Molecular dynamics simulations of 2D-confined water

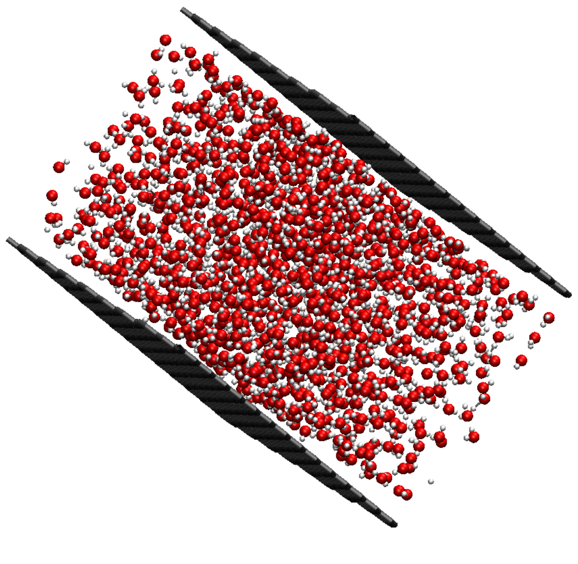
Thorin Bristow

School of Physics and Astronomy
University of Manchester
MPhys Project

This project was performed in collaboration with Izaak Lee.

January 2021

The aim of this project was to study the properties of water using molecular dynamics simulations, and in particular investigate the confined case. The structural properties, rather than thermodynamical and transport quantities, were primarily investigated. Compelling results were obtained by simulating the structural properties as a function of density, which may offer an explanation for recent experimental observations of the electronic properties of confined water at the interface.



Contents

1. Introduction

The development of graphene and other two-dimensional materials has created opportunities to investigate the properties of confined water in new ways. In 2018 Fumagalli *et al.* showed by experiment that interfacial water confined to the nanometer scale exhibits an anomalously low value of the dielectric constant [1] demonstrating a technique for investigating the dielectric properties of fluids under extreme confinement. This discovery is significant as the dielectric constant of water directly affects various forces between micro-objects and macromolecules. Furthermore, these forces determine numerous phenomena in the natural world, including solubility of molecules and ions, surface hydration processes, molecular structuring, and chemical reactions [2]. Experimental and theoretical advances in our understanding of the properties of aqueous interfaces have been reviewed by Björneholm *et al.* [3]. This project aims to assist these new experiments with a theoretical study of the structural properties of water at the interface with solid surfaces, and confined in atomically thin slits, using molecular dynamics (MD) simulations. This semester it was first necessary to learn the main concepts of MD simulation, gain familiarity with the relevant software packages, and practise running simulations. Then we were able to write various programs to analyse the trajectories obtained by simulations, first using bulk water, and subsequently for the case of water confined between graphite interfaces. For this part of the project, we focussed on the structural properties of interfacial water that have been previously studied [4–6], so that we had published results to make comparisons with. In the second semester this study will be extended to the case of a mica interface, which has not been studied previously.

Water is arguably the most important chemical compound on the Earth's surface, and is a principal constituent of all living organisms [7]. The properties of bulk water arise primarily from the network of hydrogen bonds in its structure. Extreme confinement of water substantially modifies the structural, thermodynamic, and dynamic (transport) properties of water, as it changes the structure of the hydrogen bond network [8]. Water confined to narrow slits exhibits an ordered layering structure parallel to the slit walls, and a thin interfacial (depletion) layer whose thickness is independent of the slit separation [4].

MD simulations are widely used in a variety of fields, including biophysics, chemistry, and materials science. MD simulation involves numerical modelling of the trajectories of single atoms comprising a system, and is a technique for computing the properties of a classical many-body system, where the term *classical* denotes that the motion of the constituent particles obeys the laws of classical mechanics [9]. This is a useful approximation that can accurately be applied to a wide variety of systems. The properties of the system can then be derived from the atomic trajectories using statistical physics. The first computer simulations of molecular liquids focussed on water [10, 11], and more than 80,000 simulations of water have been published since [12]. Many of the developments in simulated water have focussed on improving models of the intermolecular potential between the molecules. For classical simulations, three different types of potential are used: rigid, flexible, and polarizable. The most simple rigid models, such as the SPC or SPCE models (see Section 2.2), use a Lennard-Jones potential at the oxygen atoms, and three electrostatic charges, each positioned at the oxygen and hydrogen atom centres [12].

In this project, the structural properties of bulk water, and water confined between two graphene sheets, were investigated using MD simulations. The objective was to compare the effect of density of bulk water, and interfacial water layers, on the following properties: the number of hydrogen bonds per molecule, the density profile, and radial pair distribution functions. Classical MD was used (rather than *ab-inito*) for simplicity. LAMMPS[1] was used for the simulation [13]. Any visualisations in this report were generated using VMD[2] software [14].

2. Methodological approach

2.1. Molecular dynamics

Molecular dynamics (MD) simulations are much like real experiments in many key aspects. Firstly, a sample is prepared: a model system configuration of N particles is selected, and Newton's equations of motion for this system are solved until the properties of the system stabilise and no longer change with time [9]. This is the process of equilibration. Afterwards actual measurements are performed to determine various properties of the simulated many-body system. It can be useful to measure quantities that can be compared with real experiments. Simple examples are thermodynamic properties such as temperature and pressure. Another group of observable properties are functions that characterise the local structure of the liquid. One such function is the radial (pair) distribution function $g(r)$ (see Section 2.6).

The potential energy of a system containing N atoms can be divided into interactions depending on the coordinates of the individual atoms, pairs, triplets...etc. as follows

$$V(\mathbf{r}) = \sum_i V_1(\mathbf{r}_i) + \sum_i \sum_{j>i} V_2(\mathbf{r}_i, \mathbf{r}_j) + \sum_i \sum_{j>i} \sum_{k>j} V_3(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k) + \dots \tag{1}$$

where the $\sum_i \sum_{j>i}$ notation represents a summation over all distinct pairs i and j without counting any pair twice etc. The first term in equation 1 accounts for any external field acting on the system, while the remaining terms represent the interactions between the particles. The second term, the pair potential, is crucial, and is dependent only on the magnitude of the separation $r_{ij} = |\mathbf{r}_{ij}| = |\mathbf{r}_i - \mathbf{r}_j|$. Hence, it could be written $V_2(r_{ij})$.

Simulations are generally performed on relatively small samples of particles, as the size of the system is limited by both the storage available on the host computer, and the time taken for the program to be executed. The most time-consuming part of MD simulations is the calculation of the forces acting on each particle, where the time needed to evaluate the forces scales as N^2.

[1]Large-scale Atomic/Molecular Massively Parallel Simulator

[2]Visual Molecular Dynamics `http://www.ks.uiuc.edu/Research/vmd/`

Verlet algorithm The Verlet algorithm [15] is a second order algorithm used for the numerical integration of the equations of motion of many body systems from the trajectories of particles obtained from Newton's second law

$$\mathbf{F} = m\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t}. \tag{2}$$

This is not solvable analytically for more than three particles, rather a numerical solution is found by discretising time by intervals of dt,

$$\mathbf{r}(t+\mathrm{d}t) = \mathbf{r}(t) + \mathbf{v}(t)\cdot \mathrm{d}t + \frac{\mathbf{F}(t)}{2m}\cdot \mathrm{d}t^2 + O(\mathrm{d}t^3). \tag{3}$$

Replacing dt with $-$dt in equation 3 and combining the two equations, we obtain

$$\mathbf{r}(t+\Delta t) = 2\mathbf{r}(t) - \mathbf{r}(t-\Delta t) + \frac{\mathbf{F}(t)}{m}\cdot \Delta t^2 + O(\Delta t^4) \tag{4}$$

reaching the integration step

$$\mathbf{v}(t) = \frac{\mathbf{r}(t+\Delta t) - \mathbf{r}(t-\Delta t)}{2\Delta t} + O(\Delta t^2). \tag{5}$$

Periodic boundary conditions A major obstacle in the simulation of bulk liquid using a small system is that a large fraction of the configuration lies at the surface. For example, a sample of 1000 molecules configured in a 10×10×10 cube will have $8^3 = 512$ molecules in the interior, while 488 are positioned on the faces of the cube – nearly half of the sample. This is important, as molecules on the surface experience very different forces from those in the bulk [12]. Fortunately, surface effects can be overcome through the implementation of periodic boundary conditions [16]. This means that the configuration box is effectively replicated through space, forming an infinite lattice. In the simulation, as molecules move in the original configuration box, identical images in all neighbouring boxes move in the same way. Thus, when a molecule passes through a surface, 'leaving' the original box, its periodic image enters the box through the opposite surface. Hence, there is no effective boundary, or surface, at the edges of the configuration box, and the simulated molecules experience no surface effects.

An example of code used for incorporating periodic boundary conditions into our simulation is shown:

```
for j in range(start,end):     #for every atom
    if atom_type[j]==2:        #that is also oxygen
        if i!=j:               #other than atom_i
            x_j=x_pos[j]       #note position of atom_j
            y_j=y_pos[j]
            z_j=z_pos[j]
            if x_i-x_j>Lx/2:
                x_j=x_j+Lx
```

```
if x_i-x_j<-Lx/2:   #if atom_j is further than L/2 away
    x_j=x_j-Lx      #in a given co-ordinate then shift
if y_i-y_j>Ly/2:    #to the nearer atom by moving over
    y_j=y_j+Ly      #to the opposite 'box' by moving -L
if y_i-y_j<-Ly/2:
    y_j=y_j-Ly
```

Thermalisation The process of equilibriation is also crucial as it allows the energy of the system to converge, so that the total energy of the system remains constant during the production run (where all the desired data is produced for analysis). The convergence of the energy is a sign that the system has reached thermodynamic equilibrium, and all the computed averages of the system only have physical sense at thermodynamic equilibrium. Thus it is also necessary to know which statistical ensemble is being used. In our case, the NVT ensemble (canonical ensemble) was used as it conveniently allows for the control of the temperature using a thermostat.

2.2. SPCE model

We used the classical extended simple point charge (SPCE) model for water simulation, first introduced by Berendsen *et al.* [17]. This is a three-site model, having three point charges (interaction points) located at the nuclear positions of the oxygen and hydrogen atoms, with charges equal to $-0.8476e$ and $+0.4238e$ respectively. Each site has parameters for intermolecular electrostatic interactions, while the oxygen atoms also interact through a Lennard-Jones potential, given by

$$V_{\mathrm{LJ}} = -(A/r)^6 + (B/r)^{12} \tag{6}$$

where $A = 0.37122$ (kJ/mol)$^{1/6}$·nm, $B = 0.3428$ (kJ/mol)$^{1/12}$·nm, and r is the separation between interacting pairs.

Similarly to the SPC (simple point charge) model [18], the hydrogen-oxygen (HO) bond length is fixed at 1Å and the hydrogen-oxygen-hydrogen (HOH) bond angle at 109.5 degrees. However, the SPCE model adds an average polarization correction (self-energy correction) to the energy

$$\mathrm{U} = \mathrm{E}_{\mathrm{el}} + \mathrm{E}_{\mathrm{pol}} \tag{7}$$

where E_{el} is the electrostatic energy, and

$$\mathrm{E}_{\mathrm{pol}} = \frac{1}{2}\sum_i \frac{(\mu - \mu^0)^2}{\alpha_i} \tag{8}$$

where μ is the dipole moment of the effective pair potential, μ^0 is the dipole moment of the isolated molecule, and α_i is an isotropic scalar polarizability constant. This self-energy correction improves upon the SPC effective pair model for water, producing a more accurate radial distribution function, and better values of density and diffusion constant.

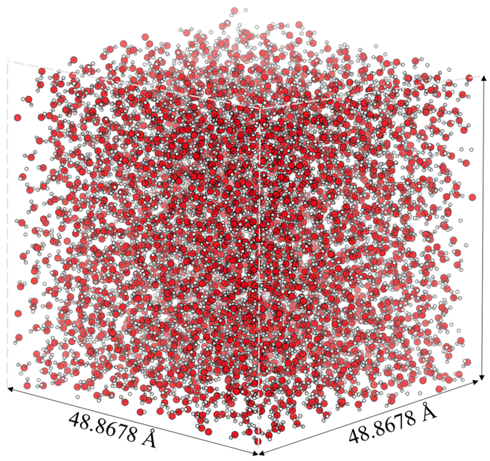
The SHAKE procedure [19] was used for the constraint algorithm to conserve intermolecular/internal constraints such as bond geometry (rigidity). SHAKE is based on the Verlet algorithm [15], and uses Cartesian coordinates.

2.3. Configurations

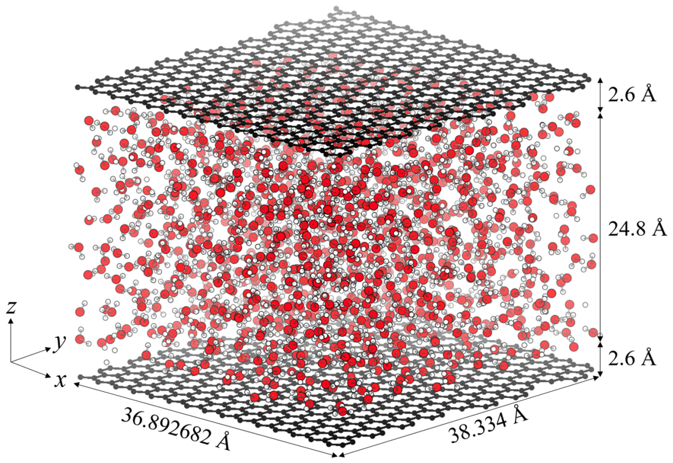
The configurations used in our simulations are tabulated in Table 1, and visualised in Figure 1. All our simulations were at ambient temperature.

Table 1: Parameters of the two systems. For the confined case, the z-axis is perpendicular to the graphene walls.

	Bulk		Confined	
Density (g/cm^3)	# atoms	Box size (Å^3)	# atoms	Box size (Å^3)
1.0	11700		4815	36.892682 x
1.2	14040	48.8678^3	5580	$\times$ 38.334000 y
1.4	16380		6333	$\times$ 30.000000 z



(a) Bulk system



(b) Confined system

Figure 1: The system configurations used in the MD simulations.

2.4. Hydrogen bond criteria

Following on from previous studies of simulated water [5, 20], a geometrical definition of the hydrogen bond between molecules was used, a definition based on the position of the first minima of the intermolecular radial distribution functions (Section 2.6), which indicate the cut-off distances for the nearest neighbours. The oxygen-oxygen cut-off distance R_{OO} is 3.5Å, and the oxygen-hydrogen cut-off distance R_{OH} is 2.4Å (see Fig.2).

These values were obtained from radial distribution functions of simulated unconstrained water at ambient conditions [21]. Thus, two molecules are considered to be bonded if their inter-oxygen distance is less than 3.5Å and their oxygen-hydrogen distance is less than 2.4Å. An angular constraint was not required because an upper bound was imposed on R_{OH}, and r_{OH} was fixed at 1Å, so the angle is already constrained by the fixed geometry. The number of hydrogen bonds per water molecule, n_{HB} was found for the bulk case (constant cross-sectional density), and for the confined case as a function of z (the code written for this analysis can be found in Appendix A) at different densities. Values for bulk water, and plots for confined water, are presented in Section 3.

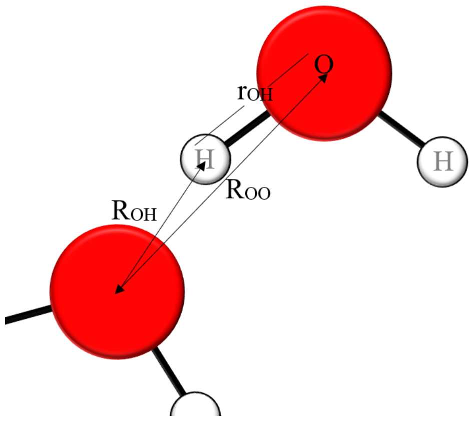


Figure 2: Labels used for analysing the hydrogen bond network of the system.

2.5. Density profile

As the density profile of bulk water is trivial, full density profiles were only computed for the confined case. For the number density profile of the confined system, an average density is made in the x and y directions parallel to the graphene sheets so that the profiles can be computed along the perpendicular z-axis. Profiles obtained at different densities are presented in Section 3 (analysis code in Appendix B).

2.6. Radial distribution function

The radial pair distribution function (PDF), $g(r)$, represents the distribution of distances between atoms in the material, normalised by the PDF of the homogenous media (where there is an equal probability density of finding a particle at any position $\mathbf{r}$). Hence, the PDF tells us how the system is structured in comparison to a perfect gas. PDFs are important for two reasons. Firstly, information about $g(r)$ can be obtained by neutron and X-ray scattering experiments on liquids, as well as light-scattering experiments on colloidal suspensions. Secondly, $g(r)$ has an important role in theories of the liquid state, and numerical results for $g(r)$ can be compared with theoretical predictions and serve as a criterion in the testing of a particular theory [9]. In our case, we computed $g(r)$ for bulk water to compare our results with those previously published, and to investigate the effect of density on the PDF. Due to layering effects, for the confined system it was necessary to compute $g(r)$ separately for each of the layers, in addition to different densities. The analysis code for the PDF calculations can be found in Appendix C.

3. Results

3.1. Bulk water

Simulating the the bulk case first allowed us to cross-check our results with the relevant literature, while providing valuable insight on the central region of the confined system which exhibits bulk-like behaviour away from the interface. The average number of hydrogen bonds per molecule at $\rho = 1.0$ g/cm^3 was $\langle n_{HB} \rangle = 3.728 \pm 0.002$. This value is comparable to previously published results of Han *et al.* (2009) [20]. Table 2 shows how the values of $\langle n_{HB} \rangle$ vary with density. The PDF data for bulk water was computed using VMD [14], as the simulation time required was too great to complete the PDFs for all three densities with the same method used for the confined case.

Table 2: Evolution of $\langle n_{HB} \rangle$ with density for bulk water.

ρ (g/cm^3)	$\langle n_{HB} \rangle$
1.0	3.728 ± 0.002
1.2	4.034 ± 0.002
1.4	4.517 ± 0.001

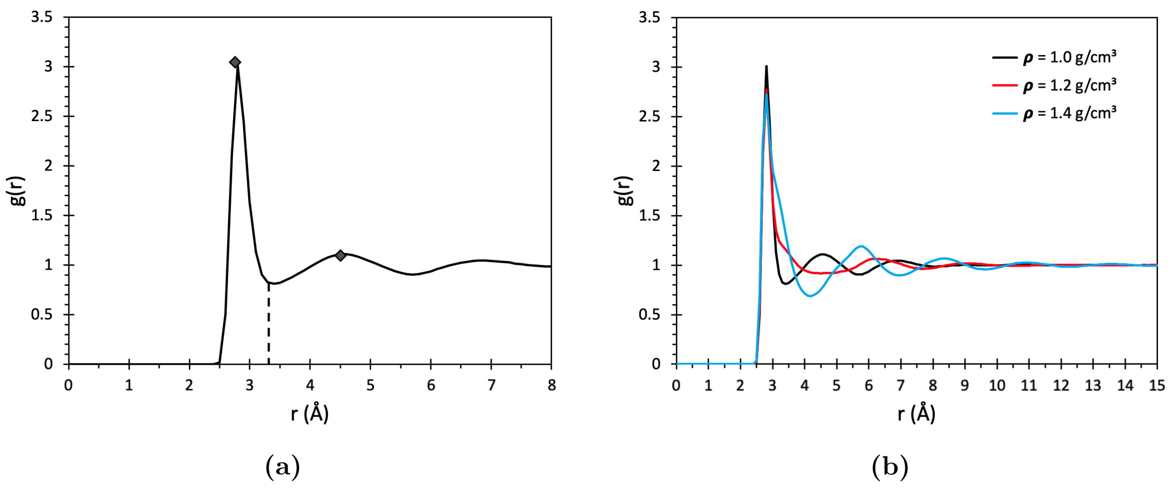


Figure 3: Oxygen-oxygen pair distribution functions for bulk water at ambient temperature: (a) PDF at $\rho = 1.0$ g/cm^3, the two diamond points and the vertical dashed line mark the first two peaks, and the position of the first minima, of previously published results [22]; (b) PDFs at three different densities.

Figure 3a shows that key features of the obtained PDF closely match the published results of Mark and Nilsson (2001) who found a first peak of 3.05 at 2.75Å, a first minimum at 3.35Å, and a second peak of 1.10 at 4.50Å [22]. Figure 3b illustrates how the pair distribution function varies with density. Increasing density reduces the height of the first peak slightly, while increasing the amplitude of subsequent peaks.

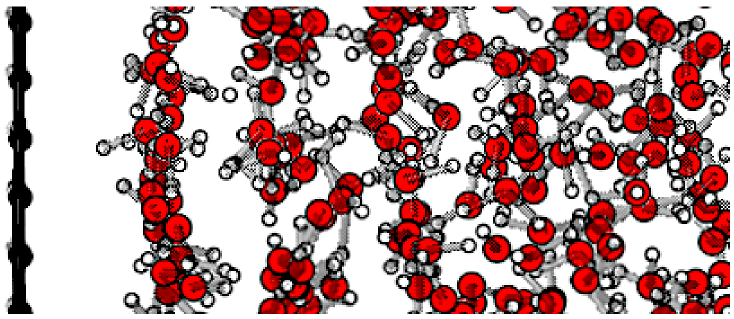


Figure 4: Visualised cross-sectional segment of the confined system at $\rho = 1.4$ g/cm^3 (compare Fig.5).

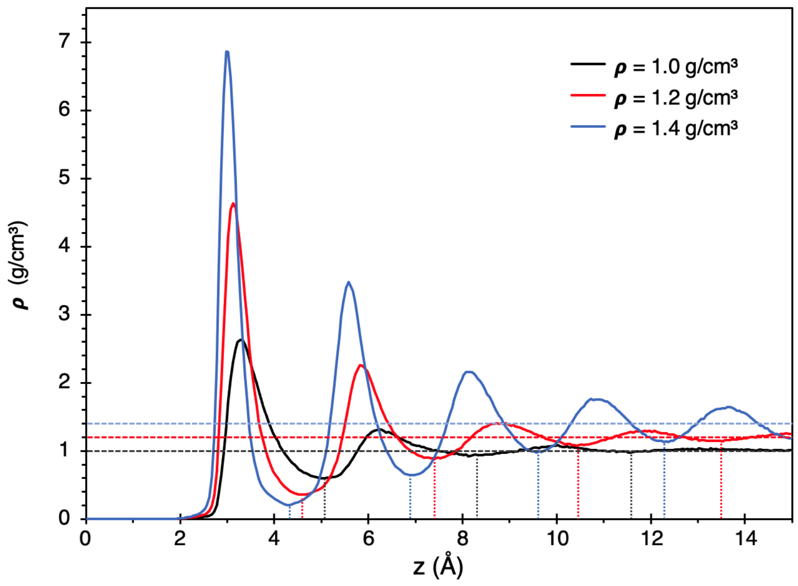


Figure 5: Mass density distribution for each density simulated. The horizontal dashed lines represent the bulk densities, and illustrate how the layered distribution of the interfacial water converges to the bulk (becomes more 'bulk-like') in the innermost layer between the grahite sheets. The vertical dashed lines mark the minima of the confined system density distributions. These were used to define the different layers.

3.2. Confined water

A visualised cross-section of the water structure is displayed in Figure 4 which shows clustering of the water molecules into layers, especially near the interface. This can be compared to the density profile (Fig.5) for $\rho = 1.4$ g/cm^3. The position of the first peak on the density profile (Fig.5) is at 3.3Å for 1.0 g/cm^3, which is comparable to published results. Mosaddeghi *et al.* (2012), using the SPCE model for a slit width of 20Å, found the first peak at z = 3.20Å for oxygen atoms and z = 3.05Å for hydrogen atoms [23]. Additionally, Marti *et al.* (2006) found a peak at approximately 3.2Å for a slit width of 32Å, though using a flexible SPC model (an extended form of SPCE) [24]. Clear changes in the structure can be seen as a function of density: as the density increases, the number of distinct layers also increases along with the amplitude of the layers. The interfacial water layer is approximately 2.6-4.0Å and is consistent with that found for similar systems in the literature including for SPCE models [4, 23] and flexible SPC models [5, 24]. The layers were defined by the progression of minima, i.e. the second layer at 1.4 g/cm^3 is defined between 4.3Å and 6.9Å. This definition for the discrete layers was needed to compute the pair distribution functions of the confined system.

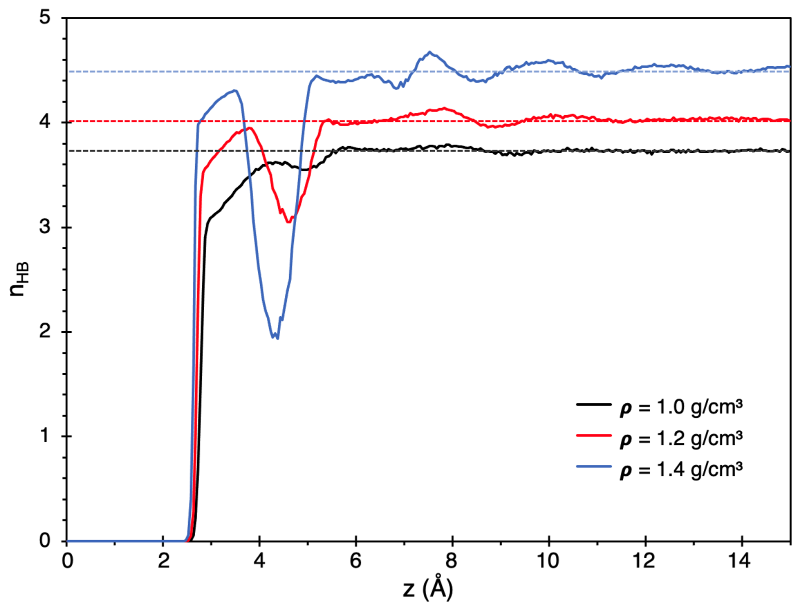


Figure 6: Hydrogen bonds per molecule as a function of z. The horizontal dashed lines represent the number of hydrogen bonds per molecule for bulk water. A notable second depletion layer can be seen around z = 4-5Å, especially at higher densities. The confined systems converge to the bulk values in the inner region of the slit.

Cicero *et al.* (2008) reported that in the interfacial layer, the number of hydrogen bonds per molecule was about 70% of the corresponding bulk value. In our simulation, n_{HB} is closer to 90% of the bulk value, though we observed a more pronounced minima (Fig.6), especially at higher densities. The aforementioned layering is also observed on this profile. Table 3 provides values of $\langle n_{HB} \rangle$ at the different densities. These are

Table 3: Evolution of $\langle \mathrm{n_{HB}} \rangle$ with density for confined water.

ρ (g/cm^3)	$\langle \mathrm{n_{HB}} \rangle$
1.0	3.732 ± 0.010
1.2	4.033 ± 0.012
1.4	4.514 ± 0.014

comparable to values for bulk water, which are within the range of uncertainty.

The radial PDFs of oxygen-oxygen atom pairs over the confined system are shown in Figure 8 for the different layers at each density, and in Figure 7 comparing the first layer for different densities. Due to the layered structure of confined water, a single PDF does not characterise the structure effectively. Hence, to characterise the structure of water in confined systems it is necessary to produce two-dimensional PDFs in the $x - y$ plane for different water layers (Fig.8). For the slit separation used in these simulations, we found that the structure of the layers is relatively homogenous; whereas for slit widths $\leq$ 10Å substantial inhomogeneity in the structure of the layers has been reported [23]. The position of the first peak in the PDFs obtained is found at ∼2.7Å, in agreement with previously published simulations [23]. This value is comparable to that of bulk structure, and thus the nearest-neighbour intermolecular distances of water remains unchanged when confined. The structure of the interfacial layer undergoes notable changes as density is increased, particularly in the range r = 3-8Å. For example, the minimum value of $g(r)$ at 1.2 g/cm^3 is higher than that of either 1.0 or 1.4 g/cm^3, while the position of the first peak is the same for all densities.

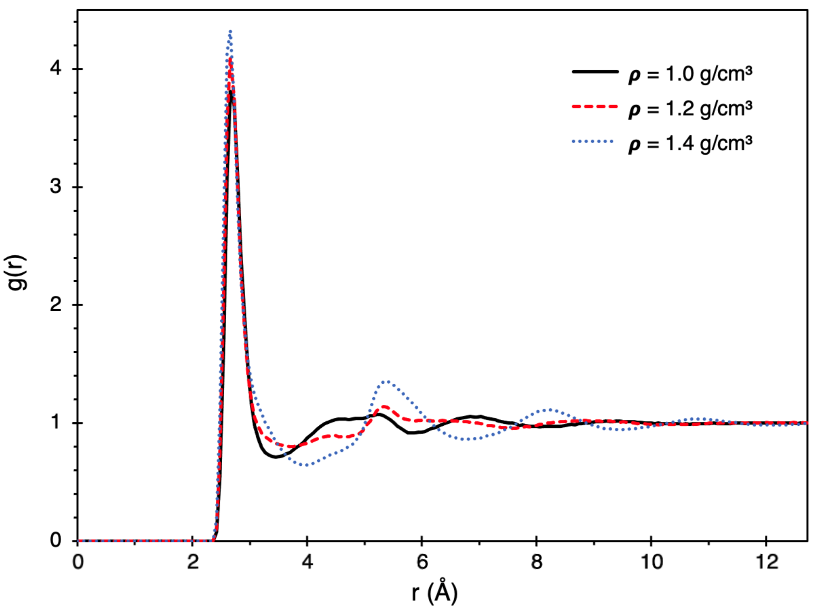


Figure 7: Pair distribution functions for the first (interfacial) layer at each density.

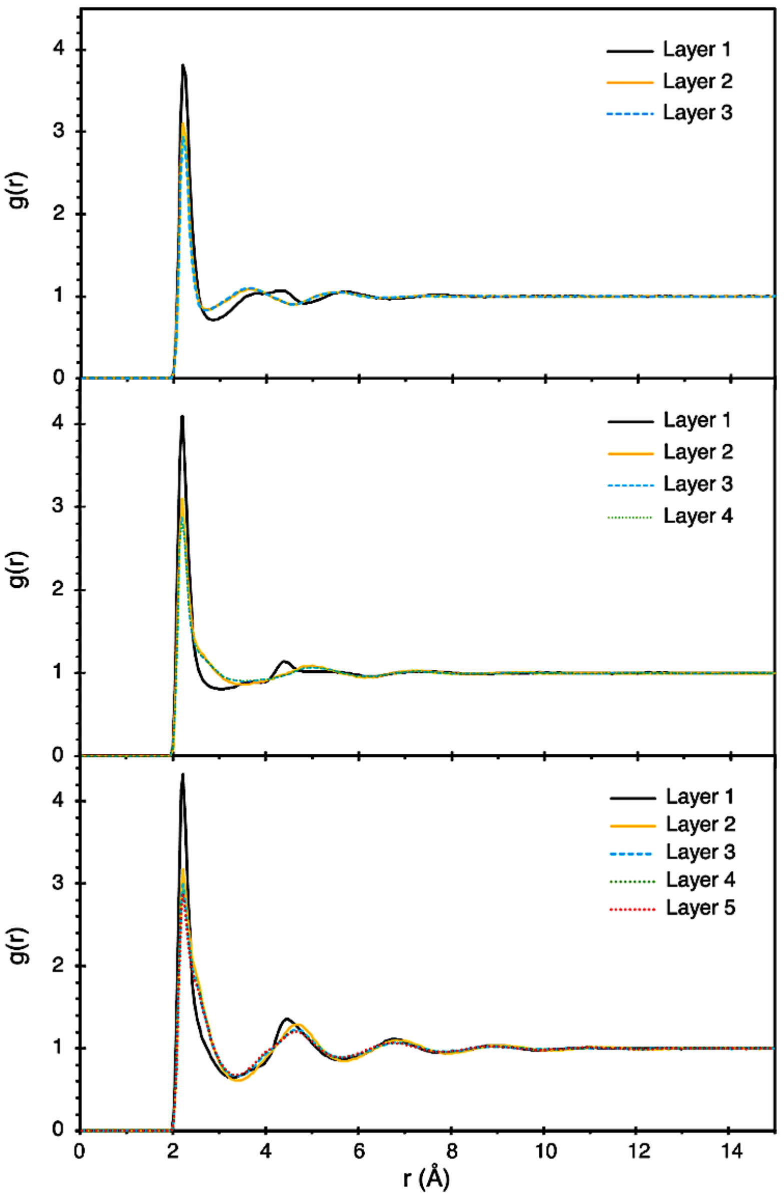


Figure 8: Pair distribution functions for different layers at each density, $\rho = 1.0, 1.2, 1.4$ g/cm^3 from top to bottom. Subsequent minima in the density profile (Fig.5) define each layer.

4. Discussion

The error on $\langle n_{HB} \rangle$ for bulk water at 1.0 g/cm^3 was calculated by taking the standard deviation of the data; the same error was assumed for $\langle n_{HB} \rangle$ in the confined case. Confined water exhibits bulk-like properties in the central region of the distribution including for density (Fig.5) and hydrogen-bonding (Fig.6). However, layering effects in the density profile are more pronounced for higher densities, with five distinct layers (in the profile bisecting the full z range between the slit walls) at 1.4 g/cm^3 in contrast to two obvious layers at 1.0 g/cm^3.

Fumagalli *et al.* (2018) detected by experiment a suppressed dielectric constant, ε, across a much thicker layer of water than expected, and observed approximately three structured water layers that are immobile [1]. Our simulations reproduce the interfacial layer demonstrated by this experiment, and support the description that the suppressed rotational freedom of the water dipoles is key to the anomalously low value of ε, as the water molecules are found to be confined to discrete layers near the interface. Furthermore, the fact that the layering effect becomes more considerable at higher densities may provide some explanation for these experimental observations.

4.1. Future directions

Our theoretical approach could be extended by introducing flexibility into the SPCE model [25, 26]. Additionally we could investigate the orientation of molecular dipole moments, and explore interfacial molecular ordering effects. Furthermore, the effect of temperature on the system could be studied, as well as properties such as the hydrogen-bond lifetime and the relaxation time.

5. Conclusion

We performed MD simulations using the SPCE model to investigate the properties of bulk water, and water confined between two graphene sheets. We showed by simulation that confined water exhibits different structural properties than bulk water, particularly layering effects, due to geometric constraints and surface interactions. The average number of hydrogen bonds per molecule was computed for both bulk and confined water, in addition to the pair distribution functions, which are both important structural parameters. We succeeded in our main objective of developing simulations that reproduce published results of key structural parameters, and extended our study by running the simulations as a function of density. This achieved compelling results that may provide an explanation for the experimental observations of Fumagalli *et al.* (2018) [1]. Next semester the confined system will be modelled using a mica (silicate) interface in place of graphene, which has not been studied previously.

References

[1] L. Fumagalli, A. Esfandiar, R. Fabregas, S. Hu, P. Ares, A. Janardanan, Q. Yang, B. Radha, T. Taniguchi, K. Watanabe, G. Gomila, K. S. Novoselov, and A. K. Geim, "Anomalously low dielectric constant of confined water," *Science*, vol. 360, pp. 1339–1342, June 2018.

[2] J. N. Israelachvili, *Intermolecular and surface forces.* Elsevier, Academic Press, third ed., 2011.

[3] O. Björneholm, M. H. Hansen, A. Hodgson, L.-M. Liu, D. T. Limmer, A. Michaelides, P. Pedevilla, J. Rossmeisl, H. Shen, G. Tocci, E. Tyrode, M.-M. Walz, J. Werner, and H. Bluhm, "Water at Interfaces," *Chemical Reviews*, vol. 116, pp. 7698–7726, May 2016.

[4] G. Cicero, J. C. Grossman, E. Schwegler, F. Gygi, and G. Galli, "Water confined in nanotubes and between graphene sheets: a first principle study," *Journal of the American Chemical Society*, vol. 130, no. 6, pp. 1871–1878, 2008.

[5] J. Martí, J. Sala, and E. Guàrdia, "Molecular dynamics simulations of water confined in graphene nanochannels: From ambient to supercritical environments," *Journal of Molecular Liquids*, vol. 153, pp. 72–78, Apr. 2010.

[6] X. Cai, W. J. Xie, Y. Yang, Z. Long, J. Zhang, Z. Qiao, L. Yang, and Y. Q. Gao, "Structure of water confined between two parallel graphene plates," *The Journal of Chemical Physics*, vol. 150, p. 124703, Mar. 2019.

[7] D. Eisenberg and W. Kauzmann, *The structure and properties of water.* Oxford: Oxford University Press, 2005.

[8] S. Chakraborty, H. Kumar, C. Dasgupta, and P. K. Maiti, "Confined water: Structure, dynamics, and thermodynamics," *Accounts of Chemical Research*, vol. 50, pp. 2139–2146, Aug. 2017.

[9] D. Frenkel and B. Smit, *Understanding molecular simulation.* No. 1 in Computational science series, San Diego, California: Acad. Press/Elsevier, second ed., 2002.

[10] J. A. Barker and R. O. Watts, "Structure of water; A Monte Carlo calculation," *Chemical Physics Letters*, vol. 3, pp. 144–145, 1969.

[11] A. Rahman and F. H. Stillinger, "Molecular dynamics study of liquid water," *The Journal of Chemical Physics*, vol. 55, no. 7, pp. 3336–3359, 1971.

[12] M. P. Allen, *Computer simulation of liquids.* Oxford: Oxford University Press, second edition ed., 2017.

[13] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," *Journal of Computational Physics*, vol. 117, pp. 1–19, Mar. 1995. http://lammps.sandia.gov.

[14] W. Humphrey, A. Dalke, and K. Schulten, "VMD – Visual Molecular Dynamics," *Journal of Molecular Graphics*, vol. 14, pp. 33–38, 1996. http://www.ks.uiuc.edu/Research/vmd/.

[15] L. Verlet, "Computer 'experiments' on classical fluids. I. thermodynamical properties of Lennard-Jones molecules," *Phys. Rev.*, vol. 159, pp. 98–103, 1967.

[16] M. Born and T. von Karmann, "Über Schwingungen in Raumgittern," *Physik. Z.*, vol. 13, pp. 297–309, 1912.

[17] H. J. C. Berendsen, J. R. Grigera, and T. P. Straatsma, "The missing term in effective pair potentials," *The Journal of Physical Chemistry*, vol. 91, no. 24, pp. 6269–6271, 1987.

[18] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, and J. Hermans, *Intermolecular Forces: Proceedings of the Fourteenth Jerusalem Symposium on Quantum Chemistry and Biochemistry Held in Jerusalem, Israel, April 13–16, 1981*, ch. Interaction Models for Water in Relation to Protein Hydration, pp. 331–342. Dordrecht: Springer Netherlands, 1981.

[19] J.-P. Ryckaert, G. Ciccotti, and H. J. Berendsen, "Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes," *Journal of Computational Physics*, vol. 23, pp. 327–341, mar 1977.

[20] S. Han, P. Kumar, and H. E. Stanley, "Hydrogen-bond dynamics of water in a quasi-two-dimensional hydrophobic nanopore slit," *Physical Review E*, vol. 79, Apr. 2009.

[21] J. Martí, "Analysis of the hydrogen bonding and vibrational spectra of supercritical model water by molecular dynamics simulations," *The Journal of Chemical Physics*, vol. 110, no. 14, pp. 6876–6886, 1999.

[22] P. Mark and L. Nilsson, "Structure and dynamics of the TIP3P, SPC, and SPC/E water models at 298 K," *The Journal of Physical Chemistry A*, vol. 105, pp. 9954–9960, Nov. 2001.

[23] H. Mosaddeghi, S. Alavi, M. H. Kowsari, and B. Najafi, "Simulations of structural and dynamic anisotropy in nano-confined water between parallel graphite plates," *The Journal of Chemical Physics*, vol. 137, p. 184703, Nov. 2012.

[24] J. Marti, G. Nagy, M. C. Gordillo, and E. Guàrdia, "Molecular simulation of liquid water confined inside graphite channels: Thermodynamics and structural properties," *The Journal of Chemical Physics*, vol. 124, p. 094703, Mar. 2006.

[25] Y. Wu, H. L. Tepper, and G. A. Voth, "Flexible simple point-charge water model with improved liquid-state properties," *The Journal of Chemical Physics*, vol. 124, p. 024503, Jan. 2006.

[26] K. Toukan and A. Rahman, "Molecular-dynamics study of atomic motions in water," *Physical Review B*, vol. 31, no. 5, pp. 2643–2648, 1985.

Listings

A. HB count code

Average number of hydrogen bonds per water molecule.

```

import matplotlib.pyplot as plt
import numpy as np
filename='(e.g.) HB_2020-11-01 20.44.01'
a=np.genfromtxt('[Working Directory]'+ filename +'.csv',delimiter=',',
   skip_header=0)
atom_type=np.array(a[:,0])
            #read in data and sort into arrays
x_pos=np.array(a[:,2])
y_pos=np.array(a[:,3])
z_pos=np.array(a[:,4])
N=4815
steps=int(len(atom_type)/N)
         #gives the number of timesteps to iterate over (change 900 if
    changing number of atoms)
#steps=10
start=0
O_start=0
end=N-1
Lx=18.446341*2
           #define box x-size
Ly=19.167000*2            #define box y-size
HB_t=[]
HB_n=[]
atm_typ=atom_type.tolist()
for item in atom_type:
    HB_n.append(0)

for t in range(0,steps):
            #for each timstep (t)
    pairs_OO=[]
    pairs_OH=[]
    r_OO=[]
    r2_OO=[]
    r_OH=[]
    for i in range(start,end):
        if atom_type[i]==2:
            #if it is an oxygen atom
         x_i=x_pos[i]
         #declare it's position
         y_i=y_pos[i]
         z_i=z_pos[i]
         for j in range(start,end):
         #for every atom
             if atom_type[j]==2:
         #that is also oxygen
                 if i!=j:
         #other than atom_i
                     x_j=x_pos[j]
         #note position of atom_j
                     y_j=y_pos[j]
                     z_j=z_pos[j]
                     if x_i-x_j>Lx/2:
                         x_j=x_j+Lx
                     if x_i-x_j<-Lx/2:
         #if atom_j is further than L/2 away ina given
                         x_j=x_j-Lx
         #co-ordinate then shift to the nearer atom by
                     if y_i-y_j>Ly/2:
         #moving over to the opposite 'box' by moving -L
                         y_j=y_j+Ly
                     if y_i-y_j<-Ly/2:
                         y_j=y_j-Ly
                     r_ij2=(x_i-x_j)**2+(y_i-y_j)**2+(z_i-z_j)**2
         #calc rij^2
                     if r_ij2<3.5**2:
         #if is < R_oo
                         pairs_OO.append(str(i)+'_'+str(j))
         #add the pair to the list of candiate H-bond pairs
                         r_OO.append(str(x_i-x_j)+'_'+str(y_i-y_j)+'
_'+str(z_i-z_j))                  #not for function, just to check
the r values are in the right range.

 for item in pairs_OO:
     div=item.find('_')
     x_i=x_pos[int(item[0:div])]
         #declare atom i's position
     y_i=y_pos[int(item[0:div])]
     z_i=z_pos[int(item[0:div])]
     for h in range(1,3):
         x_j=x_pos[int(item[div+1:])+h]
         #declare atom j's position
         y_j=y_pos[int(item[div+1:])+h]
         z_j=z_pos[int(item[div+1:])+h]
         if x_i-x_j>Lx/2:
             x_j=x_j+Lx
         if x_i-x_j<-Lx/2:
         #if atom_j is further than L/2 away ina given
             x_j=x_j-Lx
         #co-ordinate then shift to the nearer atom by
         if y_i-y_j>Ly/2:
         #moving over to the opposite 'box' by moving -L
             y_j=y_j+Ly
         if y_i-y_j<-Ly/2:
             y_j=y_j-Ly
         r_ij2=(x_i-x_j)**2+(y_i-y_j)**2+(z_i-z_j)**2
         #calc rij^2 if is < R_oh
         if r_ij2<2.4**2:
             pair_id=str(int(item[0:div]))+'_'+str(int(item[div+1:])
+h)+'_'+str(h)     #produce a pair id
             pairs_OH.append(pair_id)
             r_OH.append(str(x_i-x_j)+'_'+str(y_i-y_j)+'_'+str(z_i-
z_j))
             r2_OO.append(r_OO[pairs_OO.index(item)])

 pairs_phi=[]
 phi_=[]
 for p, item in enumerate(r2_OO):
     divi1=item.find('_')
     divi2=item.rfind('_')
     x1=float(item[:divi1])
     y1=float(item[divi1+1:divi2])
     z1=float(item[divi2+1:])
     ids=pairs_OH[p]
     div1=ids.find('_')
     div2=ids.rfind('_')
     atom1=int(ids[:div1])
     h_=int(ids[div2+1:])
     atom2=int(ids[div1+1:div2])-h_
     v_OO=[x1,y1,z1]
     item2=r_OH[p]
     divj1=item2.find('_')
     divj2=item2.rfind('_')
     x2=float(item2[:divj1])
     y2=float(item2[divj1+1:divj2])
     z2=float(item2[divj2+1:])
     v_OH=[x2,y2,z2]
     unit_vector_1 = v_OO/(np.linalg.norm(v_OO))
     unit_vector_2 = v_OH /(np.linalg.norm(v_OH))
     dot_product = np.dot(unit_vector_1, unit_vector_2)
     phi = np.arccos(np.around(dot_product,7))
     if phi<np.pi/6:
        # check if angle less than 30 degrees
         phi_.append(phi*180/np.pi)
         pairs_phi.append(pairs_OH[p])
         HB_n[atom1]=HB_n[atom1]+1
         HB_n[atom2]=HB_n[atom2]+1

 HB_count=len(pairs_phi)
 HB_t.append(HB_count*2/82)
 start=start+N                      #shift to the nesxt timetstep
 end=end+N
```

```python
    print(t)

HB_avg=(sum(HB_t)/len(HB_t))
print(HB_avg)


start=0
end=N-1
C=61
Lx=6.14878035*2                                         #define box size
Ly=6.38999985*2
Lz=15.67500019
V=Lx*Ly*(1/C)*2*15.67500019
HB_dens=[]
z_axis=[]
for t in range(0,steps):
    z_list=[]
    HB_list=[]
    for n in range(0,C-1):
        z1=-Lz+(n/C)*(2*Lz)
        z2=-Lz+((n+1)/C)*(2*Lz)
        z_list.append(0)
        HB_list.append(0)
        if t==0:
            HB_dens.append(0)
            z_axis.append((z1+z2)/2)
        for i in range(start,end):
            if atom_type[i]==2:
                if z1<z_pos[i]<z2:
                    z_list[n]=z_list[n]+1
                    HB_list[n]=HB_list[n]+int(HB_n[i])
    for i in range(0,C-1):
        if z_list[i]!=0:
            HB_dens[i]=HB_dens[i]+(HB_list[i]/z_list[i])
    start=start+N
    end=end+N
    print(t)
HB_avg=[]
for i in range(0,C-1):
    HB_avg.append(HB_dens[i]/steps)

plt.plot(z_axis,HB_avg,)  #color='k'
plt.grid()
plt.minorticks_on()
plt.ylabel('HB per molecule')
plt.xlabel('z (   )')
plt.show()
```

Listing 1: Calculating the number of hydrogen bonds per molecule.

B. Density profile code

```python

 import numpy as np                                                    #paste data filename  \/
a=np.genfromtxt('HB_extract.csv',delimiter=',',skip_header=0,usecols=(0,4))
atom_type=np.array(a[:,0])
z_pos=np.array(a[:,1])
N=4815                                #Number of atoms
steps=int(len(atom_type)/N)                  #gives the number of timesteps to iterate over (change 900 if changing number of atoms)                    #UNNCOMMENT THIS LINE AND COMMENT LINE ABOVE TO ONLY DO FIRST TIMESTEP - USEFUL FOR TESTING CHANGES
steps=100
start=0                                      #set these indices so each loop only goes through one timestep's data at a time
end=N-1                                        #change 900 here too
Lx=18.446341*2                                           #define box size
Ly=19.167000*2
Lz=45.825000
t=0
C=300                     #do with like 400/500 C
h=15
V=Lx*Ly*(1/C)*h
z_axis=[]
z_list=[]
for t in range(0,steps):
    for n in range(0,C):
        z1=(n/C)*(h)
        z2=((n+1)/C)*(h)
        if t==0:
            z_list.append(0)
            z_axis.append((z1+z2)/2)
        for i in range(start,end):
            if atom_type[i]==2:
                if z1<abs(z_pos[i])<z2:
                    z_list[n]=z_list[n]+15.9994*1.661
            if atom_type[i]==3:
                if z1<abs(z_pos[i])<z2:
                    z_list[n]=z_list[n]+1.008*1.661
    start=start+N
    end=end+N
    print(t)
z_avg=[]
for item in z_list:
    z_avg.append(item/(steps*V))                     #*26.56

import matplotlib.pyplot as plt
plt.plot(z_axis,z_avg,color='k')
plt.grid()
plt.minorticks_on()
plt.ylabel('   (g cm^-3)')
plt.xlabel('z (   )')
plt.show()
```

Listing 2: Density profile.

C. Radial pair distribution function code

```python
import scipy.interpolate
import matplotlib.pyplot as plt
import numpy as np
filename='HB_2020-10-29 12.48.58'
a=np.genfromtxt('C:\\Users\\izaak\\Documents\\Project\\Programming\\Python\\'+ filename +'.csv',delimiter=',',skip_header=0)
atom_type=np.array(a[:,0])                      #read in data and sort into arrays
atom_id=np.array(a[:,1])
x_pos=np.array(a[:,2])
y_pos=np.array(a[:,3])
z_pos=np.array(a[:,4])
steps=int(len(atom_id)/900)                 #gives the number of timesteps to iterate over (change 900 if changing number of atoms)
#steps=5 #UNNCOMMENT THIS LINE AND COMMENT LINE ABOVE TO ONLY DO FIRST TIMESTEP - USEFUL FOR TESTING CHANGES
start=0         #set these indices so each loop only goes through one timestep's data at a time
end=899                                     #change 900 here too
L=20.784904                                        #define box size
t=0
r_OO=[]
r_2=[]
for t in range(0,steps):                           #for each timstep (t)
   for i in range(start,end):
       if atom_type[i]==1:                    #if it is an oxygen atom
            x_i=x_pos[i]                       #declare it's position
            y_i=y_pos[i]
            z_i=z_pos[i]
            for j in range(start,end):                           #for every atom
                if atom_type[j]==1:            #that is also oxygen
                    if i!=j:                        #other than atom_i
                        x_j=x_pos[j]           #note position of atom_j
                        y_j=y_pos[j]
                        z_j=z_pos[j]
                        if x_i-x_j>L/2:             #if atom_j is further than L/2 away ina given
                            x_j=x_j+(L)
                        if x_i-x_j<-L/2:             #if atom_j is further than L/2 away ina given
                            x_j=x_j-L               #co-ordinate then shift to the nearer atom by
                        if y_i-y_j>L/2:         #moving over to the opposite 'box' by moving -L
                            y_j=y_j+(L)
                        if y_i-y_j<-L/2:         #moving over to the opposite 'box' by moving -L
                            y_j=y_j+(-L)
                        if z_i-z_j>L/2:
                            z_j=z_j+(L)
                        if z_i-z_j<-L/2:
                            z_j=z_j+(-L)
                        r_ij2=(x_i-x_j)**2+(y_i-y_j)**2+(z_i-z_j)**2    #calc rij^2
                        if r_ij2<(L/2)**2:                              # if is < R_oo
                            r_OO.append(np.sqrt(r_ij2))  #not for function, just to check the r values are in the right range.
   start=start+900                                             #shift to the nesxt timetstep
   end=end+900
   print(t)
C=101
r_axis=[]
r_list=[]
for n in range(0,C-1):
    r1=(n/C)*(L/2)
    r2=((n+1)/C)*(L/2)
    r_list.append(0)
    r_axis.append((r1+r2)/2)
    for item in r_OO:
        if r1<item<r2:
            r_list[n]=r_list[n]+1
r_avg=[]
for item in r_list:
    r_avg.append(float(item/(steps*300)))               #*26.56

r_final=[]
n_exp=0.033447053
r_exp=[]
for i in range(0,len(r_avg)):
        r1=(i/C)*(L/2)
        r2=((i+1)/C)*(L/2)
        V=((4/3)*(np.pi)*((r2**3)-(r1**3)))
        r_final.append(r_avg[i]/(V*n_exp))

plt.plot(r_axis, r_final,label='Normalised')
plt.grid()
plt.minorticks_on()
plt.legend()
x1,x2,y1,y2 = plt.axis()
plt.hlines(1,0,10,color='grey')
plt.ylabel('g(r)')
plt.xlabel('r (  )')
plt.show()
```

Listing 3: PDF analysis.

D. LAMMPS code

```
# ================i====================================================
# Define variables
# ====================================================================

# input data file
variable        lmp_data string "data.lmp"
               # configuration file name
variable        lmp_restart string "restart.RELAX01"
               # restart file name

# temperature and pressure
variable        T equal 298
             # temperature (in K)
variable        p equal 3/0.101325
             # pressure

# interactions / rcut
variable        rcut equal 8.5
             # cutoff for pair interactions (in A)

# number of steps / timestep
variable        nstep equal "100000"
             # number of steps
variable        dt equal 2.0
             # timestep (in fs)

# output
variable        freq_thermo equal "500"
             # frequency of the thermodynamic quantities output
variable        freq_frame equal "500"
             # frequency of the configurations output

# ====================================================================
# Initialization
# ====================================================================

units           real
             # system of units
boundary        p p p
             # periodic boundary conditions in every direction
dimension       3
             # dimension of the system
atom_style      full

```

```
# ====================================================================
# Force Field
# ====================================================================

read_data       ${lmp_data}
             # read configuration file

# 1 OW
# 2 HW

mass            1 15.9994
             # mass of atom type 1 (in atomic mass unit)
mass            2 1.008
             # mass of atom type 2 (in atomic mass unit)

pair_style lj/cut/coul/long ${rcut} ${rcut}
             # type of pair interactions
pair_modify     mix arithmetic

pair_coeff     * *  0.0          0.0
             # set the parameters of all the interactions to 0
pair_coeff     1 1  0.15529876    3.166


# 1 OW-HW
             # intramolecular O-H bond
bond_style      harmonic
bond_coeff      1 1000.00 1.000

# 5 HW-OW-HW (fix)
             # intramolecular H-O-H angle
angle_style     harmonic
angle_coeff     1 100.0 109.47

# Coulomb interaction
kspace_style     pppm 1.0e-5
             # parameter for Ewald summation (need convergence test)

group            o type 1
group            water type 1 2

# ====================================================================
# Constraints
# ====================================================================

fix fixspce water shake 0.0001 20 0 b 1 a 1
             # algorithm constraint for water molecules

# ====================================================================
# 1. RELAX / NVE limit
# ====================================================================

#velocity         water create $T 87654 dist gaussian mom yes rot yes
               # generate initial velocities (gaussian distribution)


fix              integrator water nve
              # simulation in NVE ensemble
#fix              thermostat water temp/rescale 10 298 298 5.0 1.0
                # rescaling of the velocities (for NVE)
#fix              thermostat water nvt temp ${T} ${T} $(100.0*dt) tchain
    10         # simulation with a thermostat (for NVT)
#fix              barostat all press/berendsen aniso ${p} ${p} 1000.0
               # simulation with a barostat (for NPT)

thermo           ${freq_thermo}
thermo_style     custom step temp press etotal pe ke epair ecoul evdwl
thermo_modify    flush yes

dump             dtrj all custom ${freq_frame} ${lmp_data}.lammpstrj
   type id x y z vx vy vz
dump_modify      dtrj sort id
dump             dxyz all xyz ${freq_frame} ${lmp_data}.xyz
               # prints atom type (number) and position
dump_modify      dxyz sort id

timestep         ${dt}

run ${nstep}

undump           dtrj
undump           dxyz

write_restart    restart.RELAX01
               # output restart file
print            "01.RELAX done"
```

Listing 4: in_eq.lmp

```
created by fftool

900 atoms
600 bonds
300 angles
2 atom types
1 bond types
1 angle types
0.000000 20.784904 xlo xhi
0.000000 20.784904 ylo yhi
0.000000 20.784904 zlo zhi

Masses

   1   15.999  # Ow
   2    1.008  # Hw
```

```

Atoms

 1        1    1   -0.847600   1.793210e+01   7.639830e+00   1.509490e+01   # Ow  SPCE
 2        1    2    0.423800   1.774540e+01   6.680480e+00   1.530660e+01   # Hw  SPCE
 3        1    2    0.423800   1.775600e+01   8.200100e+00   1.590430e+01   # Hw  SPCE
 4        2    1   -0.847600   5.043220e+00   7.306070e+00   3.656350e+00   # Ow  SPCE
 5        2    2    0.423800   4.233060e+00   7.547960e+00   4.190320e+00   # Hw  SPCE
 6        2    2    0.423800   5.495220e+00   6.518290e+00   4.074800e+00   # Hw  SPCE
 7        3    1   -0.847600   1.987180e+01   1.308370e+01   1.968160e+01   # Ow  SPCE
 8        3    2    0.423800   2.014700e+01   1.254530e+01   1.888510e+01   # Hw  SPCE
 9        3    2    0.423800   1.890080e+01   1.331200e+01   1.961030e+01   # Hw  SPCE
10        4    1   -0.847600   1.193130e+01   9.382890e+00   1.601920e+01   # Ow  SPCE
11        4    2    0.423800   1.221440e+01   1.024100e+01   1.559080e+01   # Hw  SPCE
12        4    2    0.423800   1.257020e+01   8.656790e+00   1.576490e+01   # Hw  SPCE
13        5    1   -0.847600   1.822490e+00   3.100430e+00   7.003440e+00   # Ow  SPCE

```
Listing 5: data.lmp