viscosity_nemd Package

by Lingnan Lin, April 2020

viscpost.py – *Module for post-processing viscosity data*

import viscpost as vp

vd = loadvisc(filename, ifplot=True)	*load a visc_ file by file name (could be path)*	*ViscData*
vd = readvisc('PEC6', 373,0.1,1e8,ifplot=True)	*Load a visc_file by information*	*ViscData*
vba = batch('PEC6',373,0.1)	*Load a batch of visc_file and calculate the average and error*	*ViscBatch*

Class ViscData

vd.material	*PEC5 or PEC6*	*string*
vd.temp	*temperature [K]*	*string*
vd.press	*pressure [MPa]*	*string*
vd.srate	*shear rate [s-1], eg 1e+08*	*string*
vd.dt	*timestep [fs], =0.5*	*float*
vd.step	*step*	*Series*
vd.time	*time [ns]*	*Series*
vd.strain	*strain*	*Series*
vd.visc	*viscosity [mPa s]*	*Series*
vd.sslength	*length of steady steady [ns], = 20*	*int*
vd.ssdata	*steady-state data*	*DataFrame*
vd.outputreq	*output frequency, = 100000*	*int*

s = vd.info(ifprint=True)	*return 'PEC5, 340K, 50MPa, 1e+08 1/s'*
vd.plot(window=50)	*plot for all the data*
vd.ssplot(window=50)	*plot only for the steady-state data*
vd.acf(data=None,Nblock=4,lags=50)	*perform autocorrelation analysis*
vd.setss(ts)	*set steady-state starting at ts [ns]*
vd.setss1(t_begin,t_end)	*set steady-state from … to …*
mean, error = vd.average(blocknum = 10)	*perform block average*

Class BatchData

vba.material	*PEC5 or PEC6*	*string*
vba.temp	*temperature [K]*	*string*
vba.press	*pressure [MPa]*	*string*
vba.srate	*shear rate [s-1], eg 1e+08*	*string*
vba.visclist	*ViscData obejcts for various sheat rates*	*list of ViscData*
vba.results	*a table of viscosity average and error for various shear rates. Columns: srate, mean, erorr, relative error*	*DataFrame*
vd = vba.get(srate)	*retrieve the ViscData via the corresponding strain rate as the key*	*ViscData*
vba.plot(model=Eyring,color='b', xlim=(1e6,1e11),ylim=(1,100)	*plot viscosity vs. shear rate fit the data to a model (Eyring by default)*	-
vba.axplot(ax,same as above)	*for multiple plots (also returns optimized parameters and the parameter standard errors)*	*[eta_N,error*], [sigma_E,error], ax*
vba.plotall()	*plot viscosity vs. time for all shear rates*	
vba.fit(model)	*fit results to a model (Eyring, Carreau)*	*popt: array per*: array *standard error*
vba.export()	*export the results to a file (.nemd) for the use of OriginLab plot*	
vba.print()	*print results to screen*	

lmpoutpost.py – *Module for post-processing general LAMMPS outputs*

import lmpoutpost as lmp

df = loadvisc(filename)	*load a visc_ file by file name (could be path)*	*DataFrame*
plot(df,dt=0.5, title=None,window=100)	*general plot, capable of multiple variables*	-
plot1(df,dt=0.5,title=None, window=100,sharex=True):	*plot a single varial against time*	*fig, ax*
blockACF(df,Nblock=4,lags=50, t0=0.5,outputfreq=100000)	*perform a block analysis*	-
mean, expanderror = blockAverage(df,b,style='blocknum'):	*perform block average. b is block number or block size depending style = 'blocknum' or 'blocksize'*	*float, float*
blockSizing(df, isplot=True, maxBlockSize=0):	*compute block average & error for varied sizes*	-

utility.py – *Module for quick visualizing and handling lammps output*

Automatically run at startup of Jupyter Lab

plot(filename)	**vba= analyze(material,temp,press)**	**copy()**

General workflow

1. Use the scripts in /lmpscript to run equilbration and NEMD simulations using LAMMPS on a HPC server.
2. When computation completed, download the files from the server to: ./data/new.
3. Open a iPython-like terminal or a Juypter notebook, cd to ./data/new, import the modules in /src to post-process and analyze the data. Here are some tips:
   - Use vba = analyze() to quick-check all the viscosity output files in ./data/new. vba is a BatchData class that has a bunch of useful functions you can play with to analyze the viscosity data.
   - Use plot('filename') to visualize the output files that compute pressure, energy, etc. as a function of time. This is primarily to check if the equilibration or steady state is reached.
   - Use vd = vba.get(srate) to quickly retrieve a dataset for a state point where srate is the shear rate (1/s), e.g, 1e8, 1e9. vd is a ViscData class that also has a bunch of useful functions for analysis.
   - Deep steady-state check: use vd.acf(), vd.ssplot(), vd.setss1(), etc
4. If steady-state is reached and desired statistical accuracy has been achieved, run copy('MaterialName') to copy the visc_ file to the ./data/visc. Move all files in ./data/new to ./data/archive. Otherwise go back to LAMMPS for longer simulation until obtaining the desired results.
5. Create a Jupyter notebook to do analysis and write report using the modules in ./src and the data in ./data/visc. Export results if necessary for publication and making figures using other software..

Some default values:

> temp, press

> t0=0.5, outputfreq=100000

> #block=10. To change, go to *viscpost->ViscData->def average*