



**HAL**  
open science

# Analyzing Melts and Fluids from Ab Initio Molecular Dynamics Simulations with the UMD Package

Razvan Caracas, Anais Kobsch, Natalia V Solomatova, Zhi Li, Francois Soubiran, Jean-Alexis Hernandez

## ► To cite this version:

Razvan Caracas, Anais Kobsch, Natalia V Solomatova, Zhi Li, Francois Soubiran, et al.. Analyzing Melts and Fluids from Ab Initio Molecular Dynamics Simulations with the UMD Package. Journal of visualized experiments: JoVE, 2021, 10.3791/61534 . hal-03408634

**HAL Id: hal-03408634**

**<https://hal.science/hal-03408634>**

Submitted on 29 Oct 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Analyzing Melts and Fluids from Ab Initio Molecular Dynamics Simulations with the UMD Package

Razvan Caracas<sup>1,2</sup>, Anais Kobsch<sup>1</sup>, Natalia V. Solomatova<sup>1</sup>, Zhi Li<sup>1</sup>, Francois Soubiran<sup>1,3</sup>, Jean-Alexis Hernandez<sup>1,2</sup>

<sup>1</sup> Ecole Normale Supérieure de Lyon, Laboratory of Geology of Lyon UMR5276, CNRS <sup>2</sup> Centre for Earth Evolution and Dynamics (CEED), University of Oslo <sup>3</sup> CEA, DAM, DIF

## Corresponding Author

Razvan Caracas  
razvan.caracas@gmail.com

## Citation

Caracas, R., Kobsch, A., Solomatova, N.V., Li, Z., Soubiran, F., Hernandez, J.A. Analyzing Melts and Fluids from Ab Initio Molecular Dynamics Simulations with the UMD Package. *J. Vis. Exp.* (175), e61534, doi:10.3791/61534 (2021).

## Date Published

September 17, 2021

## DOI

10.3791/61534

## URL

joVE.com/video/61534

## Abstract

We have developed a Python-based open-source package to analyze the results stemming from ab initio molecular-dynamics simulations of fluids. The package is best suited for applications on natural systems, like silicate and oxide melts, water-based fluids, and various supercritical fluids. The package is a collection of Python scripts that include two major libraries dealing with file formats and with crystallography. All the scripts are run at the command line. We propose a simplified format to store the atomic trajectories and relevant thermodynamic information of the simulations, which is saved in UMD files, standing for Universal Molecular Dynamics. The UMD package allows the computation of a series of structural, transport and thermodynamic properties. Starting with the pair-distribution function it defines bond lengths, builds an interatomic connectivity matrix, and eventually determines the chemical speciation. Determining the lifetime of the chemical species allows running a full statistical analysis. Then dedicated scripts compute the mean-square displacements for the atoms as well as for the chemical species. The implemented self-correlation analysis of the atomic velocities yields the diffusion coefficients and the vibrational spectrum. The same analysis applied on the stresses yields the viscosity. The package is available via the GitHub website and via its own dedicated page of the ERC IMPACT project as open-access package.

## Introduction

Fluids and melts are active chemical and physical transport vectors in natural environments. The elevated rates of atomic diffusion favor chemical exchanges and reactions, the low viscosity coupled with varying buoyancy favor large mass transfer, and crystal-melt density relations favor

layering inside planetary bodies. The absence of a periodic lattice, typical high-temperatures required to reach the molten state, and the difficulty for quenching make the experimental determination of a series of obvious properties, like density, diffusion, and viscosity, extremely challenging.

These difficulties make alternative computational methods strong and useful tools for investigating this class of materials.

With the advent of computing power and the availability of supercomputers, two major numerical atomistic simulation techniques are currently employed to study the dynamical state of a non-crystalline atomistic system, Monte Carlo<sup>1</sup> and molecular dynamics (MD)<sup>1,2</sup>. In Monte Carlo simulations the configurational space is randomly sampled; Monte Carlo methods show linear scaling in parallelization if all sampling observations are independent of each other. The quality of the results depends on the quality of the random number generator and the representativeness of the sampling. Monte Carlo methods show linear scaling in parallelization if the sampling is independent of each other. In molecular dynamics (MD) the configurational space is sampled by time-dependent atomic trajectories. Starting from a given configuration, the atomic trajectories are computed by integrating the Newtonian equations of motion. The interatomic forces can be computed using model interatomic potentials (in classical MD) or using first-principles methods (in ab initio, or first-principles, MD). The quality of the results depends on the length of the trajectory and its capability to not be attracted to local minima.

Molecular dynamics simulations contain a plethora of information, all related to the dynamical behavior of the system. Thermodynamic average properties, like internal energy, temperature, and pressure, are rather standard to compute. They can be extracted from the output file(s) of the simulations and averaged, whereas quantities related directly to the movement of the atoms as well as to their mutual relation need to be computed after extraction of the atomic positions and velocities.

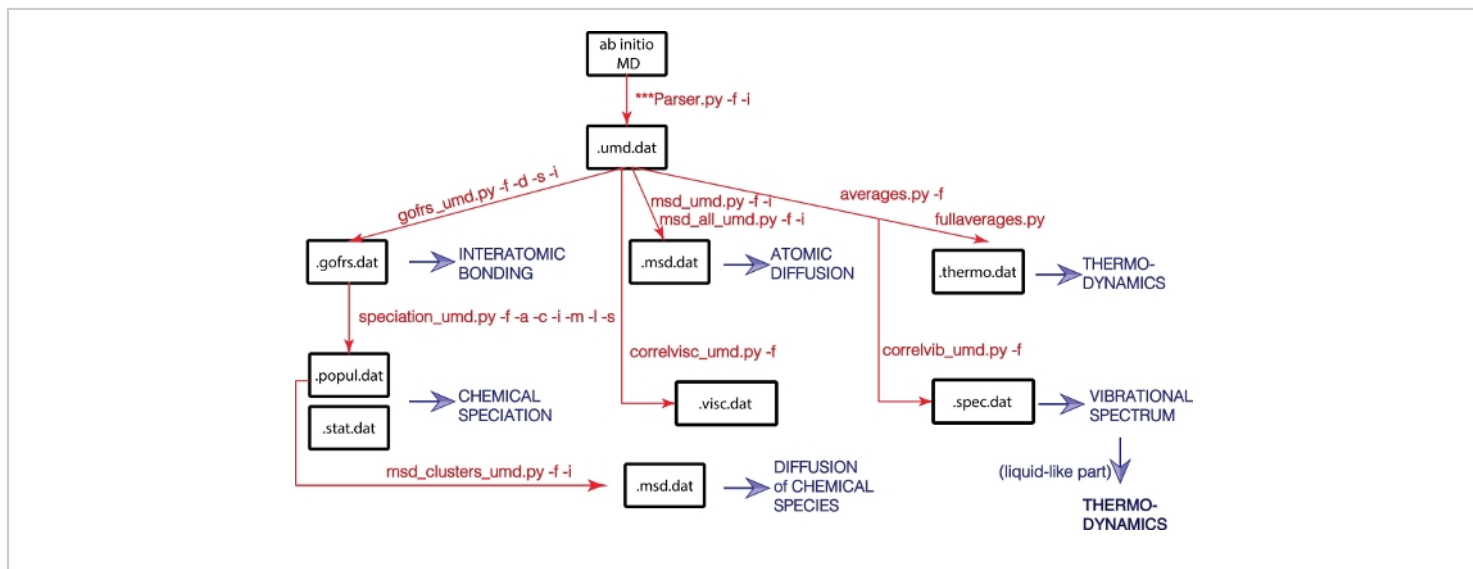
Consequently, a lot of effort has been dedicated to visualizing the results, and various packages are available today, on different platforms, open source or not [Ovito<sup>3</sup>, VMD<sup>4</sup>, Vesta<sup>5</sup>, Travis<sup>6</sup>, etc.]. All these visualization tools deal efficiently with interatomic distances, and as such, they allow the efficient computation of pair distribution functions and diffusion coefficients. Various groups performing large-scale molecular dynamics simulations have proprietary software to analyze various other properties resulting from the simulations, sometimes in shareware or other forms of limited access to the community, and sometimes limited in scope and use to some specific packages. Sophisticated algorithms to extract information about interatomic bonding, geometrical patterns, and thermodynamics are developed and implemented in some of these packages<sup>3,4,5,6,7</sup>, etc.

Here we propose **the UMD package** - an open-source package written in Python to analyze the output of molecular dynamics simulations. The UMD package allows for the computation of a wide range of structural, dynamical, and thermodynamical properties (**Figure 1**). The package is available via the GitHub website ([https://github.com/rcaracas/UMD\\_package](https://github.com/rcaracas/UMD_package)) and via a dedicated page (<http://moonimpact.eu/umd-package/>) of the ERC IMPACT project as an open-access package.

To make it universal and easier to handle, our approach is to first extract all the information related to the thermodynamic state and the atomic trajectories from the output file of the actual molecular-dynamics run. This information is stored in a dedicated file, whose format is independent of the original MD package where the simulation was run. We name these files “umd” files, which stands for Universal Molecular Dynamics. In this way, our UMD package can be easily used by any ab initio group with any software, all with a minimum effort of

adaptation. The only requirement to use the present package is to write the appropriate parser from the output of the particular MD software into the umd file format, if this is not

yet existent. For the time being, we provide such parsers for the VASP<sup>8</sup> and the QBox<sup>9</sup> packages.



**Figure 1: Flowchart of the UMD library.**

Physical properties are in blue, and major Python scripts and their options are in red. [Please click here to view a larger version of this figure.](#)

The umd files are ASCII files; typical extension is “umd.dat” but not mandatory. All the analysis components can read ASCII files of the umd format, regardless of the actual name extension. However, some of the automatic scripts designed to perform fast large-scale statistics over several simulations specifically look for files with the umd.dat extension. Each physical property is expressed on one line. Every line starts

with a keyword. In this way the format is highly adaptable and allows for new properties to be added to the umd file, all the while preserving its readability throughout versions. The first 30 lines of the umd file of the simulation of pyrolite at 4.6 GPa and 3000 K, used below in the discussion, are shown in **Figure 2**.



calculations, like specific volume, density, obtaining the reciprocal lattice from the direct one, etc. The “Atoms” class defines the atoms. They are characterized by a series of scalar properties (name, symbol, mass, number of electrons, etc.) and a series of vectorial properties (the position in space, either relative to the vectorial basis described in the Lattice class, or relative to universal Cartesian coordinates, velocities, forces, etc.). Apart from these two classes, the *crystallography.py* library contains a series of functions to perform a variety of tests and calculations, such as atomic distances, or cell multiplication. The periodic table of the elements is also included as a dictionary.

The various components of the umd package write several output files. As a general rule, they are all ASCII files, all their entries are separated by tabs, and they are made as self-explanatory as possible. For example, they always clearly

indicate the physical property and its units. The *umd.dat* files fully comply with this rule.

## Protocol

### 1. Analysis of the molecular-dynamics runs

**NOTE:** The package is available via the GitHub website ([https://github.com/rcaracas/UMD\\_package](https://github.com/rcaracas/UMD_package)) and via a dedicated page (<http://moonimpact.eu/umd-package/>) of the ERC IMPACT project as an open-access package.

1. Extract each specific set of physical properties using one or more dedicated Python scripts from the package. Run all the scripts at the command line; they all employ a series of flags, which are as consistent as possible from one script to another. The flags, their meaning, and default values are all summarized in **Table 1**.

Flag	Meaning	Script using it	Default value
-h	Short help	all	
-f	UMD file name	all	
-i	Thermalization steps to be discarded	all	0
-i	Input file containing the interatomic bonds	speciation	bonds.input
-s	Sampling of the frequency	msd, speciation	1 (every step is considered)
-a	List of atoms or anions	speciation	
-c	List of cations	speciation	
-l	Bond length	speciation	2
-t	Temperature	vibrations, rheology	
-v	Discretization of the width of the sampling window of	msd	20

	the trajectory for the mean-square displacement analysis		
-z	Discretization of the start of the sampling window of the trajectory for the mean-square displacement analysis	msd	20

**Table 1: Most common flags used in the UMD package and their most common significance.**

2. Start with transforming the output of the MD simulation performed in a first-principles code, like VASP<sup>8</sup> or QBox<sup>9</sup>, into a UMD file.

1. If the MD simulations were done in VASP, then at the command line type:

```
VaspParser.py -f <OUTCAR_filename> -i <InitialStep>
```

where `-f` flag defines the name of the VASP OUTCAR file, and the `-i` the thermalization length.

**NOTE:** The initial step, defined by `-i` allows for discarding the first steps of the simulations, which represent the thermalization. In a typical molecular-dynamics run, the first part to the calculation represents the thermalization, i.e., the time it takes the system for all atoms to describe a Gaussian-like distribution of the temperature, and for the entire system to exhibit fluctuations of the temperature, pressure, energy, etc. around equilibrium values. This thermalization part of the simulation should not be taken into account when analyzing the statistical properties of the fluid.

3. Transform the `.umd` files into `.xyz` files to facilitate visualization on various other packages, like VMD<sup>4</sup> or Vesta<sup>5</sup>. At the command line type:

```
umd2xyz.py -f <umdfile> -i <InitialStep> -s <Sampling_Frequency>
```

where `-f` defines the name of the `.umd` file, `-i` defines the thermalization period to be discarded, and `-s` the frequency of the sampling of the trajectory stored in the `.umd` file. Default values are `-i 0 -s 1`, i.e. considering all the steps of the simulation, without any being discarded.

4. Reverse the `umd` file into VASP-type POSCAR files using the `umd2poscar.py` script; snapshots of the simulations can be selected with a predefined frequency. At the command line type:

```
umd2poscar.py -f <umdfile> -i <InitialStep> -l <LastStep> -s <Sampling_Frequency>
```

where `-l` represents the last step to be transformed into POSCAR file. Default values are `-i 0 -l 10000000 -s 1`. This value of `-l` is big enough to cover a typical entire trajectory.

## 2. Perform the structural analysis

1. Run the `gofrs_umd.py` script to compute the pair distribution function (PDF)  $g_{AB}(r)$  for all the pairs of atomic types A and B (**Figure 3**). The output is written in one

ASCII file, tab-separated, with the extension *gofrs.dat*. At the command line type:

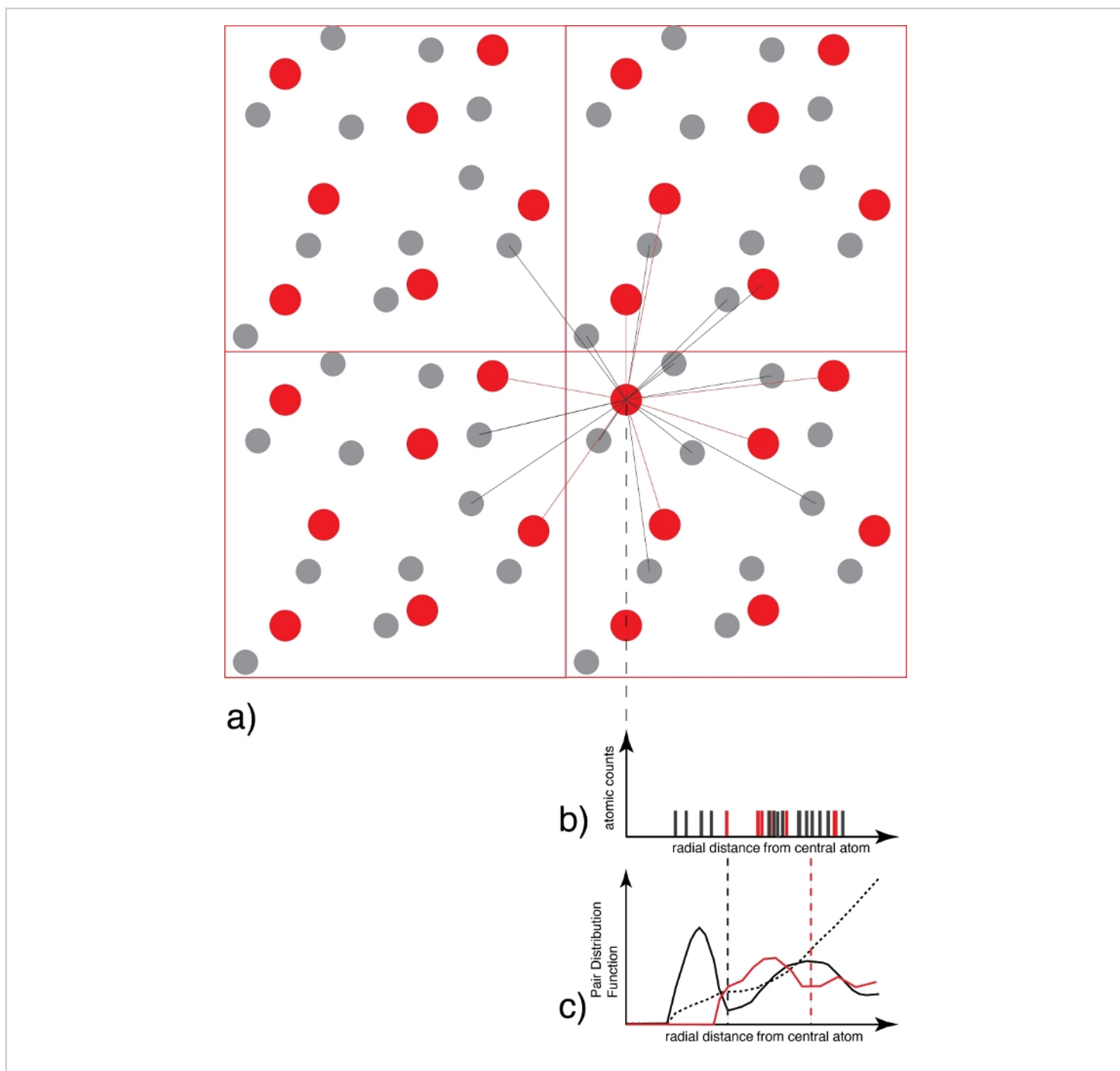
```
gofrs_umd.py -f <UMD_filename> -s <
Sampling_Frequency > -d <DiscretizationInterval> -i
<InitialStep>
```

**NOTE:** The defaults are Sampling\_Frequency (the frequency for sampling the trajectory) = 1 step; DiscretizationInterval (for plotting the  $g(r)$ ) = 0.01 Å; InitialStep (number of steps in the beginning of the trajectory that are discarded) = 0. The radial PDF,  $g_{AB}(r)$  is the average number of atoms of type B at a distance  $d_{AB}$  within a spherical shell of radius  $r$  and thickness  $dr$  centered on the atoms of type A (**Figure 3**):

$$g_{AB}(r) = \frac{N_A - 1}{4\pi\rho^2} \langle \delta(r - r_{AB}) \rangle_B$$

with  $\rho$  the atomic density,  $N_A$  and  $N_B$  the number of atoms of type A and B, and  $\delta(r-r_{AB})$  the delta function which is equal to 1 if the atoms A and B lie at a distance between  $r$  and  $r+dr$ . The abscissa of the first maximum of  $g_{AB}(r)$  gives the highest probability bond length between the atoms of type A and B, which is the closest to an average bond distance that we can determine. The first minimum delimits the extent of the first coordination sphere. Hence the integral over the PDF up to the first minimum gives the average coordination number. The sum of the Fourier transforms of the  $g_{AB}(r)$  for all the pairs of atomic types A and B yields the diffraction pattern of the fluid, as obtained experimentally with a diffractometer. However, in reality, as oftentimes the high order coordination spheres are missing from the  $g_{AB}(r)$ , the diffraction pattern cannot be obtained in its entirety.





**Figure 3: Determination of pair distribution function.**

(a) For each atom of one species (for example red), all the atoms of the coordinating species (for example grey and/or red) are counted as a function of distance. (b) The resulting distance distribution graph for each snapshot, which at this stage is only a collection of delta functions, is then averaged over all the atoms and all the snapshots and weighted by the ideal gas

distribution to generate (c) the pair distribution function that is continuous. The first minimum of the  $g(r)$  is the radius of the first coordination sphere, used later on in the speciation analysis. [Please click here to view a larger version of this figure.](#)

2. Extract the average interatomic bond distances as the radii of the first coordination spheres. For this, identify the position of the first maximum of the  $g_{AB}(r)$  functions: plot the gofrs.dat file in a spreadsheet application and search for the maxima and minima for each pair of atoms.
3. Identify the radius of the first coordination sphere, as the first minimum of the PDF,  $g_{AB}(r)$ , using spreadsheet software. This is the basis for the entire structural analysis of the fluid; the PDF yields the average bonding status of the atoms in the fluid.
4. Extract the distances of the first minima, i.e., the abscissa, and write them in a separate file, called, for example, *bonds.input*. Alternatively, run one of the *analyze\_gofr* scripts of the UMD package to identify the maxima and the minima of the  $g_{AB}(r)$  functions. At the command line type:  
**analyze\_gofr\_semi\_automatic.py**
5. Click on the position of the maximum and the minimum of the  $g_{AB}(r)$  function displayed in the graph that is opened by the program. The script automatically scans the current folder, identifies all the gofrs.dat files, and

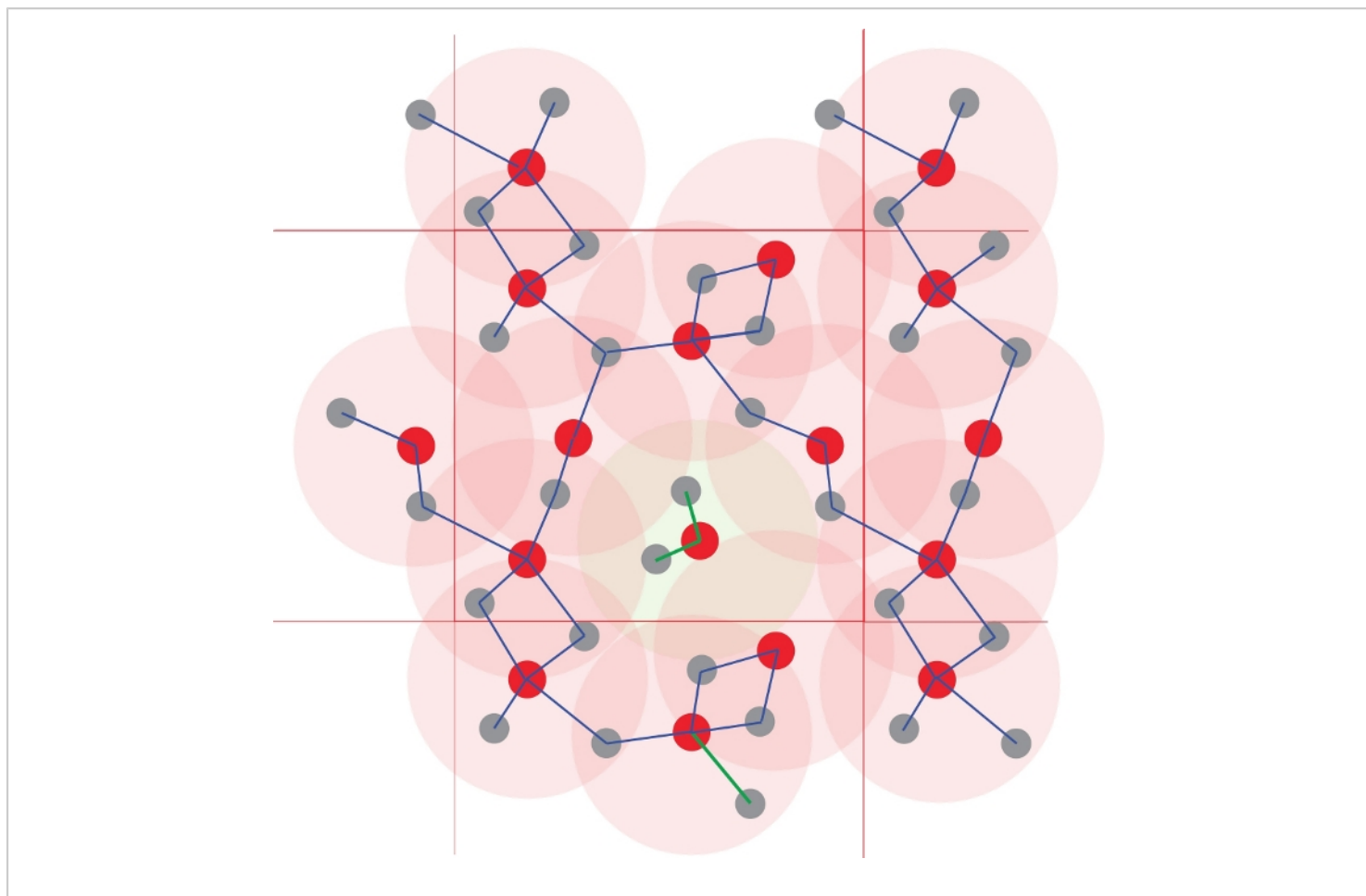
performs the analysis for each one of them. Click again on the maximum and the minimum in the window every time the script needs an educated initial guess.

6. Open and look at the automatically generated file called *bonds.input* that contains the interatomic bond distances.

### 3. Perform the speciation analysis

1. Compute the topology of bonding between the atoms, using the concept of connectivity within graph theory: the atoms are the nodes and the interatomic bonds are the paths. The *speciation\_umd.py* script needs the interatomic bond distances defined in the *bonds.input* file.

**NOTE:** The connectivity matrix is constructed at each time step: two atoms that lie at a distance smaller than the radius of their corresponding first coordination sphere are considered to be bonded, i.e., connected. Various atomic networks are built by treating the atoms as nodes in a graph whose connections are defined by this geometric criterion. These networks are the atomic species, and their ensemble defines the atomic speciation in that particular fluid (**Figure 4**).



**Figure 4: Identification of the atomic clusters.**

The coordination polyhedra are defined using interatomic distances. All atoms at a distance smaller than a specified radius are considered to be bonded. Here the threshold corresponds to the first coordination sphere (the light red circles), defined in **Figure 1**. Polymerization and thus chemical species are obtained from the networks of the bonded atoms. Note the central Red<sub>1</sub>Grey<sub>2</sub> cluster, which is isolated from the other atoms, which form an infinite polymer. [Please click here to view a larger version of this figure.](#)

- Run the speciation script to obtain the connectivity matrix and obtain the coordination polyhedra or the polymerization. At the command line type:

```
speciation_umd.py -f <UMD_filename> -s <Sampling_Frequency> -i <InputFile> -l <MaxLength> -c <Cations> -a <Anions> -m <MinLife> -r <Rings>
```

where the -i flag gives the file with the interatomic bond distances, which was produced for example in the previous step. Alternatively, run the script with one single length for all bonds defined by the -l flag.

**NOTE:** The -c flag specifies the central atoms, and the -a flag the ligands. Both central atoms and ligands can be of different types; in this case, they must be separated by

commas. The `-m` flag gives the minimum time a species must live to be considered in the analysis. By default this minimum time is zero, all occurrences being counted in the final analysis.

1. Run the `speciation_umd.py` script with the flag `-r 0`, which samples the connectivity graph at the first level to identify the coordination polyhedra. For example, a central atom, denoted as a *cation* may be surrounded by one or more *anions* (Figure 4). The speciation script identifies every single one of the coordination polyhedra. The weighted average of all the coordination polyhedra gives the coordination number, identical to the one obtained from the integration of the PDF. At the command line type:  
`speciation_umd.py -f <UMD_filename> -i <InputFile> -c <Cations> -a <Anions> -r 0`

**NOTE:** Average coordination numbers in fluids are fractional numbers. This fractionality comes from the average characteristic of the coordination. The definition based on speciation yields a more intuitive and informative representation of the structure of the fluid, where the relative proportions of the different species, i.e., coordinations, are quantified.

2. Run the `speciation_umd.py` script with the flag `-r 1`, which samples the connectivity graph at all depth levels to obtain the polymerization. The network through the atomic graph has a certain depth, as atoms are bonded further away to other bonds (e.g., in sequences of alternating cations and anions) (Figure 4).
3. Open the two files `.popul.dat` and `.stat.dat` consecutively; these constitute the output of the speciation script. Each cluster is written on one line, specifying its chemical formula, the time at which it formed, the time at which it

died, its lifetime, a matrix with the list of the atoms forming this cluster. Plot the lifetime of each atomic cluster of all the chemical species found in the simulation as found in the `.popul.dat` file (Figure 5).

4. Plot the population analysis with the abundance of each species, as found in the `.stat.dat` file. This analysis, both absolute and relative, corresponds to the actual statistics of the coordination polyhedra for the case `-r 0`; for the case of polymerization, with `-r 1` this needs to be treated carefully as some normalization over the relative number of atoms might need to be applied. The abundance corresponds to the integral over the lifetimes. The `.stat.dat` file also lists the size of each cluster, i.e., how many atoms form it.

#### 4. Compute diffusion coefficients

1. Extract the mean square displacements (MSD) of the atoms as a function of time to obtain the self-diffusivity. The standard formula of the MSD is:

$$MSD_{\alpha}(\tau) = \frac{1}{N_{\alpha}} \frac{1}{N_{init}} \sum_{i=1}^{N_{\alpha}} \sum_{t=1}^{T/2} [r_{\alpha}(t+\tau) - r_{\alpha}(t)]^2$$

where the prefactors are renormalizations. With the MSD tool, there are different ways to analyze the dynamical aspects of the fluids.

**NOTE:**  $T$  is the total time of the simulation and  $N_{\alpha}$  is the number of atoms of type  $\alpha$ . The initial time  $t_0$  is arbitrary and spans the first half of the simulation.  $N_{init}$  is the number of initial times.  $\tau$  is the width of the time interval over which the MSD are computed; its maximum value is half the time length of the simulation. In typical MSD implementations, every window starts at the end of the previous one. But a sparser sampling can speed up the computation of the MSD, without altering the slope of the

MSD. For this, the  $i$ -th window starts at time  $t_0(i)$ , but the  $(i+1)$ -th window starts at time  $t_0(i) + \tau + v$ , where the value of  $v$  is user-defined. Similarly, the width of the window is increased in discrete steps defined by the user, as such:  $\tau(i) = \tau(i-1) + z$ . The values of  $z$  (“horizontal step”) and  $v$  (“vertical step”) are positive or zero; the default for both is 20.

2. Compute the MSD using the series of `msd_umd` scripts. Their output is printed in a `.msd.dat` file, where the MSD of each atomic type, atom, or cluster is printed on one column as a function of time.

1. Compute the average MSD of each atomic type. The MSD are computed for each atom and then averaged for each atomic type. The output file contains one column for each atomic type. At the command line type:

```
msd_umd.py -f <UMD_filename> -z
<HorizontalJump> -v <VerticalJump> -b
<Ballistic>
```

2. Compute the MSD of each atom. The MSD are computed for each atom and then averaged for each atomic type. The output file contains one column for each atom in the simulation, and then one column for each atomic type. This feature allows for identifying atoms that diffuse in two different environments, like liquid and gas, or two liquids. At the command line type:

```
msd_all_umd.py -f <UMD_filename> -z
<HorizontalJump> -v <VerticalJump> -b
<Ballistic>
```

3. Compute the MSD of the chemical species. Use the population of clusters identified with the speciation script, and printed in the `.popul.dat` file. The MSD

are computed for each individual cluster. The output file contains one column for each cluster. To avoid considering large-scale polymers, place a limit on the size of the cluster; its default is 20 atoms. At the command line type:

```
msd_cluster_umd.py -f <UMD_filename> -p
<POPUL_filename> -s <Sampling_Frequency> -
b <Ballistic> -c <ClusterMaxSize>
```

**NOTE:** Defaults values are: `-b 100 -s 1 -c 20`.

3. Plot the MSD using a spreadsheet-based software (**Figure 6**). In a log-log representation of the MSD versus time, identify the slope change. Separate the first part, usually short, which represents the **ballistic** regime, i.e. the conservation of the velocity of atoms after collisions. The second longer part represents the **diffusive** regime, i.e. scattering of the velocity of atoms after collisions.
4. Compute the diffusion coefficients from the slope of the MSD as:

$$D = \frac{MSD}{2Zt}$$

where  $Z$  is the number of degrees of freedom ( $Z = 2$  for diffusion in plane,  $Z = 3$  for diffusion in space), and  $t$  is the time step.

## 5. Time correlation functions

1. Compute the time correlation functions as a measure of the inertia of the system using the general formula:

$$C(\tau) = \frac{1}{\tau} \sum_{t=0}^{\tau/2} A(t + \tau)A(t)$$

$A$  can be a variety of time-dependent variables, such as the atomic positions, atomic velocities, stresses, polarization, etc., each yielding—via the Green-Kubo

relations<sup>12,13</sup>—different physical properties, sometimes after a further transformation.

- Analyze the atomic velocities to obtain the vibrational spectrum of the liquid and alternative expression of the atomic self-diffusion coefficients.

- Run the **vibr\_spectrum\_umd.py** script to compute the atomic velocity-velocity auto-correlation (VAC) function for each atomic type and perform its fast-Fourier transform. At the command line type:

```
vibr_spectrum_umd.py -f <UMD_filename> -t <temperature>
```

where  $-t$  is the temperature that must be defined by the user. The script prints two files: the *.vels.scf.dat* file with the VAC function for each atomic type, and the *.vibr.dat* file with the vibrational spectrum decomposed on each atomic species and the total value.

- Open and read the *vels.scf.dat*. Plot the VAC function from the *vels.scf.dat* file using spreadsheet-like software.
- Keep the real part of the Fourier VAC. This is what yields the vibrational spectrum, as a function of frequency:

$$I(\omega) = \text{Re} \left[ \int_0^\infty m \langle v(t+\tau)v(t) \rangle e^{-i\omega t} \right]$$

where  $m$  are the atomic masses.

- Plot the vibrational spectrum from the *vibr.dat* file using spreadsheet-like software (**Figure 7**). Identify the finite value at  $\omega=0$  that corresponds to the diffusive character of the fluid and the various peaks of the spectrum at finite frequency. Identify the participation of each atomic type to the vibrational spectrum.

**NOTE:** The decomposition on atomic types shows that different atoms have different  $\omega=0$  contributions, corresponding to their diffusion coefficients. The general shape of the spectrum is much smoother with fewer features than for a corresponding solid.

- At the shell, read the integral over the vibrational spectrum, which yields the diffusion coefficients for each atomic species.

**NOTE:** Thermodynamic properties can be obtained by integration from the vibrational spectrum, but the results should be used with caution because of two approximations: the integration is valid within the quasi-harmonic approximation, which does not necessarily hold at high temperatures; and the gas-like part of the spectrum corresponding to the diffusion needs to be discarded. The integration should then be done only over the lattice-like part of the spectrum. But this separation usually requires several further post-processing steps and calculations<sup>14</sup>, which are not covered by the present UMD package.

- Run the **viscosity\_umd.py** script to analyze the self-correlation of the components stress tensor to estimate the viscosity of the melt. At the command line type:

```
viscosity_umd.py -f <UMD_filename> -i <InitialStep> -s <Sampling_Frequency> -o <frequency_of_origin_shift> -l <max_correlation_timelength>
```

**NOTE:** This feature is exploratory and any results must be taken with caution. Firstly, thoroughly check the convergence of the viscosity with respect to the length of the simulation.

1. Derive the viscosity of the fluid from the self-correlation of the stress tensor<sup>15</sup> as:

$$\eta = \frac{V}{3k_B T} \sum_{i,j} \int_0^\infty \langle \sigma_{ij}(t + \tau) \sigma_{ij}(t) \rangle d\tau$$

where  $V$  and  $T$  are the volume and the temperature respectively,  $k_B$  is the Boltzmann constant and  $\sigma_{ij}$  the  $ij$  off-diagonal component of the stress-tensor, expressed in Cartesian coordinates.

2. Use a more adequate fit to obtain a more robust estimate of the viscosity<sup>15,16</sup> and avoid the noise of the stress-tensor auto-correlation function that might arise from the finite size and the finite duration of the simulations. For the auto-correlation function of the stress tensor, use the following functional form<sup>15,16</sup> that yields good results:

$$\frac{1}{3} \sum_{i,j} \langle \sigma_{ij}(t + \tau) \sigma_{ij}(t) \rangle = A e^{-\frac{t}{\tau_1}} + B e^{-\frac{t}{\tau_2}} \sin(\omega t)$$

where  $A$ ,  $B$ ,  $\tau_1$ ,  $\tau_2$ , and  $\omega$  are fit parameters. After integrating, the expression for the viscosity becomes:

$$\frac{\eta k_B T}{V} = A \tau_1 + B \frac{\tau_2}{(1 + \omega^2 \tau_2^2)}$$

## 6. Thermodynamic parameters stemming from the simulations.

1. Run *averages.py* to extract the average values and the spread (as standard deviation) for pressure, temperature, density, and internal energy from the *umd* files. At the command line type:

```
averages.py -f <UMD_filename> -s
<Sampling_Frequency>
```

with `-s 0` as default.

2. Compute the statistical error of the average, using blocking methods.

**NOTE:** There are various flavors of this method. Following the work of Allen and Tildesley<sup>2</sup>, it is common to average over sequences of time blocks, of increasingly longer length, and estimate the standard deviation with respect to the arithmetic average<sup>17</sup>. Convergence may be reached in the limit of many- and long-enough block sizes, when the sampling is uncorrelated. Though the actual threshold value for the convergence usually needs to be chosen manually.

1. Use the halving method<sup>18</sup>: starting with the initial data sample, at each step  $\kappa$ , halve the number of the samples by averaging over every two corresponding consecutive samples from the previous step  $\kappa-1$ :

$$S_i^\kappa = \frac{S_{2i}^{\kappa-1} + S_{2i+1}^{\kappa-1}}{2}$$

2. Run the *fullaverages.py* script to perform the complete statistical analysis, including the error of the mean. At the command line type:

```
fullaverages.py -s <Sampling_Frequency> -u
<units>
```

**NOTE:** The script is automatized to the point of searching for all the *.umd.dat* files in the current directory and performing the analysis for all of them. Defaults are `-s 0 -u 0`. For `-u 0` the output is minimal, and for `-u 1` output is in full, with several alternative units printed out. This script requires graphical support, as it creates a graphical image for checking the convergence for estimating the error on the mean.

## Representative Results

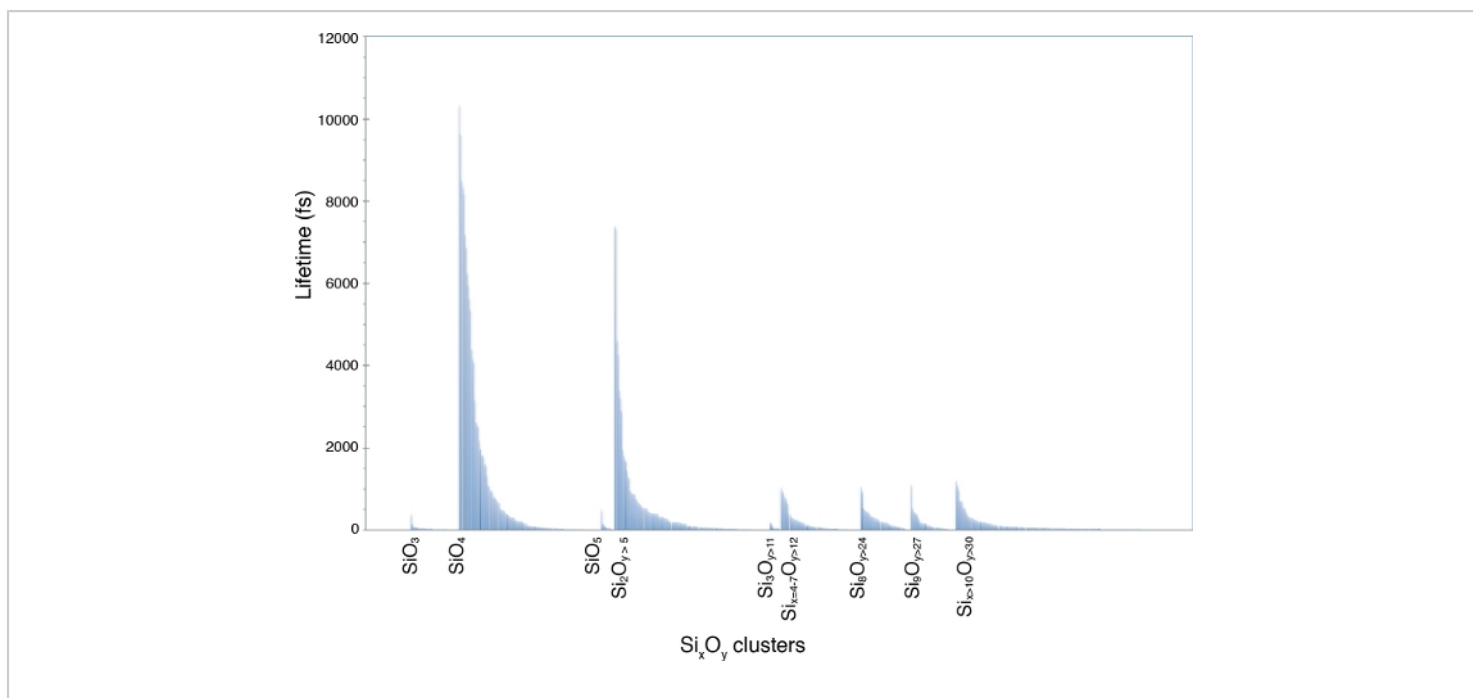
Pyrolite is a model multi-component silicate melt ( $0.5\text{Na}_2\text{O}$   $2\text{CaO}$   $1.5\text{Al}_2\text{O}_3$   $4\text{FeO}$   $30\text{MgO}$   $24\text{SiO}_2$ ) that best approximates the composition of the bulk silicate Earth—the geochemical average of our entire planet, except for its iron-based core<sup>19</sup>. The early Earth was dominated by a series of large-scale melting events<sup>20</sup>, the last one might have engulfed the entire planet, after its condensation for the protolunar disk<sup>21</sup>. Pyrolite represents the best approximant to the composition of such planetary-scale magma oceans.

Consequently, we extensively studied the physical properties of pyrolite melt in the 3,000–5,000 K temperature range and 0–150 GPa pressure range from ab initio molecular-dynamics simulations in the VASP implementation. These thermodynamic conditions entirely characterize the Earth's most extreme magma ocean conditions. Our study is an excellent example of a successful use of the UMD package for the entire in-depth analysis of the melts<sup>22</sup>. We computed the distribution and the average bond lengths, we traced the changes in cation-oxygen coordination, and compared our results to previous experimental and computational studies on amorphous silicates of various compositions. Our in-depth analysis helped decompose standard coordination numbers into their basic constituents, outline the presence of exotic

coordination polyhedra in the melt, and extract lifetimes for all the coordination polyhedra. It also outlined the importance of sampling in simulations in terms of both length of the trajectory and also number of atoms present in the system that is modeled. As for post-processing, the UMD analysis is independent of these factors, however, they should be taken into account when interpreting the results provided by the UMD package. Here, we show a few examples of how the UMD package can be used to extract several characteristic features of the melts, with an application to molten pyrolite.

The Si-O pair distribution function obtained from the `gofrs_umd.py` script shows that the radius of the first coordination sphere, which is the first minimum of the  $g(r)$  function, lies around 2.5 angstroms at  $T = 3000$  K and  $P = 4.6$  GPa. The maximum of the  $g(r)$  lies at 1.635 Å—this is the best approximation to the bond length. The long tail is due to the temperature. Using this limit as the Si-O bond distance, the speciation analysis shows that  $\text{SiO}_4$  units, which can last for up to a few picoseconds, dominate the melt (**Figure 5**). There is an important part of the melt that shows partial polymerization, as reflected by the presence of dimers like  $\text{Si}_2\text{O}_7$ , and trimers like  $\text{Si}_3\text{O}_x$  units. Their corresponding lifetime is in the order of the picosecond. Higher-order polymers all have considerably shorter lifetime.



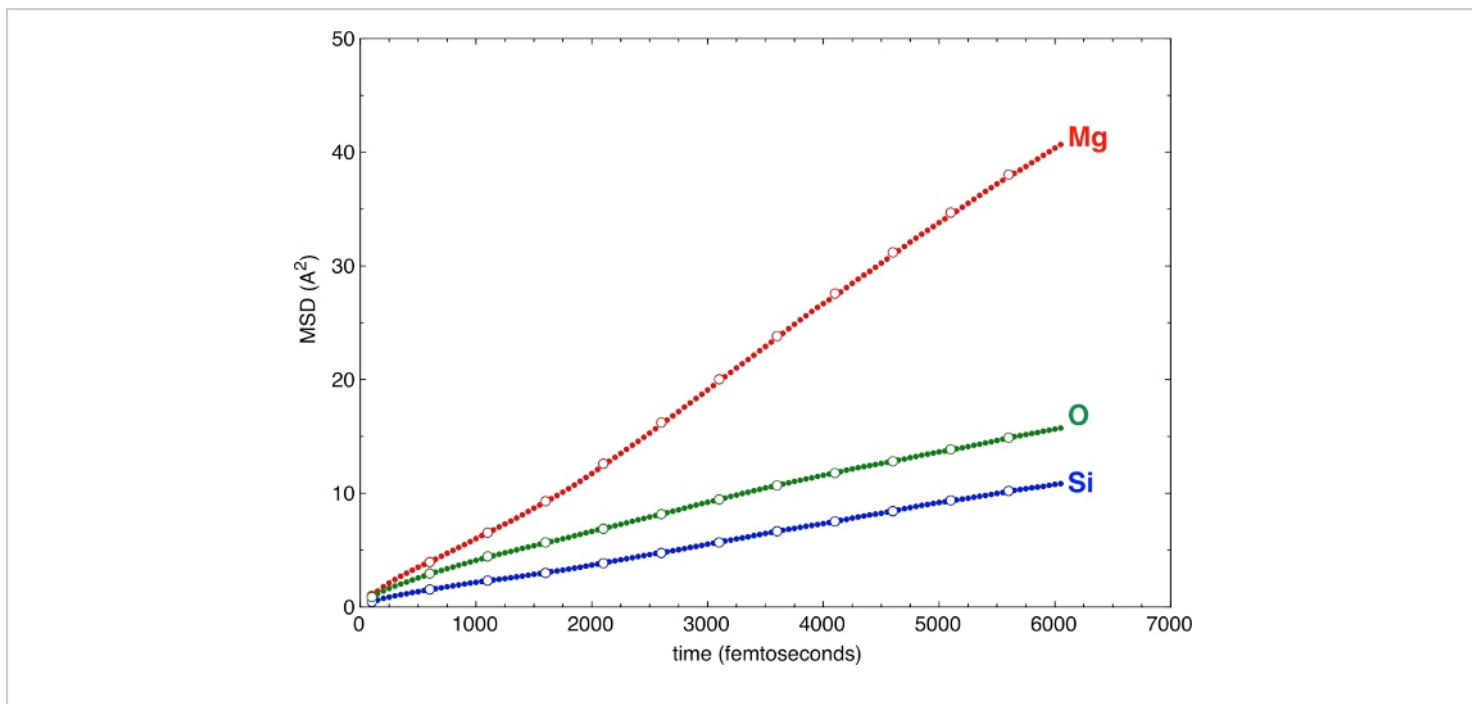


**Figure 5: Lifetime of the Si-O chemical species.**

The speciation is identified in a multi-component melt at 4.6 GPa and 3000 K. The labels mark the SiO<sub>3</sub>, SiO<sub>4</sub>, and SiO<sub>5</sub> monomers and the various Si<sub>x</sub>O<sub>y</sub> polymers. [Please click here to view a larger version of this figure.](#)

The different values of the vertical and horizontal steps, defined by the  $-z$  and  $-v$  flags above, yield various samplings of the MSD (**Figure 6**). Even large values of  $z$  and  $v$  are enough to define the slopes and thus the diffusion coefficients of the different atoms. The gain in time for the post-processing is remarkable when going to large values of  $z$  and  $v$ . The MSD offers a very strong validation criterion for the quality of the simulations. If the diffusive part of the MSD is not sufficiently long, that is a sign that the simulation is too short, and fails to reach the fluid state in a statistical sense. The minimum requirement for the diffusive part of the MSD highly depends on the system. One can require that all the atoms change their site at least once in the structure of the melt in order for

it to be considered as a fluid<sup>10</sup>. An excellent example with applications in planetary sciences is complex silicate melts at high pressures close to or even below their liquidus line<sup>11</sup>. The Si atoms, the major network-forming cations, switch sites after more than two dozen picoseconds. Simulations shorter than this threshold would be considerably under-sampling the possible configurational space. However, as the coordinating anions, namely the O atoms, move faster than the central Si atoms, they can compensate for part of the slow mobility of Si. As such, the entire system could indeed cover a better sampling of the configurational space than assumed only from the Si displacements.

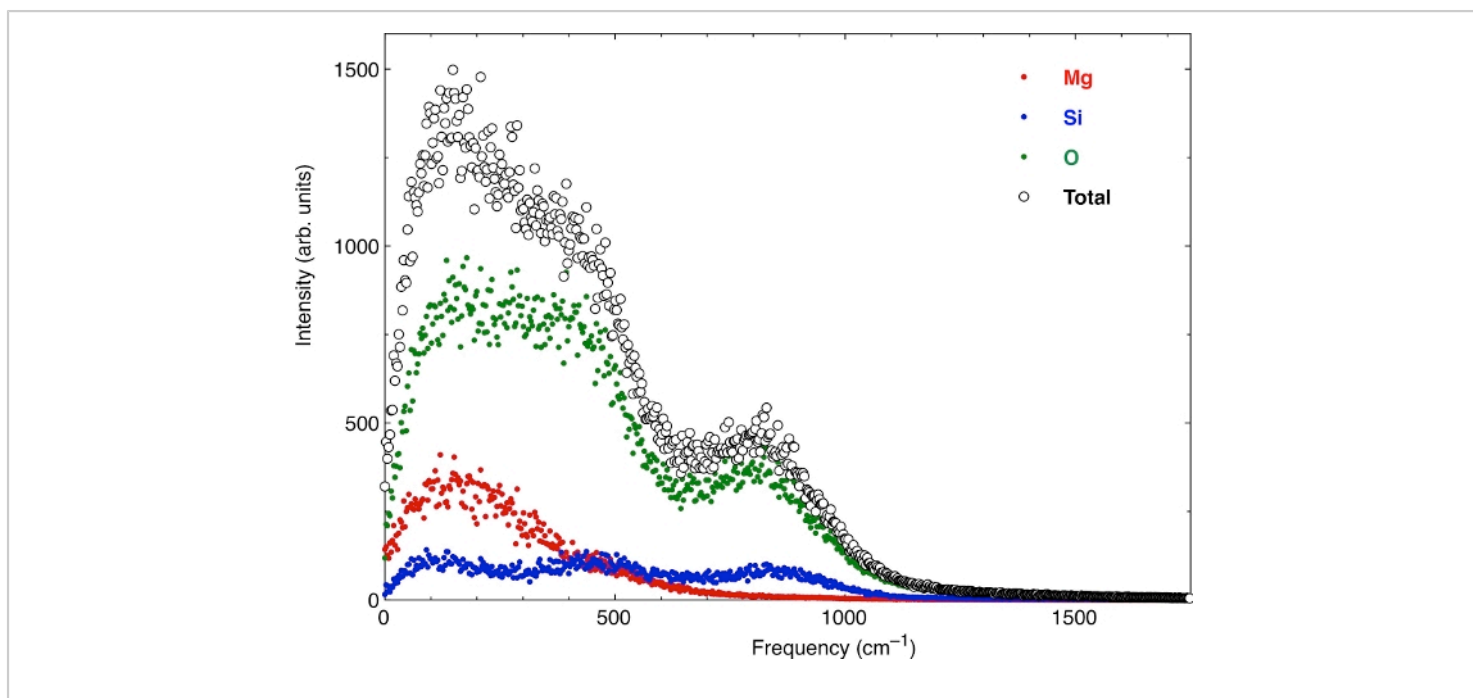


**Figure 6: Mean-square displacements (MSD).**

The MSD are illustrated for a few atomic types of a multi-component silicate melt. The sampling with various horizontal and vertical steps,  $z$  and  $v$ , yields consistent results. Solid circles:  $-z\ 50 -v\ 50$ . Open circles:  $-z\ 250 -v\ 500$ . [Please click here to view a larger version of this figure.](#)

Finally, the atomic VAC functions yield the vibrational spectrum of the melt. **Figure 7** shows the spectrum at the same pressure and temperature conditions as above. We represent the contributions of Mg, Si, and O atoms, as well as the total value. At zero frequency there is a finite value of

the spectrum, which corresponds to the diffusion character of the melt. Extraction of the thermodynamic properties from the vibrational spectrum needs to remove this gas-like diffusive character from zero but also to properly take into account its decay at higher frequencies.



**Figure 7: The vibrational spectrum of pyrolite melt.**

The real part of the Fourier transform of the atomic velocity-velocity self-correlation function yields the vibrational spectrum. Here the spectrum is computed for a multi-component silicate melt. Fluids have a non-zero gas-like diffusive character at zero frequency. [Please click here to view a larger version of this figure.](#)

## Discussion

The UMD package has been designed to work better with ab initio simulations, where the number of snapshots is typically limited to tens to hundreds of thousands of snapshots, with a few hundred atoms per unit cell. Larger simulations are also tractable provided the machine on which the post-processing runs has enough active memory resources. The code distinguishes itself by the variety of properties it can compute and by its open-source license.

The *umd.dat* files are appropriate to the ensembles that preserve the number of particles unchanged throughout the simulation. The UMD package can read files stemming from calculations where the shape and volume of the simulation box varies. These cover the most common calculations, like

NVT and NPT, where the number of particles,  $N$ , temperature  $T$ , volume,  $V$ , and/or pressure,  $P$ , are kept constant.

For the time being the pair distribution function as well as all the scripts needing to estimate the interatomic distances, like the speciation scripts, work only for orthogonal unit cells, meaning for cubic, tetragonal, and orthorhombic cells, where the angles between the axes are  $90^\circ$ .

The major lines of development for version 2.0 are removal of the orthogonality restriction for distances and adding more features for the speciation scripts: to analyze individual chemical bonds, to analyze the interatomic angles, and to implement the second coordination sphere. With help from

external collaboration, we are working on porting the code onto a GPU for faster analysis in larger systems.

## Disclosures

The authors have nothing to disclose.

## Acknowledgments

This work was supported by the European Research Council (ERC) under the European Union Horizon 2020 research and innovation program (grant agreement number 681818 IMPACT to RC), by the Extreme Physics and Chemistry Directorate of the Deep Carbon Observatory, and by the Research Council of Norway through its Centers of Excellence funding scheme, project number 223272. We acknowledge access to the GENCI supercomputers through the stl2816 series of eDARI computing grants, to the Irene AMD supercomputer through the PRACE RA4947 project, and the Fram supercomputer through the UNINETT Sigma2 NN9697K. FS was supported by a Marie Skłodowska-Curie project (grant agreement ABISSE No.750901).

## References

1. Frenkel, D., Smit, B. *Understanding Molecular Simulation*. From Algorithms to Applications, Elsevier (2001).
2. Allen, M. P., Tildesley, D. J., Allen T. *Computer Simulation of Liquids*. Oxford University Press, (1989).
3. Zepeda-Ruiz, L. A., Stukowski, A., Oettel, T., Bulatov, V. V. Probing the limits of metal plasticity with molecular-dynamics simulations, *Nature Publishing Group*. **550** (7677), 492–495 (2017).
4. Humphrey, W., Dalke, A., Schulten, K. VMD: Visual molecular dynamics. *Journal of Molecular Graphics & Modeling*. **14** (1), 33–38 (1996).
5. Momma, K., Izumi, F. VESTA3 for three-dimensional visualization of crystal, volumetric and morphology data, *Journal of Applied Crystallography*. **44** (6), 1272–1276 (2011).
6. Brehm, M., Kirchner, B. TRAVIS - A free Analyzer and Visualizer for Monte Carlo and Molecular Dynamics Trajectories. *Journal of Chemical Information and Modeling*. **51** (8), 2007–2023 (2011).
7. Stixrude, L. Visualization-based analysis of structural and dynamical properties of simulated hydrous silicate melt. *Physics and Chemistry of Minerals*. **37** (2), 103–117 (2009).
8. Kresse, G., Hafner, J. Ab initio Molecular-Dynamics for Liquid-Metals. *Physical Review B*. **47** (1), 558–561 (1993).
9. Gygi, F. Architecture of Qbox: A scalable first-principles molecular dynamics code. *IBM Journal of Research and Development*. **52** (1-2), 137–144 (2008).
10. Harvey, J.P., Asimow, P. D. Current limitations of molecular dynamic simulations as probes of thermo-physical behavior of silicate melts. *American Mineralogist*. **100** (8-9), 1866–1882 (2015).
11. Caracas, R., Hirose, K., Nomura, R., Ballmer, M. D. Melt–crystal density crossover in a deep magma ocean. *Earth and Planetary Science Letters*. **516**, 202–211 (2019).
12. Green, M. S. Markoff Random Processes and the Statistical Mechanics of Time-Dependent Phenomena. II. Irreversible Processes in Fluids. *The Journal of Chemical Physics*. **22** (3), 398–413 (1954).

13. Kubo, R. Statistical-Mechanical Theory of Irreversible Processes. I. General Theory and Simple Applications to Magnetic and Conduction Problems. *Journal of the Physical Society of Japan*. **12** (6), 570–586 (1957).
14. Lin, S.T., Blanco, M., Goddard III, W. A. The two-phase model for calculating thermodynamic properties of liquids from molecular dynamics: Validation for the phase diagram of Lennard-Jones fluids. *The Journal of Chemical Physics*. **119** (22), 11792–11805 (2003).
15. Meyer, E. R., Kress, J. D., Collins, L. A., Ticknor, C. Effect of correlation on viscosity and diffusion in molecular-dynamics simulations. *Physical Review E*. **90** (4), 1198–1212 (2014).
16. Soubiran, F., Militzer, B., Driver, K. P., Zhang, S. Properties of hydrogen, helium, and silicon dioxide mixtures in giant planet interiors. *Physics of Plasmas*. **24** (4), 041401–7 (2017).
17. Flyvbjerg, H., Petersen, H. G. Error estimates on averages of correlated data. *The Journal of Chemical Physics*. **91** (1), 461–466 (1989).
18. Tuckerman, M. E. *Statistical mechanics: theory and molecular simulation*. Oxford University Press, (2010).
19. McDonough, W. F., Sun S. S. The composition of the Earth. *Chemical Geology*. **120**, 223-253 (1995).
20. Elkins-Taton, L. T. Magma oceans in the inner solar system. *Annual Review of Earth and Planetary Sciences*. **40**, 113-139 (2012).
21. Lock, S. J. et al. The origin of the Moon within a terrestrial synestia. *J. Geophysical Research: Planets*. **123**, 910-951 (2018).
22. Solomatova, N. V., Caracas, R. Pressure-induced coordination changes in a pyrolitic silicate melt from *ab initio* molecular dynamics simulations, *Journal of Geophysical Research: Solid Earth*. **124**, 11232-11250 (2019).