CAVIAR: A simulation package for charged particles in environments surrounded by conductive boundaries

Cite as: AIP Advances 10, 035310 (2020); https://doi.org/10.1063/1.5140052
Submitted: 27 November 2019. Accepted: 21 February 2020. Published Online: 06 March 2020

Morad Biagooi, Mohammad Samanipour, S. Alireza Ghasemi, and SeyedEhsan Nedaee Oskooee

AVS Quantum Science
CAVIAR: A simulation package for charged particles in environments surrounded by conductive boundaries

Morad Biagooi, Mohammad Samanipour, S. Alireza Ghasemi, and SeyedEhsan Nedaee Oskoee

AFFILIATIONS
1 Department of Physics, Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan 45137-66731, Iran
2 Research Center for Basic Sciences and Modern Technologies (RBST), Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan 45137-66731, Iran

ABSTRACT
We introduce CAVIAR, new scientific software that is developed for molecular simulation of ionic-liquids or charged colloids inside conductive boundaries. CAVIAR imports computer-aided-design geometries and uses them to simulate boundary walls. Then based on this geometry, a finite element mesh is generated and utilized for solving the Poisson equation. To avoid the complexity due to the singularity of point charges, we propose a new method, using the advantages of the superposition theorem of the linear partial differential equations. Within this paper, the CAVIAR structure, its features, and basic scientific algorithms are discussed.

© 2020 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/). https://doi.org/10.1063/1.5140052

I. INTRODUCTION
Soft-matter physics is of high technological importance. This area of physics mainly contains the study of liquids, colloids, polymers, and biological and granular materials. Systems with charged components are an important subset of soft-matter, which present uncommon behaviors. Charged particles, both in the dielectric as well as the conductive environments, are of scientific interest, due to their important rules in the physical behavior of such systems. In particular, charged particles surrounded by conductive boundaries that are observed in many physical systems are both of technological and theoretical interest. As a special field of interest, one can refer to conductive porous electrodes in porous super-capacitors, which have gained much technological attention as a candidate for future energy storage units. Furthermore, capacitive deionization by porous electrodes as a method of water desalination is another example of the above-mentioned systems. Any progress in these two fields of research can have high impact on human’s quality of life. Due to the complexity of the charged systems, simulation is an effective and efficient way to investigate these systems; it enables us to figure out the details of physical events in different scales of time and space up to the available computational resources.

The Molecular Dynamics (MD) simulation is a well-known and popular method for investigating many-body systems. However, for systems with long-range interactions such as electrostatic and gravitational interactions, the complexity of the standard MD algorithms is on the order of $O(N^2)$, which is computationally inefficient. Furthermore, in charged systems under partial or full periodic boundary conditions, infinite summations appear that may not converge easily. To deal with this problem, Ewald sum based algorithms have been created. In these algorithms, the potential is divided into two short-range and long-range parts in a way that the long-ranged part always converges. However, the situation can even be worse when charged particles are surrounded by conductive boundaries; here, the complexity arises from the existence of induced charges on the boundaries. In order to address this problem, some methods and algorithms such as Induced Charge MMM2D (ICMM2D) for 2D, ELCIC for 2D + h, and induced charge computation (ICC) for 3D periodicity have been developed recently. The ICMM2D (Induced...
Charge MMM2D) method is an extension of MMM2D, which is able to treat planar dielectric interfaces using image charge particles (ICPs). The ICC method, which was first proposed by Tyagi et al., is a generic method that can be used for dielectric interfaces of arbitrary shapes and is based on the induced charge computation (ICC) method. The ICC method needs a Poisson solver suitable for the same geometry in the absence of the dielectric interface, and the star denotes a place holder for any Poisson solver. For example, the ICC method is used for simulating carbon electrodes with the shape of slit-pore containing ionic-liquids. There is another method introduced by Reed et al., which is based on the work of Siepmann and Sprik, which uses a variational procedure for variable charges to set the constant potential condition. This method has also been used in some other super-capacitor simulations.

In this paper, we introduce a new method for simulating charged particles in conductive media. A simple version of this algorithm was introduced and used for simulation of a partially periodic system confined between two metallic plates. Based on this algorithm, we develop CAVIAR, which is a new, multi-purpose, and cross-platform scientific software package. The name CAVIAR is from “Charged particles in environments surrounded by conductive boundaries.” CAVIAR was originally developed as a molecular dynamics simulation code for charged particles near conductive boundaries with non-trivial geometries. It can also be used to simulate problems with less complexity such as MD systems without conductive or periodic boundaries. CAVIAR has been developed in C++ with an extensible design. The rest of the paper is as follows: In Sec. II, we first present all the details about induced charge calculations and talk about the interactions of particles with geometries. In Sec. III, we introduce the package structure, interpreter, implementation of codes, parallel simulation methods, external libraries, and other open-source software that should be used as an additional tool. In Sec. IV, we discuss the practical uses of the CAVIAR with a simulation example. Section V has the results of some test simulations that compares the performance of the implemented algorithm with another software. At last, a conclusion is provided in Sec. VI.

II. METHODOLOGY

A. Poisson to Laplace transformation (PLT) algorithm

When dealing with conductive boundaries, induced charges appear on the boundaries, which in turn make the computation of the electrical potential generally a difficult task. It is well known that the relation between charge distribution \( \rho(\vec{x}) \) and electric potential \( \varphi(\vec{x}) \) obeys Poisson’s equation \( \nabla^2 \varphi = -\rho/\epsilon_0 \). Even for regular geometries such as a point charge in a cubic container with conductive walls, the analytical methods of solving the Poisson equation, like image charge methods, result in an infinite series with dramatically slow convergence. Besides, typical methods of numerical solution of the Poisson equation fail in the case of a system of point charges because of the singularity of the electrical potential at the position of point charges. In order to avoid these singularities, we use the superposition theorem and divide the electric potential into two, singular (\( \varphi_{si} \)) and smooth (\( \varphi_{sm} \)) parts, \( \varphi(\vec{x}) = \varphi_{si}(\vec{x}) + \varphi_{sm}(\vec{x}) \), where the smooth part is the solution of the Laplace equation under modified boundary conditions,

\[
\nabla^2 \varphi_{sm}(\vec{x}) = 0, \tag{1}
\]

where \( \varphi_{si} \) in each point is simply

\[
\varphi_{si}(\vec{x}) = \sum_{i=1}^{N} \frac{q_i}{4\pi\epsilon_0 |\vec{x} - \vec{x}_i|}, \tag{2}
\]

The modified boundary condition for the \( \varphi_{sm} \) is given by

\[
\varphi_{sm}|_{\vec{x}_a} = \varphi|_{\vec{x}_a} - \varphi_{si}|_{\vec{x}_a}, \tag{3}
\]

where \( \varphi|_{\vec{x}_a} \) is the original electric potential on the boundaries, as schematically illustrated in Fig. 1.

To check the accuracy of the algorithm and our package, we compare the results of the simulation of a point charge \( q \) in a conductive spherical shell of radius \( a \) at a distance of \( r \) from its center. The analytical solution to this problem is

\[
\varphi(\vec{x}) = \frac{1}{4\pi\epsilon_0} \left( \frac{q}{|\vec{x} - \vec{x}_a|} + \frac{q'}{|\vec{x} - \vec{x}_a'|} \right), \quad q' = -\frac{rq}{a}, \quad \vec{x}_a' = \frac{a^2}{r} \frac{\vec{x}_a}{|\vec{x}_a|}. \tag{4}
\]

This case is simulated in CAVIAR-1.0 for a range of point charge distance from the center of the sphere, and the numeric result is compared with its analytical solution. Figure 2 is a plot of the relative error of computed electrical potential on the point charge as well as total induced charge on the sphere as a function of relative distance of the charge from the center of the sphere, for three different levels of mesh refinement. By approaching the conductive boundaries, the gradient of the electrical potential increases and as a result, the relative error of the computed electrical potential also increases. As shown in Fig. 2(f), the computational error is reduced with the mesh refinement.

Figure 3 is the flowchart of the CAVIAR package. It contains two major parts: the main loop of the Molecular Dynamics (MD) algorithm and the force field computation of long-range electrostatic forces using the Poisson to Laplace transformation (PLT) algorithm. The MD loop consists of computing the force fields of all short-range interactions such as particle–particle as well as particle–wall interactions, the PLT procedure for evaluating the electrostatic long-range interactions, and the integration of the resulting equation of motions. The PLT procedure, flowcharts
FIG. 2. The hexahedral spherical meshes and the related run results for the PLT algorithm. (a) Non-refined mesh of a hexahedral sphere. (b)–(e) Four consecutive global refinement of the mesh of (a). R. means refinement in the captions. In each global refinement, every 3D cell is divided into eight sub-cells. (a) and (b) are plotted here for reference. (c)–(e) are used for the error calculations. These plots are a cross section of their 3D mesh. (f) Plot of the relative error of computed potential at the position of the point charge in terms of the point charge distance from the center of sphere. The error was calculated by comparing the computed results with the analytical solution. The inset plot of (f) is the total induced electrical charge on the shell. As it can be expected, with a higher level of refinement, a more accurate answer can be achieved.

of which are sketched on the right side of Fig. 3, has the following steps:

1. **PLT initialization**: This step, which is done just once, includes generating finite element mesh, refining it, and importing it into the code, then assembling it with the weak (integral) form of the Laplace equation.

2. **Apply boundary values**: Calculate and apply boundary values \( \varphi_{sm} \) as constraints on the system.

3. **Solve Finite Element (FE) equations**: In this step, the resulting discrete Laplace equation is solved, then based on it, the electrical field resulted from the smooth part of the Laplace equation is computed.

4. **Compute singular force**: In this step, the singular electrostatic force is computed by summing over all Coulomb interactions between electrical charges.

5. **Compute total electrostatic force**: By summing the smooth and singular forces applied to each electrical charge, the total electrostatic force is computed in this step.

Taking periodic systems into account, one can utilize the Ewald based methods\textsuperscript{13} to compute the singular part of the potential [Eq. (2)] (see the work of Lee and Cai\textsuperscript{37} for more information).

The value of induced charge, \( q_i \), on each 2D mesh element of the electrodes can be calculated by means of the Gauss law,\textsuperscript{36} i.e., \( \oint_S \vec{E} \cdot \vec{n} da = q_i / \epsilon \), for a closed surface \( S \) that includes the mesh...
element. Here, \( \hat{n} \) is the normal vector of the mesh element, \( \vec{E} \) is the electric field, and \( da \) is its area. The area of each mesh element and its normal is calculated once at the initialization step of the simulation. The electric field is computed by numerical differentiation of the total electric potential computed at each PLT step.

Order of the PLT algorithm depends on the number of 2D boundary cells, \( N_B \), the number of charged particles, \( N \), and the order of the finite element solver of the Laplace equation. As discussed before, PLT is divided into different steps with different orders. Setting up boundary values requires the calculation of electric potential on all of the 2D boundary cells, which is in the order of \( O(N_B N) \). Solving the Laplace equation using the finite element methods depends on the number of mesh points (\( N_m \)), which is a function of the geometry. Furthermore, it depends on the solver employed by FE; the best solvers usually are in the order of \( O(N_m \log N_m) \). Calculating a portion of electrical forces on particles with the smooth potential is done by interpolation in cells that surround them and have the order of \( O(N) \). Calculating those with singular potential is done by the Ewald method, \( O(N \log N) \), as well.

**B. Particle-boundary interaction algorithm**

A solid boundary or wall in an MD simulation can be created using fixed particles. This is the most common way in granular or atomic simulations. However, this is a time-consuming method since it imposes extra calculation needed to compute forces of these numbers of fixed atoms. Furthermore, for simulating meso-scale and macro-scale systems such as charged colloidal particles and granular materials, the surface of boundaries will be rough if it is created by particles, against those in the real world. As an alternative, geometric shapes can be used to construct the boundary walls. In this case, the nearest distance \( D \) from a Cartesian point \( P \) to a geometric object can be calculated efficiently. \( D \) has a simple form, for convenience, geometry shapes such as a sphere, cylinder, or an unlimited plane. For more complicated cases, however, it is possible to approximate the boundary’s geometry with many connected flat polygons, i.e., a polyhedron. In this case, \( D \) is the minimum of distances between particles and each polygon. To compute the collision force efficiently, as schematically illustrated in Fig. 4(a), we first compute the compression vector \( \vec{\xi} \) from all boundary polygons using the following procedure:

1. **Surface**: First, we calculate the orthogonal distance \( D \) of a particle from the surface of the polygon. If the projected point is inside the polygon and \( D \) is less than the size of particle \( R \), the compression vector is equal to \( \vec{\xi} = (D - R)\hat{n} \).
2. **Edge**: If the above step does not happen, we check the orthogonal distance \( D \) of the particle from the polygon edges. If this
is less than \( R \) and the projected point is on the edge, we set the compression vector to \( \hat{\xi} = (D - R)\hat{e} \).

3. **Vertex**: Finally, if the previous step does not happen, we calculate the distance from all of the polygon vertices. The nearest vertex, which has a distance \( D \) less than \( R \), is used to return \( \hat{\xi} = (D - R)\hat{e} \). In addition, if none of the above conditions are satisfied, there is not any collision between the particle and polygon.

As schematically illustrated in Fig. 4(b), this method works properly even when there is more than one single polygon for collision at the corner of the boundaries. The above procedure is an optimized version of the Gilbert–Johnson–Keerthi algorithm\(^{40,41} \) for contact detection in MD. This type of contact algorithm for nonspherical elements is categorized in the branch of discrete-element-method algorithms, which is used alongside MD or FE simulations in physics and engineering.\(^{42-44} \)

This method is applied to many areas of charged-particle simulation, from the atomic scale to the macro-scale simulations, as well as for a range of force fields. As an example, it can be applied to the Lennard-Jones (LJ) as an atomic level force field,\(^{39} \) or the Hookean or Hertzian interaction in the simulation of the granular materials.\(^{38} \) In the case of Hookean or Hertzian forces fields, the compression \( \xi \) of the particle into the wall is needed to be calculated. For the LJ force field, \( \xi \) is the overlapping distance between the wall and particle potentials. The compression vector is in the direction of the imposed force from the wall to the particle. This procedure is simple and efficient; if the force field parameters and time step are set correctly, it will not let the particles pass through the boundaries.

The order of this algorithm depends on the multiplication of the number of particles \( N \) and the number of triangles \( N_t \) resulting in a performance downgrade specially in the case of complicated geometries. In order to increase the performance, we define a boundary cell-list. To do that, we first decompose the simulation box into cubic cells and then a list of all boundary triangles, which intersect with each cell, to the list of that cell. Furthermore, in order to take all possible cutoffs into account, we add the index of those triangles in a distance of less than or equal to \( R \) from cell boundaries to the list of that cell. Now for calculating the particle-boundary interactions, the algorithm primarily finds the cell containing the particle and then computes the interaction of the particle with all triangles stored in the cell-list. This algorithm is also implemented in CAVIAR.

To set the initial position of particles and atoms, the famous ray-casting algorithm\(^{45,46} \) is used to determine whether a point is inside or outside the simulation volume. In the mathematical view, a polyhedron separates the space into two volumes, meaning inside and outside.\(^{17} \) In the ray-casting algorithm, one needs to emit a straight ray from the point to infinity and sums the number of surface contacts. An odd number means that the point belongs to the inside of volume, and even numbers indicate that the point belongs to the outside one. Because particles should not have any contact with boundaries at the initial stage of the simulation, we mixed this algorithm with the above-mentioned procedure when calculating the initial positions.

### III. PACKAGE STRUCTURE

#### A. Implementation

CAVIAR is a software with its own scripting interpreter. It has been developed with object-oriented programming paradigm\(^ {18} \) and modern C++11 standard.\(^ {19} \) The package uses CMake\(^ {30} \) for its cross-platform configuration tool. The only external dependency is deal. II library.\(^ {31,32} \) The core of the package is a combination of two major components: interpreter and objects. All the physics and mathematics tools of CAVIAR are called “objects.” Adding a new object such as a force field to the package is straightforward, the procedure is the duplication of a similar object file, then renaming the file name and the class within it. The “interpreter” handles the creation, calls, and the deletion of the objects. It provides meaningful run-time warnings and errors as well as interprets the CAVIAR scripting language (CASL). An example of CASL is illustrated in Fig. 5.

The CAVIAR main directory is divided into subdirectories, which are listed as follows with a short description:
FIG. 5. An example of the CAVIAR scripting language (CASL). It simulates two similar atoms with Lennard-Jones interaction. The (blue) lines starting with “#” are comments. The red words are the base class. The words next to them are the child object type and the object name. The script obeys the object-oriented structure of the code. One can create unlimited numbers of any objects and use them. For example, here we have created two “Neighborlist” objects and only the first one is used in the “Md_simulator” object.

- **include**: C++ header files for the interpreter, objects, and utility.
- **src**: C++ source files including the package main function “main_CAVIAR.cpp” and the other source files.
- **doc**: Package documentation.
- **example**: Working scripts for the tutorials and references.
- **test**: Working scripts that has been used to test the code features.
- **scripts**: Bash-shell scripts for users and developers.
- **CMake**: Additional CMake script for the package configuration.
- **tools**: Some preprocessing and post-processing C++ codes alongside other tools.

CAVIAR objects are divided into 10 base classes. They are listed as follows:

- **Atom_data**: The container of all the atoms and molecules data that are active in a simulation.
- **Constraint**: It has different constraint algorithms such as NVE, “Berendsen,” and Nosé-Hoover thermostats. In addition, bond fixes with RATTLE, “SHAKE,” and M-SHAKE algorithms are also implemented.
- **Domain**: It has the specification of the simulation box. It handles the domain decomposition in the case of parallel runs.
- **Force_field**: The base class of all the force fields. The most important implemented force fields are Lennard-Jones (LJ), geometry with LJ, geometry spring, PLT, dissipative particle dynamics (DPD), electrostatic, periodic electrostatic with Ewald sums, spring bond, and spring angle.
- **Integrator**: Different integration schemes such as Euler, leap-frog, and velocity-verlet.
- **Md_simulator**: This object connects all the objects related to an MD simulation.
- **Neighborlist**: It has a cell-list and a verlet-list as its child objects. Short-range force fields need a pointer to this type of object.
- **Shape**: It has different objects that handle the boundary shapes. The geometric force fields need a pointer to this type of object.
- **Unique**: The objects that do not belong to any base classes are put here. For example “Distribution” generates initial positions of atoms.
• **Writer**: This object is responsible for writing simulation results to the output files.

PLT force field classes are implemented in a way that it does not have any potential formula. It uses other force field potential such as simple electrostatic or periodic electrostatic to calculate the potential values. In this way, the user does not need to create a new PLT class for a new type of potential. In addition, the PLT algorithm needs a finite element routine. CAVIAR currently uses deal. II,⁵¹,⁵² which is a C++ library for FE simulations on hexahedral meshes. CAVIAR can be configured without deal. II library too. In this case, it can do MD simulations without the PLT algorithm. There are some examples provided in the package and described in the user guide that works without deal. II, including a simple point charge extended (SPC/E) water model simulation and a polymer with fluctuations.

### B. Parallelization

CAVIAR runs in both sequential and parallel mode using message passing interface (MPI). While using PLT, every time step has MD and FE calculations and each of them needs a different method of parallelization. To make the MD procedure parallel, the simulation box decomposes into smaller boxes according to the number of processors, \( N_P \). This decomposition could be 1, 2, or 3-dimensional in a way that domains have the minimum common interface between each other. This method is called the spatial-decomposition algorithm and is discussed in detail by Plimpton.⁹ For FE parallelization, the mesh is divided into \( N_P \) parts. This division depends on the deal. II library installation. The current MPI implementation of the PLT algorithm in CAVIAR uses PETSC⁶⁰–⁶³ or TRILINOS⁶⁴–⁶⁸ library, depending on deal. II configurations.

Figure 6 illustrates three different ways of parallelization for a PLT simulation with two processors. Here, for the sake of illustration, we consider just four particles in a simulation box with a rough spherical mesh. Figure 6(a) is an illustration of sequential simulation, which is used as a reference for other cases. Here, the MD domain is shown by a black solid box, whereas the FE mesh is shown by a color-filled mesh. In Figs. 6(b) and 6(c) the MD procedure is done in parallel, while FE is sequential. In this case, all the processors have to solve the same FE mesh, but the particles are divided between processors depending on their position. Setting boundary values for FE requires having information on all the charged particles. To do that, all processes calculate the singular potential due to their dedicated charged particles on the boundary points and, then, proceed with an MPI collective summation over all processes. In Figs. 6(d) and 6(e), the FE procedure is parallel while MD is not. Therefore, the FE mesh is divided between processes while data of particles is the same for the processes. We have implemented this in a way that only the root process does the MD calculations, then it broadcasts the data to all processes. Using these complete data of the particles, each process evaluates the potential on the boundary points and then solves the partial mesh according to its implicit algorithm in the library. Subsequently, the electric field is interpolated and sent to the root process for MD integration.

The parallel cases of Figs. 6(f) and 6(g) are much more complex; in the previous cases, processes have all the information of MD or FE, while here no process has all of them. Besides, MD domain decomposition does not necessarily coincide with the FE mesh decomposition. This complexity raises a parallel efficiency problem. Both cases of Figs. 6(b) and 6(c) and Figs. 6(d) and 6(e) have been implemented in CAVIAR, while the last case will appear in the next version.
IV. PACKAGE ABILITIES

The CAVIAR software package is designed to be a general MD simulator. The physical systems that can be simulated with the first version of CAVIAR are atomic or molecular MD, such as the dynamics and rheology of uncharged or charged polymer, dynamics of massive particles with gravity, electric double-layer capacitors, porous media, water droplet simulations on flat or curved surfaces, and water desalination with polarized electrodes. Besides, it is possible to simulate granular materials with CAVIAR. In the granular material simulations, one can use polyhedrons to simulate realistic flat boundaries instead of rough walls that are generated by grains. In addition, the grains can be permanently charged and affected by electromagnetic fields, or they can electrostatically interact with metallic surfaces using the PLT algorithm.

As an example of the CAVIAR package, we have simulated a simple model of ionic-liquid inside a pore with an asymmetric shape. The pore has 1D periodicity in the z-axis direction. Figures 7(a) and 7(b) are snapshots of the system at the initial stage and at the $t = 12$ ns after running the simulation, respectively. Particles are arranged on a simple cubic lattice at the initial stage of the simulation, and the yellow/red particle color indicates the positive/negative electrical charges. The yellow/red electrode demonstrates an anode/cathode. In this example, the number of particles is 1276 and the time step is equal to $\Delta t = 6$ fs. There is a potential difference between the yellow and red boundaries in which its numeric value is equal to $\Delta V = 4$ V, whereas the temperature is set to be equal to 400 K. To make the governing equations of motion dimensionless, we proceeded as follow: first, we set the following numeric values for fundamental parameters; the length scale is the ion diameter $\sigma = 5$ Å, the mass unit is the mass of the hexafluorophosphate $\hat{m} = 144$ g/mol, and the energy is scaled with $\hat{\epsilon} = 1$ kJ/mol. With these parameters, temperature scales as $\hat{T} = \epsilon/k_B$ and the time is dimensionless with $\tau = \sqrt{\hat{m} \sigma^2/\hat{\epsilon}}$. The Langevin thermostat with a damping constant equal to $\xi = 10$ ps$^{-1}$ is used to control the overall temperature of the simulation. The particles are interacting through the LJ potential,

$$u(r_{ij}) = 4\epsilon \left( \frac{\sigma}{r_{ij}} \right)^{12} - \left( \frac{\sigma}{r_{ij}} \right)^{6} + u_0, \quad r_{ij} < r_{\text{cutoff}},$$

which is cut at $r_{\text{cutoff}} = 2^{(1/6)} \sigma$ and shifted by $u_0$ to fix the discontinuity due to cutting. This assures a fully repulsive LJ potential.

![FIG. 7. MD simulation of coarse-grained ionic-liquid between two polarized electrodes at 4 V. (a) Particle distribution at the initial moment. This distribution is done automatically using the ray-casting method (Sec. II B). The left (yellow) and right (red) shapes are the anodes and cathodes, respectively. The blue transparent surface is a solid boundary with no electrostatic interaction. (b) A snapshot of the system after evolution for 12 ns. (c) A perspective snapshot of the system at 12 ns. (d) The mesh structure generated for induced charge calculations. The color gradient indicates the dimensionless electrical potential value in the LJ reduced units.](image1)

![FIG. 8. (a) The mean density of the induced charge on different walls of the electrodes of Fig. 7 as a function of simulation time. Mesh boundaries 1–5, belonging to one electrode, and 6, belonging to another one, are plotted in (b). Note that (a) is the charge density and not the total charge.](image2)
In addition to the LJ, electrostatic interactions are also taken into account. There is another pure repulsive LJ interaction with $\epsilon_{\text{wall}} = 1$ kJ/mol and $\sigma_{\text{wall}} = 3.37$ Å between the particles and the electrode walls (Sec. II B). Figure 7(d) is a plot of the generated FE mesh to compute the smooth electrostatic potential. The color-bar at the top of this mesh indicates the voltage difference across the mesh. It is worth noticing that the numeric values of this color-bar belongs to the dimensionless potential and differs from the SI values. In this simulation, 1 V equals 96.4923 in dimensionless units.

Figure 8(a) is a plot of the dynamical behavior of induced charge density on different boundaries. These boundaries are schematically plotted in Fig. 8(b) and distinguished using different colors. A unique ID has been set to each boundary. All boundaries on the left (ID < 6) are equipotential surfaces and have a potential difference with that on the right (ID = 6). Figure 8(a) shows that nearly 15 ns is required to charge the capacitor at 4 V. At the equilibrium stage, different areas of the positive electrode have the same value of induced charge density. One can see in Fig. 7(b) that the positive (yellow) particles are attracted to the negative (red) electrode and vice-versa. This simulation is provided in the package’s user guide as an example.

V. PACKAGE PERFORMANCE

In this section, some performance facts of CAVIAR simulation with the PLT algorithm are presented. As for the only available alternative to the PLT algorithm, the ICC$^*$ algorithm is used to check the PLT performance. Currently, ICC$^*$ is only implemented in the ESPResSo software package. At the time of writing this work, the latest release of ESPResSo is its version 4.1.2. In this version, ICC$^*$ can only be used alongside a P3M Laplace solver. However, the CAVIAR-1.0 does not have P3M yet; instead, it has a basic Ewald method. It should be noted that the optimal number of the mesh refinement needs to be generated and recorded as the central processing unit (CPU)-time for an MD step. In the case of the ICC$^*$ algorithm, the electrode surfaces are made of static charged particles that are called ICC charges. The simulations are done for different numbers of ICC charges, and the CPU-time for one MD time step is calculated in the same way as before.

The simulation box contains some freely moving charges. We have set up two configurations for the simulations as follows:

(i) In this case, 100 positive/negative particles are set in a $10 \times 10$ grid in the $z = 7.5/2 = 12.5$ plane [Figs. 10(a) and 10(b)]. This configuration creates a dipole plane that induces charges on the metallic boundaries.

(ii) In the second case, 50 positive/negative charged atoms are randomly distributed in a cubic box of length $L = 10$ in the center of the simulation box [Figs. 10(c) and 10(d)]. This configuration creates approximately zero value for the induced charges.

Figure 11 contains the plots of simulation time and the induced charges for different configurations and algorithms. As it is clear, both algorithms have qualitatively similar behavior in CPU-time usage [Figs. 11(b) and 11(d)]. The induced charges [Figs. 11(a) and 11(c)] looks completely the same. Since increasing the number of refinements or ICC charges does not affect the value of the induced charges much, it shows that these two methods can work in low amount of resources.

However, in the PLT algorithm, we do not need to set particles to generate walls, but it is needed in the ICC$^*$ method. It should be noted that the optimal number of the mesh refinement needs to be investigated for each problems before long production runs.

The CAVIAR software package is developed to assist the scientists and researchers in their research. It is a new package but it has some advantages that can make it interesting for the users.

- It is easy to install on any computer, it has the least number of external dependency.
- It is a general purpose MD software, from MD atomic simulations up to granular and gravitational particles.

![FIG. 9. Visualizations of the PLT mesh at different levels of global refinement. The green area shows the conductive boundaries. (a) Initial non-refined mesh. (b) Refine level 1. (c) Refine level 2. (d) Refine level 3.](https://example.com/figure9.png)
FIG. 10. Visualization of the initial configuration of the benchmark systems. Red and yellow spheres are the positive and negative charged particles. The green spheres are the ICC charges, and the green plates indicate the position of metallic boundaries in the PLT mesh. (a) ICC* for case (i). (b) PLT for case (i). (c) ICC* for case (ii). (d) PLT for case (ii). Different levels of refinements are illustrated in Fig. 9.

FIG. 11. Comparing the CPU-time and the induced charge for different cases of PLT and ICC* algorithm. (a) Induced charge with CAVIAR. (b) CPU-time with CAVIAR. (c) Induced charge with ESPResSo. (d) CPU-time with ESPResSo. (a) and (c) shows similar behaviors. (b) and (d) are qualitatively similar. The simulations are done in one process on a computer with an Intel Pentium (R) CPU P6100 @ 2.00 GHz CPU.
• It has a particle-geometry interaction algorithm. Comparing with other algorithms, there is no need of particle distribution on the boundaries. This makes simulations of complex geometries possible by utilizing computer-aided-design (CAD) software.
• CAVIAR has the capability to simulate any kind of boundary conditions, including non-periodic or periodic in one or more dimensions.
• It has an automatic initial particle distribution generator, which is able to generate any desired distribution inside the simulation domain or in a given polyhedron. It also can mix multiple polyhedrons by using boolean operators to generate complex initial particle distributions.
• Some of the MD constraint algorithms are designed only for one type of the MD integrator, i.e., depending on the implemented integrator on the MD software, only some of the constraint can be added. As an example, one can refer to the SHAKE and RATTLE algorithm, both of which are used to simulate water molecules. The former is compatible with the SHAKE and the latter uses velocity-shake for integration. The CAVIAR is designed completely modular, so the choice of the integrator is user dependent and according to it, the suitable constraint can be selected by the user.
• It is easy to modify the CAVIAR code by the researchers and developers to add new features. It is completely object-oriented, and it has a developer-friendly software design.
• The CAVIAR structure has been completely re-designed many times to ensure a flexible and long-life code.
• It can be used as a C++ library in other codes.

VI. CONCLUSION

In this paper, the PLT algorithm for the MD simulations of charged particles under metallic boundary conditions is introduced. This algorithm was implemented with an FE Poisson solver; then, a test simulation for a point charge inside a conductive spherical shell is conducted. It was shown that PLT is highly accurate and the errors can easily be reduced with mesh refinement. In addition, another algorithm for the calculation of the interactions of MD particles with polyhedrons is described. With this algorithm, the polyhedrons can be used as MD walls. The advantage of these algorithms is that there is no need to put static MD particles as walls, which reduces the computation cost. It was shown that calculating the initial position of particles inside complex geometries can be done with a ray-casting algorithm.

These algorithms, as well as other standard MD methods, are implemented in the CAVIAR software package. This package can be used for a wide range of MD simulations. An example of its usage is given in this paper. CAVIAR is an object-oriented and highly extensible package. The package’s basic structure and the implemented methods are described in this paper. Different parallelization schemes for mixing FE and MD are also introduced.

This package has the capability to utilize other Poisson solving methods, such as the boundary element (BE) method. The BE method has higher efficiency of solving the Poisson equation in some special geometries. This feature will be added to the next version of the CAVIAR package. In addition, novel methods of the Poisson equation solving such as Isogeometric Formulation (IF) can be utilized in the PLT algorithm. In the IF method, instead of discretizing the boundaries into a polyhedron mesh, a continuous description of the boundary surfaces is used, which in turn helps to improve the performance of the PLT algorithm.

ACKNOWLEDGMENTS

M.B. acknowledges Morteza Jalalvand for his help and discussion on CAVIAR developments. This package is distributed under version-3.0 of GNU Lesser General Public License (LGPL): https://opensource.org/licenses/lgpl-3.0.html, and it is available at https://www.caviar-md.org and https://github.com/caviar-md.

The authors declare that they have no conflict of interest.

REFERENCES

1. M. Kleman and O. D. Lavrentovich, Soft Matter Physics: An Introduction (Springer Science and Business Media, 2007).
2. P. Simon and Y. Gogotsi, “Materials for electrochemical capacitors,” Nanoscience and Technology: A Collection Of Reviews From Nature Journals (World Scientific, 2010), pp. 320–329.
3. Z. Bo, C. Li, H. Yang, K. Ostrivko, J. Yan, and K. Cen, “Design of supercapacitor electrodes using molecular dynamics simulations,” Nano-Micro Lett. 10, 33 (2018).
4. J. Vatamanu, M. Vatamanu, O. Borodin, and D. Bedrov, “A comparative study of room temperature ionic liquids and their organic solvent mixtures near charged electrodes,” J. Phys.: Condens. Matter 28, 464002 (2016).
5. A. Vlad and A. Balducci, “Supercapacitors: Porous materials get energized,” Nat. Mater. 16, 161 (2017).
6. P. Biesheuvel, H. Hamelers, and M. Suss, “Theory of water desalination by porous electrodes with immobile chemical charge,” Colloids Interface Sci. Commun. 9, 1–5 (2015).
7. S. Porada, R. Zhao, A. Van Der Wal, V. Presser, and P. Biesheuvel, “Review on the science and technology of water desalination by capacitive deionization,” Prog. Mater. Sci. 58, 1388–1442 (2013).
8. M. Simoncelli, N. Gafoud, A. Sene, M. Haefele, B. Daffos, P.-L. Taberna, M. Salanne, P. Simon, and B. Rotenberg, “Blue energy and desalination with nanoporous carbon electrodes: Capacitance from molecular simulations to continuous models,” Phys. Rev. X 8, 021024 (2018).
9. J. S. Plimpton, “Fast parallel algorithms for short-range molecular dynamics,” J. Comput. Phys. 117, 1–19 (1995).
10. W. Smith and T. Forester, “DI POLY _2.0: A general-purpose parallel molecular dynamics simulation package,” J. Mol. Graphics 14, 136–141 (1996).
11. H.-J. Limbach, A. Arnold, B. A. Mann, and C. Holm, “ESPResSo—An extensible simulation package for research on soft matter systems,” Comput. Phys. Commun. 174, 704–727 (2006).
12. D. Halverson, T. Brandes, O. Lenz, A. Arnold, S. Bevc, V. Starchenko, K. Kremer, T. Stuehn, and D. Reith, “ESPResSo++: A modern multiscale simulation package for soft matter systems,” Comput. Phys. Commun. 184, 1129–1149 (2013).
13. P. P. Ewald, “Die Berechnung optischer und elektrostatischer Gitterpotentiale,” Ann. Phys. 369, 253–287 (1921).
14. A. Arnold and C. Holm, “MMD1D: A method for calculating electrostatic interactions in one-dimensional periodic geometries,” J. Chem. Phys. 123, 144103 (2005).
15. A. Arnold and C. Holm, ”MMD1D: A method for calculating electrostatic interactions in one-dimensional periodic geometries” [J. Chem. Phys. 123, 144103 (2005)].
16. A. Arnold and C. Holm, ”MMD2D: A fast and accurate summation method for electrostatic interactions in 2D slab geometries,” Comput. Phys. Commun. 148, 327–348 (2002).
17. S. A. Ghasemi, A. Neelov, and S. Goedecker, “A particle-particle, particle-density algorithm for the calculation of electrostatic interactions of particles with slaklike geometry,” J. Chem. Phys. 127, 224102 (2007).

18. M. Deserno and C. Holm, “How to mesh up ewald sums. I. A theoretical and numerical comparison of various particle mesh routines,” J. Chem. Phys. 109, 7678–7693 (1998).

19. M. Deserno and C. Holm, “How to mesh up ewald sums. II. An accurate error estimate for the particle–particle–particle-mesh algorithm,” J. Chem. Phys. 109, 7694–7701 (1998).

20. A. Arnold, J. de Joannis, and C. Holm, “Electrostatics in periodic slab geometries,” J. Chem. Phys. 117, 2496–2502 (2002).

21. J. de Joannis, A. Arnold, and C. Holm, “Electrostatics in periodic slab geometries II,” J. Chem. Phys. 117, 2593–2512 (2002).

22. S. Tyagi, A. Arnold, and C. Holm, “ICM12MD: An accurate method to include planar dielectric interfaces via image charge summation,” J. Chem. Phys. 127, 154723 (2007).

23. S. Tyagi, A. Arnold, and C. Holm, “Electrostatic layer correction with image charges: A linear scaling method to treat slab 2D+ h systems with dielectric interfaces,” J. Chem. Phys. 129, 204102 (2008).

24. S. Tyagi, M. Süzen, M. Sega, M. Barbosa, S. S. Kantorovich, and C. Holm, “An iterative, fast, linear-scaling method for computing induced charges on arbitrary dielectric boundaries,” J. Chem. Phys. 132, 154112 (2010).

25. S. Kesselheim, M. Sega, and C. Holm, “The ICC algorithm: A fast way to include dielectric boundary effects into molecular dynamics simulations,” arXiv:1003.1277 (2010).

26. A. Arnold, O. Lenz, S. Kesselheim, R. Weeber, F. Fahrenberger, D. Roehm, P. Kosovon, and C. Holm, “ESPResSo 3.1: Molecular dynamics software for coarse-grained models,” in Meshfree Methods for Particular Differential Equations VI (Springer, 2013), pp. 1–23.

27. B. Doda, D. Gillespie, W. Nonner, D. Henderson, and B. Eisenberg, “Computing induced charges in inhomogeneous dielectric media: Application in a Monte Carlo simulation of complex ionic systems,” Phys. Rev. E 69, 046702 (2004).

28. K. Breitsprecher, M. Abele, S. Kondrat, and C. Holm, “The effect of finite pore length on ion structure and charging,” J. Chem. Phys. 147, 104708 (2017).

29. K. Breitsprecher, C. Holm, and S. Kondrat, “Charge me slowly, I am in a hurry: Optimizing charge–discharge cycles in nanoporous supercapacitors,” ACS Nano 12, 9733–9741 (2018).

30. K. Reed, O. J. Lanning, and P. A. Madden, “Electrochemical interface between an ionic liquid and a model metallic electrode,” J. Chem. Phys. 126, 084704 (2007).

31. J. I. Siepmann and M. Sprik, “Influence of surface topology and electrostatic potential on water/electrode systems,” J. Chem. Phys. 102, 511–524 (1995).

32. C. Merlet, M. Salanne, and B. Rotenberg, “New coarse-grained models of imidazolium ionic liquids for bulk and interfacial molecular simulations,” J. Chem. Phys. 116, 7687–7693 (2012).

33. C. Merlet, D. T. Limmer, M. Salanne, R. Van Roij, P. A. Madden, D. Chandler, and B. Rotenberg, “The electric double layer has a life of its own,” J. Phys. Chem. C 118, 18291–18298 (2014).

34. A. C. Forse, C. Merlet, J. M. Griffin, and C. P. Grey, “New perspectives on the charging mechanisms of supercapacitors,” J. Am. Chem. Soc. 138, 5731–5744 (2016).

35. R. Rostami, S. A. Ghasemi, and E. N. Oskoei, “A highly accurate and efficient algorithm for electrostatic interactions of charged particles confined by parallel metallic plates,” J. Chem. Phys. 145, 124118 (2016).

36. D. Jackson, Classical Electrodynamics (John Wiley & Sons, 2007).

37. H. Lee and W. Cai, “Ewald summation for Coulomb interactions in a periodic lattice,” J. Chem. Phys. 116, 7694–7701 (1998).

38. S. Nosé, “A unified formulation of the constant temperature molecular dynamics methods,” J. Chem. Phys. 81, 511–519 (1984).

39. W. C. Hoover, “Canonical dynamics: Equilibrium phase-space distributions,” Phys. Rev. A 31, 1695 (1985).

40. H. C. Andersen, “Rattle: A “velocity” version of the shake algorithm for molecular dynamics calculations,” J. Comput. Phys. 52, 24–34 (1983).

41. P. R. Ypma, G. Ciccotti, and H. J. Berendsen, “Numerical integration of the cartesian equations of motion with a system with constraints: Molecular dynamics of n-alkanes,” J. Comput. Phys. 23, 327–341 (1977).

42. V. Kraüler, W. F. Van Gunsteren, and P. H. Hüenerberger, “A fast SHAKE algorithm to solve distance constraint equations for small molecules in molecular dynamics simulations,” J. Comput. Chem. 22, 501–508 (2001).

43. S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, “PETSc web page,” http://www.mcs.anl.gov/petsc (2019).

44. S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, “PETSc users manual,” Technical Report No. ANL-95/11, Revision 3.11, Argonne National Laboratory, 1999.
M. Heroux, R. Bartlett, V. H. R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams, “An overview of trilinos,” Technical Report No. SAND2003-2927, Sandia National Laboratories, 2003.

M. A. Heroux, J. M. Willenbring, and R. Heaphy, “Trilinos developers guide,” Technical Report No. SAND2003-1898, Sandia National Laboratories, 2003.

M. A. Heroux, J. M. Willenbring, and R. Heaphy, “Trilinos developers guide Part II: ASCI software quality engineering practices version 1.0,” Technical Report No. SAND2003-1899, Sandia National Laboratories, 2003.

M. A. Heroux and J. M. Willenbring, “Trilinos users guide,” Technical Report No. SAND2003-2952, Sandia National Laboratories, 2003.

M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, “An overview of the trilinos project,” ACM Trans. Math. Software 31, 397–423 (2005).

F. Weik, R. Weeber, K. Szuttor, K. Breitsprecher, J. de Graaf, M. Kuron, J. Landsgeell, H. Menke, D. Sean, and C. Holm, “ESPResSo 4.0—An extensible software package for simulating soft matter systems,” Eur. Phys. J.: Spec. Top. 227, 1789–1816 (2019).

B. Hassani, N. Moghaddam, and S. Tavakkoli, “Isogeometrical solution of Laplace equation,” Asian J. Civil Eng. 10, 579 (2009).

J. Dölz, H. Harbrecht, S. Kurz, S. Schöps, and F. Wolf, “A fast isogeometric BEM for the three dimensional Laplace- and Helmholtz problems,” Comput. Methods Appl. Mech. Eng. 330, 83–101 (2018).

J. A. Cottrell, T. J. Hughes, and Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA (John Wiley & Sons, 2009).