SoftwareX 13 (2021) 100663

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

EZFF: Python library for multi-objective parameterization and uncertainty quantification of interatomic forcefields for molecular dynamics

Aravind Krishnamoorthy [a], Ankit Mishra [a], Deepak Kamal [b], Sungwook Hong [a,c], Ken-ichi Nomura [a,*], Subodh Tiwari [a], Aiichiro Nakano [a], Rajiv Kalia [a], Rampi Ramprasad [b], Priya Vashishta [a]

[a] Collaboratory of Advanced Computing and Simulations, University of Southern California, Los Angeles, CA 90089-0242, United States of America
[b] Georgia Institute of Technology, 771 Ferst Drive, Northwest Atlanta, Atlanta, GA 30332, United States of America
[c] Department of Physics & Engineering, California State University, Bakersfield, CA 93311, United States of America

ARTICLE INFO

Article history:
Received 9 October 2020
Received in revised form 13 January 2021
Accepted 15 January 2021

Keywords:
Molecular dynamics
Interatomic forcefield
Genetic algorithm
Global optimization

ABSTRACT

Parameterization of interatomic forcefields is a necessary first step in performing molecular dynamics simulations. This is a non-trivial global optimization problem involving quantification of multiple empirical variables against one or more properties. We present EZFF, a lightweight Python library for parameterization of several types of interatomic forcefields implemented in several molecular dynamics engines against multiple objectives using genetic-algorithm-based global optimization methods. The EZFF scheme provides unique functionality such as the parameterization of hybrid forcefields composed of multiple forcefield interactions as well as built-in quantification of uncertainty in forcefield parameters and can be easily extended to other forcefield functional forms as well as MD engines.

Code metadata

Current code version	*v0.9.4*
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-20-00066
Legal Code License	*MIT License*
Code versioning system used	*git*
Software code languages, tools, and services used	*Python*
Compilation requirements, operating environments & dependencies	*Message Passing Interface (MPI) libraries*
If available Link to developer documentation/manual	https://ezff.readthedocs.io/en/latest/
Support email for questions	cacs@usc.edu

Software metadata

Current software version	*0.9.4*
Permanent link to executables of this version	https://github.com/arvk/EZFF/archive/v0.9.4.zip
Legal Software License	*MIT License*
Computing platforms/Operating Systems	*Linux, OSX*
Installation requirements & dependencies	*Message Passing Interface (MPI) libraries*
If available, link to user manual – if formally published include a reference to the publication in the reference list	https://ezff.readthedocs.io/en/latest/
Support email for questions	cacs@usc.edu

1. Introduction

Molecular Dynamics (MD) is an important technique in computational chemistry, biology and materials science for simulating

* Corresponding author.
E-mail address: knomura@usc.edu (Ken-ichi Nomura).

https://doi.org/10.1016/j.softx.2021.100663

the structure, dynamics and thermodynamic properties at the atomic scale. While parameter-free *ab initio* quantum molecular dynamics simulations have been successful in simulating atomic dynamics in small (<1000s of atoms) systems over brief timescales (< 100s of ps), modeling realistically complex systems (over 10^6 atoms) over chemically and biologically relevant timescales ($\sim$ μs to ms) requires the use of classical molecular dynamics simulations, where interatomic interactions are approximated by empirical/semi-empirical forcefields, which are sets of parameterized mathematical functions.

The reliability of results from classical MD simulations, and their predictive power, fundamentally depend upon the quality of the forcefields used. Therefore, parameterization of forcefields is a necessary first step in performing high-quality MD simulations. This parameterization process either involves matching forces and energies for various input clusters as implemented in the *potfit* package [1–3] or the identification of an optimal set of numerical parameters that best approximates experimental or quantum chemical reference data [4,5] for material systems under investigation [1]. Further, forcefields must be parameterized to simultaneously reproduce several materials properties, necessitating multi-objective optimization techniques. The large number of optimizable empirical parameters (up to several hundred parameters for complex force fields like ReaxFF [6] and COMB [7]) as well as a non-trivial correlation between these variables makes forcefield parameterization a highly complex global optimization problem [8].

Owing to the complexity of handling high-dimensional parameter and objective space, most existing parameterization schemes transform this into more computationally tractable analogues. One of the earliest schemes, the sequential one-parameter parabolic interpolation (SOPPI) [9], casts this as a sequence of one-dimensional local parabolic minimizations, where a single parameter is optimized to minimize a single weighted sum of several objectives. While computationally simple, the SOPPI method has several significant shortcomings, primarily the propensity of the algorithm to converge to a neighboring local minimum, rather than a global minimum as well as poor convergence characteristics if the optimization is started from a poor initial guess [8,10]. Further, SOPPI is an inherently sequential method that cannot take advantage of vast capabilities of today's highly parallel supercomputers. Non-deterministic methods like simulated annealing and differential evolution, as implemented in packages like *potfit*, are more robust against convergence to local minima, but are constrained to optimizing a single weighted sum of multiple objectives.

These shortcomings are partially addressed in recent multi-objective schemes like GARFfield [11], which use evolutionary algorithms to perform global minimization of a weighted sum of multiple objectives, using an *a priori* user-provided weighting scheme. Other schemes such as Multi-objective evolutionary strategies [12] and MOGA [13] Rotation-invariant Particle Swarm Optimization with isotropic Gaussian Mutation (RIPSOGM) [8] have been developed that evolve the entire Pareto Frontier of multiple forcefield populations at once, without the need to specify *a priori* weights for the different objectives. Existing software frameworks for forcefield optimization are also commonly limited to the parameterization of a single predefined functional forcefield form, such as the Forcefield Toolkit (ffTK) [14] and general automated atomic model parameterization (GAAMP) [15] frameworks for the CHARMM forcefield, Paramfit [16] for AMBER forcefields and MOGA [13] for ReaxFF forcefields. However, to the authors' knowledge, there is no existing general multi-objective global optimization framework that is applicable to parameterization of different forcefield functional forms implemented in different MD engines.

Here, we introduce EZFF, a flexible Python-based multi-objective forcefield optimizer framework for parameterization of multiple forcefield functional forms, including reactive forcefields such as ReaxFF and COMB, using different molecular dynamics engines (LAMMPS [17], GULP [18], RXMD [19] etc.) against multiple user-definable objectives using an *a posteriori* Pareto-dominant multi-objective methods that are proven to be effective for forcefield parameterization [20,21]. In the next section, we describe the EZFF framework and typical workflow for forcefield parameterization as well as the different objective functions currently supported for force field development. In Section 3, we illustrate the application of EZFF to the development of several forcefields, including ReaxFF, Stillinger–Weber etc.

2. Software description

The EZFF source code is written entirely in Python 3 to take advantage of the large user base, and close integration with large number of scientific libraries for data processing, analysis and optimization. Specifically, EZFF makes use of the open source Platypus library [22] for performing evolutionary optimization. Through Platypus, EZFF supports an ensemble of genetic algorithms, including NSGA2 [23], NSGA3 [24], Evolutionary strategies like IBEA, Differential evolution (GDE3), and particle swarm methods like OMOPSO [25] capable of exploring different regions in the parameter space for nonconvex, discontinuous, and multimodal solutions [11], as evident in the several previous studies that have used hard-coded genetic algorithms for forcefield training [26–28].

Fig. 1 describes the typical workflow for multi-objective optimization of a classical forcefield using EZFF. The forcefield optimization process begins with three inputs from the user:

1. The functional form of the forcefield to be optimized, as well as the numerical parameters to be determined are provided in a forcefield template file. This forcefield template is identical to a valid forcefield file, where the optimizable parameters are replaced by named variables enclosed in double angular brackets '<< >>'.
2. The user is also required to provide, in a separate file, the maximum and minimum permissible values of these parameters. During global optimization, EZFF generates new forcefields by randomly sampling each variable within the provided minimum and maximum bounds.
3. Finally, the user must provide a set of one or more structural, chemical and energetic properties that the forcefield must reproduce. Deviation from these ground truths values define the objectives (i.e. errors) that must be minimized during the global optimization process.

The inputs are collected together in a single user-defined Python script (run.py in Fig. 1), which defines functions for the computation of objectives/errors for the forcefield parameterization, as well as other important properties for global optimization (such as the GA algorithm to be used, population size at each epoch, number of epochs and parallelization scheme to be used). Based on the user-defined values, EZFF generates n valid forcefields based on the template and parameters sampled randomly from the permissible ranges, where n is the population size. These different forcefields are evaluated by the user-chosen MD engines, which are spawned in parallel by mpi4py (or sequentially, if MPI is not available). Each MPI ranks creates its own working directory with necessary input files, executable and further utility files as defined by the user (as defined in the *run.py* script). Individual forcefields are assigned to all available MPI ranks in a round robin fashion. Each MPI rank runs an serial instance of user-defined MD engine such as GULP, LAMMPS, RXMD etc,

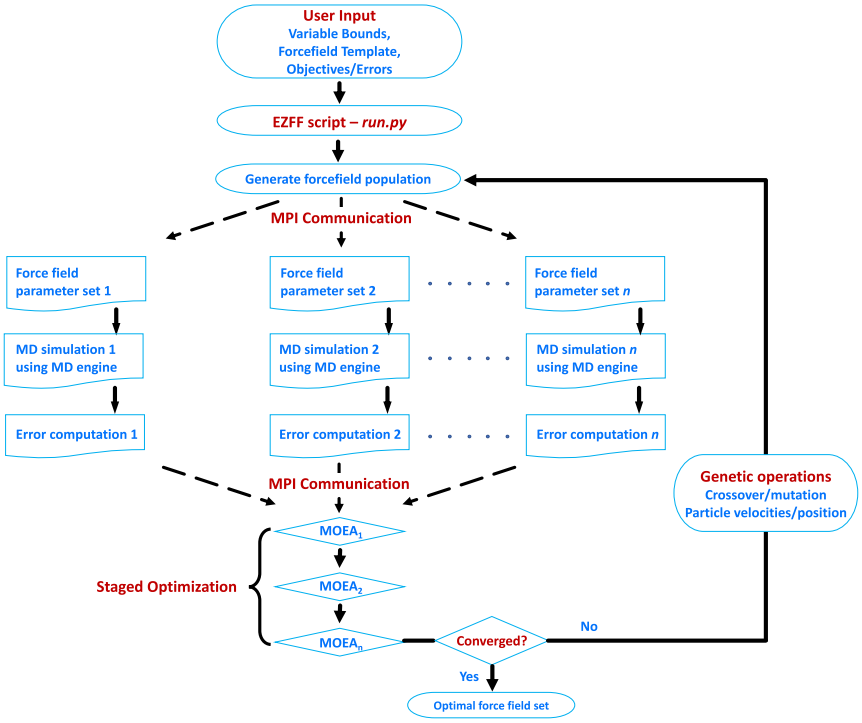

Fig. 1. EZFF forcefield parameterization workflow. The forcefield parameterization process begins with user specification of forcefield templates and variable bounds in the EZFF script. These are used to generate an ensemble of forcefield candidates, which are evaluated independently by MPI ranks spawned by the master EZFF process. The errors (i.e. objectives) computed by these MPI ranks are communicated back to the master process, which performs genetic operations to spawn the next generation of forcefield candidates.

to produce a simulation corresponding to a specified generated population. The material properties correspond to each forcefield is then compared to their ground truth values to compute the error(s) corresponding to each forcefield in the population. These errors/objectives for each MPI rank is communicated to the main EZFF thread, which uses Platypus and user-defined genetic algorithm to perform crossover and mutation operations and particle displacements to generate the next generation of forcefield candidates for evaluation.

EZFF provides several modules for each stage of this parameterization process (Fig. 2) including function definitions to support different tasks required for fitting force fields and implementing the parallel workflow interface with different simulation engines. EZFF is the main module that defines the OptProblem and Algorithm classes for flexible definition of optimization parameters (like error function, genetic algorithms, stopping criterion etc.). Module FFIO defines methods to handle I/O operations on forcefield templates, parameter ranges and EZFF-generated forcefields. LAMMPS and RXMD software for performing MD to evaluate the generated forcefields. These modules include functions to spawn and run these MD engines as well as methods to read data from the execution of these MD programs. In addition, EZFF also includes custom interfaces to popular Density Functional Theory (DFT) based programs like VASP and QChem to read ground truth values for energies, forces, frequencies and other properties. These custom interfaces enable EZFF to import ground truth values both from popular simulation programs like VASP, QChem, GULP and LAMMPS, as well as other simulation packages that provide unique capabilities like RXMD [19] and QXMD [29], which are not currently accessible via other parsers like ASE [30] and pymatgen [31].

The errors module defines several common objectives used to evaluating the quality of forcefields such as atomic positions (including bond lengths and angles), atomic charges, crystal structure and lattice constants, vibration energies, elastic constants, phonon frequencies, bond stretching and dissociation energies.

The utilities module contains functions of generic utility including unit conversion and complex force field template generation such as ReaxFF.

2.1. Installation and compilation

The library is available on the Python Package Index (PyPI) and can be installed using the command

```
pip install EZFF
```

Alternatively, the latest developmental version can be downloaded from the publicly available Github repo at https://github.com/arvk/EZFF and can be compiled by executing setup.py in the root directory of the EZFF tree (Fig. 2). The user is responsible for ensuring the installation of the two EZFF dependencies (mpi4py [32,33] and xtal [34]) separately.

2.2. Software functionalities

EZFF provides a lightweight and extensible Python interface that enables, for the first time, serial and parallel multi-objective global parameterization of multiple types of simple and complex reactive and non-reactive forcefields such as ReaxFF, COMB, Stillinger–Weber, Lennard Jones via multiple MD engines. Uniquely, EZFF also allows for the parameterization of hybrid forcefields composed of multiple interatomic interactions with different functional forms.

The modular design of EZFF provides a quick and facile method to change optimization algorithms during forcefield parameterization. This enables strategies such as staged optimization, where diversity preserving genetic algorithms like NSGA-III can be initially employed to more completely sample the parameters space

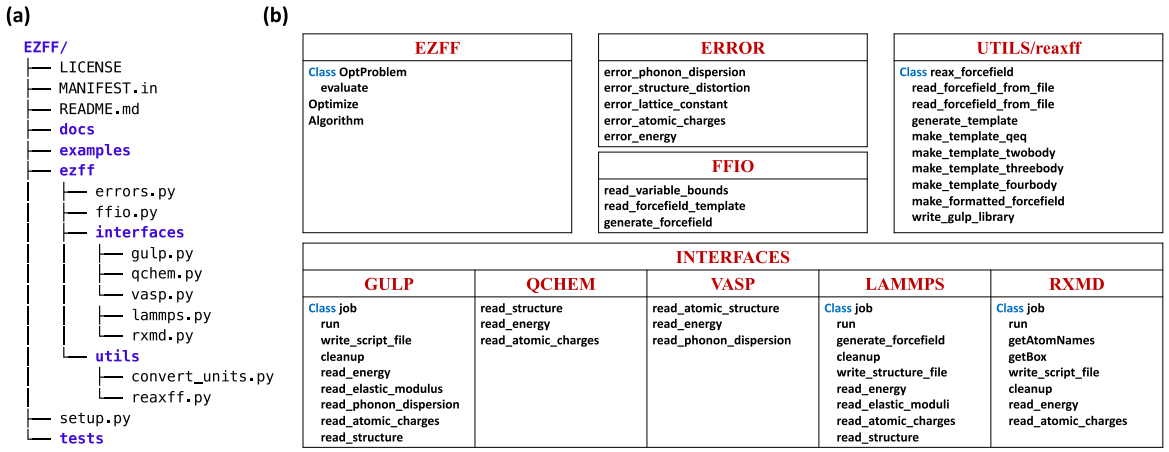

Fig. 2. Organization of EZFF forcefield fitting code. (a) Tree structure of the directory and the files necessary to run EZFF. (b) Module diagram showing definitions of various classes and internal organization of flow of control.

followed by a second stage where other multi-objective optimization schemes like differential evolution can be used to more efficiently converge to local minima in the objective phase space.

Optimization algorithms used in EZFF evolve and keep track of the entire Pareto front at every epoch during optimization. Therefore, we can perform Pareto-frontal uncertainty quantification for forcefields generated by EZFF. This Pareto-frontal breakdown of different forcefields for each epoch provides a natural way to establish one of the primary sources of uncertainty in molecular dynamics simulations — namely the uncertainty in forcefield parameters. This Pareto-frontal uncertainty quantification approach offers an alternative method to estimate the errors in forcefield parameters [13,35–39], to complement the predominantly Bayesian approaches employed in other packages used in prior studies [40].

Illustrative examples

We present 2 examples to demonstrate the unique capabilities of EZFF in parameterizing reactive and non-reactive forcefields. In the first example, we demonstrate the parameterization of a hybrid forcefield, consisting of multiple functional forms. The second example covers the optimization of reactive forcefields for modeling accurate metal-polymer interfaces

Example 1: Optimization of Hybrid Forcefields for Layered Two-dimensional Materials

Two-dimensional and layered materials are being actively investigated for their unique electronic structure and mechanical and transport properties arising out of their quantum confinement along one dimension. In the case of layered transition metal dichalcogenides like MoS_2 interatomic interactions can broadly be divided into strong covalent interactions between nearest-neighbor Mo and S atoms, and longer-range van der Waals interactions between MoS_2 sheets along the c axis (Fig. 3).

These interactions are well described by an in-plane Stillinger–Weber interaction between Mo and S atoms combined with out-of-plane Lennard Jones interactions between S atoms in adjacent MoS_2 layers. Specifically, this system can be described by a hybrid forcefield that includes:

The total potential energy of the given system of N atoms located at $[\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N]$ in the SWFF can be written as

$$\mathrm{E_{SW}}\left(\mathbf{r}_1, \mathbf{r}_2, \ldots \mathbf{r}_N\right) = \sum_{i<j} \mathrm{V}_2(r_{ij}) + \sum_{i<j<k} \mathrm{V}_3(r_{ij}, r_{jk}, \theta_{ijk})$$

where $r_{ij} = |\mathbf{r}_j - \mathbf{r}_i|$. The 2-body term, V_2, is defined as

$$\mathrm{V}_2\left(r_{ij}\right) = A\left(\frac{B}{r_{ij}^4} - 1\right)\exp\left(\frac{\gamma}{r_{ij} - r_{\mathrm{cut}}}\right)$$

The two-body term is defined by 3 optimizable parameters, A, B and γ.

The 3-body term, V_3 around a central atom i is given by three optimizable parameters, λ, γ_1 and γ_2 and has the following functional form. Geometric parameters, including interaction cut-off distances, r_{cut}, r_{cut1} and r_{cut2} and equilibrium angles, θ_0 are held fixed during parameterization.

$$\mathrm{V}_3\left(r_{ij}, r_{ik}, \theta_{ijk}\right) = \lambda \exp\left(\frac{\gamma_1}{r_{ij} - r_{cut1}} - \frac{\gamma_2}{r_{ik} - r_{cut2}}\right)(\cos\theta - \cos\theta_0)^2$$

Interactions between adjacent MoS_2 layers, α and β, are described by Lennard Jones interactions between sulfur atoms at r1, r2, r3

$$\mathrm{E_{LJ}}\left(\mathbf{r}_1^{\alpha}, \mathbf{r}_2^{\alpha}, \ldots \mathbf{r}_1^{\beta}, \mathbf{r}_2^{\beta}, \ldots\right) = \sum_{i,j} \varepsilon \left[\left(\frac{\sigma}{\left|r_i^{\alpha} - r_j^{\beta}\right|}\right)^{12} - 2 \times \left(\frac{\sigma}{\left|r_i^{\alpha} - r_j^{\beta}\right|}\right)^{6}\right] \forall \left|r_i^{\alpha} - r_j^{\beta}\right| < r_{cut}^{LJ}$$

This longer-range non-bonded interaction is defined by three optimizable parameters, ε, σ and r_{cut}^{LJ}.

These parameterizable variables and permissible ranges are defined in the forcefield template and parameter ranges files as shown in Fig. 4. Fig. 5 presents complete script to perform parameterization task.

Fig. 6 shows that by using NSGA-III along with EZFF quickly identifies a set of forcefields that simultaneous optimizes all 4 objectives considered in this example.

Example 2: Optimization of ReaxFF forcefields for Al-polymeric materials using RXMD

Understanding the electronic properties at metal–organic interfaces are becoming increasingly crucial as electronic devices like batteries and capacitors move towards smaller scales and higher efficiency. But the experimental exploration of interfacial electronic properties is mired with challenges owing to their immense chemical and morphological complexity [41–43]. To sidestep this difficulty, the community currently uses first-principles computations on highly simplified models of interfacial structures to access these properties [44,45]. But, the understanding of electronic processes is currently limited by the ideal nature of the interfacial structures used in these simulations. While generation of realistic interface structures using first-principle

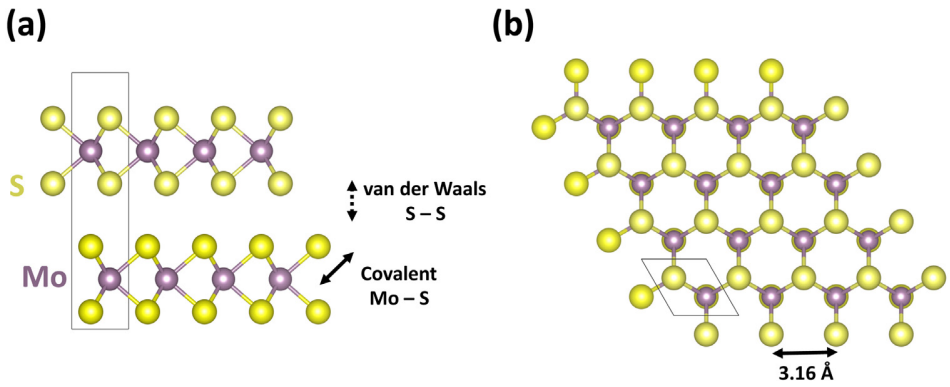

Fig. 3. Crystal structure and interactions in MoS_2. (a) The layered MoS_2 crystal is held together by strong in-plane covalent Mo-S interactions and relatively weaker S–S van der Waals interactions. (b) The honeycomb crystal structure of MoS2 has a lattice constant of 3.16 Å. The unit cell for MoS_2 is indicated by thin black lines.

Forcefield Template

```
# E = A.exp(rho/(r-rmax)).(B/r**4 - 1)
# atom1 atom2 A gamma B rmin rmax <3*flags>
sw2
S1    S1   <<A_SS>>    <<G_SS>>    <<B_SS>>    0.00 4.317  0 0 0
Mo    S1   <<A_MoS>>   <<G_MoS>>   <<B_MoS>>   0.00 3.191  0 0 0
Mo    Mo   <<A_MoMo>>  <<G_MoMo>>  <<B_MoMo>>  0.00 4.317  0 0 0
S2    S2   <<A_SS>>    <<G_SS>>    <<B_SS>>    0.00 4.317  0 0 0
Mo    S2   <<A_MoS>>   <<G_MoS>>   <<B_MoS>>   0.00 3.191  0 0 0

# E(three) = lambda * exp(gamma0/(r12-rmax12) + gamma1/(r13-rmax13))(cos-ct0)**2
# atom1 atom2 atom3 lambda theta0 gamma0 gamma1 <rmin12> rmax12 <rmin13> rmax13 <rmin23>
rmax23 <4*flags>
sw3
Mo S1 S1 <<L_MoSS>> <<theta_0>> <<G_MoSS>> <<G_MoSS>> 3.191 3.191 4.317 0 0 0 0
S1 Mo Mo <<L_SMoMo>> <<theta_0>> <<G_SMoMo>> <<G_SMoMo>> 3.191 3.191 4.317 0 0 0 0
Mo S2 S2 <<L_MoSS>> <<theta_0>> <<G_MoSS>> <<G_MoSS>> 3.191 3.191 4.317 0 0 0 0
S2 Mo Mo <<L_SMoMo>> <<theta_0>> <<G_SMoMo>> <<G_SMoMo>> 3.191 3.191 4.317 0 0 0 0


# E = A/(r**m) - B/(r**n)
# atom1 atom2 (A B/epsilon sigma) <rmin> rmax <2*flags>
lennard epsilon
S1 S2 <<A_LJ>> <<B_LJ>> <<S_LJ>> 1 0
```

Parameter Ranges

```
A_SS 0.9495 1.5825
G_SS 0.34125 0.56875
B_SS 29.025 48.375
A_MoS 5.21475 8.69125
G_MoS 0.273 0.455
B_MoS 6.39 10.65
A_MoMo 3.594 5.99
G_MoMo 0.42375 0.70625
B_MoMo 13.9875 23.3125
L_MoSS 9.13935 15.2323
theta_0 61.5 102.5
G_MoSS 0.828 1.38
L_SMoMo 21.716 36.1933
G_SMoMo 1.78275 2.97125
A_LJ 0.0237 0.0395
B_LJ 2.685 4.475
S_LJ 4.5 7.5
```

Fig. 4. Inputs for EZFF forcefield parameterization. The forcefield template file is characterized by the presence of named variables (green, enclosed in dual angular brackets, <<>>), which will be replaced by numerical values during the optimization process. The minimum and maximum permissible values for these variables are provided in a separate parameter ranges file, as shown above.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

based methods remains intractable due its complexity, structure generation using MD simulations utilizing reactive forcefields like ReaxFF [6,46] remains a feasible option. Here, we attempt to generate a ReaxFF forcefield using EZFF which can accurately capture the interaction between Aluminum (which is a common electrode) and $C-H-O$ based organic molecules/polymers to facilitate easy creation of realistic Al-organic interfaces.

ReaxFF is a class of semi-empirical bond-order-based forcefields for describing reactive dynamics involving bond breaking and formation and are well suited to describe highly heterogeneous material systems. ReaxFF forcefields are composed of several hundred parameterizable variables that describe various 2-body, 3-body and 4-body interactions between different atomic species, which makes global optimization of these potentials highly challenging. Recently, Hong and van Duin parameterized a new ReaxFF forcefield for Al/C/H/O materials against interaction energies between organic radicals and Al [47]. However, this work fails to capture the interaction energy between fully saturated organic molecules and the Al surface correctly resulting in unrealistic interface structures. Therefore, we use EZFF to reparameterize the ReaxFF forcefield from Hong and van Duin to better reproduce interaction energies between an Al (111) surface and two representative saturated organic molecules, CH_4 and C_2H_6. Specifically, EZFF is used to parameterize only variables controlling the Al–C and Al–H 2-body interactions to most accurately reproduce the DFT-computed interaction energy between Al and CH_4/C_2H_6.

Fig. 7 shows the full run.py script to perform forcefield parameterization using the RXMD [19] as the MD engine to evaluate the quality of different ReaxFF forcefield. The script uses the Indicator-Based Evolutionary Algorithm (IBEA) to optimize the ReaxFF parameters over 100 epochs against two objectives — deviations from DFT-computed energies for the Al–CH_4 and Al–C_2H_6 systems respectively.

Figs. 8a and 8b show that IBEA converges rapidly within 100 epochs producing optimal forcefields that replicate the adsorption energy profiles for both CH_4 and C_2H_6 molecules and Fig. 9 shows that the adsorption energy profiles from the optimized forcefields are in much better agreement with the DFT values than those generated from the original forcefield from Hong and van Duin.

3. Impact

Parameterization of interatomic forcefields is a highly time-consuming and cumbersome process, whose complexity, along with the lack of general best-practices guidelines has led the process of forcefield construction to be considered an 'art'. EZFF attempts to improve the process of forcefield parameterization by

```
import ezff
from ezff.interfaces import vasp, gulp
import numpy as np

bounds = ezff.read_variable_bounds('variable_bounds', verbose=False)
template = ezff.read_forcefield_template('template')

# DEFINE GROUND TRUTHS
gt_disp_GM = vasp.read_phonon_dispersion('/staging/pv/kris658/EZFF/data/GM/band.dat')
gt_relax_structure = vasp.read_atomic_structure('/staging/pv/kris658/EZFF/data/POSCAR')
gt_c11 = 260.0 #GPa for C11 of MoSe2

def my_error_function(variable_values):
    myrank = pool.rank
    # FOR THE RELAXED STRUCTURE - GM Phonon Dispersion
    path = str(myrank)+'/relaxed_GM'
    relaxed_job = gulp.job(path=path)
    relaxed_job.structure = gt_relax_structure
    relaxed_job.forcefield = ezff.generate_forcefield(template, variable_values, FFtype = 'SW')
    relaxed_job.options['pbc'] = True
    relaxed_job.options['relax_atoms'] = True
    relaxed_job.options['relax_cell'] = True
    relaxed_job.options['phonon_dispersion'] = True
    relaxed_job.options['phonon_dispersion_from'] = '0 0 0'
    relaxed_job.options['phonon_dispersion_to'] = '0.5 0.0 0'
    relaxed_job.run() # Submit job and read output
    # Read output from completed GULP job and cleanup job files
    disp_GM = relaxed_job.read_phonon_dispersion()
    md_relaxed_moduli = relaxed_job.read_elastic_moduli()
    md_relaxed_structure = relaxed_job.read_structure()
    relaxed_job.cleanup()  # FINISH RELAXED JOB
    # Compute 4 errors from the GULP job
    error_abc, error_ang = ezff.error_lattice_constant(MD=md_relaxed_structure, GT=gt_relax_structure)
    a_lattice_error = np.linalg.norm(error_abc[0])   # Error in 'a' lattice constant
    c_lattice_error = np.linalg.norm(error_abc[2])   # Error in 'c' lattice constant
    md_c11 = md_relaxed_moduli[0][0,0]
    modulus_error_c11 = np.linalg.norm(md_c11 - gt_c11)
    phon_error_GM = ezff.error_phonon_dispersion(MD=disp_GM, GT=gt_disp_GM, weights='acoustic')
    return [a_lattice_error, c_lattice_error, modulus_error_c11, phon_error_GM]

pool = ezff.Pool()
problem = ezff.OptProblem(num_errors=4, variable_bounds=bounds, error_function=my_error_function, template=template)
algorithm = ezff.Algorithm(problem, 'NSGAIII', population = 256, pool = pool)
ezff.optimize(problem, algorithm, iterations = 100)
pool.close()
```

Fig. 5. Complete script (run.py) for preforming parameterization of a hybrid Stillinger–Weber and Lennard-Jones forcefield using EZFF.

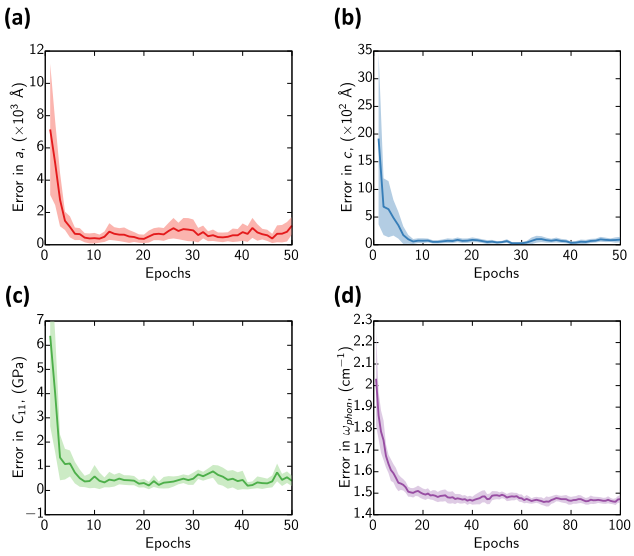

Fig. 6. Quality of forcefields during optimization. (a–d) show the computed error for each of the four objectives used for parameterizing the hybrid Stillinger–Weber and Lennard-Jones forcefield. At each epoch, the mean (dark line) and standard deviation (light fill) of the 20 best forcefields on the Pareto front are plotted. The NSGA2 algorithm converges quickly to produce good forcefields within 50 epochs.

providing a simple workflow, in an easy-to-understand scripting language to optimize a wide range of empirical forcefields of varying levels of complexity. The highly parallelized parameterization process enables rapid prototyping and testing of multiple forcefields before performing production molecular dynamics simulations. The parameterization of hybrid forcefields opens doors for the direct parameterization of interatomic interactions for highly heterogeneous material systems, including those containing interfaces between two distinct phases. This would foster an ensemble of exploratory studies into the rich and largely unexplored space of interfacial properties which are rather exotic compared to bulk materials.

4. Conclusions

In this paper, we described EZFF, a lightweight and flexible Python library for multi-objective global parameterization of different types of interatomic forcefields for molecular dynamics simulations. The highly parallelized and scalable framework will enable quick prototyping of several forcefield function forms, as well as hybrid forcefields composed of multiple interatomic interactions. EZFF also admits staged optimization strategies using multiple optimization algorithms for generating high-quality forcefields with built-in Pareto-frontal uncertainty quantification, thus greatly simplifying the currently cumbersome process for construction and validation of forcefields.

The EZFF codebase is meant to continuously evolve in future releases, welcoming suggestions and contributions from users. Future versions will include capability to use dynamic properties (mean square displacements, and various correlation functions) as objectives in the parameterization process, enabling fitting of forcefields to dynamic material properties.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Mishra, Kamal, Ramprasad, and Tiwari were supported by the Office of Naval Research through a Multi-University Research Initiative (MURI) grant N00014-17-1-2656. Nakano and Kalia were supported by the FMR Artificial Intelligence Driven Cybermanufacturing of Quantum Material Architectures by the National Science Foundation under Award 2036359. Krishnamoorthy, Hong,

```
import ezff
from ezff.interfaces import gulp, vasp
from ezff.utils.reaxff import reax_forcefield

# Define ground truths
structure_ch4 = vasp.read_atomic_structure('ground_truths/ch4')
energy_ch4 = vasp.read_energy('ground_truths/ch4')
structure_c2h6 = vasp.read_atomic_structure('ground_truths/c2h6')
energy_c2h6 = vasp.read_energy('ground_truths/c2h6')

def my_error_function(rr):
    path = str(pool.rank)

    # Calculate CH4 structure
    ch4_job = gulp.job(path = path)
    ch4_job.structure = structure_ch4
    ch4_job.forcefield = ezff.generate_forcefield(template, rr, FFtype = 'reaxff')
    ch4_job.options['pbc'] = True
    ch4_job.options['relax_atoms'] = False
    ch4_job.options['relax_cell'] = False
    # Run gulp calculation
    ch4_job.run(command='gulp')
    # Read output from completed GULP job and clean-up
    ch4_md_energy = ch4_job.read_energy()
    ch4_job.cleanup()

    # Calculate C2H6 structure
    c2h6_job = gulp.job(path = path)
    c2h6_job.structure = structure_c2h6
    c2h6_job.forcefield = ezff.generate_forcefield(template, rr, FFtype = 'reaxff')
    print(c2h6_job.forcefield[0])
    c2h6_job.options['pbc'] = True
    c2h6_job.options['relax_atoms'] = False
    c2h6_job.options['relax_cell'] = False
    # Run gulp calculation
    c2h6_job.run(command='gulp')
    # Read output from completed GULP job and clean-up
    c2h6_md_energy = c2h6_job.read_energy()
    c2h6_job.cleanup()

    # Calculate errors
    ch4_error = ezff.error_energy(ch4_md_energy, energy_ch4-energy_ch4[-1], weights = 'uniform')
    c2h6_error = ezff.error_energy(c2h6_md_energy, energy_c2h6-energy_c2h6[-1], weights = 'uniform')

    return [ch4_error, c2h6_error]


pool = ezff.Pool()

if pool.is_master():
    # Generate forcefield template and variable ranges
    FF = reax_forcefield('AlCHO.ff')
    FF.make_template_twobody('Al','C')
    FF.make_template_twobody('Al','H')
    FF.generate_templates()

# Read template and variable ranges
bounds = ezff.read_variable_bounds('param_ranges', verbose=False)
template = ezff.read_forcefield_template('ff.template.generated')

problem = ezff.OptProblem(num_errors=2, variable_bounds=bounds, error_function=my_error_function, template=template)
algorithm = ezff.Algorithm(problem, 'NSGAII', population = 239, pool = pool)
ezff.optimize(problem, algorithm, iterations = 100, write_forcefields = 1)
pool.close()
```

Fig. 7. Complete script (run.py) for preforming parameterization of a ReaxFF forcefield for the polymer-Al system using EZFF.

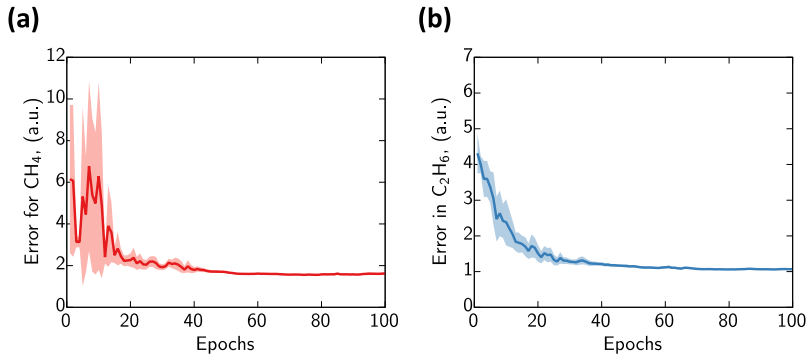

Fig. 8. Quality of ReaxFF forcefields during genetic optimization. (a) and (b) show the computed error in the adsorption energy profile respectively for the Al–CH_4 and Al–C_2H_6 systems. At each epoch, the mean (dark line) and standard deviation (light fill) of the 20 best forcefields on the Pareto front are plotted. The IBEA algorithm employed here converges quickly to produce good forcefields within 50 epochs.

Nomura, and Vashishta were supported as part of the Computational Materials Sciences Program funded by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, under Award DE-SC0014607.This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562 and The Partnership for Advance Computing Environment (PACE) at Georgia Institute of Technology.

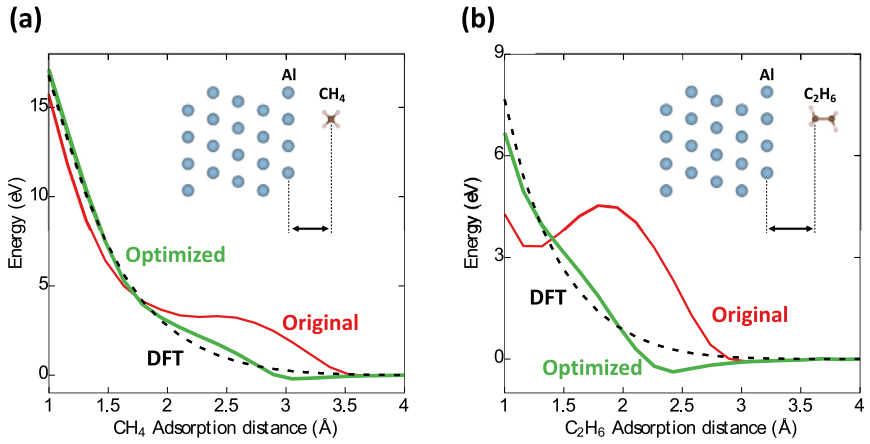

Fig. 9. Adsorption energy profile of optimized ReaxFF forcefield. Al-molecule interaction energies from the optimized forcefield (100th epoch) are in closer agreement with the DFT ground truth values than those from the original forcefield from Hong and van Duin.

References

[1] Brommer P, Gahler F. Potfit: effective potentials from ab initio data. Modelling Simulation Mater Sci Eng 2007;15(3):295–304.
[2] Brommer P, Gähler F. Effective potentials for quasicrystals from ab-initio data. Phil Mag 2006;86(6–8):753–8.
[3] Brommer P, et al. Classical interaction potentials for diverse materials from ab initio data: a review of potfit. Modelling Simulation Mater Sci Eng 2015;23(7):074002.
[4] Furman D, et al. Reactive force field for liquid hydrazoic acid with applications to detonation chemistry. J Phys Chem C 2016;120(9):4744–52.
[5] Furman D, Kosloff R, Zeiri Y. Effects of nanoscale heterogeneities on the reactivity of shocked erythritol tetranitrate. J Phys Chem C 2016;120(50):28886–93.
[6] van Duin ACT, et al. ReaxFF: A reactive force field for hydrocarbonds. J Phys Chem A 2001;105(41):9396–409.
[7] Shan TR, et al. Charge-optimized many-body potential for the hafnium/hafnium oxide system. Phys Rev B 2010;81(12).
[8] Furman D, et al. Enhanced particle swarm optimization algorithm: Efficient training of ReaxFF reactive force fields. J Chem Theory Comput 2018;14(6):3100–12.
[9] Vanduin ACT, Baas JMA, Vandegraaf B. Delft molecular mechanics - a new approach to hydrocarbon force-fields - inclusion of a geometry-dependent charge calculation. J Chem Soc Faraday Trans 1994;90(19):2881–95.
[10] Senftle TP, et al. The ReaxFF reactive force-field: development, applications and future directions. NPJ Comput Mater 2016;2.
[11] Jaramillo-Botero A, Naserifar S, Goddard WA. General multiobjective force field optimization framework, with application to reactive force fields for silicon carbide. J Chem Theory Comput 2014;10(4):1426–39.
[12] Larentzos JP, et al. Parameterizing complex reactive force fields using multiple objective evolutionary strategies (MOES). Part 1: ReaxFF models for cyclotrimethylene trinitramine (RDX) and 1, 1-diamino-2, 2-dinitroethene (FOX-7). J Chem Theory Comput 2015;11(2):381–91.
[13] Mishra A, et al. Multiobjective genetic training and uncertainty quantification of reactive force fields. NPJ Comput Mater 2018;4(1):42.
[14] Mayne CG, et al. Rapid parameterization of small molecules using the force field toolkit. J Comput Chem 2013;34(32):2757–70.
[15] Huang L, Roux B. Automated force field parameterization for nonpolarizable and polarizable atomic models based on ab initio target data. J Chem Theory Comput 2013;9(8):3543–56.
[16] Betz R, Walker RC. Paramfit: A program for automated forcefield parameter generation using a genetic algorithm. In: Abstracts of papers of the American Chemical Society. 2012, p. 243.
[17] Plimpton S. Fast parallel algorithms for short-range molecular-dynamics. J Comput Phys 1995;117(1):1–19.
[18] Gale JD. GULP: A computer program for the symmetry-adapted simulation of solids. J Chem Soc Faraday Trans 1997;93(4):629–37.
[19] Nomura K, et al. A scalable parallel algorithm for large-scale reactive force-field molecular dynamics simulations. Comput Phys Comm 2008;178(2):73–87.
[20] Zitzler E, Thiele L. An evolutionary approach for multiobjective optimization: The strength Pareto approach. TIK Report, Zurich, Switzerland: Computer Engineering and Networks Laboratory (TIK), ETH Zurich; 1998.
[21] Zitzler E, Laumanns M, Thiele L. SPEA2: Improving the strength Pareto evolutionary algorithm. TIK Report, Zurich, Switzerland: Computer Engineering and Networks Laboratory (TIK), ETH Zurich; 2001, p. 21.
[22] Hadka D. Project-platypus/platypus: A free and open source python library for multiobjective optimization. 2019 [cited 2019 29 2019]; Available from: https://github.com/Project-Platypus/Platypus.
[23] Srinivas N, Deb K. Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 1994;2(3):221–48.
[24] Deb K, Jain H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach. Part I: Solving problems with box constraints. IEEE Trans Evol Comput 2014;18(4):577–601.
[25] Sierra MR, Coello CAC. Improving PSO-based multi-objective optimization using crowding, mutation and epsilon-dominance. Evol Multi-Criterion Optim 2005;3410:505–19.
[26] Wang JM, Kollman PA. Automatic parameterization of force field by systematic search and genetic algorithms. J Comput Chem 2001;22(12):1219–28.
[27] Larsson HR, van Duin ACT, Hartke B. Global optimization of parameters in the reactive force field ReaxFF for SiOH. J Comput Chem 2013;34(25):2178–89.
[28] Handley CM, Deeth RJ. A multi-objective approach to force field optimization: Structures and spin state energetics of d(6) Fe(II) complexes. J Chem Theory Comput 2012;8(1):194–202.
[29] Shimojo F, et al. QXMD: An open-source program for nonadiabatic quantum molecular dynamics. SoftwareX 2019;10:100307.
[30] Larsen A Hjorth, et al. The atomic simulation environment—a Python library for working with atoms. J Phys: Condens Matter 2017;29(27):273002.
[31] Ong SP, et al. Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. Comput Mater Sci 2013;68:314–9.
[32] Dalcin L, Paz R, Storti M. MPI for Python. J Parallel Distrib Comput 2005;65(9):1108–15.
[33] Dalcin L, et al. MPI for Python: Performance improvements and MPI-2 extensions. J Parallel Distrib Comput 2008;68(5):655–62.
[34] Krishnamoorthy A. USCCACS/xtal: xtal is an umbrella package for various tools used to manipulate atomic trajectories. 2019 [cited 2019 October 28, 2019]; Available from: https://github.com/USCCACS/xtal.
[35] Xiaowang Z. Uncertainty quantification and reduction of molecular dynamics models. In: Stephen MF, Peter H Jan, editors. Uncertainty quantification and model calibration. Rijeka: IntechOpen; 2017, p. 89–111.
[36] Cailliez F, Pernot P. Statistical approaches to forcefield calibration and prediction uncertainty in molecular simulation. J Chem Phys 2011;134(5).
[37] Moore AP, et al. Understanding the uncertainty of interatomic potentials' parameters and formalism. Comput Mater Sci 2017;126:308–20.
[38] Angelikopoulos P, Papadimitriou C, Koumoutsakos P. Bayesian uncertainty quantification and propagation in molecular dynamics simulations: A high performance computing framework. J Chem Phys 2012;137(14):144103.
[39] Longbottom S, Brommer P. Uncertainty quantification for classical effective potentials: an extension to potfit. Modelling Simulation Mater Sci Eng 2019;27(4):044001.
[40] Frederiksen SL, et al. Bayesian ensemble approach to error estimation of interatomic potentials. Phys Rev Lett 2004;93(16):165501.
[41] Demirkan K, et al. Reactivity and morphology of vapor-deposited al/polymer interfaces for organic semiconductor devices. J Appl Phys 2008;103(3).
[42] Lazzaroni R, et al. The chemical and electronic-structure of the interface between aluminum and conjugated polymers. Electrochim Acta 1994;39(2):235–44.

[43] Faupel F, et al. Metal diffusion in polymers and on polymer surfaces. In: Gupta D, editor. Diffusion processes in advanced technological materials. Berlin, Heidelberg: Springer Berlin Heidelberg; 2005, p. 333–63.
[44] Heimel G, et al. The interface energetics of self-assembled monolayers on metals. Acc Chem Res 2008;41(6):721–9.
[45] Chen LH, et al. Charge injection barriers at metal/polyethylene interfaces. J Mater Sci 2016;51(1):506–12.
[46] Li Y, et al. Scalable reactive molecular dynamics simulations for computational synthesis. Comput Sci Eng 2018;1.
[47] Hong S, van Duin ACT. Atomistic-scale analysis of carbon coating and its effect on the oxidation of aluminum nanoparticles by ReaxFF-molecular dynamics simulations. J Phys Chem C 2016;120(17):9464–74.