***MATLAB Script for Creating Energy Map***

% This MATLAB script takes an .xyz and a per-atom breakdown of energy

% And produces a surface for each (pair, bond, angle, dihedral, improper, kspace)

% Jacob Gissinger - 2016

clear**;**clc**;**

% let's first import coordinates and import and sort energies

% let's then import energies, recording timestep for each

% these are file paths to be edited

output\_filename **=** 'sample\_LAMMPS\_per\_atom.output'**;** %created via per/atom LAMMPS command

xyz\_filename **=** 'sample\_coordinates.xyz'**;**

lammps\_output **=** 'sample\_LAMMPS.log'**;**

% how many timesteps in your trajectory?

num\_timesteps **=** 124**;** % tested by counting, e.g. grep -o 'ITEM: TIMESTEP' sample\_LAMMPS\_per\_atom.output | wc -l

% the following paragraph depends on chosen format of LAMMPS output.

% can be created with HOME->Import Data->Generate Script

filename **=** lammps\_output**;**

delimiter **=** **{**','**,**' '**};**

formatSpec **=** '%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%[^\n\r]'**;**

fileID **=** fopen**(**filename**,**'r'**);**

dataArray **=** textscan**(**fileID**,** formatSpec**,** 'Delimiter'**,** delimiter**,** 'MultipleDelimsAsOne'**,** true**,** 'EmptyValue' **,**NaN**,** 'ReturnOnError'**,** false**);**

fclose**(**fileID**);**

thermo\_data **=** **[**dataArray**{**1**:end-**1**}];**

clearvars filename delimiter formatSpec fileID dataArray ans**;**

% this find the beginning of the thermo output section

% it skips any minimization section with one frame output

**for** i **=** 1**:**size**(**thermo\_data**,**1**)**

**if** strcmp**(**'Step'**,**thermo\_data**(**i**,**1**))**

**if** strcmp**(**'Loop'**,**thermo\_data**(**i**+**2**,**1**))**

**continue;**

**end**

thermo\_start **=** i **+** 1**;**

**end**

**if** strcmp**(**'=>>'**,**thermo\_data**(**i**,**1**))**

thermo\_end **=** i **-** 1**;**

**end**

**end**

thermo\_data **=** cellfun**(@**str2double**,**thermo\_data**(**thermo\_start**:**thermo\_end**,:));**

% the following may need modification based on the format of the per-atom dump file

delimiter **=** ' '**;**

formatSpec **=** '%s%s%s%s%s%s%s%s%\*s%\*s%[^\n\r]'**;**

pa\_pe\_fileID **=** fopen**(**output\_filename**,**'r'**);**

textscan**(**pa\_pe\_fileID**,** formatSpec**,** 3**,** 'Delimiter'**,** delimiter**,** 'MultipleDelimsAsOne'**,** true**,** 'EmptyValue' **,**NaN**,** 'ReturnOnError'**,** false**);**

num\_atoms **=** textscan**(**pa\_pe\_fileID**,** formatSpec**,** 1**,** 'Delimiter'**,** delimiter**,** 'MultipleDelimsAsOne'**,** true**,** 'EmptyValue' **,**NaN**,** 'ReturnOnError'**,** false**);**

num\_atoms **=** str2double**(**num\_atoms**{**1**});**

frewind**(**pa\_pe\_fileID**);**

num\_header\_lines **=** 9**;**

timestep **=** zeros**(**1**,**num\_timesteps**);**

% this is number of per-atom energy contributions dumped to file

num\_properties **=** 6**;**

% we now loop through all frames!

**for** frame **=** 1**:**num\_timesteps

% tic; optional to record loop performance

% first retrieve per-atom info for current frame

formatSpec **=** '%s%s%s%s%s%s%s%s%\*s%\*s%[^\n\r]'**;**

temp\_ts **=** textscan**(**pa\_pe\_fileID**,** formatSpec**,** 2**,** 'Delimiter'**,** delimiter**,** 'MultipleDelimsAsOne'**,** true**,** 'EmptyValue' **,**NaN**,** 'ReturnOnError'**,** false**);**

temp\_ts **=** str2double**([**temp\_ts**{**1**}]);**

timestep**(**frame**)** **=** temp\_ts**(**2**,**1**);**

temp\_ts **=** textscan**(**pa\_pe\_fileID**,** formatSpec**,** num\_header\_lines **-** 2**,** 'Delimiter'**,** delimiter**,** 'MultipleDelimsAsOne'**,** true**,** 'EmptyValue' **,**NaN**,** 'ReturnOnError'**,** false**);**

formatSpec **=** '%d%\*s%f%f%f%f%f%f%\*s%\*s%[^\n\r]'**;**

frame\_pa\_pe **=** textscan**(**pa\_pe\_fileID**,** formatSpec**,** num\_atoms**,** 'Delimiter'**,** delimiter**,** 'MultipleDelimsAsOne'**,** true**,** 'EmptyValue' **,**NaN**,** 'ReturnOnError'**,** false**);**

**[**B**,**sort\_indices**]** **=** sort**([**frame\_pa\_pe**{**1**}]);**

frame\_pa\_pe **=** **[**frame\_pa\_pe**{**2**:**7**}];**

frame\_pa\_pe **=** frame\_pa\_pe**(**sort\_indices**,:);**

% now we match an xyz to that frame info

xyz\_fileID **=** fopen**(**xyz\_filename**,**'r'**);**

**for** i **=** 1**:**num\_timesteps

formatSpec **=** '%s%s%s%[^\n\r]'**;**

temp\_ts **=** textscan**(**xyz\_fileID**,** formatSpec**,** 2**,** 'Delimiter'**,** delimiter**,** 'MultipleDelimsAsOne'**,** true**,** 'ReturnOnError'**,** false**);**

temp\_ts **=** str2double**([**temp\_ts**{**3**}]);**

xyz\_timestep **=** temp\_ts**(**2**);**

formatSpec **=** '%d%f%f%f%[^\n\r]'**;**

**if** xyz\_timestep **==** timestep**(**frame**);**

type\_xyz **=** textscan**(**xyz\_fileID**,** formatSpec**,** num\_atoms**,** 'Delimiter'**,** delimiter**,** 'MultipleDelimsAsOne'**,** true**,** 'EmptyValue' **,**NaN**,** 'ReturnOnError'**,** false**);**

types **=** **[**type\_xyz**{**1**}];**

xyz **=** **[**type\_xyz**{**2**:**4**}];**

**break;**

**else**

textscan**(**xyz\_fileID**,** formatSpec**,** num\_atoms**,** 'Delimiter'**,** delimiter**,** 'MultipleDelimsAsOne'**,** true**,** 'EmptyValue' **,**NaN**,** 'ReturnOnError'**,** false**);**

**end**

**end**

% USER MODIFICATION REQUIRED:

% when using NPT ensemble, we must find box size for each timestep

**for** i **=** 1**:**size**(**thermo\_data**,**1**)**

**if** thermo\_data**(**i**,**1**)** **==** timestep**(**frame**)**

box\_dims **=** **[**thermo\_data**(**i**,**19**)** thermo\_data**(**i**,**20**);** % 19, 20 should be modified to correspond to colomn number for xlo, xhi thermo output

thermo\_data**(**i**,**21**)** thermo\_data**(**i**,**22**);** % 12, 22 should be modified to correspond to colomn number for ylo, yhi thermo output

thermo\_data**(**i**,**23**)** thermo\_data**(**i**,**24**)];** % 23, 24 should be modified to correspond to colomn number for zlo, zhi thermo output

**break;**

**end**

**end**

lx **=** diff**(**box\_dims**(**1**,:));** ly **=** diff**(**box\_dims**(**2**,:));** lz **=** diff**(**box\_dims**(**3**,:));**

side\_lengths **=** **[**lx ly lz**];**

% now we add ghost atoms to xyz

% ghost atoms are artifact of LAMMPS' parallelism

cutoff **=** 8**;** %choose cutoff to consider ghost atoms for interpolation

% let's add ghost atoms

ghost\_atoms **=** zeros**(**0**,**3**);**

ghost\_pa\_pe **=** zeros**(**0**,**num\_properties**);**

% The following was required to properly take into account periodic boundaries:

% images\_mask: column 1 = atoms to move negatively in x by box length

% column 2 = atoms to move negatvely in y by box length

% column 3 = atoms to move negatvely in z by box length

% positive for x, y, z for columns 4, 5, 6, resp.

images\_mask **=** false**(**num\_atoms**,**0**);**

**for** i **=** **[**2 1**]**

**for** j **=** **[**1 2 3**]**

**if** i **==** 2

images\_mask**(:,end+**1**)** **=** xyz**(:,**j**)** **>** **(**box\_dims**(**j**,**i**)** **-** cutoff**);**

**elseif** i **==** 1

images\_mask**(:,end+**1**)** **=** xyz**(:,**j**)** **<** **(**box\_dims**(**j**,**i**)** **+** cutoff**);**

**end**

**end**

**end**

% now we're ready to annex all our ghosts

% first level takes care of first degree periodicities (on faces)

**for** i **=** **[-**3 **-**2 **-**1 1 2 3**]**

% index helper is 0 if i is negative, 1 if i is positive

index\_helper1 **=** abs**(**i**)+**3**\***sign**(**sign**(**i**)+**1**);**

this\_ghost\_mask **=** images\_mask**(:,**index\_helper1**);**

temp\_ghosts **=** xyz**(**this\_ghost\_mask**,:);**

temp\_ghosts**(:,**abs**(**i**))** **=** ...

temp\_ghosts**(:,**abs**(**i**))** **+** sign**(**i**)\***side\_lengths**(**abs**(**i**));**

ghost\_atoms**(end+**1**:end+**size**(**temp\_ghosts**,**1**),:)** **=** temp\_ghosts**;**

ghost\_pa\_pe**(end+**1**:end+**size**(**temp\_ghosts**,**1**),:)** **=** frame\_pa\_pe**(**this\_ghost\_mask**,:);**

% second level takes care of second degree periodicities (along edges)

**for** j **=** **[-**3 **-**2 **-**1 1 2 3**]**

**if** **(**abs**(**i**)** **==** abs**(**j**))** **||** **(**j **<** i**);**

**continue**

**end**

% index helper is 0 if j is negative, 1 if j is positive

index\_helper2 **=** abs**(**j**)+**3**\***sign**(**sign**(**j**)+**1**);**

this\_ghost\_mask **=** images\_mask**(:,**index\_helper1**)** **&** ...

images\_mask**(:,**index\_helper2**);**

temp\_ghosts **=** xyz**(**this\_ghost\_mask**,:);**

temp\_ghosts**(:,**abs**(**i**))** **=** ...

temp\_ghosts**(:,**abs**(**i**))** **+** sign**(**i**)\***side\_lengths**(**abs**(**i**));**

temp\_ghosts**(:,**abs**(**j**))** **=** ...

temp\_ghosts**(:,**abs**(**j**))** **+** sign**(**j**)\***side\_lengths**(**abs**(**j**));**

ghost\_atoms**(end+**1**:end+**size**(**temp\_ghosts**,**1**),:)** **=** temp\_ghosts**;**

ghost\_pa\_pe**(end+**1**:end+**size**(**temp\_ghosts**,**1**),:)** **=** frame\_pa\_pe**(**this\_ghost\_mask**,:);**

**end**

**end**

% for corner ghosts, let's do something a little different

temp **=** combnk**([-**1 **-**2 **-**3 1 2 3**],**3**);**

corner\_dims **=** temp**(**ismember**(**sort**(**abs**(**temp**),**2**),[**1 2 3**],**'rows'**),:);**

**for** i **=** 1**:**size**(**corner\_dims**,**1**)**

index\_helpers **=** abs**(**corner\_dims**(**i**,:))** **+** 3.**\***sign**(**sign**(**corner\_dims**(**i**,:))+**1**);**

this\_ghost\_mask **=** images\_mask**(:,**index\_helpers**(**1**))** **&** ...

images\_mask**(:,**index\_helpers**(**2**))** **&** ...

images\_mask**(:,**index\_helpers**(**3**));**

temp\_ghosts **=** xyz**(**this\_ghost\_mask**,:);**

temp\_ghosts**(:,**1**:**3**)** **=** temp\_ghosts**(:,**1**:**3**)** **+** ...

repmat**(**sign**(**corner\_dims**(**i**,:)).\***side\_lengths**(**abs**(**corner\_dims**(**i**,:))),** ...

size**(**temp\_ghosts**,**1**),**1**);**

ghost\_atoms**(end+**1**:end+**size**(**temp\_ghosts**,**1**),:)** **=** temp\_ghosts**;**

ghost\_pa\_pe**(end+**1**:end+**size**(**temp\_ghosts**,**1**),:)** **=** frame\_pa\_pe**(**this\_ghost\_mask**,:);**

**end**

ghosted\_xyz **=** **[**xyz**;** ghost\_atoms**];**

ghosted\_pa\_pe **=** **[**frame\_pa\_pe**;** ghost\_pa\_pe**];**

num\_interp\_points **=** 100**;**

**[**X**,**Y**,**Z**]** **=** meshgrid**(**linspace**(**box\_dims**(**1**,**1**),**box\_dims**(**1**,**2**),**num\_interp\_points**),** ...

linspace**(**box\_dims**(**2**,**1**),**box\_dims**(**2**,**2**),**num\_interp\_points**),** ...

linspace**(**box\_dims**(**3**,**1**),**box\_dims**(**3**,**2**),**num\_interp\_points**));**

**for** i **=** 1**:**num\_properties

interp\_pa\_pe\_struct **=** scatteredInterpolant**(**ghosted\_xyz**(:,**1**),** ...

ghosted\_xyz**(:,**2**),** ...

ghosted\_xyz**(:,**3**),** ...

ghosted\_pa\_pe**(:,**i**));**

interped\_pa\_pe **=** interp\_pa\_pe\_struct**(**X**,**Y**,**Z**);**

% this is most important output variable.

% it stores the axial average of each frame, for each energy contribution

% it can then be averaged again over the trajectory for analysis, visualization

xy\_interped\_pa\_pe**(:,:,**frame**,**i**)** **=** mean**(**interped\_pa\_pe**,**3**);**

% we're done!

disp**([**'Done with prop ' num2str**(**i**)]);**

% time\_frame = toc;

% disp(['Done with frame ' num2str(frame) ', which took ' num2str(time\_frame) ' seconds to complete']); % optional output speed of each frame loop

**end**

**end**

fclose**(**pa\_pe\_fileID**);**

fclose**(**xyz\_fileID**);**