The Journal of Open Source Software

YetAnotherSimulationSuite.jl: An Atomic Simulation Suite in Julia

Brian C. Ferrari [1]

1 Leiden Institute of Chemistry, Leiden University, Leiden 2300 RA, The Netherlands

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 17 September 2025
Published: unpublished

Summary

`YetAnotherSimulationSuite.jl` (YASS) is a simulation suite for atomic simulations in Julia (Bezanson et al., 2017). YASS aims to be highly performant, memory efficient, flexible, and extensible. YASS was developed with a focus on making it incredibly easy to add or customize anything within the package, making niche research methods more accessible to the average user.

YASS features a wide variety of capabilities useful for molecular simulations:

- Reading and writing over 20 different file types
- Basic manipulation of atomic structures
- Geometry and cell optimizations
- Harmonic frequency calculations
- Velocity autocorrelation
- Molecular dynamics simulations
- Radial and angular distribution functions

Statement of Need

Within the field of atomic simulations, there exists a vast number of software packages for performing these simulations. There are those that focus on ease of use and flexibility (i.e., ASE (Larsen et al., 2017)), and those that focus on speed and memory efficiency (i.e., LAMMPS(Thompson et al., 2022), GROMACS(Abraham et al., 2015; Lindahl et al., 2001; Pronk et al., 2013; Van Der Spoel et al., 2005), JaxMD(Schoenholz & Cubuk, 2020)). In an effort to offer both ease of use and speed, ASE offers interfaces to many of the performance-focused simulation suites. However, using these interfaces limits the flexibility originally offered by ASE and can reduce the memory efficiency offered by the faster simulation suite. The Julia programming language offers the perfect solution to this problem, as it can be as performant as C++ with the simplicity of Python.

In theoretical chemistry, it is quite common for two packages with similar functionality to flourish; for instance, both LAMMPS and GROMACS offer similar functionality, but users tend to prefer the interface of one over the other. This helps ensure that for all types of users there is an interface that feels more natural for them. Currently, `Molly.jl` (Greener, 2024) and `NQCDynamics.jl` (Gardner et al., 2022) are the only packages in the Julia ecosystem that perform molecular dynamics (MD) simulations. The latter focuses on the more niche topic of nonadiabatic quantum classical dynamics (NQCD), leaving only the former as an option for users wanting a general atomic simulation suite. This creates a problem for users who dislike the `Molly.jl` interface but want to perform MD in Julia. YASS solves this problem by offering users an alternative interface for atomic simulations.

Examples

Two simple examples are shown here to illustrate how YASS can be used for vibrational frequency analysis of molecular systems. The examples focus only on this topic, as the process also utilizes many additional YASS features. The first example below shows the steps required to calculate the harmonic frequencies of a water molecule.

```julia
using Optim
using YetAnotherSimulationSuite

# Read initial structure
molecule = readSystem("h2o.xyz")

# Run geometry optimization
optimized = opt(TIP4Pf(), LBFGS(), molecule)

# Save optimized structure
write("optimized.xyz", optimized)

# Calculate frequencies and normal modes
freqs, modes = getHarmonicFreqs(TIP4Pf(), optimized)
```

The second example, shown below, calculates the vibrational density of states (VDOS) using the velocity autocorrelation function (VACF). In this example, the `water_at_250K.xyz` file should contain a cell with water at 250 Kelvin. YASS can also be used to generate this initial configuration, but for simplicity, it is not shown here.

```julia
using YetAnotherSimulationSuite

# Read initial structure
water = readSystem("water_at_250K.xyz")

# Create NVE ensemble
ensemble = NVE(water)

# Run 10 picosecond simulation with 0.1 fs timestep
traj = run(TIP4Pf(), water, 10u"ps", 0.1u"fs", ensemble)

# Extract velocities and masses
vel, mas = getVelMas(traj)

# Configure VACF calculation
inp = vacfInps(
    vel,        # Velocity trajectories
    mas,        # Atomic masses
    1e16u"Hz", # Sampling frequency (1/fs = 1e15 Hz)
    true,       # Normalize VACF
    Hann,       # Window function
    4,          # FFT padding factor
    true        # Mirror the data
)

# Calculate VDOS
out = VDOS(inp)
```

More examples and detailed explanations can be found in the YASS documentation.

Dependencies

YASS relies on several packages and libraries that deserve to be credited.

- `Chemfiles` is used for I/O operations.
- `Optim.jl`(Mogensen & Riseth, 2018) is used for geometry and cell optimization.
- `OrdinaryDiffEq.jl` (Rackauckas & Nie, 2017) is used as the integrator for MD simulations.
- `FiniteDifferences.jl` is used for computing Jacobians for harmonic frequency analysis.
- `FFTW` (Frigo & Johnson, 1998) is used for fast Fourier transforms.
- `StaticArrays.jl` is used for memory efficiency.
- `JLD2.jl` is used to store complex data structures.
- `Clustering.jl` is used for molecular identification.

Acknowledgements

BCF thanks Katie Slavicinska for creating the YASS logo.

References

Abraham, M. J., Murtola, T., Schulz, R., Páll, S., Smith, J. C., Hess, B., & Lindahl, E. (2015). GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, *1*, 19–25. https://doi.org/10.1016/j.softx.2015.06.001

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Frigo, M., & Johnson, S. G. (1998). FFTW: An adaptive software architecture for the FFT. *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, *3*, 1381–1384. https://doi.org/10.1109/icassp.1998.681704

Gardner, J., Douglas-Gallardo, O. A., Stark, W. G., Westermayr, J., Janke, S. M., Habershon, S., & Maurer, R. J. (2022). NQCDynamics. Jl: A julia package for nonadiabatic quantum classical molecular dynamics in the condensed phase. *The Journal of Chemical Physics*, *156*(17).

Greener, J. G. (2024). Differentiable simulation to develop molecular dynamics force fields for disordered proteins. *Chemical Science*, *15*, 4897–4909. https://doi.org/10.1039/D3SC05230C

Larsen, A. H., Mortensen, J. J., Blomqvist, J., Castelli, I. E., Christensen, R., Dułak, M., Friis, J., Groves, M. N., Hammer, B., Hargus, C., & others. (2017). The atomic simulation environment—a python library for working with atoms. *Journal of Physics: Condensed Matter*, *29*(27), 273002.

Lindahl, E., Hess, B., & Van Der Spoel, D. (2001). GROMACS 3.0: A package for molecular simulation and trajectory analysis. *Molecular Modeling Annual*, *7*(8), 306–317. https://doi.org/10.1007/s008940100045

Mogensen, P., & Riseth, A. (2018). Optim: A mathematical optimization package for julia. *Journal of Open Source Software*, *3*(24). https://doi.org/10.21105/joss.00615

Pronk, S., Páll, S., Schulz, R., Larsson, P., Bjelkmar, P., Apostolov, R., Shirts, M. R., Smith, J. C., Kasson, P. M., Van Der Spoel, D., & others. (2013). GROMACS 4.5: A high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics*, *29*(7), 845–854. https://doi.org/10.1093/bioinformatics/btt055

Rackauckas, C., & Nie, Q. (2017). Differentialequations. Jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, *5*(1), 15–15.

Schoenholz, S. S., & Cubuk, E. D. (2020). *{JAX} {MD}: End-to-end differentiable, hardware accelerated, molecular dynamics in pure python*. https://openreview.net/forum?id=r1xMnCNYvB

Thompson, A. P., Aktulga, H. M., Berger, R., Bolintineanu, D. S., Brown, W. M., Crozier, P. S., In't Veld, P. J., Kohlmeyer, A., Moore, S. G., Nguyen, T. D., & others. (2022). LAMMPS-a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications*, *271*, 108171. https://doi.org/10.1016/j.cpc.2021.108171

Van Der Spoel, D., Lindahl, E., Hess, B., Groenhof, G., Mark, A. E., & Berendsen, H. J. (2005). GROMACS: Fast, flexible, and free. *Journal of Computational Chemistry*, *26*(16), 1701–1718. https://doi.org/10.1002/jcc.20291