Wetting at the Nanoscale: A Molecular Dynamics Study – Supplementary Materials

Mohammad Khalkhali, Nasser Kazemi, Hao Zhang, and Qingxia Liu

Codes corresponding to the contact angle calculation method introduced in this study are provided in this document. Reader who would like to have more detailed information such as codes of other contact angle calculation methods and sample of molecular dynamics simulation outputs are encouraged to see “Resources” section of Dr. Khalkhali’s website:

<http://sites.ualberta.ca/~khalkhal/>

All algorithms were coded in MATLAB Version 8.6.0.267264 and designed to read LAMMPS dump files as inputs.

Functions:

* **read\_LAMMPS\_traj**: This function reads a LAMMPS trajectory file and returns the coordinates of the centre of masses of water molecules.
* **hit\_and\_count**: This Function applies the hit-and-count algorithm to remove outliers from a data set.
* **fine\_percision**: This function applies the fine precision droplet identification process to remove gas atoms which are very close to the liquid droplet surface.
* **contact\_angle**: This function calculates the contact angle of a liquid droplet of a solid surface through convex hull triangulation.
* **weighted\_distribution:** This function constructs a histogram of weighted data.
* **run.m**: This is a sample run file to perform contact angle calculation.

read\_LAMMPS\_traj

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This function reads a LAMMPS trajectory file and returns the coordinates

% of the centre of masses of water molecules. The LAMMPS trajectory file

% should contain just one snapshot. Oxygen and Hydrogen atoms of water

% should be the last two atom types in the trajectory file (Hw is the last

% atom type). Box\_x, Box\_y and Box\_z are dimensions of the simulation box.

%

% Author: Mohammad Khalkhali, Sep 2016

% Reference: Khalkhali et al. J. Chem. Phys. (2017)

%

% This program is a free software: you can redistribute it and/or modify

% it under the terms of the GNU General Public License as published

% by the Free Software Foundation, either version 3 of the License, or

% any later version.

%

% This program is distributed in the hope that it will be useful,

% but WITHOUT ANY WARRANTY; without even the implied warranty of

% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

% GNU General Public License for more details: http://www.gnu.org/licenses/

%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [fileID,x\_org,y\_org,z\_org] = read\_LAMMPS\_traj(Filename,Box\_x,...

Box\_y,Box\_z,format\_spec)

fileID = fopen(Filename,'r'); %opening the trajectory file to read

if (fileID < 0)

return

end

% reading the trajectory file

datacell = textscan(fileID, format\_spec, 'HeaderLines', 9);

fclose(fileID);

Data = cell2mat(datacell);

[dummy, index] = sort(Data(:,1));

ID\_org = Data(index,1);

Type\_org = Data(index,2);

x\_org = Data(index,3);

y\_org = Data(index,4);

z\_org = Data(index,5);

% Keeping only the water molecules: Ow and Hw

max\_type=max(Type\_org);

i=find(Type\_org>=max\_type-1);% Ow and Hw are the last two atom types

ID\_org=ID\_org(i);

Type\_org=Type\_org(i);

x\_org=x\_org(i);

y\_org=y\_org(i);

z\_org=z\_org(i);

j = find(Type\_org == max\_type-1); %O atoms in water

% update positions of H atoms considering the periodic boundary

% condition

x\_org(j+1) = x\_org(j+1) - Box\_x\*round((x\_org(j+1)-x\_org(j))/Box\_x);

x\_org(j+2) = x\_org(j+2) - Box\_x\*round((x\_org(j+2)-x\_org(j))/Box\_x);

y\_org(j+1) = y\_org(j+1) - Box\_y\*round((y\_org(j+1)-y\_org(j))/Box\_y);

y\_org(j+2) = y\_org(j+2) - Box\_y\*round((y\_org(j+2)-y\_org(j))/Box\_y);

z\_org(j+1) = z\_org(j+1) - Box\_z\*round((z\_org(j+1)-z\_org(j))/Box\_z);

z\_org(j+2) = z\_org(j+2) - Box\_z\*round((z\_org(j+2)-z\_org(j))/Box\_z);

% calculating centre of mass of water molecules

x\_org = (16\*x\_org(j)+x\_org(j+1)+x\_org(j+2))/18;

y\_org = (16\*y\_org(j)+y\_org(j+1)+y\_org(j+2))/18;

z\_org = (16\*z\_org(j)+z\_org(j+1)+z\_org(j+2))/18;

end

hit\_and\_count:

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This Function applies the hit-and-count algorithm to remove outliers from

% a data set

%

% Authors:

% Mohammad Khalkhali

% Nasser Kazemi

% (Sep 2016)

%

% References:

% Khalkhali et al. J. Chem. Phys. (2017)

%

% This program is a free software: you can redistribute it and/or modify

% it under the terms of the GNU General Public License as published

% by the Free Software Foundation, either version 3 of the License, or

% any later version.

%

% This program is distributed in the hope that it will be useful,

% but WITHOUT ANY WARRANTY; without even the implied warranty of

% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

% GNU General Public License for more details: http://www.gnu.org/licenses/

%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x\_final,y\_final,z\_final] = hit\_and\_count(x\_org,y\_org,z\_org,...

binsize,numer\_atom\_in\_bin)

% binning of the data using binsize

x\_bin=floor(x\_org/binsize)+1;

y\_bin=floor(y\_org/binsize)+1;

z\_bin=floor(z\_org/binsize)+1;

% applying hit-and-run in X direction

B=unique(x\_bin);

C=histc(x\_bin(:),B);

e=find(C>=numer\_atom\_in\_bin);

KK=find(x\_bin>=B(e(1)) & x\_bin<=B(e(end)));

x=x\_bin(KK);

y=y\_bin(KK);

z=z\_bin(KK);

% ID\_drop=ID\_org(KK);

% Type\_drop=Type\_org(KK);

x\_drop=x\_org(KK);

y\_drop=y\_org(KK);

z\_drop=z\_org(KK);

% applying hit-and-run in Y direction

B=unique(y);

C=histc(y(:),B);

e=find(C>=numer\_atom\_in\_bin);

KK=find(y>=B(e(1)) & y<=B(e(end)));

x=x(KK);

y=y(KK);

z=z(KK);

% ID\_drop=ID\_drop(KK);

% Type\_drop=Type\_drop(KK);

x\_drop=x\_drop(KK);

y\_drop=y\_drop(KK);

z\_drop=z\_drop(KK);

% applying hit-and-run in Z direction

B=unique(z);

C=histc(z(:),B);

e=find(C>=numer\_atom\_in\_bin);

KK=find(z>=B(e(1)) & z<=B(e(end)));

x=x(KK);

y=y(KK);

z=z(KK);

% ID\_final=ID\_drop(KK);

% Type\_final=Type\_drop(KK);

x\_drop=x\_drop(KK);

y\_drop=y\_drop(KK);

z\_drop=z\_drop(KK);

x\_final=x\_drop;

y\_final=y\_drop;

z\_final=z\_drop;

end

fine\_percision

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This function applies the fine precision droplet identification process

% to remove gas atoms which are very close to the liquid droplet surface

% and was not removed through hit-and-count process.

%

% Parameters:

% R\_step: number of radial bins

% delta\_R: size of the radial bin

% (see the publication for details)

%

% Authors:

% Nasser Kazemi

% (Sep 2016)

%

% References:

% Khalkhali et al. J. Chem. Phys. (2017)

%

% This program is a free software: you can redistribute it and/or modify

% it under the terms of the GNU General Public License as published

% by the Free Software Foundation, either version 3 of the License, or

% any later version.

%

% This program is distributed in the hope that it will be useful,

% but WITHOUT ANY WARRANTY; without even the implied warranty of

% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

% GNU General Public License for more details: http://www.gnu.org/licenses/

%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x\_final,y\_final,z\_final] = fine\_percision(x\_final,y\_final,...

z\_final,R\_step,delta\_R)

dx\_ave=max(x\_final)-min(x\_final);

dy\_ave=max(y\_final)-min(y\_final);

D=max(dx\_ave,dy\_ave);

d=D/2;

Z\_max=max(z\_final);

%radious calculation

R=(d.^2+Z\_max.^2)./(2\*Z\_max);

x\_ave=min(x\_final)+dx\_ave/2;

y\_ave=min(y\_final)+dy\_ave/2;

z\_ave=Z\_max-R;

%Find the distance from origin of the remaining points

% after his-and-count

l2=zeros(size(x\_final));

for j=1:length(l2)

l2(j)= sqrt((x\_final(j)-x\_ave).^2+...

(y\_final(j)-y\_ave).^2+(z\_final(j)-z\_ave).^2);

end

%Define radious search region for evalutation of scattered points

R\_interval=R-R\_step\*delta\_R:delta\_R:R+R\_step\*delta\_R;

R\_interval(end)=R\_interval(end)+3;

% define three regions:smaller than search zone radious,

% search zone and bigger than search limit radious.

%First remove bigger than limit radious data

e=find(l2<R\_interval(end));

x\_final=x\_final(e);

y\_final=y\_final(e);

z\_final=z\_final(e);

l2=l2(e);

%Second keep data less than minimum radious evaluation

% (do not touch)

e=find(l2<R\_interval(1));

x\_pass=x\_final(e);

y\_pass=y\_final(e);

z\_pass=z\_final(e);

%Third evaluate data in the search zone for further action

e=find(l2>=R\_interval(1) & l2<=R\_interval(end));

x\_zone=x\_final(e);

y\_zone=y\_final(e);

z\_zone=z\_final(e);

l2=l2(e);

x\_temp=0;

y\_temp=0;

z\_temp=0;

for j=1:length(R\_interval)-1

e=find(l2 >= R\_interval(j) & l2 <= R\_interval(j+1));

x=x\_zone(e);

y=y\_zone(e);

z=z\_zone(e);

for m=1:length(x)

count=0;

for n=2:length(x\_zone)

distance=sqrt((x(m)-x\_zone(n)).^2+ ...

(y(m)-y\_zone(n)).^2 ...

+(z(m)-z\_zone(n)).^2);

if (distance<4.5 && distance>0)

count=count+1;

end

end

if count>=1

x\_temp=[x\_temp;x(m)];

y\_temp=[y\_temp;y(m)];

z\_temp=[z\_temp;z(m)];

end

end

end

x\_final=[x\_pass;x\_temp(2:end)];

y\_final=[y\_pass;y\_temp(2:end)];

z\_final=[z\_pass;z\_temp(2:end)];

end

contact\_angle

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This function calculates the contact angle of a liquid droplet of a solid

% surface through convex hull triangulation.

%

% Parameters:

% z\_min: Triangles with at least one vertex position lower that z\_min\*max(z)

% are contributing in the contact angle calculation.

% precision = This parameter is used to recognize triangles contributing

% to the area of the base of the droplet.

%

% Authors:

% Mohammad Khalkhali

% Nasser Kazemi

% (Sep 2016)

%

% References:

% Khalkhali et al. J. Chem. Phys. (2017)

%

% This program is a free software: you can redistribute it and/or modify

% it under the terms of the GNU General Public License as published

% by the Free Software Foundation, either version 3 of the License, or

% any later version.

%

% This program is distributed in the hope that it will be useful,

% but WITHOUT ANY WARRANTY; without even the implied warranty of

% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

% GNU General Public License for more details: http://www.gnu.org/licenses/

%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [K,V,angle,BaseA\_temp] = contact\_angle(x\_final,y\_final,z\_final,...

z\_min,precision)

%apply convex hull algorithm to define the boundary

[K,V] = convhull(x\_final,y\_final,z\_final,'simplify',true);

angle=zeros(2,length(K(:,1)));

BaseA\_temp = 0;

%calculating angles

for jj=1:length(K(:,1))

% applying z\_limit

if (any(z\_final(K(jj,:)) < min(z\_final) + z\_min\*max(z\_final)))

x=x\_final(K(jj,:));

y=y\_final(K(jj,:));

z=z\_final(K(jj,:));

p1=[x(1) y(1) z(1)];

p2=[x(2) y(2) z(2)];

p3=[x(3) y(3) z(3)];

n = cross(p1-p2, p1-p3);%normal to the triangular facet

angle(1,jj)=acos(n(3)./norm(n))\*180/pi;

angle(2,jj)=norm(n);

% applying precision

if ((abs(z(1)-z(2))<precision && ...

abs(z(1)-z(3))<precision && ...

abs(z(3)-z(2))<precision))

BaseA\_temp = BaseA\_temp + norm(n); % area of the base

angle(2,jj)=0;

end

if (all(z\_final(K(jj,:)) < min(z\_final) + z\_min\*max(z\_final)))

angle(2,jj)=0;

end

else

angle(1,jj)=0;

angle(2,jj)=0;

end

end

end

weighted\_distribution

function w = weighted\_distribution(Distribution,Weight)

D=find(Distribution>-1);

% calculating weighted histogram acording to the area of triangles

edges = [0:180];

vals = Distribution(D);

weights = Weight(D);

Nedge = length(edges);

w = zeros(size(edges));

for n = 1:Nedge-1

ind = find(vals >= edges(n) & vals < edges(n+1));

if ~isempty(ind)

w(n) = sum(weights(ind));

end

end

ind = find(vals == edges(end));

if ~isempty(ind)

w(Nedge) = sum(weights(ind));

end

end

run

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This file runs the calculation for contact angle of a liquid droplet

% on a solid surface using the method of Khalkhali et al. In this method

% contact angle is calculated through convex hull triangulation (see

% the referenced Publication for more details)

%

% Authors:

% Mohammad Khalkhali

% Nasser Kazemi

% (Sep 2016)

%

% References:

% Khalkhali et al. J. Chem. Phys. (2017)

%

% Functions:

% read\_LAMMPS\_traj: reads a LAMMPS trajectory file and returns the

% positions of centre of masses of water molecules.

% hit\_and\_count: runs hit and count algorithm to identify points in liquid

% droplet from those in the gas phase.

% fine\_percision: applied the fine precision droplet identification process

% to remove near-droplet gas molecules.

% contact\_angle: calculates the contact angles along the contact line using

% the convex hull triangulation. This function also returns the area of the

% base of the droplet and the area of corresponding triangles used to

% calculate each contact angle value.

% weighted\_distribution: calculates weighted histogram of contact angles

% according to the area of corresponding triangles

%

% This program is a free software: you can redistribute it and/or modify

% it under the terms of the GNU General Public License as published

% by the Free Software Foundation, either version 3 of the License, or

% any later version.

%

% This program is distributed in the hope that it will be useful,

% but WITHOUT ANY WARRANTY; without even the implied warranty of

% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

% GNU General Public License for more details: http://www.gnu.org/licenses/

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc;

clear all;

fclose ('all');

close('all');

trajName = '../Graphite\_Water(3nm)/Graphite\_Water.lammpstrj';

% This format\_spec is needed when reading lammps trajectories in

% read\_LAMMPS\_traj function. There are 5 entries at each line of our

% trajectories

format\_spec = '%f %f %f %f %f';

StartStep = 4990000;

EndStep = 5000000;

step = 1000;

% Parameters of hit-and-count function (refer to the publication for details)

binsize=2;

numer\_atom\_in\_bin=5;

% Parameters of fine\_percision function (refer to the publication for details)

% fine presicion step of identifying droplet limits is usually redundant

% (hit and count step is sufficient).

fine\_percision\_check = 0; % calls fine\_percision function if is 1

delta\_R = 5;

R\_step = 1;

% Draws graphs corresponding to each step for the first time step. It is

% recommended to check if parameters droplet identification process are

% ajusted properly.

graphcheck = 0;

% Triangles with atleast one vertex position lower that z\_min\*max(z) are

% contributing in the contact angle calculation.

z\_min = 0.08;

% This parameter is used to recognize triangles contributing to the area of

% the base of the droplet.

precision = 1.0;

Distribution=[];

Weight=[];

BaseA = 0;

for tt = StartStep:step:EndStep

fprintf('Current time step = %d\r', tt);

Filename = sprintf('%s.%d',trajName,tt); %trajectory file name

% Reading box dimentions to calculates centre of mass of water

% molecules correctly: some water molecules may pass the boundary.

% Since it is a NVT simulation we just need to read box dimentions

% once.

if (tt == StartStep)

fileID = fopen(Filename,'r');

if (fileID < 0)

fprintf('can not open %s file',Filename);

break;

end

%reading box sizes

header = textscan(fileID, '%s',9,'delimiter', '\n');

data = textscan(header{1}{6},'%f %f');

Box\_x = data{2}-data{1};

data = textscan(header{1}{7},'%f %f');

Box\_y = data{2}-data{1};

data = textscan(header{1}{8},'%f %f');

Box\_z = data{2}-data{1};

fclose(fileID);

end

[fileID,x\_org,y\_org,z\_org] = read\_LAMMPS\_traj(Filename,Box\_x,...

Box\_y,Box\_z,format\_spec);

if (fileID < 0)

fprintf('can not open %s file',Filename);

break;

end

% Drawing the droplet before applying hit-and-run

if (tt == StartStep && graphcheck == 1)

figure;

scatter3(x\_org,y\_org,z\_org,...

'MarkerEdgeColor','k',...

'MarkerFaceColor',[0 .25 .25]);

view(45,45)

title('original');

size1 = size(x\_org,1);

axis tight

end

% Applying hit and count method to remove outliers

[x\_final,y\_final,z\_final] = hit\_and\_count(x\_org,...

y\_org,z\_org,binsize,numer\_atom\_in\_bin);

if (tt == StartStep && graphcheck == 1)

figure;

scatter3(x\_final,y\_final,z\_final,...

'MarkerEdgeColor','k',...

'MarkerFaceColor',[0 .25 .25]);

view(45,45)

size2 = size(x\_final,1);

str = sprintf('After hit-and-count (%d points were removed)', ...

size1-size2);

title(str);

axis tight

end

% Applying fine precission method to remove near droplet outliers

if (fine\_percision\_check == 1)

[x\_final,y\_final,z\_final] = fine\_percision(x\_final,y\_final,...

z\_final,R\_step,delta\_R);

end

if (tt == StartStep && graphcheck == 1)

figure;

scatter3(x\_final,y\_final,z\_final,...

'MarkerEdgeColor','k',...

'MarkerFaceColor',[0 .25 .25]);

view(45,45)

size3 = size(x\_final,1);

str = sprintf('After hit-and-count and fine-percision (%d points were removed)', ...

size2-size3);

title(str);

axis tight

end

% Calculating contact angle distribution and interfacial area using convex hull

[K,V,angle,BaseA\_temp] = contact\_angle(x\_final,y\_final,z\_final,...

z\_min,precision);

if (tt == StartStep && graphcheck == 1)

figure;

hold on;

trisurf(K,x\_final,y\_final,z\_final,'facecolor','r','facealpha',0.5)

scatter3(x\_final,y\_final,z\_final,...

'MarkerEdgeColor','k',...

'MarkerFaceColor',[0 .75 .75], ...

'MarkerFaceAlpha',0.5);

view(45,45)

axis tight

end

BaseA = BaseA + BaseA\_temp;

Distribution = cat(2,Distribution,angle(1,:));

Weight = cat(2,Weight,angle(2,:));

end

fprintf('\n');

% Calculating weighted histogram of angular values

w = weighted\_distribution(Distribution,Weight);

BaseA = 0.5\*BaseA/((EndStep-StartStep)/step+1);

angle = 0:180;

s0=w;

s1=conv(w,hanning(30),'same');

theta\_ave = sum(angle.\*w)/sum(w);

% drawing angular distribution

% drawing angular distribution

figure;

plot(0:180,100\*s1/sum(s1));

xlabel('Contact Angle (degree)');

ylabel('Probability (%)');

str(1) = {'Khalkhali et al. meathod'};

str(2) = {sprintf('\\theta\_{ave}=%f',theta\_ave)};

title(str);

file(:,1)=0:180;

file(:,2)=s0;

file(:,3)=s1;

output = sprintf('../Results/MK.txt');

dlmwrite(output,file,'delimiter','\t');

% toc