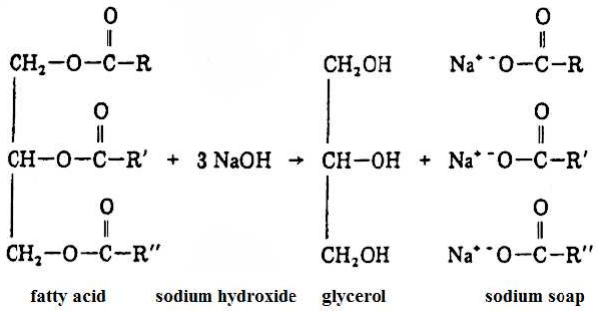
Effect of the precipitation of acid soap and alkanoic acid crystallites on the bulk pH

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

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

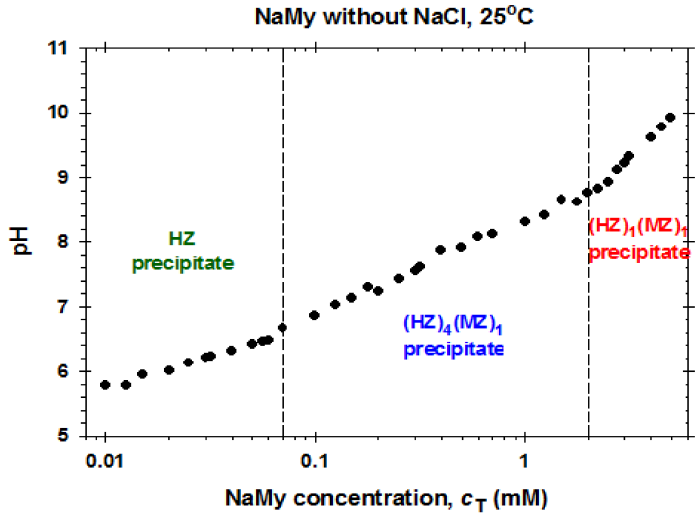
where $K_{\mathrm{W}} = 6.81 \times 10^{-15}$ M^2, $K_{\mathrm{A}} = 1.995 \times 10^{-5}$ M, and $Q_{\mathrm{MZ}} = 2.84$ M. The activity coefficient $\gamma_{\pm}$ is calculated from the semi-empirical formula:

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

where I is the ionic strength and

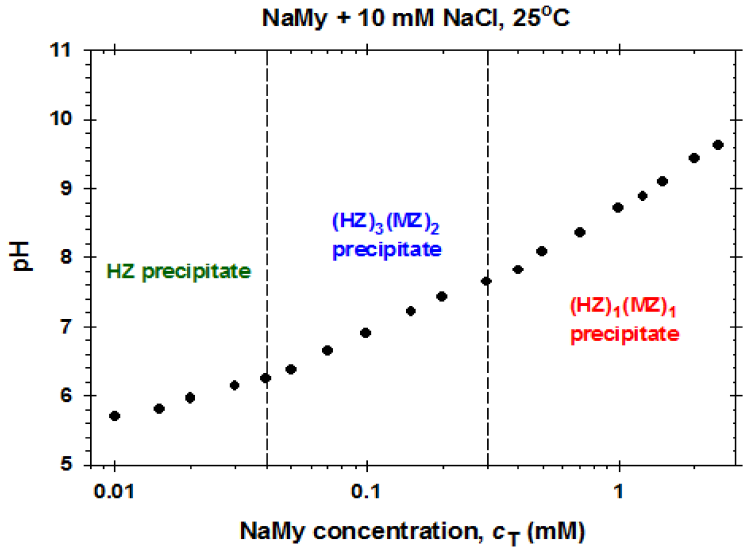
$$\mathrm{pH} = -\log_{10}(\gamma_{\pm} C_{\mathrm{H}}). \tag{3}$$

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.

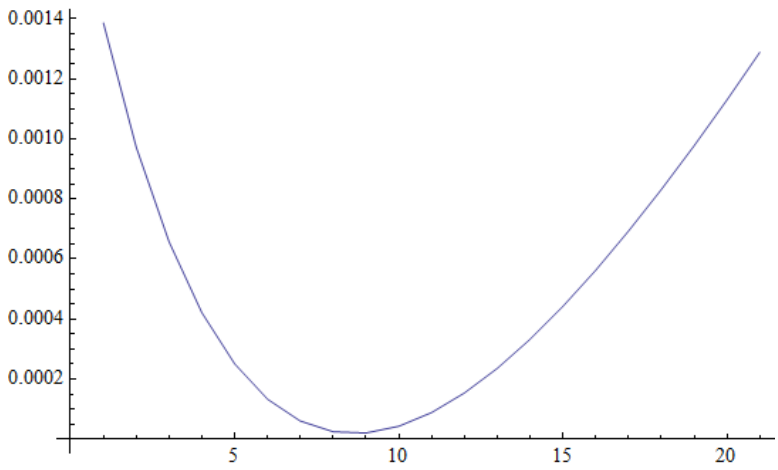
Solution

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

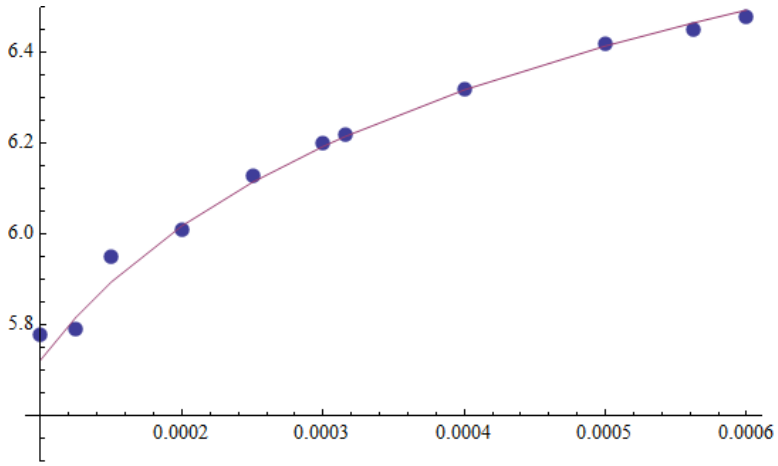
$$P(K_{\mathrm{CO_2}}) = \frac{1}{n} \sum_{k=1}^{n} \left[1 - \frac{\mathrm{pH}_{th}(k)}{\mathrm{pH}_{exp}(k)} \right]^2$$

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 $K_{\mathrm{CO_2}}$.

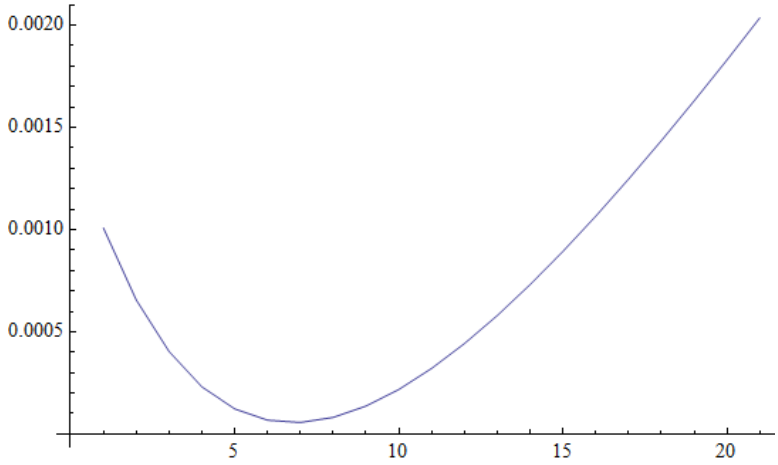
First case (first interval) – values of $P(K_{\mathrm{CO_2}})$**,** $n = 20$



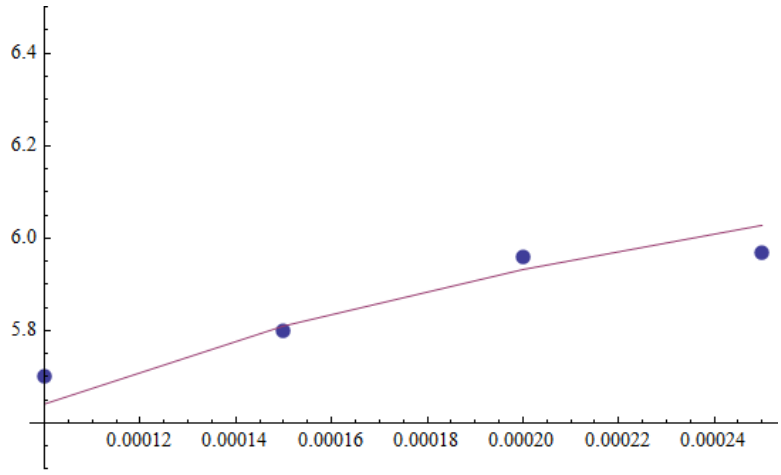
First Case (first interval) – fit of the theoretically evaluated data for pH with the experimentally obtained one ($K_{CO_2} \approx 1.8 \times 10^{-10}$)



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.

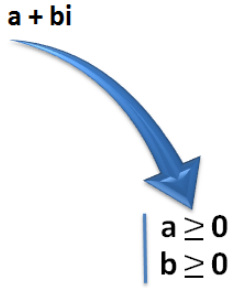
The aim is a fast algorithm to detect the positive solution.

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.

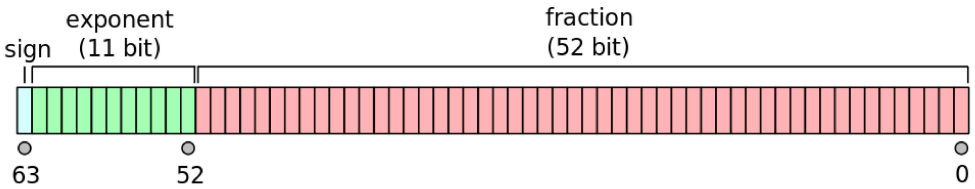
This way, the complexity of the first algorithm is $O(n * m)$, where n is the number of the solutions of the system and m is the number of the components in each solution.

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.

Circular arc spline approximation of pointwise curves for use in NC programing

Ana Avdzhieva, Dragomir Aleksov, Ivan Hristov, Nikolai Shegunov,
Pencho Marinov

1. Introduction

We consider a numerical control (NC) cutting machine which can cut only line segments and circular arcs. Thermal cutting processes require constant tool velocity because

- too slow velocity leads to overheating and melting,
- too fast velocity interrupts the cutting process.

The inputs with which the machine works are sets of points in a particular order which are in Cartesian plane.
From a set of points (inputs) we must create a sequence of line segments and circular arcs that pass through some of the points and are "sufficiently close" to the others – ϵ error condition. The case in which the points can be approximated with straight line segments is well investigated. We are interested in the sets of points which can only be approximated by arcs. Below we formulate this particular task.

2. The problem

A sequence of N points is given. A curve must be created, composed of circular arcs, such that:

- it passes through/nearby the given points in the same sequence;
- the Hausdorff distance between the points and the curve does not exceed a certain value ϵ;
- it is composed of minimal number of arcs;
- the output should consist of sets of the type:
  $\{(x_1, y_1), (x_2, y_2), (x_c, y_c), E\}$,

where (x_1, y_1) and (x_2, y_2) are respectively the initial and the final points of a certain arc, (x_c, y_c) is its center and $E = +1$ if the direction of the arc is counter

clockwise or $E = -1$ if the direction of the arc is clockwise.

Remark. Local minimum – fitting an arc to each set of 3 points – is not a solution of the task.

2.1. Summary of the approach

- We begin with a program for finding the center and the radius of a circle that passes through three fixed points.
- Having such a program we make another one for finding the "best" arc that connects two fixed points (which have at least two inner points between them). This arc passes through the two fixed points and through one of the points between them.
- Next we find the "best" arc between any two points (that have at least two inner points) of the set of points we are given.
- From the set of arcs that we have created, we exclude those that do not satisfy our error condition.
- From the arcs that are left we may choose different ways to get from the initial point to the last. We chose such a path that contains minimal number of arcs. Usually the connecting points are spread almost uniformly throughout the set we are given.

2.2. An arc through three fixed points

Let us have the points $P_1(x_1, y_1)$, $P_2(x_2, y_2)$, $P_3(x_3, y_3)$, Fig. 1. The midpoints A and B of the line segments connecting (x_1, y_1) and (x_2, y_2) and (x_2, y_2) and (x_3, y_3) have coordinates (x_A, y_A), (x_B, y_B). Obviously

$$x_A = \frac{x_2 + x_1}{2} \ , \ x_B = \frac{x_3 + x_2}{2}$$

and

$$y_A = \frac{y_2 + y_1}{2} \ , \ y_B = \frac{y_3 + y_2}{2}.$$

The equations of the lines that pass through the points $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ and $P_2(x_2, y_2)$, $P_3(x_3, y_3)$ are respectively

$$l_1 : A_1 x + B_1 y + C_1 = 0$$

and

$$l_2 : A_2 x + B_2 y + C_2 = 0,$$

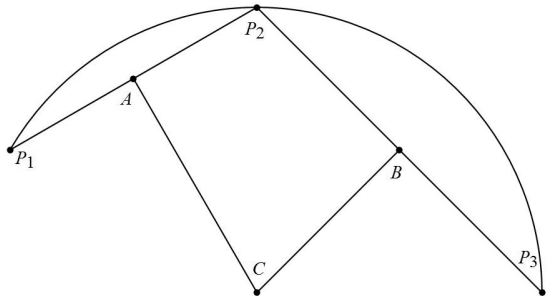


Figure 1: The center C of the circle through P_1, P_2, P_3

where $A_1 = y_2 - y_1$, $B_1 = x_2 - x_1$, $C_1 = -x_1(y_2 - y_1) + y_1(x_2 - x_1)$, $A_2 = y_3 - y_2$, $B_2 = x_3 - x_2$, $C_2 = -x_2(y_3 - y_2) + y_2(x_3 - x_2)$. Now, since the vectors $p_1(A_1, B_1)$ and $p_2(A_2, B_2)$ are orthogonal respectively to the lines l_1 and l_2 and we have the coordinates of A and B, we can easily find the equations of the line bisectors of the arcs that are orthogonal to l_1 and l_2 and pass respectively through (x_A, y_A) and (x_B, y_B). We have

$$b_1 : B_1 x - A_1 y + (-B_1 x_A + A_1 y_A) = 0,$$

$$b_2 : B_2 x - A_2 y + (-B_2 x_B + A_2 y_B) = 0.$$

The center $C(p, q)$ of the circle is where the two line bisectors intersect. Its coordinates are the solution of the system

$$B_1 x - A_1 y + (-B_1 x_A + A_1 y_A) = 0,$$

$$B_2 x - A_2 y + (-B_2 x_B + A_2 y_B) = 0.$$

So we have that

$$p = -\frac{-A_2 B_1 x_A + A_1 B_2 x_B + A_1 A_2 y_A - A_1 A_2 y_B}{A_2 B_1 - A_1 B_2},$$

$$q = -\frac{-B_1 B_2 x_A + B_1 B_2 x_B + A_1 B_2 y_A - A_2 B_1 y_B}{A_2 B_1 - A_1 B_2}$$

As for the radius of the circle, it is equal to the distance between the center and any point on it. We can use the point $P_1(x_1, y_1)$. We have that

$$r = \sqrt{(x_1 - p)^2 + (y_1 - q)^2}.$$

The direction of the arc is positive (negative) exactly when the orientation of the triangle $\overrightarrow{P_1P_2P_3}$ is positive (negative). This orientation is equal to the sign of the determinant

$$\begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_2 & y_3 - y_2 \end{vmatrix}.$$

2.3. "Best" arc

Let us consider the task for connecting two fixed points $P_0(x_0, y_0)$ and $P_{n+1}(x_{n+1}, y_{n+1})$ (which have n inner points, $n \geq 2$) of our input set. First we build all the arcs that connect the two end points and pass through an inner one - that makes n arcs. Let r_i and $C_i(p_i, q_i)$, $i = 1, \ldots, n$ be respectively the radii and the centers of these arcs. For every arc with a center (p_i, q_i) and radius r_i, $(i = 1, \ldots, n)$ we calculate its Hausdorff distance to the inner points P_j, $j = 1, \ldots, n$.

$$d_{i,j} = \left| \sqrt{(x_j - p_i)^2 + (y_j - q_i)^2} - r_i \right|.$$

We now denote

$$d_i := \max\{d_{i,1}, \ldots, d_{i,n}\}.$$

For the i-th arc d_i is its greatest Hausdorff distance to an inner point. We remind that we now consider all the arcs that connect two fixed points and pass through a third between them. For the "best" arc of such kind we chose the k-th arc for which

$$d_k = \min\{d_1, \ldots, d_n\}.$$

"Best" arc – new suggestions.

The input set is the same: two fixed points $P_0(x_0, y_0)$ and $P_{n+1}(x_{n+1}, y_{n+1})$ (which have n inner points, $n \geq 2$). The midpoint M of the segment P_0P_{n+1} has coordinates (x_M, y_M). Obviously

$$x_M = \frac{x_0 + x_{n+1}}{2}, \qquad y_M = \frac{y_0 + y_{n+1}}{2}.$$

The equations of the line that passes through the point M and is perpendicular to the segment P_0P_{n+1} are:

$$c : \begin{cases} x_C = x_M + d * y10/w \\ y_C = y_M + d * x01/w \end{cases}$$

where: $x01 = x_0 - x_{n+1}$, $\quad y10 = y_{n+1} - y_0$, $\quad w^2 = (x01)^2 + (y10)^2$.

For $i = 1, \ldots, n$ we calculate the oriented distance d_i from M to the C_i-center of the circle through the points P_0, P_i, P_{n+1}

$$d_i = \frac{((x_i - x_M)^2 + (y_i - y_M)^2 - w^2/4).w}{2((x_i - x_M).y10 + (y_i - y_M).x01)}, \qquad d = \frac{1}{n}\sum_{i=1}^{n} d_i.$$

Next we define the center C of the optimal arc: C is at distance d from M. The radius of the arc is $r = \sqrt{d^2 + w^2/4}$. We calculate the errors e_i for the points P_i. Note that $e_i = \sqrt{(x_i - x_M)^2 + (y_i - y_M)^2} - r$ is the Euclidean distance between P_i and the point Q_i, which lies on this circle and on the radius through the point P_i. At the same time e_i is the Hausdorff distance between P_i and the optimal arc. More precisely this is one-side Hausdorff distance from given points to the found arc.

2.4. Next stages

Now we consider all the combinations of two points from our input set that have at least two inner points. For all such pairs of points we take the best (according to one of the ways previously described) arc that connects them. Since not all these arcs are close enough to all of their inner points (for an example we can rarely connect the first and last point with only one arc) we exclude those for which the distance between them and their inner points (at least one of them) is more than ϵ. Now we have a set of suitable arcs.
We may consider the problem for constructing a curve (made of arcs) from the first to the last point as a question for finding a path in a graph. We consider each point of the input set as a node and the arcs (connecting some of them and satisfying the error condition) as ribs.

For construction of the adjacency matrix $A = (a_{ij})_{i=1,\ldots,N}^{j=0,\ldots,N}$ we first set A to have only zeros. For $i = 1, \ldots, N-3$ (N is the number of the input points) we consider the best arc (rib) connecting the i-th and the j-th points ($j = i + 3, \ldots, N$). If this arc satisfies the error condition we predefine $a_{ij} = 1$.
We compare different paths by the length of their shortest arc (according to the number of inner points). One approach is to find all the paths in the graph we have derived and then chose the one in which the shortest arc is as long as possible. However, we have adapted an algorithm for finding a path with smallest amount of ribs. Usually the nodes we get are spread uniformly.

3. Numerical experiments

We have applied our approach to real examples. On Figure 2 the black curve consists of 200 points, that lie on the parabolic curve $y = 300 - 200 * (1 - x/500)^2$ and the white inner segments are the arcs (6 is their number), approximate the points.

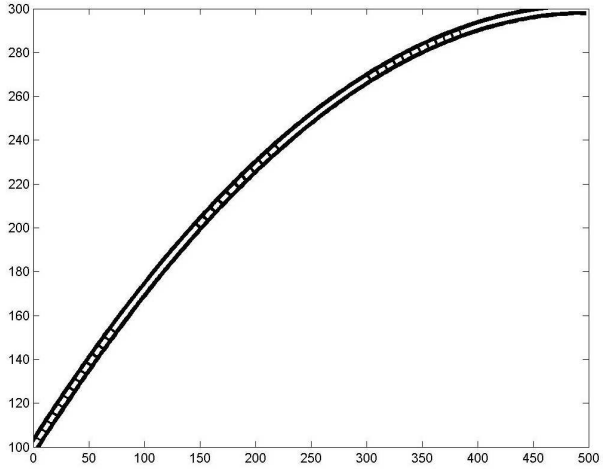


Figure 2: Approximation of the data by 6 arcs

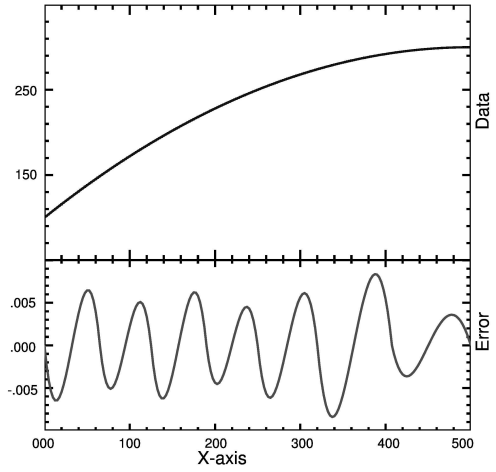


Figure 3: Approximation by 7 arcs (above) and the error of approximation (below)

On Figure 3 we show the approximation of the same data by 7 arcs and below we demonstrate how the error of approximation changes. The maximal error with 5 arcs is about 0.0183, but with 7 arcs – less than 0.0085. The output data for these two cases are:

Number of arcs is $N_{arc} = 5$

```
A (500.000,300.000) (370.000,286.480) (500.58361,  -337.37167)  1
A (370.000,286.480) (270.000,257.680) (514.73312,  -404.07667)  1
A (270.000,257.680) (177.500,216.795) (553.71037,  -509.27917)  1
A (177.500,216.795) ( 85.000,162.220) (628.01623,  -652.46917)  1
A ( 85.000,162.220) (  0.000,100.000) (744.77653,  -828.28417)  1
```

Number of arcs is $N_{arc} = 7$

```
A (500.000,300.000) (407.500,293.155) (500.20991,  -331.25917)  1
A (407.500,293.155) (320.000,274.080) (506.36069,  -370.56000)  1
A (320.000,274.080) (250.000,250.000) (525.32585,  -436.58167)  1
A (250.000,250.000) (190.000,223.120) (556.08512,  -513.63000)  1
A (190.000,223.120) (125.000,187.500) (602.69383,  -607.08750)  1
A (125.000,187.500) ( 65.000,148.620) (669.89912,  -719.13000)  1
A ( 65.000,148.620) (  0.000,100.000) (761.34933,  -850.08750)  1
```

4. Summary

To recap, the problem was how to create a sequence of arcs

- passing through some of the given points and being sufficiently close to the others points,
- arcs must be as long as possible.

We did the following activities:

- examined the problem in the literature,
- developed an algorithm for constructing a sequence of arcs,
- tested our approach with a real data,
- improved the method,
- compared the results.

References

[1] Kazimierz Jakubczyk. Approximation of Smooth Planar Curves by Circular Arc Splines. May 30, 2010 (rev. January 28, 2012)

[2] O. Aichholzer, F. Aurenhammer, T. Hackl, B. Jüttler, M. Oberneder, and Z. Sír. Computational and structural advantages of circular boundary representation.