Accurate estimation of bicarbonate and acetic acid concentrations with wider ranges in anaerobic media using classical FOS/TAC titration method

Xiaojun Liu[1], Laura André[2], Maël Mercier-Huat[1,2,3], Jean-Marie Grosmaître[3], André Pauss[1] and Thierry Ribeiro[2*]

[1] Université de technologie de Compiègne, ESCOM, TIMR (Integrated Transformations of Renewable Matter), Centre de recherches Royallieu, CS 60319, 60203 Compiègne CEDEX, France
[2] Institut Polytechnique UniLaSalle Beauvais, Département des Sciences et Techniques Agro-Industrielles, rue Pierre Waguet, BP 30313, 60026 Beauvais Cedex, France
[3] Cultimer France Producteurs Associés, 35120 Dol-de-Bretagne, France

[SUPPLEMENTARY MATERIALS]

List

Table S1. pK_a of main compounds in an anaerobic digester at 25°C (data extracted from Sun et al., 2016).

Table S2. TAC and FOS values reported in literature and calculated using Nordmann method versus the respective corrected values using new formulas (raw data collected from Jobling Purser et al., 2014).

Figure S1. Programming flowchart for the simulation of FOS/TAC titration curves.

Scilab code 1 for calculation of the sodium bicarbonate contents at different acetic acid concentrations

Scilab code 2 for calculation of the acetic acid contents at different sodium bicarbonate concentrations

***Table S1**. pK_a of main compounds in an anaerobic digester at 25°C (data extracted from Sun et al., 2016 [18]).*

Acid/Base couple	Dissociation reaction	pK_a (–)
Volatile fatty acid		
Acetic acid	$CH_3COOH \rightleftharpoons CH_3COO^- + H^+$	4.76
Propionic acid	$C_2H_5COOH \rightleftharpoons C_2H_5COO^- + H^+$	4.87
Butyric acid	$C_3H_7COOH \rightleftharpoons C_3H_7COO^- + H^+$	4.82
Valeric acid	$C_4H_9COOH \rightleftharpoons C_4H_9COO^- + H^+$	4.82
Alkalinity		
H_2CO_3/HCO_3^-	$H_2CO_3 \rightleftharpoons HCO_3^- + H^+$	6.35
HCO_3^-/CO_3^{2-}	$HCO_3^- \rightleftharpoons CO_3^{2-} + H^+$	10.3

$H_3PO_4/H_2PO_4^-$	$H_3PO_4 \rightleftharpoons H_2PO_4^- + H^+$	2.15
$H_2PO_4^-/HPO_4^{2-}$	$H_2PO_4^- \rightleftharpoons HPO_4^{2-} + H^+$	7.20
HPO_4^{2-}/PO_4^{3-}	$HPO_4^{2-} \rightleftharpoons PO_4^{3-} + H^+$	12.4
NH_4^+/NH_3	$NH_4^+ \rightleftharpoons NH_3 + H^+$	9.24

Table S2*. TAC and FOS values reported in literature and calculated using Nordmann method versus the respective corrected values using new formulas (raw data collected from [16]).*

Sample	**$TAC_{Nordmann}$ (mg $CaCO_3 \cdot L^{-1}$)**	**$FOS_{Nordmann}$ (mg HAc·L^{-1})**	**$FOS/TAC_{Nordmann}$ (–)**	**TAC_{new} (mg $CaCO_3 \cdot L^{-1}$)**	**FOS_{new} (mg HAc·L^{-1})**	**FOS/TAC-$_{new}$ (–)**
Energy crop and slurry No. 121	10,800	20,778	1.92	4,277	22,233	5.20
Food waste No. 45	18,252	2,692	0.15	18,778	813	0.04

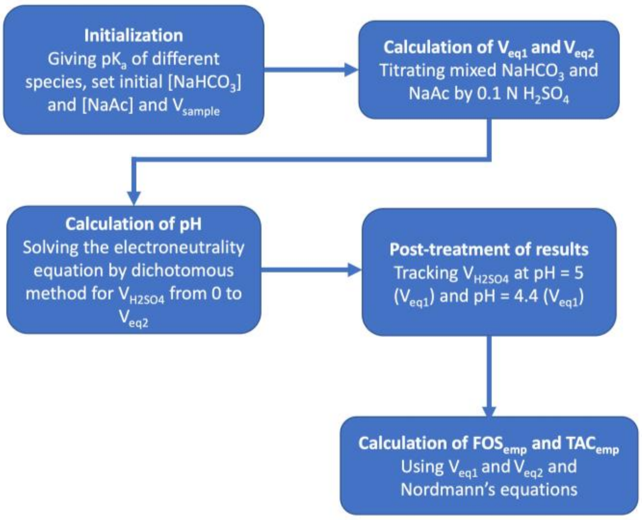
Initialization
Giving pK_a of different species, set initial [$NaHCO_3$] and [NaAc] and V_{sample}
Calculation of V_{eq1} and V_{eq2}
Titrating mixed $NaHCO_3$ and NaAc by 0.1 N H_2SO_4
Calculation of pH
Solving the electroneutrality equation by dichotomous method for V_{H2SO4} from 0 to V_{eq2}
Post-treatment of results
Tracking V_{H2SO4} at pH = 5 (V_{eq1}) and pH = 4.4 (V_{eq1})
Calculation of FOS_{emp} and TAC_{emp}
Using V_{eq1} and V_{eq2} and Nordmann's equations

Figure S1. *Programming flowchart for the simulation of FOS/TAC titration curves.*

Scilab code 1 for calculation of the sodium bicarbonate contents at different acetic acid concentrations

```
// titration_FOS_TAC_Bicarbonate-simul_vs_Bicarbonate-theor.sce
//  English version made on 08 november 2021
//  Accurate estimation of bicarbonate and acetic acid concentrations with wider ranges in anaerobic media using classical FOS/TAC titration method // Authors : X. Liu, M. Mercier-Huat, L. André, A. Pauss and T. Ribeiro // Université de Technologie de Compiègne (France) and Institut Polytechnique UniLaSalle (France)
//  Simulation of NaHCO3 contents by Nordmann's empirical formulas (FOS / TAC) and our newly proposed formulas for different theoretical NaHCO3 contents at different HAc levels
//  This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

//        ! IMPORTANT !
// To run the code, you must install fmincon from the Scilab command line (and restart Scilab): atomsInstall fmincon

// All mass concentrations are expressed in mg/L

clear
funcprot(0)
clc

function [HAc, Ac, CO2, HCO3, CO3, zp]=HAc_bicarbonate(h)
    HAc=conc_HAc.*h./(h+Ka_HAc)
     Ac=conc_HAc*Ka_HAc./(h+Ka_HAc)
    deno_Ci=(h.*h)+(h.*Ka1_Ci)+(Ka1_Ci*Ka2_Ci)
     CO2=conc_Ci.*h.*h./deno_Ci
     HCO3=conc_Ci*Ka1_Ci.*h./deno_Ci
     CO3=conc_Ci*Ka1_Ci*Ka2_Ci./deno_Ci
   zp=(h+conc_Na)-((1e-14./h)+Ac+HCO3+2.*CO3+2*conc_H2SO4)
endfunction

HAc_data=[0 50 100 500 1000 3000 5000 7000 10000 15000 20000 40000]  //mg/L
Bicarbonate_data=[0 50 100 500 1000 3000 5000 7000 10000 15000 20000 40000] //mg/L
N_data=length(Bicarbonate_data)
Ka1_Ci=10^(-6.35);Ka2_Ci=10^(-10.35)
Ka_HAc=10^(-4.76)
pH_init=8

////  Conditions of titration
txt=['Sample volume (mL)';'Water volume for dilution (mL)';'Concentration of the titrant - H2SO4 (mol/L)';'First pH equivalence point (-)';'Second pH equivalence point (-)';'Number of steps of titration (min 5000)'];
sig=x_mdialog('Enter the values, or accept the default values  (!! use the point as decimal separator !!)',txt,['5';'15';'0.05';'5';'4.4';'7000'])
volume_sample=evstr(sig(1))
volume_water_dilution=evstr(sig(2))
conc_init_H2SO4=evstr(sig(3))
high_endpoint=evstr(sig(4))
low_endpoint=evstr(sig(5))
N=evstr(sig(6))

for kk=1:length(HAc_data)
    for j=1:N_data
        conc_mass_init_Ci=Bicarbonate_data(j)  // mg/L
        conc_mass_init_HAc=HAc_data(kk) // mg/L

        // convert mg/L to mol/L
        conc_init_Ci=conc_mass_init_Ci/84/1000  // mol/L
        conc_init_HAc=conc_mass_init_HAc/60/1000  // mol/L
        conc_init_Na=conc_init_Ci+conc_init_HAc

        Veq = (conc_init_Ci+conc_init_HAc)*volume_sample/2/conc_init_H2SO4
        if  HAc_data(kk)<=50;Veq=Veq*2;end  // extention of titration in the absence of HAc
        if Veq==0 // avoid Veq=0 for [HAc]=0 and [Bicarbonate]=0
```

```
         Veq=10^(-3)*volume_sample/2/conc_init_H2SO4 // titration until pH=3
      end

      /// calculation of concentration with dilution factors
      volume_H2SO4=linspace(0,Veq,N)
      dilution_factor=volume_sample./(volume_sample+volume_H2SO4+volume_water_dilution)
      conc_Ci=conc_init_Ci.*dilution_factor
      conc_HAc=conc_init_HAc.*dilution_factor
      conc_Na=conc_init_Na.*dilution_factor
      conc_H2SO4=conc_init_H2SO4.*volume_H2SO4./(volume_sample+volume_H2SO4+volume_water_dilution)

      /// calculation of pH during the titration by a dichotomous method
      pH=pH_init*ones(1,N)
      for z=1:20
         [HAc,Ac,CO2,HCO3,CO3,zp]=HAc_bicarbonate(10^(-pH))
         pH=pH+sign(zp)*14*0.5^z
      end

      // identification of the equivalent volumes corresponding to the 2 reference pH points of the titration
      for ii=1:N
         while pH(ii)>high_endpoint
            ii=ii+1
         end
         Veq1=(volume_H2SO4(ii)+volume_H2SO4(ii-1))/2
         pH_Veq1=(pH(ii)+pH(ii-1))/2
         while pH(ii)>low_endpoint
            ii=ii+1
         end
         Veq2=(volume_H2SO4(ii)+volume_H2SO4(ii-1))/2
         pH_Veq2=(pH(ii)+pH(ii-1))/2
         break
      end

// Calculation of [HAc] et [bicarbonate] for HACH titrateur using Nordmann method (note that the sample volume must be 5 mL)
      TAC_old(j,kk)=Veq1*50045*2*conc_init_H2SO4/volume_sample
      Bicarbonate_old(j,kk)=84/100* TAC_old(j,kk)*2
      HAc_old(j,kk)=(((Veq2-Veq1)*4*1.66)-0.15)*500  // 1.66 et 0.15 are purely empirical factors given by McGhee (1968)

      Veq1_titration(j,kk)=Veq1 // to export the results
      DeltaV_titration(j,kk)=Veq2-Veq1
   end
end

// identification of model parameters
function y=f(x, V)
   y=x(1)*V(1)+x(2)*V(2)+x(3) // Model: [NaHCO3]new = Bicarbonate_new_a * Veq1 + Bicarbonate_new_b * (Veq2-Veq1) in mg/L
endfunction

function SSE=regres_f(x, V, y_theo)
   e=f(x,V)-y_theo
   SSE=sum(e.^2)
endfunction

      opt = optimoptions("fmincon"); // fmincon optimization method
      V=list(Veq1_titration,DeltaV_titration)
      Bicarbonate_theo=zeros(length(Bicarbonate_data),length(HAc_data))

      for i=1:length(Bicarbonate_data)
         Bicarbonate_theo(i,:)=Bicarbonate_data(i)
```

```
    end
    Bicarbonate_new_a0=1000 // initial values for regression
    Bicarbonate_new_b0=-1000
    Bicarbonate_new_c0=0
    x0_f=[Bicarbonate_new_a0 Bicarbonate_new_b0 Bicarbonate_new_c0]

    problem_f = struct();
    problem_f.objective = list(regres_f,V,Bicarbonate_theo);
    problem_f.x0 = x0_f;
    problem_f.options = opt;
    thetaopt_f = fmincon(problem_f);
    Bicarbonate_new_a=thetaopt_f(1)
    Bicarbonate_new_b=thetaopt_f(2)
    Bicarbonate_new_c=thetaopt_f(3)
    SSE=regres_f(thetaopt_f,V,Bicarbonate_theo)
    e_tot=Bicarbonate_theo-mean(Bicarbonate_theo)
    SST=sum(e_tot.^2)
    R2=1-SSE/SST

//  show modeling results
format(7)
    disp('pH1 = '+string(high_endpoint)+'    pH2 = '+string(low_endpoint))
    disp('Bicarbonate_new_a = '+string(thetaopt_f(1))+'  Bicarbonate_new_b = '+string(thetaopt_f(2))+'  Bicarbonate_new_c = '+string(thetaopt_f(3)))
    disp('[NaHCO3] = '+string(thetaopt_f(1))+' * A '+string(thetaopt_f(2))+' * B + '+ string(thetaopt_f(3)) +' ([NaHCO3] in mg/L)')
    disp('with A = Veq1 and B = Veq2 - Veq1    (A and B in mL)')
    disp('R2 = '+string(R2))

    Bicarbonate_old_to_new_a=Bicarbonate_new_a./1000.9
    Bicarbonate_old_to_new_b=Bicarbonate_new_b./3320
    Bicarbonate_old_to_new_c=75*Bicarbonate_new_b./3320+Bicarbonate_new_c
    disp('[NaHCO3] old to new = '+string(Bicarbonate_old_to_new_a)+' * TAC old '+string(Bicarbonate_old_to_new_b)+' * FOS old '+string(Bicarbonate_old_to_new_c)+'    ([NaHCO3] in mg/L)')
    disp('TAC old to new = '+string(0.595*Bicarbonate_old_to_new_a)+' * TAC old '+string(0.595*Bicarbonate_old_to_new_b)+' * FOS old '+string(0.595*Bicarbonate_old_to_new_c)+'    (TAC in mg/L)')

// Calculate relative errors and Mean Absolute Percentage Error (MAPE)
  sum_relative_errors_new=0 // initial values to calculate the relative errors
  sum_relative_errors_old=0
  sum_relative_errors_corr=0

  relative_error_new(1,:)=0 // We do not calculate the relative errors for [bicarbonate] = 0
  relative_error_old(1,:)=0
  relative_error_corr(1,:)=0

for kk=1:length(HAc_data)
  for j=1:N_data
    Bicarbonate_new(j,kk)=(Veq1_titration(j,kk)*Bicarbonate_new_a)+(DeltaV_titration(j,kk)*Bicarbonate_new_b)+Bicarbonate_new_c
    Bicarbonate_old_to_new(j,kk)=(TAC_old(j,kk)*Bicarbonate_old_to_new_a)+(HAc_old(j,kk)*Bicarbonate_old_to_new_b)+Bicarbonate_old_to_new_c

    e_new(j,kk)=Bicarbonate_new(j,kk)-Bicarbonate_data(j)
    e_old(j,kk)=Bicarbonate_old(j,kk)-Bicarbonate_data(j)
    e_corr(j,kk)=Bicarbonate_old_to_new(j,kk)-Bicarbonate_data(j)

    if Bicarbonate_data(j)>0
      relative_error_new(j,kk)=abs(e_new(j,kk))/Bicarbonate_data(j)*100  //in %
      relative_error_old(j,kk)=abs(e_old(j,kk))/Bicarbonate_data(j)*100  //in %
      relative_error_corr(j,kk)=abs(e_corr(j,kk))/Bicarbonate_data(j)*100   //in %


      sum_relative_errors_new=sum_relative_errors_new+relative_error_new(j,kk)   //in %
      sum_relative_errors_old=sum_relative_errors_old+relative_error_old(j,kk)  //in %
      sum_relative_errors_corr=sum_relative_errors_corr+relative_error_corr(j,kk) //in %
    end
  end
end

  MAPE_new_theoretical_Bicarbonate=sum_relative_errors_new/((length(Bicarbonate_data)-1)*length(HAc_data)-1)//We exclude Bicarbonate_data(1,:) which is equal to 0
  MAPE_old_theoretical_Bicarbonate=sum_relative_errors_old/((length(Bicarbonate_data)-1)*length(HAc_data)-1)
  MAPE_corr_theoretical_Bicarbonate=sum_relative_errors_corr/((length(Bicarbonate_data)-1)*length(HAc_data)-1)

format(7)
  disp('MAPE of [NaHCO3]new vs [NaHCO3]theoretical = '+string(MAPE_new_theoretical_Bicarbonate)+'%')
  disp('MAPE of [NaHCO3]old vs [NaHCO3]theoretical = '+string(MAPE_old_theoretical_Bicarbonate)+'%')
  disp('MAPE of [NaHCO3]corr vs [NaHCO3]theoretical = '+string(MAPE_corr_theoretical_Bicarbonate)+'%')

//  put simulation results in graphics
bisector=[0  0
Bicarbonate_data($) Bicarbonate_data($)]

clf()
subplot(131)
  plot(Bicarbonate_data',Bicarbonate_old,'*')
  plot(bisector(:,1),bisector(:,2),'k--')
  replot([0,0,Bicarbonate_data($)*1.05,Bicarbonate_data($)*1.5])
  xtitle('Calculated [NaHCO3] versus theoretical [NaHCO3], Nordmann','Theoretical [NaHCO3] (mg/L)','[NaHCO3] calculated with the empirical formula of Nordmann (mg/L)')
  legend('HAc= '+string(HAc_data(1))+' mg/L','HAc= '+string(HAc_data(2))+' mg/L','HAc= '+string(HAc_data(3))+' mg/L','HAc= '+string(HAc_data(4))+' mg/L','HAc= '+string(HAc_data(5))+' mg/L','HAc= '+string(HAc_data(6))+' mg/L','HAc= '+string(HAc_data(7))+' mg/L','HAc= '+string(HAc_data(8))+' mg/L','HAc= '+string(HAc_data(9))+' mg/L','HAc= '+string(HAc_data(10))+' mg/L','HAc= '+string(HAc_data(11))+' mg/L','HAc= '+string(HAc_data(12))+' mg/L',pos=2)

subplot(132)
  plot(Bicarbonate_data',Bicarbonate_new,'*')
  plot(bisector(:,1),bisector(:,2),'k--')
  replot([0,0,Bicarbonate_data($)*1.05,Bicarbonate_data($)*1.5])
  xtitle('Calculated [NaHCO3] versus theoretical [NaHCO3], new formula','Theoretical [NaHCO3] (mg/L)','[NaHCO3] calculated with new formula (mg/L)')
  legend('HAc= '+string(HAc_data(1))+' mg/L','HAc= '+string(HAc_data(2))+' mg/L','HAc= '+string(HAc_data(3))+' mg/L','HAc= '+string(HAc_data(4))+' mg/L','HAc= '+string(HAc_data(5))+' mg/L','HAc= '+string(HAc_data(6))+' mg/L','HAc= '+string(HAc_data(7))+' mg/L','HAc= '+string(HAc_data(8))+' mg/L','HAc= '+string(HAc_data(9))+' mg/L','HAc= '+string(HAc_data(10))+' mg/L','HAc= '+string(HAc_data(11))+' mg/L','HAc= '+string(HAc_data(12))+' mg/L',pos=2)


subplot(133)
  plot(Bicarbonate_data',Bicarbonate_old_to_new,'*')
  plot(bisector(:,1),bisector(:,2),'k--')
  replot([0,0,Bicarbonate_data($)*1.05,Bicarbonate_data($)*1.5])
  xtitle('Calculated [NaHCO3] versus theoretical [NaHCO3], correction of Nordmann values by new formula','Theoretical [NaHCO3] (mg/L)','Empirical [NaHCO3] corrected by new formula (mg/L)')
  legend('HAc= '+string(HAc_data(1))+' mg/L','HAc= '+string(HAc_data(2))+' mg/L','HAc= '+string(HAc_data(3))+' mg/L','HAc= '+string(HAc_data(4))+' mg/L','HAc= '+string(HAc_data(5))+' mg/L','HAc= '+string(HAc_data(6))+' mg/L','HAc= '+string(HAc_data(7))+' mg/L','HAc= '+string(HAc_data(8))+' mg/L','HAc= '+string(HAc_data(9))+' mg/L','HAc= '+string(HAc_data(10))+' mg/L','HAc= '+string(HAc_data(11))+' mg/L','HAc= '+string(HAc_data(12))+' mg/L',pos=2)
```

```
show_window

//end of code
```

Scilab code 2 for calculation of the acetic acid contents at different sodium bicarbonate concentrations

```
//  titration_FOS_TAC_HAc-simul_vs_HAc-theor.sce
//  English version made on 08 november 2021
//  Accurate estimation of bicarbonate and acetic acid concentrations with wider ranges in anaerobic media using classical FOS/TAC titration method // Authors : X. Liu, M. Mercier-Huat, L. André, A. Pauss and T. Ribeiro // Université de Technologie de Compiègne (France) and Institut Polytechnique UniLaSalle (France)
//  Simulation of HAc contents by Nordmann's empirical formulas (FOS / TAC) and our newly proposed formulas for different theoretical HAc contents et different NaHCO3 levels
//  This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

//        ! IMPORTANT !
// To run the code, you must install fmincon from the Scilab command line (and restart Scilab): atomsInstall fmincon

// All mass concentrations are expressed in mg/L

clear
funcprot(0)
clc

function [HAc, Ac, CO2, HCO3, CO3, zp]=HAc_bicarbonate(h)
   HAc=conc_HAc.*h./(h+Ka_HAc)
    Ac=conc_HAc*Ka_HAc./(h+Ka_HAc)
   deno_Ci=(h.*h)+(h.*Ka1_Ci)+(Ka1_Ci*Ka2_Ci)
    CO2=conc_Ci.*h.*h./deno_Ci
    HCO3=conc_Ci*Ka1_Ci.*h./deno_Ci
    CO3=conc_Ci*Ka1_Ci*Ka2_Ci./deno_Ci
   zp=(h+conc_Na)-((1e-14./h)+Ac+HCO3+2.*CO3+2*conc_H2SO4)
endfunction

HAc_data=[0 50 100 500 1000 3000 5000 7000 10000 15000 20000 40000]  //mg/L
Bicarbonate_data=[0 50 100 500 1000 3000 5000 7000 10000 15000 20000 40000] //mg/L
N_data=length(HAc_data)
Ka1_Ci=10^(-6.35);Ka2_Ci=10^(-10.35)
Ka_HAc=10^(-4.76)
pH_init=8

//  Conditions of titration
txt=['Sample volume (mL)';'Water volume for dilution (mL)';'Concentration of the titrant - H2SO4 (mol/L)';'First pH equivalence point (-)';'Second pH equivalence point (-)';'Number of steps of titration (min 5000)'];
sig=x_mdialog('Enter the values, or accept the default values  (!! use the point as decimal separator !!)',txt,['5';'15';'0.05';'5';'4.4';'7000'])
volume_sample=evstr(sig(1))
volume_water_dilution=evstr(sig(2))
conc_init_H2SO4=evstr(sig(3))
high_endpoint=evstr(sig(4))
low_endpoint=evstr(sig(5))
N=evstr(sig(6))

for kk=1:length(Bicarbonate_data)
   for j=1:N_data
      conc_mass_init_Ci=Bicarbonate_data(kk)  // mg/L
      conc_mass_init_HAc=HAc_data(j) //mg/L

      // convert mg/L to mol/L
      conc_init_Ci=conc_mass_init_Ci/84/1000  // mol/L
      conc_init_HAc=conc_mass_init_HAc/60/1000  // mol/L
      conc_init_Na=conc_init_Ci+conc_init_HAc
      Veq = (conc_init_Ci+conc_init_HAc)*volume_sample/2/conc_init_H2SO4
      if  HAc_data(j)<=50;Veq=Veq*2;end  // extention of titration in the absence of bicarbonate
      if Veq==0 // avoid Veq=0 for [HAc]=0 and [Bicarbonate]=0
        Veq=10^(-3)*volume_sample/2/conc_init_H2SO4 // titration until pH=3
      end

      /// calculation of concentration with dilution factors
      volume_H2SO4=linspace(0,Veq,N)
      dilution_factor=volume_sample./(volume_sample+volume_H2SO4+volume_water_dilution)
      conc_Ci=conc_init_Ci.*dilution_factor
      conc_HAc=conc_init_HAc.*dilution_factor
      conc_Na=conc_init_Na.*dilution_factor
      conc_H2SO4=conc_init_H2SO4.*volume_H2SO4./(volume_sample+volume_H2SO4+volume_water_dilution)

      /// calculation of pH during the titration by a dichotomous method
      pH=pH_init*ones(1,N)
      for z=1:20
        [HAc,Ac,CO2,HCO3,CO3,zp]=HAc_bicarbonate(10^(-pH))
        pH=pH+sign(zp)*14*0.5^z
      end

//   identification of the equivalent volumes corresponding to the 2 reference pH points of the titration
      for ii=1:N
        while pH(ii)>high_endpoint
          ii=ii+1
        end
        Veq1=(volume_H2SO4(ii)+volume_H2SO4(ii-1))/2
        pH_Veq1=(pH(ii)+pH(ii-1))/2
        while pH(ii)>low_endpoint
          ii=ii+1
        end
        Veq2=(volume_H2SO4(ii)+volume_H2SO4(ii-1))/2
        pH_Veq2=(pH(ii)+pH(ii-1))/2
        break
      end

// Calculation of [HAc] et [bicarbonate] for HACH titrateur using Nordmann method (note that the sample volume must be 5 mL)
      TAC_old(j,kk)=Veq1*50045*2*conc_init_H2SO4/volume_sample
      HAc_old(j,kk)=(((Veq2-Veq1)*4*1.66)-0.15)*500  // 1.66 et 0.15 are purely empirical factors given by McGhee (1968)
      Veq1_titration(j,kk)=Veq1  // to export the results
      DeltaV_titration(j,kk)=Veq2-Veq1
   end
end

//  identification of model parameters
function y=f(x, V)
   y=x(1)*V(1)+x(2)*V(2)+x(3) // // Model: [HAc]new = HAc_new_a * Veq1 + HAc_new_b * (Veq2-Veq1) + HAc_new_c in mg/L
endfunction

function SSE=regres_f(x, V, y_theo)
   e=f(x,V)-y_theo
   SSE=sum(e.^2)
endfunction

      opt = optimoptions("fmincon"); // fmincon optimization method
      V=list(Veq1_titration,DeltaV_titration)
        HAc_theo=zeros(length(HAc_data),length(Bicarbonate_data))
      for i=1:length(HAc_data)
        HAc_theo(i,:)=HAc_data(i)
      end

      HAc_new_a0=0 // initial values for regression
      HAc_new_b0=1000
      HAc_new_c0=0
      x0_f=[HAc_new_a0 HAc_new_b0 HAc_new_c0]

      problem_f = struct();
      problem_f.objective = list(regres_f,V,HAc_theo);
      problem_f.x0 = x0_f;
      problem_f.options = opt;
      thetaopt_f = fmincon(problem_f);
      HAc_new_a=thetaopt_f(1)
      HAc_new_b=thetaopt_f(2)
      HAc_new_c=thetaopt_f(3)
      SSE=regres_f(thetaopt_f,V,HAc_theo)
      e_tot=HAc_theo-mean(HAc_theo)
      SST=sum(e_tot.^2)
      R2=1-SSE/SST

//  show modeling results
format(6)
      disp('pH1 = '+string(high_endpoint)+'    pH2 = '+string(low_endpoint))
      disp('HAc_new_a = '+string(thetaopt_f(1))+'  HAc_new_b = '+string(thetaopt_f(2))+'  HAc_new_c = '+string(thetaopt_f(3)))
      disp('[HAc] = '+string(thetaopt_f(1))+' * A + '+string(thetaopt_f(2))+' * B '+string(thetaopt_f(3))+'  ([HAc] in mg/L)')
      disp('with A = Veq1 and B = Veq2 - Veq1    (A and B in mL)')
      disp('R2 = '+string(R2))

      HAc_old_to_new_a=HAc_new_a./1000.9
      HAc_old_to_new_b=HAc_new_b./3320
      HAc_old_to_new_c=75*HAc_new_b./3320+HAc_new_c
      disp('FOS old to new = '+string(HAc_old_to_new_a)+' * TAC old +'+string(HAc_old_to_new_b)+' * FOS old +
'+string(HAc_old_to_new_c)+'    ([HAc] in mg/L)')

// Calculate relative errors and Mean Absolute Percentage Error (MAPE)
   sum_relative_errors_new=0 // initial values to calculate the relative errors
   sum_relative_errors_old=0
   sum_relative_errors_corr=0

   relative_errors_new(1,:)=0  // We do not calculate the relative errors for [bicarbonate] = 0
   relative_errors_old(1,:)=0
   relative_errors_corr(1,:)=0

for kk=1:length(Bicarbonate_data)
   for j=1:N_data
      HAc_new(j,kk)=Veq1_titration(j,kk)*HAc_new_a+DeltaV_titration(j,kk)*HAc_new_b+HAc_new_c
      HAc_old_to_new(j,kk)=(TAC_old(j,kk)*HAc_old_to_new_a)+(HAc_old(j,kk)*HAc_old_to_new_b)+HAc_old_to_new_c

      e_new(j,kk)=HAc_new(j,kk)-HAc_data(j)
      e_old(j,kk)=HAc_old(j,kk)-HAc_data(j)
      e_corr(j,kk)=HAc_old_to_new(j,kk)-HAc_data(j)

      if HAc_data(j)>0
        relative_errors_new(j,kk)=abs(e_new(j,kk))/HAc_data(j)*100 //in %
        relative_errors_old(j,kk)=abs(e_old(j,kk))/HAc_data(j)*100  //in %
        relative_errors_corr(j,kk)=abs(e_corr(j,kk))/HAc_data(j)*100    //in %
```

```
         sum_relative_errors_new=sum_relative_errors_new+relative_errors_new(j,kk)   //in %
         sum_relative_errors_old=sum_relative_errors_old+relative_errors_old(j,kk) //in %
         sum_relative_errors_corr=sum_relative_errors_corr+relative_errors_corr(j,kk)    //in %
      end
   end
end

   MAPE_HAc_new_theoretical=sum_relative_errors_new/(length(Bicarbonate_data)*length(HAc_data)-1) //We exclude HAc_data(1,:) which is equal to 0
   MAPE_HAc_old_theoretical=sum_relative_errors_old/(length(Bicarbonate_data)*(length(HAc_data)-1)-1)
   MAPE_HAc_corr_theoretical=sum_relative_errors_corr/(length(Bicarbonate_data)*(length(HAc_data)-1)-1)

format(6)
   disp('MAPE of [HAc]new vs [HAc]theoretical = '+string(MAPE_HAc_new_theoretical)+'%')
   disp('MAPE of [HAc]old vs [HAc]theoretical = '+string(MAPE_HAc_old_theoretical)+'%')
   disp('MAPE of [HAc]corr vs [HAc]theoretical = '+string(MAPE_HAc_corr_theoretical)+'%')

//  put simulation results in graphics
bisector=[0  0
HAc_data($) HAc_data($)]

clf()
subplot(131)
   plot(HAc_data',HAc_old,'*')
   plot(bisector(:,1),bisector(:,2),'k--')
   replot([0,0,HAc_data($)*1.05,HAc_data($)*1.05])
   xtitle('Calculated [HAc] versus theoretical [HAc], Nordmann','Theoretical [HAc] (mg/L)','[HAc] calculated with the empirical formula of Nordmann (mg/L)')
   legend('[NaHCO3]= '+string(Bicarbonate_data(1))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(2))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(3))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(4))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(5))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(6))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(7))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(8))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(9))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(10))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(11))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(12))+' mg/L',pos=2)

subplot(132)
   plot(HAc_data',HAc_new,'*')
   plot(bisector(:,1),bisector(:,2),'k--')
   replot([0,0,HAc_data($)*1.05,HAc_data($)*1.05])
   xtitle('Calculated [HAc] versus theoretical [HAc], new formula','Theoretical [HAc] (mg/L)','[HAc] calculated with new formula (mg/L)')
   legend('[NaHCO3]= '+string(Bicarbonate_data(1))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(2))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(3))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(4))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(5))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(6))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(7))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(8))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(9))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(10))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(11))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(12))+' mg/L',pos=2)

subplot(133)
   plot(HAc_data',HAc_old_to_new,'*')
   plot(bisector(:,1),bisector(:,2),'k--')
   replot([0,0,HAc_data($)*1.05,HAc_data($)*1.05])
   xtitle('Calculated [HAc] versus theoretical HAc, correction of Nordmann values by new formula','Theoretical [HAc] (mg/L)','Empirical [HAc] corrected by new formula (mg/L)')
   legend('[NaHCO3]= '+string(Bicarbonate_data(1))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(2))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(3))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(4))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(5))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(6))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(7))+' mg/L','[Na-
HCO3]= '+string(Bicarbonate_data(8))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(9))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(10))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(11))+' mg/L','[NaHCO3]= '+string(Bicarbonate_data(12))+' mg/L',pos=2)

show_window

//end of code
```