EMPIRICAL FORCE FIELDS IN JULIA FOR MOLECULAR SIMULATION*

NAVEEN ARUNACHALAM†

Abstract. Molecular simulation is an important means of understanding physical systems at the atomic scale. Molecular mechanics (MM) simulations use extended ball-and-spring models to simulate atoms in a chemical environment, and the force constants of these models are determined by parameter sets known as force fields. This paper outlines the implementation of the Universal Force Field in Julia (UFF.jl), which provides force field parameters for a wide range of the periodic table, including transition metals. The UFF.jl package is then used to exhaustively generate force field parameters for subsets of the periodic table, and its performance is analyzed for various system sizes in both parallel and single-threaded environments. Then, the package is used to optimize the structure of butane using global optimization and local optimization, the latter of which uses forward-mode automatic differentiation of the UFF energy function.

Key words. Molecular Mechanics, Computational Chemistry, Julia, Parallel Computing

MSC codes. 92E10

1. Introduction.

Research and development in chemistry can often be expensive or hazardous due to labor costs, chemical hazards, and conditions such as high pressures and temperatures. In the past, chemists have turned to chemists have turned to physical simulation of matter at the atomic scale as an alternative to performing experiments in a laboratory setting. The adoption of so-called in silico techniques has led to the development high throughput virtual screening (HTVS), which allows the chemist to filter a large library of molecules for promising leads by performing separate in silico chemical experiments or characterizations for each molecule in the library. As suggested by Figure 1, these techniques have led to the discovery of new therapeutics [5] and materials [7], and will result in faster generation of novel matter in the future as computational tools and modeling techniques continue to improve over time.

In practice, physical simulations of matter suffer from a tradeoff between time efficiency, accuracy, and system size given a fixed set of computational resources. To resolve this issue, a hierarchy of methods have been developed for systems of different scales. For example, single atoms and small molecules can be simulated to a high degree of chemical accuracy using quantum mechanical methods such as density functional theory (DFT) and coupled cluster (CC) theory; these methods are numerically expensive as they attempt to directly solve the Schrödinger equation for multiple bodies. On the other hand, the largest length and time scales can be simulated using classical mechanics; for example, a rocket's orbit in space can be simulated accurately using classical laws of motion. However, at intermediate length and time scales (nanometers to micrometers; picoseconds to milliseconds), the ideal choice between quantum mechanical and classical simulation becomes unclear. Ideally, we would like to perform physical simulations taking advantage of both, while taking maximum advantage of the computing resources available. To do this, we turn to molecular mechanics (MM) methods implemented in a parallel computing environment.

Molecular mechanics (MM) methods are a class of simulation methods that model molecules as classical systems; for example, one possible MM approach for simulating

*Project for 18.337: Parallel Computing and Scientific Machine Learning

†MIT, Cambridge, MA (narunach@mit.edu, http://naveenarun.github.io).

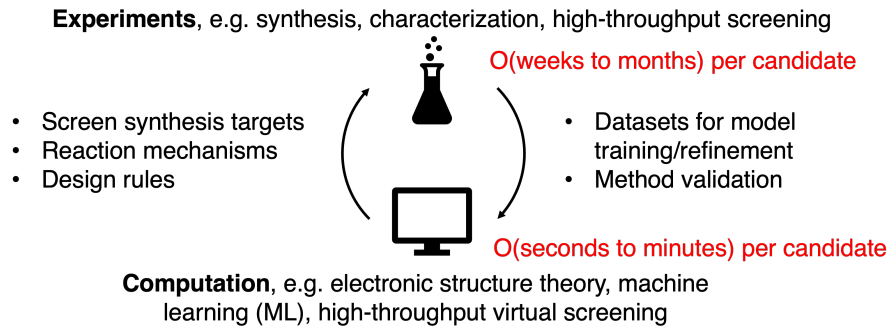

FIG. 1. *Relationship between computation and experiment.*

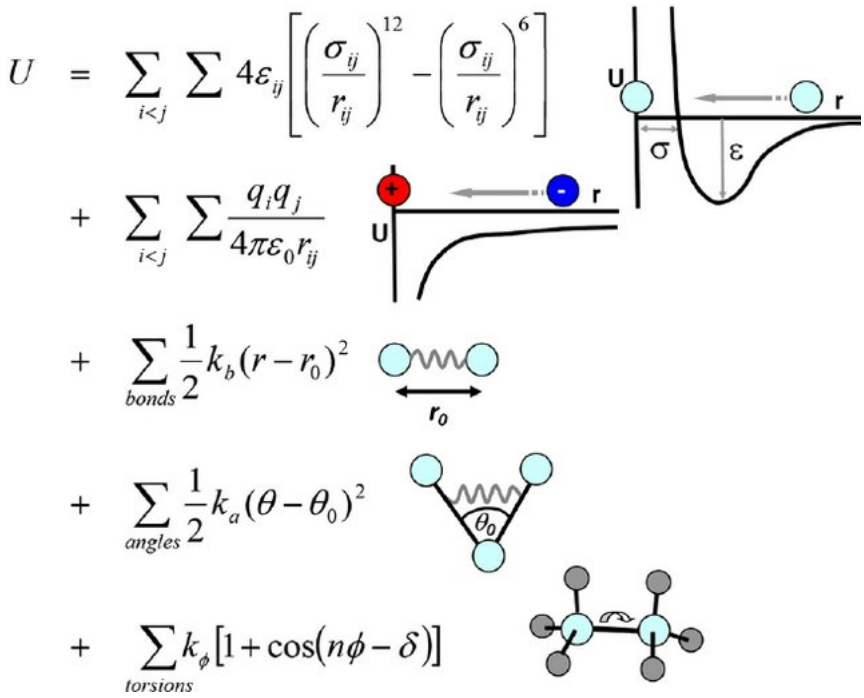

FIG. 2. *The forms of various terms included within force fields [6].*

1000 water molecules for 1 ns might involve setting up 1000 ball-and-spring structures with a specified spring constant for the O-H bond and another specified spring constant for the O-H-O angle, and then allowing the structures to interact with each other at a distance by representing long-range interaction energies with the Lennard-Jones potential (which tends to accurately describe the behavior of noble gases). Because all the forces in the system are specified by the imposed potential energy functions (some examples are provided in Figure 2), the positions of the atoms in the system can then be integrated over time, e.g. for 1000 timesteps of 1 ps each. Because these tasks (setting up long-distance potentials, imposing bond-bond and angle potentials, integrating positions over time) are so common in molecular mechanics simulations, there exist dozens of software suites that efficiently perform these tasks (e.g. LAMMPS

[12] and NAMD [8]), each of which are tailored to different application domains (e.g. inorganic vs organic, small-molecule vs. large-molecule) and architectures (e.g. GPU vs non-GPU).

2. Problem Statement. Although many molecular mechanics software suites provide a wide range of powerful tools, many could benefit from being scriptable, open-source, and taking fuller advantage of CPU and/or GPU parallelism. The CESMIX software suite [1], combined with JuliaMolSim [2], attempt to address this need by providing molecular simulation tools that can be directly set up and called within Julia in a parallel environment. Currently, the two codebases provide tools for both quantum mechanical and classical simulations of atoms and molecules. For MM simulations, Molly.jl can be used to load in OpenMM force field files [3] (e.g. for CHARMM, AMBER) or GROMACS force field files (e.g. OPLS [4]) to specify various intra- and intermolecular potential energy functions. However, it is currently unclear how one should introduce a non-OpenMM force field (or a learned force field) into either suite; on the one hand, the CESMIX suite tends to be specialized in atomistic simulations, while on the other hand, JuliaMolSim provides rigid tooling for reading in force field parameters. Thus, the focus of this project is to introduce a standalone tool called UFF.jl to interface with both suites to represent and load force field parameters into MM simulations.

The need for such a class becomes apparent when working with the universal force field (UFF) [9], which is not associated with an OpenMM or GROMACS force field file. UFF is an empirical potential that provides parameters for bond strengths, long-distance interactions, angle forces, torsional barriers, and more across the entire periodic table. Because it provides parameters across the entire periodic table, it is useful for the simulation of metal-containing molecules such as coordination complexes (like the one shown in Figure 3, which consist of transition metals surrounded by organic ligands. In HTVS environments, the force field can be used to generate a guess of a molecule's 3D structure before a higher level of theory such as DFT is used to refine the 3D structure. Because UFF is useful on a practical level for many simulation tasks, implementations can be found in a wide range of existing software packages such as LAMMPS and NAMD, but as of March 2022 the only UFF implementation in the Julia ecosystem is in PorousMaterials.jl [11], with support limited to long-range atomic interactions. The lack of molecular implementations of UFF is likely due to the absence of tooling for integrating custom force field parameters into Julia's MM ecosystem. This project aims to address this issue by providing an implementation of UFF that can be directly loaded into a parent molecular mechanics framework, and allowing users to perform UFF and custom force field simulations in a way that takes advantage of the straightforward differentiability and parallelism provided by the Julia environment.

3. Implementation Details. Within UFF, the potential energy of a molecule is represented as a sum of two-body, three-body, and four-body interactions, some of which are bonded interactions and some of which are nonbonded. The potential energy can be summarized by the following formula:

$$E = E_R + E_\theta + E_\phi + E_\omega + E_{vdw} + E_{el}$$

E_R represents the bond-stretching interaction; E_θ, E_ϕ, and E_ω represent angle bending, dihedral angle torsion, and inversion respectively. Two nonbonded interactions, E_{vdw} and E_{el}, are also included, and they represent the van der Waals inter-

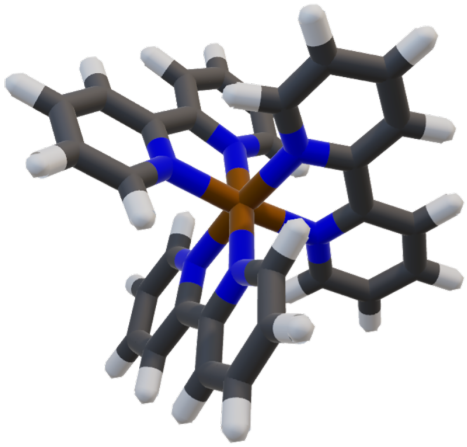

FIG. 3. *Example of a transition metal complex, which contains a central transition metal surrounded by organic ligands.*

Symbol	Description	Units
r_I	UFF atomic radius	Å
θ_0	equilibrium bond angle associated with atom	degrees
X_I	atomic van der Waals radius	Å
D_I	atomic van der Waals energy	kcal/mol
ζ	van der Waals shape term used to derive x_I and D_I	unitless
Z_I	effective atomic charge	charge
V_I	sp^3 torsional barrier coefficient	kcal/mol
U_I	sp^2 torsional barrier coefficient	kcal/mol
χ	GMP [10] electronegativity correction parameter	unitless

TABLE 1
Atomic parameters provided by UFF.

action and electrostatic interactions respectively. All terms in this expression can be derived from a set of 9 parameters per atom, summarized in the table above. In this work, we implement the two-, three-, and four-body bonded terms E_R, E_θ, and E_ϕ, as well as the two-body nonbonded terms E_{vdw} and E_{el}.

3.1. Bond-Stretching. The potential function for the bond-stretching interaction between atoms I and J is expressed as a harmonic oscillator by the formula

$$E_R = \frac{1}{2} k_{IJ} (r - r_{IJ})^2$$

where

$$r_{IJ} = r_I + r_J + r_{BO} + r_{EN}$$

for which the bond order correction (r_{BO}) is

$$r_{BO} = -\lambda (r_I + r_J) \ln(n)$$

and the electronegativity correction (r_{EN}) is

$$r_{EN} = r_I r_J \frac{(\sqrt{\chi_I} - \sqrt{\chi_J})^2}{\chi_I r_I + \chi_J r_J}$$

In the above expressions, λ is a constant, $\lambda = 0.1332$ (empirically determined from propane/propene/propyne) and n is a bond order.

The force constant k_{IJ} can be evaluated using the empirical expression

$$k_{IJ} = 664.12 \frac{Z_I Z_J}{r_{IJ}^3}$$

3.2. Angular Bending. Next, the angular bending term for atoms I, J, and K, where J is the central atom, can be calculated as

$$E_\theta = K_{IJK}(C_0 + C_1 \cos\theta + C_2 \cos 2\theta)$$

where

$$C_2 = \frac{1}{4\sin^2\theta_0}$$
$$C_1 = -4C_2 \cos\theta_0$$
$$C_0 = C_2(2\cos^2\theta_0 + 1)$$

for general nonlinear bonds. However, for linear, trigonal-planar, square-planar, and octahedral geometries, $n = 1, 2, 3, 4$ should be used respectively in the following formula instead:

$$E_\theta = \frac{K_{IJK}}{n^2}(1 - \cos n\theta)$$

In both expressions, K_{IJK} is defined as

$$K_{IJK} = \beta \frac{Z_I Z_K}{r_{IK}^5} r_{IJ} r_{JK} (3 r_{IJ} r_{JK} (1 - \cos^2\theta_0) - r_{IK}^2 \cos\theta_0)$$

where

$$r_{IK}^2 = r_{IJ}^2 + r_{JK}^2 - 2 r_{IJ} r_{JK} \cos\theta$$

and

$$\beta = \frac{664.12}{r_{IJ} r_{JK}}$$

3.3. Dihedral Torsion. The torsional term E_ϕ is calculated as

$$E_\phi = \frac{1}{2} V_\phi (1 - \cos n\phi_0 \cos n\phi)$$

where n and ϕ_0 vary based on the nature of the central two atoms (for example, for an $sp^3 - sp^3$ bond, $n = 3$ and $\phi_0 = 180°$. For two sp^3 centers, the torsional barrier V_ϕ is

$$V_{sp^3} = \sqrt{V_J V_K}$$

whereas the barrier for a bond involving an sp^2 center is

$$V_{sp^2} = 5\sqrt{U_J U_K}(1 + 4.18 \ln BO_{JK})$$

where BO_{JK} is the bond order between atoms J and K.

3.4. Inversion. To calculate the inversion term for atom I bonded to atoms J, K, and L, the following formula can be used:

$$E_\omega = K_{IJKL}(C_0 + C_1 \cos\omega_{IJKL} + C_2 \cos 2\omega_{IJKL})$$

where ω_{IJKL} is the angle between IL and the IJK plane.

The original UFF paper does not provide much guidance on how to derive K_{IJKL}, and specifies that for atom types other than C, N, O, P, As, Sb, and Bi, the inversion term is 0. Thus, we omit the inversion term for the time being.

3.5. van der Waals Interaction. The form of E_{vdw} is the Lennard-Jones 12-6 potential,

$$E_{vdw} = D_{IJ}\left(-2\left(\frac{x_{IJ}}{x}\right)^6 + \left(\frac{x_{IJ}}{x}\right)^{12}\right)$$

where x is the distance between two atoms. In this formula, x_{IJ} and D_{IJ} are calculated as the geometric means of their atomic counterparts, i.e.

$$x_{IJ} = \sqrt{x_I x_J}$$
$$D_{IJ} = \sqrt{D_I D_J}$$

In the UFF formulation, the terms in the 12-6 potential are derived by matching ζ terms in the expression

$$E_{vdw} = \left(D_{IJ}\left(\frac{6}{\zeta - 6}\right)e^{\zeta}\right)e^{-\zeta(x/x_{IJ})} - \frac{\left(D_{IJ}\left(\frac{\zeta}{\zeta-6}\right)x_{IJ}^6\right)}{x^6}$$

to the form of the 12-6 potential.

3.6. Electrostatic Interactions. The electrostatic contribution to the energy is expressed as

$$E_{el} = 332.0637\frac{Q_I Q_J}{\epsilon R_{IJ}}$$

where ϵ is 1 and R_{IJ} is the distance between atoms I and J.

Note that in UFF, both types of nonbonded interactions (van der Waals and electrostatic) are excluded for atoms that are up to 2 bonds away from each other.

The functional forms of the bonded potentials are shown in Figure 4.

4. Results and Discussion. A thorough performance analysis of the code was performed to determine the efficiency and scaling behavior of parameter generation. Two types of scaling were studied: system size, i.e. the number of unique atom types and degree of parallelization. Additionally, the speed of empirical simulation was tested relative to degree of parallelization.

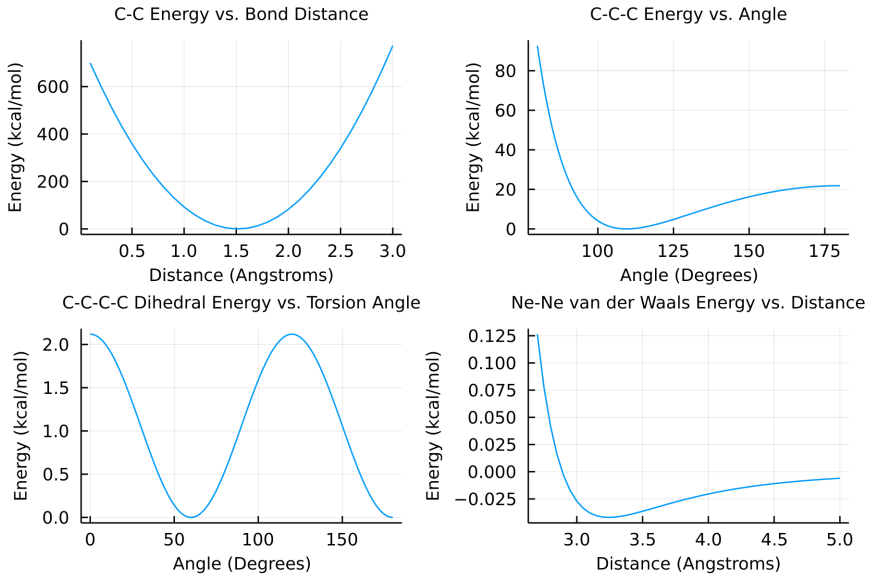

FIG. 4. *Calculated forms of the various potentials provided by UFF*

4.1. Forms of UFF Potentials. Several UFF potentials were plotted to ensure that results were chemically reasonable. The results are shown below.

As expected, the optimal C-C bond distance is approximately 1.5 Angstroms, the optimal C-C-C angle is about 109.5 degrees, and the optimal dihedral angles for a C-C-C-C chain are either 60 degrees or 180 degrees. Additionally, the van der Waals potential energy surface for Ne atoms has the expected functional form. These results are chemically reasonable, and demonstrate that the implementation can be meaningfully used for optimizing chemical structures.

4.2. Performance Analysis of Parameter Generation Time vs. System Size. For a system containing N atoms, there are $N(N+1)$ bond-stretching parameters to generate ($\binom{N+2-1}{2}$ unique pairs, each of which requires 2 parameters: a force constant and an equilibrium bond distance), $2N(N^2-N+2)$ angular bending parameters $(3\binom{N}{3}+2\binom{N}{2}+N$ ways to create a unique chain of 3 atoms, and for each unique 3-atom bond there are three Fourier coefficients and one force constant), and $\frac{N(N+1)}{2}$ dihedral torsion parameters (one barrier coefficient per central pair of atoms). Using similar reasoning, we find that there are $N(N+1)$ van der Waals parameters to generate and $\frac{N(N+1)}{2}$ electrostatic parameters to generate.

Thus, the total number of fixed parameters to generate is

$$\underbrace{N(N+1)}_{\text{bond-stretching}} + \underbrace{2N(N^2-N+2)}_{\text{angle-bending}} + \underbrace{\frac{N(N+1)}{2}}_{\text{torsional}} + \underbrace{N(N+1)}_{\text{van der Waals}} + \underbrace{\frac{N(N+1)}{2}}_{\text{electrostatic}}$$

$$= 2N^3 + N^2 + 7N$$

$$\approx O(N^3)$$

Figure 5 illustrates the effect of system size (i.e. number of unique atom types) on performance.

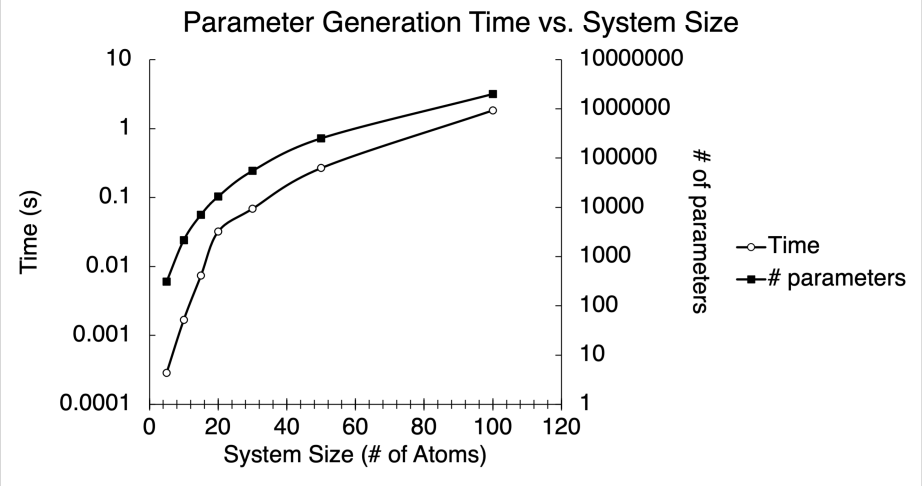

FIG. 5. *Plot illustrating how the time needed to generate a full set of parameters for N unique atom types scales with N.*

The time needed to generate all parameters for N unique atom types scales approximately as N^3. Parameter generation does not exceed 1 second for system sizes under about 90. The usefulness of Julia's performance is illustrated here by the fact that about 2 million parameters were generated in about 1.8 seconds for the system size of 100 - especially considering the many floating point operations involved in the calculation of each parameter and the fact that the code ran on a single thread on a Macbook. In order to explore how far the code's performance can be enhanced, a study of performance under multithreading was performed as well, as described in the next section.

4.3. Performance Analysis of Parameter Generation Time vs. Degree of Parallelization. All parameters were generated for system sizes ranging from 5 to 128 using both a threaded and nonthreaded implementation. The results are shown in Figure 6.

From these results, it is clear that for small system sizes, the overhead of creating threads and allocating tasks exceeds to time needed to actually calculate all parameters, meaning that for systems with up to about 25 unique atom types, it is preferable to generate parameters in a single thread. However, for larger system sizes, there are significant benefits to parallelism: for a system with 128 unique atom types, performing tasks on 4 threads results in a 2.6 times speedup, indicating that the time to calculate parameters far exceeds the time needed to create and manage threads.

Thus, for large system sizes, it is useful to parallelize parameter generation; however for small system sizes, implementing parallelization is not as important and can even be detrimental to performance. System sizes of about 25 unique atoms perform equally well under threaded and non-threaded implementations.

4.4. Local Optimization of Butane using Forward-Mode Automatic Differentiation. Forward-mode automatic differentiation (AD) was employed to generate a local minimum structure for butane. The optimized structure from Avogadro was used as an initial structure, and Julia's ForwardDiff package was used to generate gradients for the length-12 coordinate vector (4 atoms with 3 coordinates per

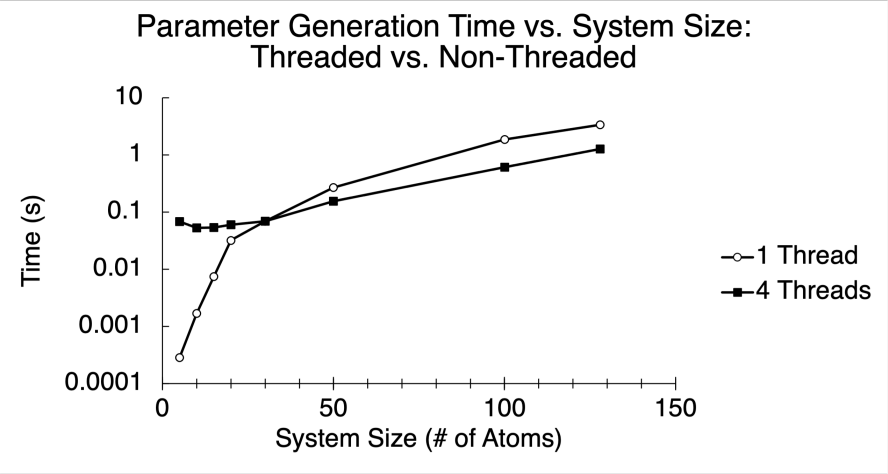

FIG. 6. *Performance of UFF.jl for generating full parameter sets in parallel and non-parallel environments.*

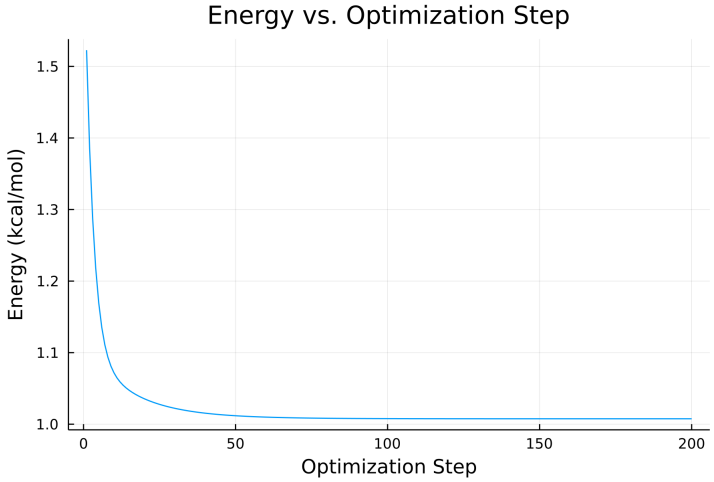

FIG. 7. *Optimization trajectory for butane using forward-mode automatic differentiation to obtain gradients.*

atom). The result was optimization to a structure with an energy of 1.01 kcal/mol, which is lower than the original energy of 1.52 kcal/mol. The results of the optimization trajectory are shown in Figure 7.

4.5. Global Optimization of Butane using Optim.jl. In order to test the implementation of UFF in Julia, a test case was performed where butane was optimized against its UFF energy. A planar input structure was drawn in Avogadro and optimized within the program; both structures can be seen in Figure 8. The structure optimized by Avogadro settled to a local minimum with a dihedral angle of 60 degrees; however, a global minimum of 180 degrees exists because the van der Waals interaction causes a repulsive force between the first and fourth carbon. Optim.jl was used to generate a globally optimal structure based on the coordinates of the four carbons, resulting in a planar molecule with a dihedral angle of 180 degress. Thus, the Julia

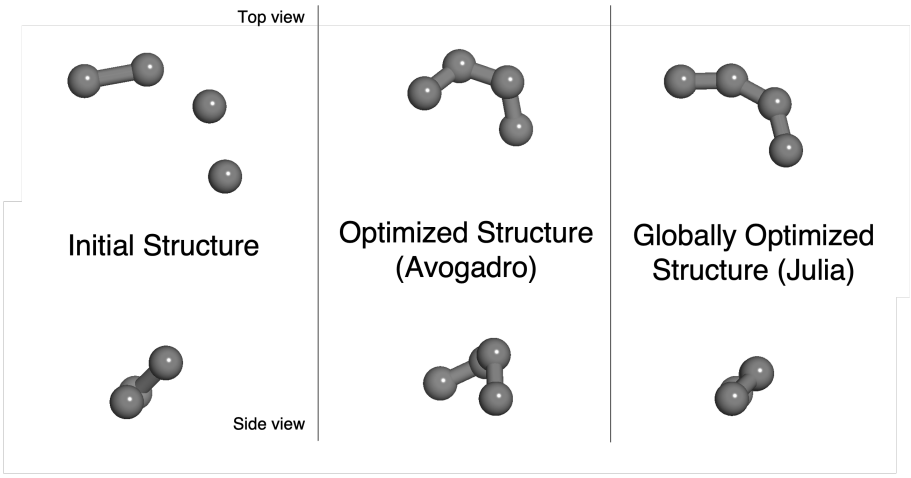

FIG. 8. *Structures of butane before and after optimization.*

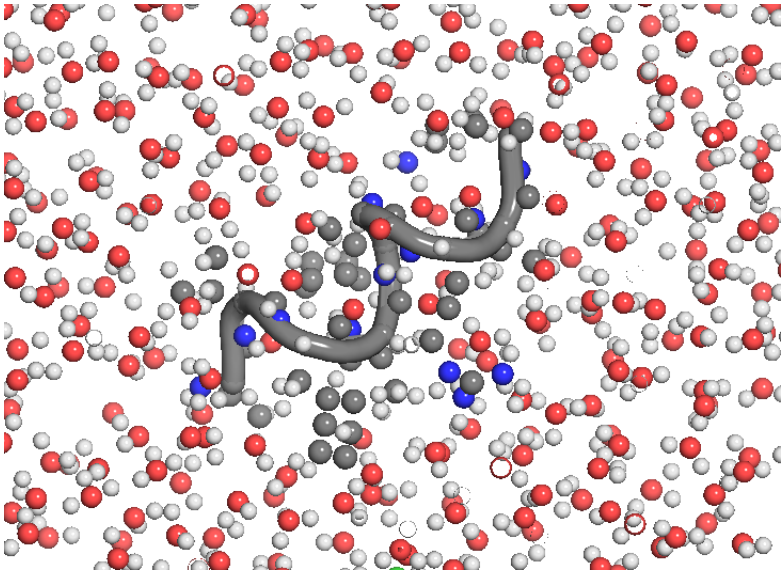

FIG. 9. *Example of a larger-scale simulation involving a protein in H_2O solvent.*

implementation was able to successfully find an optimal structure for butane. Within a structured molecular mechanics framework, therefore, the implementation can be employed for large-scale simulations with many more atoms and atom types.

4.6. Future Work. The next step for this implementation is to integrate it into the CESMIX suite (or any other molecular simulation package for Julia) so that the UFF parameters can be used in large-scale simulations. For example, in larger systems, such as the solvated protein shown in Figure 9, UFF parameters can be calculated for all chains of atoms present, and a simulation package can decide which sets of atoms should be included in energy calculations.

Additionally, it would be of interest to employ UFF paramaters in a distributed

computing environment, for example by dividing the energy calculation tasks for various sets of atoms among different nodes in a computer cluster. These are tasks that a wrapper package should handle, agnostic of the underlying force field.

The UFF force field comes with many exceptions and edge cases, which need to be implemented meticulously. The implementation used in this work ignores many edge cases and uses the most general formulations available; thus for future work it is important to include exceptions and edge cases in the code. Additionally, many modifications to UFF exist, and it would be of interest to provide flags so that users can select which variant(s) they would like to use.

5. Conclusions. The Universal Force Field (UFF) was implemented in Julia and tested on a range of sample systems. Performance analysis showed that the time needed to generate a parameter set for N unique atom types scales as N^3, and that a multithreaded approach to parameter generation works best for larger N. For smaller N, a single-threaded approach is much faster due to the lack of overhead for thread setup and management; however, the time savings are significant for larger N. Quite notably, without the use of explicit force calculations or a parent molecular mechanics framework, it was possible to optimize the structure of butane via global optimization and via local gradient-based optimization, the latter of which was performed via forward-mode automatic differentiation. These feats were made relatively straightforward due to features unique to Julia, including the macro system which allows for straightforward multithreading and automatic differentiation. Future work will focus on integrating the UFF code into molecular mechanics codes, such as those in the CESMIX suite.

Acknowledgments. I would like to thank Dallas Foster for communications about the implementation of this project, as well as the CESMIX team for developing tools for molecular simulation via Julia. I also thank the 18.337 team for guidance in learning Julia and implementing high-performance code.

REFERENCES

[1] https://github.com/cesmix-mit/.
[2] https://github.com/JuliaMolSim/.
[3] https://github.com/openmm/openmmforcefields.
[4] P. Bauer, B. Hess, and E. Lindahl, *Gromacs 2022.1 manual*, (2022), https://doi.org/10.5281/ZENODO.6451567, https://zenodo.org/record/6451567.
[5] A. Dhasmana, S. Raza, R. Jahan, M. Lohani, and J. M. Arif, *High-throughput virtual screening (HTVS) of natural compounds and exploration of their biomolecular mechanisms*, in New Look to Phytomedicine, Elsevier, 2019, pp. 523–548, https://doi.org/10.1016/b978-0-12-814619-4.00020-3, https://doi.org/10.1016/b978-0-12-814619-4.00020-3.
[6] R. Notman and J. Anwar, *Breaching the skin barrier — insights from molecular simulation of model membranes*, Advanced Drug Delivery Reviews, 65 (2013), pp. 237–250, https://doi.org/10.1016/j.addr.2012.02.011, https://doi.org/10.1016/j.addr.2012.02.011.
[7] Ömer H. Omar, M. del Cueto, T. Nematiaram, and A. Troisi, *High-throughput virtual screening for organic electronics: a comparative study of alternative strategies*, Journal of Materials Chemistry C, 9 (2021), pp. 13557–13583, https://doi.org/10.1039/d1tc03256a, https://doi.org/10.1039/d1tc03256a.
[8] J. C. Phillips, D. J. Hardy, J. D. C. Maia, J. E. Stone, J. V. Ribeiro, R. C. Bernardi, R. Buch, G. Fiorin, J. Hénin, W. Jiang, R. McGreevy, M. C. R. Melo, B. K. Radak, R. D. Skeel, A. Singharoy, Y. Wang, B. Roux, A. Aksimentiev, Z. Luthey-Schulten, L. V. Kalé, K. Schulten, C. Chipot, and E. Tajkhorshid, *Scalable molecular dynamics on CPU and GPU architectures with NAMD*, The Journal of Chemical Physics, 153 (2020), p. 044130, https://doi.org/10.1063/5.0014475, https://doi.org/10.1063/5.0014475.
[9] A. K. Rappe, C. J. Casewit, K. S. Colwell, W. A. Goddard, and W. M. Skiff, *UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations*,

Journal of the American Chemical Society, 114 (1992), pp. 10024–10035, https://doi.org/10.1021/ja00051a040, https://doi.org/10.1021/ja00051a040.

[10] A. K. Rappe and W. A. Goddard, *Charge equilibration for molecular dynamics simulations*, The Journal of Physical Chemistry, 95 (1991), pp. 3358–3363, https://doi.org/10.1021/j100161a070, https://doi.org/10.1021/j100161a070.

[11] C. Simon, A. H. York, Sturluson, C. Simon, A. Henle, N. Gantzler, M. T. Huynh, CalebLaird, J. TagBot, and M. Piibeleht, *Simonensemble/porousmaterials.jl: v0.4.0*, 2022, https://doi.org/10.5281/ZENODO.1400838, https://zenodo.org/record/1400838.

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