GSoC'22 Proposal
Constraint Algorithms in Molecular Dynamics
Julia : Molly.jl
Pranay Venkatesh

Student Info:

- Name : Pranay Venkatesh
- City : Chennai, India
- Email : f20191004@pilani.bits-pilani.ac.in
- Primary Language : English
- Secondary Language : Tamil
- Timezone : GMT + 5:30
- GitHub Profile : https://github.com/chemicalfiend

Current Education:

- University : Birla Institute of Technology and Science, Pilani
- Field : Double Major in Chemistry and Chemical Engineering
- Current Year : 3rd Year (6th semester overall)
- Expected Graduation : May 2024

1 Accepted PRs:

- PR #44 : For Molecular Dynamics analysis, it helps to have a velocity logger to calculate the velocity autocorrelation function, which is a good measure of "memory" for non-equilibrium systems where properties change with respect to time. In PR #44 I implemented the velocity logger and velocity autocorrelation function.
- PR #52 : MorseBond was added in PR #52, a type of specific interaction between 2 bonded atoms in a molecule.

2 Project Information:

2.1 Introduction to MD and Molly.jl :

Molecular Dynamics (MD) is an atomistic simulation technique that provides a perspective for the dynamic evolution of a system. It plots the trajectory of atoms and molecules by numerically solving equations of motion and is able to model a variety of physical systems from gases to proteins and other macromolecules. It is hence a handy simulation tool for researchers working with complex fluids, soft matter and biological macromolecules.

Molly.jl is a pure Julia package for molecular dynamics and other atomistic and molecular simulations. It has a focus on being differentiable and fast. Currently, one of the key areas of improvement apparent is the introduction of constraint algorithms into the Molly.jl system. MD software such as LAMMPS and GROMACS are able to use these algorithms to fix bond lengths and angles for various groups of molecules.

Having contributed to Molly.jl for a few months and worked with other MD software before, one area where Molly.jl lags behind other software is with regards to speed on the GPU. Hence one of my goals while working on this project is to improve Molly.jl performance on GPUs.

2.2 Constraint Algorithms:

Constraint algorithms are used to maintain distances or angles between atoms. Hence, we want to handle the dynamics of a system where some degrees of freedom are restricted while others are able to change over time. By neglecting evolution along some degrees of freedom, constraint algorithms are able to maintain high computational efficiency. Effectively all simulation packages for macromolecules use bond constraints since they allow the timestep to be increased from 0.5 fs to 2 fs. Addition of these constraint algorithms would allow for wider usage of Molly.jl for macromolecule simulations.

The standard molecular simulation problem is to find the trajectory of a system of N particles over time, following an equation of motion, for example :

$$m_i \ddot{x}_i = f_i = -\frac{\partial V}{\partial x_i}$$

Where x is one of 3N Cartesian co-ordinates of positions. V is the potential energy of the system and f is force acting on the atom.

The constraint algorithms hope to solve a problem which has to satisfy M constraint equations, in addition to the 3N equations of motion. One example of such a constraint would be fixing bond lengths between atoms :

$$\sigma_k = ||\mathbf{r}_i - \mathbf{r}_j||^2 - d_k^2 = 0$$

The constraints σ_k are functions of time and evolve along with the coordinates. Our goal is to now develop a scheme where we can find the trajectory of the system over time such that it satisfies the constraints and evolves according to the equations of motion.

Using Lagrange multiplier method, we can devise:

$$m_i \ddot{x}_i = f_i - \sum_{\alpha} \lambda_\alpha \frac{\partial \sigma_\alpha}{\partial x_i}$$

From here, we solve for the Lagrange multipliers λ. After that we can derive the trajectory of the system using :

$$\mathbf{r}(t+\Delta t) = \mathbf{r}_u(t+\Delta t) + \sum_k \frac{\lambda_k}{m_i}\frac{\partial \sigma_k}{\partial r}(\Delta t)^2$$

Where $\mathbf{r}_u$ represents the unconstrained solution.

When using constraint equations such as the σ_k given above for bond lengths, the equations generated are often large and difficult to solve analytically. Hence non-linear solvers such as Newton-Raphson method are used to numerically acquire the Lagrange multipliers together.

$$\lambda^{(l+1)} = \lambda^{(l)} - \mathbf{J}^{-1}\sigma(t+\Delta t)$$

Where $\mathbf{J}$ is the Jacobian matrix. Only a few particles contribute to each of the constraint equations, hence most terms in the Jacobian are 0, making it a sparse matrix, specifically a block matrix. This can be solved for each block at a time.

Various algorithms solve for the Lagrange multipliers in different ways and then find the constrained trajectories in different ways. Each of them have their own use cases. For example, the SETTLE algorithm solves the equations analytically, 3 constraints at a time. This is useful for rigid water molecules.

3 Project Schedule:

3.1 Phase 0 [April 15 - June 12] :

Period prior to GSoC selection and Community Bonding period.

- Familiarize usage of constraint algorithms in various MD software.
- Read the relevant papers for various constraint algorithms and their use cases.
- Discuss project specifications in great detail with mentors.
- Devise a scheme for integration of constraint algorithms into Molly.jl ecosystem.
- Learn more about GPU computing, Parallel algorithms and optimizing performance.

3.2 Phase 1 [June 13 - July 25]

Implementation of SHAKE and related constraint algorithms with documentation, unit testing and examples.

- June 13 - June 20 : Complete integration of constraints into Molly ecosystem.

- June 20 - June 27 : Implementation of SHAKE algorithm, with testing
- June 27 - July 2 : Improving performance of SHAKE. Begin implementation of related algorithms such as RATTLE, WIGGLE, etc.
- July 3 - July 20 : Complete implementations of various SHAKE-related algorithms.
- July 21 - 25 : Complete documentation, unit testing and examples for SHAKE and related algorithms. Prepare for phase 1 evaluation.

3.3 Phase 2 [July 29 - September 4]

- July 29 - August 3 : Implementation of SETTLE constraint algorithms, with documentation and examples related to rigid water molecules (a common use case of SETTLE).
- August 3 - August 10 : Implementation of LINCS (Linear Constraint Solver) and its parallelized version (P-LINCS).
- August 10 - August 17 : Documentation, testing and examples for the algorithms above.
- August 17 - August 28 : Exploring compatibility with GPU acceleration, improving performance.
- August 29 - September 4 : Final submission of code for evaluation.

4 Deliverables for Evaluation:

- Addition of constraints to the Molly.jl ecosystem.
- Implementations of various bond geometry constraint algorithms including SHAKE and related algorithms such as M-SHAKE, WIGGLE, RATTLE, P-SHAKE, etc.
- Implementations of SETTLE, LINCS and other MD constraint algorithms.
- Improving performance as well as extending GPU compatibility.
- Proper documentation and testing for the above-mentioned components.

5 Code Portfolio

The code I have written for past projects and research work are available on my github page at chemicalfiend. One notable example is the code I wrote for generating cylindrical nanotube geometries as part of an undergraduate research project, which can be found here. Another program that I am currently developing is for morphology analysis after completing a molecular dynamics simulation. It virtually performs a Grazing Incidence X-Ray Scattering analysis using a 2-D Fast Fourier Transform in Julia, based on a paper by Eric Jankowski in Boise State University. This work can be found here, although it is still buggy and incomplete. As part of a school computer science project, I created a time table management system in Java, which can be found on my pastebin here.

6 Why Molly.jl and why me?

I am interested in soft matter (in particular Organic Semiconductors and Photovoltaics) and theoretical work surrounding it. Having worked with LAMMPS and HOOMD-blue to simulate various photovoltaic systems as part of my undergraduate research work, I gained a lot of insight into how MD simulations work and became interested in parttaking actively in research surrounding MD simulations. I have also long been interested in free and open source software, which is usable by the community. Since I was interested in implementing and experimenting with simulations, I wanted to explore simple and easy-to-use existing platforms where I could test my codes. Since I was familiar with the Julia programming language, I got introduced to Molly.jl by joining the JuliaMolSim Slack forum after attending the Chemistry Birds of a Feather session at JuliaCon 2021.

7 Relevant Experience

I have experience working with MD simulations in LAMMPS and HOOMD-Blue as part of an undergraduate research project in my university. I have an intermediate level of knowledge in numerical analysis. My experience in this area comes from having taken up courses in my university, theoretical and advanced courses online and reading textbooks and papers.

I also have some prior experience working with Julia, particularly in scientific computing. I have been using Julia since 2020, where I learned how to use Symbolics.jl to help learn and practice for maths courses such as linear algebra. I have also used Julia extensively for molecular simulation in the past, particularly in post-simulation analysis.

8 Programming Experience:

- Julia - Highly proficient
- Python - Moderate proficiency
- Fortran-90 - Moderate proficiency
- Java - Proficient

- C - Beginner
- MATLAB - Beginner
- Git - Highly proficient

9 Time Commitment:

My summer vacation is between May 23 and the end of July. Since I do not differentiate between weekdays and weekends, I am able to maintain a uniform schedule throughout the work week. However, my mentors may not be available during the weekends and on some weekends I might attend a family gathering with my relatives. Taking these into account, I would be able to work around 30 hours on average during these weeks. I prefer to work later during the day (from 4PM to 2AM) and this should help communicate with my mentor in a different timezone.

My semester begins again in the middle of August, at which point I will be able to work around 15-18 hours a week, mostly on weekends.