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

Andreas F. Hofmann
Centre for Estuarine and Marine Ecology
Netherlands Institute of Ecology
The Netherlands

Abstract

AquaEnv is an integrated development toolbox for aquatic chemical model generation focused on (ocean) acidification and CO2 air-water exchange.

- It contains all elements necessary to model the pH, the related CO2 air-water exchange, as well as aquatic acid-base chemistry in general for an arbitrary marine, estuarine or freshwater system. Also chemical batches can be modelled.
- Next to the routines necessary to calculate desired information, **AquaEnv** also contains a suite of tools to visualize this information.
- Furthermore, **AquaEnv** can not only be used to build dynamic models of aquatic systems, but it can also serve as a simple desktop tool for the experimental aquatic chemist to generate and visualize all possible derived information from a set of measurements with one single easy to use R function.
- Additionally, the sensitivity of the system to variations in the input variables can be visualized.
- **AquaEnv** also contains a number of example "applications" that make use of the aquatic modelling toolbox that **AquaEnv** provides:
  - a theoretical titration simulator
  - and a routine to determine 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 ($K^*_{CO_2}$)

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, R.

Contents

1 Introduction

AquaEnv is a toolbox for aquatic modelling that serves several purposes

- It provides 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, KO_CO2, KO_O2, Ksp_aragonite,` and `Ksp_calcite`.

- It is designed to make its use as easy as possible: all the information that can be calculated from the set of parameters know of a system or sample can be obtained by one single function: `aquaenv`. This function returs a list of class *aquaenv* that contains next 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) (Please note that if values for $[\sum B(OH)_3]$, $[\sum H_2SO_4]$, $[\sum HF]$ are given as input parameters, these parameters are used and not the ones calculated from salinity.)
  - the hydrostatic pressure calculated from the given depth and 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, Roy, Vogel, PorterMoore, Pearson, Good, Millero, and Campbell (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), NH4$^+$(Millero, Yao, and Aicher 1995), $H2_S$ (Millero 1995), H_3PO4(Millero 1995), H_2PO4^- (Millero 1995), $HPO4^{2-}$ (Millero 1995), SiOH4 (Millero, Plese, and Fernandez 1988), $SiOOH3^-$ (Wischmeyer, Del Amo, Brzezinski, and Wolf-Gladrow 2003), HNO2 (Riordan, Minogue, Healy, O'Driscoll, and Sodeau 2005), HNO3, H2SO4 (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 partial pressure 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 pCO_2 are given, pH and [TA] are calculated.
  - if either one of the pairs pH and $[CO_2]$ or pCO_2, pH and [TA], or [TA] and $[CO_2]$ or pCO_2 is given, $[\sum CO_2]$ is calculated

  - if sufficient information is given and the flag `speciation=TRUE` is set, a full speciation of $[\sum \mathrm{CO2}]$, $[\sum \mathrm{NH4}]$, $[\sum \mathrm{H_2S}]$, $[\sum \mathrm{HNO3}]$, $[\sum \mathrm{HNO2}]$, $[\sum \mathrm{H_3PO4}]$, $[\sum \mathrm{Si(OH)_4}]$, $[\sum \mathrm{B(OH)_3}]$, $[\sum \mathrm{H_2SO_4}]$, $[\sum \mathrm{HF}]$, as well as water itself is calculated
  - if the flag `revelle=TRUE` is set, the revelle factor (Zeebe and Wolf-Gladrow 2001) is calculated. item if the flag `revelle=TRUE` is set, all necessary quantities for the explicit "direct substitution approach" (DSA) to pH modelling as given in Hofmann, Meysman, Soetaert, and Middelburg (2008b) 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, Meysman, Soetaert, and Middelburg (2008a) are calculated. Finally, the ionization fractions as defined by Stumm and Morgan (1996) and used in Hofmann, Middelburg, Soetaert, Wolf-Gladrow, and Meysman (2008c) are calculated for the full speciation.

- Input for `aquaenv` has to be supplied in standard SI units, the free proton pH scale and in molinity[1] (mol/kg-solution). Conversion of input parameters to this 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 temperature T, salinity S or 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 type *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 e.g. with **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 type *aquaenv* which allows the user to use the whole suite of visualisation tools that is provided by the function `plot` in **AquaEnv**.

- As mentioned above Hofmann *et al.* (2008b), Hofmann *et al.* (2008a), and Hofmann *et al.* (2008c) describe methods for an "explicit" pH modelling which allows for the quantification of the influences of kinetically modelled processes on the pH. Objects of type *aquaenv* provide all needed quantities (partial derivatives of [TA], ionization fractions, etc.) to employ both of those methods in dynamic models. Furthermore, **AquaEnv** provides the functionality to cumulatively plot the obtained influences on the pH.

- As an example of how to use the toolbox that is **AquaEnv**, two applications are provided

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

– The function `titration`: creates theoretical titrations which can be used e.g. to create bjerrum plots, something that can also be done with the function `plot` 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 **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(10, 35)
> test$Tc
> test[["Tc"]]
```

Maximally, i.e., if the enough input data is supplied to define the pH of the system and the flags `speciation`, `dsa`, and `revelle` are `TRUE` while the flag `skeleton` is `FALSE`, an object of class *aquaenv* contains the following elements

element	unit	explanation
Tc	°C	temperature
Tk	K	absolute temperature
S	"psu" (no unit)	salinity
Cl	‰	chlorinity
I	mol/kg-H_2O	ionic strength
d	m	depth
hydroP	bar	hydrostatic pressure
density	kg/m^3	(seawater) density
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
SumBr	mol/kg-soln	[$\sum HBr$], total 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
CO2_sat	mol/kg-soln	CO_2 saturation concentration at an atmospheric partial pressure/fugacity of Fugacity$CO2
O2_sat	mol/kg-soln	O_2 saturation concentration at an atmospheric partial pressure/fugacity of Fugacity$O2
K_W	$(\text{mol/kg-soln})^2$, 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	$(\text{mol/kg-soln})^2$	stoichiometric equilibrium solubility product of calcite $Ksp^*_{cal} = [Ca^{2+}][CO_3^{2-}]$
Ksp_aragonite	$(\text{mol/kg-soln})^2$	stoichiometric equilibrium solubility product of aragonite $Ksp^*_{ara} = [Ca^{2+}][CO_3^{2-}]$
TA	mol/kg-soln	[TA], total alkalinity
pH	-, free scale	pH
pCO2	atm,	partial pressure (fugacity) of CO_2 in 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	$[H2PO_4^-]$
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
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 $[TA] = f([H^+], [\sum CO_2], ...)$
dTAdH	-	$\frac{\partial[TA]}{[\partial[H^+]}$: buffer factor with $[TA] = f([H^+], [\sum CO_2], ...)$
dTAdKdKdS	-	$\sum_i \frac{\partial[TA]}{\partial K_i^*} \frac{\partial K_i^*}{\partial S}$ with $[TA] = f([H^+], [\sum CO_2], ..., K_i^*)$
dTAdKdKdT	-	$\sum_i \frac{\partial[TA]}{\partial K_i^*} \frac{\partial K_i^*}{\partial T}$ with $[TA] = f([H^+], [\sum CO_2], ..., K_i^*)$
dTAdKdKdd	-	$\sum_i \frac{\partial[TA]}{\partial K_i^*} \frac{\partial K_i^*}{\partial d}$ with $[TA] = f([H^+], [\sum CO_2], ..., K_i^*)$
dTAdKdKdSumH2SO4	-	$\sum_i \frac{\partial[TA]}{\partial K_i^*} \frac{\partial K_i^*}{\partial[\sum H_2SO_4]}$ with $[TA] = f([H^+], [\sum CO_2], ..., K_i^*)$
dTAdKdKdSumHF	-	$\sum_i \frac{\partial[TA]}{\partial K_i^*} \frac{\partial K_i^*}{\partial[\sum HF]}$ with $[TA] = f([H^+], [\sum CO_2], ..., K_i^*)$

For elements that are calculated according to certain literature references, those references are given in appendix B.

3 Using **AquaEnv**

3.1 Basic features

3.1.1 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`, `KO_CO2`, `KO_O2`, `Ksp_aragonite`, and `Ksp_calcite` can be calculated directly, witout creating an object of class *aquaenv*. This is done via functions that bear the same as those elements

```
> K_CO2(15, 30)
> K0_CO2(15, 30)
> Ksp_calcite(15, 30, 100)
```

3.1.2 Minimal *aquaenv* definition

Minimally, an object of class *aquaenv* can be defined with just a temperature and salinity value

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

Optionally, a mean depth 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(Tc = 15, S = 30, d = 10)
> ae$Ksp_calcite
> names(ae)
```

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

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

3.1.3 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 SumCO2 and pH, SumCO2 and TA, SumCO2 and CO2, or SumCO2 and pCO2, a full *aquaenv* system can be defined.

```
> Tc <- 15
> S <- 30
> d <- 10
> SumCO2 <- 0.002
> pH <- 8
> TA <- 0.002140323
> pCO2 <- 0.000533576
> CO2 <- 2.055419e-05
> ae <- aquaenv(Tc, S, d, SumCO2 = SumCO2, pH = pH)
> ae$TA
> ae <- aquaenv(Tc, S, d, SumCO2 = SumCO2, TA = TA)
> ae$pH
> ae <- aquaenv(Tc, S, d, SumCO2 = SumCO2, CO2 = CO2)
> ae$pH
> names(ae)
```

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

Furtermore all the quantities needed for the explicit pH modelling approaches as given in Hofmann *et al.* (2008b) and Hofmann *et al.* (2008c) can be calculated by setting the flag `dsa` to `TRUE`. The revelle factor can be calculated using the flag `revelle`.

```
> ae <- aquaenv(Tc, S, d, SumCO2 = SumCO2, pCO2 = pCO2, dsa = TRUE,
+     revelle = TRUE)
> ae$dTAdH
> ae$revelle
```

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

```
> ae <- aquaenv(Tc, S, d, SumCO2 = SumCO2, CO2 = CO2, pCO2 = pCO2)
> ae <- aquaenv(Tc, S, d, SumCO2 = SumCO2, pH = pH, TA = TA)
> ae <- aquaenv(Tc, S, d, SumCO2 = SumCO2, pH = pH, CO2 = CO2)
> ae <- aquaenv(Tc, S, d, SumCO2 = SumCO2, pH = pH, pCO2 = pCO2)
> ae <- aquaenv(Tc, S, d, SumCO2 = SumCO2, TA = TA, CO2 = CO2)
> ae <- aquaenv(Tc, S, d, SumCO2 = SumCO2, TA = TA, pCO2 = pCO2)
```

3.1.4 Calculating $[\sum CO_2]$

$[\sum CO_2]$ can be calculated by giving a constant pair of either pH and CO2, pH and pCO2, pH and TA, TA and CO2, or TA and pCO2

```
> pCO2 <- 0.0006952296
> CO2 <- 2.678134e-05
> pH <- 7.888569
> TA <- 0.0021
> Tc <- 15
> S <- 30
> d <- 10
> ae <- aquaenv(Tc, S, d, SumCO2 = NULL, pH = pH, CO2 = CO2)
> ae$SumCO2
> ae <- aquaenv(Tc, S, d, SumCO2 = NULL, pH = pH, pCO2 = pCO2)
> ae$SumCO2
> ae <- aquaenv(Tc, S, d, SumCO2 = NULL, pH = pH, TA = TA)
> ae$SumCO2
> ae <- aquaenv(Tc, S, d, SumCO2 = NULL, TA = TA, CO2 = CO2)
```

```
> ae$SumCO2
> ae <- aquaenv(Tc, S, d, SumCO2 = NULL, TA = TA, pCO2 = pCO2)
> ae$SumCO2
```

3.1.5 Cloning an object of class *aquaenv*

It is possible to clone an obect of class *aquaenv*, either 1 to 1 or with different pH, TA, or K_CO2

```
> Tc <- 15
> S <- 30
> SumCO2 <- 0.002
> TA <- 0.00214
> ae <- aquaenv(Tc, S, SumCO2 = SumCO2, TA = TA)
> ae$pH
> ae1 <- aquaenv(ae = ae)
> ae1$pH
> pH <- 9
> ae2 <- aquaenv(ae = ae, pH = pH)
> ae2$TA
> TA <- 0.002
> ae3 <- aquaenv(ae = ae, TA = TA)
> ae3$pH
> K_CO2 <- 1e-06
> ae4 <- aquaenv(ae = ae, k_co2 = K_CO2)
> ae4$pH
```

Note that K_CO2 as an input variable is in lower cases!

3.1.6 Preparing input variables

Input variables 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`.

```
> Tc <- 15
> S <- 10
> pH_NBS <- 8.142777
> SumCO2molar <- 0.002016803
> pH_free <- convert(pH_NBS, "pHscale", "nbs2free", Tc = Tc, S = S)
> SumCO2molin <- convert(SumCO2molar, "conc", "molar2molin", Tc = Tc,
+     S = S)
> ae <- aquaenv(Tc, S, SumCO2 = SumCO2molin, pH = pH_free)
> ae$pH
> ae$SumCO2
```

3.1.7 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 <- 0.002
> pH <- 8
> Tc <- 1:15
> S <- 30
> d <- 10
> ae <- aquaenv(Tc, S, d, SumCO2 = SumCO2, pH = pH, revelle = TRUE)
> ae$revelle
```

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.

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.

```
> plot(ae, xval = Tc, xlab = "T/(deg C)", newdevice = FALSE)
```

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

The following chunks of example code show other possible definitions of objects of class *aquaenv* with vectors as input variables.

```
> ae <- aquaenv(Tc = 15, S = 20:30, d = 10, SumCO2 = SumCO2, pH = pH,
+     dsa = TRUE)
> plot(ae, xval = 20:30, xlab = "S")

> ae <- aquaenv(Tc = 15, S = 30, d = seq(1, 1000, 100), SumCO2 = SumCO2,
+     pH = pH, revelle = TRUE)
> plot(ae, xval = seq(1, 1000, 100), xlab = "depth/m")

> ae <- aquaenv(Tc = 1:15, S = 30, d = 10, SumCO2 = SumCO2, TA = 0.0023)
> plot(ae, xval = 1:15, xlab = "T/(deg C)")

> ae <- aquaenv(Tc = 15, S = 20:30, d = 10, SumCO2 = SumCO2, TA = TA)
> plot(ae, xval = 20:30, xlab = "S")

> ae <- aquaenv(Tc = 15, S = 30, d = seq(1, 1000, 200), SumCO2 = SumCO2,
+     TA = TA, revelle = TRUE, dsa = TRUE)
> plot(ae, xval = seq(1, 1000, 200), xlab = "depth/m")
```

Interesting to note is that also, e.g., SumCO2, TA, pH and SumNH4 can be vectors

```
> ae <- aquaenv(10, 20, SumCO2 = seq(0.001, 0.002, 1e-04), TA = 0.002)
> plot(ae, xval = ae$SumCO2, xlab = "SumCO2/(mol/kg-soln)", what = c("pH",
+     "CO2", "HCO3", "CO3"), newdevice = FALSE, mfrow = c(1, 4))

> ae <- aquaenv(10, 20, SumCO2 = 0.002, TA = seq(0.001, 0.002,
+     1e-04))
> plot(ae, xval = ae$TA, xlab = "TA/(mol/kg-soln)", what = c("pH",
+     "CO2", "HCO3", "CO3"), newdevice = FALSE, mfrow = c(1, 4))

> ae <- aquaenv(10, 20, SumCO2 = 0.002, pH = seq(8, 8.2, 0.001))
> plot(ae, xval = ae$pH, xlab = "pH (free scale)", what = c("pH",
+     "CO2", "HCO3", "CO3"), newdevice = FALSE, mfrow = c(1, 4))

> ae <- aquaenv(10, 20, SumCO2 = 0.002, SumNH4 = seq(1e-04, 2e-04,
+     1e-05), pH = 8)
> ae$NH3
```

3.1.8 Calculating $[\sum CO_2]$ from input vectors

The functionality of calculating SumCO2 can also be used together with vectors as input variables.

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

Two further examples

```
> ae <- aquaenv(Tc = 15, S = 20:30, SumCO2 = NULL, pH = pH, pCO2 = pCO2)
> plot(ae, xval = 20:30, xlab = "S")

> ae <- aquaenv(Tc = 15, S = 30, d = seq(1, 1000, 100), SumCO2 = NULL,
+     pH = pH, TA = TA)
> plot(ae, xval = seq(1, 1000, 100), xlab = "depth/m")
```

3.1.9 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**, and will be shown later in this document.

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

3.1.10 Converting elements in an obect of class *aquaenv*

Elements of an object of class *aquaenv* are calculated in, e.g., the concentration 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(10, 30)
> ae$SumBOH3
> ae <- convert(ae, "mol/kg-soln", "umolkg-H2O", 1e+06/ae$molal2molin,
+     "unit")
> ae$SumBOH3
```

3.1.11 Quantities needed for explicit pH modelling

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

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

This command calculated the buffer factor and the partial derivatives of [TA] with respect to other summed quantities referred to in Hofmann *et al.* (2008b)

```
> ae$dTAdH
> ae$dTAdSumCO2
```

as well the sums 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, SumH2SO4, od SumHF) as introduced in Hofmann *et al.* (2008a).

```
> ae$dTAdKdKdS
> ae$dTAdKdKdSumH2SO4
```

Furthermore the ionization fractions used for the pH dependent fractional stoichiometric pH modelling approach described in Hofmann *et al.* (2008c) are calculated as well

```
> ae$c1
```

3.2 The `plot.aquaenv` function

In the previous sections, the `plot` function has been introduced. What actually is called if the first element of the arguments list of `plot` is an object of type *aquaenv* is the function `plot.aquaenv`. 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 (this feature is needed here to create a plot that will be woven into the LaTeX document by Sweave).

For example

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

and

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

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".

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

These features make the function `plot.aquaenv` different from standard R plotting functions. However, if the flags `newdevice` and `setpar` are set to `FALSE`, `plot.aquaenv` behaves like a "normal" R plotting function

```
> par(mfrow = c(1, 2))
> plot(ae, xval = 20:30, xlab = "S", what = "K_CO2", lwd = 3, col = "red",
+     newdevice = FALSE, setpar = FALSE)
> plot(ae, xval = 20:30, xlab = "S", what = "K_HCO3", cex = 3,
+     type = "b", col = "blue", newdevice = FALSE, setpar = FALSE)
```

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

3.3 Using objects of class *aquaenv* in dynamic models

3.3.1 Ordinary dynamic models

It is convenient to use objects of class *aquaenv* in a dynamic model, e.g. solved using the R package **deSolve**. This can be illustrated with an example. (For information about how to set up a dynamic model with **deSolve**, consult the documentation of **deSolve**).

```
> require(deSolve)
```

A list of parameters is specified

```
> parameters <- list(S = 25, Tc_min = 5, Tc_max = 25, d = 10, k = 0.4,
+     rOx = 3e-07, rNitri = 2e-07, rPP = 6e-06, ksDINPP = 1e-06,
+     ksNH4PP = 1e-06, D = 0.1, O2_io = 0.000296, NO3_io = 3.5e-05,
+     SumNH4_io = 8e-06, SumCO2_io = 0.00232, TA_io = 0.002435,
+     C_Nratio = 8, a = 30, b = 50, modeltime = 100)
```

A model function is defined which will be executed every timestep of the numerical integration. An object of class *aquaenv* is created in each timestep, some of its elements are used to calculated kinetic rate expressions and the whole object is returned as output.

```
> Waddenzeebox <- function(timestep, currentstate, parameters) {
+     with(as.list(c(currentstate, parameters)), {
+         Tc <- c(seq(Tc_min, Tc_max, (Tc_max - Tc_min)/(modeltime/2)),
+             seq(Tc_max, Tc_min, -(Tc_max - Tc_min)/(modeltime/2)))[[round(timestep) +
+             1]]
+         ae <- aquaenv(Tc = Tc, S = S, SumCO2 = SumCO2, SumNH4 = SumNH4,
+             TA = TA)
+         ECO2 <- k * (ae$CO2_sat - ae$CO2)
+         EO2 <- k * (ae$O2_sat - O2)
+         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
+         ROx <- rOx
+         ROxCarbon <- ROx * C_Nratio
+         pNH4PP <- 0
+         RPP <- 0
+         if ((timestep > a) && (timestep < b)) {
+             RPP <- rPP * ((SumNH4 + NO3)/(ksDINPP + (SumNH4 +
+                 NO3)))
+             pNH4PP <- 1 - (ksNH4PP/(ksNH4PP + SumNH4))
+         }
+         else {
+             RPP <- 0
+         }
+         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)
+         transport <- c(TO2 = TO2, TNO3 = TNO3, TSumNH4 = TSumNH4,
+             TTA = TTA, TSumCO2 = TSumCO2)
+         airseaexchange <- c(ECO2 = ECO2, EO2 = EO2)
+         return(list(ratesofchanges, list(transport, airseaexchange,
+             ae)))
+     })
+ }
```

The model is solved

```
> with(as.list(parameters), {
+     initialstate <<- c(O2 = O2_io, NO3 = NO3_io, SumNH4 = SumNH4_io,
+         SumCO2 = SumCO2_io, TA = TA_io)
+     times <<- c(0:modeltime)
+     output <<- as.data.frame(vode(initialstate, times, Waddenzeebox,
+         parameters, hmax = 1))[-1, ]
+ })
```

and the output can be plotted in the same way as a two dimensional object of class *aquaenv* by converting it to an object of class *aquaenv* using the `from.data.frame` flag of the function `aquaenv`

```
> plot(aquaenv(ae = output, from.data.frame = TRUE), xval = output$time,
+     xlab = "time/d", mfrow = c(10, 10), newdevice = FALSE)
```

3.3.2 Models using the explicit pH modelling approach

3.3.2.1 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.* (2008b,a,c), they can be readily 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.* (2008b) and additionally employs fractional stoichiometry as given in Hofmann *et al.* (2008c). 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, Tc = 15, d = 10, k = 0.4, rOx = 3e-07,
+     rNitri = 2e-07, rPP = 6e-07, ksSumNH4 = 1e-06, D = 0.1, O2_io = 0.000296,
+     NO3_io = 3.5e-05, SumNH4_io = 8e-06, SumCO2_io = 0.00232,
+     TA_io = 0.002435, C_Nratio = 8, a = 30, b = 50, modeltime = 100)
```

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

```
> boxmodel <- function(timestep, currentstate, parameters) {
+     with(as.list(c(currentstate, parameters)), {
+         ae <- aquaenv(Tc = Tc, S = S, SumCO2 = SumCO2, SumNH4 = SumNH4,
+             TA = TA, dsa = TRUE)
+         ECO2 <- k * (ae$CO2_sat - ae$CO2)
+         EO2 <- k * (ae$O2_sat - O2)
+         RNit <- rNitri
+         ROx <- rOx
+         if ((timestep > a) && (timestep < b)) {
+             RPP <- rPP * (SumNH4/(ksSumNH4 + SumNH4))
+         }
+         else {
+             RPP <- 0
+         }
+         dO2 <- EO2 - C_Nratio * ROx - 2 * RNit + C_Nratio * RPP
+         dNO3 <- RNit
+         dSumCO2 <- ECO2 + C_Nratio * ROx - C_Nratio * RPP
+         dSumNH4 <- ROx - RNit - RPP
+         dTA <- ROx - 2 * RNit - RPP
+         dH <- (dTA - (dSumCO2 * ae$dTAdSumCO2 + dSumNH4 * ae$dTAdSumNH4))/ae$dTAdH
+         DSApH <- -log10(H)
+         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(dO2, dNO3, dSumNH4, dSumCO2, dTA,
+             dH, dH_stoich)
+         processrates <- c(ECO2 = ECO2, EO2 = EO2, 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, list(processrates, DSA, ae)))
+     })
+ }
```

The model is solved

```
> with(as.list(parameters), {
+     H_init <<- 10^(-(aquaenv(Tc = Tc, S = S, SumCO2 = SumCO2_io,
+         SumNH4 = SumNH4_io, TA = TA_io, speciation = FALSE)$pH))
+     initialstate <<- c(O2 = O2_io, NO3 = NO3_io, SumNH4 = SumNH4_io,
+         SumCO2 = SumCO2_io, TA = TA_io, H = H_init, H_stoich = H_init)
+     times <<- c(0:modeltime)
+     output <<- as.data.frame(vode(initialstate, times, boxmodel,
+         parameters, hmax = 1))[-1, ]
+ })
```

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

```
> what <- c("SumCO2", "TA", "SumNH4", "NO3", "ECO2", "EO2", "RNit",
+     "ROx", "RPP", "dTAdH", "dTAdSumCO2", "dTAdSumNH4", "c1",
+     "c2", "c3", "n1", "n2", "rhoHECO2", "rhoHRNit", "rhoHROx",
+     "rhoHRPP", "dH_ECO2", "dH_RNit", "dH_ROx", "dH_RPP", "pH",
+     "DSApH", "DSAstoichpH")
> plot(aquaenv(ae = output, from.data.frame = TRUE), xval = output$time,
+     what = what, xlab = "time/d", mfrow = c(6, 5), size = c(20,
+         13), newdevice = FALSE)
```

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 = output, from.data.frame = TRUE), xval = output$time,
+     what = what, xlab = "time/d", size = c(7, 5), ylab = "mol-H/(kg-soln*d)",
+     legendposition = "bottomright", cumulative = TRUE, newdevice = FALSE)
```

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

```
> ylim <- range(output$DSApH, output$DSAstoichpH, output$pH)
> plot(output$DSApH, ylim = ylim, type = "l", xlab = "time/d",
+     ylab = "pH (free scale)")
> par(new = TRUE)
> plot(output$DSApH, ylim = ylim, type = "l", col = "red", xlab = "",
+     ylab = "")
> par(new = TRUE)
> plot(output$DSAstoichpH, ylim = ylim, type = "l", col = "blue",
+     xlab = "", ylab = "")
```

3.3.2.2 In three separate models

3.3.2.2.1 The implicit pH modelling approach

A list of parameters

```
> parameters <- list(S = 35, Tc = 15, SumCO2_t0 = 0.002, TA_t0 = 0.0022,
+     kc = 0.5, kp = 1e-06, n = 2, modeltime = 20, outputsteps = 100)
```

The model function

```
> boxmodel <- function(timestep, currentstate, parameters) {
+     with(as.list(c(currentstate, parameters)), {
+         ae <- aquaenv(Tc = Tc, S = S, 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 <<- as.data.frame(vode(initialstate, times, boxmodel,
+         parameters, hmax = 1))
+ })
```

Visualizing the output

```
> what <- c("SumCO2", "TA", "Rc", "Rp", "omega_calcite", "pH")
> plot(aquaenv(ae = output, from.data.frame = TRUE), xval = output$time,
+     xlab = "", mfrow = c(2, 3), size = c(12, 5), what = what,
+     newdevice = FALSE)
```

3.3.2.2.2 The explicit pH modelling approach

A list of parameters

```
> parameters <- list(S = 35, Tc = 15, SumCO2_t0 = 0.002, TA_t0 = 0.0022,
+     kc = 0.5, kp = 1e-06, n = 2, modeltime = 20, outputsteps = 100)
```

The model function

```
> boxmodel <- function(timestep, currentstate, parameters) {
+     with(as.list(c(currentstate, parameters)), {
+         ae <- aquaenv(Tc = Tc, S = S, 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(Tc = Tc, S = S, 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))
+ })
```

Visualizing the output

```
> what <- c("SumCO2", "TA", "Rc", "Rp", "omega_calcite", "pH",
+     "dHRc", "dHRp")
> plot(aquaenv(ae = output, from.data.frame = TRUE), xval = output$time,
+     xlab = "time/d", mfrow = c(3, 3), size = c(15, 10), what = what,
+     newdevice = FALSE)
```

Cumulatively plotting the influences of the two processes on the pH

```
> what <- c("dHRc", "dHRp")
> plot(aquaenv(ae = output, from.data.frame = TRUE), xval = output$time,
+     xlab = "time/d", what = what, ylab = "mol-H/(kg-soln*d)",
+     legendposition = "topright", cumulative = TRUE, newdevice = FALSE)
```

3.3.2.2.3 The fractional stoichiometric approach A list of parameters

```
> parameters <- list(S = 35, Tc = 15, SumCO2_t0 = 0.002, TA_t0 = 0.0022,
+     kc = 0.5, kp = 1e-06, n = 2, modeltime = 20, outputsteps = 100)
```

The model function

```
> boxmodel <- function(timestep, currentstate, parameters) {
+     with(as.list(c(currentstate, parameters)), {
+         ae <- aquaenv(Tc = Tc, S = S, 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(Tc = Tc, S = S, 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))
+ })
```

Visualizing the output

```
> what <- c("SumCO2", "TA", "Rc", "Rp", "omega_calcite", "pH",
+     "dHRc", "dHRp", "rhoc", "rhop")
> plot(aquaenv(ae = output, from.data.frame = TRUE), xval = output$time,
+     xlab = "time/d", mfrow = c(3, 4), size = c(15, 10), what = what,
+     newdevice = FALSE)
```

3.4 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

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 this elements is a suitable `xval` for plotting an *aquaenv* object generated by `titration`.

3.4.1 Titration with HCl

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

An object of type 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(Tc = 15, S = 35, SumCO2 = 0.0035, SumNH4 = 2e-05,
+     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 sample 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

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

Then, 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", "pCO2")
> plot(ae, xval = ae$delta_mass_titrant, xlab = "HCl solution added [kg]",
+     what = what, size = c(12, 8), mfrow = c(4, 4), newdevice = FALSE)
```

titrant concentration offset, or the moles of added titrant

```
> plot(ae, xval = ae$delta_conc_titrant, xlab = "[HCl] offset added [mol/kg-soln]",
+     what = what, size = c(12, 8), mfrow = c(12, 8))
> plot(ae, xval = ae$delta_moles_titrant, xlab = "HCl added [mol]",
+     what = what, size = c(12, 8), mfrow = c(12, 8))
```

However, it is also possible to plot this selection of elements against the calculated free scale pH

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

As mentioned earlier, the function `plot.aquaenv` offers the possibility of creating bjerrum plots from objects obtained with `titration`. The simplest way to do that is (remember the `newdevice=FALSE` is just needed to produce plots that are nicely woven into this vignette)

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

Or just select a few elements

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

Again, the plotting style can be customized

```
> plot(ae, what = what, bjerrum = TRUE, lwd = 4, palette = c("cyan",
+     "magenta", "yellow"), bg = "gray", legendinset = 0.1, legendposition = "topleft",
+     newdevice = FALSE)
```

However, 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, newdevice = FALSE)
```

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)]), newdevice = FALSE)
```

3.4.2 Titration with NaOH

Similar to the titration with HCl, also a titration with NaOH can be simulated

```
> ae <- titration(aquaenv(Tc = 15, S = 35, SumCO2 = 0.0035, SumNH4 = 2e-05,
+     pH = 2), mass_sample = 0.01, mass_titrant = 0.02, conc_titrant = 0.01,
+     S_titrant = 0.5, steps = 50, type = "NaOH")
```

Plottinge 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", "pCO2")
> 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, newdevice = FALSE)
```

```
> 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)]), newdevice = FALSE)
```

3.4.3 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. 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(Tc = 15, S = 35, SumCO2 = 0.0035, SumNH4 = 2e-05,
+     pH = 11.3), mass_sample = 100, mass_titrant = 0.5, conc_titrant = 3,
+     S_titrant = 0.5, steps = 100)
```

Plotting everything

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

Plotting selectively and with different elements for `xval`

```
> what <- c("TA", "pH", "CO2", "HCO3", "CO3", "BOH3", "BOH4", "OH",
+     "NH4", "NH3", "H2SO4", "HSO4", "SO4", "HF", "F", "pCO2")
> 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

```
> 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

```
> 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)]), newdevice = FALSE)
```

3.5 Calculating information from titration curves: the function `TAfit`

3.5.1 A little theory

While titrating a sample of natural seawater with HCl there 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(Tc = 15, S = 35, SumCO2 = 0.0035, SumNH4 = 2e-05,
+     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$delta_mass_titrant, xlab = "HCl solution added [kg]",
+     what = "pH", xlim = c(0, 0.015), newdevice = FALSE)
> par(new = TRUE)
> plot(ae$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$pH)), type = "l",
+     col = "green", xlim = c(0, 0.015), ylab = "", xlab = "",
+     yaxt = "n")
> abline(h = 0, col = "green")
> abline(v = ae$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, Anderson, Hassellov, Hulth, and Olsson 1997; Anderson, Turner, Wedborg, and Dyrssen 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$delta_mass_titrant, xlab = "HCl solution added [kg]",
+     what = "pH", xlim = c(0, 0.015), newdevice = FALSE)
> prot1 <- c()
> for (i in 1:length(ae$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 actually 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 not that here the value of the concentration of total alkalinity and total carbonate does not equal their total amount)

```
> ae <- titration(aquaenv(Tc = 15, S = 35, SumCO2 = 0.0035, SumNH4 = 2e-05,
+     pH = 11.3), mass_sample = 100, mass_titrant = 0.5, conc_titrant = 3,
+     S_titrant = 0.5, steps = 100)
> plot(ae, xval = ae$delta_mass_titrant, xlab = "HCl solution added [kg]",
+     what = "pH", xlim = c(0, 0.5), newdevice = FALSE)
> prot1 <- c()
> for (i in 1:length(ae$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 $[\sum CO_2]$ 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 $[\sum CO_2]$ of a sample by non linear curve fitting.

3.5.2 Determining [TA] and $[\sum CO_2]$ by non linear curve fitting

3.5.2.1 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(Tc = 15, S = 35, 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 SumCO2 (SumNH4, SumH2S, SumH3PO4, SumSiOH4, SumHNO3, SumHNO2, SumBOH3, SumH2SO4, SumHF) 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
```

Thus, we see that `TAfit` calculates the correct SumCO2 and TA values.

`TAfit` can also take E (V) values as input variables, so we generate E values using E0=0.4 V and the Nernst equation. However, to do so we first need to convert our pH curve to the seawater pH scale. According to (DOE 1994, p.7, ch.4, sop.3), the Nernst equation relates E to the total proton concentration, but, if fluoride is present, its effect (as proton donor/acceptor) is also measured. Hence, we use the seawater scale here.

```
> swstitcurve <- convert(titcurve[, 2], "pHscale", "free2sws",
+     Tc = 15, S = 35)
> Etitcurve <- cbind(titcurve[, 1], (0.4 - ((Constants$R/10) *
+     initial_ae$Tk/Constants$F) * log(10^-swstitcurve)))
```

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.

```
> fit2 <- TAfit(initial_ae, Etitcurve, conc_titrant = 0.01, mass_sample = 0.01,
+     S_titrant = 0.5, Evals = TRUE, verbose = 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
> initial_ae$K_CO2
```

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
```

Furthermore, TA, SumCO2, K_CO2 and E0 can be fitted at the same time.

```
> Etitcurve <- cbind(titcurve[, 1], (0.4 - ((Constants$R/10) *
+     initial_ae$Tk/Constants$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`

```
> 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)
> fit7 <- TAfit(initial_ae, noisetitcurve, conc_titrant = 0.01,
+     mass_sample = 0.01, seawater_titrant = TRUE, verbose = TRUE)
> fit7
```

The flag `verbose=TRUE` 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 = "")
```

3.5.2.2 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_dickson`. Following quantities are given

```
> conc_titrant <- 0.3
> mass_sample <- 0.2
> S_titrant <- 14.835
> SumBOH3 <- 0.00042
> SumH2SO4 <- 0.02824
> SumHF <- 7e-05
```

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_dickson`, the mass of titrant is given in g which needs to be converted to kg

```
> sam <- cbind(sample_dickson[, 1]/1000, sample_dickson[, 2])
```

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(Tc = 25, S = 35, SumBOH3 = SumBOH3,
+     SumH2SO4 = SumH2SO4, SumHF = SumHF), sam, conc_titrant, mass_sample,
+     S_titrant = S_titrant, debug = TRUE)
> dicksonfit
```

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

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

Let us calculate a theoretical titration without salinity correction

```
> dicksontitration1 <- titration(aquaenv(Tc = 25, S = 35, SumCO2 = 0.0022,
+     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(Tc = 25, S = 35, SumCO2 = 0.0022,
+     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$delta_mass_titrant,
+     what = "pH", xlim = c(0, 0.0025), ylim = c(3, 8.2), newdevice = FALSE,
+     col = "red", xlab = "delta mass titrant")
> par(new = TRUE)
> plot(dicksontitration2, xval = dicksontitration2$delta_mass_titrant,
+     what = "pH", xlim = c(0, 0.0025), ylim = c(3, 8.2), newdevice = FALSE,
+     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])
```

3.5.2.2.2 Does fitting K_CO2 as well improve the fit?

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

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(Tc = 25, S = 35, SumCO2 = 0.0022,
+     SumBOH3 = SumBOH3, SumH2SO4 = SumH2SO4, SumHF = SumHF, TA = 0.00245,
+     k_w = 4.32e-14, k_co2 = 1e-06, k_hco3 = 8.2e-10, k_boh3 = 1.78e-09,
+     k_hso4 = (1/12.3), k_hf = (1/408)), 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-06,
+     k_hco3 = 8.2e-10, k_boh3 = 1.78e-09, k_hso4 = (1/12.3), k_hf = (1/408))
> plot(dicksontitration3, xval = dicksontitration3$delta_mass_titrant,
+     what = "pH", xlim = c(0, 0.0025), ylim = c(3, 8.2), newdevice = FALSE,
```

```
+     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(Tc = 25, S = 35, SumBOH3 = SumBOH3,
+     SumH2SO4 = SumH2SO4, SumHF = SumHF, k_w = 4.32e-14, k_co2 = 1e-06,
+     k_hco3 = 8.2e-10, k_boh3 = 1.78e-09, k_hso4 = (1/12.3), k_hf = (1/408)),
+     sam, conc_titrant, mass_sample, S_titrant = S_titrant, debug = TRUE,
+     k_w = 4.32e-14, k_co2 = 1e-06, k_hco3 = 8.2e-10, k_boh3 = 1.78e-09,
+     k_hso4 = (1/12.3), k_hf = (1/408))
> dicksonfit3
```

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

3.5.2.3 Real data

3.5.2.3.1 `sample1`

The dataset `sample1` is a titration curve for TA determination, obtained in the lab of the NIOO-CEME, Yerseke, The Netherlands.

Metadata of the sample which is:

x-value	volume of titrant added (in ml) with a Metrohm Omega 665 Dosimat
y-value	pH measured on the NBS scale with a Metrohm Omega 714 pH Meter (Buffer pH 4: citric acid, sodium hydroxide, chlor hydrogen; Buffer pH 7: di-sodium hydrogen phosphate, potassium dihydrogen phosphate)
Tc	23.5 °C
S	34
volume of the sample	10 ml
titrant type	HCl
titrant concrntration	0.01 N = 0.01 M

We assume that the salinity of the titrant solution is 0.5 (the salinity of a solution with a ionic strength of 0.01 mol/kg-solution (0.01N) according to: I = (19.924 S) / (1000 - 1.005

S)). Furthermore, the the x values have to be converted to kg titrant, assuming the density of the titrant solution is aequivalent to the density of seawater with S=0.5, and last but not least 1 ml = 0.000001 m3.

```
> sample <- cbind((sample1[, 1] * 1e-06 * aquaenv(Tc = 23.5, S = 0.5)$density),
+     sample1[, 2])
```

We reduce the amount of datapoints: for runtime reasons

```
> sample <- sample[seq(1, length(sample[, 1]), 3), ]
```

We convert the concentration of the titrant from molarity (mol/l) to mol/kg-soln

```
> conc_titrant <- convert(0.01, "conc", "molar2molin", Tc = 23.5,
+     S = 0.5)
```

We calculate the mass of the sample in kg (and 10 ml = 0.00001 m3)

```
> mass_sample <- 1e-05 * aquaenv(Tc = 23.5, S = 34)$density
```

No the fitting procedure can be performed. Note that the datapoints are not equally spaced and the pH scale of the measured pH values is the NBS scale.

```
> sample1_fit <- TAfit(aquaenv(Tc = 23.5, S = 34), sample, conc_titrant,
+     mass_sample, S_titrant = 0.5, equalspaced = FALSE, pHscale = "nbs",
+     debug = TRUE, verbose = TRUE)
> sample1_fit
```

Note that AquaEnv uses the Davies eqation to calculate the activity coefficient for the proton upon conversion from/to NBS scale. The Davies equation is only valid up to a ionic strength (I) of 0.5 (Zeebe and Wolf-Gladrow 2001), that is around S=24.48. To accurately determine [TA] for samples with higher salinity using the function `TAfit`, one needs to provide either pH values on the free, total or seawater scale or E (V) values.

The [TA] value obtained with the Peene-Method was 0.0058740 mol/l, i.e.

```
> convert(0.005874, "conc", "molar2molin", Tc = 23.5, S = 34)
```

the difference in umol/kg-soln is

```
> (0.005724596 - sample1_fit$TA) * 1e+06
```

and the SumCO2 value was 0.0522 mgC/l, i.e.

```
> convert((0.0522/12), "conc", "molar2molin", Tc = 23.5, S = 34)
```

the difference in umol/kg-soln is

```
> (0.004239359 - sample1_fit$SumCO2) * 1e+06
```

Note the deviation of the measured curve from the simulated curve, especially at high pH's (and high salinities) as the titration procceeds, salinity drops due to dilution with non saline titrant (HCl solution) and the agreement between the curves gets better

```
> plot(sample[, 1], sample[, 2], xlim = c(0, 0.0066), ylim = c(3,
+     9), type = "l", xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = c(0, 0.0066), ylim = c(3,
+     9), type = "l", col = "red")
```

this shows that NBS pH values with initial salinities of more than 25 are not excellent input values for `TAfit`.

3.5.2.3.2 `sample1` to `sample3`

The datasets `sample2` and `sample3` are similar to `sample1` and their metadata can be found in their helpfiles.

Preparing the input for `TAfit`

```
> sam1 <- cbind((sample1[, 1] * 1e-06 * aquaenv(Tc = 23.5, S = 0.5)$density),
+     sample1[, 2])
> sam2 <- cbind((sample2[, 1] * 1e-06 * aquaenv(Tc = 22.9, S = 0.5)$density),
+     sample2[, 2])
> sam3 <- cbind((sample3[, 1] * 1e-06 * aquaenv(Tc = 23.3, S = 0.5)$density),
+     sample3[, 2])
> conc_titrant1 <- convert(0.01, "conc", "molar2molin", Tc = 23.5,
+     S = 0.5)
> conc_titrant2 <- convert(0.01, "conc", "molar2molin", Tc = 22.9,
+     S = 0.5)
> conc_titrant3 <- convert(0.01, "conc", "molar2molin", Tc = 23.3,
+     S = 0.5)
> mass_sample1 <- 1e-05 * aquaenv(Tc = 23.5, S = 34)$density
> mass_sample2 <- 1e-05 * aquaenv(Tc = 22.9, S = 33.4)$density
> mass_sample3 <- 1e-05 * aquaenv(Tc = 23.3, S = 33.8)$density
```

Performing the fitting

```
> sample1_fit <- TAfit(aquaenv(Tc = 23.5, S = 34), sam1, conc_titrant1,
+     mass_sample1, S_titrant = 0.5, equalspaced = FALSE, pHscale = "nbs",
+     debug = TRUE, verbose = TRUE)
```

```
> tit1 <- tit
> calc1 <- calc
> sample2_fit <- TAfit(aquaenv(Tc = 22.9, S = 33.4), sam2, conc_titrant2,
+     mass_sample2, S_titrant = 0.5, equalspaced = FALSE, pHscale = "nbs",
+     debug = TRUE, verbose = TRUE)
> tit2 <- tit
> calc2 <- calc
> sample3_fit <- TAfit(aquaenv(Tc = 23.3, S = 33.8), sam3, conc_titrant3,
+     mass_sample3, S_titrant = 0.5, equalspaced = FALSE, pHscale = "nbs",
+     debug = TRUE, verbose = TRUE)
> tit3 <- tit
> calc3 <- calc
```

Visualizing the final fits

```
> par(mfrow = c(1, 3))
> plot(sam1[, 1], sam1[, 2], xlim = c(0, 0.0066), ylim = c(3, 9),
+     type = "l", xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit1$delta_mass_titrant, calc1, xlim = c(0, 0.0066), ylim = c(3,
+     9), type = "l", col = "red")
> plot(sam2[, 1], sam2[, 2], xlim = c(0, 0.0066), ylim = c(3, 9),
+     type = "l", xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit2$delta_mass_titrant, calc2, xlim = c(0, 0.0066), ylim = c(3,
+     9), type = "l", col = "red")
> plot(sam3[, 1], sam3[, 2], xlim = c(0, 0.0066), ylim = c(3, 9),
+     type = "l", xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit3$delta_mass_titrant, calc3, xlim = c(0, 0.0066), ylim = c(3,
+     9), type = "l", col = "red")
```

Comparing the resulting values to the values obtained with the graphical method (J. Peene)

```
> TA1peene <- convert(0.005874, "conc", "molar2molin", Tc = 23.5,
+     S = 34)
> TA2peene <- convert(0.005856, "conc", "molar2molin", Tc = 22.9,
+     S = 33.4)
> TA3peene <- convert(0.005794, "conc", "molar2molin", Tc = 23.3,
+     S = 33.8)
> SumCO21peene <- convert(0.0522/12, "conc", "molar2molin", Tc = 23.5,
+     S = 34)
> SumCO22peene <- convert(0.043488/12, "conc", "molar2molin", Tc = 22.9,
+     S = 33.4)
> SumCO23peene <- convert(0.05376/12, "conc", "molar2molin", Tc = 23.3,
+     S = 33.8)
> c(TA1peene, sample1_fit$TA)
```

```
> c(TA2peene, sample2_fit$TA)
> c(TA3peene, sample3_fit$TA)
> c(SumCO21peene, sample1_fit$SumCO2)
> c(SumCO22peene, sample2_fit$SumCO2)
> c(SumCO23peene, sample3_fit$SumCO2)
```

3.5.2.3.3 `sample1` to `sample3`: fitting K_CO2 as well

Performing the fitting. Note that, with fitting K_CO2 as well, it might be necessary to choose a small initial stepwidth-factor and/or bring the parameter `TASumCO2guess` close to expected values for [TA] and $[\sum CO_2]$, otherwise the optimization routine might create an error, e.g. by "trying" negative K_CO2 values.

```
> sample1_fit <- TAfit(aquaenv(Tc = 23.5, S = 34), sam1, conc_titrant1,
+     mass_sample1, S_titrant = 0.5, equalspaced = FALSE, pHscale = "nbs",
+     debug = TRUE, K_CO2fit = TRUE, nlscontrol = nls.lm.control(factor = 0.1),
+     verbose = TRUE)
> tit1 <- tit
> calc1 <- calc
> sample2_fit <- TAfit(aquaenv(Tc = 22.9, S = 33.4), sam2, conc_titrant2,
+     mass_sample2, S_titrant = 0.5, equalspaced = FALSE, pHscale = "nbs",
+     debug = TRUE, K_CO2fit = TRUE, TASumCO2guess = 0.006, nlscontrol = nls.lm.control(fa
+     verbose = TRUE)
> tit2 <- tit
> calc2 <- calc
> sample3_fit <- TAfit(aquaenv(Tc = 23.3, S = 33.8), sam3, conc_titrant3,
+     mass_sample3, S_titrant = 0.5, equalspaced = FALSE, pHscale = "nbs",
+     debug = TRUE, K_CO2fit = TRUE, nlscontrol = nls.lm.control(factor = 0.1),
+     verbose = TRUE)
> tit3 <- tit
> calc3 <- calc
```

Visualizing the resulting fits

```
> par(mfrow = c(1, 3))
> plot(sam1[, 1], sam1[, 2], xlim = c(0, 0.0066), ylim = c(3, 9),
+     type = "l", xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit1$delta_mass_titrant, calc1, xlim = c(0, 0.0066), ylim = c(3,
+     9), type = "l", col = "red")
> plot(sam2[, 1], sam2[, 2], xlim = c(0, 0.0066), ylim = c(3, 9),
+     type = "l", xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit2$delta_mass_titrant, calc2, xlim = c(0, 0.0066), ylim = c(3,
+     9), type = "l", col = "red")
> plot(sam3[, 1], sam3[, 2], xlim = c(0, 0.0066), ylim = c(3, 9),
```

```
+     type = "l", xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit3$delta_mass_titrant, calc3, xlim = c(0, 0.0066), ylim = c(3,
+     9), type = "l", col = "red")
```

Comparing the resulting values to the values obtained with the graphical method (J. Peene)

```
> TA1peene <- convert(0.005874, "conc", "molar2molin", Tc = 23.5,
+     S = 34)
> TA2peene <- convert(0.005856, "conc", "molar2molin", Tc = 22.9,
+     S = 33.4)
> TA3peene <- convert(0.005794, "conc", "molar2molin", Tc = 23.3,
+     S = 33.8)
> SumCO21peene <- convert(0.0522/12, "conc", "molar2molin", Tc = 23.5,
+     S = 34)
> SumCO22peene <- convert(0.043488/12, "conc", "molar2molin", Tc = 22.9,
+     S = 33.4)
> SumCO23peene <- convert(0.05376/12, "conc", "molar2molin", Tc = 23.3,
+     S = 33.8)
> c(TA1peene, sample1_fit$TA)
> c(TA2peene, sample2_fit$TA)
> c(TA3peene, sample3_fit$TA)
> c(SumCO21peene, sample1_fit$SumCO2)
> c(SumCO22peene, sample2_fit$SumCO2)
> c(SumCO23peene, sample3_fit$SumCO2)
```

3.5.2.3.4 `sample1` to `sample3`: fitting ≈ 10 datapoints

To test if the fit wil be significantly worse if the amount of datapoints in the titration curve is reduced, we reduce the amount of datapoints in the curve to around 10.

```
> sam1 <- cbind((sample1[, 1] * 1e-06 * aquaenv(Tc = 23.5, S = 0.5)$density),
+     sample1[, 2])[seq(1, length(sample1[, 1]), 10), ]
> sam2 <- cbind((sample2[, 1] * 1e-06 * aquaenv(Tc = 22.9, S = 0.5)$density),
+     sample2[, 2])[seq(1, length(sample2[, 1]), 10), ]
> sam3 <- cbind((sample3[, 1] * 1e-06 * aquaenv(Tc = 23.3, S = 0.5)$density),
+     sample3[, 2])[seq(1, length(sample3[, 1]), 10), ]
```

Performing the fitting

```
> sample1_fit <- TAfit(aquaenv(Tc = 23.5, S = 34), sam1, conc_titrant1,
+     mass_sample1, S_titrant = 0.5, equalspaced = FALSE, pHscale = "nbs",
+     debug = TRUE, verbose = TRUE)
> tit1 <- tit
> calc1 <- calc
> sample2_fit <- TAfit(aquaenv(Tc = 22.9, S = 33.4), sam2, conc_titrant2,
```

```
+     mass_sample2, S_titrant = 0.5, equalspaced = FALSE, pHscale = "nbs",
+     debug = TRUE, verbose = TRUE)
> tit2 <- tit
> calc2 <- calc
> sample3_fit <- TAfit(aquaenv(Tc = 23.3, S = 33.8), sam3, conc_titrant3,
+     mass_sample3, S_titrant = 0.5, equalspaced = FALSE, pHscale = "nbs",
+     debug = TRUE, verbose = TRUE)
> tit3 <- tit
> calc3 <- calc
```

Visualizing the resulting fit

```
> par(mfrow = c(1, 3))
> plot(sam1[, 1], sam1[, 2], xlim = c(0, 0.0066), ylim = c(3, 9),
+     type = "l", xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit1$delta_mass_titrant, calc1, xlim = c(0, 0.0066), ylim = c(3,
+     9), type = "l", col = "red")
> plot(sam2[, 1], sam2[, 2], xlim = c(0, 0.0066), ylim = c(3, 9),
+     type = "l", xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit2$delta_mass_titrant, calc2, xlim = c(0, 0.0066), ylim = c(3,
+     9), type = "l", col = "red")
> plot(sam3[, 1], sam3[, 2], xlim = c(0, 0.0066), ylim = c(3, 9),
+     type = "l", xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit3$delta_mass_titrant, calc3, xlim = c(0, 0.0066), ylim = c(3,
+     9), type = "l", col = "red")
```

Comparing the resulting values to the values obtained with the graphical method (J. Peene)

```
> TA1peene <- convert(0.005874, "conc", "molar2molin", Tc = 23.5,
+     S = 34)
> TA2peene <- convert(0.005856, "conc", "molar2molin", Tc = 22.9,
+     S = 33.4)
> TA3peene <- convert(0.005794, "conc", "molar2molin", Tc = 23.3,
+     S = 33.8)
> SumCO21peene <- convert(0.0522/12, "conc", "molar2molin", Tc = 23.5,
+     S = 34)
> SumCO22peene <- convert(0.043488/12, "conc", "molar2molin", Tc = 22.9,
+     S = 33.4)
> SumCO23peene <- convert(0.05376/12, "conc", "molar2molin", Tc = 23.3,
+     S = 33.8)
> c(TA1peene, sample1_fit$TA)
> c(TA2peene, sample2_fit$TA)
> c(TA3peene, sample3_fit$TA)
> c(SumCO21peene, sample1_fit$SumCO2)
```

```
> c(SumCO22peene, sample2_fit$SumCO2)
> c(SumCO23peene, sample3_fit$SumCO2)
```

That means using just ≈ 10 datapoints is feasible, as the fits do not get worse significantly

3.5.2.3.5 `sample4_1` to `sample4_2`: standard seawater

`sample4_1` to `sample4_4` are titration curves of reference material (standard seawater) "Batch 82" (A. Dickson) for TA determination, obtained in the lab of the NIOO-CEME, Yerseke, The Netherlands. Metadata can again be obtained from the helpfiles of the datasets.

We assume the salinity of the titrant solution is 0.5 (the salinity of a solution with a ionic strength of 0.01 mol/kg-solution (0.01N) according to: I = (19.924 S) / (1000 - 1.005 S). We convert the x values to kg titrant, and assume the density of the titrant solution is aequivalent to the density of seawater with S=0.5. Finally, 1 ml = 0.000001 m^3.

```
> sam4_1 <- cbind((sample4_1[, 1] * 1e-06 * aquaenv(Tc = 21, S = 0.5)$density),
+     sample4_1[, 2])
```

We convert the concentration of the titrant from molarity (mol/l) to mol/kg-soln

```
> conc_titrant <- convert(0.01, "conc", "molar2molin", Tc = 21,
+     S = 0.5)
```

We calculate the mass of the sample in kg (10 ml = 0.00001 m3)

```
> mass_sample <- 1e-05 * aquaenv(Tc = 21, S = 35.356)$density
```

We perform the fitting procedure: note that the datapoints are NOT equally spaced and the pH scale of the measured pH values is the NBS scale, (`debug=TRUE` toggles the debug mode: the last simulated titration tit and the converted pH profile calc are made global variables for investigation and plotting)

```
> sample4_1_fit <- TAfit(aquaenv(Tc = 21, S = 34.356, SumH3PO4 = 3e-07,
+     SumSiOH4 = 2e-06, SumHNO2 = 1e-08, SumHNO3 = 8.3e-07), sam4_1,
+     conc_titrant, mass_sample, S_titrant = 0.5, equalspaced = FALSE,
+     pHscale = "nbs", debug = TRUE)
> sample4_1_fit
```

The fit can be visualized

```
> xlim <- range(sam4_1[, 1], tit$delta_mass_titrant)
> ylim <- range(sam4_1[, 2], calc)
> plot(sam4_1[, 1], sam4_1[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

(Note that the fit is very good on the steep slopes and not so good inbetween. This is because the sample titration curve is not equally spaced. There are smaller titration steps in the steep parts, resulting in more points there. Since **AquaEnv** minimizes the sum of squares of all points without weigthing the points according to their density in a certain region of the curve, the regions with higher point density have implicitly a higher weight during the fitting procedure.)

and the results can be compared to results obtained with the graphical method (J.Peene)

```
> peeneTA <- convert(0.002418, "conc", "molar2molin", Tc = 21,
+     S = 35.356)
> peeneSumCO2 <- convert(0.025872/12, "conc", "molar2molin", Tc = 21,
+     S = 35.356)
> c(peeneTA, sample4_1_fit$TA)
> c(peeneSumCO2, sample4_1_fit$SumCO2)
```

`sample4_2` is a replica titration of `sample4_1`.

Preparations

```
> sam4_2 <- cbind((sample4_2[, 1] * 1e-06 * aquaenv(Tc = 22.8,
+     S = 0.5)$density), sample4_2[, 2])
> conc_titrant <- convert(0.01, "conc", "molar2molin", Tc = 22.8,
+     S = 0.5)
> mass_sample <- 1e-05 * aquaenv(Tc = 22.8, S = 35.356)$density
```

Fitting

```
> sample4_2_fit <- TAfit(aquaenv(Tc = 22.8, S = 34.356, SumH3PO4 = 3e-07,
+     SumSiOH4 = 2e-06, SumHNO2 = 1e-08, SumHNO3 = 8.3e-07), sam4_2,
+     conc_titrant, mass_sample, S_titrant = 0.5, equalspaced = FALSE,
+     pHscale = "nbs", debug = TRUE)
> sample4_2_fit
```

Visualisation

```
> xlim <- range(sam4_2[, 1], tit$delta_mass_titrant)
> ylim <- range(sam4_2[, 2], calc)
> plot(sam4_2[, 1], sam4_2[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

Comparison to graphical method

```
> peeneTA <- convert(0.00239, "conc", "molar2molin", Tc = 22.8,
+     S = 35.356)
```

```
> peeneSumCO2 <- convert(0.02532/12, "conc", "molar2molin", Tc = 22.8,
+     S = 35.356)
> peeneCalcSumCO2 <- convert(0.02625722/12, "conc", "molar2molin",
+     Tc = 22.8, S = 35.356)
> c(peeneTA, sample4_2_fit$TA)
> c(peeneSumCO2, peeneCalcSumCO2, sample4_2_fit$SumCO2)
```

3.5.2.3.6 `sample4_3` to `sample4_4`: E curves

The datasets `sample4_3` and `sample4_4` have been obtained by manually titrating samples of the standard seawater mentioned above and recording the "raw" potential (E) values coming from the electrode.

Preparations

```
> sam4_3 <- cbind((sample4_3[, 1] * 1e-06 * aquaenv(Tc = 21, S = 0.5)$density),
+     sample4_3[, 2]/1000)
> conc_titrant <- convert(0.01, "conc", "molar2molin", Tc = 21,
+     S = 0.5)
> mass_sample <- 1e-05 * aquaenv(Tc = 21, S = 0)$density
```

Fitting

```
> sample4_3_fit <- TAfit(aquaenv(Tc = 21, S = 34.356, SumH3PO4 = 3e-07,
+     SumSiOH4 = 2e-06, SumHNO2 = 1e-08, SumHNO3 = 8.3e-07), sam4_3,
+     conc_titrant, mass_sample, S_titrant = 0.5, Evals = TRUE,
+     verbose = TRUE, electrode_polarity = "neg", debug = TRUE)
> sample4_3_fit
```

Visualizing

```
> xlim <- range(sam4_3[, 1], tit$delta_mass_titrant)
> ylim <- range(sam4_3[, 2], calc)
> plot(sam4_3[, 1], sam4_3[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

Fitting K_CO2 as well

```
> sample4_3_fit_1 <- TAfit(aquaenv(Tc = 21, S = 34.356, SumH3PO4 = 3e-07,
+     SumSiOH4 = 2e-06, SumHNO2 = 1e-08, SumHNO3 = 8.3e-07), sam4_3,
+     conc_titrant, mass_sample, S_titrant = 0.5, Evals = TRUE,
+     verbose = TRUE, electrode_polarity = "neg", K_CO2fit = TRUE,
+     debug = TRUE)
> sample4_3_fit_1
```

Visualizing

```
> xlim <- range(sam4_3[, 1], tit$delta_mass_titrant)
> ylim <- range(sam4_3[, 2], calc)
> plot(sam4_3[, 1], sam4_3[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

Fitting with the K values from Dickson (1981)

```
> sample4_3_fit_2 <- TAfit(aquaenv(Tc = 21, S = 34.356, SumH3PO4 = 3e-07,
+     SumSiOH4 = 2e-06, SumHNO2 = 1e-08, SumHNO3 = 8.3e-07, k_w = 4.32e-14,
+     k_co2 = 1e-06, k_hco3 = 8.2e-10, k_boh3 = 1.78e-09, k_hso4 = (1/12.3),
+     k_hf = (1/408)), sam4_3, conc_titrant, mass_sample, S_titrant = 0.5,
+     Evals = TRUE, verbose = TRUE, electrode_polarity = "neg",
+     k_w = 4.32e-14, k_co2 = 1e-06, k_hco3 = 8.2e-10, k_boh3 = 1.78e-09,
+     k_hso4 = (1/12.3), k_hf = (1/408), debug = TRUE)
> sample4_3_fit_2
```

Visualizing

```
> xlim <- range(sam4_3[, 1], tit$delta_mass_titrant)
> ylim <- range(sam4_3[, 2], calc)
> plot(sam4_3[, 1], sam4_3[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

Fitting with titrant salinity assumed equal to sample salinity

```
> sample4_3_fit_3 <- TAfit(aquaenv(Tc = 21, S = 34.356, SumH3PO4 = 3e-07,
+     SumSiOH4 = 2e-06, SumHNO2 = 1e-08, SumHNO3 = 8.3e-07), sam4_3,
+     conc_titrant, mass_sample, Evals = TRUE, verbose = TRUE,
+     electrode_polarity = "neg", debug = TRUE)
> sample4_3_fit_3
```

Visualizing

```
> xlim <- range(sam4_3[, 1], tit$delta_mass_titrant)
> ylim <- range(sam4_3[, 2], calc)
> plot(sam4_3[, 1], sam4_3[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

Fitting with titrant concentration assumed to be exactly 0.01 mol/kg-soln

```
> sample4_3_fit_4 <- TAfit(aquaenv(Tc = 21, S = 34.356, SumH3PO4 = 3e-07,
+     SumSiOH4 = 2e-06, SumHNO2 = 1e-08, SumHNO3 = 8.3e-07), sam4_3,
+     conc_titrant = 0.01, mass_sample, S_titrant = 0.5, Evals = TRUE,
+     verbose = TRUE, electrode_polarity = "neg", debug = TRUE)
> sample4_3_fit_4
```

Visualizing

```
> xlim <- range(sam4_3[, 1], tit$delta_mass_titrant)
> ylim <- range(sam4_3[, 2], calc)
> plot(sam4_3[, 1], sam4_3[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

Preparations for `sample4_4`

```
> sam4_4 <- cbind((sample4_4[, 1] * 1e-06 * aquaenv(Tc = 21, S = 0.5)$density),
+     sample4_4[, 2]/1000)
> conc_titrant <- convert(0.01, "conc", "molar2molin", Tc = 21,
+     S = 0.5)
> mass_sample <- 1e-05 * aquaenv(Tc = 21, S = 0)$density
```

Fitting

```
> sample4_4_fit <- TAfit(aquaenv(Tc = 22.4, S = 34.356, SumH3PO4 = 3e-07,
+     SumSiOH4 = 2e-06, SumHNO2 = 1e-08, SumHNO3 = 8.3e-07), sam4_4,
+     conc_titrant, mass_sample, S_titrant = 0.5, Evals = TRUE,
+     verbose = TRUE, electrode_polarity = "neg", debug = TRUE)
> sample4_4_fit
```

Visualizing

```
> xlim <- range(sam4_4[, 1], tit$delta_mass_titrant)
> ylim <- range(sam4_4[, 2], calc)
> plot(sam4_4[, 1], sam4_4[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

Fitting with concentrations of summed quantities calculated from salinity

```
> sample4_4_fit_1 <- TAfit(aquaenv(Tc = 22.4, S = 34.356), sam4_4,
+     conc_titrant, mass_sample, S_titrant = 0.5, Evals = TRUE,
+     verbose = TRUE, electrode_polarity = "neg", debug = TRUE)
> sample4_4_fit_1
```

Visualizing

```
> xlim <- range(sam4_4[, 1], tit$delta_mass_titrant)
> ylim <- range(sam4_4[, 2], calc)
> plot(sam4_4[, 1], sam4_4[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

Fitting K_CO2 as well

```
> sample4_4_fit_2 <- TAfit(aquaenv(Tc = 22.4, S = 34.356, SumH3PO4 = 3e-07,
+     SumSiOH4 = 2e-06, SumHNO2 = 1e-08, SumHNO3 = 8.3e-07), sam4_4,
+     conc_titrant, mass_sample, S_titrant = 0.5, Evals = TRUE,
+     verbose = TRUE, electrode_polarity = "neg", K_CO2fit = TRUE,
+     debug = TRUE)
> sample4_4_fit_2
```

Visualizing

```
> xlim <- range(sam4_4[, 1], tit$delta_mass_titrant)
> ylim <- range(sam4_4[, 2], calc)
> plot(sam4_4[, 1], sam4_4[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

3.5.2.3.7 Some rudimentary statistics on `sample4_1` to `sample4_4`

(Beware, these statistics are just "back of the envelope" calculations, since not all the measurements are independent and for different quantities not the same number of measurements have been carried out.)

The graphical method

```
> peeneTA <- c(2353.904, 2326.871)
> peeneDIC <- c(2098.849, 2054.267, 2130.305)
> paste(mean(peeneTA), "+/-", sd(peeneTA))
> paste(mean(peeneDIC), "+/-", sd(peeneDIC))
```

pH curve fitting

```
> pHTA <- c(sample4_1_fit$TA, sample4_2_fit$TA) * 1e+06
> pHDIC <- c(sample4_1_fit$SumCO2, sample4_2_fit$SumCO2) * 1e+06
> paste(mean(pHTA), "+/-", sd(pHTA))
> paste(mean(pHDIC), "+/-", sd(pHDIC))
```

E curve fitting

```
> ETA <- c(sample4_3_fit$TA, sample4_3_fit_1$TA, sample4_3_fit_2$TA,
+     sample4_3_fit_3$TA, sample4_3_fit_4$TA, sample4_4_fit$TA,
+     sample4_4_fit_1$TA, sample4_4_fit_2$TA) * 1e+06
> EDIC <- c(sample4_3_fit$SumCO2, sample4_3_fit_1$SumCO2, sample4_3_fit_2$SumCO2,
+     sample4_3_fit_3$SumCO2, sample4_3_fit_4$SumCO2, sample4_4_fit$SumCO2,
+     sample4_4_fit_1$SumCO2, sample4_4_fit_2$SumCO2) * 1e+06
> paste(mean(ETA), "+/-", sd(ETA))
> paste(mean(EDIC), "+/-", sd(EDIC))
```

pH and E curve fitting combined

```
> totalTA <- c(pHTA, ETA)
> totalDIC <- c(pHDIC, EDIC)
> paste(mean(totalTA), "+/-", sd(totalTA))
> paste(mean(totalDIC), "+/-", sd(totalDIC))
```

3.5.2.3.8 `sample5_1` **to** `sample5_2`**: freshwater samples** `sample5_1` to `sample5_3` are titration curves of freshwater samples obtained in the lab of the NIOO-CEME, Yerseke, The Netherlands. `sample5_1` and `sample5_2` are pH curves.

Preparations

```
> sam5_1 <- cbind((sample5_1[, 1] * 1e-06 * aquaenv(Tc = 21, S = 0.5)$density),
+     sample5_1[, 2])
> conc_titrant <- convert(0.01, "conc", "molar2molin", Tc = 21,
+     S = 0.5)
> mass_sample <- 1e-05 * aquaenv(Tc = 17.5, S = 0)$density
```

Fitting

```
> sample5_1_fit <- TAfit(aquaenv(Tc = 17.5, S = 0), sam5_1, conc_titrant,
+     mass_sample, S_titrant = 0.5, equalspaced = FALSE, pHscale = "nbs",
+     debug = TRUE, verbose = TRUE)
> sample5_1_fit
```

Visualizing

```
> xlim <- range(sam5_1[, 1], tit$delta_mass_titrant)
> ylim <- range(sam5_1[, 2], calc)
> plot(sam5_1[, 1], sam5_1[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

Comparing to graphical method

```
> peeneTA <- convert(0.003314, "conc", "molar2molin", Tc = 17.5,
+     S = 0)
> c(peeneTA, sample5_1_fit$TA)
```

Preparations for `sample5_2`

```
> sam5_2 <- cbind((sample5_2[, 1] * 1e-06 * aquaenv(Tc = 21, S = 0.5)$density),
+     sample5_2[, 2])
> conc_titrant <- convert(0.01, "conc", "molar2molin", Tc = 21,
+     S = 0.5)
> mass_sample <- 1e-05 * aquaenv(Tc = 18.3, S = 0)$density
```

Fitting

```
> sample5_2_fit <- TAfit(aquaenv(Tc = 18.3, S = 0), sam5_2, conc_titrant,
+     mass_sample, S_titrant = 0.5, equalspaced = FALSE, pHscale = "nbs",
+     debug = TRUE, verbose = TRUE)
> sample5_2_fit
```

Visualizing

```
> xlim <- range(sam5_2[, 1], tit$delta_mass_titrant)
> ylim <- range(sam5_2[, 2], calc)
> plot(sam5_2[, 1], sam5_2[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

Comparing to the graphical method

```
> peeneTA <- convert(0.00329, "conc", "molar2molin", Tc = 18.3,
+     S = 0)
> peeneSumCO2 <- convert(0.039647/12, "conc", "molar2molin", Tc = 18.3,
+     S = 0)
> c(peeneTA, sample5_2_fit$TA)
> c(peeneSumCO2, sample5_2_fit$SumCO2)
```

Fitting K_CO2 as well

```
> sample5_2_fit_1 <- TAfit(aquaenv(Tc = 18.3, S = 0), sam5_2, conc_titrant,
+     mass_sample, S_titrant = 0.5, equalspaced = FALSE, pHscale = "nbs",
+     verbose = TRUE, K_CO2fit = TRUE, debug = TRUE)
> sample5_2_fit_1
```

Visualizing

```
> xlim <- range(sam5_2[, 1], tit$delta_mass_titrant)
> ylim <- range(sam5_2[, 2], calc)
> plot(sam5_2[, 1], sam5_2[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

3.5.2.3.9 `sample5_3`**: E curve** The dataset `sample5_3` has been obtained by manually titrating samples of the freshwater sample mentioned above and recording the "raw" potential (E) values coming from the electrode.

Preparations

```
> sam5_3 <- cbind((sample5_3[, 1] * 1e-06 * aquaenv(Tc = 21, S = 0.5)$density),
+     sample5_3[, 2]/1000)
> conc_titrant <- convert(0.01, "conc", "molar2molin", Tc = 21,
+     S = 0.5)
> mass_sample <- 1e-05 * aquaenv(Tc = 21, S = 0)$density
```

Fitting

```
> sample5_3_fit <- TAfit(aquaenv(Tc = 21, S = 0.1), sam5_3, conc_titrant,
+     mass_sample, S_titrant = 0.5, Evals = TRUE, verbose = TRUE,
+     electrode_polarity = "neg", debug = TRUE)
> sample5_3_fit
```

Visualizing

```
> xlim <- range(sam5_3[, 1], tit$delta_mass_titrant)
> ylim <- range(sam5_3[, 2], calc)
> plot(sam5_3[, 1], sam5_3[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

Fitting K_CO2 as well

```
> sample5_3_fit_1 <- TAfit(aquaenv(Tc = 21, S = 0.1), sam5_3, conc_titrant,
+     mass_sample, S_titrant = 0.5, Evals = TRUE, verbose = TRUE,
+     electrode_polarity = "neg", K_CO2fit = TRUE, debug = TRUE)
> sample5_3_fit_1
```

Visualizing

```
> xlim <- range(sam5_3[, 1], tit$delta_mass_titrant)
> ylim <- range(sam5_3[, 2], calc)
```

```
> plot(sam5_3[, 1], sam5_3[, 2], xlim = xlim, ylim = ylim, type = "l",
+     xlab = "", ylab = "")
> par(new = TRUE)
> plot(tit$delta_mass_titrant, calc, xlim = xlim, ylim = ylim,
+     type = "l", col = "red")
```

3.5.2.3.10 Some rudimentary statistics on `sample5_1` to `sample5_4`

(Beware, these statistics are just "back of the envelope" calculations, since not all the measurements are independent and for different quantities not the same number of measurements have been carried out.)

The graphical method

```
> peeneTA <- c(3310.61, 3286.456)
> peeneDIC <- c(3300.358)
> paste(mean(peeneTA), "+/-", sd(peeneTA))
> paste(mean(peeneDIC), "+/-", sd(peeneDIC))
```

pH curve fitting

```
> pHTA <- c(sample5_1_fit$TA, sample5_2_fit$TA, sample5_2_fit_1$TA) *
+     1e+06
> pHDIC <- c(sample5_1_fit$SumCO2, sample5_2_fit$SumCO2, sample5_2_fit_1$SumCO2) *
+     1e+06
> paste(mean(pHTA), "+/-", sd(pHTA))
> paste(mean(pHDIC), "+/-", sd(pHDIC))
```

E curve fitting

```
> ETA <- c(sample5_3_fit$TA, sample5_3_fit_1$TA) * 1e+06
> EDIC <- c(sample5_3_fit$SumCO2, sample5_3_fit_1$SumCO2) * 1e+06
> paste(mean(ETA), "+/-", sd(ETA))
> paste(mean(EDIC), "+/-", sd(EDIC))
```

pH and E curve fitting combined

```
> totalTA <- c(pHTA, ETA)
> totalDIC <- c(pHDIC, EDIC)
> paste(mean(totalTA), "+/-", sd(totalTA))
> paste(mean(totalDIC), "+/-", sd(totalDIC))
```

4 Extending **AquaEnv**

It is very 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 type 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"

The function is defined as

```
> simpletitration <- function(aquaenv, volume, amount, steps, type) {
+     directionTAchange <- switch(type, HCl = -1, NaOH = +1)
+     TAconcchangeperstep <- convert(((amount/steps)/volume), "conc",
+         "molar2molin", aquaenv$Tc, aquaenv$S)
+     aquaenvtemp <- aquaenv
+     for (i in 1:steps) {
+         TA <- aquaenvtemp$TA + (directionTAchange * TAconcchangeperstep)
+         aquaenvtemp <- cloneaquaenv(aquaenvtemp, TA = TA)
+         aquaenv <- merge(aquaenv, aquaenvtemp)
+     }
+     aquaenv[["DeltaCTitrant"]] <- convert((amount/volume)/steps *
+         (1:(steps + 1)), "conc", "molar2molin", aquaenv$Tc, aquaenv$S)
+     return(aquaenv)
+ }
```

and can be used to create a bjerrum plot

```
> ae <- simpletitration(aquaenv(Tc = 15, S = 35, SumCO2 = 0.0035,
+     SumNH4 = 2e-05, 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)]), newdevice = FALSE)
```

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

`ae`	an object of type 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"

defined as

```
> simpleTAfit <- function(ae, pHmeasurements, volume, amount, TAguess = 0.0025,
+     type = "HCl") {
+     ae$Na <- NULL
+     residuals <- function(state) {
+         ae$SumCO2 <- state[[1]]
+         pHcalc <- simpletitration(aquaenv(ae = ae, TA = state[[2]]),
+             volume = volume, amount = amount, steps = (length(pHmeasurements) -
+                 1), type = type)$pH
+         residuals <- pHmeasurements - pHcalc
+         return(residuals)
+     }
+     require(minpack.lm)
+     out <- nls.lm(fn = residuals, par = c(TAguess, TAguess))
+     result <- list(out$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)
+ }
```

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

```
> pHmeasurements <- ae$pH
> fit <- simpleTAfit(aquaenv(Tc = 15, S = 35, SumNH4 = 2e-05),
+     pHmeasurements, volume = 100, amount = 1.5)
> fit
```

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)
Emerson2008	Emerson and Hedges (2008)
Follows2006	Follows, Ito, and Dutkiewicz (2006)
Hofmann2008	Hofmann *et al.* (2008b)
Lewis1998	Lewis and Wallace (1998)
Millero1981	Millero and Poisson (1981)
Millero1988	Millero *et al.* (1988)
Millero1995	Millero (1995)
Millero1995a	Millero *et al.* (1995)
Mucci1983	Mucci (1983)
Riordan2005	**?**
Roy1993b	Roy *et al.* (1993b)
Sundquist1979	Sundquist, Plummer, and Wigley (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**
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).
SumBr, 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).
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).
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 pure water (Millero 1979) is cited and a function for seawater is derived. In Millero (1995) it is stated that for S<5 the fresh water formula of (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 Tk. 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 Millero (1979) is restated wrongly 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.)
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, Roy, Lawson, Vogel, Moore, Davis, and Millero (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	(Zeebe and Wolf-Gladrow 2001, p.73)
dTAdKdKdS, dTAdKdKdT, dTAdKdKdd, dTAdKdKd-SumH2SO4, dTAdKdKd-SumHF	Hofmann *et al.* (2008a)

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 Tc from the above references are pressure corrected using the given depth d and the calculated hydrostatic pressure hydroP 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*. VCH Weinheim, 2nd edition.

Boudreau BP (1996). "A method-of-lines code for carbon and nutrient diagenesis in aquatic sediments." *Computers & Geosciences*, **22**(5), 479–496. URL <GotoISI>://A1996UQ20500003.

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. URL <GotoISI>://A1981LY51900007.

Dickson AG (1984). "Ph Scales and Proton-Transfer Reactions in Saline Media Such as Sea-Water." *Geochimica Et Cosmochimica Acta*, **48**(11), 2299–2308. URL <GotoISI>://A1984TT08000013.

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 HSO4⁻ in Synthetic Sea-Water from 273.15-K to 318.15-K." *Journal of Chemical Thermodynamics*, **22**(2), 113–127. URL <GotoISI>://A1990CY50200001.

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. URL <GotoISI>://A1990DK56300004.

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. URL <GotoISI>://A1987L675400008.

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. URL <GotoISI>://A1979GJ26800001.

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. URL <GotoISI>://A1979GJ26800002.

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.

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. URL <GotoISI>://000235695500003.

Gran G (1952). "Determination of the Equivalence Point in Potentiometric Titrations .2." *Analyst*, **77**(920), 661–671. URL <GotoISI>://A1952UG08300014.

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. URL <GotoISI>://A1973Q206600014.

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. URL <GotoISI>://000072760700007.

Hofmann A, Meysman F, Soetaert K, Middelburg J (2008a). "Factors governing the pH in a heterotrophic, turbid, tidal estuary." *Biogeosciences Discussion*.

Hofmann AF, Meysman FJR, Soetaert K, Middelburg JJ (2008b). "A step-by-step procedure for pH model construction in aquatic systems." *Biogeosciences J1 - BG*, **5**(1), 227–251. URL http://www.biogeosciences.net/5/227/2008/L1-http://www.biogeosciences.net/5/227/2008/bg-5-227-2008.pdf.

Hofmann AF, Middelburg J, Soetaert K, Wolf-Gladrow DA, Meysman F (2008c). "pH modelling in aquatic environments: from an alkalinity-based to a proton-based view." *in preparation*.

Lewis EL, Wallace DWR (1998). "Program Developed for CO_2 System Calculations."

Millero FJ (1979). "Thermodynamics of the Carbonate System in Seawater." *Geochimica Et Cosmochimica Acta*, **43**(10), 1651–1661. URL <GotoISI>://A1979HP85900006.

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. URL <GotoISI>://A1995QH99100003.

Millero FJ, Plese T, Fernandez M (1988). "The Dissociation of Hydrogen-Sulfide in Seawater." *Limnology and Oceanography*, **33**(2), 269–274. URL <GotoISI>://A1988N101700008.

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. URL <GotoISI>://A1981LY51900008.

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. URL <GotoISI>://A1995RV93400004.

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. URL <GotoISI>://A1983RF42100006.

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. URL <GotoISI>://000226779800009.

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. URL <GotoISI>://A1993MT06600012.

Roy RN, Roy LN, Vogel KM, PorterMoore 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*, **52**(2), 183–183. URL <GotoISI>://A1996UH75200007.

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

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. URL <GotoISI>://A1979GY02300024.

Weiss RF (1970). "Solubility of Nitrogen, Oxygen and Argon in Water and Seawater." *Deep-Sea Research*, **17**(4), 721–735. URL <GotoISI>://A1970H126100004.

Weiss RF (1974). "Carbon dioxide in water and seawater: the solubility of a non-ideal gas." *Marine Chemistry*, **2**, 203–215.

Williams DR (2004). "NASA Earth Fact Sheet." URL http://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html.

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. URL <GotoISI>://000183605000002.

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

Affiliation:

Andreas F. Hofmann
Centre for Estuarine and Marine Ecology (CEME)
Netherlands Institute of Ecology (NIOO)
4401 NT Yerseke, Netherlands E-mail: a.hofmann@nioo.knaw.nl
URL: http://www.nioo.knaw.nl/ppages/ahofmann