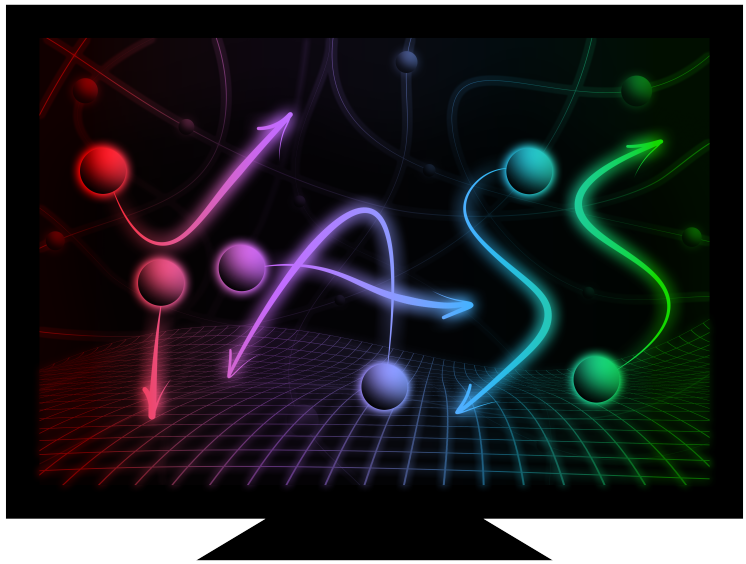
From adsorptation to dissipation: insights from computer simulations of solid H2O and CO

Ferrari, B.C.

Chapter 3

YetAnotherSimulationSuite.jl: An Atomic Simulation suite in Julia



This chapter is based on:

Ferrari, B. C., "YetAnotherSimulationSuite.jl: An Atomic Simulation Suite in Julia." Journal of Open Source Software 10.116 (2025): 9480.

Open-Source Repository Software Documentation

Throughout the PhD I have continuously developed a simulation suite in the Julia [1] programming language. The initial development of this package began with the need to speed-up molecular dynamics (MD) simulations to make incredibly long timescale simulations accesible. Although a wide variety of simulation suites (hence the name) existed, including those focused on achieving the fastest speeds (*ie.* LAMMPS [2]), the perfect combination of speed, memory efficiency, flexibility and ease-of-use did not exist.

3.1 Features

In its current state, *YetAnotherSimulationSuite.jl* (YASS) offers a comprehensive set of capabilities for molecular simulations. It can perform geometry and cell optimizations, allowing users to find the most stable structures and lattice parameters for their systems. The suite supports molecular dynamics simulations in both the NVE (constant number of particles, volume, and energy) and NVT (constant number of particles, volume, and temperature) ensembles, enabling the study of system behavior under different thermodynamic conditions. YASS also provides tools for vibrational mode analysis using the harmonic approximation, which helps in understanding the vibrational properties of molecules. Additionally, it includes functionality for velocity autocorrelation function analysis, which is useful for extracting dynamic information such as the vibrational density of states beyond the harmonic approximation. The suite can compute radial and angular distribution functions, offering insights into the structural organization of particles within a system. Furthermore, YASS comes with several built-in potentials, including TIP4P/2005f [3], SPC-F [4], and two different CO–CO potentials from the literature, specifically those developed by van Hemert *et al.* [5] and Chen *et al.* [6]. These features make YASS a flexible and extensible tool for a wide range of molecular simulation tasks.

YASS was also developed with a focus on flexibility and extensibility, making it incredibly easy to add or customize anything within the package. Every part of the simulation suite, even down to the units used in the simulations, are left flexible and extensible to allow users to customize.

3.2 Installation

YASS can be installed using the Julia package manager via:

```
pkg> add YetAnotherSimulationSuite
```

If you are more adventerous, you can consider installing YASS from GitHub. This will get updates more frequently, which gives users more features but also comes with increased chances of bugs.

```
pkg> add https://github.com/Cavenfish/YetAnotherSimulationSuite.jl
```

The Julia package manager will handle all dependencies and YASS should work straight out of the box.

3.3 Benchmarks

Julia often delivers substantial performance gains over Python for numerical and scientific code because it is JIT-compiled, type-stable, and generates native LLVM code, so well-written Julia can approach C/Fortran speeds. However, that speed comes with trade-offs: just-in-time compilation (and package precompilation) introduces startup latency, and Julia's compilation artifacts and runtime can consume more memory than lightweight Python interpreters. In practice, Julia is most advantageous for long-running, compute-intensive workflows; for short scripts or very memory-constrained environments you should weigh the startup and memory overheads or use precompilation strategies to mitigate them. In the following, gold (Au) clusters are used throughout with a Leonard-Jones potential.

3.3.1 Geometry Optimization

Geometry optimizations using the LBFGS algorithm were performed on gold clusters containing 100, 500, 1000, and 5000 atoms. The results are shown in Table 3.1.

Table 3.1: Geometry optimization benchmark results for gold clusters.

Au Atoms	Iterations	YASS Time (s)	ASE Time (s)	YASS Speedup
100	1500	11.8	8.4	0.71x
500	1500	29.6	64.0	2.16x
1000	1500	108.7	218.8	2.01x

3.3.2 Harmonic Frequencies

Harmonic frequency calculations were performed on gold clusters containing 100, 500, and 1000 atoms. The results are shown in Table 3.2.

Table 3.2: Harmonic frequency calculation benchmark results for gold clusters.

Au Atoms	YASS Time (s)	ASE Time (s)	YASS Speedup
100	11.2	3.3	0.29x
500	22.7	89.8	3.95x
1000	87.1	490.8	5.63x

3.3.3 Molecular Dynamics

Molecular dynamics simulations were performed on gold clusters containing 100, 500, and 1000 atoms for 1000, 2000 and 5000 time steps. The results are shown in Table 3.3.

Table 3.3: Molecular dynamics simulation benchmark results for gold clusters.

Au Atoms	Time Steps	YASS Time (s)	ASE Time (s)	YASS Speedup
100	1000	15.8	4.8	0.30x
100	2000	16.1	9.3	0.58x
100	5000	16.6	22.4	1.35x
500	1000	21.9	32.3	1.47x
500	2000	27.7	65.9	2.38x
500	5000	44.2	160.7	3.63x
1000	1000	35.6	79.4	2.23x
1000	2000	55.9	165.3	2.95x
1000	5000	112.5	406.1	3.61x
1000	15000	315.3	1332.8	4.22x

3.4 Memory Considerations

Currently, YASS has a roughly 1 GB memory overhead due to dependencies, buffer allocations, and compilation artifacts. This overhead is typical for Julia packages with similar functionality, but may be significant for users with limited memory resources. This overhead is static and does not scale with system size, so larger simulations will see a smaller relative impact. Future optimizations may reduce this overhead.

3.4.1 Geometry Optimization

Table 3.4: Memory usage for geometry optimization benchmarks.

Au Atoms	Iterations	Memory Usage (GB)
100	1500	1.06
500	1500	1.06
1000	1500	1.06

3.4.2 Harmonic Frequency Calculation

Table 3.5: Memory usage for harmonic frequency calculation benchmarks.

Au Atoms	Memory Usage (GB)
100	1.07
500	1.16
1000	1.41

3.4.3 Molecular Dynamics Simulation

Table 3.6: Memory usage for molecular dynamics simulation benchmarks.

Au Atoms	Time Steps	Memory Usage (GB)
100	1000	1.12
500	1000	1.15
1000	1000	1.19
5000	1000	1.55

3.5 Examples

A basic example of optimizing the geometry of a molecule in YASS:

```
using Optim
using YetAnotherSimulationSuite

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

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

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

Additional optional optimization criteria can be passed to the *opt* function (more on that in the YASS documentation). After optimizing the geometry its easy to calculate the harmonic normal modes of the molecule in YASS.

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

If anharmonicity is desired, YASS can also be used to calculate the vibrational density of states (VDOS) through the velocity autocorrelation function (VACF). This method captures anharmonicity in the vibrational modes.

```
using YetAnotherSimulationSuite

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

# Run 5 picosecond simulation with 0.1 fs timestep
traj = run(TIP4Pf(), water, 5u"ps", 0.1u"fs", NVE())

# You can also specify the start time
traj = run(TIP4Pf(), water, (5u"ps", 10u"ps"), 0.1u"fs", NVE())

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

# Configure VACF calculation
inp = vacfInps(
  vel,        # Velocity trajectories
  mas,        # Atomic masses
  1e15u"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)
```

3.A Bibliography

[1] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[2] Aidan P Thompson, H Metin Aktulga, Richard Berger, Dan S Bolintineanu, W Michael Brown, Paul S Crozier, Pieter J In't Veld, Axel Kohlmeyer, Stan G Moore, Trung Dac Nguyen, *et al.* Lammps-a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications*, 271:108171, 2022.

[3] Miguel A González and José LF Abascal. A flexible model for water based on TIP4P/2005. *The Journal of Chemical Physics*, 135(22), 2011.

[4] Kahled Toukan and Aneesur Rahman. Molecular-dynamics study of atomic motions in water. *Physical Review B*, 31(5):2643, 1985.

[5] Marc C van Hemert, Junko Takahashi, and Ewine F van Dishoeck. Molecular dynamics study of the photodesorption of CO ice. *The Journal of Physical Chemistry A*, 119(24):6354–6369, 2015.

[6] Jun Chen, Jun Li, Joel M Bowman, and Hua Guo. Energy transfer between vibrationally excited carbon monoxide based on a highly accurate six-dimensional potential energy surface. *The Journal of Chemical Physics*, 153(5):054310, 2020.