SUPPLEMENTARY MATERIAL

Theoretical understanding of morphological distribution of hydrogen in amorphous calcium silicate hydrate

Chongsong Zhou*a,*, and Lilin Luba

*aDepartment of Chemistry Biology & Environmental Engineering, Hunan Provincial Key Laboratory of Xiangnan Rare-Precious Metals Compounds and Applications, Xiangnan University, Chenzhou City, Hunan Province, People's Republic of China

bState Key Laboratory of Refractories and Metallurgy, Wuhan University of Science and Technology, Wuhan City, Hubei Province, People's Republic of China

*e-mail: youandi2008@163.com
The parameters in the ClayFF force field

According to Table 1S, the atomic types of ClayFF force field in this work are assigned as follows. The cah of 1.0500 charge is used for the Ca atoms, and the st of 2.1000 charge is used for Si atoms. The oh of -0.9500 charge, the o* of -0.8200 charge and the ob of -1.0500 charge is used for O atoms in OH group, H2O and the other part, respectively. The ho of 0.4250 charge and the h* of 0.4100 charge is used for H atoms in OH and H2O, respectively. Thereby the species for CaO, SiO2, Ca(OH)2 and Si(OH)4 can maintain electrical neutrality during ion exchange.

Table 1S. The parameters in the ClayFF force field

<table>
<thead>
<tr>
<th>Type</th>
<th>R0 (Å)</th>
<th>D0 (kcal·mol⁻¹)</th>
<th>element</th>
<th>charge</th>
<th>Atom type specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>h*</td>
<td>4.5775</td>
<td>0.0000</td>
<td>H</td>
<td>0.4100</td>
<td>water hydrogen</td>
</tr>
<tr>
<td>ho</td>
<td>4.5775</td>
<td>0.0000</td>
<td>H</td>
<td>0.4250</td>
<td>hydroxyl hydrogen</td>
</tr>
<tr>
<td>o*</td>
<td>3.5532</td>
<td>0.1554</td>
<td>O</td>
<td>-0.8200</td>
<td>water oxygen</td>
</tr>
<tr>
<td>oh</td>
<td>3.5532</td>
<td>0.1554</td>
<td>O</td>
<td>-0.9500</td>
<td>hydroxyl oxygen</td>
</tr>
<tr>
<td>ob</td>
<td>3.5532</td>
<td>0.1554</td>
<td>O</td>
<td>-1.0500</td>
<td>bridging oxygen</td>
</tr>
<tr>
<td>st</td>
<td>3.7064</td>
<td>1.8405E-06</td>
<td>Si</td>
<td>2.1000</td>
<td>tetrahedral silicon</td>
</tr>
<tr>
<td>cah</td>
<td>6.2428</td>
<td>5.0298E-06</td>
<td>Ca</td>
<td>1.0500</td>
<td>hydroxide calcium</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bond stretch</th>
<th>species</th>
<th>species j</th>
<th>k1 (kcal·mol⁻¹·Å⁻2)</th>
<th>R0 (Å)</th>
</tr>
</thead>
<tbody>
<tr>
<td>o*</td>
<td>h*</td>
<td></td>
<td>554.1349</td>
<td>1.000</td>
</tr>
<tr>
<td>oh</td>
<td>ho</td>
<td></td>
<td>554.1349</td>
<td>1.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>angle bend</th>
<th>species i</th>
<th>species j</th>
<th>species k</th>
<th>K2 (kcal·mol⁻¹·rad⁻²)</th>
<th>θ (degree)</th>
</tr>
</thead>
<tbody>
<tr>
<td>h*</td>
<td>o*</td>
<td>h*</td>
<td></td>
<td>45.7696</td>
<td>109.47</td>
</tr>
<tr>
<td>metal</td>
<td>oh</td>
<td>ho</td>
<td></td>
<td>30.0000</td>
<td>109.47</td>
</tr>
</tbody>
</table>

Ensuring the adequacy of simulation time

In order to ensure the adequacy of simulation time, the Qn of two C-S-H models are analyzed as the Table 2S. The mean absolute deviation of Qn between after 1000 ps and 100 ps simulation is only
1.42% and 1.32% for \( x = 0.4 \) and \( x = 1.5 \) C-S-H model, respectively, indicating that a simulation time of 100 ps is sufficient. (The \( x \) represents the molar number of H\(_2\)O molecules dissociated into OH)

**Table 2S.** The \( Q^n \) difference between after 1000 ps and 100 ps simulation

<table>
<thead>
<tr>
<th>( x )</th>
<th>Simulation time (ps)</th>
<th>( Q^0 ) (%)</th>
<th>( Q^1 ) (%)</th>
<th>( Q^2 ) (%)</th>
<th>( Q^3 ) (%)</th>
<th>( Q^4 ) (%)</th>
<th>Mean absolute deviation (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4</td>
<td>1000</td>
<td>26.24</td>
<td>38.72</td>
<td>24.45</td>
<td>9.41</td>
<td>1.18</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>25.47</td>
<td>42.22</td>
<td>22.86</td>
<td>8.23</td>
<td>1.22</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Difference</td>
<td>0.77</td>
<td>-3.50</td>
<td>1.59</td>
<td>1.18</td>
<td>-0.04</td>
<td>1.42</td>
</tr>
<tr>
<td>1.5</td>
<td>1000</td>
<td>38.33</td>
<td>38.18</td>
<td>18.00</td>
<td>4.62</td>
<td>0.88</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>39.53</td>
<td>40.28</td>
<td>15.32</td>
<td>4.26</td>
<td>0.61</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Difference</td>
<td>-1.20</td>
<td>-2.10</td>
<td>2.68</td>
<td>0.36</td>
<td>0.27</td>
<td>1.32</td>
</tr>
</tbody>
</table>

Examining the reproducibility of modelling

Ten original C-S-H models of \( x = 1.0 \) are randomly constructed using the same modeling method, and then are relaxed under the same simulation conditions. Their \( Q^n \) are listed in Table 3S. All standard deviations of \( Q^n \) being within 2.69% indicates the reproducibility of the modeling method.

**Table 3S.** The \( Q^n \) results of ten C-S-H models of \( x = 1.0 \) for examining the reproducibility of modelling

<table>
<thead>
<tr>
<th>Serial number</th>
<th>( Q^0 ) (%)</th>
<th>( Q^1 ) (%)</th>
<th>( Q^2 ) (%)</th>
<th>( Q^3 ) (%+( Q^4 ) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>31.13</td>
<td>44.40</td>
<td>23.98</td>
<td>0.50</td>
</tr>
<tr>
<td>2</td>
<td>31.38</td>
<td>45.17</td>
<td>23.46</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>30.46</td>
<td>48.50</td>
<td>21.04</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>25.50</td>
<td>44.70</td>
<td>29.80</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>31.40</td>
<td>42.00</td>
<td>26.10</td>
<td>0.50</td>
</tr>
<tr>
<td>6</td>
<td>27.40</td>
<td>46.30</td>
<td>26.30</td>
<td>0.00</td>
</tr>
<tr>
<td>7</td>
<td>25.20</td>
<td>47.00</td>
<td>27.80</td>
<td>0.00</td>
</tr>
<tr>
<td>8</td>
<td>28.60</td>
<td>41.80</td>
<td>29.10</td>
<td>0.50</td>
</tr>
<tr>
<td>9</td>
<td>27.30</td>
<td>48.20</td>
<td>24.50</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>27.30</td>
<td>47.20</td>
<td>25.00</td>
<td>0.50</td>
</tr>
</tbody>
</table>

The standard deviation 2.39 2.36 2.69 0.26
The layered C-S-H structure

**Figure 1S.** The top view snapshot of C-S-H structure at x = 1.0 after 100 ps simulation. The yellow tetrahedron is the SiO$_4$ tetrahedron; and the blue ball is for O while the white ball is for H in H$_2$O; and the red stick is for O while the white stick is for H in OH; and the green cross is for Ca.

As can be seen from Figure 1S, the simulation results show that all Si atoms can form a silicone tetrahedron and generate a longer SiO$_4$ chain further. However, all Ca atoms are dispersed freely around SiO$_4$ chains with the OH and H$_2$O to form a layered C-S-H. $^2$

The mean square displacement (MSD) of water molecules in different C-S-H models

**Figure 2S.** MSD of H$_2$O in pure water and C-S-H models.
The mean square displacement (MSD) of water molecules is reduced from $1.50 \times 10^{-9} \text{ m}^2\cdot\text{s}^{-1}$ in pure water phase to about $1.50 \times 10^{-10} \text{ m}^2\cdot\text{s}^{-1}$ in C-S-H models (Figure 2S). The decreasing by an order of magnitude is attributed to the decrease in the fluidity of water molecules in C-S-H. It can be deduced that the water molecules in C-S-H form stable structural water, which promotes the formation of a hydrogen bond network, and it may play an important role in linking the layered silicon chains and in superior mechanical properties.

The proportional change of three types of H atoms in C-S-H

The percentage of hydrogen atoms in H$_2$O, Si-OH, and Ca-OH under different $x$ can be obtained based on the distribution ratio of OH in Ca-OH and Si-OH, and the results are shown in figure 3S. The ratio of H atoms in Ca-OH to total H atoms increases from 26% to 45%, and the ratio of H atoms in Si-OH to total H atoms increases from 13% to 22%. Obviously, the proportion of H atoms in Ca-OH is always higher than that in Si-OH.

![Figure 3S](image)

*Figure 3S. The proportional variation of three hydrogen forms in C-S-H with different $x$*
THE awk CODE FOR CALCULATING QN

BEGIN {vartime=strftime("%Y/%m/%d %H:%M:%S",sys/time());print "start at: ",vartime;
  
  print("FILENAME");
  split(FILENAME,Frame, ");
  fileNamePrefix=Frame[1];
  structureFile=fileNamePrefix".xtd"
  asciiTrjFile=fileNamePrefix"_trj.txt"
  QnOutPutFile=fileNamePrefix"_Qn.xls"

  if (print("the Trj2Ascii.exe program must be specified the right road");
    system("F:\trj2acsii\Trj2Ascii.exe "FILENAME" >>asciiTrjFile);
  )
  
  threshold=3.20;   # define the standard value of whether is connected with each other or not.
  
  ### reading the order of atom and the forcefield type of atoms from .xtd file
  while(getline < structureFile)==0 {
    if($1="<Atom3d")&($6=?=10)) {
      split($i,select,"=");

      if (select[1]=="UserID=")         { n++; numberOrder[n]=select[2];                                              }
      else if (select[1]=="ForcefieldType=") { forcefieldType[n]=select[2];                                              }
      else if (select[1]=="Components=")     { split(select[2],elementType,","); elementSymbol[n]=elementType[1]; break; }
    }

  }

close(structureFile);

  ### classify the atoms by their forcefield and counting their number
  for(e=1;e<=n;e++) {
    if     ( forcefieldType[e]=="cah" )   {  ncah++; CahID[ncah]=e;}
    else if( forcefieldType[e]=="cao" )   {  ncao++; CaoID[ncao]=e;}
    else if( forcefieldType[e]=="st"  )   {  nst++;  SiID[nst]=e;  }
    else if( forcefieldType[e]=="ob"  )   {  nob++;  ObID[nob]=e;  }
    else if( forcefieldType[e]=="ho"  )   {  nho++;  HoID[nho]=e;  }
    else if( forcefieldType[e]=="o*"  )   {  now++;  OwID[now]=e;  }
    else if( forcefieldType[e]=="h*"  )   {  nhw++;  HwID[nhw]=e;  }

    }
}

### analysis the Qn from the trajectory file

  printf("time	a	b	c	alpha	beta
gamma
density	Q0	Q1	Q2	Q3	Q4	Q0%	Q1%	Q2%	Q3%	Q4%	MSL	Qfactor	Q3+Q4%
ps
") > QnOutPutFile;

  printf("time	a	b	c	alpha	beta
gamma
density	Q0	Q1	Q2	Q3	Q4	Q0%	Q1%	Q2%	Q3%	Q4%	MSL	Qfactor	Q3+Q4%
ps
") >> QnOutPutFile;

  while((getline < asciiTrjFile)==0) {
    if($1=="Frame")&($2=="Number:")) {
      printf("time	a	b	c	alpha	beta
gamma
density	Q0	Q1	Q2	Q3	Q4	Q0%	Q1%	Q2%	Q3%	Q4%	MSL	Qfactor	Q3+Q4%
ps
") >> QnOutPutFile;

      frameNumber++;
      
      getline < asciiTrjFile;  # unscramble the trajectory file and analyzing the Qn
      time[frameNumber]=$2; temp[frameNumber]=$4;
      getline < asciiTrjFile;  # unscramble the trajectory file and analyzing the Qn
      TotalPE[frameNumber]=$2; BondE[frameNumber]=$3;
      getline < asciiTrjFile;  # unscramble the trajectory file and analyzing the Qn
      AngleE[frameNumber]=$1; TorsionE[frameNumber]=$2; TotalE[frameNumber]=$3;
      getline < asciiTrjFile;  # unscramble the trajectory file and analyzing the Qn
      ElectrostaticE[frameNumber]=$2;
      getline < asciiTrjFile;  # unscramble the trajectory file and analyzing the Qn
      ThreeBodyE[frameNumber]=$3; 
      getline < asciiTrjFile;  # unscramble the trajectory file and analyzing the Qn
      TotalInternalE[frameNumber]=$3;
      getline < asciiTrjFile;  # unscramble the trajectory file and analyzing the Qn
      AvgTotalPE[frameNumber]=$2; TotalKE[frameNumber]=$3;
      getline < asciiTrjFile;  # unscramble the trajectory file and analyzing the Qn
      AvgTotalKE[frameNumber]=$2; TotalKE[frameNumber]=$3;
      getline < asciiTrjFile;  # unscramble the trajectory file and analyzing the Qn
      if($1=="DefCell:"  ) {

        VectorC11[frameNumber]=$1; VectorC22[frameNumber]=$2; VectorC33[frameNumber]=$3;
        VectorC12[frameNumber]=$4; VectorC13[frameNumber]=$5; VectorC23[frameNumber]=$6;
        for(j=1; frameNumber[j]=sqrt((VectorC11[frameNumber][j])^2+(VectorC12[frameNumber][j])^2+(VectorC13[frameNumber][j])^2);
        
        if($1=="Coordinates:"  ) {

          atomX[frameNumber,1]=S; atomY[frameNumber,1]=S; atomZ[frameNumber,1]=S;
          for(p=2;p<=n;p++) {
            getline < asciiTrjFile; atomX[frameNumber,p]=S; atomY[frameNumber,p]=S; atomZ[frameNumber,p]=S;
          }
        }
      }
      
    }
  }
else if (Si[frameNumber,m] == 3) Q3[frameNumber]++;  
else if (Si[frameNumber,m] == 2) Q2[frameNumber]++;  
else if (Si[frameNumber,m] == 1) Q1[frameNumber]++;  
if      (Si[frameNumber,m] == 0) Q0[frameNumber]++;  
for(m=1;m<=nst;m++) {  
   if($1=="Velocities:"){  
      ForceX[frameNumber,1] = $3; ForceY[frameNumber,1] = $4; ForceZ[frameNumber,1] = $5;  
   }  
   else if($1=="Forces:"){  
      velocityX[frameNumber,1] = $3; velocityY[frameNumber,1] = $4; velocityZ[frameNumber,1] = $5;  
   }  
   velocityX[frameNumber,p] = $3;  
   velocityY[frameNumber,p] = $4;  
   velocityZ[frameNumber,p] = $4;  
}  
else if($1=="Frame") break;  
for(i=1;i<=nst;i++) { Si[frameNumber,i] = 0;  
}  
for(j=1;j<=nst;j++) {  
   # calculation the distance of two Si atoms  
   dxSquare = (atomX[frameNumber,SiID[i]] - atomX[frameNumber,SiID[j]])^2; dySquare = (atomY[frameNumber,SiID[i]] - atomY[frameNumber,SiID[j]])^2; dzSquare = (atomZ[frameNumber,SiID[i]] - atomZ[frameNumber,SiID[j]])^2;  
   distance = sqrt(dxSquare + dySquare + dzSquare);  
   if(distance <= threshold) { Si[frameNumber,i]++; Si[frameNumber,j]++; }  
   newdxSquare = (tempXj - atomX[frameNumber,SiID[i]])^2; newdySquare = (tempYj - atomY[frameNumber,SiID[i]])^2; newdzSquare = (tempZj - atomZ[frameNumber,SiID[i]])^2;  
   if((distance <= threshold) && (distance > 0)) { Si[frameNumber,i]++; Si[frameNumber,j]++; }  
}  
flagA = a[frameNumber] * sqrt( (fractionalA[frameNumber,SiID[i]] - newfractionalAj)^2 );  
flagB = b[frameNumber] * sqrt( (fractionalB[frameNumber,SiID[i]] - newfractionalBj)^2 );  
flagC = c[frameNumber] * sqrt( (fractionalC[frameNumber,SiID[i]] - newfractionalCj)^2 );  
if(flagA > 0) { if(fractionalA[frameNumber,SiID[i]] < 0.5) newfractionalAj = fractionalA[frameNumber,SiID[i]] + 1; else newfractionalAj = fractionalA[frameNumber,SiID[i]] - 1; }  
if(flagB > 0) { if(fractionalB[frameNumber,SiID[i]] < 0.5) newfractionalBj = fractionalB[frameNumber,SiID[i]] + 1; else newfractionalBj = fractionalB[frameNumber,SiID[i]] - 1; }  
if(flagC > 0) { if(fractionalC[frameNumber,SiID[i]] < 0.5) newfractionalCj = fractionalC[frameNumber,SiID[i]] + 1; else newfractionalCj = fractionalC[frameNumber,SiID[i]] - 1; }  
newFractionalA[frameNumber,SiID[i]] = newfractionalAj * FractionalA[frameNumber,SiID[i]];  
newFractionalB[frameNumber,SiID[i]] = newfractionalBj * FractionalB[frameNumber,SiID[i]];  
newFractionalC[frameNumber,SiID[i]] = newfractionalCj * FractionalC[frameNumber,SiID[i]];  
}  
for(m=1;m<=nst;m++) {  
   if  (Si[frameNumber,m] == 0) Q0[frameNumber]++;  
   else if (Si[frameNumber,m] == 1) Q1[frameNumber]++;  
   else if (Si[frameNumber,m] == 2) Q2[frameNumber]++;  
   else if (Si[frameNumber,m] == 3) Q3[frameNumber]++;  
   else if (Si[frameNumber,m] == 4) Q4[frameNumber]++;  
}  
Mass = ncah + ncao * 40.078 + nst * 28.0855 + (nob + noh + now) * 15.9994 + (nho + nhw) * 1.008;  
density[frameNumber] = Mass / Volume[frameNumber];  
#MSL and MSLL is the mean silicon linking length  
MSL[frameNumber] = ((Q0[frameNumber] + Q1[frameNumber] + Q2[frameNumber] + Q3[frameNumber] + Q4[frameNumber]) / frameNumber);  
MSLL[frameNumber] = ((Q0[frameNumber] + 0.5 * Q2[frameNumber] + Q4[frameNumber]) / frameNumber);  
p0 = 100 * Q0[frameNumber] / frameNumber; p1 = 100 * Q1[frameNumber] / frameNumber; p2 = 100 * Q2[frameNumber] / frameNumber; p3 = 100 * Q3[frameNumber] / frameNumber; p4 = 100 * Q4[frameNumber] / frameNumber;  
if(p0 >= 100) Q0[frameNumber] = 0; else Q0 = 0;  
p3 = p3 + p4;  
print("%.3f%.3f%.3f%.3f%.3fACOS(%.6f)180/3.1415926ACOS(%.6f)180/3.1415926ACOS(%.6f)180/3.1415926ACOS(%.6f)180/3.1415926ACOS(%.6f)ACOS(%.6f)ACOS(%.6f)ACOS(%.6f)ACOS(%.6f)ACOS(%.6f)ACOS(%.6f)");  
frameNumber, Siname, density[frameNumber], Q0[frameNumber], Q1[frameNumber], Q2[frameNumber], Q3[frameNumber], Q4[frameNumber], p0, p1, p2, p3, p4, MSL[frameNumber], Qf, p34 >> QnOutputFile;
> close(QnOutputFile);
> }
>
> close(FILENAME);
> system("rm 'ascitTrjFile');
> exit;
>
> END{vartime=strftime("%Y/%m/%d %H:%M:%S",systime());print " end at: ", vartime; }

REFERENCES