure for pH model construction in aquatic systems." *Biogeosciences*, **5**(1), 227–251. doi:10.5194/bg-5-227-2008.

Hofmann AF, Meysman FJR, Soetaert K, Middelburg JJ (2009). "Factors governing the pH in a heterotrophic, turbid, tidal estuary." *Biogeosciences*, **6**, 1539–1561. doi:10.5194/bg-6-1539-2009.

Hofmann AF, Middelburg J, Soetaert K, Wolf-Gladrow DA, Meysman F (2010a). "Proton cycling, buffering, and reaction stoichiometry in natural waters." *Marine Chemistry*, **121**, 246–255. doi:10.1016/j.marchem.2010.05.004.

Hofmann AF, Soetaert K, Middelburg JJ, Meysman FJR (2010b). "AquaEnv - An Aquatic Acid-Base Modelling Environment in R." *Aquatic Geochemistry*, **16**, 507–546. doi:10.1007/s10498-009-9084-1.

Khoo KH, Ramette RW, Culberson CH, Bates RG (1977). "Determination of hydrogen ion concentrations in seawater from 5 to 40 °C: standard potentials at salinities from 20 to 45 ‰." *Analytical Chemistry*, **49**, 29–34. doi:10.1021/ac50009a016.

Lewis EL, Wallace DWR (1998). "Program Developed for CO_2 System Calculations." ORNL/CDIAC-105. Carbon Dioxide Information Analysis Center, Oak Ridge National Laboratory, U.S. Department of Energy, Oak Ridge, Tennessee.

Lueker TJ, Dickson AG, Keeling CD (2000). "Ocean pCO_2 calculated from dissolved inorganic carbon, alkalinity, and equations for K_1 and K_2: validation based on laboratory measurements of CO_2 in gas and seawater at equilibrium." *Marine Chemistry*, **70**(1-3), 105–119. doi:10.1016/S0304-4203(00)00022-0.

Millero FJ (1979). "The Thermodynamics of the Carbonate System in Seawater." *Geochimica et Cosmochimica Acta*, **43**(10), 1651–1661. `doi:10.1016/0016-7037(79)90184-4`.

Millero FJ (1982). "The Thermodynamics of Seawater, Part 1. The PVT Properties." *Ocean Science and Engineering*, **7**(4), 403–460.

Millero FJ (1995). "Thermodynamics of the Carbon-Dioxide System in the Oceans." *Geochimica et Cosmochimica Acta*, **59**(4), 661–677. `doi:10.1016/0016-7037(94)00354-O`.

Millero FJ, Graham TB, Huang F, Bustos-Serrano H, Pierrot D (2006). "Dissociation constants of carbonic acid in seawater as a function of salinity and temperature." *Marine Chemistry*, **100**(1-2), 80–94. `doi:10.1016/j.marchem.2005.12.001`.

Millero FJ, Plese T, Fernandez M (1988). "The Dissociation of Hydrogen-Sulfide in Seawater." *Limnology and Oceanography*, **33**(2), 269–274. `doi:10.4319/lo.1988.33.2.0269`.

Millero FJ, Poisson A (1981). "International One-Atmosphere Equation of State of Seawater." *Deep-Sea Research Part a-Oceanographic Research Papers*, **28**(6), 625–629. `doi:10.1016/0198-0149(81)90122-9`.

Millero FJ, Yao WS, Aicher J (1995). "The Speciation of Fe(II) and Fe(III) in Natural-Waters." *Marine Chemistry*, **50**(1-4), 21–39. `doi:10.1016/0304-4203(95)00024-L`.

Mucci A (1983). "The Solubility of Calcite and Aragonite in Seawater at Various Salinities, Temperatures, and One Atmosphere Total Pressure." *American Journal of Science*, **283**(7), 780–799. `doi:10.2475/ajs.283.7.780`.

Perez FF, Fraga F (1987a). "Association Constant of Fluoride and Hydrogen-Ions in Seawater." *Marine Chemistry*, **21**(2), 161–168. `doi:10.1016/0304-4203(87)90036-3`.

Perez FF, Fraga F (1987b). "The pH Measurements in Seawater on the NBS Scale." *Marine Chemistry*, **21**(4), 315–327. `doi:10.1016/0304-4203(87)90054-5`.

Riordan E, Minogue N, Healy D, O'Driscoll P, Sodeau JR (2005). "Spectroscopic and optimization modeling study of nitrous acid in aqueous solution." *Journal of Physical Chemistry A*, **109**(5), 779–786. `doi:10.1021/jp040269v`.

Roy RN, Roy LN, Lawson M, Vogel KM, Moore CP, Davis W, Millero FJ (1993a). "Thermodynamics of the Dissociation of Boric-Acid in Seawater at S=35 from 0-Degrees-C to 55-Degrees-C." *Marine Chemistry*, **44**(2-4), 243–248. `doi:10.1016/0304-4203(93)90206-4`.

Roy RN, Roy LN, Vogel KM, Porter-Moore C, Pearson T, Good CE, Millero FJ, Campbell DM (1993b). "The dissociation constants of carbonic acid in seawater at salinities 5 to 45 and temperatures O to 45 degrees C (vol 44, pg 249, 1996)." *Marine Chemistry*, **44**(2–4), 249–267. `doi:10.1016/0304-4203(93)90207-5`.

Skoog DA, West DM (1982). *Fundamentals of Analytical Chemistry*. Holt-Saunders International Editions. Holt-Saunders.

Soetaert K, Petzoldt T, Setzer RW (2010). "Solving Differential Equations in R: Package deSolve." *Journal of Statistical Software*, **33**(9), 1–25. `doi:10.18637/jss.v033.i09`.

Stumm W, Morgan JJ (1996). *Aquatic Chemistry: Chemical Equilibria and Rates in natural Waters*. Wiley Interscience, New York.

Sundquist ET, Plummer LN, Wigley TML (1979). "Carbon-Dioxide in the Ocean Surface - Homogeneous Buffer Factor." *Science*, **204**(4398), 1203–1205. doi:10.1126/science.204.4398.1203.

Weiss RF (1970). "The Solubility of Nitrogen, Oxygen and Argon in Water and Seawater." *Deep-Sea Research*, **17**(4), 721–735. doi:10.1016/0011-7471(70)90037-9.

Weiss RF (1974). "Carbon dioxide in water and seawater: the solubility of a non-ideal gas." *Marine Chemistry*, **2**(3), 203–215. doi:10.1016/0304-4203(74)90015-2.

Williams DR (2004). "NASA Earth Fact Sheet."

Wischmeyer AG, Del Amo Y, Brzezinski M, Wolf-Gladrow DA (2003). "Theoretical constraints on the uptake of silicic acid species by marine diatoms." *Marine Chemistry*, **82**(1-2), 13–29. doi:10.1016/S0304-4203(03)00033-1.

Zeebe RE, Wolf-Gladrow D (2001). CO_2 *in Seawater: Equilibrium, Kinetics, Isotopes*. Number 65 in Elsevier Oceanography Series. Elsevier.

Affiliation:

Karline Soetaert
Royal Netherlands Institute
of Sea Research (NIOZ)
Yerseke, The Netherlands
and
Mathilde Hagens
Utrecht University