The Journal of Open Source Software

IonDiff: command-line tool to identify ionic diffusion events and hopping correlations in molecular dynamics simulations

Cibrán López[1,2], **Riccardo Rurali**[3], and **Claudio Cazorla**[1,2¶]

1 Departament de Física, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain. **2** Barcelona Research Center in Multiscale Science and Egineering, Universitat Politècnica de Catalunya, 08019 Barcelona, Spain. **3** Institut de Ciència de Materials de Barcelona, ICMAB-CSIC, Campus UAB, 08193 Bellaterra, Spain. ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Molecular dynamics (MD) simulations of fast-ion conductors render the trajectories of the atoms conforming them. However, extracting meaningful insights from this data is often a challenge since most common analysis rely on active supervision of the simulations and definition of arbitrary material-dependent parameters, thus frustrating high throughput screenings. In particular, to the best of our knowledge, determining exact ionic migrating paths and the level of coordination between mobile particles in diffusive events have not been previously addressed in a systematic and quantitative manner, despite its central role in the understanding and design of high performance solid-state electrolytes. Here, we introduce a completely unsupervised approach for analysing ion-hopping events in MD simulations. Based on k-means clustering, our algorithm identifies with precision which and when particles diffuse during a simulation, thus identifying their exact migrating paths. This analysis allows also for the quantification of correlations between many diffusing ions as well as of key atomistic descriptors like the duration/length of diffusion events and residence times, to cite some examples. Moreover, the present implementation introduces an optimized code for computing the full ion diffusion coefficient, that is, entirely considering ionic correlations, thus going beyond the dilute limit approximation.

Statement of need

Fast-ion conductors (FIC) are materials in which some of their constituent atoms diffuse with large drift velocities comparable to those found in liquids (Hull, 2004; Sagotra & Cazorla, 2017). FIC are the pillars of many energy conversion and storage technologies like solid-state electrochemical batteries and fuel cells. Molecular dynamics (MD) simulations is a computational method that employs Newton's laws to evaluate the trajectory of ions in complex atomic and molecular systems. MD simulations of FIC are highly valuable since they can accurately describe the diffusion and vibration of the ions conforming them. Nevertheless, there is a painstaking lack of handy computational tools for analyzing the outputs of FIC MD simulations in an unsupervised and materials-independent manner, thus frustrating the fundamental understanding and possible rational design of FIC.

IonDiff

IonDiff efficiently addresses the challenge described above by implementing unsupervised machine learning approaches in a repository of Python scripts designed to extract the exact migrating paths of diffusive particles from MD simulations, along with other physically relevant quantities like the degree of correlation between diffusive ions, ionic residence times in metastable positions and the length and duration of ionic hops. Additionally, IonDiff efficiently and seamlessly evaluates full ion diffusion coefficients, which in contrast to tracer ion diffusion coefficients fully encompass ionic correlations. Periodic boundary conditions are fully accounted for by IonDiff.

The repository is divided into three independent functionalities:

- *identify_diffusion*: extraction of the migrating paths from a given MD simulation. It generates a **DIFFUSION** file in the folder containing the inputs and outputs of the MD simulation. This file contains all the necessary atomistic information for the following analysis of ionic diffusion events.
- *analyze_correlations*: analysis of the correlations between ionic diffusion events extracted from a series of MD simulations (the **DIFFUSION** file for each of these simulations will be generated if it does not exist yet). A more technically detailed description of this functionality can be found in the Methods section and in (López et al., 2024b).
- *analyze_descriptors*: extraction and analysis of spatio-temporal descriptors involving the ionic diffusion events identified in the MD simulations. In this library, an optimized approach for computing the full ionic diffusion coefficient (i.e., including ionic cross correlations, proven to be non-negligible in FIC (López et al., 2024b; Molinari et al., 2021; Sasaki et al., 2023)) is implemented. A technically detailed description of this functionality can be found in (López et al., 2024b).

The minimal input needed (besides the file containing the actual atomistic trajectories) consists in an **INCAR** file with the **POTIM** and **NBLOCK** flags (indicating the simulation time step and the frequency with which the configurations are written, respectively). After installation, all routines are easily controlled from the command line. More detailed information can be found in the documentation of the project (including specific **README**s within each folder).

The script allows graphing the identified diffusion paths for each simulated particle and provides the confidence interval associated with the results retrieved by the algorithm. An example of the analysis performed on an *ab initio* MD (AIMD) simulation based on density functional theory (DFT) is shown in Figure 1. The AIMD configurations file employed in this example is available online at (López et al., 2024a), along with many other AIMD simulations comprehensively analyzed in two previous works (López et al., 2023, 2024b).

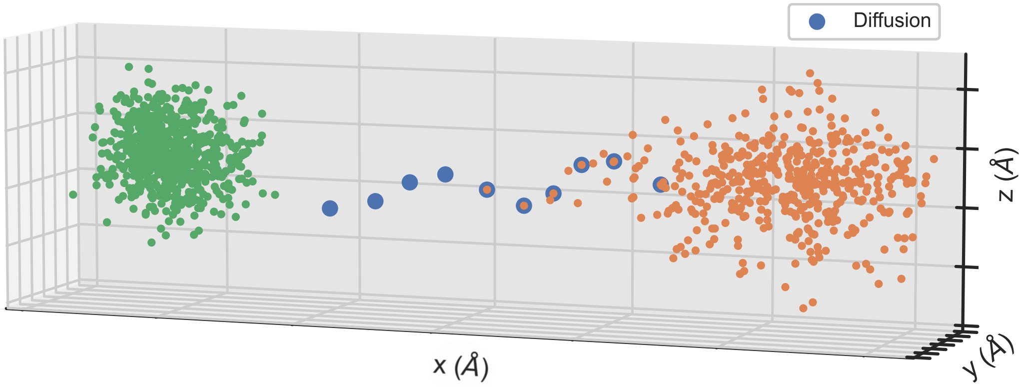

Figure 1: Example of the performance of our unsupervised algorithm at extracting the diffusive path for an arbitrary particle in an AIMD simulation of $SrCoO_{3-x}$ at a temperature of 400K. Green and orange dots reproduce two different ionic vibrational centers while the blue dots represent the ion diffusion path between them.

Moreover, users may find information regarding their previous executions of the scripts in the *logs* folder, which should be used to track possible errors on the data format and more. Finally, a number of tests for checking out all **IonDiff** functions can be found in the *tests* folder.

Mainly, our code is based on the sklearn (Pedregosa et al., 2011) implementation of the k-means clustering method. The default values of the sklearn hyperparameters are the ones used by IonDiff, although these can be varied at wish by the user. Additionally, the python libraries numpy (Harris et al., 2020) and matplotlib (Hunter, 2007) are used to perform numerical analysis and plotting, respectively. The current IonDiff version reads information from VASP (Kresse & Furthmüller, 1996) simulations; future releases, already under active development, will extend its scope to simulation data obtained from other quantum and classical molecular dynamics packages.

Methods

Ionic conductivity

The (full) ionic diffusion coefficient consists on two parts (Molinari et al., 2021; Sasaki et al., 2023), one that involves the mean-square displacement of a particle with itself (MSD$_{self}$) and another that represents the mean-squared displacement of a particle with all others (MSD$_{distinct}$). MSD$_{distinct}$ accounts for the influence of many-atoms correlations in ionic diffusive events. Typically, the distinct part of the MSD is neglected in order to accelerate the estimation and convergence of diffusion coefficients. However, many-ions correlations have been recently demonstrated to be essential in FIC (López et al., 2024b) hence should not be disregarded in practice. IonDiff provides a novel implementation of the full ionic diffusion coefficient, exploiting the matricial representation of this calculation.

The ionic conductivity (σ) is computed like (Sasaki et al., 2023):

$$\begin{aligned}\sigma = \lim_{\Delta t \to \infty} \frac{e^2}{2 n_d V k_B T} \Bigg[\sum_i z_i^2 \langle [\mathbf{r}_i(t_0 + \Delta t) - \mathbf{r}_i(t_0)]^2 \rangle_{t_0} + \\ + \sum_{i,j \neq i} z_i z_j \langle [\mathbf{r}_i(t_0 + \Delta t) - \mathbf{r}_i(t_0)] \cdot [\mathbf{r}_j(t_0 + \Delta t) - \mathbf{r}_j(t_0)] \rangle_{t_0} \Bigg]\end{aligned} \tag{1}$$

where e, V, k_B, and T are the elementary charge, system volume, Boltzmann constant, and temperature of the MD simulation, respectively, z_i the ionic charge and $\mathbf{r}_i = x_{1i}\hat{i} + x_{2i}\hat{j} + x_{3i}\hat{k}$ the Cartesian position of particle i, n_d the number of spatial dimensions, Δt the time window, and t_0 the temporal offset of Δt. Thus, for those simulations in which only one atomic species diffusses, the three-dimensional ionic diffusion coefficient reads:

$$\begin{aligned}D = \lim_{\Delta t \to \infty} \frac{1}{6 \Delta t} \Bigg[\sum_i \langle [\mathbf{r}_i(t_0 + \Delta t) - \mathbf{r}_i(t_0)]^2 \rangle_{t_0} + \\ + \sum_{i,j \neq i} \langle [\mathbf{r}_i(t_0 + \Delta t) - \mathbf{r}_i(t_0)] \cdot [\mathbf{r}_j(t_0 + \Delta t) - \mathbf{r}_j(t_0)] \rangle_{t_0} \Bigg] = \\ = \lim_{\Delta t \to \infty} \frac{1}{6 \Delta t} \left[\text{MSD}_{self}(\Delta t) + \text{MSD}_{distinct}(\Delta t) \right]\end{aligned} \tag{2}$$

All the ionic displacements appearing in Eq. (2) can be computed just once and stored in a four-dimensional tensor thus allowing for simple vectorization and very much fast processing with python libraries (e.g., numpy) as compared to traditional calculation loops. Then, for a simulation with n_t time steps, $n_{\Delta t}$ temporal windows, and n_p number of atoms for the diffusive species, we only need to compute:

$$\Delta x(\Delta t, i, d, t_0) = x_{di}(t_0 + \Delta t) - x_{di}(t_0) \tag{3}$$

being $\Delta x(\Delta t, i, d, t_0)$ a four rank tensor of dimension $n_{\Delta t} \times n_t \times n_p \times n_d$ that stores all mean displacements of temporal length Δt for particle i in space dimension d. This leads to:

$$\begin{aligned} \text{MSD}_{self}(\Delta t) &= \frac{1}{n_p} \sum_{i=1}^{n_p} \langle \sum_d \Delta x(\Delta t, i, d, t_0) \cdot \Delta x(\Delta t, i, d, t_0) \rangle_{t_0} \\ \text{MSD}_{distinct}(\Delta t) &= \frac{2}{n_p(n_p - 1)} \sum_{i=1}^{n_p} \sum_{j=i+1}^{n_p} \langle \sum_d \Delta x(\Delta t, i, d, t_0) \cdot \Delta x(\Delta t, j, d, t_0) \rangle_{t_0} \end{aligned} \tag{4}$$

Note that we keep D_{self} and $D_{distinct}$ separate since this allows for a straightforward evaluation of the D contributions resulting from the ionic correlations without increasing the code complexity.

In terms of memory resources, this implementation scales linearly with the length of the temporal window, the total duration of the simulation and the number of mobile ions.

Ionic hop identification

Our method for identifying vibrational centers from sequential ionic configurations relies on k-means clustering, an unsupervised machine learning algorithm. This method assumes isotropy in the fluctuations of non-diffusive particles. Importantly, our approach circumvents the need for defining arbitrary, materials-dependent threshold distances to analyze ionic hops.

K-means algorithm conforms spherical groups that, for every subgroup $G = \{G_1, G_2, \dots, G_k\}$ in a dataset, minimize the sum of squares:

$$\sum_{i=1}^{N} \min_{\mu_j \in G_j} \left(\| \mathbf{x}_i - \mu_j \|^2 \right) \tag{5}$$

where $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ are the N data points and μ_j the mean at G_j.

This approach is particularly well-suited for crystals, as atoms typically fluctuate isotropically around their equilibrium positions. For materials where atoms exhibit strong anisotropic vibrations, IonDiff also permits the selection of alternative clustering schemes, such as spectral clustering, which is effective for cases where group adjacency is significant. Nevertheless, in a previous work (López et al., 2024b), it was found that the performance of k-means clustering in identifying ionic hops in standard and technologically relevant fast-ion conductors was generally superior to that of other clustering approaches.

The number of clusters, or equivalently, ionic vibrational centers, determined by IonDiff for a molecular dynamics (MD) simulation is the one that maximizes the average silhouette ratio. This metric assesses the similarity of a point within its own cluster and its dissimilarity in comparison to other clusters. The average silhouette ratio is defined as:

$$S(k) = \frac{b(k) - a(k)}{\max\left(a(k), b(k)\right)} \tag{6}$$

where:

$$
\begin{aligned}
a(k) &= \frac{1}{|G_I| - 1} \sum_{j=1}^{k} \|\mathbf{x}_k - \mathbf{x}_j\|^2 \\
b(k) &= \min_{J \neq I} \frac{1}{|G_J|} \sum_{j=1}^{k} \|\mathbf{x}_k - \mathbf{x}_j\|^2
\end{aligned}
\tag{7}
$$

Once the number of vibrational centers, along with their real-space location and temporal evolution, are determined, ionic diffusion paths are delineated as the segments connecting two distinct vibrational centers over time Figure 1. In other words, the points located between different ionic vibrational centers, that is, different k-means clusters, are regarded as part of the ionic diffusion path connecting them. Due to the discrete nature of the generated trajectories and intricacies of the k-means clustering approach, establishing the precise start and end points of ionic diffusion paths is challenging. Consequently, we adopt an arbitrary yet physically plausible threshold distance of 0.5 Å from the midpoint of the vibrational centers to define the extremities of diffusive trajectories. Tests performed in (López et al., 2024b) have shown that reasonable variations of this parameter value have negligible effects on the analysis results obtained with IonDiff.

Correlations between mobile ions

To quantitatively evaluate the correlations and level of concertation between a variable number of mobile ions, we developed the following algorithm. Beginning with a given sequence of ionic configurations from a molecular dynamics simulation, we compute the correlation matrix for diffusive events. Initially, we assign a value of "1" to each diffusing particle and "0" to each vibrating particle at every time frame. This binary assignment is facilitated by the ionic hop identification algorithm introduced earlier.

Due to the discrete nature of the ionic trajectories and to enhance numerical convergence in subsequent correlation analysis, the multistep time functions are approximated using Gaussians with widths equal to their half-maxima (commonly known as the "full-width-at-half-maximum" or FWHM method used in signal processing). Subsequently, we compute the $N \times N$ correlation matrix, where N represents the number of potentially mobile ions, using all gathered simulation data. However, this correlation matrix may be challenging to converge due to its statistical nature, especially in scenarios with limited mobile ions and time steps, typical of AIMD simulations.

Moreover, uncorrelated ion hops occurring simultaneously could be erroneously interpreted as correlated. To address these practical challenges, we compute a reference correlation matrix based on a randomly distributed sequence of ionic hops, with the Gaussian FWHM matching the mean diffusion time determined during the simulation. It is important to note that due to the finite width of the Gaussians, this reference matrix is not exactly the identity matrix.

Next, covariance coefficients in the original correlation matrix larger (smaller) than the corresponding random reference values were considered as true correlations (random noise) and rounded off to one (zero) for simplification. To ensure an accurate assessment of many-ion correlations, different hops of the same ion are treated as independent events. Ultimately, this process results in a correlation matrix comprising ones and zeros, facilitating the determination of the number of particles that remain concerted during diffusion.

Acknowledgements

C.C. acknowledges support from the Spanish Ministry of Science, Innovation, and Universities under the fellowship RYC2018-024947-I and PID2020-112975GB-I00 and grant TED2021-130265B-C22. C.L. acknowledges support from the Spanish Ministry of Science, Innovation, and Universities under the FPU grant, and the CSIC under the "JAE Intro SOMdM 2021" grant program. The authors thankfully acknowledge the computer resources at MareNostrum, and the technical support provided by Barcelona Supercomputing Center (FI-1-0006, FI-2022-2-0003, FI-2023-1-0002, FI-2023-2-0004, and FI-2023-3-0004). R.R. acknowledges financial support from the MCIN/AEI/10.13039/501100011033 under grant no. PID2020-119777GB-I00, the Severo Ochoa Centres of Excellence Program (CEX2019-000917-S), and the Generalitat de Catalunya under grant no. 2017SGR1506.

References

Harris, C. R., Millman, K. J., & van der Walt, S. J. et al. (2020). Array programming with NumPy. *Nature*, *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hull, S. (2004). Superionics: crystal structures and conduction processes. *Rep. Prog. Phys.*, *67*, 1233. https://doi.org/10.1088/0034-4885/67/7/R05

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.*, *9*, 90–95. https://doi.org/10.1109/MCSE.2007.55

Kresse, G., & Furthmüller, J. (1996). Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B*, *54*, 11169. https://doi.org/10.1103/PhysRevB.54.11169

López, C., Emperador, A., & Saucedo, E. et al. (2023). Universal ion-transport descriptors and classes of inorganic solid-state electrolytes. *Mater. Horiz.*, *10*, 1757–1768. https://doi.org/10.1039/D2MH01516A

López, C., Rurali, R., & Cazorla, C. (2024a). DFT-AIMD database. *NOMAD*. https://doi.org/10.17172/NOMAD/2024.04.01-1

López, C., Rurali, R., & Cazorla, C. (2024b). How concerted are ionic hops in inorganic solid-state electrolytes? *J. Am. Chem. Soc.*, *146*, 8269–8279. https://doi.org/10.1021/jacs.3c13279

Molinari, N., Xie, Y., Leifer, I., Marcolongo, A., Kornbluth, M., & Kozinsky, B. (2021). Spectral denoising for accelerated analysis of correlated ionic transport. *Phys. Rev. Lett.*, *127*, 025901. https://doi.org/10.1103/PhysRevLett.127.025901

Pedregosa, F., Varoquaux, G., & Gramfort, A. (2011). Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, *12*, 2825–2830. https://doi.org/10.48550/arXiv.1201.0490

Sagotra, A. K., & Cazorla, C. (2017). Stress-mediated enhancement of ionic conductivity in fast-ion conductors. *ACS Appl. Mater. Interfaces.*, *9*, 38773. https://doi.org/10.1021/acsami.7b11687

Sasaki, R., Gao, B., Hitosugi, T., & Tateyama, Y. (2023). Nonequilibrium molecular dynamics for accelerated computation of ion–ion correlated conductivity beyond Nernst–Einstein limitation. *Npj Comput. Mater.*, *9*, 48. https://doi.org/10.1038/s41524-023-00996-8