9-23-2019

Solution of Simultaneous Chemical Equilibria in Heterogeneous Systems: Implementation in Matlab

D. Scott Smith
Wilfrid Laurier University, ssmith@wlu.ca

Solution of simultaneous chemical equilibria in heterogeneous systems: implementation in Matlab

D. Scott Smith [a]

[a] *Wilfrid Laurier University, Waterloo, Ontario, Canada, N2L 3C5*

23 September 2019

Abstract

A Matlab script to solve simultaneous equilibria in solution, including precipitation/dissolution equilibria is described.

Key words: chemical equilibrium, matlab, Newton-Raphson, heterogeneous equilibria

1 Chemical Speciation

Solving for the equilibrium position of a set of simultaneous reactions subject to the constraints of mass balance and mass action is a common problem in environmental modeling. There exist many available computer programs to solve these types of chemical equilibrium problems, examples include MINEQL (Allison et al., 1991) and PHREEQ (Saini-Eidukat and Yahin, 1999). For engineering and scientific practices though these stand-alone programs are not necessarily completely useful. Often, engineers and scientists will use so-called high-level programming languages such as Matlab or Scilab to implement their larger scale problems. Problems such as reactive transport, or simulation of industrial scale processes can be "coded" into Matlab in a very flexible task-specific manner.

Many processes require the solution of simultaneous equilibria and thus the speciation code presented here was developed for Matlab. In the researcher's lab this code is used to fit experimental data to chemical equilibrium binding constants for metal/organic and metal/phosphate systems but with slight modifications the code could be embedded in Matlab code for other applications (such as transport modeling).

Email address: ssmith@wlu.ca (D. Scott Smith).

The Matlab code was written based on the paper by (Carrayrou et al., 2002). The Matlab code is available at the following link and any interested users are free to modify it as they see fit. An example calculation for the Fe(III) system in water is presented below.

2 Iron(III) system

Mass balance and mass action for solving chemical equilibrium problems can all be represented in Tableau notation (Smith and Ferris, 2001; Morel and Hering, 1993). For a given defined system it is desirable to determine the equilibrium concentration of all species. Using the iron (III) system as an example a list of species of interest could include H^+, OH^-, Fe^{3+}, $FeOH^{2+}$, $Fe(OH)_2{}^+$, and $Fe(OH)_4{}^-$. There are other possible species but to keep the discussion simple these are the only ones included here. Also, for dilute solutions water can be assumed as a fixed component. The list includes six species so it is necessary to define six relationships in order to solve this system. The situation can be simplified if it is realized that each of these species are not independent. We can select components from the list of species and use those components to solve the equilibrium problem. For example, if we know H^+ (pH) and Fe^{3+} we can determine the concentration of all other species from their logK values (mass action). In matrix notation we think of pH and Fe^{3+} as spanning the basis set.

Now we need two equations and two unknowns. First relation is mass balance of iron and the second is proton balance (related to electroneutrality). Now we can write the tableau:

H	Fe	logK	species
1	0	0	H^+
0	1	0	Fe^{3+}
-1	0	-14	OH^-
-1	1	-2.19	$FeOH^{2+}$
-2	1	-5.67	$Fe(OH)_2{}^+$
-4	1	-21.6	$Fe(OH)_4{}^-$
TOTH	Fe_T		

Table 1
Tableau for Fe–H system

Table 1 completely defines the equilibrium problem. The entries in the columns are the stoichiometric coefficients required for formation of each species. For example, there is 1 H^+ and 0 Fe^{3+} in H^+. Also, $Fe(OH)_4{}^-$ is formed with one iron and removing 4 protons from water. Given the tableau, all that remains is to determine the values for $[H^+]$ and $[Fe^{3+}]$ for specified TOTH and Fe_T.

If you multiply across rows it is possible to determine species concentration and if you sum

down columns the total values (mass balance) are recovered. This is best understood by writing out some entries:

$$[H^+] = [H^+]^1 \times [Fe^{3+}]^0 \times 10^0 \tag{1}$$
$$[Fe^{3+}] = [H^+]^0 \times [Fe^{3+}]^1 \times 10^0 \tag{2}$$
$$[OH^-] = [H^+]^{-1} \times [Fe^{3+}]^0 \times 10^{-14} \tag{3}$$
$$[FeOH^{2+}] = [H^+]^{-1} \times [Fe^{3+}]^1 \times 10^{-2.19} \tag{4}$$
$$.... \tag{5}$$

It is necessary to always write the formation reactions for each species from the components. For example to understand equation (3) consider that K_w corresponds to the following reaction:

$$H_2O \rightleftharpoons H^+ + OH^- \quad K_w = [H^+][OH^-] \tag{6}$$

If equation (6) is rearranged to solve for [OH−] then equation (3) is obtained. As another example, to understand equation (4) consider that K_{H1} for the first hydrolysis of iron (III) corresponds to the following reaction:

$$Fe^{3+} + H_2O \rightleftharpoons FeOH^{2+} + H^+ \quad K_{H1} = \frac{[FeOH^{2+}][H^+]}{[Fe^{3+}]} \tag{7}$$

if equation (7) is rearranged to solve for $[FeOH^{2+}]$ then equation (4) is obtained. To explain how the summation down the columns is the mass balance. Consider total iron ....

$$Fe_T = 0[H^+] + 1[Fe^{3+}] + 0[OH^-] + 1[FeOH^{2+}] + 1[Fe(OH)_2{}^+] + 1[Fe(OH)_4{}^-] \tag{8}$$

notice that the coefficients are the entries down the iron column in the tableau.

The problem can easily be expressed in matrix notation. We'll define the 1×2 vector of unknown component concentrations as $\mathbf{X}$ so we can write $\mathbf{X} = \begin{pmatrix} \log[H^+] & \log[Fe^{3+}] \end{pmatrix}$. There is a

6×1 vector of species concentrations as well

$$\mathbf{C} = \begin{pmatrix} [\mathrm{H}^+] \\ [\mathrm{Fe}^{3+}] \\ [\mathrm{OH}-] \\ [\mathrm{FeOH}^{2+}] \\ [\mathrm{Fe(OH)_2}^+] \\ [\mathrm{Fe(OH)_4}-] \end{pmatrix} \tag{9}$$

Total concentrations are in a 2×1 vector $\mathbf{T} = \begin{pmatrix} \mathrm{TOTH} \\ \mathrm{Fe_T} \end{pmatrix}$. The logK values are summarized in the 6×1 vector

$$\mathbf{K} = \begin{pmatrix} 0 \\ 0 \\ \log K_w \\ \log K_{H1} \\ \log K_{H2} \\ \log K_{H4} \end{pmatrix} \tag{10}$$

Finally, we need a 6×2 matrix of stoichiometric coefficients

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ -1 & 1 \\ -2 & 1 \\ -4 & 1 \end{pmatrix} \tag{11}$$

Now the minimization problem is to determine $\mathbf{X}$ that minimizes the residuals in the mass balance. This calculation is performed as follows:

$$\text{minimize } \mathbf{R} \text{ as a function of } \mathbf{X} \text{ where} \quad \mathbf{R} = \mathbf{A}' \times (10^{\mathbf{C}}) - \mathbf{T} \tag{12}$$

$$\text{and} \quad \mathbf{C} = 10^{(\mathbf{K} + \mathbf{A} \times \mathbf{X}')} \tag{13}$$

Minimization can be performed using all element of $\mathbf{R}$ using Newton-Raphson method for example or using other nonlinear optimization methods on some summation of $\mathbf{R}$ such as

sum of squares or sum of absolute values of residuals (Brassard and Bodurtha, 2000). In this case $\mathbf{R}$ would be a 2×1 vector of mass balance residuals.

The Fe system versus pH can be solved according to the method outlined above and the results are shown in the Figure below.

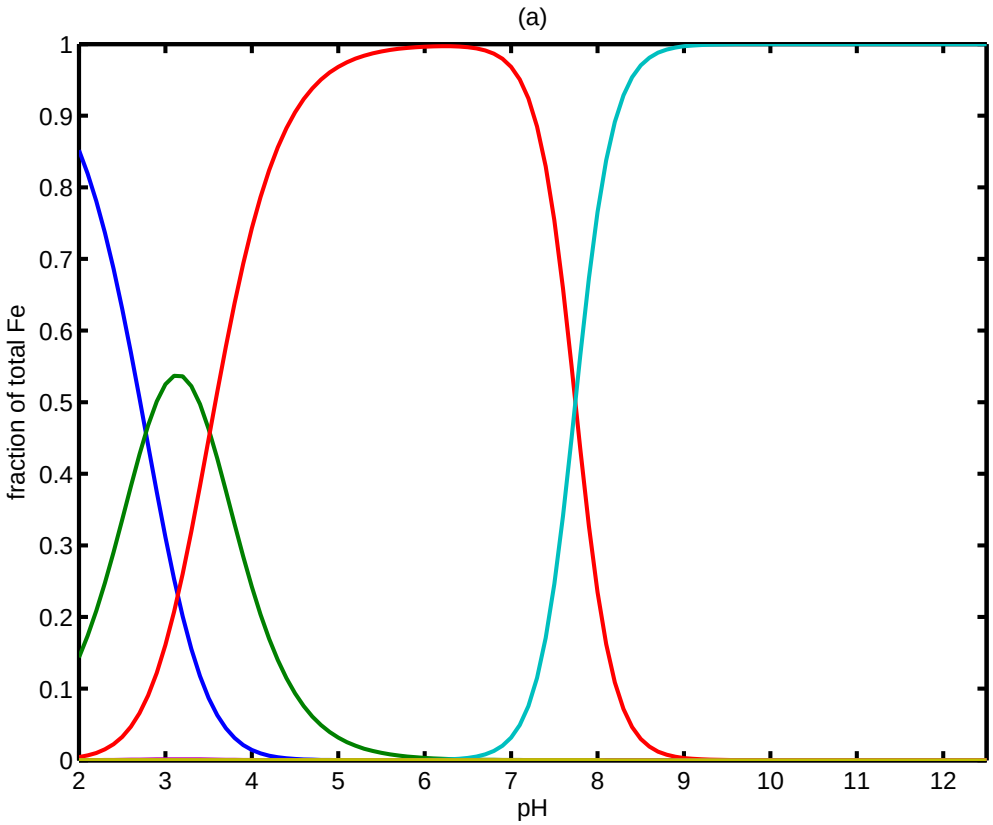

Fig. 1. Total iron of 10 μ M speciation versus pH not allowing precipitation

For supersaturated systems the criteria must be changed to allow for precipitation of mineral phases. The solid phases must be taken account in the mass balance expression and the new K_{sp} value must be satisfied. Usually speciation codes change the set of components by taking the precipitated species as a new component. Here I use the approach of Carrayrou et al. (2002) and simply add a new unknown and a new relationship for each solid phase that is precipitated.

For the Fe(III) system the solid phase I'll consider here is amorphous ferric hydroxide $Fe(OH)_{3(s)}$. We need to write the reaction as a precipitation reaction:

$$Fe^{3+} + 3H_2O \rightleftharpoons Fe(OH)_{3(s)} + 3H^+ \tag{14}$$

This relationship can be added to the existing tableau

Table 2 defines the chemical equilibrium problem. The speciation versus pH was calculated

H	Fe	logK	species
1	0	0	H^+
0	1	0	Fe^{3+}
-1	0	-14	OH^-
-1	1	-2.19	$FeOH^{2+}$
-2	1	-5.67	$Fe(OH)_2{}^+$
-4	1	-21.6	$Fe(OH)_4{}^-$
-3	1	-6	$Fe(OH)_{3(s)}$
TOTH	Fe_T		

Table 2
Tableau for Fe–H system including one possible solid phase.

for this system using the Matlab program developed here. For $Fe_T = 10\mu M$ and pH in the range 2 to 12.5 the results are shown in Figure 2. Note, by specifying the pH we do not need to input TOTH. At fixed pH this system only has one variable to solve for, $[Fe^{3+}]$

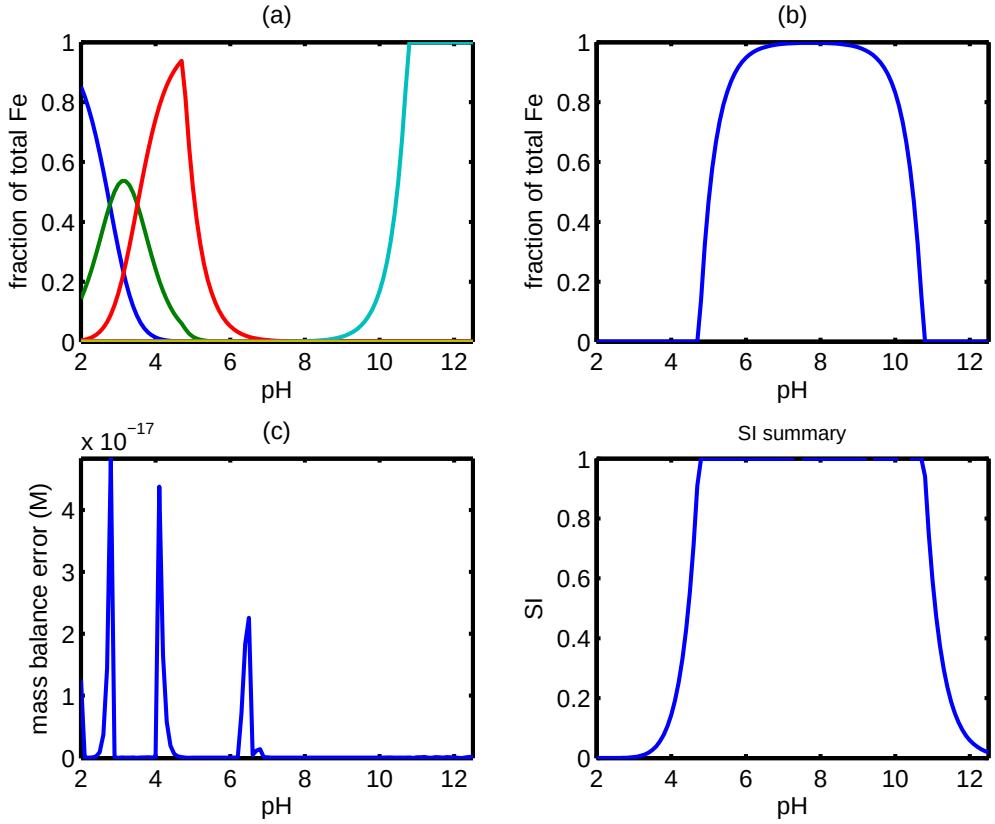

Fig. 2. Total iron of 10 μ M speciation versus pH. (a) soluble species versus pH (b) solid species (amorphous $Fe(OH)_{3(s)}$) (c) mass balance error as the absolute value of the sum of all values in **R** (d) plot of the saturation index versus pH.

The corresponding matlab code used to generate Figure 2 is given as Appendix 1. The

function call to this code is given as follows:

```
chem_equilib_Fe_fixedpH(2:0.1:12.5,[1e-5])
```

where the script is saved in a file name

```
chem_equilb_Fe_fixedpH.m
```

and takes a pH vector and total Fe as input parameters.

Anyone interested in utilizing this approach to solving simultaneous (and heterogeneous) equilibria can modify the program as necessary. All that is required is to change the matrix definitions to reflect the appropriate tableau and to change the initial guesses.

3 Numerical Method

In describing the numerical method used to solve simultaneous equilibria it is helpful to break up the Tableau into smaller matrices. For N_{cp} solid phases, N_C solution species and N_X components we can define the following matrices:

name	**dimension**	**description**
Asolution	$N_C \times N_X$	stoichiometry matrix for solution species
Asolid	$N_{cp} \times N_X$	stoichiometry matrix for solid species
Ksolution	$N_X \times 1$	$\log K$ values for solution species
Ksolid	$N_{cp} \times 1$	$\log K$ values for solid species
T	$N_X \times 1$	total of each component
Xsolution	$N_X \times 1$	component concentrations
Xcp	$N_{cp} \times 1$	solid phase concentrations
C	$N_C \times 1$	species concentrations
Rmass	$N_X \times 1$	mass balance residuals
RSI	$N_{cp} \times 1$	saturation index criteria

The Newton-Raphson method is an iterative gradient method. Starting with an initial guess the residual vector and the gradient matrix, the Jacobian, is calculated. The Jacobian is used to refine the guess and the process is repeated until a convergence criteria is satisfied.

The mass balance residual vector (**Rmass**) is calculated as follows:

$$\mathbf{logC} = \mathbf{Ksolution} + \mathbf{Asolution} \times \log_{10}(\mathbf{Xsolution}) \tag{15}$$
$$\mathbf{C} = 10.^{\mathbf{logC}} \tag{16}$$
$$\mathbf{Rmass} = \mathbf{Asolution}^T \times \mathbf{C} + \mathbf{Asolid} \times \mathbf{Xsolid} - \mathbf{T} \tag{17}$$
$$\tag{18}$$

The residual for the saturation index criteria is calculated as follows:

$$\mathbf{Q} = \mathbf{Asolid}^T \times \log_{10}(\mathbf{Xsolution}) \tag{19}$$
$$\mathbf{SI} = 10.^{(\mathbf{Q}+\mathbf{Ksolid})} \tag{20}$$
$$\mathbf{RSI} = 1 - \mathbf{SI} \tag{21}$$
$$\tag{22}$$

Notice the residual function for the saturation index is expressed as one minus the saturation index. This criteria forces the solution towards a saturation index (SI) of 1, which is the criteria for equilibria with a solid phase.

The overall objective function, $\mathbf{R}$ is obtained by "stacking" **Rmass** and **RSI**

$$\mathbf{R} = \begin{pmatrix} \mathbf{Rmass} \\ \mathbf{RSI} \end{pmatrix} \tag{23}$$

In the same way the overall optimization parameter vector $\mathbf{X}$ is defined as"stacking" **Xsolution** and **Xcp**

$$\mathbf{X} = \begin{pmatrix} \mathbf{Xsolution} \\ \mathbf{Xcp} \end{pmatrix} \tag{24}$$

Consider the elements of **Asolution** being $a_{i,j}$ and for **Asolid** the corresponding elements are $ap_{i,j}$. Similarly the elements of the saturation index vector are si_i and for the solution component vector xs_i. The gradient matrix ($\mathbf{Z}$) for each iteration can be calculated as follows:

$$Z_{j,k}\,|_{k=1,Nx}^{j=1,Nx} = \sum_{i=1}^{Nc} a_{i,j} \cdot a_{i,k} \left(\frac{[C_i]}{xs_k} \right) \tag{25}$$

$$Z_{j,k}\,|_{k=Nx+1,Nx+Ncp}^{j=1,Nx} = ap_{j,k-Nx} \tag{26}$$

$$Z_{j,k}\,|_{k=1,Nx}^{j=Nx+1,Nx+Ncp} = -ap_{k,j-Nx} \times \left(\frac{si_{j-Nx}}{xs_k} \right) \tag{27}$$

$$Z_{j,k}\,|_{k=Nx+1,Nx+Ncp}^{j=Nx+1,Nx+Ncp} = 0 \tag{28}$$

$$\tag{29}$$

Note that in Equation 28 there was a typo on the original Carrayrou et al. (2002) paper in that the negative sign was missing.

At each iteration the jacobian ($\mathbf{Z}$) and the residual vector $\mathbf{R}$ are calculated. The next guess for $\mathbf{X}$ is determined as follows:

$$\mathbf{deltaX} = -\mathbf{Z}^{-1} \cdot \mathbf{R} \tag{30}$$

$$one\_over\_del = \max[1, -1 \times \mathbf{deltaX}^T ./ (0.5 \times X^T)] \tag{31}$$

$$del = \frac{1}{one\_over\_del} \tag{32}$$

$$\mathbf{X} = \mathbf{X} + del \times \mathbf{deltaX} \tag{33}$$

Note, the *del* term is introduced so that the iterations will not go into negative concentration values. This procedure is presented by (Bethke, 1996).

References

Allison, J., Brown, D. S., Novo-Gradac, K., 1991. MINTEQA2/PROEFA2. A geochemical assessment model for environmental systems. U.S. Environmental Protection Agency, Athens, Georgia: Environmental Research Laboratory.

Bethke, C. M., 1996. Geochemical reacton modeling. Oxford University Press.

Brassard, P., Bodurtha, P., 2000. A feasible set for chemical speciation problems. Computers & Geosci. 26, 277–291.

Carrayrou, J., Mosé, R., Behra, P., 2002. New efficient algorithm for solving thermodynamic chemistry. AIChE Journal 48, 894–904.

Morel, F. M. M., Hering, J. G., 1993. Principles and applications of aquatic chemistry. Wiley-Interscience.

Saini-Eidukat, B., Yahin, A., 1999. Web-phreeq: a WWW instructional tool for modeling the distribution of chemical species in water. Computers and Geosciences 25, 347–353.

Smith, D. S., Ferris, F. G., 2001. Computational and experimental methods to determine metal binding in biofilms. In: Doyle, R. (Ed.), Methods in enzymology volume 337: microbial growth in biofilms part B special environments and physiochemical aspects. Academic Press, San Diego, pp. 225–242.

Appendix 1: fixed pH Fe-H speciation matlab script with embedded Newton-Raphson functions

```
% Carrayrou  et al AIChE journal 2002, 48, 894-904.
% implement their method using thier notation
% try HFO ppte as an example calc and at fixed pH

function II=chem_equilib_Fe_fixedpH(pH,T);

warning('off'); figure(1); clf

[KSOLUTION,KSOLID,ASOLUTION,ASOLID,SOLUTIONNAMES,SOLIDNAMES]=get_equilib_defn;

% initial guess
Fe_guess=[-5.5];
guess=[10.^Fe_guess];

numpts=size(pH,2);
Ncp=size(ASOLID,1);

iterations=1000; criteria=1e-16;

for i=1:size(SOLIDNAMES,1)
    txt=[SOLIDNAMES(i,:),'=zeros(numpts,1);']; eval(txt)
end

for i=1:size(pH,2)

    % adjust for fixed pH
    [Ksolution,Ksolid,Asolution,Asolid]=get_equilib_fixed_pH(KSOLUTION,KSOLID,ASOLUTION,ASOLID,pH(i));

    Asolid_SI_check=Asolid; Ksolid_SI_check=Ksolid;

    % number of different species
    Nx=size(Asolution,2); Ncp=size(Asolid,1); Nc=size(Asolution,1);

    % calculate species using NR

    solids=zeros(1,Ncp);

    if i==1; [species,err,SI]=NR_method_solution(Asolution,Asolid,Ksolid,Ksolution,T',[guess(1:Nx)]',iterations,criteria); end
    if i>1;
        [species,err,SI]=NR_method_solution(Asolution,Asolid,Ksolid,Ksolution,T',[species(2:Nx+1)],iterations,criteria);
    end

    for qq=1:Ncp

        [Y,I]=max(SI);

        if Y>1.000000001
            Iindex(qq)=I;
            Asolidtemp(qq,:)=Asolid_SI_check(I,:);
            Ksolidtemp(qq,:)=Ksolid_SI_check(I,:);
            solidguess(qq)=T(I)*0.5;
            if i>1; txt=['solidguess(qq)=',SOLIDNAMES(I,:),'(i-1);']; eval(txt); end
            guess=[species(2:Nx+1)' solidguess];
            [species,err,SItst,solids]=NR_method(Asolution,Asolidtemp',Ksolidtemp,Ksolution,T',guess',iterations,criteria);
             for q=1:size(solids,1);
                txt=[SOLIDNAMES(Iindex(q),:),'(i)=solids(q);']; eval(txt)
              end
        end

        Q=Asolid*log10(species(2:Nx+1)); SI=10.^(Q+Ksolid); Ifirst=I;

    end

    Q=Asolid*log10(species(2:Nx+1)); SI=10.^(Q+Ksolid);
    SI_summary(i,:)=SI;

    species_summary(i,:)=species;
    mass_err_summary(i,:)=(err(1));

    Asolidtemp=[]; Ksolidtemp=[];

end

for i=1:size(species_summary,2)
    txt=[SOLUTIONNAMES(i,:),'=species_summary(:,i);']; eval(txt)
end

figure(1)
subplot(221);
h=plot(pH,Fe/T,pH,FeOH/T,pH,FeOH2/T,pH,FeOH4/T,pH,Fe2OH2/T,pH,Fe3OH4/T);
set(gca,'fontsize',12); set(h,'linewidth',2); set(gca,'linewidth',2)
h=xlabel('pH'); set(h,'fontsize',12); h=ylabel('fraction of total Fe'); set(h,'fontsize',12)
title('(a)'); %legend('Fe','FeOH','FeOH2','FeOH4','Fe2OH2','Fe3OH4','Location','Best','Orientation','vertical')
axis([min(pH) max(pH) 0 1])

subplot(222); h=plot(pH,HFO/T);
set(gca,'fontsize',12); set(h,'linewidth',2); set(gca,'linewidth',2)
```

```
h=xlabel('pH'); set(h,'fontsize',12); h=ylabel('fraction of total Fe'); set(h,'fontsize',12)
title('(b)')
axis([min(pH) max(pH) 0 1])

subplot(223); h=plot(pH,(mass_err_summary(:,1)));
set(gca,'fontsize',12); set(h,'linewidth',2); set(gca,'linewidth',2)
h=xlabel('pH'); set(h,'fontsize',12); h=ylabel('mass balance error (M)'); set(h,'fontsize',12)
title('(c)')
axis([min(pH) max(pH) min(mass_err_summary(:,1)) max(mass_err_summary(:,1))])

subplot(224); h=plot(pH,SI_summary); title('SI summary')
set(gca,'fontsize',12); set(h,'linewidth',2); set(gca,'linewidth',2)
h=xlabel('pH'); set(h,'fontsize',12); h=ylabel('SI'); set(h,'fontsize',12)
axis([min(pH) max(pH) 0 1])

figure(1)

II=species_summary;

end

% -------------------- NR method solids present

function [species,err,SI,solids]=NR_method(Asolution,Asolid,Ksolid,Ksolution,T,guess,iterations,criteria)

Nx=size(Asolution,2); Ncp=size(Asolid,2); Nc=size(Asolution,1); X=guess;

for II=1:iterations

    Xsolution=X(1:Nx); Xsolid=X(Nx+1:Nx+Ncp);

    logC=(Ksolution)+Asolution*log10(Xsolution); C=10.^(logC); % calc species

    Rmass=Asolution'*C+Asolid*Xsolid-T;

    Q=Asolid'*log10(Xsolution); SI=10.^(Q+Ksolid);
    RSI=ones(size(SI))-SI;

    % calc the jacobian

    z=zeros(Nx+Ncp,Nx+Ncp);

    for j=1:Nx;
        for k=1:Nx;
                for i=1:Nc; z(j,k)=z(j,k)+Asolution(i,j)*Asolution(i,k)*C(i)/Xsolution(k); end
           end
    end

    for j=1:Nx;
        for k=Nx+1:Nx+Ncp;
                %t=Asolid';
        %        z(j,k)=t(k-Nx,j);
        z(j,k)=Asolid(j,k-Nx);
           end
    end


    for j=Nx+1:Nx+Ncp;
        for k=1:Nx
                z(j,k)=-1*Asolid(k,j-Nx)*(SI(j-Nx)/Xsolution(k));
          end
    end


    for j=Nx+1:Nx+Ncp
        for k=Nx+1:Nx+Ncp
            z(j,k)=0;
        end
    end


    R=[Rmass; RSI]; X=[Xsolution; Xsolid];

    deltaX=z\(-1*R);
    one_over_del=max([1, -1*deltaX'./(0.5*X')]);
    del=1/one_over_del;
    X=X+del*deltaX;

    tst=sum(abs(R));
    if tst<=criteria; break; end

end

logC=(Ksolution)+Asolution*log10(Xsolution); C=10.^(logC); % calc species
RSI=ones(size(SI))-SI;

Rmass=Asolution'*C+Asolid*Xsolid-T;

err=[Rmass];
species=[C];
solids=Xsolid;
```

```
end

% ---------- NR method just solution species

function [species,err,SI]=NR_method_solution(Asolution,Asolid,Ksolid,Ksolution,T,guess,iterations,criteria)

Nx=size(Asolution,2); Ncp=size(Asolid,1); Nc=size(Asolution,1); X=guess;

for II=1:iterations

    Xsolution=X(1:Nx);

    logC=(Ksolution)+Asolution*log10(Xsolution); C=10.^(logC); % calc species

    Rmass=Asolution'*C-T;

    Q=Asolid*log10(Xsolution); SI=10.^(Q+Ksolid);
    RSI=ones(size(SI))-SI;

    % calc the jacobian

    z=zeros(Nx,Nx);

    for j=1:Nx;
        for k=1:Nx;
                for i=1:Nc; z(j,k)=z(j,k)+Asolution(i,j)*Asolution(i,k)*C(i)/Xsolution(k); end
           end
    end

    R=[Rmass]; X=[Xsolution];

    deltaX=z\(-1*R);
    one_over_del=max([1, -1*deltaX'./(0.5*X')]);
    del=1/one_over_del;
    X=X+del*deltaX;

    tst=sum(abs(R));
    if tst<=criteria; break; end

end

logC=(Ksolution)+Asolution*log10(Xsolution); C=10.^(logC); % calc species
RSI=ones(size(SI))-SI;

Rmass=Asolution'*C-T;

err=[Rmass];

species=[C];

end

% ----- equilib definition ----------------

function [KSOLUTION,KSOLID,ASOLUTION,ASOLID,SOLUTIONNAMES,SOLIDNAMES]=get_equilib_defn;

    KSOLUTION=[...
                        0
                        0
     -1.404333360950650e+01
    -2.773333609506498e+00
    -6.286667219012998e+00
    -2.177333443802598e+01
     25.14+2*-14
     49.7+4*-14];

    ASOLUTION=[...
    %H           Fe
     1          0
     0          1
    -1         0
    -1          1
    -2          1
    -4          1
    -2         2
    -4         3      ];

  SOLUTIONNAMES=strvcat('H','Fe','OH','FeOH','FeOH2','FeOH4','Fe2OH2','Fe3OH4');


    % -------------- solid values

    KSOLID=[...
    -6.000000000000000e+00];

    ASOLID=[...
    -3     1 ];

 SOLIDNAMES=strvcat('HFO');


end
```

```
% ---------- for fixed pH ----------------

function [Ksolution,Ksolid,Asolution,Asolid]=get_equilib_fixed_pH(KSOLUTION,KSOLID,ASOLUTION,ASOLID,pH)

    [N,M]=size(ASOLUTION);
    Ksolution=KSOLUTION-ASOLUTION(:,1)*pH;
    Asolution=[ASOLUTION(:,2:M)];
    [N,M]=size(ASOLID);
    Ksolid=KSOLID-ASOLID(:,1)*pH;
    Asolid=[ASOLID(:,2:M)];

end
```

4 LICENSE