Teaching Materials Science Using a Client-Side Force-Directed Graph Framework

Georgios Varnavides*
Department of Materials Science and Engineering
Massachusetts Institute of Technology, Cambridge, MA 02139, USA

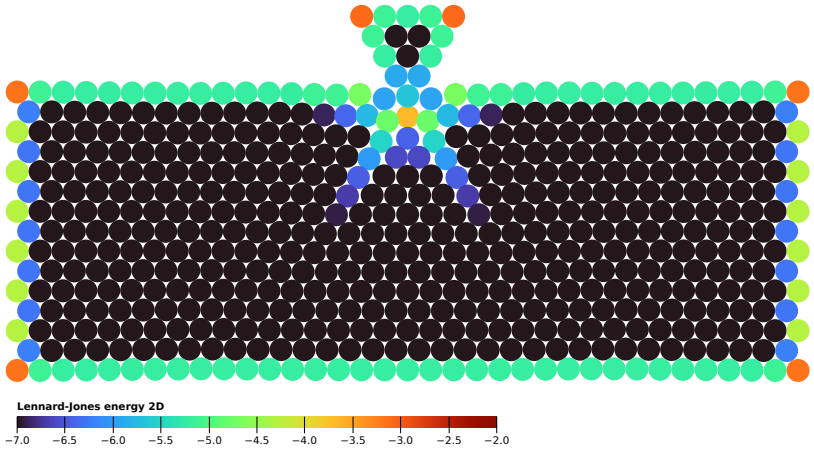

Figure 1: Snapshot of an indentation experiment simulation using the developed framework. Triangular indentation probe is made up of rigid particles moving at a constant velocity. The thin 'beam' is made up of hexagonally-closed packed particles interacting with each other and the indentation probe using a normalized Lennard-Jones interatomic potential. The beam's left and right edges are held fixed. All particles are coloured according to their Lennard-Jones potential. Lines of higher energy concentration signal the onset of dislocations forming, allowing the beam to deform plastically.

Abstract

Recent developments in just-in-time compilers for dynamic languages have significantly improved the performance of web-based physics engines. In conjunction with the ubiquity of interactive visualizations and the level of maturity of Document Object Model manipulation libraries, this provides a unique opportunity to leverage such client-side physics engines for education. Here, we present an extension module to a popular force-directed graph layout library which uses velocity Verlet integration to teach materials science concepts based on the Lennard Jones interatomic potential. We find the ability to seamlessly integrate our numerical integrator with common interaction techniques, such as clicking, dragging, and brushing to improve the pedagogy of the simulations.

Index Terms: Force-directed graphs — Visualization systems and education tools;

1 Introduction

Force-directed graph layouts describe a broad class of algorithms which attempt to visualize the relationships between nodes in undirected graphs and hierarchical trees, via means of a physical simulation [3, 8, 10]. Such graph layouts have recently enjoyed increased popularity in the domain of information visualization, given the rise of interactive visualizations and advancements in just-in-time compilers for Javascript [4, 5]. Further, the level of maturity, and integration with web-standards, of various Document Object Model (DOM) manipulation libraries, such as D3.js [1], suggest that physical simulations, even beyond the scope of typical graph layouts, can be carried out dynamically and interactively on the client-side. Here, we explore the merits of this idea in the context of undergraduate education. In particular, we extend one such popular force-directed graph implementation, namely D3.js's *d3-force* module, to perform energy-conserving velocity Verlet integration and implement new forces commonly used in molecular dynamics simulations of condensed matter systems. We then use this module to develop an interactive narrative [7], illustrating various core concepts in materials science, such as the formation of dislocations during plastic deformation (Fig. 1).

*e-mail: gvarnavi@mit.edu

2 Related Work

Despite using physical simulations at their core, force-directed graph algorithms solve a specific problem with notably different objectives and priorities than molecular dynamics. For example, contrary to molecular dynamics simulations, accuracy is often not the primary concern and instead visual clarity and stability are prioritized during design decisions [6]. As such, the work presented here is best compared against both force-directed graph algorithms, but also client-side dynamic physics engines.

2.1 Force-Directed Graph Layout Frameworks

While there exist multiple force-directed graph layout implementations, here we focus on some well-suited for dynamic and interactive exploration of graph data. One popular such framework is included as a core module in D3.js, aptly called *d3-force*. It implements a simplified velocity Verlet numerical integrator, with a unit mass and time-step (see Sect. 3), converging to a stable configuration using

'temperature' annealing. The module presented here is very closely related to *d3-force*, and in-fact builds on *d3-force* source code to address two short-comings (in the context of molecular dynamics):

1. Link constraints in *d3-force* use iterative-relaxation (effectively peeking-ahead at node's anticipated positions) to increase stability; this has consequences in energy-conservation.

2. The unit-time step assumption used in *d3-force*, followed by a simplification of the equations of motion leads to the numerical integrator accumulating error.

An alternative to *d3-force*, albeit still often used in conjunction with d3.js for displaying, is cola.js or 'WebCoLa', which instead uses constraint-based optimization techniques to converge to a local optimum. Such constraint optimization often leads to higher quality layouts, with increased stability. However, as it is solving an optimization problem – as opposed to solving some equations of motion – it is less applicable to molecular dynamics simulations.

2.2 Client-Side Dynamic Physics Engines

Perhaps a more natural comparison to the work presented here on dynamic molecular dynamics simulations are client-side Javascript physics engines. Among these, *matter.js* and *planck.js* are most closely-related as they also use a Verlet numerical integrator to solve the equations of motion [2, 9]. A great deal of attention has been spent in making *matter.js* and *planck.js* work with client-side applications in mind with the resulting simulations being seamlessly interactive as-well as performant. In contrast to d3.js, it is built on top of HTML5 and thus works by manipulating the canvas – as opposed to scalable vector graphics.

3 Methods

Molecular dynamics simulations numerically integrate Newton's equations of motions for an ensemble of forces acting on particles, by taking small finite time-steps. At each set of particle positions, the forces acting on the particles as-well as their velocities are used to predict the new particle positions a finite time step away. An important family of numerical integrators used in molecular dynamics simulations were re-discovered by L. Verlet in 1967 [11].

3.1 Position Verlet Integration

Starting from Newton's equations of motion

$$\vec{F} = m\,a\left(\vec{x}(t)\right) = m\,\ddot{\vec{x}}(t), \tag{1}$$

where F and m are the force and mass experienced by a particular particle, and $a\left(\vec{x}(t)\right)$ is the instantaneous acceleration of particle particle at position $\vec{x}(t)$ at time t. The second-order differential equation can be solved with appropriate initial conditions $\vec{x}(t_0) = \vec{x}_0$ and $\dot{\vec{x}}(t_0) = \vec{v}_0$ at time $t_n = t_0 + n\Delta t$ using the recurrence relation:

$$\begin{aligned}
\vec{x}_1 &= \vec{x}_0 + \vec{v}_0\Delta t + a\left(\vec{x}_0\right)\Delta t^2 + \mathcal{O}(\Delta t^4) \\
\vec{x}_{n+1} &= 2\vec{x}_n - \vec{x}_{n-1} + a\left(\vec{x}_n\right)\Delta t^2 + \mathcal{O}(\Delta t^4) \\
\vec{v}_n &= \frac{1}{2\Delta t}\left(\vec{x}_{n+1} - \vec{x}_{n-1}\right) + \mathcal{O}(\Delta t^4)
\end{aligned} \tag{2}$$

While Equation 2 is fairly simple to implement and is accurate up to fourth-order in the time-step Δt, it requires one stores (at-least) three consecutive particle positions and evaluates the velocities one time-step behind the positions.

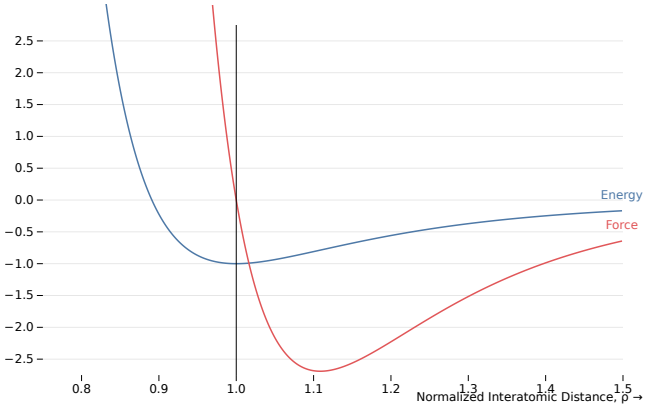

Figure 2: Normalized Lennard Jones interatomic potential (blue) and its negative gradient (red), illustrating the force experienced by a pair of particles separated a normalized distance ρ apart.

3.2 Velocity Verlet Integration

The Velocity Verlet variation addresses those short-comings by evaluating both the positions and velocities at the same time:

$$\begin{aligned}
\vec{x}_{n+1} &= \vec{x}_n + \vec{v}_n\Delta t + \frac{a\left(\vec{x}_n\right)\Delta t^2}{2} + \mathcal{O}(\Delta t^3) \\
\vec{v}_{n+1} &= \vec{v}_n + \frac{\Delta t}{2}\left(a\left(\vec{x}_{n+1}\right) + a\left(\vec{x}_n\right)\right) + \mathcal{O}(\Delta t^3)
\end{aligned} \tag{3}$$

In particular, we evaluate Equation 3 in three steps:

$$\begin{aligned}
\vec{v}_{n+1/2} &= \vec{v}_n + \frac{\Delta t}{2}a\left(\vec{x}_n\right) \\
\vec{x}_{n+1} &= \vec{x}_n + \vec{v}_{n+1/2}\Delta t \\
\vec{v}_{n+1} &= \vec{v}_{n+1/2} + \frac{\Delta t}{2}a\left(\vec{x}_{n+1}\right)
\end{aligned} \tag{4}$$

3.3 Central-Body Interatomic Potentials

Central-force potentials are a family of rotationally-invariant potentials which only depend on the distance r between two particles. Here, we focus on the consequences of the Lennard-Jones potential, although we note the formalism developed here can accept general Javascript functions as interatomic potential inputs. The Lennard-Jones potential is a model for a particle's potential energy when it is separated from another particle. The model is simple – particles attract each other when they are far apart and repel when they are very near. All of the physics and its consequences derive from the shape of the potential, given by Fig. 2. When the distance is small, the potential energy decreases with increasing distance. The force is positive and repulsive because force is minus the gradient of potential energy. When the distance is large, the potential energy slope is positive and so the force is negative: the particles attract.

Mathematically, the Lennard-Jones potential is given by a linear combination of two terms: An attractive term that goes to zero as $-\frac{1}{r^6}$ as $r \to \infty$; A repulsive term that goes to infinity as $\frac{1}{r^{12}}$ as $r \to 0$

$$U_{ljp} = \frac{a}{r^{12}} + \frac{b}{r^6}, \tag{5}$$

with a and b being material-properties we could extract using experimental observations. To understand the dynamics of materials with this family of interatomic potentials however, we non-dimensionalize Equation 5 by introducing an equilibrium distance

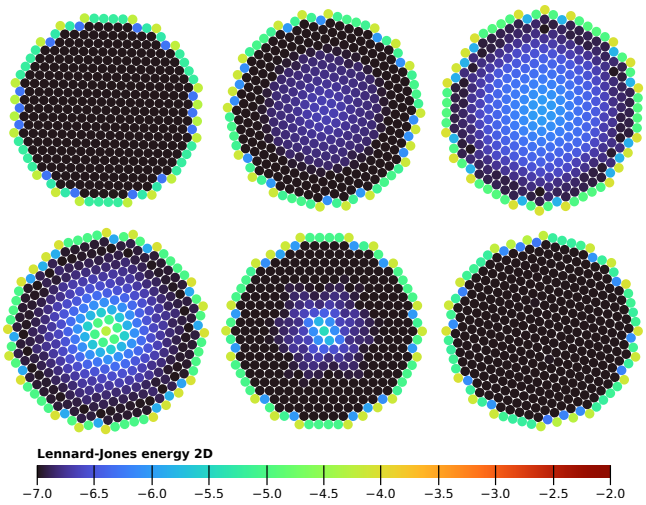

Figure 3: Series of snapshots during the simulation of a rotating nanoparticle. The initial clockwise rotation, results in the formation of compression waves throughout the particle, due to the effect of surface tension form the outer-most (higher-energy) atoms.

$\rho = r/r_{min}$ and dividing the potential by the energy at that equilibrium distance:

$$\tilde{U}_{ljp} = \frac{1}{\rho^{12}} - \frac{2}{\rho^{6}}. \tag{6}$$

3.4 Interactivity

We use *Observablehq*'s reactive notebook environment to deploy the interactive narratives, as we find it is effective in intertwining textual explanations with multiple interactive visualizations. Interactivity is achieved using five main techniques:

- Dynamic molecular dynamics simulations
- Precomputed molecular dynamics animations
- Reactive inputs (buttons, sliders, toggles)
- Brushing and linking of visualizations
- Dragging/dropping atoms to influence positions directly

4 Results

The interactive narrative steps through a broad range of materials science concepts which can all be derived from the Lennard-Jones potential using molecular dynamics simulations, including:

- Lattice vibrations and dispersion relations
- Physical origins of surface energy
- Lattice instabilities
- Compression waves
- Materials deformation during indentation experiments

4.1 Compression waves

Here, we showcase one example concept as a means of demonstration. The exploration starts with a spinning hexagonally-closed packed nanoparticle. The atoms are colored according to their Lennard-Jones energy, with atoms deep inside the nanoparticle ('bulk') having a lower energy, due to having six nearest-neighbors each. By contrast, the atoms at the surface of the nanoparticle have fewer nearest-neighbors, leading to more dangling (or unsatisfied) bonds and thus have higher energy. As the particle spins clockwise, the competition between this surface tension and the centrifugal force arising from the atoms' inertia leads to the creation of compression waves which 'pulsate' through the particle as the simulation progresses. The users can further adjust the initial rotation speed and tip this balance one way or another, leading the particle to 'rip' apart. Finally, users can drag individual atoms as the simulation is running and experience how 'sensitive' atoms are to such perturbations.

5 Discussion

This paper introduces an extension module to a popular force-directed graph layout library, to accurately and interactively perform molecular dynamics simulations on the web. This has advantages in pedagogy, in part due to the ease-of-sharing afforded by the reactive notebook platform used but also, due to the interactivity and real-time computation of these simulations. This in turn is afforded due to advancements in just-in-time compilers, making Javascript one of the fastest dynamic languages, and the level of maturity DOM-manipulation libraries such as D3.js have in information visualization.

6 Future Work

The molecular dynamics extension *d3-force* has a number of obvious areas of future development. First, while the general central-body forces handled by the module are reasonable approximations to many materials, they are, by construction, spherically symmetric. This is not very realistic in crystals. In the future, we hope to extend these pair interactions to account for a more general class of interatomic potentials called bond-order potentials, such as the three-body Tersoff potential. Second, the module currently implements the Barnes-Hut algorithm to reduce to the computational complexity of the molecular dynamics simulation from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$. Unfortunately, the performance gains in condensed matter systems, i.e. close-packed particles, are reduced since the threshold parameter (*theta*) has to be chosen to be quite low for reasonable accuracy. In the future, we hope to investigate other methods to improve the perfomance of the algorithm.

Acknowledgments

The authors gratefully acknowledge Craig Carter and Amina Matt of the Massachusetts Institute of Technology for fruitful discussions and the initial co-authoring and editing of the molecular dynamics simulations.

References

[1] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.

[2] L. Brummitt. Matter.js: a 2d rigid body physics engine for the web. `https://github.com/liabru/matter-js`, 2014.

[3] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, vol.42:149–160, 1984.

[4] A. Gal, B. Eich, M. Shaver, D. Anderson, D. Mandelin, M. R. Haghighat, B. Kaplan, G. Hoare, B. Zbarsky, J. Orendorff, J. Ruderman, E. W. Smith, R. Reitmaier, M. Bebenita, M. Chang, and M. Franz. Trace-based just-in-time type specialization for dynamic languages. *SIGPLAN Not.*, 44(6):465–478, June 2009.

[5] B. Hackett and S.-y. Guo. Fast and precise hybrid type inference for javascript. *SIGPLAN Not.*, 47(6):239–250, June 2012.

[6] T. Jakobsen. Advanced character physics. In *IN PROCEEDINGS OF THE GAME DEVELOPERS CONFERENCE 2001*, page 19, 2001.

[7] D. E. Knuth. Literate Programming. *The Computer Journal*, 27(2):97–111, 01 1984.

[8] S. G. Kobourov. Spring Embedders and Force Directed Graph Drawing Algorithms. *arXiv e-prints*, page arXiv:1201.3011, Jan. 2012.

[9] A. Shakiba. Matter.js: a 2d rigid body physics engine for the web. `https://github.com/shakiba/planck.js`, 2017.

[10] W. T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, s3-13(1):743–767, 1963.

[11] L. Verlet. Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Phys. Rev.*, 159:98–103, Jul 1967.