## Supporting information

## Multi-scale model Integrated particle dynamics and evolutionary algorithm with the coarse-grained particle partitioning ratio strategy for calculating the dissolution behaviors of supercritical carbon dioxide in polymer fluids

Li Mengshan1, \*, Wu Wei 1, Chen Bingsheng 1, Guan Lixin 1, Wu Yan 1 and Wang Nan 2,\*

1 College of Physics and Electronic Information, Gannan Normal University, Ganzhou, 341000, Jiangxi, China

2 College of Life Sciences, Jiaying University, Meizhou, 514015, Guangdong, China

\*Correspondence: jcimsli@163.com, 201401018@jyu.edu.cn ; Tel.:+867978393668

**Supporting information 3: O**perating environment and code

Soft: Matlab 2010a

OS: Windows 7 SP1 64-bit OS (4.00 GB of memory and Intel (R) Core ™ i5-4460 processor)

Code:

clear all;

clc;

……..

…….

% VARIABLE DESCRIPTION IS GIVEN IN initialize.m

% FUNCTION DESCRIPTION IS AS FOLLOWS:

% initialize(): USED FOR INITIALIZING THE SYSTEM INCLUDING ITS PROPERTIES

% print\_init(): USED FOR PRINTING THE INITIALIZED VALUES TO THE SCREEN

% eval\_force(): USED FOR CALCULATING THE FORCE ON THE PARTICLES

% integrate(): USED FOR SOLVING THE EQUATIONS OF MOTION USING VELOCITY-VERLET

% visualize(): USED FOR INSTANT VISUALIZATION OF THE RESULTS

% print\_traj(): USED FOR PRINTING TRAJECTORY DATA IN LAMMPS FORMAT

global MASS KB TEMPERATURE NUM\_ATOMS LENGTH TSTEP SIM\_NO;

global EPS SIG R\_CUT POT\_E KIN\_E TOT\_E;

global POSITION VELOCITY FORCE;

fid = fopen('output.dat','w');

fidw = fopen('code.m','w');

fid2 = fopen('Trajectory.lammpstrj','w');

tic; %INBUILT VARIABLE TO MEASURE THE PERFORMANCE OF THE CODE

% THE GLOBAL VARIABLES ARE INITIALIZED IN THE FUNCTION

initialize();

% THE GLOBAL VARIABLES ARE PRINTED IN HERE IN THIS FUNCTION

print\_init();

% EVALUATE FORCE FOR THE FIRST TIME

eval\_force();

for(i=1:SIM\_NO)

integrate();

TOT\_E = POT\_E + KIN\_E;

TEMPERATURE = KIN\_E \* 2/(3\*NUM\_ATOMS\*KB);

if(rem(i,50) == 0)

disp(sprintf(['Loop :\t',num2str(i),'\t Potential Energy :\t',num2str(POT\_E),'\t Kinetic Energy :\t',num2str(KIN\_E),'\t Total Energy :\t',num2str(TOT\_E),'\t Temperature:\t',num2str(TEMPERATURE)]));

fprintf(fid,['Loop :\t',num2str(i),'\t Potential Energy :\t',num2str(POT\_E),'\t Kinetic Energy :\t',num2str(KIN\_E),'\t Total Energy :\t',num2str(TOT\_E),'\t Temperature:\t',num2str(TEMPERATURE),'\n']);

visualize();

print\_traj(fid2,i);

end

end

disp(['Total Simulation Time is:',num2str(toc)]);

function eval\_force()

% eval\_force.m IS USED FOR EVALUATING FORCE

% THE STRATEGY USUALLY ADOPTED FOR A LENNARD-JONES OR AS A MATTER OF FACT

% ANY PAIR-WISE INTERACTING SYSTEM IS AS FOLLOWS:

% 1. EVALUATE THE DISTANCE BETWEEN TWO PAIRS OF ATOMS

% 2. ENSURE THAT MINIMUM IMAGE CONVENTION (MIC) IS FOLLOWED

% 3. IF THE DISTANCE OBTAINED THROUGH MIC IS GREATER THAN THE CUT OFF

% DISTANCE MOVE TO NEXT PAIR

% 4. ELSE EVALUATE POTENTIAL ENERGY AND CALCULATE FORCE COMPONENTS

% 5. F(i,j) = -F(j,i)

global NUM\_ATOMS LENGTH;

global EPS SIG R\_CUT POT\_E;

global POSITION FORCE;

dr = zeros(3,1);

FORCE(:) = 0.0;

POT\_E = 0.0;

for ( i=1:NUM\_ATOMS )

for ( j=i+1:NUM\_ATOMS )

dist2 = 0.0;

% VARIABLE dist2 STORES DISTANCE BETWEEN PAIR (i,j)

% FIRST FIND OUT THE DIFFERENCE IN X,Y AND Z COORDINATES

% VARIABLE dr IS USED FOR THIS PURPOSE

for(k = 1:3)

dr(k) = POSITION(i,k) - POSITION(j,k);

% THESE STEPS ENSURE MINIMUM IMAGE CONVENTION IS FOLLOWED

if(dr(k) > LENGTH/2.0)

dr(k) = dr(k) - LENGTH;

end

if(dr(k) < -LENGTH/2.0)

dr(k) = dr(k) + LENGTH;

end

% MINIMUM IMAGE CONVENTION ENDS HERE

dist2 = dist2 + dr(k)\*dr(k);

% dist2 IS BASED UPON MIC

end

if(dist2 <= R\_CUT\*R\_CUT)

% IF THE CUT OFF CRITERIA IS SATISFIED

dist2i = power(SIG,2)/dist2;

dist6i = power(dist2i,3);

dist12i = power(dist6i,2);

POT\_E = POT\_E + 4.0 \* EPS \* (dist12i - dist6i);

% STORES THE POTENTIAL ENERGY

Ff = 24.0 \* EPS \* (2\*dist12i-dist6i);

Ff = Ff \* dist2i;

for(k = 1:3)

FORCE(i,k) = FORCE(i,k) + Ff\*dr(k);

FORCE(j,k) = FORCE(j,k) - Ff\*dr(k);

end

end

end

end

end

function initialize()

% FUNCTION initalize IS USED FOR INITIALIZING ALL THE GLOBAL VARIABLES

% REDUCED SYSTEM IS USED IN THIS ANALYSIS

% DECLARE GLOBAL VARIABLES. THE VARIABLE DESCRIPTIONS ARE GIVEN BELOW:

% MASS = MASS OF AN ATOM

% KB = BOLTZMANN CONSTANT

% TEMPERATURE = TEMPERATURE OF THE SYSTEM

% NUM\_ATOMS = NUMBER OF ATOMS IN THE SYSTEM

% LENGTH = LENGTH OF THE CUBIC SIMULATION BOX

% TSTEP = TIME STEP FOR CONDUCTING THE ANALYSIS

% SIM\_NO = TOTAL NUMBER OF SIMULATIONS

% EPS = DENOTES THE PARAMETER EPSILON OF L-J INTERACTION

% SIG = DENOTES THE PARAMETER SIGMA OF L-J INTERACTION

% R\_CUT = DENOTES THE CUTOFF DISTANCE IN FORCE CALCULATION

% POT\_E = POTENTIAL ENERGY

% KIN\_E = KINETIC ENERGY

% TOT\_E = TOTAL ENERGY

% POSITION = A STRUCTUR VARIABLE DENOTION POSITION AT EACH STEP WITH x,y,z AS EMBEDDED VARIABLES

% VELOCITY = A STRUCTUR VARIABLE DENOTION VELOCITY AT EACH STEP WITH x,y,z AS EMBEDDED VARIABLES

% FORCE = A STRUCTUR VARIABLE DENOTION FORCE AT EACH STEP WITH x,y,z AS EMBEDDED VARIABLES

global MASS KB TEMPERATURE NUM\_ATOMS LENGTH TSTEP SIM\_NO;

global EPS SIG R\_CUT POT\_E KIN\_E TOT\_E;

global POSITION VELOCITY FORCE;

MASS = 1;

KB = 1;

TEMPERATURE = 1.5;

NUM\_PER\_DIM = 5;

NUM\_ATOMS = power(NUM\_PER\_DIM,3);

%LENGTH = 10;

% LENGTH OF THE SYSTEM CAN BE INITIALIZED DIRECTLY

Density=0.5;

% LENGTH OF THE SYSTEM CAN BE INITIALIZED THROUGH DENSITY

LENGTH = power(NUM\_ATOMS/Density,1.0/3.0);

TSTEP = 1e-3;

SIM\_NO = 1000000;

EPS = 1.0;

SIG = 1.0;

R\_CUT = 2.5\*SIG;

POT\_E = 0.0;

KIN\_E = 0.0;

TOT\_E = 0.0;

POSITION = zeros(NUM\_ATOMS,3);

VELOCITY = zeros(NUM\_ATOMS,3);

FORCE = zeros(NUM\_ATOMS,3);

% NOW WE ARE INITIALIZING THE LOCATION OF THE PARTICLES. THE PARTICLES ARE

% INITIALIZED USING A SIMPLE CUBIC SYSTEM WITH LATTICE CONSTANT REDUCED TO

% OVERCOME OVERLAPPING OF TWO PARTICLES IN SUBSEQUENT STEPS

Number=NUM\_PER\_DIM;

Lat\_Const = LENGTH /(Number + 2.0);

Nplace=1;

for(i=1:Number)

for(j=1:Number)

for(k=1:Number)

if(Nplace<=NUM\_ATOMS)

POSITION(Nplace,1)= i\*Lat\_Const;

POSITION(Nplace,2)= j\*Lat\_Const;

POSITION(Nplace,3)= k\*Lat\_Const;

end

Nplace = Nplace + 1;

end

end

end

end

function integrate()

%integrate.m IS USED FOR INTEGRATING THE SOLUTIONS USING VELOCITY VERLET METHOD

% VELOCITY VERLET METHOD CALCULATES UPDATED POSITION AND VELOCITY OF ALL

% THE PARTICLES AS FOLLOWS:

% X(T+dT) = X(T) + V(T)dT + 0.5 A(T)dT\*dT

% V(T+dT) = V(T) + 0.5(F(T+dT) + F(T))/MASS

global MASS NUM\_ATOMS TSTEP;

global KIN\_E LENGTH;

global POSITION VELOCITY FORCE;

old\_force = FORCE; %OLD FORCE IS NEEDED TO EVALUATE VELOCITY

for(i = 1:NUM\_ATOMS)

for(j = 1:3)

POSITION(i,j) = POSITION(i,j) + VELOCITY(i,j)\*TSTEP + 0.5\*FORCE(i,j)\*TSTEP\*TSTEP/MASS;

% THESE STEPS ENSURE PERIODIC BOUNDARY CONDITIONS IN THE SYSTEM

if(POSITION(i,j) > LENGTH)

POSITION(i,j) = POSITION(i,j) - LENGTH;

end

if(POSITION(i,j) < 0.0)

POSITION(i,j) = POSITION(i,j) + LENGTH;

end

% PERIODIC BOUNDARY CONDITIONS END OVER HERE

end

end

eval\_force();

KIN\_E = 0.0;

for(i = 1:NUM\_ATOMS)

for(j = 1:3)

VELOCITY(i,j) = VELOCITY(i,j) + 0.5\*(FORCE(i,j) + old\_force(i,j))\*TSTEP/MASS;

% KINETIC ENERGY IS CALCULATED NEXT

KIN\_E = KIN\_E + 0.5\*power(VELOCITY(i,j),2);

end

end

end

function print\_init()

%print\_init.m prints the initially global variables

global MASS KB TEMPERATURE NUM\_ATOMS LENGTH TSTEP SIM\_NO;

disp(sprintf(['SIMULATION HAS BEGUN. \n PLEASE GIVE CREDITS TO AFRRE LAB, IIT KHARAGPUR WHILE USING THE CODE \n']))

disp(sprintf(['Mass of the particles is ::\t\t',num2str(MASS)]));

disp(sprintf(['Boltzmann Constant is ::\t\t',num2str(KB)]));

disp(sprintf(['Temperature of the system is ::\t\t',num2str(TEMPERATURE)]));

disp(sprintf(['Number of Particles ::\t\t',num2str(NUM\_ATOMS)]));

disp(sprintf(['Temperature of the system is ::\t\t',num2str(TEMPERATURE)]));

disp(sprintf(['Box Length is ::\t\t',num2str(LENGTH)]));

disp(sprintf(['Time Step for analysis is ::\t\t',num2str(TSTEP)]));

disp(sprintf(['Total Number of Simulations ::\t\t',num2str(SIM\_NO)]));

end

……….

………

………

function print\_traj(fid2,time\_step)

%This function prints the trajectory data in the LAMMPS FORMAT for

%visualization in VMD

global POSITION VELOCITY NUM\_ATOMS LENGTH;

fprintf(fid2,['ITEM: TIMESTEP :\n',num2str(time\_step)]);

fprintf(fid2,['\nITEM: NUMBER OF ATOMS :\n',num2str(NUM\_ATOMS)]);

fprintf(fid2,['\nITEM: BOX BOUNDS ff ff ff :\n',num2str(0),'\t',num2str(LENGTH)]);

fprintf(fid2,['\n',num2str(0),'\t',num2str(LENGTH)]);

fprintf(fid2,['\n',num2str(0),'\t',num2str(LENGTH)]);

fprintf(fid2,['\nITEM: ATOMS id type xu yu zu vx vy vz :\n']);

for(i = 1:NUM\_ATOMS)

fprintf(fid2,[num2str(i),'\t',num2str(0),'\t',num2str(POSITION(i,1)),'\t',num2str(POSITION(i,2)),'\t',num2str(POSITION(i,3)),'\t',num2str(VELOCITY(i,1)),'\t',num2str(VELOCITY(i,2)),'\t',num2str(VELOCITY(i,3)),'\n']);

end

end

function visualize()

global POSITION

plot3(POSITION(:,1),POSITION(:,2),POSITION(:,3),'.b');

pause(0.005);

end

CSPSO

%%

clear

clc

tic

%%

%

c1 = %%;

c2 = %%;

maxgen = 100000; %

sizepop = 50000; %

Vmax = 1;

Vmin = -1;

popmax = 5;

popmin = -5;

%%

for i = 1:sizepop

%

pop(i,:) = 5 \* rands(1,2); %

V(i,:) = rands(1,2); %

%

fitness(i) = fun(pop(i,:)); %

end

%

[bestfitness bestindex] = min(fitness);

zbest = pop(bestindex,:); %

gbest = pop; %

fitnessgbest = fitness; % best

fitnesszbest = bestfitness; % gbest

%%

for i = 1:maxgen

for j = 1:sizepop

% V

V(j,:) = V(j,:) + c1\*rand\*(gbest(j,:) - pop(j,:)) + c2\*rand\*(zbest - pop(j,:));

V(j,find(V(j,:)>Vmax)) = Vmax;

V(j,find(V(j,:)<Vmin)) = Vmin;

%pop update

pop(j,:) = pop(j,:) + 0.5\*V(j,:);

pop(j,find(pop(j,:)>popmax)) = popmax;

pop(j,find(pop(j,:)<popmin)) = popmin;

% Self

if rand > 0.8

k = ceil(2\*rand);

pop(j,k) = rand;

end

% fitness

fitness(j) = fun(pop(j,:));

end

% gbest

if fitness(j) < fitnessgbest(j)

gbest(j,:) = pop(j,:);

fitnessgbest(j) = fitness(j);

end

% Update

if fitness(j) < fitnesszbest

zbest = pop(j,:);

fitnesszbest = fitness(j);

end

yy(i) = fitnesszbest;

end

toc

%% Result

plot(yy);

title(['IT ' 'iter=' num2str(maxgen)]);

xlabel('It');

ylabel('Fitness');

% fun function

function y = fun(x)

y = %%%;