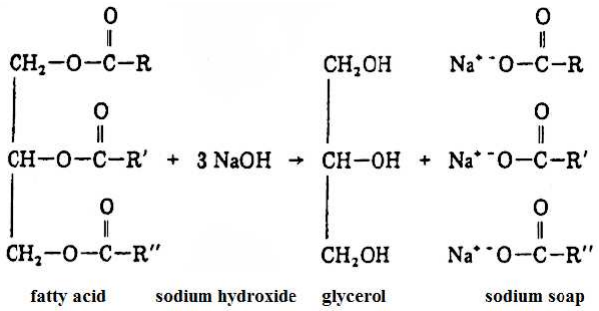
Effect of the precipitation of acid soap and alkanoic acid crystallites on the bulk pH

Gergana Velikova, Ivan Georgiev, Milena Veneva

Introduction

As known from everyday life, physical properties of solutions drastically differ from those of the pure solvents. One distinctive characteristic of the mixtures is their surface tension (γ) after the addition of solute. In many cases, such as multifarious industry productions, the usage of surface-active substances is desired, because of their ability to lower the value of γ. By definition, the surface tension is described as the tension of the liquid molecules on the interface, caused by their interactions with the molecules in the bulk of the liquid, which are thermodinamicaly more favorable.

One possible application of surface-active substances as soaps and detergents are in the emulsion industry. In particular, soaps are widely used for hygienic purposes for hundreds of years. They are products of a chemical reaction between fatty acids (mostly from C_{12} to C_{18} saturated and C_{18} mono-, di- and triunsaturated ones), and sodium or potassium hydroxide in a process called saponification.



Before any of the aforementioned detergents can be used as a cleaning agent they need to be solubilized. In fact, the solubility of soaps and other ionic surfactants depends strongly on the temperature. Therefore, their chemical presence in solution is low, before the temperature reaches the Krafft point. Another key feature of soap colloid solutions is the presence of micelles. As known from the experience, all types of surfactants exist as monomers in the solution, before they reach the critical micelle concentration (CMC), and start self-assembling into micelles. It is important to realize that the soap's micelles cleaning mechanism is

the trapping of substances, which are insoluble in water.

The water solutions of such soaps include different chemical species such as ions of water, soap and hydrogen carbonates, the last ones are results of the solubility of CO_2 in water. Moreover, because of the industry needs, the behavior of the system is examined in the presence of sodium chloride salt and under different acidity.

Formulation of the problem

Chemical settling and goals

One of the most important characteristics of the industrial cleaning products is their optimal pH, which is monitored with addition of fatty acid salts. Given all introductory points, the one component system including sodium soap, water, sodium chloride, and dissolved carbon dioxide is modelled in the paper. Further, chemical species in the complex mixture, coefficients, and constants used in the model formulation are denoted as: K_{A} – fatty acid's dissociation constant; K_{W} – water's dissociation constant; Q_{MZ} – rate constant of the soap production; $K_{\mathrm{CO_2}}$ – used, because of the solubility of the CO_2 from the atmosphere; C_{H} – concentration of the hydrogen cations; C_{Z} – concentration of the fatty acid anions; C_{M} – concentration of the metal cations; C_{MZ} – concentration of the soaps; C_{OH} – concentration of the hydroxide anions; $C_{\mathrm{HCO_3}}$ – concentration of the hydrogencarbonate anions; C_{A} – concentration of the added salt (NaCl); C_{B} – concentration of the added base (NaOH); C_{HZ} – concentration of the undissociated fatty acid. The rate coefficient of soap production and all other dissociation constants are of the type of rate constants. In addition, because of the nature of the manufacturing process, we assume that all reactions are in equilibrium. Therefore, the system of ordinary differential equations from the reaction scheme simplifies to a system of polynomial equations with more than one variable.

$$F_j(x_1, x_2, \ldots, x_N) = b_j, j = \overline{1, N}$$

In that case, we expect to obtain more than one solution and we need to set goals for our numerical implementation of the formulated mathematical model:

goal 1: fast algorithm for solving the system;

goal 2: fast algorithm to detect the positive solution among all of the system's solutions;

goal 3: fitting the theoretically evaluated data for pH with the experimentally obtained one;

goal 4: high precision of the solution.

Mathematical Model

The system of polynomial equations is in the form

$$\begin{aligned} C_{\mathrm{H}} C_{\mathrm{Z}} \gamma_{\pm}^2 &= K_{\mathrm{A}} C_{\mathrm{HZ}} \\ C_{\mathrm{M}} C_{\mathrm{Z}} \gamma_{\pm}^2 &= Q_{\mathrm{MZ}} C_{\mathrm{MZ}} \\ C_{\mathrm{H}} C_{\mathrm{OH}} \gamma_{\pm}^2 &= K_{\mathrm{W}} \\ C_{\mathrm{H}} C_{\mathrm{HCO_3}} \gamma_{\pm}^2 &= K_{\mathrm{CO_2}} \\ I = C_{\mathrm{H}} + C_{\mathrm{M}} &= C_{\mathrm{OH}} + C_{\mathrm{HCO_3}} + C_{\mathrm{Z}} + C_{\mathrm{A}} \\ m_{\mathrm{M}} &= C_{\mathrm{T}} + C_{\mathrm{A}} + C_{\mathrm{B}} - C_{\mathrm{M}} - C_{\mathrm{MZ}} \\ m_{\mathrm{Z}} &= C_{\mathrm{T}} - C_{\mathrm{Z}} - C_{\mathrm{HZ}} - C_{\mathrm{MZ}} \end{aligned} \tag{1}$$

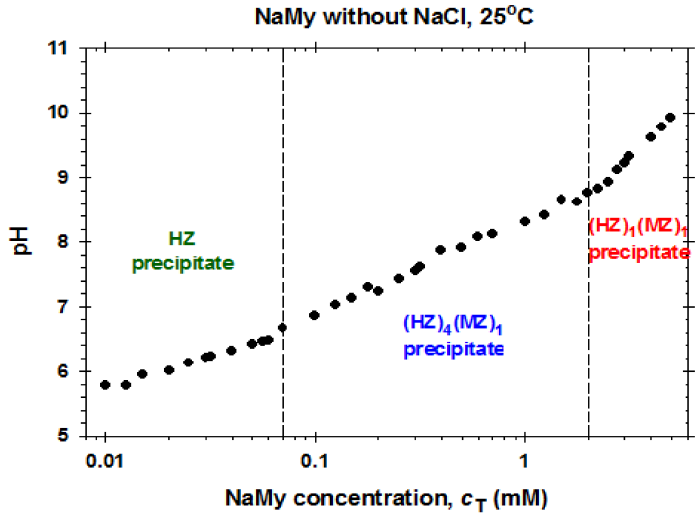
where $K_{\mathrm{W}} = 6.81 \times 10^{-15}$ M^2, $K_{\mathrm{A}} = 1.995 \times 10^{-5}$ M, and $Q_{\mathrm{MZ}} = 2.84$ M. The activity coefficient $\gamma_{\pm}$ is calculated from the semi-empirical formula:

$$\log_{10} \gamma_{\pm} = 0.055 I - \frac{0.5115\sqrt{I}}{1 + 1.316\sqrt{I}} \tag{2}$$

where I is the ionic strength and

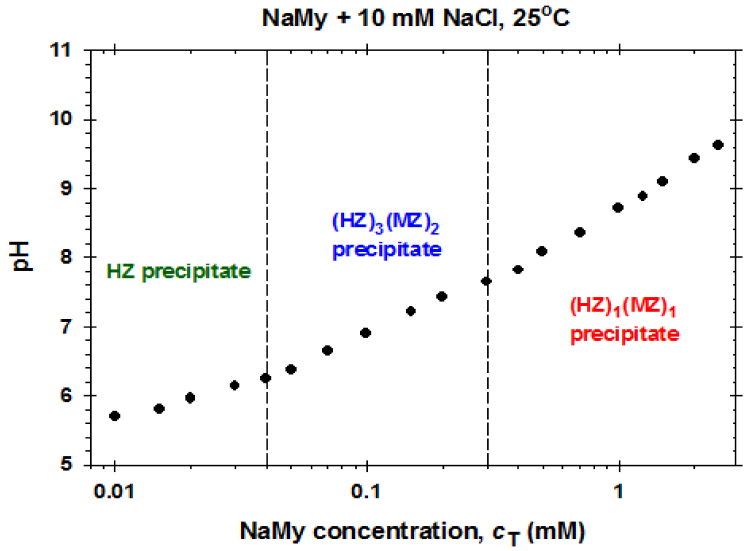
$$\mathrm{pH} = -\log_{10}(\gamma_{\pm} C_{\mathrm{H}}). \tag{3}$$

First case – without NaCl



- $C_{\mathrm{A}} = 0$ M and $C_B = 0$ M

Second case – with NaCl



- $C_{\mathrm{A}} = 0.01$ M and $C_{\mathrm{B}} = 0\,\mathrm{M}$

First and second case – first interval

- solution with fatty acid precipitates
- $C_{\mathrm{HZ}} = S_{\mathrm{HZ}} = 5.25 \times 10^{-7}$ M
- $m_{\mathrm{M}} = 0$

$\Rightarrow$ fit $K_{\mathrm{CO_2}}$
$\Rightarrow$ comparison between the obtained $K_{\mathrm{CO_2}}$ values in the two cases.

First and second case – second interval

- solution with precipitate of $j : n$ acid soap
- $\dfrac{m_{\mathrm{M}}}{n} = \dfrac{m_{\mathrm{Z}}}{n+j}$
- $C_{\mathrm{H}}^{j} C_{\mathrm{M}}^{n} C_{Z}^{j+n} \gamma_{\pm}^{2j+2n} = K_{jn}, \quad \text{if } j = 4 \text{ and } n = 1$
- $C_{\mathrm{H}}^{j} C_{\mathrm{M}}^{n} C_{\mathrm{Z}}^{j+n} \gamma_{\pm}^{2j+2n} = K_{jn}, \quad \text{if } j = 3 \text{ and } n = 2$

$\Rightarrow$ fit K_{41}
$\Rightarrow$ fit K_{32}

First and second case – third interval

- solution with precipitate of $j : n$ acid soap
- $\dfrac{m_{\mathrm{M}}}{n} = \dfrac{m_{\mathrm{Z}}}{n+j}$
- $C_{\mathrm{H}}^{j} C_{\mathrm{M}}^{n} C_{\mathrm{Z}}^{j+n} \gamma_{\pm}^{2j+2n} = K_{jn}, \quad \text{if } j = 1 \text{ and } n = 1$

$\Rightarrow$ fit K_{11}
$\Rightarrow$ comparison between the obtained K_{11} values in the two cases.

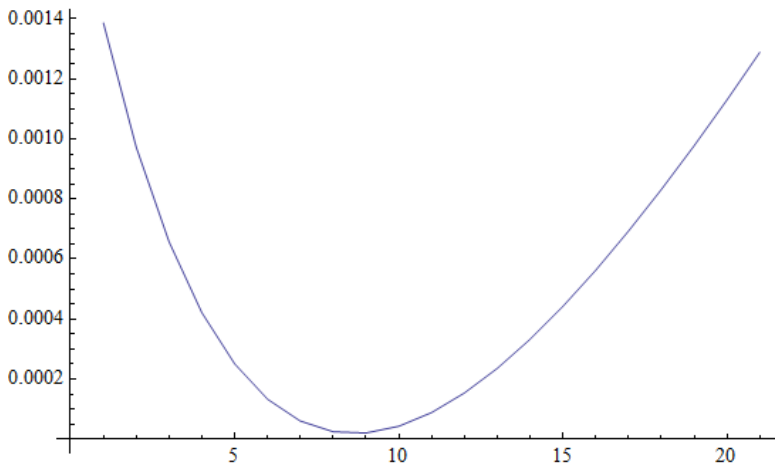
Solution

In order to fit the theoretically evaluated data with the experimentally obtained one, we minimize the following functional:

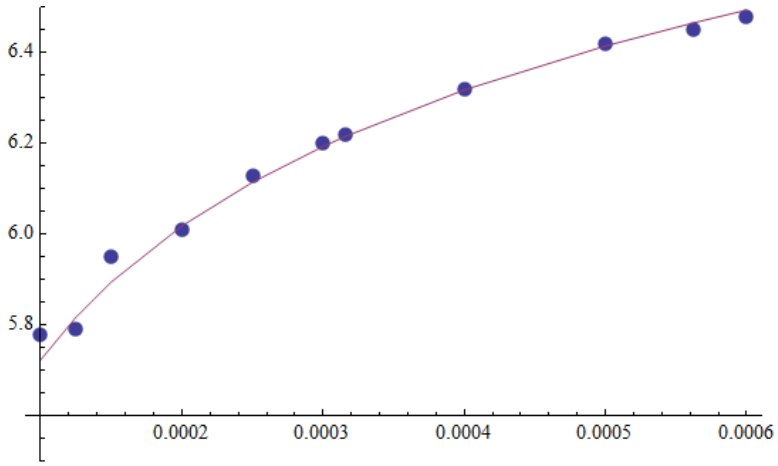
$$P(K_{\mathrm{CO_2}}) = \frac{1}{n} \sum_{k=1}^{n} \left[1 - \frac{\mathrm{pH}_{th}(k)}{\mathrm{pH}_{exp}(k)} \right]^2$$

by numerical variation of $K_{\mathrm{CO_2}}$. Here pH_{th} are the values for pH obtained from (1)–(3) and pH_{exp} are the measured experimental data. Using software for symbolic computations (like Mathematica) one can find a good initial approximation for the parameter $K_{\mathrm{CO_2}}$.

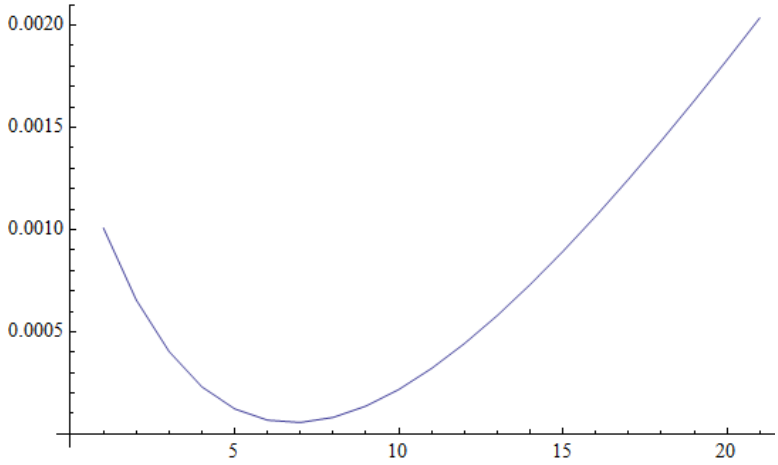
First case (first interval) – values of $P(K_{\mathrm{CO_2}})$**,** $n = 20$



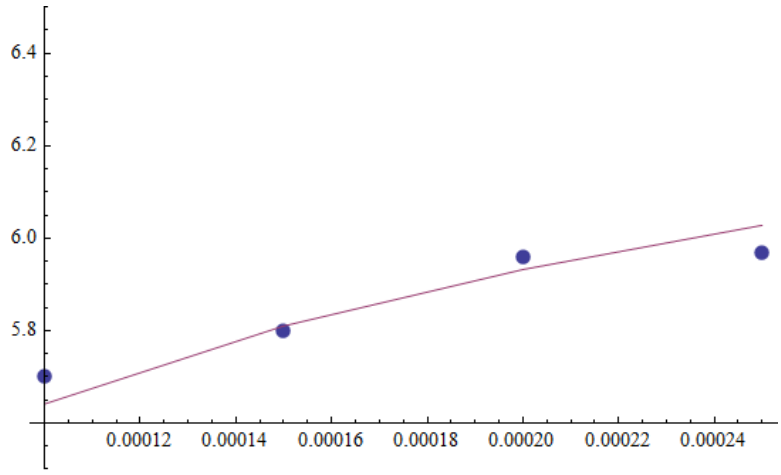
First Case (first interval) – fit of the theoretically evaluated data for pH with the experimentally obtained one ($K_{CO_2} \approx 1.8 \times 10^{-10}$)



Second Case (first interval) – values of $P(K_{CO_2})$, $n = 20$



Second Case (first interval) – fit of the theoretically evaluated data for pH with the experimentally obtained ones ($K_{CO_2} \approx 2 \times 10^{-10}$)



Using the obtained value of K_{CO_2} and the same technique one can fit the parameters K_{32} and K_{11} for the second and respectively the third interval.

Fast algorithm for finding the positive solution

So far we have talked about solving the system of equations we have and fitting the theoretically evaluated data for pH with the experimentally obtained one. However, a very important step of the problem solving is to detect quickly the positive solution aomung the whole set of the system's solutions.

The problem now is the following:

- we have a system of no more than 20 polynomial equations;
- there is no estimation for the number of the solutions that such a system can have, because this number depends on the type of the crystals that are used;
- the components of the solutions could be complex numbers;
- according to a hypothesis from the practice the system can have only one positive solution.

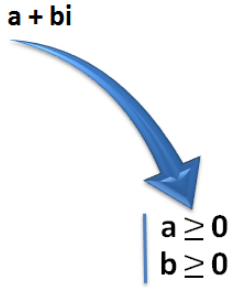
The aim is a fast algorithm to detect the positive solution.

We are going to show two different algorithms, each of them was implemented both in *C++* and *Matlab*. In order to compare the two algorithms, we have been given an example – system, which consists of 16 equations with 16 variables.

The solutions obtained with *Mathematica* are 9, only one of which is positive. For the needs of the computer programs we have written, we assume that each component of each solution is a complex number.

First approach

The first approach is to compare each component of each solution with 0:



So, the algorithm is the following: we take the first component of the first solution. If the real part of this component is not negative, then we compare the imaginary part of this component with 0. If this part is also not negative, we take the second component of the current solution and continue in the same manner. If we find a negative part in a component, we reject the current solution and continue with the next one. Because of the fact that existence of only one positive solution is just a hypothesis, our algorithm does not stop if it finds a solution, which consists of only positive components, but continues searching for other positive solutions.

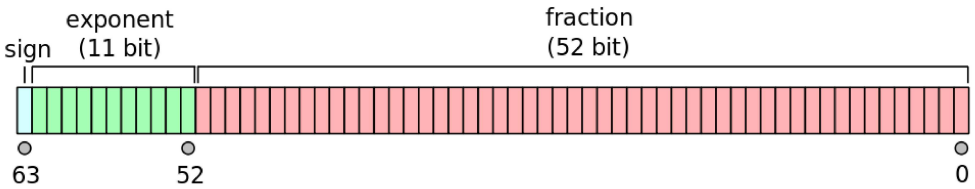
This way, the complexity of the first algorithm is $O(n * m)$, where n is the number of the solutions of the system and m is the number of the components in each solution.

Second approach

In order to garantee the needed precision of the solution, we represent the real and the imaginary part of each component of each solution as a double-precision floating-point number. The benefit is that each double-precision floating-point number has 15 decimal digits in the decimal part of the mantis and the absolute value of such a number is between 10^{-308} and 10^{308}.

Each double-precision floating-point number is represented in the computer's memory as $8B = 64$ bits (according to the standard IEEE). In the picture below you can see what each of these 64 bits is used for. The most important bit for

our second approach is the sign bit. It contains 0 if the number is ≥ 0 and 1, if it is negative.



Thus, the second approach is the following: instead of comparing lexicographically all the bits in the binary representation of a number with the binary representation of 0, as we did in our first approach, we compare only the sign bit of the current number with the sign bit of 0, which is 0. The remaining part of the first algorithm is not changed.

Then:

- the complexity of the algorithm *comparison with 0* is: $O(l * n * m)$;
- the complexity of the algorithm *bit comparison* is: $O(n * m)$,

where l is the number of the bits in the binary representation of the numbers, which we consider. In our case it is 64.

In the worst case scenario, the second algorithm works as fast as the first one. It depends on the optimizations that the processor makes.

Comparison between the two algorithms

C++/Fortran vs. Matlab/Mathematica

- *C++* and *Fortran* are compiled programming languages, which means that the source code of the program is transformed into a machine code before the execution of the program;
- *Matlab* and *Mathematica* are interpreted programming languages, which means that the programs are executed directly, which usually makes them slower because of the overhead of the processor.

$\Rightarrow$ *C++* and *Fortran* are better for scientific computations.

Implementation with MATLAB – time (in seconds)

Bit Comparison	Comparison with 0
3.683144e-005	2.888495e-005
5.576608e-005	3.135687e-005
2.870890e-005	2.804986e-005
4.362872e-005	4.470864e-005
3.317848e-005	3.355881e-005

A number of tests (~ 50) were made. Only two of them show that the algorithm bit comparison is faster than the algorithm comparison with 0 (these are the results in the last two rows at the table below). According to all of the other tests (such results are shown in the first three rows at the table below) we conclude that the algorithm *bit comparison* is slower than the algorithm *comparison with 0*. The reason is that the function, which *Matlab* uses for finding the sign bit, probably has the following implementation (with some optimizations): $\operatorname{sign} v = -(v < 0)$. We cannot be sure, because the function is build-in. The same situation is observed in *Mathematica*. So, using of *Matlab* (and *Mathematica*, too) for solving this problem cannot give us satisfying results.

Implementation with C++ – time

As an example we consider a system having 9 solutions, each with 16 components:

- the average time of the algorithm *comparison with 0*: 1 μs;
- the average time of the algorithm *bit comparison*: 0 μs.

Number of Solutions	Time (µs) - C++		Time (µs) - Matlab	
	Bit Comparison	Comparison with 0	Bit Comparison	Comparison with 0
9	0	1	29	28
801	15	18	560	597
1601	31	39	1090	1113
8001	186	237	5377	5454

This means that the average time of the algorithm *bit comparison* is in nanoseconds. In order to compare the average time for the execution of both implementations of the two algorithms, we test them for bigger number of solutions. In the table above one can see that for 8001 solutions within which only one is positive the algorithm *comparison with* 0 is slower than the algorithm *bit comparison* and the difference in times is 50 μs.

References

[1] Peter Kralchevsky, Krassimir Danov, Cenka Pishmanova, Stefka Kralchevska, Nikolay Christov, Kavssery Ananthapadmanabhan, Alex Lips. *Effect of the Precipitation of Neutral-Soap, Acid-Soap, and Alkanoic Acid Crystallites on the Bulk pH and Surface Tension of Soap Solution.* Langmuir (2007), 23, 3538–3553.

[2] Mariana Boneva, Krassimir Danov, Peter Kralchevsky, Stefka Kralchevska, Kavssery Ananthapadmanabhanb, Alex Lipsc. *Coexistance of micelles and crystallites in solutions of potassium myristate: Soft matter vs. solid matter.* Colloids and Surfaces A: Physicochem. Eng. Aspects 354 (2010) 172–187.

[3] Krassimir Danov, Peter Kralchevsky, Kavssery Ananthapadmanabhan. *Miccele-monomer equilibra in solutions of ionic surfactants and in ionic-nonionic mixtures: A generalized phase separation model.* Advances in Colloid and Interface Science 206 (2014) 17–45.

[4] K. Birdi. *Surface and Colloid Chemistry: Principles and Applications* (2009), 244 pages.

[5] Peter Atkins, Julio de Paula. *Physical Chemistry.* 9th Edition (2009), 972 pages.

[6] Preslav Nakov, Panayot Dobrikov. *Programirane=++Algoritmi.* 3rd Edition (2005), 703 pages.