


Machine Learning for Parameter Auto-tuning in Molecular Dynamics Simulations: Efficient Dynamics of Ions near Polarizable Nanoparticles

Journal Title
XX(X):1–15
©The Author(s) 2015
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/


JCS Kadupitiya¹, Geoffrey C. Fox¹ and Vikram Jadhao¹

Abstract

Simulating the dynamics of ions near polarizable nanoparticles (NPs) is extremely challenging due to the need to solve the Poisson equation at every simulation timestep. Recently, a molecular dynamics (MD) method based on a dynamical optimization framework bypassed this obstacle by representing the polarization charge density as virtual dynamic variables, and evolving them in parallel with the physical dynamics of ions. We highlight the computational gains accessible with the integration of machine learning (ML) methods for parameter prediction in MD simulations by demonstrating how they were realized in MD simulations of ions near polarizable NPs. An artificial neural network based regression model was integrated with MD and predicted the optimal simulation timestep and critical parameters characterizing the virtual system on-the-fly with 94.3% success. The integration of ML method with hybrid OpenMP/MPI parallelized MD simulations generated accurate dynamics of thousands of ions in the presence of polarizable NPs for over 10 million steps (with a maximum simulated physical time over 30 ns) while reducing the computational time from thousands of hours to tens of hours yielding a maximum speedup of ≈ 3 from ML-only acceleration and a maximum overall speedup of ≈ 600 from ML-hybrid Open/MPI combined method. Extraction of ionic structure in concentrated electrolytes near oil-water emulsions demonstrates the success of the method. The approach can be generalized to select optimal parameters in other molecular dynamics applications and energy minimization problems.

Keywords

Machine Learning, Nanoscale Simulations, Energy Minimization, Hybrid MPI/OpenMP, Parallel Computing, Auto-tuning

1 Introduction

Many biological and synthetic nanoparticle (NP) systems are polarized in the presence of electric fields generated by surrounding ions and other macromolecular charged species (Levin 2005; Clapham 2007; Abruja et al. 2008). Examples include proteins and DNA in an aqueous cellular medium, emulsions where oil and water are partitioned, and gold NPs dispersed in water. Accurate knowledge of ionic structure near the surface of these NPs enables the understanding of many nanoscale phenomena associated with these materials such as protein conformational changes (Honig and Nicholls 1995), DNA precipitation (Raspaud et al. 1998), spontaneous emulsification (Sacanna et al. 2007), and nanoparticle self-assembly (Levin 2005). Extracting this structure by simulating the dynamics of ions in the presence of polarizable NPs using coarse-grained models is challenging due to the need to compute polarization (induced) charges in order to propagate the ion configuration (Allen et al. 2001; Marchi et al. 2001; dos Santos et al. 2011; Fahrenberger et al. 2014). This computation typically involves solving the second-order Poisson differential equation in 3-dimensional space at each simulation timestep, making the use of conventional nanoscale simulation methods very time consuming and inefficient. Because of these computational challenges, the problem of extracting ionic structure near polarizable NPs has been a subject of intense research (Marchi et al. 2001;

Boda et al. 2004; Allen et al. 2001; dos Santos et al. 2011; Jadhao et al. 2012, 2013; Fahrenberger et al. 2014; Gan et al. 2015; dos Santos and Netz 2018).

The problem is often re-casted in terms of energy minimization for which different candidate functionals and associated minimization methods have been proposed (Marchi et al. 2001; Boda et al. 2004; Allen et al. 2001). Among these techniques, a molecular dynamics (MD) method based on the dynamical optimization of an energy functional enabled the replacement of the expensive solution of the Poisson equation at each simulation step with an on-the-fly computation of surface polarization charges (Jadhao et al. 2012, 2013; Jing et al. 2015). The main focus of this paper is to highlight the computational gains accessible by integrating machine learning (ML) methods for parameter auto-tuning in MD simulations by demonstrating how these gains were realized in the MD simulations of ions near polarizable NPs based on the aforementioned dynamical optimization framework.

¹Intelligent Systems Engineering, Indiana University, 2805 E. 10th Street, Bloomington, Indiana 47408, USA

Corresponding author:

Vikram Jadhao, Intelligent Systems Engineering, Indiana University, 2805 E. 10th Street, Bloomington, Indiana 47408, USA
Email: vjadhao@iu.edu

In the dynamical optimization framework, inspired by the Car-Parrinello method for simulating ion-electron systems (Car and Parrinello 1985), an energy functional of the polarization charge density was dynamically optimized resulting in the physical dynamics of ions in parallel with the update of the virtual variables characterizing the polarization (induced) charge density. The virtual system was evolved in a manner that kept the induced charges close to the free-energy minimum (“ground state”) corresponding to the evolving ionic configuration. The advantages associated with the on-the-fly computation of polarization effects in conjunction with the reduction in computational costs achieved by solving for the scalar induced density variable enabled the study of electrolyte ion solutions near polarizable NPs using this framework (Jadhao et al. 2012; Jing et al. 2015). However, the applicability of the original method was limited by the absence of a framework that automates the process of selecting the “good” parameters characterizing the virtual system as well as the optimal simulation timestep. These quantities determine the stability, accuracy, and overall efficiency of the dynamical optimization framework and they were found by a tedious process of trial and error that was informed by in-domain experience. Further, these parameters were selected at the start of the simulation and held fixed throughout the simulation, to often relatively conservative values, in order to ensure the long-time stability of the dynamics of ions.

Recent years have witnessed a remarkable growth in the use of ML to enhance computational methods aimed at understanding phenomena in materials science, biology, neuroscience, and physics (Bartók et al. 2017; Ch’ng et al. 2017; Balakrishnan and Puthusserypady 2005; Schoenholz 2018; Liu et al. 2017; Long et al. 2015; Ferguson 2017). ML has been applied to identify interesting parameter spaces (Spellings and Glotzer 2018), predict parameters (Balakrishnan and Puthusserypady 2005), update configurations (Botu and Ramprasad 2015), infer assembly landscapes (Long et al. 2015; Ferguson 2017), and classify phases of matter (Ch’ng et al. 2017). Inspired by these recent developments, we describe an approach to integrate ML with MD methods to predict and auto-tune relevant parameters and simulation timestep. This approach is applied to the dynamical optimization framework to predict on-the-fly the virtual system parameters and simulation timestep that keep the polarization charge density close to the ground state determined by the evolving ionic configuration at all times during the simulation. The demonstration of the use of ML to predict and tune the MD simulation timestep has broad applicability. Similarly, we expect that the idea of using ML for predicting virtual system parameters can be extended to enhance the original Car-Parrinello molecular dynamics techniques (Car and Parrinello 1985) for simulating ion-electron systems.

The use of ML to enhance the performance of the dynamical optimization framework is demonstrated using an $O(n^2)$ algorithm to propagate the dynamics of ions and virtual system variables that is accelerated by implementing a hybrid OpenMP/MPI parallelization approach to reduce the computing time associated with the evaluation of the forces and energies. The target applications of the framework are systems where the effects of NP surface charge and

ion correlations typically lead to ion distributions that reach constant bulk value within a few nanometers of the NP surface such that a comprehensive study of ion densities near NP surfaces can be performed by including thousands of ions in a large simulation cell with reflective boundaries (dos Santos et al. 2011; Messina 2002; Boda et al. 2004). Many synthetic and biological systems including oil-water emulsions, gold nanoparticles (1 - 10 nm in diameter), and globular proteins exhibit this scenario, and ion distributions in these systems have been analyzed using $O(n^2)$ methods that are competitive with $O(n \log n)$ methods for these moderately-sized systems (Allen et al. 2001; Boda et al. 2004; dos Santos et al. 2011; Messina 2002; Hatlo and Lue 2008; Jadhao et al. 2012). Further, attempts to ameliorate this scaling via the use of Ewald sums (Deserno and Holm 1998) or multigrid methods (Sagui and Darden 2001) would introduce more variables and parameters into the system making the assessment of the coupling of the ML-enabled parameter selection process with the simulation of ions and virtual system difficult. The use of such methods is thus avoided in this first study of developing ML-based enhancements for the treatment of polarizability effects in ionic soft-matter simulations; future work will include integrating this approach with fast Ewald solvers for incorporating long-range effects.

As we discuss later in the results section, the ML-enhanced dynamical optimization framework leads to an increase in both the efficiency and stability of the associated MD simulations, while retaining the accuracy of the unautomated, non-adaptive framework. This combination of ML and parallel computing in the context of nanoscale simulation of ions is the first of its kind and paves the way for employing the simulation framework for developing online applications for web-based platforms like nanoHUB (Klimeck et al. 2008), where the user interacts with the simulation software under limited interaction with the developer and/or in-domain expert. An application that simulates the self-assembly of ions near polarizable NPs by employing the unique features of this framework was recently deployed on the nanoHUB cloud (Kadupitiya et al. 2018). As is evident by the use of ML in numerous commercial platforms, scientific simulation workflow and software applications will increasingly employ an ML layer in the future. Understanding the integration of ML in scientific applications is thus critical; the work presented here contributes towards this goal.

2 Background and Related Work

2.1 Model and the Energy Functional

The problem of evaluating polarization effects in simulation of charged systems has been extensively explored by several research groups using different approaches (Marchi et al. 2001; dos Santos et al. 2011; Boda et al. 2004; Tyagi et al. 2010; Barros et al. 2014; Gan and Xu 2011; Jadhao et al. 2012; Wynveen and Bresme 2006; Allen et al. 2001). Explicit simulation of solvent (environment) and NPs is possible (Wynveen and Bresme 2006) using advanced computational techniques such as fast multipole methods and local electrostatics algorithms (Rottler and Maggs 2004); the role of solvent is also crucial in understanding properties of

ion dynamics at the molecular scale [Beckstein et al. \(2004\)](#). However, many phenomena can not be suitably investigated using fully atomistic models due to the prohibitively large number of degrees of freedom associated with such systems. This has led to the study of coarse-grained models that treat ions explicitly but replace the molecular structure of the solvent and the NP with continuous dielectric environments. Systems where the different material parts are adequately captured by piecewise-uniform dielectric permittivities (e.g. NP and solvent, protein and cellular medium) have attracted particular attention ([Allen et al. 2001](#); [Boda et al. 2004](#); [dos Santos et al. 2011](#); [Xu 2013](#); [Jadhao et al. 2012, 2013](#); [Barros and Luijten 2014](#); [Tyagi et al. 2010](#)). For these model systems, solving for the polarization (induced) charge density reduces the computational costs because the unknown induced charge density resides only on the two-dimensional interface (boundary) between the NP and the surrounding medium. We work with such a coarse-grained model.

The dynamical optimization framework for extracting ion distributions from simulations of the coarse-grained model that treats the solvent and NP as dielectric continua is based on the true energy functional of induced charge density introduced in [Jadhao et al. \(2012\)](#):

$$\begin{aligned} \mathcal{F}[\omega] = & \frac{1}{2} \iint \rho_{\mathbf{r}} G_{\mathbf{r}, \mathbf{r}'} (\rho_{\mathbf{r}'} + \Omega_{\mathbf{r}'}[\omega]) d\mathbf{r}' d\mathbf{r} \\ & - \frac{1}{2} \iint \Omega_{\mathbf{r}}[\omega] G_{\mathbf{r}, \mathbf{r}'} (\omega_{\mathbf{r}'} - \Omega_{\mathbf{r}'}[\omega]) d\mathbf{r}' d\mathbf{r}, \end{aligned} \quad (1)$$

where ρ and ω are the ion and induced charge densities respectively. The function $G(\mathbf{r}, \mathbf{r}') = |\mathbf{r} - \mathbf{r}'|^{-1}$ is the bare Green's function and Ω is given by

$$\Omega_{\mathbf{r}}[\omega] = \nabla \cdot \left(\chi_{\mathbf{r}} \nabla \int (\rho_{\mathbf{r}'} + \omega_{\mathbf{r}'} d\mathbf{r}') \right), \quad (2)$$

where $\chi(\mathbf{r})$ is the dielectric susceptibility. $\chi(\mathbf{r})$ is related to the spatially-varying dielectric permittivity $\epsilon(\mathbf{r})$ via the relation $\epsilon = 1 + 4\pi\chi$. The minimization of $\mathcal{F}[\omega]$ leads to the equation:

$$\omega = \Omega. \quad (3)$$

Solving this equation is equivalent to solving the Poisson equation; its solution produces the correct induced density ([Jadhao et al. 2012, 2013](#)). At its minimum, $\mathcal{F}[\omega]$ evaluates to the true electrostatic energy of the system. These features allow $\mathcal{F}[\omega]$ to be optimized dynamically as the ions move to their new positions.

The functional $\mathcal{F}[\omega]$ can be transformed into a functional of only the surface (two-dimensional) induced charge density for the case of polarizable NP in a solvent where the NP and the solvent are modeled as materials of different, but uniform, permittivities ([Jadhao et al. 2012, 2013](#)). The discretized form of this transformed functional obtained by meshing the NP surface into M finite elements is given as:

$$\begin{aligned} \mathcal{F}[\{\omega_k\}] = & \frac{1}{2} \sum_{i=1}^N \sum_{j \neq i}^N q_i \overset{\circ}{\overset{\circ}{K}}_{\mathbf{r}_i, \mathbf{r}_j} q_j + \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^M q_i \overset{\circ}{\overset{\circ}{K}}_{\mathbf{r}_i, \mathbf{s}_k} \omega_k a_k \\ & + \frac{1}{2} \sum_{k=1}^M \sum_{l=1}^M \omega_k \overset{\circ}{\overset{\circ}{K}}_{\mathbf{s}_k, \mathbf{s}_l} \omega_l a_k a_l, \end{aligned} \quad (4)$$

where ω_k , \mathbf{s}_k , and a_k are, respectively, the induced charge, position vector, and area associated with the k^{th} finite

element. Here, N is the total number of ions, and q_i and \mathbf{r}_i are the charge and position vector of the i^{th} ion respectively. The terms $\overset{\circ}{\overset{\circ}{K}}$, $\overset{\circ}{\overset{\circ}{K}}$, and $\overset{\circ}{\overset{\circ}{K}}$ in (4) are the effective potentials of interaction between two ions, between an ion and an induced charge, and between two induced charges; explicit expressions of these functions can be found in the original papers ([Jadhao et al. 2012, 2013](#)). $\mathcal{F}[\{\omega_k\}]$ can be minimized dynamically using MD methods that treat the induced charges on the surface as dynamic variables; the details of this dynamical optimization framework are provided in Section 3.

2.2 Nanoscale Simulation of Ions near Polarizable Materials

We review the techniques of computing ion distributions in systems described by the coarse-grained model of ions near NP-solvent dielectric interface separating NP and solvent that exhibit different (but uniform) dielectric response (permittivities) ([Boda et al. 2004](#); [Tyagi et al. 2010](#); [Allen et al. 2001](#); [Barros et al. 2014](#); [Jadhao et al. 2012, 2013](#); [Jing et al. 2015](#)), focusing on methods that are broadly applicable and are not limited by the choice of NP geometry or dielectric permittivity profile.

We first outline the methods based on variational approaches to the problem of evaluating the polarization effects as these techniques are most closely related to the work presented here. In this approach, one transforms the original problem of solving the Poisson differential equation into an optimization problem. A variety of functionals employing various electrostatic quantities as field variables have been proposed to formulate the variational optimization problem ([Jackson 1999](#); [Marcus 1956](#); [Felderhof 1977](#); [Reiner and Radke 1990](#); [York and Karplus 1999](#); [Allen et al. 2001](#); [Attard 2003](#); [Rottler and Maggs 2004](#); [Lipparini et al. 2010](#); [Villasenor and Buneman 1992](#); [Nakano et al. 1994](#)). [Allen et al. \(2001\)](#) performed an explicit (static) optimization of a functional of the induced surface charge density $\omega(\mathbf{s})$ at each MD step to solve the Poisson equation and propagate ions. [Marchi et al. \(2001\)](#) worked with a true energy functional of the polarization vector and implemented a dynamical optimization framework to propagate ion dynamics in parallel with the evaluation of polarization vector fields. However, the choice of the polarization vector as the variable field needed a three-dimensional specification leading to increased computational costs that can be avoided by choosing $\omega(\mathbf{s})$ as the variable to solve for.

Another class of methods for computing $\omega(\mathbf{s})$ involves transforming the problem into a matrix formulation; examples include the induced charge computation (ICC) methods ([Boda et al. 2004](#)), which use matrix inversions to solve for $\omega(\mathbf{s})$. Evaluating matrix inversion involves $O(M^3)$ calculations where M is the number of surface mesh elements. Techniques to improve upon this scaling have been subsequently developed ([Tyagi et al. 2010](#)). Alternatively, iterative methods to solve the matrix equation have been proposed ([Barros and Luijten 2014](#)). In particular, the generalized minimum residual method solves the matrix equation without explicitly constructing the inverse matrix and yields a converged result for $\omega(\mathbf{s})$ at each simulation timestep within 4 - 5 iterations ([Gan et al. 2015](#)).

The evaluation of $\omega(\mathbf{s})$ in all the above approaches needs the ionic configuration to be static and requires considerable computational effort, whether in the form of matrix inversions or convergence of the iterative optimization steps, at each simulation step to guarantee the overall stability of the simulation. In Section 3, we present the details of a recently developed dynamical optimization framework that enables the simultaneous (on-the-fly) updates of $\omega(\mathbf{s})$ and the ionic configuration in the same simulation step.

2.3 Parameter Prediction using Machine Learning

Machine Learning (ML) abstractions for parameter prediction and tuning have been extensively employed in the performance enhancement of bigdata or deep learning frameworks. Denil et al. (2013) used ANN and convolutional deep learning NN to predict the parameters found in image classification tasks. The ANN was able to obtain an accuracy of 95%. Yigitbasi et al. (2013) employed ML-based auto-tuning for diverse MapReduce applications and cluster configurations in Hadoop framework. Their work showed that support vector regression exhibits good accuracy while being computationally efficient for performance modeling of MapReduce applications.

ML methods have been increasingly employed in predicting parameters and classifying properties associated with physical systems (Spellings and Glotzer 2018; Schoenholz 2018; Liu et al. 2017; Morningstar and Melko 2017; Ch'ng et al. 2017; Behler 2016; Botu and Ramprasad 2015; Long et al. 2015; Ferguson 2017). Spellings and Glotzer (2018) applied a simple feedforward artificial neural network (ANN) to discover interesting areas of parameter space corresponding to crystal formation in the self-assembly of colloidal building blocks. Balakrishnan and Puthusserypady (2005) proposed a fast and simple prediction method for two class Brain-Computer Interface (BCI) simulations using ANN and identified this method as the best ML approach for brain image classification.

In addition to classification based approaches, regression based prediction schemes have been employed in different domain areas (Eng et al. 2014; Kazemi and Sullivan 2014; Cherkassky and Ma 2004; Chen and Yu 2014; Balachandran et al. 2016; Quan et al. 2014; Yadav et al. 2016). Eng et al. (2014) used random forest regression algorithm to predict host tropism of influenza A virus proteins with an accuracy above 96 %. Similarly, ensemble of regression trees were employed to perform face alignment for real-time applications (in one millisecond) by Kazemi and Sullivan (2014). Support vector machine based regression (SVR) has been used for wind speed prediction by Chen and Yu (2014). Balachandran et al. (2016) have also used SVR to create an adaptive ML model to aid the design of a new materials with desired elastic properties and enhanced long-term performance using minimum number of iterations. ANN based regression has been studied by Quan et al. (2014) to yield short term load prediction of electrical power systems based on wind power forecasting. Yadav et al. (2016) have employed ANN based regression for forecasting solar radiation.

Recent research has also focused on exploring ML based approaches to generate trajectories (configurations) in MD and Monte Carlo simulations in order to enhance the performance of the respective methods. Botu and Ramprasad (2015) employed kernel ridge regression (KRR) to accelerate the *ab initio* MD method for nuclei-electron systems by learning the selection of probable configurations in MD simulations (Botu and Ramprasad 2015). (Liu et al. 2017) employed an ANN to select efficient updates for Monte Carlo simulations of classical Ising spin models.

These explorations have inspired us to use ML to predict virtual system parameters and design an adaptive framework that updates the simulation timestep and auto-tunes the virtual parameters characterizing the dynamics of ions near polarizable NPs to yield a more stable and efficient MD simulation. Related work in the area of adapting timestep in a simulation has largely focused on using analytical approaches to multiple time step integration (Luehr et al. 2014; Tuckerman et al. 1992). Recent work has also focused on adaptive ensemble simulations to enhance the computational efficiency of biomolecular simulations Kasson and Jha (2018). We also note that the development of autotuning technology for high-performance applications, which involves automatic generation of a search space of possible kernels for a computational task to identify the best possible kernel, has been extensively explored to reduce execution time and enhance programmer productivity (Whaley and Dongarra 1998), with recent work involving the use of ML-based approaches for identifying the search space (Balaprakash et al. 2018). In Section 4, we describe the results of our experiments with different regression-based ML models to identify and tune optimal virtual parameters for the dynamical optimization framework.

3 Dynamical Optimization Framework for Simulating Ions near Polarizable NPs

In this section, we provide the details of the dynamical optimization framework for simulating ions in the presence of polarizable NPs. This framework uses Car-Parrinello molecular dynamics (CPMD) technique (Car and Parrinello 1985; Fois et al. 1993) to dynamically optimize an energy functional of the polarization charge density which results in the propagation of the ionic configuration in tandem with an accurate update of the polarization charges (Jadhao et al. 2012, 2013). These details will help clarify the use of the ML-based enhancement strategies outlined in Section 4.

3.1 Extended Lagrangian, Equations of Motion and CPMD Simulation

To implement the dynamical optimization of $\mathcal{F}[\{\omega_k\}]$, the induced charges $\{\omega_k\}$ are treated as dynamic virtual variables. A fictitious kinetic energy is associated with this virtual system:

$$\mathcal{K} = \sum_{k=1}^M \frac{1}{2} \mu_k \dot{\omega}_k^2, \quad (5)$$

where μ_k is the mass of the k^{th} virtual variable (ω_k). The extended Lagrangian \mathcal{L} with $\mathcal{F}[\{\omega_k\}]$ as its electrostatic potential energy is constructed by including \mathcal{K} as an

additional term:

$$\mathcal{L} = \mathcal{H} + \sum_{i=1}^N \frac{1}{2} m_i \dot{\mathbf{r}}_i^2 - \mathcal{F}[\{\omega_k\}] - \mathcal{H}[\{\mathbf{r}_i\}]. \quad (6)$$

In (6), the second term is the usual total kinetic energy associated with N ions (physical system), with m_i being the mass of the i^{th} ion. The final term contains a set of Lennard-Jones potentials to model the ion-ion and ion-NP steric interactions. Note that $\mathcal{F}[\{\omega_k\}] \equiv \mathcal{F}[\{\omega_k\}, \{\mathbf{r}_i\}]$ is also a function of the set of ion positions $\{\mathbf{r}_i\}$.

The Lagrangian \mathcal{L} yields the following Euler-Lagrange equations of motion:

$$m_i \ddot{\mathbf{r}}_i = -\nabla_{\mathbf{r}_i} \mathcal{F}[\{\omega_k\}, \{\mathbf{r}_i\}], \quad (7)$$

$$\mu_k \ddot{\omega}_k = -\nabla_{\omega_k} \mathcal{F}[\{\omega_k\}, \{\mathbf{r}_i\}], \quad (8)$$

for the i^{th} ion and the k^{th} induced charge, respectively. These equations are used to evolve the induced charge configuration *on the fly* using the CPMD method. Following (7), each ion is moved by the force $-\nabla_{\mathbf{r}_i} \mathcal{F}[\{\omega_k\}; \mathbf{r}_i]$ in a timestep Δ during which each induced charge is updated via the force $-\nabla_{\omega_k} \mathcal{F}[\{\omega_k\}; \mathbf{r}_i]$ following (8). To simulate the behavior of the ions at temperature T , the extended system of ions and the virtual variables is coupled to a set of Nosé-Hoover thermostats (this coupling modifies the equations of motion (7) and (8) similar to a canonical MD routine). This two-temperature approach is a standard feature of CPMD (Sprik 1991; Blöchl and Parrinello 1992; Fois et al. 1993). The ions couple to a thermostat at temperature T , while the virtual system is coupled to one at T_v .

Velocity-Verlet algorithm is used to generate the dynamics of the extended system. The dynamics is associated with a conserved quantity, the total energy of the extended system, that can be derived from the Lagrangian \mathcal{L} :

$$\mathcal{E} = \sum_{i=1}^N \frac{1}{2} m_i \dot{\mathbf{r}}_i^2 + \mathcal{H} + \mathcal{F}[\{\omega_k\}] + \mathcal{H}[\{\mathbf{r}_i\}] + \mathcal{T} + \mathcal{T}_v. \quad (9)$$

Here, \mathcal{T} and \mathcal{T}_v are the energy terms associated with the thermostats controlling the temperature of the physical and virtual systems respectively. The extended energy and \mathcal{H} are monitored at periodic intervals during a CPMD simulation to assess the stability and accuracy of the simulation.

Virtual masses μ_k are chosen to be proportional to the areas of the mesh points. The value of the proportionality constant μ depends on the attributes of the physical system (e.g., NP charge, dielectric profile, ion valencies) as well as the simulation timestep Δ . The parameters T_v and μ are optimized to ensure the stability and accuracy of the simulation (see Section 3.3); further technical details of the method can be found in Jadhao et al. (2012).

3.2 OpenMP/MPI Hybrid Parallelization

A system with N ions near an unpolarizable NP surface effectively translates into a system with M additional dynamical variables in the case of a polarizable NP within the dynamical optimization framework. Due to the long-range nature of the electrostatic interaction, the associated computational costs scale roughly as $O((N + M)^2)$, with a prefactor that can be large owing to the

complexity of the terms involved in the expressions for forces derived from $\mathcal{F}[\{\omega_k\}]$. Indeed, performance profiling report generated using Performance Counters for Linux (PERF) showed that the sequential program spends the largest amount of computation time (64.33% of the total) calculating the forces between the ions for each step of the simulation. To reduce the computing time associated with the evaluation of these forces and enhance the performance of the simulation framework, a hybrid OpenMP/MPI parallel programming model is adopted. The hybrid model has advantages over pure MPI or pure OpenMP, when cache performance is taken into consideration. This strategy provides non-uniform memory access (NUMA) traffic and inter-node communication (Rabenseifner et al. 2009) to support maximum access locality and minimum number of cache misses. We note that the simulations associated with the dynamical optimization framework presented in the previous publications employed the OpenMP (shared memory) parallelization model, and consequently, were limited in their application scope (Jadhao et al. 2012, 2013; Jing et al. 2015).

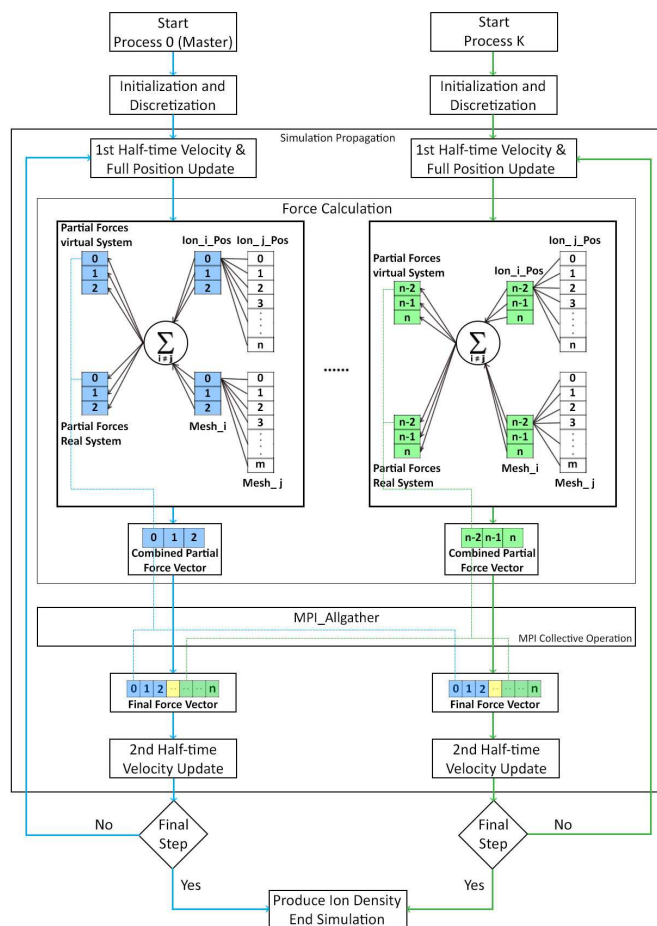


Figure 1. Distributed and shared memory parallelization approach with MPI and OpenMP. Hybrid parallelization is implemented inside the Force Calculation block.

The hybrid masteronly model is implemented by combining the distributed memory MPI approach and the shared memory OpenMP approach (Rabenseifner et al. 2009), and is applied to the force and energy calculation subroutines in the dynamical optimization framework. The

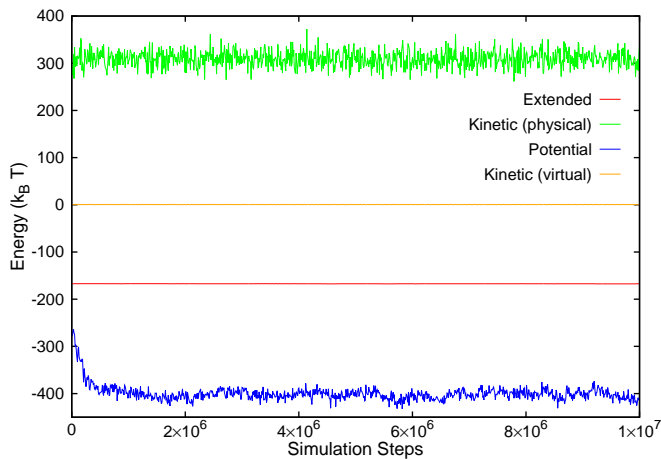


Figure 2. Energy profiles of 206 ions near a polarizable NP (whose surface is meshed with 1082 grid points). Data is shown for ≈ 10 nanoseconds (10 million simulation steps) of simulation time. Conservation of the extended (total) energy and ≈ 0 value for the virtual kinetic energy highlight the first key feature of the dynamical optimization framework.

model uses one MPI process per node and OpenMP on the cores of the node, with no MPI calls inside the parallel regions. The domain decomposition is enabled under a two-level mechanism. On the MPI level, a coarse-grained domain decomposition is performed using boundary conditions as explained in Fig. 1. The second level of domain decomposition is achieved through OpenMP loop level parallelization inside each MPI process.

3.3 Key Framework Features

We identify two key features of the dynamical optimization framework that encode the accuracy and stability of the simulation, and guide the process of designing the ML-based enhancement strategies presented in Section 4. The first key feature is the conservation of the extended energy \mathcal{E} , given by Eq. (9), and the approximate conservation of the energy of the physical system that is captured by demanding that the kinetic energy of the virtual system nearly vanishes:

$$\mathcal{K} \approx 0. \quad (10)$$

In other words, the framework ensures that the physical system remains unaffected as much as possible by the presence of virtual system.

The energy profiles of a typical, successful CPMD simulation of ions near a polarizable, spherical NP at room temperature are shown in Fig. 2: the extended energy \mathcal{E} is constant and the total fictitious kinetic energy stays stable and close to 0 throughout the entire simulation run (for ≈ 10 ns). In practice, this feature is incorporated in the simulation by appropriately choosing values of simulation timestep Δ , virtual variable mass μ , and the virtual system temperature $T_v \ll T$ ($T_v \approx 0$). These parameters are selected to control large, abrupt rise in the kinetic energy associated with the virtual system as the simulation progresses.

This feature is encoded in the quantity R which measures the ratio of the fluctuations in \mathcal{E} and the fluctuations in the kinetic energy of the physical system. R determines the stability and accuracy of the simulation. For good

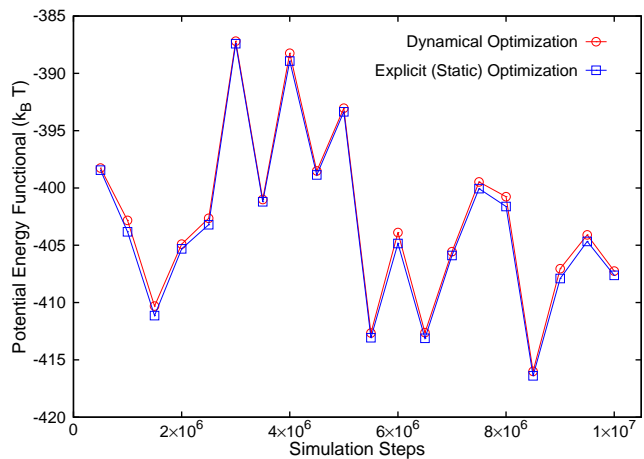


Figure 3. Comparison of the functional optimized dynamically (circles) and the functional optimized at regular intervals keeping the ionic configuration static during the optimization process (squares). Results are shown for the same system as in Fig. 2. The matching of the two functionals illustrates the second key feature of the framework: the accurate tracking of the induced charge density.

energy conservation (constant \mathcal{E}), it is demanded that the simulations satisfy the condition $R < 0.05$ as noted in the literature (Marchi et al. 2001). The latter inequality implicitly satisfies the requirement that \mathcal{K} is kept close to the value dictated by the low temperature T_v .

The second important feature considers the effectiveness of the framework to reproduce the induced charge distribution accurately at each simulation time step. At regular intervals during the course of the simulation, the ion coordinates and induced charge densities on the NP surface are stored. Then, an ordinary (static) minimization of the functional \mathcal{F} is carried out to explicitly determine the (numerically) exact induced density. The tracking of the induced density distributions on the NP surface can be assessed by evaluating the matching of \mathcal{F} optimized on the fly with the electrostatic energy value obtained by optimizing the functional explicitly; Fig. 3. This functional matching is the second key feature. In practice, we compute the functional deviation, f_d , which measures the average difference between the dynamically optimized functional \mathcal{F} and the energy functional obtained via direct (static) minimization. To pass the test of stability and accuracy, we enforce $|f_d| < 1\%$.

R and f_d are central to the success of the simulations based on this framework and determine the associated “good” virtual system parameters μ and T_v . In general, higher μ leads to better energy conservation and lower R , while lower T_v keeps the virtual system from excessive heating and generates lower f_d . Having just these two features biases the prediction of $\mu(T_v)$ towards higher (lower) values for a system of ions characterized with a generic input parameter pattern. Very high values of μ and/or very low values of T_v can affect the overall stability of the simulation as the virtual system can be prohibitively slow (due to “heavy” virtual masses and “cooler” associated temperature) to react to the evolving ionic configuration, resulting in inaccurate induced charge updates. An experienced domain expert would typically avoid these choices. To enhance

the stability of the simulation and bias the selection of the virtual parameters towards those picked by a domain expert, another quantity, R_v is introduced. R_v is the ratio of the fluctuations in \mathcal{E} and the fluctuations in \mathcal{K} . Lower R_v implies that fluctuations in \mathcal{K} (that can arise from lower virtual mass μ or higher T_v) are sufficiently strong to endow the virtual system with the necessary dynamics to adapt to the evolving ionic system. Unlike R and f_d that exhibit universal bounds informed by the physical dynamics of ions, the bound on R_v depends on the set of systems investigated, and is informed largely by past domain experience. For counterion-only systems used as training set for the ML-based methods, $R_v < 0.15$ is enforced. For systems characterized with electrolytes that are expected to exhibit a greater number of ions (with both positive and negative valencies), R_v can assume larger values depending on the salt concentration.

Quantities R , f_d , and R_v determine the choice of the optimal virtual parameters. However, these quantities and the success of the simulation depend critically on another important parameter: the simulation timestep Δ . The simulation timestep in the CPMD-based dynamical optimization framework depends on both the physical and virtual system parameters, the latter being unknown *a priori*. Conversely, the optimal values of virtual parameters (μ , T_v) that ensure a long-time stable simulation are dependent on Δ . This complicates the process for choosing a reliable, yet efficient, value for Δ , μ , and T_v and one typically chooses a conservative Δ that is smaller than the value used in conventional MD simulations ($\Delta \approx 1 - 5$ femtoseconds for an MD simulation of monovalent electrolyte ions in water at room temperature).

4 ML-based Enhancement of the Dynamical Optimization Framework

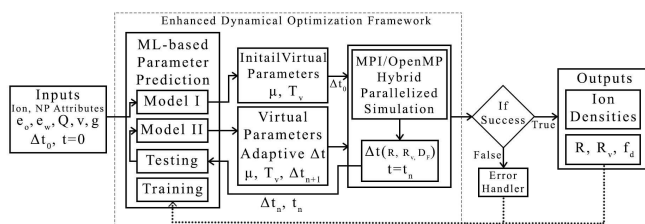


Figure 4. System overview of the enhanced dynamical optimization framework.

We now describe an ML-based procedure to increase the efficiency and improve the stability of the dynamical optimization framework while retaining the simulation accuracy. We present an ML technique that uses the aforementioned key features encoded in quantities R , f_d , and R_v to enhance the performance of the dynamical optimization framework by 1) predicting and auto-tuning the optimal virtual parameters μ and T_v , and 2) adapting the timestep to the largest allowable value during the simulation. The ML technique is combined with MPI/OpenMP Hybrid parallel programming model, described in Section 3, to carry out the simulation.

Figure 4 shows the overview of the enhanced framework. ML-based parameter prediction was implemented using two

ML models (ML model I and II). First, the ion and NP model attributes, as well as the initial timestep $\Delta = \Delta t_0$, were fed to ML model I to predict the initial virtual system parameters μ and T_v . The predicted parameters and Δt_0 were used to start the simulation that was parallelized using the MPI/OpenMP hybrid programming model. At intermediate times t_n during the simulation, ML model II was used to predict the new timestep Δt_{n+1} and associated virtual parameters that continue the simulation for the subsequent time block (t_n, t_{n+1}). The ion distributions near the polarizable NP were sampled during the simulation run and the ion densities were stored as simulation output. ML model II also checks if the simulation is successful up to that point in time before dynamically tuning the parameters for next iterations to come. If simulation had failed due to imposed R , f_d , and R_v criteria, program will abort and call the error handler to display appropriate error messages. In addition, during the simulation, the quantities R , f_d , and R_v were computed and saved as output for retraining both ML models after a set number of simulation runs are executed. For every 1000 new simulation runs, both models were retrained.

After reviewing and experimenting with many ML techniques for parameter tuning and prediction including polynomial regression, support vector regression (SVR), decision tree regression, and random forest regression (Section 2.3 and Section 5.1), the artificial neural network (ANN) was adopted for enhancing the dynamical optimization framework. Figure 5 shows the details of the ANN-based ML model II employed to predict the virtual system parameters and the adaptive timestep for simulations based on the framework; the ANN was trained to select the largest allowable Δ that satisfies the tests of stability and accuracy encoded in the ML features. ML model I exhibits a similar process but is trained without the time and timestep parameters. The data preparation and preprocessing techniques, feature extraction and regression techniques as well as their validation for both models are discussed below.

4.1 Data Preparation and Preprocessing

Counterion-only systems (no added electrolyte) were considered for generating the training set. The polarization effects are expected to be strongest in these systems as added electrolyte screens the ion-NP electrostatic interactions. Further, counterion-only systems are relatively smaller and enable a broader exploration of parameter space to train the ML models. Interestingly, as we discuss later, results from counterion-only systems were employed to successfully extract and infer ionic distributions associated with electrolyte systems for up to $O(0.1)$ M concentration, exhibiting the transfer learning aspects of the ML-based procedure employed.

Prior domain experience and backward elimination using the adjusted R squared was considered for creating the training data set. Using this process, 5 input parameters that significantly affect the polarization charges on the NP surface were identified: NP permittivity ϵ_{NP} , solvent permittivity ϵ_S , NP charge Q , counterion valency v , and NP mesh size M . While the temperature of the physical system and the size of the NP and ions affect the polarization charges and associated ionic distributions, in

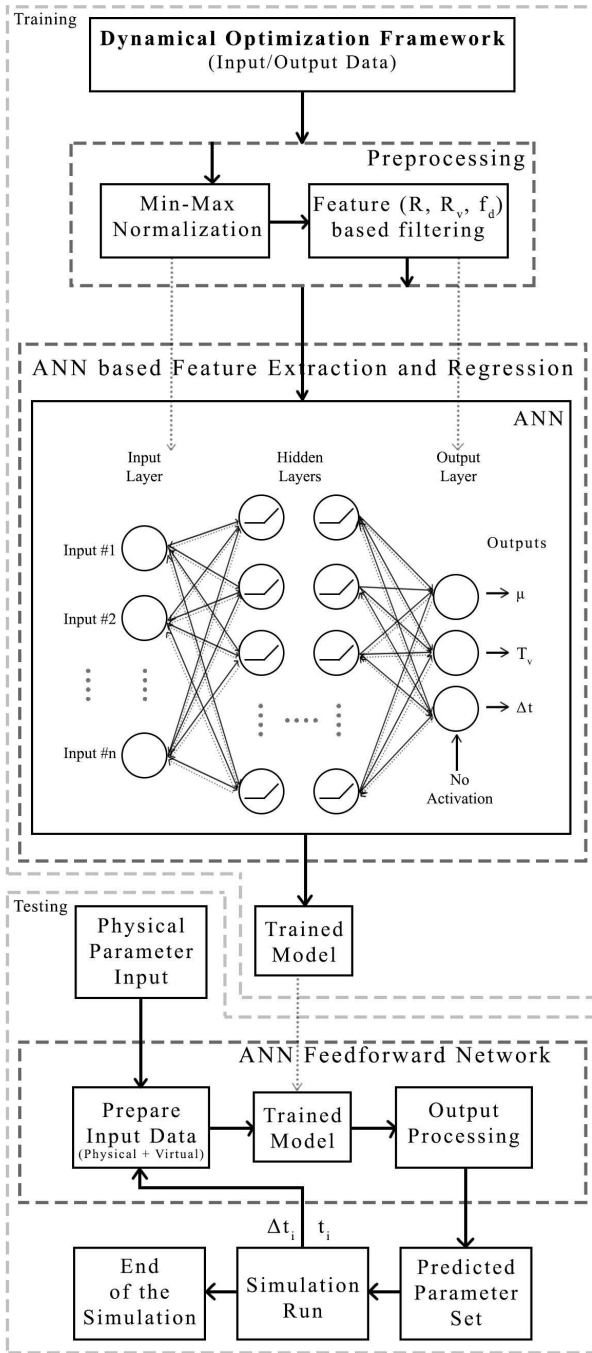


Figure 5. ML procedure (model II) for determining the parameters of the virtual system and the adaptive timestep defined in the dynamical optimization framework.

this initial study, these were considered fixed; NP diameter was taken to be ≈ 7.5 times the ion diameter (≈ 0.35 nanometers), and temperature was fixed at 298 K. Despite the aforementioned potential for transfer learning associated with the ML procedure, additional parameters such as salt concentration and co-ion attributes should be included in the training set to predict the optimal parameters associated with the simulation of electrolyte systems; future work will explore the training with these additional input parameters. Virtual parameter mass μ and virtual system temperature T_v were selected as the output parameters. Few discrete values for each of the input/output parameters were experimented with and swept over to create and run 13600

simulations for training the ML model I. The range for different ionic system parameters were selected based on physically meaningful and experimentally-relevant values: $\epsilon_{NP} \in (2, 160)$; $\epsilon_s \in (2, 160)$; $Q \in (-20, -100)$; $v \in 1, 2, 3$. For the mesh size and the virtual system parameters, the range were chosen based on previous trial and error procedure: $M \in (132, 1692)$, μ was swept from 1 to 40 using random discrete values to cover the range, and T_v was swept from 0.001 to 0.005. All the simulations were performed for ≈ 1 nanoseconds.

To support on-the-fly tuning of Δ and associated selection of μ, T_v during the simulation, ML model II was trained with two additional parameters. Simulation time $t \equiv t_n$ and timestep $\Delta \equiv \Delta t_{n+1}$ were added as input and output parameters respectively to the system parameters explored in ML model I. 20 discrete simulation time values $t_n \approx 0.1, 0.2, \dots, 2$ ns, and 4 discrete timestep values $\Delta = 0.001, 0.002, 0.003, 0.004$, were swept to generate 54400 simulation configurations.

As described in Section 3.3, R, f_d , and R_v encode the key features of energy conservation and accurate tracking of the induced charges that measure the success of the dynamical optimization framework. Acceptable threshold values 0.05, 1 percent, and 0.15 were identified for R, f_d , and R_v respectively for the range of systems included in the training set. These quantities were treated as output features to filter the datasets to only keep the input parameter configurations resulting in successful simulation runs. From the data set for initial parameter prediction, 4530 input/output configurations were selected as successful. From the data set for adaptive timestep prediction, 15640 input/output configurations were selected. Each of these datasets were separated as training and testing using a ratio of 0.7:0.3. Min–max normalization filter was applied to normalize the input data in the data preprocessing stage.

4.2 Feature extraction and Regression

The ANN algorithm with two hidden layers (Fig. 5) was implemented in Python for regression of two continuous variables in ML model I, and for regression of three continuous variables in ML model II. In both models, outputs of the hidden layers were wrapped with the *relu* function; the latter was found to converge faster compared to the sigmoid function. No wrapping functions were used in the output layers of the algorithm.

By performing a grid search, hyper-parameters such as the number of first hidden layer units, second hidden layer units, batch size, and number of epochs were optimized to 13, 8, 25, and 150 respectively. Adam optimizer was used as the back propagation algorithm. The weights in the hidden layers and in the output layer were initialized to random values using a normal distribution at the beginning. The mean square loss function was used for error calculation in both ML models. To stop overtraining the network, a drop out mechanism for hidden layer neurons during the training time was employed. ANN implementation, training and testing was programmed with the aid of keras and sklearn ML libraries (Chollet et al. 2015; Buitinck et al. 2013; Abadi et al. 2016).

5 Results and Discussion

5.1 Initial Virtual Parameter Prediction

Several regression models were implemented and tested to predict the initial virtual parameters μ and T_v . These models were tested on 1359 input parameter sets comprising of values within the range for which the models were trained for. Additionally, the models were tested on 450 completely random input sets of parameters selected without any range restrictions. Table 1 shows the success rate and mean square error for testing data sets as well as random input data sets. Success rate was calculated based on R , f_d , and R_v . Reported mean square error (MSE) values are calculated using k-fold cross-validation techniques with $k = 10$. ANN based regression model was able to predict the initial virtual system parameters correctly with 94.3% success rate (MSE of 1.56), outperforming other non-linear regression models as evident from the Table 1. ANN based regression model was also able to achieve success rate of 87.2 on completely random input parameters. This ANN based regression model was adopted as ML model I.

Table 1. Comparison of the regression models for initial virtual parameter prediction (numbers are percentages)

Model	Testing Sets		Random Sets
	Success %	MSE	Success %
Polynomial	45.7	12.25	16.5
Support Vector	78.3	4.56	58.4
Decision Tree	70.4	6.93	48.1
Random Forest	75.3	4.88	55.1
ANN based	94.3	1.56	87.2

Table 2 shows the predicted μ and T_v for selected systems along with the quantities R , f_d , and R_v that characterize the key features of the framework, energy conservation and tracking of induced charges, that determine the accuracy and stability of the simulation. The predicted μ and T_v values produced stable dynamics of ions near polarizable NP as evidenced by the values of R , R_v , and f_d that lie within allowed ranges ($R < 0.05$, $|f_d| < 1\%$, $R_v < 0.15$).

Table 2. Predicted parameters by ML model I and simulation accuracy and stability

Inputs	Prediction		Results		
	μ	T_v	R	f_d	R_v
$\epsilon_{NP}, \epsilon_S, Q, v$					
2, 10, -60, 1	9	0.001	0.001	-0.01	0.12
2, 78.5, -30, 3	7	0.002	0.003	-0.6	0.13
50, 78.5, -60, 2	18	0.001	0.001	-0.4	0.06
80, 160, -90, 3	30	0.002	0.002	-0.7	0.09
100, 120, 30, 2	36	0.005	0.002	-0.1	0.10
5, 71, -24, 2	42	0.001	0.007	-0.6	0.13
44, 37, -114, 1	38	0.006	0.002	-0.11	0.11
30, 35, -108, 3	43	0.007	0.002	-0.12	0.05
15, 78, -102, 1	17	0.025	0.005	-0.27	0.11

We note that when the ANN was trained utilizing only R and f_d quantities, higher μ and lower T_v values were predicted as expected from the arguments presented in

Section 3.3. These virtual parameter choices are not optimal for the stability of the simulation and will not be picked by the experienced, in-domain expert. Inclusion of R_v as another feature for training the model enabled the ANN to predict the virtual system parameters that were likely to enhance the stability of the simulation and be selected by a domain expert.

5.2 Auto-tuning CPMD Simulation Parameters

Similar to ML model I, ML model II employed the ANN based regression model trained with two additional parameters: simulation timestep Δ and time t . This model was trained to select the largest allowed Δ and auto-update the associated optimal virtual parameters μ, T_v at t_n for the simulation during the interval (t_n, t_{n+1}) based on the ML output features R, f_d , and R_v computed in the previous time interval (t_{n-1}, t_n) . Figure 6 illustrates the computational gains resulting from the auto-tuning of Δ using ML model II for systems of counterions near a nanoparticle (NP). NPs of different dielectric permittivity (2, 40 and 60) in water (dielectric permittivity 78.5) are considered. The effect of varying ion valency ($v = 1, 2$) is also probed. Other input parameters are NP charge $Q = -100$, NP mesh size $M = 1272$, and $\epsilon_S = 78.5$. Symbols indicate the gains associated with the enhanced framework with adaptive timestep, dashed line is the result from the non-adaptive simulation with static timestep. Compared to the simulation with non-adaptive Δ , the auto-tuning of Δ extended the simulation of the ionic system to a longer physical time for the same number of simulation steps; a speedup of $1.5 - 3\times$ was observed depending on the system configuration, also see Table 3. For the system with $\epsilon_{NP} = 2, v = 1$, the auto-tuning yielded a total simulated physical time of 4.5 ns compared to the 2.2 ns obtained with non-adaptive simulation. Figure 6 (inset) shows the variation in Δ as a function of the computational steps for the same systems. The tuning of Δ changed with the attributes of the ions and nanoparticle. Generally, longer Δ values (and associated higher speedup) were obtained for systems exhibiting a weaker dielectric contrast.

Table 3 quantifies the speed by showing the performance comparison of the aforementioned ion-NP systems for two million time steps on 4 MPI nodes each with 16 OMP threads and a fixed walltime of ≈ 20 hours. ML-based adaptive timestep tuning enabled the simulation of the system for a longer physical time under fixed compute resources and walltime. A maximum speedup of $3.02\times$ was achieved for the ion-NP system defined with the input parameters: $\epsilon_{NP} = 60, \epsilon_S = 78.5, Q = -100, v = 1$ and $M = 1272$, without adjusting any MPI or OpenMP parameters.

The ML-enhanced framework with adaptive timestep also improved the overall stability of the MD simulation. Figure 7 shows the key output features associated with the simulation of the ion-NP system characterized with $\epsilon_{NP} = 2, v = 1$ and other input parameters being the same as in Fig. 6. Fluctuations in the output feature f_d , which measures the deviation of the on-the-fly optimized functional from the statically optimized functional, illustrate the stability of the simulation. Auto-tuning of parameters produced diminished fluctuations in f_d compared to the non-adaptive framework. Figure 7 (inset) shows the variation of virtual parameter (μ) with simulated physical time for the same ion-NP system. By

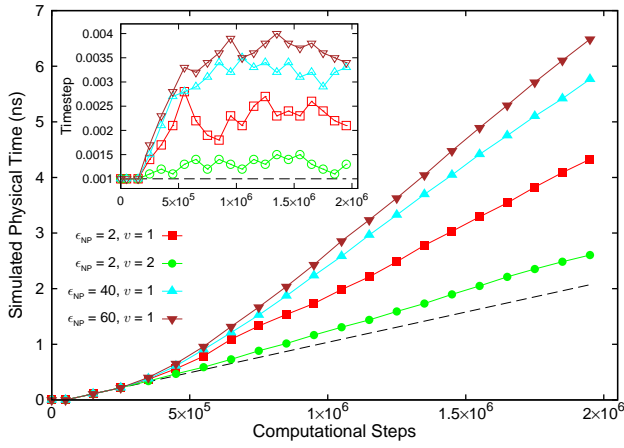


Figure 6. Simulated physical time t associated with the dynamics of ions as a function of computational steps S for a system of counterions near a charged, polarizable NP. The legend denotes the dielectric permittivity of the NP and ion valency ($\epsilon_{\text{NP}}; v$) associated with each system. Symbols (closed) are the results from using ML-enabled tuning of simulation timestep Δ and dashed line is the result for the non-adaptive case. For a given S , the adaptive framework yields a longer simulated physical time compared to the non-adaptive framework. (Inset) Simulation timestep (in units of ≈ 1.09 picoseconds) for the same systems; open symbols correspond to the system denoted with the corresponding closed symbols in the outset legend. ML-enabled adaptive framework produces comparatively higher timestep values; black dashed line denotes $\Delta = 0.001$ associated with the non-adaptive case. Δ values represent the average timestep over a period of $t_n \rightarrow t_{n+1}$.

Table 3. Performance comparison of different ion-NP systems simulated for 2 million time steps (≈ 20 hrs walltime) on 4 MPI nodes each with 16 OMP threads.

Physical System	Time (ns)	Speedup	Stability
Non-adaptive			
2, 78.5, -100, 1	2.18	1.00x	0.01552
ML-based Adaptive			
2, 78.5, -100, 1	4.24	1.95x	0.00048
2, 78.5, -100, 2	2.87	1.35x	0.00088
40, 78.5, -100, 1	6.08	2.79x	0.00076
60, 78.5, -100, 1	6.57	3.02x	0.00063

definition, the non-adaptive model produced constant μ . On the other hand, the adaptive model produced the auto-tuning of μ which was correlated with the more stable dynamics (red circles characterizing f_d in the outset of Fig. 7) generated by the ML-enhanced adaptive framework. Indeed, the variance in f_d data for non-adaptive model ($f_{d|\sigma^2} = 0.01552$) was much higher than that for adaptive model ($f_{d|\sigma^2} = 0.00048$). Table 3 shows the variance of f_d for the same ion-NP systems analyzed above; in all cases, variance was found to be significantly smaller (indicating higher stability) for the adaptive model compared to the non-adaptive case. This enhanced stability can be attributed to the optimal updates of parameters μ, T_v during the intermediate times of the simulation.

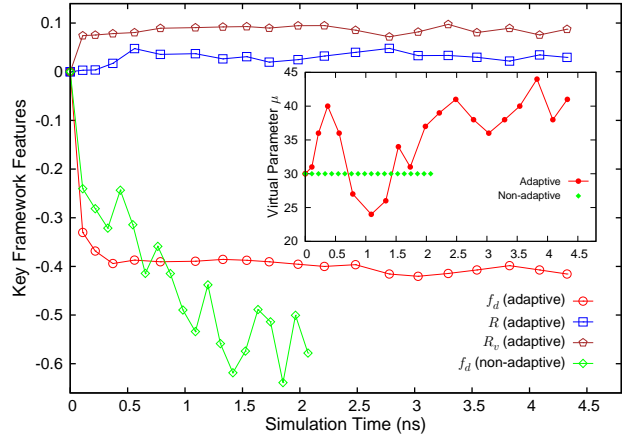


Figure 7. Key output features R, f_d, R_v as a function of simulated physical time t for the counterion-NP system characterized with NP charge $Q = -100$, ion valency $v = 1$, NP permittivity $\epsilon_{\text{NP}} = 2$, solvent permittivity $\epsilon_s = 78.5$, and mesh size $M = 1272$. ML-enabled auto-tuning of Δ and virtual parameters produces enhanced stability (diminished fluctuations in f_d) compared to the non-adaptive case (strong fluctuations in f_d). (Inset) ML-enabled auto-tuning of the virtual parameter μ for the same system (closed circles) and fixed μ for the non-adaptive case (closed diamonds).

In addition to increasing the efficiency and stability of the simulation, the framework with ML-enabled auto-tuning of parameters (adaptive framework) retained the accuracy associated with the framework with non-adaptive timestep and virtual parameters (non-adaptive framework). The accuracy can be assessed by comparing the density profiles of the ions computed using the two approaches. For different ϵ_{NP} values (other input parameters same as above), the peak densities computed using simulations based on adaptive framework were found to be in agreement with those calculated using the non-adaptive framework as shown in Figure 8; data from either approach falls on the dashed line which indicates linear correlation. Top-left inset of Figure 8 shows the variation of the counterion density (in σ^{-3} , where $\sigma = 0.357$ nm is the ion diameter) as a function of the distance from the NP (of radius 7.5σ) for the specific case of $\epsilon_{\text{NP}} = 2$. The density profiles extracted from the adaptive and non-adaptive frameworks were found to be in good agreement (relative error in either distributions was found to be $\approx 1\%$). Bottom-right inset of Fig. 8 shows the variation of the peak density of counterions as a function of the dielectric permittivity ϵ_{NP} of the NP. Both approaches yield similar peak densities. Lowering ϵ_{NP} leads to an increase in the repulsive force on the counterions due to the induced charges on the surface of the NP, leading to the reduction in the peak density of counterions near the NP surface.

5.3 Benchmarking ML-enhanced Simulations

The enhanced dynamical optimization framework was benchmarked using BigRed2 cluster nodes. These nodes have maximal achieved performance of 596.4 teraFLOPS, and feature a hybrid architecture based on two Cray, Inc., 344 XE6 (CPU-only) compute nodes, providing a total of 1,020 compute nodes, 21,824 processor cores, and 43,648 GB of RAM. Each XE6 node has two AMD Opteron 16-core Abu

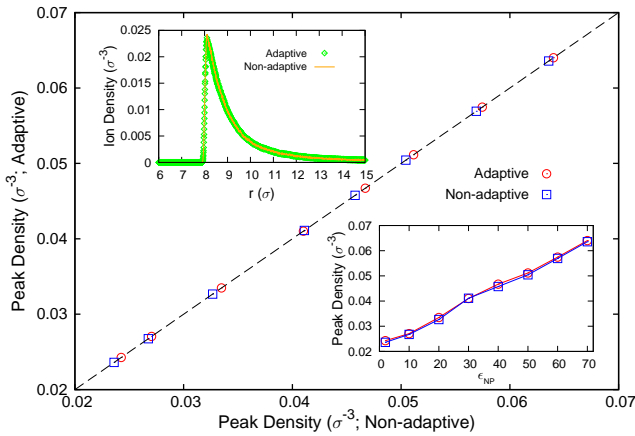


Figure 8. Correlation between the peak densities associated with the distribution of counterions for ion-NP systems characterized by different NP permittivity ϵ_{NP} ; other parameters are the same as listed in the caption of Figure 7. Blue squares are values from non-adaptive simulation, red circles are results from ML-enabled adaptive simulation. (Top-left inset) Density distribution of counterions for the system with $\epsilon_{NP} = 2$; symbols are the results from adaptive model, line corresponds to the non-adaptive case. (Bottom-right inset) Peak densities from the two models as a function of ϵ_{NP} .

Dhabi x86.64 CPUs and 64 GB of RAM; each XK7 node has one AMD Opteron 16-core Interlagos x86.64 CPU, and 32 GB of RAM.

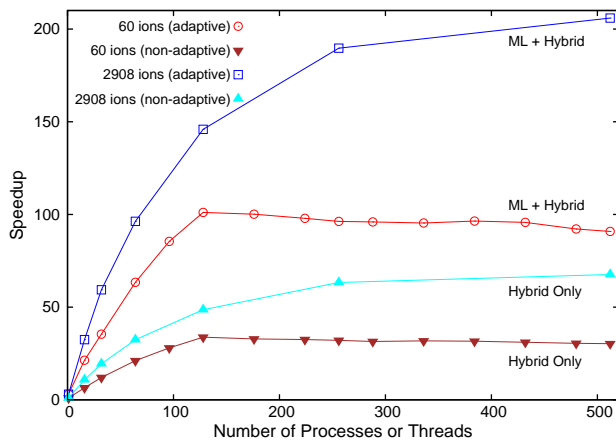


Figure 9. Strong scaling plot of the performance of MPI/OpenMP hybrid method with and without ML-enabled adaptive timestep enhancement. Data is shown for 60 and 2908 ions; in both systems, NP is meshed with 1082 mesh points (induced charges). For both systems, ML + hybrid method outperforms hybrid-only implementation.

Figure 9 compares the strong scaling plot of the performance of the dynamical optimization framework parallelized using the MPI/OpenMP hybrid model with and without adaptive timestep enhancement. The reported speedup is defined as the ratio of serial run-time to the time taken by the parallelized simulations (with and without ML-enabled auto-tuning of timestep) to simulate a set t_b nanoseconds of physical system dynamics. In Figure 9, results are shown for simulation of a system of 60 ions and 1082 mesh points as well as a larger system of 2908 ions and 1082 mesh points for $t_b \approx 2$ nanoseconds. The larger system

of 2908 ions is comprised of both positive and negative ions characterized by an electrolyte concentration $c \approx 0.2$ M (with 60 counterions, 1424 positive electrolyte ions, and 1424 negative electrolyte ions). While the ML models were not trained with c as an input parameter, simulations of the electrolyte systems using ML-predicted timestep and optimal virtual parameters associated with counterion-only systems were successful for over $c \approx 0.2$ M (see Section 5.4), enabling the benchmarking of simulations of larger systems. With non-adaptive timestep, the hybrid model produced the maximum speedup of 33.80 with 128 processes (8 MPI nodes and 16 OpenMP threads inside each MPI node) for the smaller system. For the same configuration, the hybrid model with ML-enabled auto-tuning of the simulation timestep was able to achieve a maximum speedup of 101.07 with 128 processes. Thus, the runtime for this system was reduced from 55 hours to 30 minutes (68 minutes without adaptive time-stepping). We note that the maximum speedup was calculated without considering the execution time reduction gained from the memory optimization techniques. For the hybrid model, we found that the optimal configuration of OpenMP threads is socket bound as noted in the literature (Rabenseifner et al. 2009). As a result, the number of optimal OpenMP threads in our experiment was 16 for any number of MPI processes.

When implemented to a larger system with a total number of 2908 ions and 1082 mesh points exhibiting induced charges, the combination of the hybrid methodology and the ML-based selection of adaptive timestep reduced the execution time of simulation for 2 nanoseconds from 88 days to 10.2 hours (32 hours without adaptive time-stepping) with a speedup of over 200. Clearly, the optimum number of MPI processes are proportional to the problem size when OpenMP thread affinity is set to the socket resulting in a well weak scaling system. The maximum speedup of 620.76 was obtained for 1024 processes (threads) executing a simulation of 5816 ions and 1272 mesh points for $t_b \approx 0.5$ nanoseconds.

5.4 Application: Concentrated Electrolytes near an Oil-Water Emulsion Droplet

The ML-enhanced framework was applied to compute the distribution of monovalent electrolyte ions outside a charged oil-in-water emulsion droplet (de Graaf et al. 2008; Bier et al. 2008) at room temperature $T = 298$ K. Positive and negative ions were considered to be of the same size to simplify the system and focus on analyzing the effects of polarization charges on the density distributions. Such model systems have been considered in previous numerical studies of electrolyte ions near polarizable nanospheres (Messina 2002; dos Santos et al. 2011; Shen et al. 2017). All ions were modeled as Lennard-Jones (LJ) spheres of diameter $\sigma = 0.357$ nm. The oil-water emulsion droplet was modeled as a spherical, dielectric interface with surface charge $Q = -60e$ and radius $a = 7.5\sigma \approx 2.7$ nm. The whole system of ions and the droplet was taken to be in a large spherical simulation cell of radius $b = 40\sigma \approx 14$ nm. The emulsion surface and the simulation cell boundary were modeled as spherical LJ walls. All excluded-volume interactions were modeled using the repulsive 6-12 LJ potential with $\epsilon_{LJ} = 1 k_B T$ and cutoff $r_c = 2^{1/6}\sigma$.

The dielectric permittivity of oil was taken to be $\epsilon_o = 2$, while water was associated with $\epsilon_w = 78.5$. The difference in the polarizable properties of oil and water lead to surface polarization charges. Electrostatic interactions arising from the bare and induced charge interactions in the system were modeled using the forces originating from the functional described in Eq. (4). The oil-water dielectric interface (which can be considered as the surface of the NP) was meshed with $M = 1082$ mesh points; higher M values were found to yield similar densities indicating that $M = 1082$ was large enough to obtain converged density profiles. In addition to the system with no added electrolyte, systems with $c \approx 0.02, 0.1, 0.2$ M were considered to analyze the effects of changing the electrolyte concentration c on the ionic distributions. Together with the 60 counterions (associated with the charged oil-water droplet), these concentrated electrolytes correspond to systems with a total of 350, 1514, and 2968 ions respectively. Simulation of the smallest system (60 ions) assuming non-polarizable NP surface was also performed for assessing the role of surface polarizability.

It should be noted that the ML procedure was only trained for smaller systems with counterions ($c = 0$, thus in the absence of co-ions). Further the training was performed for relatively smaller computational time (up to 2 million). With the application of the ML-enhanced framework to the aforementioned systems, we are elucidating the transferability of the features learned for the smaller system to different but related physical systems (with finite c , presence of co-ions, and long-time dynamics). Such an extended application of the developed ML models is possible, in part, because addition of electrolytes weaken the effective interaction between counterion and oil-water surface as a result of the screened electrostatic forces.

The aforementioned attributes of the physical system supply the input parameters for the enhanced dynamical optimization framework. Following the process elucidated in Fig. 4, these system inputs were first passed to the ML model I to predict the required virtual system parameters to kickstart the simulation. Then, ML model II was employed to auto-tune the timestep and virtual parameters during the simulation executed using the hybrid OpenMP/MPI method up to 2 million time steps. For the rest of the simulation run, ML model II was disabled and ML model I was used. The total number of steps S were selected based on the convergence of the density distributions; S was system-dependent and converged results were obtained after $\approx 5 - 20$ nanoseconds (14 - 60 hours) depending on the electrolyte concentration.

In all simulations, regardless of system sizes and presence/absence of electrolytes, good energy conservation was observed with $R < 0.05$. Similarly, the induced charges on the oil-water dielectric interface were accurately tracked by the on-the-fly optimization framework ($f_d < 1\%$; inset in Figure 10 shows the accurate tracking on the functional for the $c = 0.02$ M case). These two key features demonstrated the success of the ML-based virtual parameter selection process. As noted before, the bound on R_v is system-size and system-feature-dependent that is informed by prior domain experience; for counterion-only systems, $R_v < 0.15$ as per the expected limit while for electrolyte systems the bound on R_v was higher and increased with increasing c . These

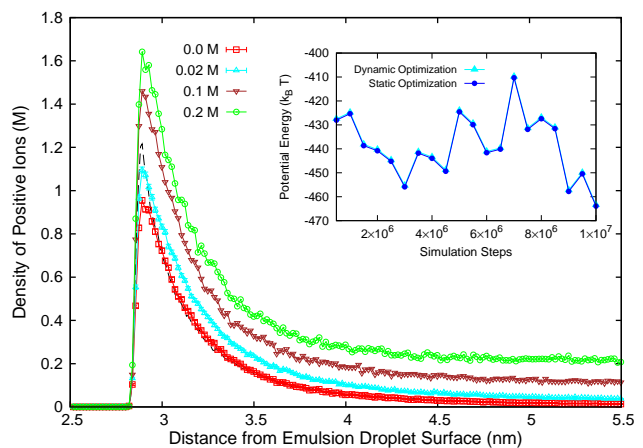


Figure 10. Ionic density profiles extracted from ML-enhanced MD simulations based on the dynamical optimization framework. Outset shows the density of positive ions for electrolytes of concentration $c \approx 0.0, 0.02, 0.1, 0.2$ M near negatively-charged oil-water emulsion with dielectric permittivity of oil and water as 2 and 78.5 respectively. Black dashed line refers to the result for the emulsion assumed to be unpolarizable (with oil permittivity as 78.5). (Inset) Comparison of the functional optimized dynamically (diamonds) and the functional optimized at regular intervals keeping the ionic configuration static during the optimization process (circles) for $c \approx 0.02$ M system.

findings demonstrated that ML models could be trained on smaller systems and applied to larger systems to obtain efficient and stable dynamics of ions in the latter case.

Figure 10 shows the density profiles of the positive ions associated with the four systems. For all concentrations, the densities reach a constant value in the bulk away from the polarizable oil-water surface (negative ion densities, not shown, also reach a constant value in the bulk). Positive ions are found to accumulate near the dielectric interface, with the peak density increasing with c . Similarly, negative ions are depleted near the interface due to the repulsion from the bare charge on the oil-water surface as well as the induced charge. Comparison of the no electrolyte result with the case where the surface is considered to be unpolarizable (with a permittivity equal to that of water) is also shown. The polarization charges on the surface lead to depletion of ions from the interface. Increasing concentration leads to the rise in the peak density; the overall behavior is determined by the competition between the ion-induced charge repulsion near the surface and the ion-ion electrostatic and steric correlations. The position of the peak density remains relatively unaltered regardless of c value, as observed previously for monovalent electrolytes Messina (2002).

The inset in Fig. 10 shows the comparison of the potential energy functional optimized dynamically with the energy obtained after optimizing at regular intervals keeping the ionic configuration static during the optimization process for the $c \approx 0.02$ M system. The potential energy and the associated induced charges it characterizes were accurately tracked at all times for up to 10 million time steps (≈ 20 ns). The stability and accuracy evident from this plot demonstrates the success of the ML-based parameter selection process; other systems showed similar agreement

between dynamically optimized and statically optimized energy functionals.

6 Conclusion and Outlook

We illustrated the computational gains accessible by integrating machine learning (ML) methods for parameter auto-tuning in MD simulations by demonstrating the enhancement of stability and efficiency of MD simulations of ions near polarizable NPs based on the dynamical optimization framework. The Neural Network (NN) based ML model yielded the highest success rate among the non-linear regression models employed to predict the virtual system parameters at the start of the simulation. When integrated with the MD simulation, the NN model predicted the timestep and the associated optimal virtual parameters on-the-fly with 94.3% success rate. The auto-tuning of the simulation timestep resulting from this ML-enhanced, adaptive simulation framework enabled the simulation of ions for a longer physical time (over 30 ns) with a net speedup of $\approx 1.5 - 3$ (depending on the system configuration) compared to the non-adaptive sequential simulation model. The combination of the ML procedure with the hybrid parallelization method generated stable dynamics of thousands of ions in the presence of polarized NPs with computational time reducing from thousands of hours to tens of hours yielding a maximum speed up of ≈ 600 . Compared to the non-adaptive simulation with static initial virtual parameters, the stability of the adaptive framework increased by an order of magnitude.

This enhanced simulation framework has many applications and we demonstrated its utility by generating stable, accurate dynamics of ions in the presence of a polarized nanoemulsion droplet for ≈ 10 million simulation steps (physical time over 20 ns). Additionally, we showed the broad applicability of the approach by demonstrating that the ML models could be trained on a smaller system and applied to larger systems characterized by new attributes (not used for training) with success in producing accurate and stable dynamics. At the same time, the approach reveals a limit on the concentration and physical time that one can simulate based on training a counterion-only system. Future efforts will involve exploring the training of ML models with electrolyte concentration and attributes as input parameters. We will also explore integrating the current ML-enabled enhanced framework with fast Ewald solvers to support periodic boundary conditions and reduce the $O(n^2)$ scaling in particle system size to $O(n \log n)$ (Marchi et al. 2001; Barros et al. 2014).

The use of ML to enhance the simulation framework enables users across the globe, with a diversity of in-domain experience, to simulate ions near polarizable NPs via the use of web-based applications hosted on services like nanoHUB. A tool powered by this enhanced framework was recently published on nanoHUB (Kadupitiya et al. 2018). We will extend our framework to enable the process of using the data generated by this nanoHUB application for continuous training of the ML-based parameter prediction procedure. The demonstration of integrating ML-based parameter prediction methods with MD-based simulations can be extended to other energy minimization problems. The

implications of ML-enabled tuning of simulation timestep also suggest new avenues of exploring ML to advance simulation methods, such as ML-informed dynamic grid sizes in mesh-based problems.

Acknowledgements

This work is supported by the National Science Foundation under the Network for Computational Nanotechnology (NCN) program through award 1720625 (Network for Computational Nanotechnology - Engineered nanoBIO Node). We thank XSEDE Startup Allocation through Grant TG-DMR170089, and BigRed2 supercomputer at Indiana University for computing resources.

References

- Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M et al. (2016) Tensorflow: a system for large-scale machine learning. In: *OSDI*, volume 16. pp. 265–283.
- Abrua HD, Kiya Y and Henderson JC (2008) Batteries and electrochemical capacitors. *Physics Today* 61(12): 43.
- Allen R, Hansen JP and Melchionna S (2001) Electrostatic potential inside ionic solutions confined by dielectrics: a variational approach. *Phys. Chem. Chem. Phys.* 3: 4177–4186.
- Attard P (2003) Variational formulation for the electrostatic potential in dielectric continua. *The Journal of Chemical Physics* 119(3): 1365–1372.
- Balachandran PV, Xue D, Theiler J, Hogden J and Lookman T (2016) Adaptive strategies for materials design using uncertainties. *Scientific reports* 6: 19660.
- Balakrishnan D and Puthusserypady S (2005) Multilayer perceptrons for the classification of brain computer interface data. In: *Bioengineering Conference, 2005. Proceedings of the IEEE 31st Annual Northeast*. IEEE, pp. 118–119.
- Balaprakash P, Dongarra J, Gamblin T, Hall M, Hollingsworth JK, Norris B and Vuduc R (2018) Autotuning in high-performance computing applications. *Proceedings of the IEEE* 106(99): 1–16.
- Barros K and Luijten E (2014) Dielectric effects in the self-assembly of binary colloidal aggregates. *Phys. Rev. Lett.* 113: 017801.
- Barros K, Sinkovits D and Luijten E (2014) Efficient and accurate simulation of dynamic dielectric objects. *The Journal of Chemical Physics* 140(6): 064903. DOI:10.1063/1.4863451.
- Bartók AP, De S, Poelking C, Bernstein N, Kermode JR, Csányi G and Ceriotti M (2017) Machine learning unifies the modeling of materials and molecules. *Science Advances* 3(12). DOI: 10.1126/sciadv.1701816.
- Beckstein O, Tai K and Sansom MS (2004) Not ions alone: barriers to ion permeation in nanopores and channels. *Journal of the American Chemical Society* 126(45): 14694–14695.
- Behler J (2016) Perspective: Machine learning potentials for atomistic simulations. *The Journal of chemical physics* 145(17): 170901.
- Bier M, Zwanikken J and van Roij R (2008) Liquid-liquid interfacial tension of electrolyte solutions. *Phys. Rev. Lett.* 101: 046104.
- Blöchl PE and Parrinello M (1992) Adiabaticity in first-principles molecular dynamics. *Phys. Rev. B* 45: 9413–9416.

- Boda D, Gillespie D, Nonner W, Henderson D and Eisenberg B (2004) Computing induced charges in inhomogeneous dielectric media: Application in a monte carlo simulation of complex ionic systems. *Phys. Rev. E* 69(4): 046702.
- Botu V and Ramprasad R (2015) Adaptive machine learning framework to accelerate ab initio molecular dynamics. *International Journal of Quantum Chemistry* 115(16): 1074–1083.
- Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, Niculae V, Prettenhofer P, Gramfort A, Grobler J et al. (2013) Api design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*.
- Car R and Parrinello M (1985) Unified approach for molecular dynamics and density-functional theory. *Phys. Rev. Lett.* 55(22): 2471–2474.
- Chen K and Yu J (2014) Short-term wind speed prediction using an unscented kalman filter based state-space support vector regression approach. *Applied Energy* 113: 690–705.
- Cherkassky V and Ma Y (2004) Practical selection of svm parameters and noise estimation for svm regression. *Neural networks* 17(1): 113–126.
- Ch’ng K, Carrasquilla J, Melko RG and Khatami E (2017) Machine learning phases of strongly correlated fermions. *Phys. Rev. X* 7: 031038. DOI:10.1103/PhysRevX.7.031038.
- Chollet F et al. (2015) Keras.
- Clapham DE (2007) Calcium signaling. *Cell* 131(6): 1047 – 1058.
- de Graaf J, Zwanikken J, Bier M, Baarsma A, Oloumi Y, Spelt M and van Roij R (2008) Spontaneous charging and crystallization of water droplets in oil. *The Journal of Chemical Physics* 129(19): 194701.
- Denil M, Shakibi B, Dinh L, De Freitas N et al. (2013) Predicting parameters in deep learning. In: *Advances in neural information processing systems*. pp. 2148–2156.
- Deserno M and Holm C (1998) How to mesh up ewald sums. i. a theoretical and numerical comparison of various particle mesh routines. *The Journal of chemical physics* 109(18): 7678–7693.
- dos Santos AP, Bakhshandeh A and Levin Y (2011) Effects of the dielectric discontinuity on the counterion distribution in a colloidal suspension. *The Journal of Chemical Physics* 135(4): 044124.
- dos Santos AP and Netz RR (2018) Dielectric boundary effects on the interaction between planar charged surfaces with counterions only. *The Journal of chemical physics* 148(16): 164103.
- Eng CL, Tong JC and Tan TW (2014) Predicting host tropism of influenza a virus proteins using random forest. *BMC medical genomics* 7(3): S1.
- Fahrenberger F, Xu Z and Holm C (2014) Simulation of electric double layers around charged colloids in aqueous solution of variable permittivity. *The Journal of Chemical Physics* 141(6): 064902. DOI:10.1063/1.4892413.
- Felderhof BU (1977) Fluctuations of polarization and magnetization in dielectric and magnetic media. *The Journal of Chemical Physics* 67(2): 493–500.
- Ferguson AL (2017) Machine learning and data science in soft materials engineering. *Journal of Physics: Condensed Matter* 30(4): 043002.
- Fois ES, Penman JI and Madden PA (1993) Control of the adiabatic electronic state in ab initio molecular dynamics. *The Journal of Chemical Physics* 98(8): 6361–6368.
- Gan Z, Wu H, Barros K, Xu Z and Luijten E (2015) Comparison of efficient techniques for the simulation of dielectric objects in electrolytes. *Journal of Computational Physics* 291: 317 – 333. DOI:https://doi.org/10.1016/j.jcp.2015.03.019.
- Gan Z and Xu Z (2011) Multiple-image treatment of induced charges in monte carlo simulations of electrolytes near a spherical dielectric interface. *Phys. Rev. E* 84: 016705.
- Hatlo MM and Lue L (2008) The role of image charges in the interactions between colloidal particles. *Soft Matter* 4: 1582–1596.
- Honig B and Nicholls A (1995) Classical electrostatics in biology and chemistry. *Science* 268(5214): 1144–1149.
- Jackson JD (1999) *Classical Electrodynamics*. 3 edition. Wiley, New York.
- Jadhao V, Solis FJ and Olvera de la Cruz M (2012) Simulation of charged systems in heterogeneous dielectric media via a true energy functional. *Phys. Rev. Lett.* 109: 223905.
- Jadhao V, Solis FJ and Olvera de la Cruz M (2013) A variational formulation of electrostatics in a medium with spatially varying dielectric permittivity. *The Journal of Chemical Physics* 138(5): 054119.
- Jing Y, Jadhao V, Zwanikken JW and de la Cruz MO (2015) Ionic structure in liquids confined by dielectric interfaces. *The Journal of Chemical Physics* 143(19): 194508. DOI:10.1063/1.4935704.
- Kadupitiya J, Brunk N, Ali S, Fox GC and Jadhao V (2018) Nanosphere electrostatics lab. DOI:doi:10.4231/D3WP9T88W. Online on nanoHUB.
- Kasson PM and Jha S (2018) Adaptive ensemble simulations of biomolecules. *Current opinion in structural biology* 52: 87–94.
- Kazemi V and Sullivan J (2014) One millisecond face alignment with an ensemble of regression trees. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1867–1874.
- Klimeck G, McLennan M, Brophy SP, III GBA and Lundstrom MS (2008) nanohub.org: Advancing education and research in nanotechnology. *Computing in Science Engineering* 10(5): 17–23. DOI:10.1109/MCSE.2008.120.
- Levin Y (2005) Strange electrostatics in physics, chemistry, and biology. *Physica A: Statistical Mechanics and its Applications* 352(1): 43 – 52.
- Lipparini F, Scalmani G, Mennucci B, Cancès E, Caricato M and Frisch MJ (2010) A variational formulation of the polarizable continuum model. *The Journal of Chemical Physics* 133(1): 014106.
- Liu J, Qi Y, Meng ZY and Fu L (2017) Self-learning monte carlo method. *Phys. Rev. B* 95: 041101.
- Long AW, Zhang J, Granick S and Ferguson AL (2015) Machine learning assembly landscapes from particle tracking data. *Soft Matter* 11(41): 8141–8153.
- Luehr N, Markland TE and Martnez TJ (2014) Multiple time step integrators in ab initio molecular dynamics. *The Journal of Chemical Physics* 140(8): 084116. DOI:10.1063/1.4866176.
- Marchi M, Borgis D, Levy N and Ballone P (2001) A dielectric continuum molecular dynamics method. *The Journal of Chemical Physics* 114(10): 4377–4385.

- Marcus RA (1956) On the theory of oxidation-reduction reactions involving electron transfer. i. *The Journal of Chemical Physics* 24(5): 966–978.
- Messina R (2002) Image charges in spherical geometry: Application to colloidal systems. *The Journal of Chemical Physics* 117(24): 11062–11074.
- Morningstar A and Melko RG (2017) Deep learning the ising model near criticality. *arXiv preprint arXiv:1708.04622* .
- Nakano A, Vashishta P and Kalia RK (1994) Massively parallel algorithms for computational nanoelectronics based on quantum molecular dynamics. *Computer Physics Communications* 83(2): 181 – 196.
- Quan H, Srinivasan D and Khosravi A (2014) Short-term load and wind power forecasting using neural network-based prediction intervals. *IEEE transactions on neural networks and learning systems* 25(2): 303–315.
- Rabenseifner R, Hager G and Jost G (2009) Hybrid mpi/openmp parallel programming on clusters of multi-core smp nodes. In: *Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on*. IEEE, pp. 427–436.
- Raspaud E, Olvera de la Cruz M, Sikorav J and Livolant F (1998) Precipitation of dna by polyamines: a polyelectrolyte behavior. *Biophys J* 74(1): 381–93.
- Reiner ES and Radke CJ (1990) Variational approach to the electrostatic free energy in charged colloidal suspensions: general theory for open systems. *J. Chem. Soc., Faraday Trans.* 86: 3901–3912.
- Rottler J and Maggs AC (2004) Local molecular dynamics with coulombic interactions. *Phys. Rev. Lett.* 93(17): 170201.
- Sacanna S, Kegel WK and Philipse AP (2007) Thermodynamically stable pickering emulsions. *Phys. Rev. Lett.* 98: 158301.
- Sagui C and Darden T (2001) Multigrid methods for classical molecular dynamics simulations of biomolecules. *The Journal of Chemical Physics* 114(15): 6578–6591.
- Schoenholz SS (2018) Combining machine learning and physics to understand glassy systems. *Journal of Physics: Conference Series* 1036(1): 012021.
- Shen M, Li H and De La Cruz MO (2017) Surface polarization effects on ion-containing emulsions. *Physical review letters* 119(13): 138002.
- Spellings M and Glotzer SC (2018) Machine learning for crystal identification and discovery. *AIChE Journal* 64(6): 2198–2206.
- Sprink M (1991) Computer simulation of the dynamics of induced polarization fluctuations in water. *The Journal of Physical Chemistry* 95(6): 2283–2291.
- Tuckerman M, Berne BJ and Martyna GJ (1992) Reversible multiple time scale molecular dynamics. *The Journal of chemical physics* 97(3): 1990–2001.
- Tyagi S, Suzen M, Segal M, Barbosa M, Kantorovich SS and Holm C (2010) An iterative, fast, linear-scaling method for computing induced charges on arbitrary dielectric boundaries. *The Journal of Chemical Physics* 132(15): 154112.
- Villasenor J and Buneman O (1992) Rigorous charge conservation for local electromagnetic field solvers. *Computer Physics Communications* 69(2): 306 – 316.
- Whaley RC and Dongarra JJ (1998) Automatically tuned linear algebra software. In: *Supercomputing, 1998. SC98. IEEE/ACM Conference on*. IEEE, pp. 38–38.
- Wynveen A and Bresme F (2006) Interactions of polarizable media in water: A molecular dynamics approach. *The Journal of Chemical Physics* 124(10): 104502. DOI:10.1063/1.2177244.
- Xu Z (2013) Electrostatic interaction in the presence of dielectric interfaces and polarization-induced like-charge attraction. *Phys. Rev. E* 87: 013307. DOI:10.1103/PhysRevE.87.013307.
- Yadav AK, Malik H and Chandel S (2016) Selection of most relevant input parameters using weka for artificial neural network based solar radiation prediction models. *Renewable and Sustainable Energy Reviews* 31: 509–519.
- Yigitbasi N, Willke TL, Liao G and Epema D (2013) Towards machine learning-based auto-tuning of mapreduce. In: *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on*. IEEE, pp. 11–20.
- York DM and Karplus M (1999) A smooth solvation potential based on the conductor-like screening model. *The Journal of Physical Chemistry A* 103(50): 11060–11079.