SensinSilico

Table of Content

Introduction

SensinSilico is a simulation software published together with the peer-reviewed article ***Timing Matters: The Overlooked Issue of Response Time Mismatch in pH-Dependent Analyte Sensing using Multiple Sensors*** in the peer-reviewed journal *Analyst*. It studies the error propagation when combining two (electrochemical) sensors with different response times. This document is meant as an installation and operation manual to use the software and explains the relevant application's functions.

Purpose and Scope

In nature, most biogeochemical processes are based on the interchange of weak acid-base equilibria. Yet, in some of these equilibria, it occurs that one species may be harmful or toxic to the environment while the other species is harmless or even essential to biogeochemical processes. To describe both species as well as the overall behaviour of the acid-base equilibrium, it has become common in environmental analysis to monitor the pH and one of the two parameters of the equilibrium. By obtaining the values of the two, it becomes possible to calculate the third parameter and subsequently the total parameter.

Two examples of acid-base equilibria that are critical to environmental processes are the ammonium/ammonia exchange, expressed as total ammonia-nitrogen (TAN)[1,2], and the sulfide equilibrium, which includes H_2S and HS^- as total dissolved sulfide (TDS)[3,4]:

TAN: $\mathbf{NH_3 + H^+ \rightarrow NH_4{}^+}$ and TDS: $\mathbf{H_2S \rightarrow HS^- + H^+}$

When two different sensors are required to determine a certain (total) parameter in a heterogeneous system as is done in the above-mentioned examples, sensor alignment and experiment design become critical. Especially in heterogeneous systems where parameters may fluctuate greatly within a few µm, it must be ensured that the individual sensors are positioned exactly at the same place. However not only the location is crucial, but also timing since parameters can vary strongly over time. Additionally, each sensor has an individual (delayed) response time. To derive individual parameters correctly from the measured ones, the timing between the different sensors must be coordinated to avoid error propagation and falsified results.

Since we know that reality is not always ideal and time pressure can lead to non-ideal measurement conditions, we have developed the software SensinSilico. The aim is to simulate the error propagation that can occur when calculating sum parameters from two individual sensors with different response times (e.g. one slow and one fast sensor). While all weak monovalent acid-base equilibria can be simulated in theory, TAN and TDS, as common examples, have been pre-programmed. The simulation assumes a target total parameter concentration and a set of user-defined pH step changes. The target concentration is then compared to the concentration calculated from the simulated data from the two individual sensors with different response times. The error propagation in the calculation procedure associated with the sensor response time mismatch can then be determined within the software.

Organization

In the next chapter, Background | Functions of the Application, we describe the theory required to understand the simulation, how to prepare the input parameters for each function and how to calculate the error propagation software. To illustrate the handling of the software, we have created a step-by-step guide for the TAN equilibrium. We recommend recapping this example to get a feel for how the software works. In the last chapter, **JupyterNotebook | Exploring Sensor Response Curves**, we refer to the Jupyter Notebook that can be used to explore different sensor response curves. If you have difficulties running the software or the Jupyter Notebook or encounter any bugs, we encourage reaching out to us (cf. **Point of Contact**).

Point of Contact

In case of encountering any bugs or if you have any further questions, please feel free to reach out to info@silvi-azieger.com.

Reference

(1) Water quality standards: Total Ammonia Nitrogen - Responsible Seafood Advocate. Global Seafood Alliance. https://www.globalseafood.org/advocate/water-quality-standards-total-ammonia-nitrogen/ (accessed 2023-06-13).

(2) Mook, W. T.; Chakrabarti, M. H.; Aroua, M. K.; Khan, G. M. A.; Ali, B. S.; Islam, M. S.; Abu Hassan, M. A. Removal of Total Ammonia Nitrogen (TAN), Nitrate and Total Organic Carbon (TOC) from Aquaculture Wastewater Using Electrochemical Technology: A Review. Desalination 2012, 285, 1–13. https://doi.org/10.1016/j.desal.2011.09.029.

(3) Steininger, F.; Koren, K.; Revsbech, N. P.; Marzocchi, U. Microsensor for Total Dissolved Sulfide (TDS). Chemosphere 2023, 323, 138229. https://doi.org/10.1016/j.chemosphere.2023.138229.

(4) Hydrogen sulfide toxic, but manageable - Responsible Seafood Advocate. Global Seafood Alliance. https://www.globalseafood.org/advocate/hydrogen-sulfide-toxic-but-manageable/ (accessed 2023-06-13).

Background | Application’s Functions

For the sake of illustration, we focus here on the total ammonia (TAN) example. For simulating total dissolved sulfide (TDS) or other weak monovalent acids, one only has to adjust the respective parameter values according to **Table 2-3.**

It is assumed that pH and one of the two species in the acid-base equilibrium will be monitored over time in this and any similar experiment. The respective other species are calculated based on the **Henderson-Hasselbalch** equilibrium for weak acid/base pairs and the total parameter is the sum of the two individual species at each time point.

For our demonstration, we used values for the sensor properties based on what we could find on sensor manufacturer’s websites:

TABLE 1 *Commercially available sensors for pH, NH_4^+, NH_3 and H_2S and their corresponding response times*

Analyte	Type	Response time /s	Supplier
pH	Optical	<60	Pyroscience[1]
	Optical	<120	Presens[2]
	Potentiometric	1-60	Krohne[3]
	Potentiometric	<10	Unisense[4]
	Potentiometric	<45	MetrOhm[5]
NH_4^+	ISE	<180	Hach[6]
NH_3	Severinghaus-type	<60	Fisher[7]
	Severinghaus-type	<600	MetrOhm[8]
H2S	Amperometric	<10	Unisense[9]
	Amperometric	<25	Sulfilogger[10]

Detailed Description of Functions

Since we focus on pH-dependent two-sensor systems, we start our simulation with the pH value and its fluctuation over time, which is set by the user. As a target concentration, we simulate a step function assuming an infinitive fast sensor response. For the example, we assume pH fluctuations of 4, 12.4, and 8.3 with a plateau time of 75 seconds. Furthermore, we assume that the system under investigation is based on a weak acid/base equilibrium, allowing us to use the **Henderson–Hasselbalch** equation to determine the analyte concentrations.

Henderson Hasselbalch

$$\mathrm{pH} = \mathrm{pK}_a + \log \frac{[Base]}{[Acid]} \quad \text{and} \quad [\mathrm{Total}] = [\mathrm{Base}] + [\mathrm{Acid}] \qquad \text{(1) and (2)}$$

Equations (1) and **(2)** describe the starting point for our simulation. For simulating the TAN equilibrium, the base and acid are NH_3 and NH_4^+. To determine the total parameter (TAN), the pH value as well as one of the analyte species is monitored. Hence, transforming **equation (2)** and combining it with **equation (1)** results in the general description of the base or acid species:

$$[\mathrm{Base}] = \frac{[\mathrm{Total}]}{1 + 10^{\mathrm{pH}-\mathrm{pK_a}}} \quad \text{and} \quad [\mathrm{Acid}] = \frac{[\mathrm{Total}]}{1 + 10^{\mathrm{pK_a}-\mathrm{pH}}} \tag{(3) and (4)}$$

And more specifically for the TAN simulation experiment:

$$[\mathrm{NH_3}] = \frac{[\mathrm{TAN}]}{1 + 10^{\mathrm{pH}-\mathrm{pK_a}}} \quad \text{and} \quad [\mathrm{NH_4}^+] = \frac{[\mathrm{TAN}]}{1 + 10^{\mathrm{pK_a}-\mathrm{pH}}}$$

For the simulation (cf. **Figure 1**), we assume a target TAN concentration of 100 mg/L and that the pH value and ammonia concentration are monitored over time. The ammonium concentration NH_4^+ and subsequently the TAN concentration will be calculated based on **equations (4)** and **(2)**. Then, we can compare how much the error propagation affects the calculated total parameter concentration when calculated from two individual sensors with different sensor response times.

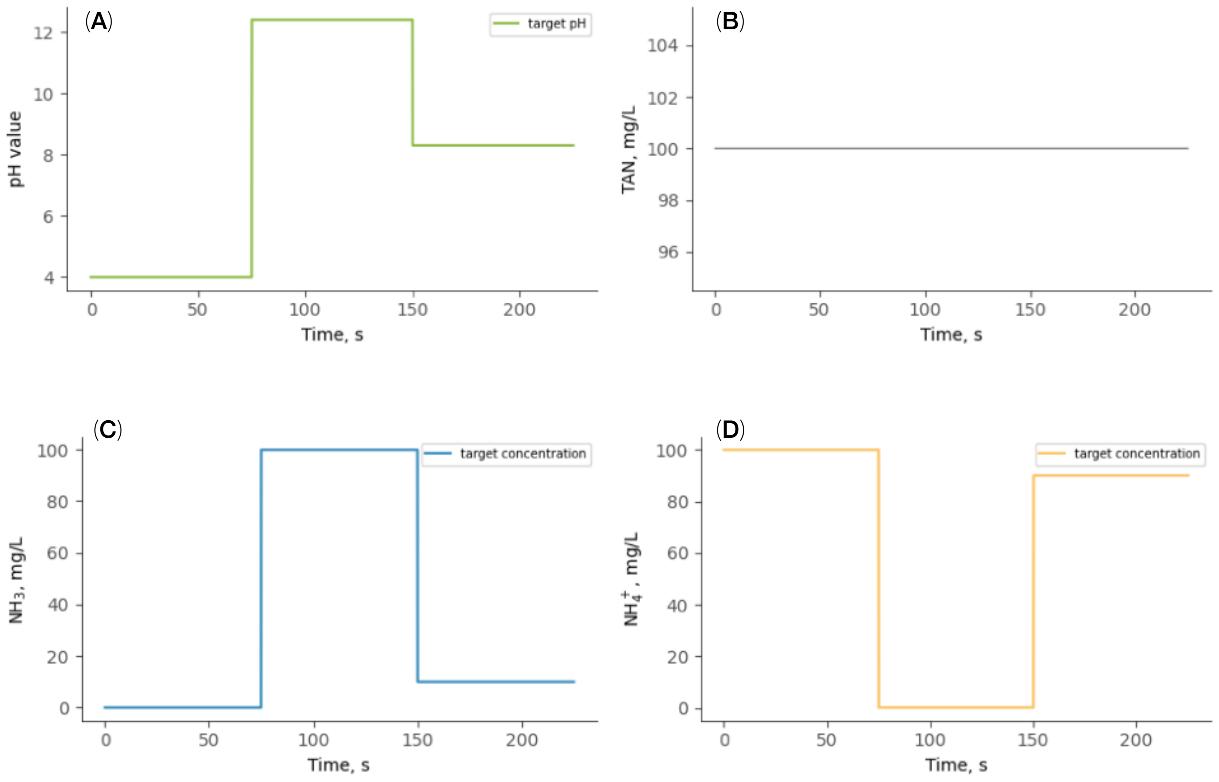


FIGURE 1 *User-defined pH values* (A) *and total parameter concentration* (B) *over time and resulting target concentration for the individual ammonia species* (C-D). *For the simulation of the target concentrations. An infinitely fast sensor response is assumed, resulting in the respective step function.*

However, sensors do not respond infinitely fast. In reality, sensors have a certain response time, usually defined as t_{90} time. To account for this delayed sensor response, we simulated a sigmoidal response pattern, which can be described by a **Gompertzian curve** for sigmoidal time series:

$$f(t) = a \cdot exp^{(-exp(b-c\cdot t))} + d \tag{5}$$

PARAMETERS

- *a* is the difference between the actual and the target concentration or pH value.
- *b* sets the displacement along the *x*-axis (translates the graph to the left or right). The sensor resolution feeds into this parameter with $b = log(log(1/\tau))$ and τ being the sensor resolution in seconds.
- *c* specifies the growth rate or *y*-scaling of the sigmoidal curve. The sensor resolution and response time both feed into this parameter with $1/t_{90} \cdot [b - log(log(1/(1-r)))]$, where t_{90} is the sensor response time, r is the sensor resolution, and b is the displacement factor along the *x*-axis as previously defined.
- *d* specifies the displacement along the *y*-axis. For our simulation, it sets the initial plateau from which the sigmoidal curve is simulated.

Applying the Gompertz curve to the target pH fluctuation, we can simulate the delayed response of the pH sensor depending on its specifications as shown in Figure 2.

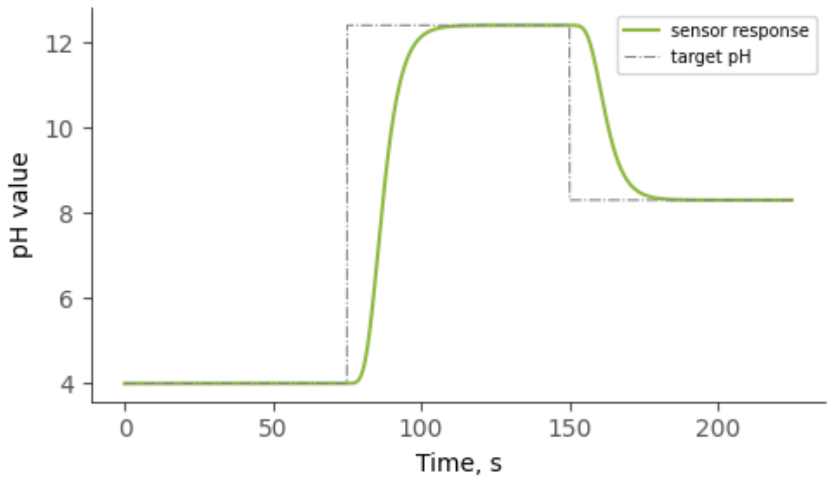


FIGURE 2 *Sensor response of an ideal and real pH sensor. The response of the target/ideal sensor response with an infinitely fast response time is shown as a grey dash-dotted line, while the response of the real sensor with a t_{90} time of 20 s is shown as a green solid curve.*

Based on the target pH value and total ammonia concentration, a sensor response for the measuring sensor including its sensor response is simulated (cf. Figure 3). In this example, the measuring sensor is an NH_3 sensor. The NH_4^+ concentration is then calculated at each time point based on the sensor response of the NH_3 sensor and the simulated reading of the pH sensor. The TAN concentration is then calculated as the sum of the NH_3 and NH_4^+ at each time point. The following plots show the resulting error propagation. It has to be stated that the same procedure applies when NH_4^+ is selected as the measuring sensor and NH_3 is calculated.

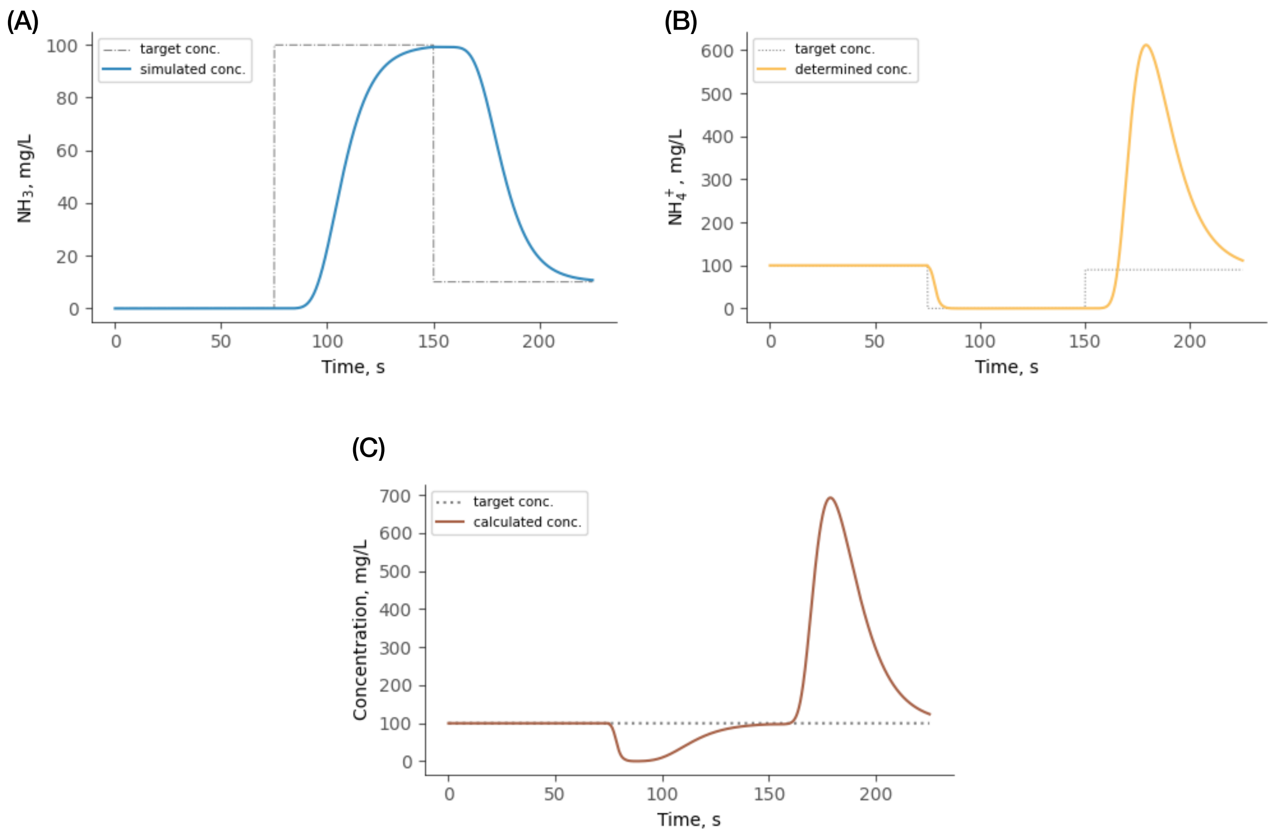


FIGURE 3 *Simulation of error propagation for the calculation of TAN. The* pH *(same as Fig. 2) and* NH_3 (A) *are monitored over time considering the respective sensor response times. The* t_{90} *response times of the real sensor* pH *and* NH_3 *are 20 s and 60 s, respectively. From these simulated values the* NH_4^+ *concentration* (B) *and further* TAN (C) *can be calculated.*

It should be mentioned that we also considered the **Nernst equation** to convert the sensor signal to the analyte concentration and vice versa.

$$E = E_0 - 2.303 \cdot R \cdot \frac{T + 273.15\,K}{n \cdot F} \cdot pH \qquad (6)$$

PARAMETERS

- *E_0* is the standard reduction potential in V
- *R* is the universal gas constant 8.314 in J/mol·K
- *F* is the Faraday constant or the charge of an electron 96485 in C/mol
- *n* is the number of electrons involved in the redox reaction
- *T* is the temperature in °C
- pH is the apparent pH value

Preparation of function inputs

All parameters required for the simulation are listed and functionally described below. Global parameters are fixed and cannot be changed in the software, while others are subject to analysis and can therefore be varied in the software interface.

Overview of global (fixed) parameters

TABLE 2 *Auxiliary parameters used for simulation.*

Parameter	Value	Description
tsteps	10^{-3}	Time steps for simulating the time series; in seconds
R	8.314	Universal gas constant; in J/mol*K
F	96485	Faraday constant; in C/mol
n	1	Number of electrons involved in the redox reaction
ph_res	10^{-3}	Resolution of the pH sensor
ph_E0	0.22	Standard potential of the reference electrode of the pH meter; in V
sens_res	10^{-9}	Resolution of the measuring sensor
conc_calib	1	Reaction quotient for calibrating the measuring sensor (here NH_3)
sens_E	0.09	Potential of the measuring sensor at a certain calibration concentration (*conc_calib*); in V

Overview of the sensor parameters used in the TAN example

TABLE 3 *Sensor specifications used for the simulation. Can be adjusted by the user on the initial page when starting the software.*

Parameter	Value	Description
Plateau time	75.0	Plateau time as the time a pH value is supposed to be stable; in seconds
pH value(s)	4, 12.4, 8.3	List of pH values given as float number, separated by a comma “,”
Response time	20, 50	Response time of the pH and measuring sensor; values are given as float numbers in seconds
c(TAN)	100	Total parameter concentration (here TAN) given as a float number in mg/L. It is possible to add a list of target concentrations separated by a comma “,”
pK_a	9.25	Acid dissociation constant as the equilibrium constant of the system to be studied. For TAN we assumed a pK_a of 9.25, while for TDS we suggest using a pKa of 7.

Results of functions (Simulation Output)

Simulation of the error propagation for the calculated analyte and the total parameter displayed in the response signal over time (and changing pH value).

- Response curves can be exported and saved as images (either directly by right-mouse click and selecting export or via the button “SAVE ALL”).
- Individual (measured and simulated) sensor responses can be exported as text files via “SAVE REPORT”.
- Error expressed as overall deviation from the target concentration of the total parameter can be calculated from the total parameter plot.

Calculating the overall error

An error is defined as the deviation between the measured and the target value. While one can calculate the difference between the two values at a certain time point, we were more interested in the overall error over time (cf. Figure 4). In our simulation, this equals the area between the sensor response and the target concentration of the total parameter.

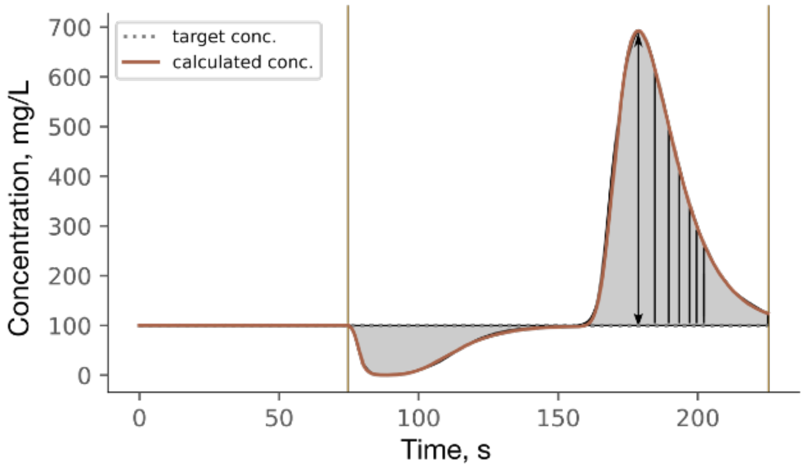


FIGURE 4 *Simulated sensor response of the total parameter (TAN) as a result of the error propagation of the measured and calculated (pH dependent) analyte. The deviation of the calculated concentration (solid brown line) to the target concentration (black, dotted line) at a certain time is indicated as a (vertical) black line. Hence, the total error within a given time range is given as the area underneath the response curve. The total error is marked as a grey area.*

To determine the error of the simulated measurement, we determine the integral of the sensor response compared to the integral of the target concentration within a user-defined time range. The integral of the total signal resulting from the individual sensor responses is determined using Simpson’s rule, a numerical integration of discrete but equally spaced sample points. Simpson’s rule is based upon a quadratic interpolation as follows:

$$\int_a^b f(x)dx \simeq \frac{1}{3} \cdot h[f(a) + 4 \cdot f(\frac{a+b}{2}) + f(b)] = \\ = \frac{b-a}{6}[f(a) + 4 \cdot f(\frac{a+b}{2}) + f(b))] \tag{7}$$

where $h = (b - a)/2$ is the step size.

To determine the integral of the target concentration, the area of the individual steps is determined as the sum of individual rectangles:

$$\int_a^b f(x)dx \simeq c_1(SUM) \cdot (t_1 - t_0) + c_2(SUM) \cdot (t_2 - t_1) + \ldots \tag{8}$$

where $c_i(SUM)$, with $i = 1,2,...$, is the user-defined target concentration of the total parameter and t_j with $j = 1,2,...$, are the individual integration limits sub-dividing the target concentration within the user-defined integration range into rectangles (cf. Figure 5).

The difference between the area of the simulated total concentration and the area of the target concentration equals the overall error.

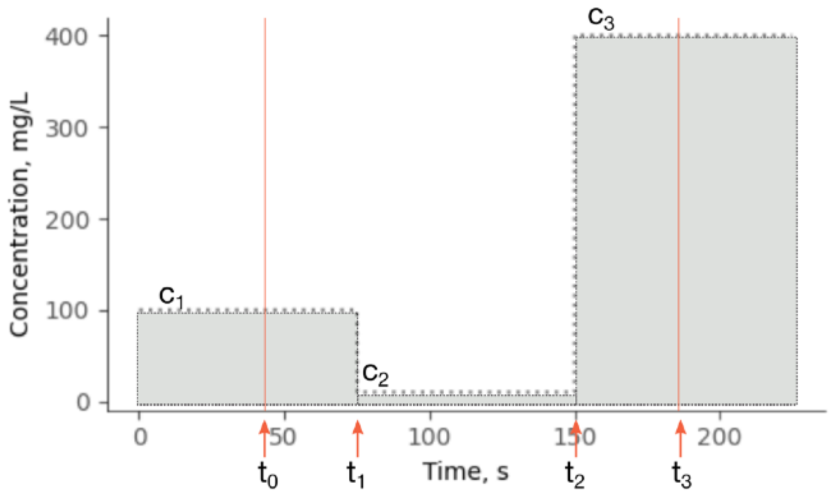


FIGURE 5 *Explanatory plot to illustrate the integration of the target concentration of the total parameter concentration. Since we assume infinitely fast responding sensors for the target concentrations, we also obtain a step function for the total parameter concentration. When the user selects integration limits at times* t_0 *and* t_3*, the total integral can be calculated as the sum of the area of each rectangle.*

References

(1) PyroScience GmbH | pH meters. pyroscience.com/en/products/ph (accessed 2023-06-12).

(2) Presens | Optical pH Sensors. https://www.presens.de/products/ph/sensors (accessed 2023-06-12).

(3) Krohne GmbH | pH sensors and measuring systems. https://krohne.com/en/products/process-analytics/analytical-sensors-measuring-systems/ph-sensors-measuring-systems (accessed 2023-06-12).

(4) Unisense | pH microelectrode for research applications. Unisense. https://unisense.com/products/ph-microelectrode/ (accessed 2023-06-12).

(5) Metrohm | 780/781 pH/ionmeter. https://www.metrohm.com/da_dk/products/ph-ion-measurement/Laboratory-ion-pH-meters.html (accessed 2023-06-12).

(6) Hach | A-ISE sc Low cost ISE Ammonium probe (immersion) with RFID, 10 m cable. https://ie.hach.com/a-ise-sc-low-cost-ise-ammonium-probe-immersion-with-rfid-10-m-cable/product?id=26371051351 (accessed 2023-06-12).

(7) Thermo Scientific Orion | High-Performance Ammonia Electrode - pH and Electrochemistry, Probes and Electrodes. https://www.fishersci.com/shop/products/orion-high-performance-ammonia-electrode/13643500 (accessed 2023-06-12).

(8) Metrohm | NH3-selective gas membrane electrode (low conc.). https://www.metrohm.com/en/products/6/0506/60506100.html (accessed 2023-06-12).

(9) Unisense | H2S Microsensor for hydrogen sulfide research. Unisense. https://unisense.com/products/h2s-microsensor/ (accessed 2023-06-12).

(10)SulfiLogger | H2S sensor. https://sulfilogger.com/sulfilogger-sensor/ (accessed 2023-06-12).

Operating Instructions (step-by-step-guide)

Where to download the software application and how to get started?

To run the software application, the source code must be downloaded from GitHub. It is publicly available via the following link: https://github.com/silviaelisabeth/SensorResponse-inSilico.

Download the application folder including the subfolder (***PICTURE***). To run the software application, open the file named SensinSilico.py with a Python editor or IDE of your choice and run the script. We suggest using PyCharm, VS Code or the Terminal/Command Prompt as we tested the software application with PyCharm and VS Code 1.74.2 (Universal) for software development and testing.

For the software application to run properly, leave all the files in the folder together without restructuring or removing any parts.

Requirements

When using the source code, python3 and the following Python packages are required for execution:

numpy (version 1.24.2), pandas (version 1.5.3), PyQt5 (version 5.15.9), pyqtgraph (version 0.13.3), scipy (version 1.10.1), and seaborn (version 0.12.2).

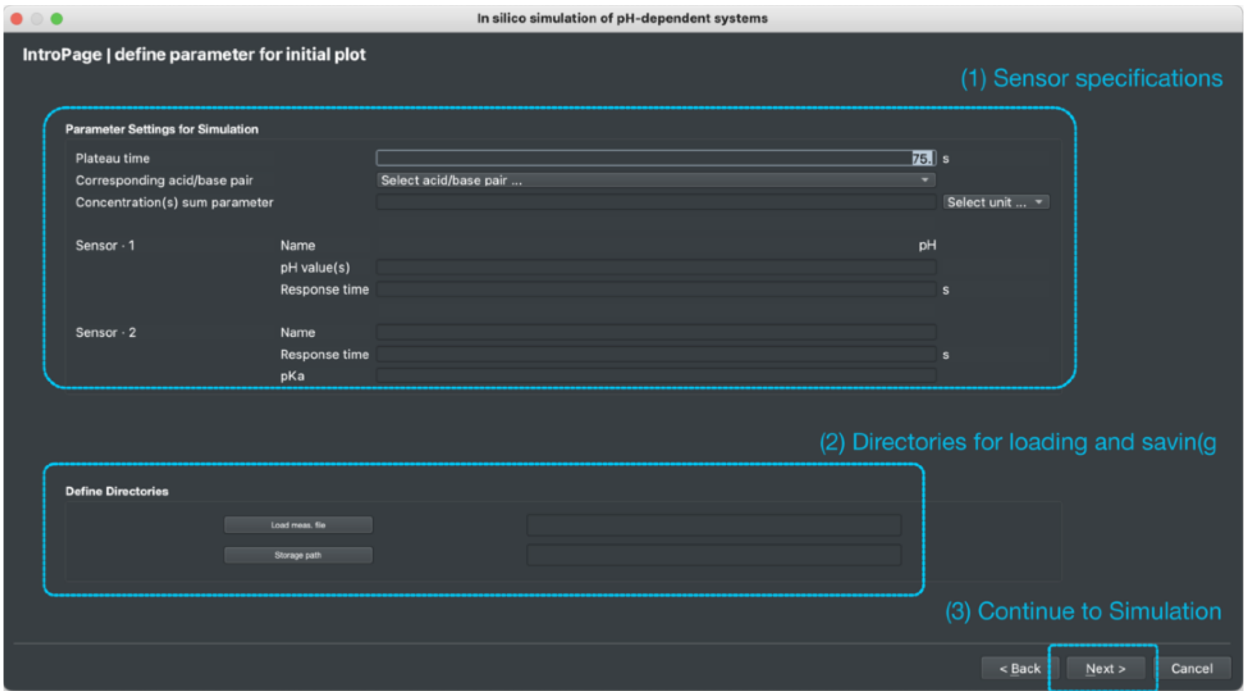
The package versions listed are the versions used when the software is released. Older versions might cause errors and bugs, whereas newer versions should not make much difference. The software has been tested for both MacOS (MacBook Pro – Apple M1, Version 13.4) and Windows (Version 10.0.19045). If you have difficulties running the software, please do not hesitate to reach out (cf. Points of Contact).

IntroPage | Define Parameter

Upon executing the script SensinSilico.py a separate window should open displaying the IntroPage of the software application (cf. Figure 6).

In the upper part of the IntroPage the global parameters for the simulation must be defined. Furthermore, the properties of the two single sensors have to be defined, where sensor 1 always defines the pH sensor. For the second sensor, either a sensor can be defined as part of the H_2S/HS^- or NH_3/ NH_4^+ equilibrium or a parameter of another weak monovalent acid/base equilibrium can be selected. The corresponding acid/base pair can be selected from the drop-down menu. In case the user wants to add another system, this can be specified and added to the drop-down menu by selecting "*Add other acid/base pair*". However, the user needs to define which parameter will be measured with the sensor. This is done by labelling sensor 2 with one of the corresponding acid/base pair names. For example, if you want to study the TAN equilibrium, select either NH_3 or NH_4^+ as your measurement sensor. Once you have specified the equilibrium, you need to add the characteristics, such as the expected sensor response time and the pK_a of your system. We suggest a pK_a of 9.25 for TAN, and 7.0 for TDS respectively.

In addition, a list of target pH values to be studied must be defined for the pH sensor. As response time, assume a common response time for commercially available pH sensors as presented in Table 1. To finish the preparation, add a concentration for the sum parameter or even a list of target concentrations, separated by a comma. From the drop-down menu, the user can select a preferred concentration unit which is either ppm, mg/L, or mol/L.


In silico simulation of pH-dependent systems
IntroPage | define parameter for initial plot
(1) Sensor specifications
Parameter Settings for Simulation
Corresponding acid/base pair
Select acid/base pair ...
Concentration(s) sum parameter
Select unit ...
Sensor - 1
Name
pH
pH value(s)
Response time
s
Sensor - 2
Name
Response time
s
pKa
(2) Directories for loading and savin(g
Define Directories
Load meas. file
Storage path
(3) Continue to Simulation
< Back
Next >
Cancel

Plateau time 75 s

FIGURE 6 *Front page of SensinSilico. Here, the initial sensor specifics* (1) *such as the response time of the* pH *sensor and the monitored analyte must be defined. Furthermore, specify the species that shall be monitored and set the* pK_a *value of the acid/base equilibrium. Optional, one can load previous simulations or specify the directory where to save the results* (2). *When all required parameters are defined, press NEXT* (3) *to continue to the simulation page.*

In case any previously simulated experiment shall be imported or any output shall be saved, define the directory using the respective buttons in the lower part (”LOAD MEAS. FILE” and “STORAGE PATH”). Pressing “NEXT” will lead you to the second page of the software wizard. In case it is not possible to access the second page, make sure that all required data is added/defined.

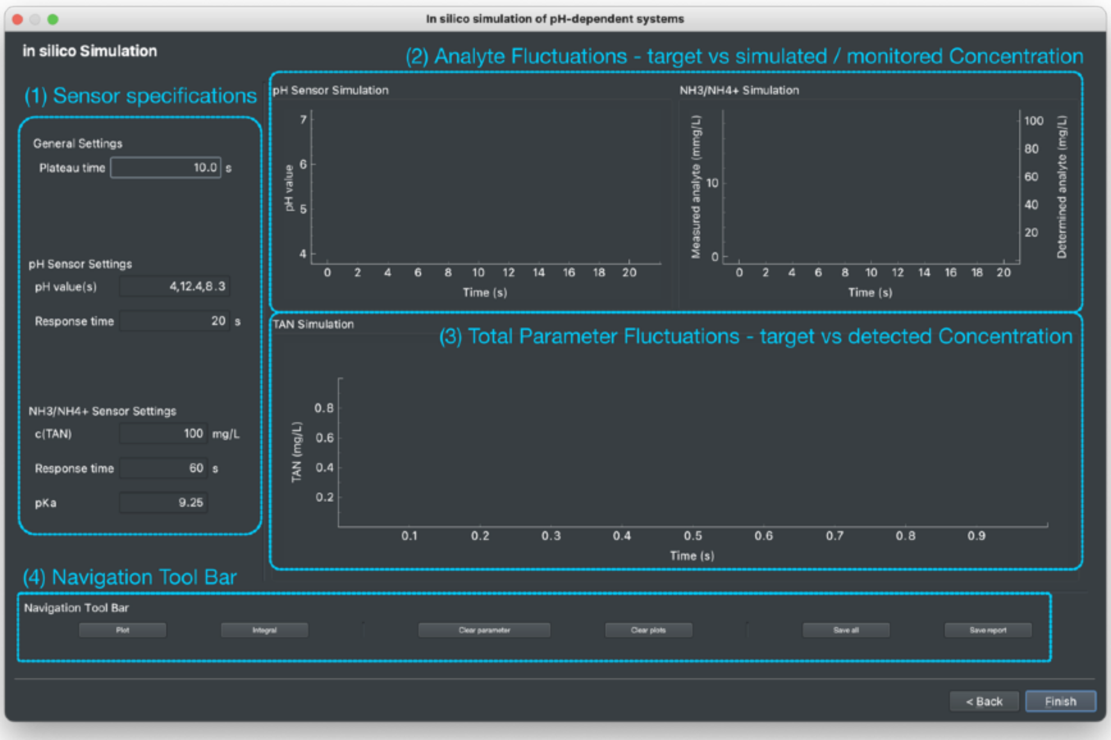
REQUIRED INPUT

Plateau time	Time period of how long one pH value or sum parameter concentration is assumed to be stable; time given as float number in seconds
Corresponding acid/ base pair	From the drop-down menu, select either one of the pre-defined two parameter systems or add another system that is based on a Henderson-Hasselbalch equilibrium.
Concentration(s) of sum parameter	Target concentration(s) of the total parameter to be simulated. Numbers given as float numbers separated by a comma. Select the preferred unit from the drop-down menu. The suer can choose between ppm, mg/L, or mol/L.
Sensor 1	It is assumed that the first sensor is always a pH sensor
▶ pH value(s)	Target pH values (between 1-14); values given as float numbers separated by a comma
▶ Response time	Response time of a pH sensor; here expressed as t_{90} in seconds
Sensor 2	Measuring sensor which usually has a different response time than the pH sensor
▶ Name	Specify which parameter of the corresponding acid/base pair was measured. When choosing from a pre-defined two-sensor system, select either NH_3 or NH_4^+ for a TAN simulation or H_2S or HS^- for a TDS simulation
▶ Response time	Response time of the measuring sensor expressed as t_{90} in seconds
▶ pK_a	Acid dissociation constant describing the equilibrium constant of the system to be studied

In the software, use points (”.”) for decimal separators.

Simulation Page

Generate the simulation



General Settings
Plateau time 10.0 s

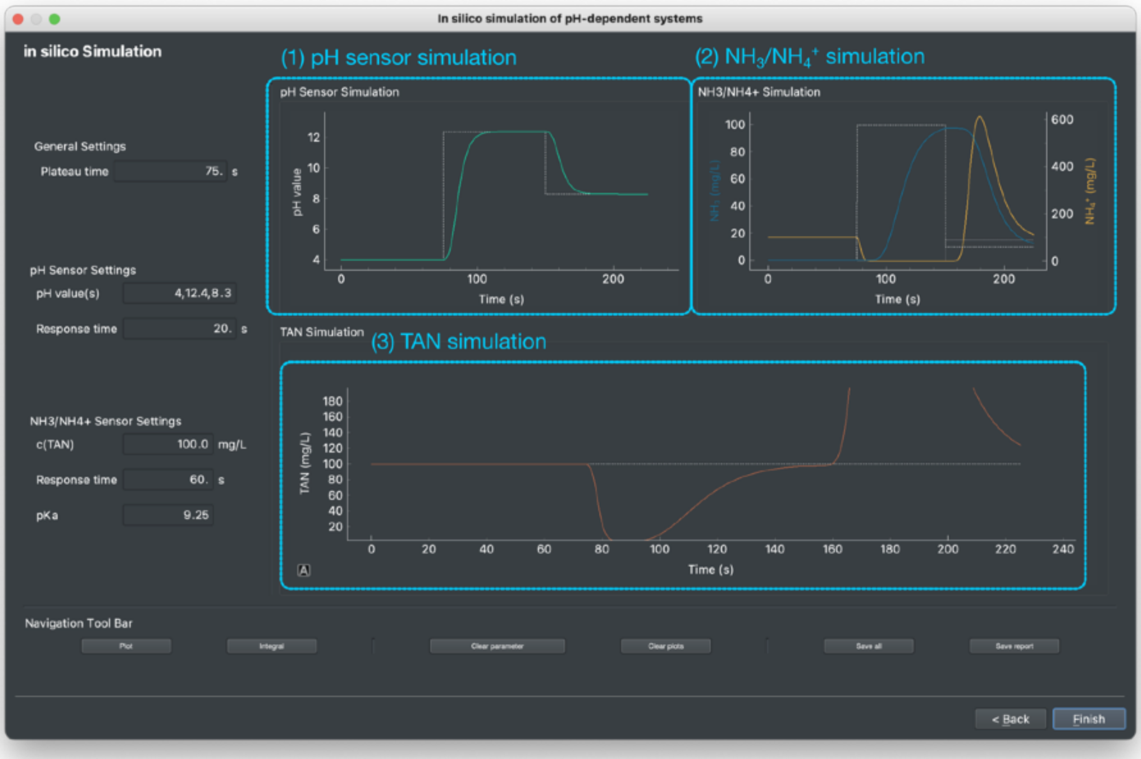
pH Sensor Settings
pH value(s) 4,12.4,8.3
Response time 20 s

NH3/NH4+ Sensor Settings
c(TAN) 100 mg/L
Response time 60 s
pKa 9.25

FIGURE 7 *Overview of the Simulation Page. On the left side* (1) *the sensor specifications are listed as previously defined. They can be adjusted in case a comparative analysis should be simulated. On the top right* (2)*, the individual analyte concentrations over time are displayed. The left panel shows the target and simulated* pH *fluctuation while the right panel will display the measuring sensor and its corresponding calculated acid/base form. The latter is a pure simulation based on the calculations described in the Background chapter. The panel at the bottom* (3) *will show the target and simulated fluctuations of the total parameter concentration. At the bottom* (4)*, a navigation toolbar will support the basic operational steps in the software.*

In the left panel, all previously defined parameters are summarised and the pH-dependent equilibrium that you intend to study is specified. On the right, three panels display the individual sensors (two plots at the top) and the resulting total parameter (bottom plot). In the bottom panel, the navigation toolbar presents an overview of what you can do with your simulation results.

If all required parameters are added/defined, press "PLOT" in the Navigation toolbar to start the simulation. The simulation for the individual sensors will be displayed in the top two plots and the resulting total parameter will be displayed in the bottom plot (cf. Figure 7).



General Settings
Plateau time 75. s

pH Sensor Settings
pH value(s) 4,12.4,8.3
Response time 20. s

NH3/NH4+ Sensor Settings
c(TAN) 100.0 mg/L
Response time 60. s
pKa 9.25

FIGURE 8 *Example of a simulation for total ammonia nitrogen (TAN). The top left panel* (1) *displays the* pH *values defined by the user (here 4, 12.4 and 8.3) and the individual species for the defined acid/base equilibrium* (2). *The bottom panel* (3) *then shows the simulated total ammonia nitrogen fluctuations compared to its target concentration. The target concentrations are displayed in white dotted lines while the simulated sensor responses are displayed as coloured solid curves.*

Starting with a pH of 4, there is a pH change to 12.4 and subsequently to 8.3 with a plateau time of 75 seconds (Figure 8(1) – pH Sensor Simulation). As a white dash-dotted line, a step function indicates the target pH change assuming instantaneous pH change. The simulated real sensor response (considering the defined sensor properties) is shown as a solid green line.

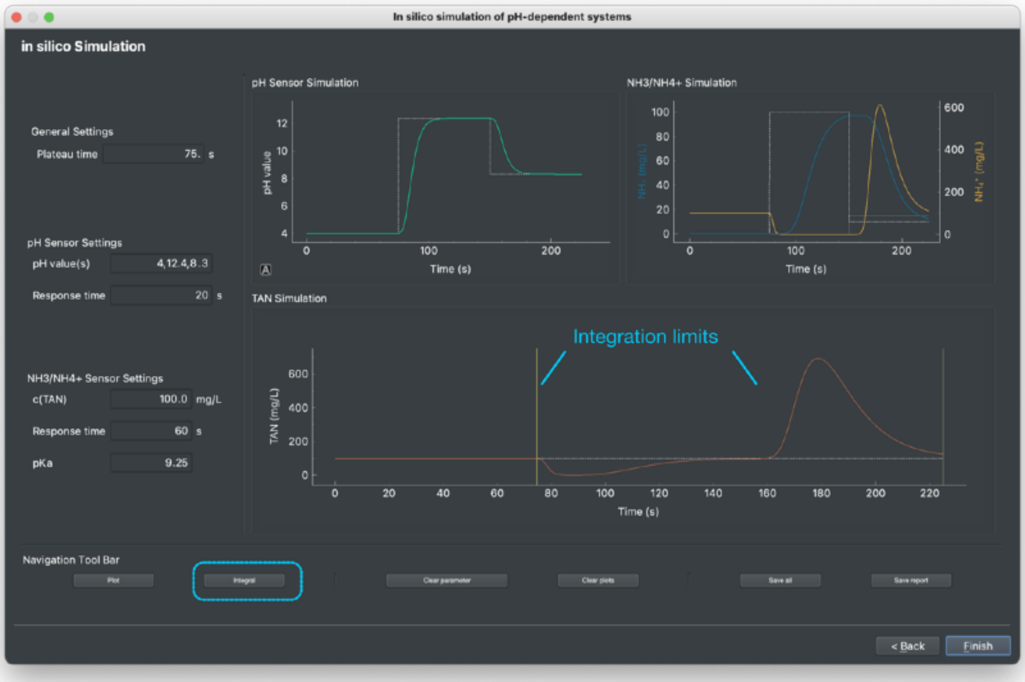
In the panel to the right (cf. Figure 8(2); NH_3/NH_4^+ Simulation), both analytes are displayed together using individual y-axes. On the primary y-axis, the analyte is displayed that has been specified by the user as the measuring sensor (here NH3; solid blue line), while the corresponding calculated analyte (here NH_4^+) is displayed as an orange solid line using the secondary y-axis. The white step functions represent the target function assuming an instantaneous change according to the pH change while the coloured lines consider a delayed sensor response.

In the panel below (cf. Figure 8(3); TAN Simulation), the total analyte concentration over time (and changing pH) is displayed. While the white dash-dotted line represents the target concentration as defined by the user, the brown line displays the total analyte concentration when summing up the two individual sensor concentrations visualising the error propagation due to delayed sensor response. The initial view is set to focus on the target concentration ±90%. To display the entire sensor response, the user may specify the scene by right-clicking in the plot or by pressing the [A] in the bottom left corner. To zoom in, first select the 1-button Mouse Mode (right-click option). Afterwards, you can zoom in by drawing a yellow rectangle across the area of interest with your cursor.

Determining the overall error of the total parameter

To determine the overall error of the total parameter caused by the different sensor response times, the area between the simulated total response curve and the target concentration must be determined. Further details on the theory and calculation steps can be found in the theory section (Background | Application's Functions).

To define the time range in which the overall error shall be calculated, simply click on the plot while holding the COMMAND/WINDOWS key. A vertical line appears in the plot defining the time range for integration. The user may fine-tune the vertical line which can easily be dragged along the x-axis. Once the time range is selected, press the "INTEGRAL" button in the Navigation toolbar (cf. Figure 9). A separate window will pop up, summarising the Integration range, the target pH and total concentration together with the integral and the resulting error (cf. Figure 10).


In silico simulation of pH-dependent systems

in silico Simulation

General Settings
Plateau time 75. s

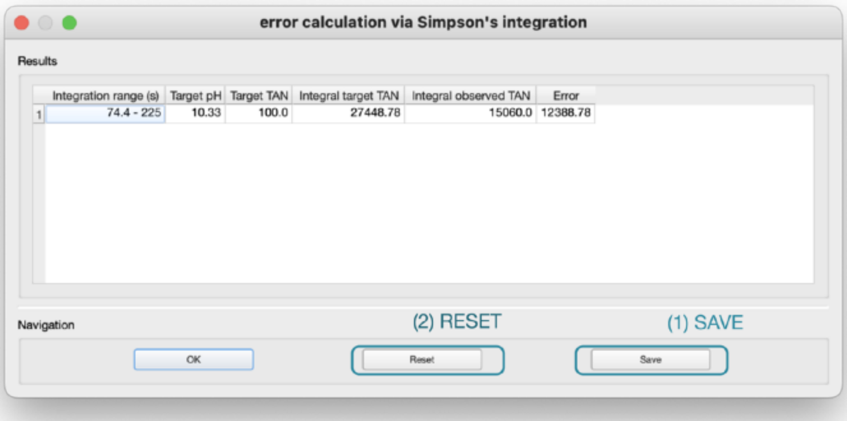
pH Sensor Settings
pH value(s) 4,12,4,8.3
Response time 20 s

NH3/NH4+ Sensor Settings
c(TAN) 100.0 mg/L
Response time 60 s
pKa 9.25

FIGURE 9 *For calculating the overall deviation between the target and simulated total parameter concentration, one needs to specify the time range within which the deviation shall be calculated. This can be done by clicking into the plot while pressing the command/windows key. The vertical lines can then be dragged along the x-axis to fine-tune the integration range. When finished, press integral to calculate the overall error.*

Save output

It is possible to save individual results such as only the integral table or the simulation graph as a text file or the entire simulation including text files and plots.

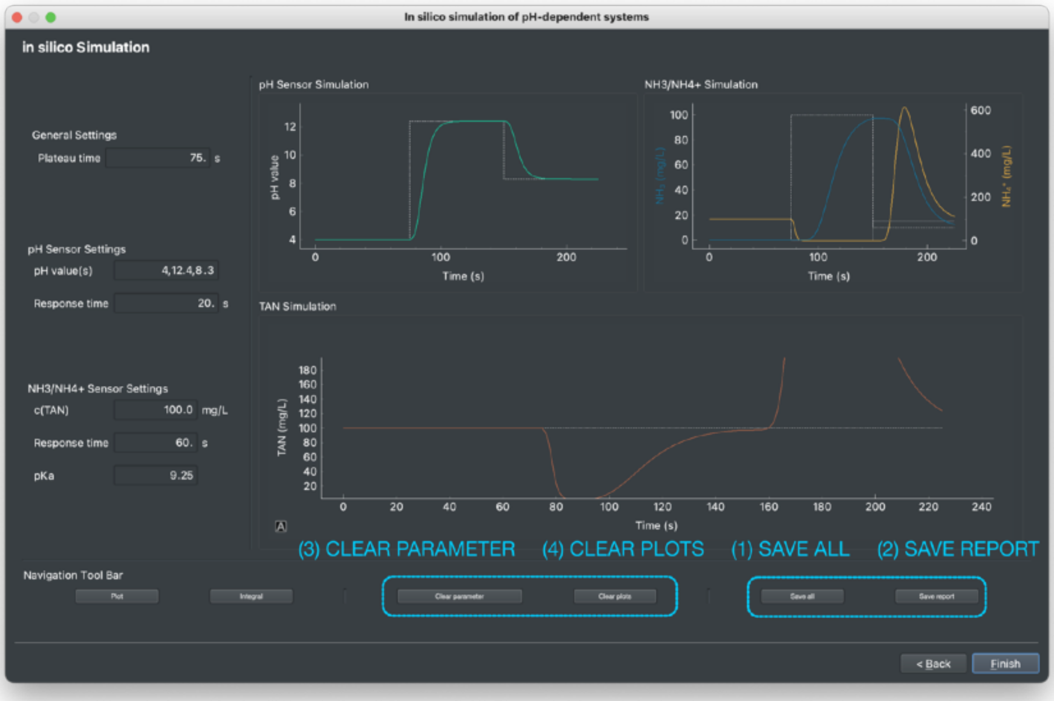


	Integration range (s)	Target pH	Target TAN	Integral target TAN	Integral observed TAN	Error
1	74.4 - 225	10.33	100.0	27448.78	15060.0	12388.78

FIGURE 10 *Pop-up window upon pressing integral. Here, a summary of the target ph and the total parameter is shown, as well as the calculated integrals of the target vs. the simulated total parameter and its resulting error. The buttons* (1) *and* (2) *allow the separate saving of the integral and resetting of the entire error calculation.*

If only the integral table is to be saved, press the "SAVE" button in the separate pop-up window (cf. **Figure 10(1)**) and define the directory and file name under which the simulation is to be saved.

To save only the simulation graph, select "EXPORT" after right-clicking in the respective simulation plot and define the directory and file name under which the simulation is to be saved. Note that this graph will be saved as it is, i.e., with a dark background. If only the calculations are to be saved, but no images, select "SAVE REPORT" (cf. **Figure 11(2)**). Then the entire simulation as well as an extra file summarising the previously defined integration and error calculation will be saved. With "SAVE ALL" (cf. **Figure 11(1)**), both the text files and the diagrams will be saved in the specified directory.



General Settings
Plateau time 75. s

pH Sensor Settings
pH value(s) 4,12.4,8.3
Response time 20. s

NH3/NH4+ Sensor Settings
c(TAN) 100.0 mg/L
Response time 60. s
pKa 9.25

FIGURE 11 *Additional operations in* SensinSilico *to either reset the plots* (4) *and parameters* (3) *for the current simulation or to save the output* (1-2)*. For the latter, the user can save only the simulation and potential error calculations as a text file* (2) *or save the simulation plots* (1)*.*

Terminate and Restart Operations

Reset function

The parameters and plots must be cleared individually by pressing “CLEAR PARAMETER” and/or “CLEAR PLOTS” in the Navigation toolbar in the main window (cf. Figure 11(3-4)). In case the simulation should remain but only the integration table should be deleted, press “RESET” in the respective pop-up window (cf. Figure 10(2)).

Terminate application

To close the application, either press “FINISH” in the Navigation toolbar or close the main window with the shortcut COMMAND + Q when working on a Mac or ALT + F4 for Windows.

JupyterNotebook | Exploring Sensor Response Curves

Where to download the Notebook and how to get started?

To explore different sensor response curves, navigate to the subfolder ***SCRIPTS*** and download both, the Jupyter Notebook Code_snippes.ipynb and the related function database dbs_CodeSnippes.py from GitHub. It is publicly available via the following link: https://github.com/silviaelisabeth/SensorResponse-inSilico.

To run the notebook, we suggest using VS Code or the Terminal/Command Prompt as we tested the notebook with VS Code 1.74.2 (Universal) for software development and testing. For the notebook to run properly, leave all the files in the folder together without restructuring or removing any parts.

Requirements

When using the source code, python3 and the following Python packages are required for execution:

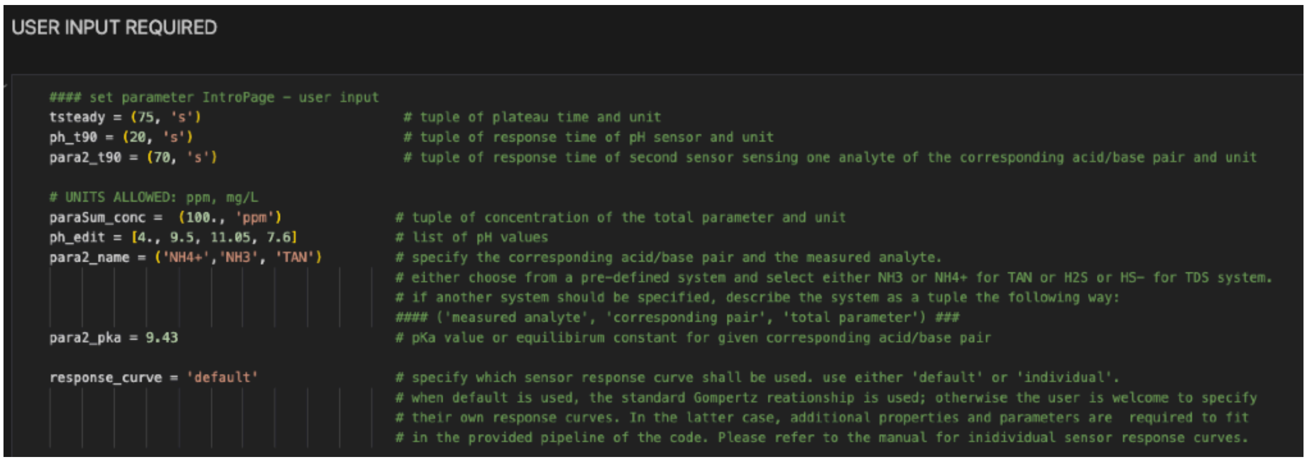
numpy (version 1.24.2), pandas (version 1.5.3), pyqtgraph (version 0.13.3), scipy (version 1.10.1), and seaborn (version 0.12.2).

The package versions listed are the versions used when the software is released. Older versions might cause errors and bugs, whereas newer versions should not make much difference. The software has been tested for both MacOS (MacBook Pro – Apple M1, Version 13.4) and Windows (Version 10.0.19045). If you have difficulties running the software, please do not hesitate to reach out (cf. Points of Contact).

Explaining the individual cells of the Notebook

For using the Jupyter Notebook, we assume a general knowledge of how to use a Jupyter Notebook, meaning we assume the user knows how to execute cells and how to define functions in python3.

1ST CELL · SPECIFYING THE INPUT PARAMETERS

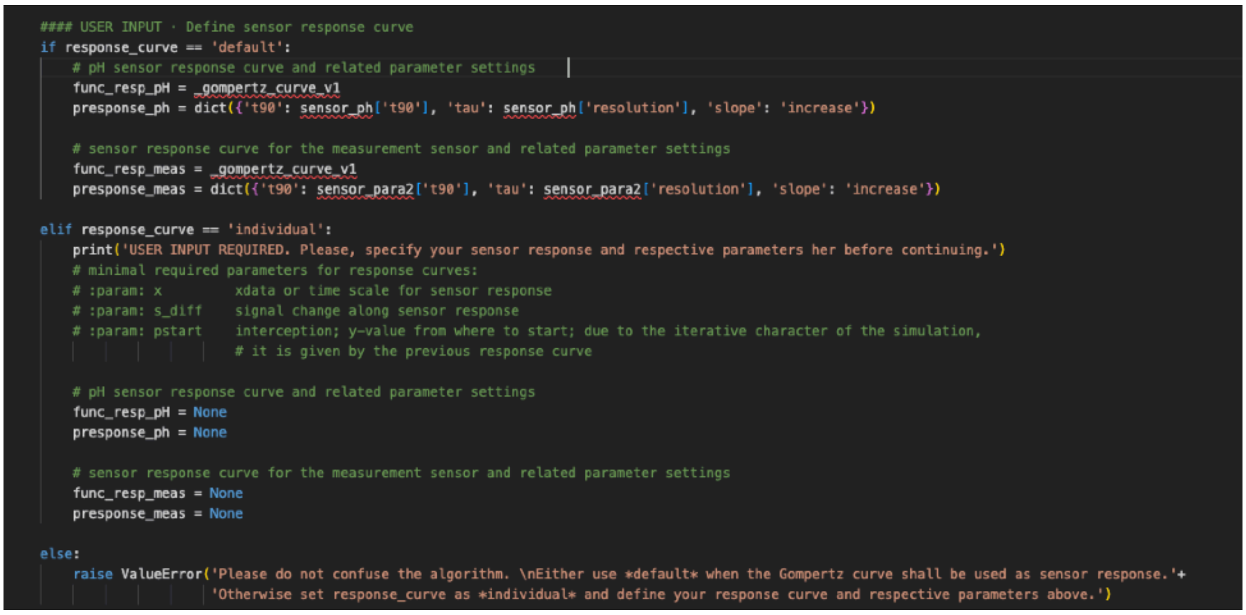

USER INPUT REQUIRED

```
#### set parameter IntroPage - user input
tsteady = (75, 's')                  # tuple of plateau time and unit
ph_t90 = (20, 's')                   # tuple of response time of pH sensor and unit
para2_t90 = (70, 's')                # tuple of response time of second sensor sensing one analyte of the corresponding acid/base pair and unit

# UNITS ALLOWED: ppm, mg/L
paraSum_conc =  (100., 'ppm')       # tuple of concentration of the total parameter and unit
ph_edit = [4., 9.5, 11.05, 7.6]     # list of pH values
para2_name = ('NH4+','NH3', 'TAN')  # specify the corresponding acid/base pair and the measured analyte.
                                    # either choose from a pre-defined system and select either NH3 or NH4+ for TAN or H2S or HS- for TDS system.
                                    # if another system should be specified, describe the system as a tuple the following way:
                                    #### ('measured analyte', 'corresponding pair', 'total parameter') ###
para2_pka = 9.43                    # pKa value or equilibirum constant for given corresponding acid/base pair

response_curve = 'default'          # specify which sensor response curve shall be used. use either 'default' or 'individual'.
                                    # when default is used, the standard Gompertz reationship is used; otherwise the user is welcome to specify
                                    # their own response curves. In the latter case, additional properties and parameters are  required to fit
                                    # in the provided pipeline of the code. Please refer to the manual for inidividual sensor response curves.
```

FIGURE 12 *The first cell of the Jupyter Notebook requires user input to specify the two-sensor system that shall be studied as well as certain related parameters.*

Here, similar to the GUI explained before, the user has to specify certain parameters, including the system that shall be studied. Regarding the latter, be aware of how to define any new system besides TAN or TDS. In case a new sensing system shall be studied, use a tuple to describe the system with the parameter to be monitored mentioned first and the total parameter to be mentioned last. The middle string specifies the corresponding pair to the measured parameter (*'measured parameter', 'corresponding parameter', 'total parameter'*). As of now, the code is designed for two sensor systems where one sensor is always a pH sensor and the other sensor can be any other analyte, describing a pH-depending system. The remaining parameters are the same as explained for the GUI in Chapter Operating Instructions. The last parameter *response_curve*, however, is new allowing the user to explore other sensor response curves other than the Gompertz curve mentioned in Chapter Background | Application's Functions.

2ND CELL · SPECIFYING THE SENSOR RESPONSE CURVE



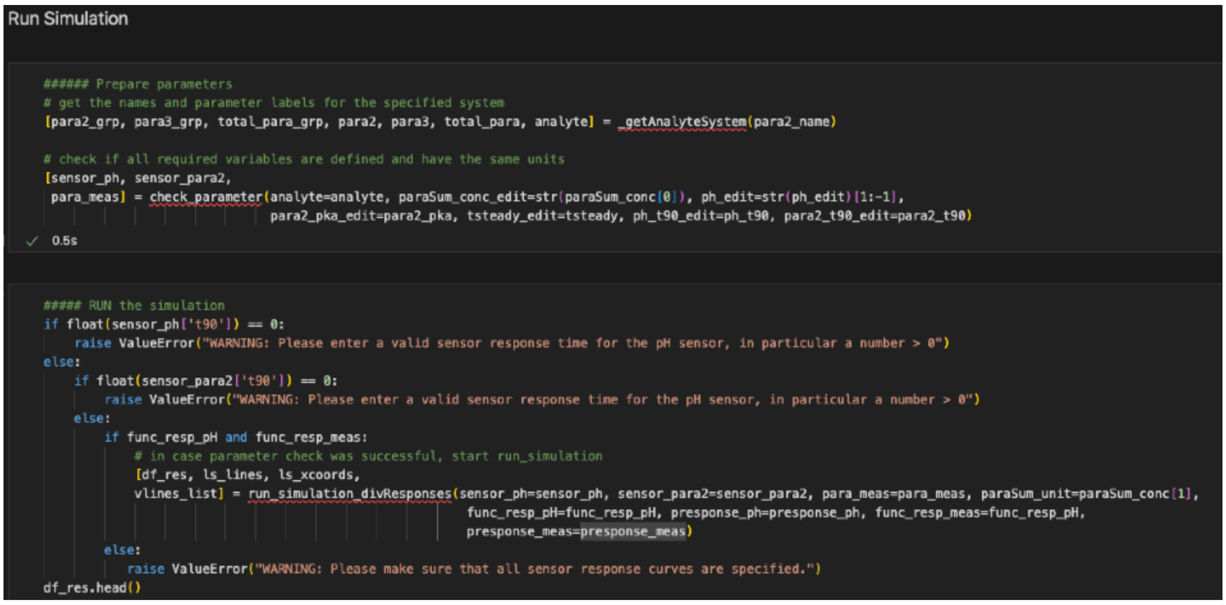
```python
#### USER INPUT · Define sensor response curve
if response_curve == 'default':
    # pH sensor response curve and related parameter settings
    func_resp_pH = _gompertz_curve_v1
    presponse_ph = dict({'t90': sensor_ph['t90'], 'tau': sensor_ph['resolution'], 'slope': 'increase'})

    # sensor response curve for the measurement sensor and related parameter settings
    func_resp_meas = _gompertz_curve_v1
    presponse_meas = dict({'t90': sensor_para2['t90'], 'tau': sensor_para2['resolution'], 'slope': 'increase'})

elif response_curve == 'individual':
    print('USER INPUT REQUIRED. Please, specify your sensor response and respective parameters her before continuing.')
    # minimal required parameters for response curves:
    # :param: x         xdata or time scale for sensor response
    # :param: s_diff    signal change along sensor response
    # :param: pstart    interception; y-value from where to start; due to the iterative character of the simulation,
                        # it is given by the previous response curve

    # pH sensor response curve and related parameter settings
    func_resp_pH = None
    presponse_ph = None

    # sensor response curve for the measurement sensor and related parameter settings
    func_resp_meas = None
    presponse_meas = None

else:
    raise ValueError('Please do not confuse the algorithm. \nEither use *default* when the Gompertz curve shall be used as sensor response.'+
                     'Otherwise set response_curve as *individual* and define your response curve and respective parameters above.')
```

FIGURE 13 *The second cell of the Jupyter Notebook allows the user to explore different sensor response curves. Please refer to the text below to learn about specific requirements for the sensor response curve.*

If the parameter *response_curve* is set to **default**, the Gompertz curve will be used as sensor response for both sensors. However, if *response_curve* is set to **individual,** the user can specify other response curves. While the user is thereby free to define any (reasonable) response function, the following three parameters must be included in any user-defined function:

x	Time scale along which the sensor response is observed; data given as list of float numbers in seconds.
s_diff	This is a scaling factor to adjust the general sensor response curve to the specific signal plateaus. This value will be defined due to the time-dependent character of the simulation; value given as float number.
pstart	Interception or y-value from where the sensor response shall begin. Due to the iterative character of the simulation, this value is given from the previous step; value given as float number.

3RD & 4TH CELL · EXECUTING THE SIMULATION



Run Simulation

```
###### Prepare parameters
# get the names and parameter labels for the specified system
[para2_grp, para3_grp, total_para_grp, para2, para3, total_para, analyte] = _getAnalyteSystem(para2_name)

# check if all required variables are defined and have the same units
[sensor_ph, sensor_para2,
 para_meas] = check_parameter(analyte=analyte, paraSum_conc_edit=str(paraSum_conc[0]), ph_edit=str(ph_edit)[1:-1],
                              para2_pka_edit=para2_pka, tsteady_edit=tsteady, ph_t90_edit=ph_t90, para2_t90_edit=para2_t90)
```

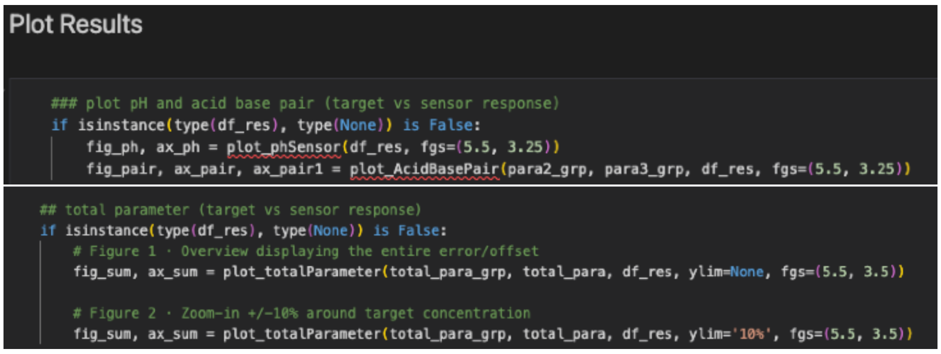
```
✓ 0.5s
```

```
##### RUN the simulation
if float(sensor_ph['t90']) == 0:
    raise ValueError("WARNING: Please enter a valid sensor response time for the pH sensor, in particular a number > 0")
else:
    if float(sensor_para2['t90']) == 0:
        raise ValueError("WARNING: Please enter a valid sensor response time for the pH sensor, in particular a number > 0")
    else:
        if func_resp_pH and func_resp_meas:
            # in case parameter check was successful, start run_simulation
            [df_res, ls_lines, ls_xcoords,
            vlines_list] = run_simulation_divResponses(sensor_ph=sensor_ph, sensor_para2=sensor_para2, para_meas=para_meas, paraSum_unit=paraSum_conc[1],
                                                       func_resp_pH=func_resp_pH, presponse_ph=presponse_ph, func_resp_meas=func_resp_pH,
                                                       presponse_meas=presponse_meas)
        else:
           raise ValueError("WARNING: Please make sure that all sensor response curves are specified.")
df_res.head()
```

FIGURE 14 *The third and forth cell of the Jupyter Notebook will simulate the target and actual sensor response based on the provided parameters for the system specified.*

The third and forth cell will simulate the target and actual sensor responses of all sensors as well as the corresponding and total analyte based on the provided parameters for the system specified. While the third cell include preparation steps such as aligning the time scale of the sensors, unifying the units and getting the correct labels of the two-sensor system to be used in the plots later on, the forth cell executes the main simulation function. If the function concludes without any error or bug, a DataFrame containing the simulation results will be displayed.

5TH & 6TH CELL · PLOTTING RESULTS



Plot Results

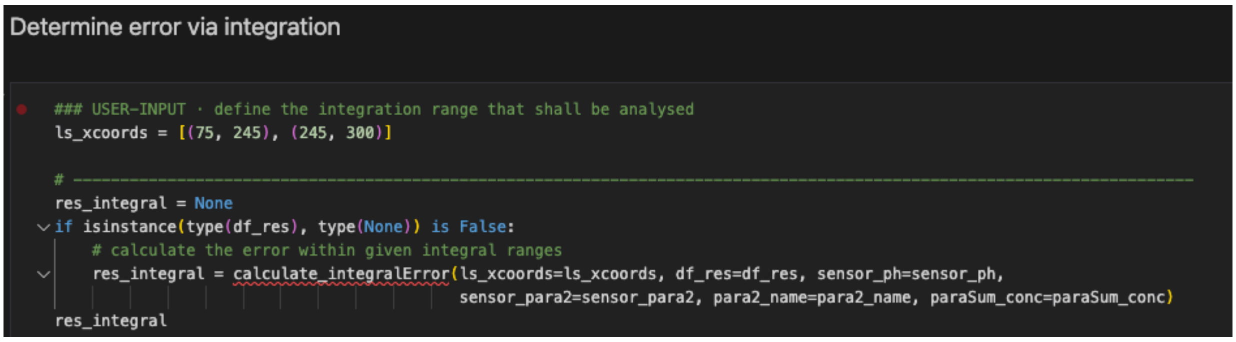
```
### plot pH and acid base pair (target vs sensor response)
if isinstance(type(df_res), type(None)) is False:
    fig_ph, ax_ph = plot_phSensor(df_res, fgs=(5.5, 3.25))
    fig_pair, ax_pair, ax_pair1 = plot_AcidBasePair(para2_grp, para3_grp, df_res, fgs=(5.5, 3.25))
```

```
## total parameter (target vs sensor response)
if isinstance(type(df_res), type(None)) is False:
    # Figure 1 · Overview displaying the entire error/offset
    fig_sum, ax_sum = plot_totalParameter(total_para_grp, total_para, df_res, ylim=None, fgs=(5.5, 3.5))

    # Figure 2 · Zoom-in +/-10% around target concentration
    fig_sum, ax_sum = plot_totalParameter(total_para_grp, total_para, df_res, ylim='10%', fgs=(5.5, 3.5))
```

FIGURE 15 *The fifth and sixth cell of the Jupyter Notebook will plot the sensor response of the individual sensors, the corresponding acid/base pair as well as the total parameter for the system specified. In each case the simulated sensor response will be compared with the target sensor response.*

As described for the GUI, the next cells in the Jupyter Notebook plot and display the sensor response curves for the specified system. The fifth cell will plot the pH sensor response in an individual figure and the corresponding acid/base pair in a separate figure. The sixth cell will then plot the total parameter concentration in a separate figure. Here, the first figure will display the entire sensor response and its error range (meaning the full y-scale is shown), while the second figure will show a zoomed-in figure. The zoom-in level can be adjusted by the user and is initially set to 10% around the target concentration.

7TH CELL · CALCULATING THE ERROR VIA INTEGRATION


Determine error via integration

```
### USER-INPUT · define the integration range that shall be analysed
ls_xcoords = [(75, 245), (245, 300)]

# -----------------------------------------------------------------------------------------------------------------
res_integral = None
if isinstance(type(df_res), type(None)) is False:
    # calculate the error within given integral ranges
    res_integral = calculate_integralError(ls_xcoords=ls_xcoords, df_res=df_res, sensor_ph=sensor_ph,
                                           sensor_para2=sensor_para2, para2_name=para2_name, paraSum_conc=paraSum_conc)
res_integral
```

FIGURE 16 *In the seventh cell of the Jupyter Notebook we will calculate the error within the user-defined integration ranges. The error is the deviation between the simulated and target integral using Simpson's integration rule.*

In the last cell of the Notebook, the error will be calculated within the user-defined integration ranges. For that, the user needs to provide a list of tuples that define the integration range. Then, the error will be calculated as described in the Chapter Background | Application's Functions.