Multiscale simulation of plasma flows using active learning

A. Diaw,[*] K. Barros, J. Haack, C. Junghans, B. Keenan, Y. W. Li, D. Livescu, N. Lubbers, M. McKerns, R. S. Pavel, D. Rosenberger, I. Sagert, and T. C. Germann
Los Alamos National Laboratory, Los Alamos, New Mexico 87544, USA

(Received 15 May 2020; accepted 3 August 2020; published 24 August 2020)

Plasma flows encountered in high-energy-density experiments display features that differ from those of equilibrium systems. Nonequilibrium approaches such as kinetic theory (KT) capture many, if not all, of these phenomena. However, KT requires closure information, which can be computed from microscale simulations and communicated to KT. We present a concurrent heterogeneous multiscale approach that couples molecular dynamics (MD) with KT in the limit of near-equilibrium flows. To reduce the cost of gathering information from MD, we use active learning to train neural networks on MD data obtained by randomly sampling a small subset of the parameter space. We apply this method to a plasma interfacial mixing problem relevant to warm dense matter, showing considerable computational gains when compared with the full kinetic-MD approach. We find that our approach enables the probing of Coulomb coupling physics across a broad range of temperatures and densities that are inaccessible with current theoretical models.

DOI: 10.1103/PhysRevE.102.023310

I. INTRODUCTION

In plasma modeling, a number of dynamical and structural data need to be collected at microscopic scales using quantum molecular dynamics simulations [1–3], for example. Experimental results and their analysis, on the other hand, are determined by measurements at the macroscopic scale in space over long scales in time [4,5]. Therefore, one major disparity that is currently inhibiting progress in this area is the extrapolation of microscale information into macroscopically relevant scales. Inertial confinement fusion implosions [6–8], for example, are fundamentally multiscale in nature; an accurate understanding of the connection between experimental observables and the underlying microphysics is needed. Large-scale properties of these systems [9–12] are crucially affected by microscale information such as equations of state, and ionic and electronic transport coefficients. This microscopic information is often incorporated in hydrodynamic codes [13] through theoretical models rather than the more reliable atomistic simulations; however, molecular dynamics (MD) [14] simulations cannot reach engineering scales. For these reasons, a large number of multiscale techniques have been developed to enable scale bridging between MD simulations and meso- and macroscale models.

In most multiscale methods, a well-defined set of equations at the macroscale is assumed, and the appropriate closure information is then computed by a finer-scale model and communicated to the macroscale. These approaches perform well for relatively simple systems [15], but performing MD simulations at every time step for each of the macroscale grid cells can quickly become impractical without some sort of guidance [16,17].

We present here an approach to enhancing the performance of multiscale modeling by using active learning (AL) [18,19] to iteratively build surrogate models of the fine-scale response. Specifically, a machine learning model is trained on data from fine-scale simulations and responds quickly to queries from the coarse-scale simulation. If the machine learning model believes that it cannot accurately respond to a given query, then new fine-scale simulations are spawned and their results added to the training dataset. We use a the "query-by-committee" algorithm [20] to decide when to obtain new data. We employ an ensemble of neural networks to model the fine-scale data. Ensemble averaging over multiple models is a useful variance reduction technique. More importantly, the *ensemble variance* is a metric for uncertainty of the machine learning prediction. New fine-scale simulations are launched whenever this uncertainty metric (ensemble variance) exceeds a threshold.

We demonstrate this multiscale coupling method through investigation of materials mixing in hot dense plasma. This problem is relevant to the Marble experimental campaign [21–23] which attempts to quantify the effects of the mixing on Inertial Confinement Fusion [24]. In this experiment, target capsules are filled with a deuterated foam with engineered pores of a specified diameter. The pores are filled with a gas containing tritium. In our case, we are solving the multi-ionic Vlasov-Bhatnagar-Gross-Krook kinetic equation [25,26], which we evolve using a standard second-order finite-volume scheme with the monotonized central difference limiter [26]. The electrons are assumed to be in thermal equilibrium and treated as a fluid characterized by its density n_e and T_e. The communication and interfaces between the computing platform, surrogate modeler, coarse-scale code, and fine-scale code is handled by the Generic Learning User Enablement (GLUE) code.

[*] diaw@lanl.gov

This paper is organized as follows. Section II A describes our multiscale method that couples molecular dynamics to kinetic theory (KT) models and discusses how the missing information in KT is obtained from MD simulations. In Sec. II B we introduce the active learning algorithms, and in Sec. II C we present the features of the GLUE code, which handles the communication between the fine-scale model, coarse-scale model, and the surrogate model. A demonstration application of this framework to atomistic mixing in hot dense plasma follows in Sec. III. Finally, in Sec. IV, we give some brief conclusions and discuss ongoing extensions of this framework.

II. COUPLING SCHEME

In this section, we describe the components of the multiscale framework. First, we present the fine- and coarse-scale models and provide some details on how they are connected. Next, we give the details of the active learning surrogate model. Finally, we discuss some of the computer science infrastructure required to interface these components together.

A. Heterogenous multiscale method for plasma

We consider two levels of approximation in this work: Molecular dynamics is the most accurate and taken to be our ground truth, while KT is a compromise in terms of speed and accuracy of the physics with respect to MD. Below, we briefly describe the two models and specify the information that needs to be passed between the two scales.

We begin with the Hamiltonian formulation of classical mechanics to describe a plasma mixture of N_ν species. Each species i has N_i particles for a total of $N = \sum_{i=1}^{N_\nu} N_i$ particles. The Hamiltonian of this system is

$$H = \frac{1}{2}\sum_{\alpha=1}^{N}\left[m_\alpha \mathbf{u}_\alpha \cdot \mathbf{u}_\alpha + \sum_{\beta=1,\beta\neq\alpha}^{N} \phi_{\alpha\beta}(|\mathbf{r}_\alpha - \mathbf{r}_\beta|)\right], \qquad (1)$$

where m_α, $\mathbf{u}_\alpha$, and $\mathbf{r}_\alpha$ are the mass, velocity, and position of particle α, respectively, and $\phi_{\alpha\beta}(r)$ is the two-body interaction potential appropriate to the species of particles indexed by α and β. The dynamics is governed by Hamilton's equations:

$$\dot{\mathbf{r}}_\alpha = \frac{1}{m_\alpha}\nabla_{\mathbf{u}_\alpha} H = \mathbf{u}_\alpha, \qquad (2)$$

$$\dot{\mathbf{u}}_\alpha = -\frac{1}{m_\alpha}\nabla_{\mathbf{r}_\alpha} H = -\frac{1}{m_\alpha}\nabla_{\mathbf{r}_\alpha}\sum_\beta \phi_{\alpha\beta}. \qquad (3)$$

One typically solves these equations using a symplectic integration scheme that exactly conserves an approximate energy function, thereby yielding good global accuracy. Statistics collected from MD can in principle provide numerical validation of any classical theory. Periodic boundary conditions were used to best approximate the large system size limit. In this work, MD simulations were carried out with the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) code [27]. Ions interactions $\phi_{\alpha\beta}$ were modeled with a screened-Coulomb potential [9],

$$\phi_{\alpha\beta}(r) = \frac{Z_\alpha^* Z_\beta^* e^2}{r}\exp(-r/\lambda), \qquad (4)$$

where Z_α^* and Z_β^* are the ionization charges of ions α and β, e is the elementary charge, and λ is the electronic screening length given by

$$\lambda^{-2} = \frac{4\pi e^2 n_e}{\sqrt{T^2 + \left(\frac{2}{3}E_F\right)^2}}. \qquad (5)$$

Here the electron density is estimated using $n_e = \langle Z^*\rangle n$, with Z^* the average charge of the ions and n the ionic total density, and E_F is the Fermi energy. The mean ionization charge state Z^* of the different species are estimated using the Thomas-Fermi model [28]. This potential has been shown to give qualitatively good results for the warm dense conditions considered in this work [29,30]. However, for nondegenerate electrons, the electron-ion interaction needs accommodation of quantum effects. Since the pioneering work of Hansen and McDonald [31], quantum statistical potential (QSP) has been a popular approach to accounting for electron-ion quantum degeneracy effects [32,33]. Dutta and Dufty [34] compared modified-Kelgb potential [35] against the gold standard of path-integral quantum Monte Carlo (PIMC) [36,37]; showing an extremely wide range of physical conditions where QSPs are nearly perfect, it is only at very low densities that we can see a modest deviation. Further improvements of the electron-ion interactions are under way using effective ion potential from the average atom model of Refs. [38,39]. It combines an average atom model based on density-functional theory to calculate the electronic structure of the plasma with the integral equations of fluid theory for the ion interactions, enabling an efficient computation of the plasma properties.

Our starting point for coupling MD with KT (and, in the future, hydrodynamics) is through transport coefficients, in this case the mutual diffusion coefficients. Transport processes are the manifestation of nonequilibrium effects occurring at microscopic scale. Using the fluctuation-dissipation theorem, Green and Kubo [40,41] have shown that these nonequilibrium effects are embedded in equilibrium correlation functions [42], thus providing a way to estimating transport coefficients through the correlation functions of equilibrium molecular dynamics simulations.

The Green-Kubo formula uses the velocity autocorrelation function to determine the self-diffusion coefficient of species i,

$$D_i = \frac{1}{3}\int_0^\infty dt\,\langle \mathbf{u}_\alpha(t - t_0)\cdot \mathbf{u}_\alpha(t_0)\rangle_{\{\alpha,t_0\}}. \qquad (6)$$

The angular brackets represent an average over all N_s particles (indexed by α) of species i and over all statistically equivalent initial times t_0. The net momentum of the box is taken to be zero.

For mixtures, the mutual diffusion $D_{i,j}$ between species i and j can be expressed compactly if one neglects cross-correlation between particles. Under this assumption, the Darken relation is [43]

$$D_{i,j} = D_i D_j \sum_{k=1}^{N_k}\frac{x_k}{D_k}, \qquad (7)$$

where x_i is the density fraction of species i. The Darken relation has been demonstrated to give accurate results of interdiffusion for mixtures across coupling regimes as shown in Refs. [43–45].

In the multiscale coupling scheme, the coarse-scale model will require measurements of the diffusion D_i and mutual diffusion D_{ij} coefficients at specific temperature T and species fractions x_i. The required steps for the MD simulation are as follows: (1) Estimate the mean-free paths of all species; (2) set the simulation domain to four times the largest species mean free path, which is sufficiently large to avoid Knudsen finite-size effects [46–48]; (3) randomly populate the simulation domain with particles according to the target species densities; (4) perform an initial MD simulation with a Nosé-Hoover thermostat enabled to equilibrate the positions and velocities according to the Boltzmann distribution at fixed temperature for 5×10^4 time steps (we choose a time step of $10^{-2}/\omega_p$, where ω_p is the ionic plasma frequency, to maintain constant total energy over the simulation duration); (5) perform an MD simulation in the microcanonical ensemble to determine trajectories and generate velocity fluctuations for 10^5 time steps; and (6) use Eqs. (6) and (7) to compute the mutual diffusion coefficients. The MD results are then stored in an SQLite database.

Our coarse-scale model is a multicomponent Bhatnagar-Gross-Krook (BGK) [25,49] model which was derived by approximating the collision operator in the Boltzmann equations with a relaxation-time form assuming a near equilibrium approximation. The ion dynamics is described by the BGK equation, which can be expressed as

$$\frac{\partial f_i}{\partial t} + \mathbf{v} \cdot \nabla f_i + \frac{Z_i e}{m_i} \mathbf{E} \cdot \nabla f_i = \sum_j Q_{ij}^{\mathrm{BGK}}[f_i, f_j], \tag{8}$$

where $f_i(\mathbf{r}, \mathbf{v}, t)$ is the distribution function of species i ions and $\mathbf{E}$ is the electric field which satisfies Gauss's law,

$$-\frac{1}{4\pi e} \nabla \cdot \mathbf{E} = n_e(\mathbf{r}, t) - \sum_{i=1} Z_i \int d\mathbf{v} f_i(\mathbf{r}, \mathbf{v}, t), \tag{9}$$

where n_e is the electron density. Electrons are treated as a background fluid with a density n_e and a temperature T_e, within the finite-temperature Thomas-Fermi model [28]. Finally, the BGK operator is given by:

$$Q_{ij}^{\mathrm{BGK}}[f_i, f_j] = \nu_{ij}(\mathcal{M}_{ij}[f_i, f_j] - f_i), \tag{10}$$

where ν_{ij} correspond to the average collision frequencies for a single i particle with the j particles and $\mathcal{M}_{ij}$ are target distribution functions,

$$\mathcal{M}_{ij}[f_i, f_j] = n_i \left(\frac{m_i}{2\pi k_j T_{ij}} \right)^{3/2} \exp\left[-\frac{m_i(\mathbf{v} - \mathbf{u_{ij}})^2}{2k_j T_{ij}} \right], \tag{11}$$

which are constructed to satisfy particle number, momentum, and energy conservation, as well as Boltzmann's H-theorem. Expressions of the parameters, $\mathbf{u}_{ij}$, T_{ij} can be found in Ref. [26].

The KT model requires closure information in terms of the collision frequencies ν_{ij}. While there is some freedom in how these parameters are defined, typically they are constructed such that they give the Boltzmann diffusion coefficients in the hydrodynamic limit [49–51]. For a binary mixture, this formula is simply

$$\nu_{ij} = \frac{k_B T}{\mu_{ij} D_{ij}}, \tag{12}$$

where $T = (m_i T_i + m_j T_j)/(m_i + m_j)$ is the average temperature, μ_{ij} the reduced mass between species i and j, and D_{ij} is the mutual diffusion between species i and j. Molecular dynamics simulations [30,44,52–54] are accepted as ground truth for microphysical properties such as interdiffusion in high-energy density plasmas. Our goal here is to gather these values from small MD simulations so that the KT equations can be closed with MD-accurate coefficients. Both the BGK and (eventually) hydrodynamic models assume that the system is near a local thermodynamic equilibrium, so we can exploit timescale separation and use an MD simulation embedded in the macroscopic cell with periodic boundary conditions to compute local transport properties.

B. ML surrogate model: Ensemble of deep neural networks

Collecting molecular dynamics information for each cell in our coarse-scale models is very expensive. To reduce this cost we can use a surrogate model to inform the kinetic code. However, it is important to do so in a way that captures the behavior of MD simulations across the entire space of potential MD simulations. One way to reduce the cost of many MD calculations is to use machine learning; a significant but static number of MD calculations can be done ahead of time, potentially in parallel, across the space of MD inputs, and a surrogate model built on top of these calculations can be applied for any point in the parameter space. In some existing approaches, precomputed tables (e.g., Ref. [55]) are interpolated. However, a table-building approach will not scale well to mixtures with many species, as the size of a table scales exponentially with the dimensionality of the input space of the problem. In addition, another danger here is that it may be difficult to anticipate which MD calculations are needed ahead of time. To reduce this cost, we employ an approach using active learning. In this paradigm, we run some number of MD simulations ahead of time to build an initial surrogate model. These points were chosen by randomly sampling the parameter space. When we query the surrogate model, we also request a model uncertainty; if the model reports that its prediction carries a large uncertainty, then we spawn a new MD simulation to collect ground-truth data. Periodically, as new MD data become available, we retrain the surrogate model using all available MD data. In this way, machine learning forms a flexible surrogate that can adapt to new macroscale conditions on the fly. The availability of a high-quality surrogate model significantly reduces the computational time spent performing MD.

Since a coarse-scale simulation may contain a large number of cells and run for many time steps, it is important to select a machine learning architecture that can perform well with large datasets. Given this, we find it prudent to choose a parametric model over nonparametric models; nonparametric models typically become slower to evaluate as the dataset becomes larger because the model complexity scales with the training dataset. It is also important to choose a method that leads to a simple but effective uncertainty quantification scheme. We selected an ensemble of neural networks as a good candidate, with the ensemble variance as a metric of model uncertainty. This is an invocation of the query-by-committee algorithm [20], an active learning algorithm where new training points are selected based on the disagreement of an ensemble ("committee") of models. We emphasize that there are many possible models and uncertainty quantification strategies available and that the GLUE code can take advantage of any of them; our choice is pragmatic and we believe it is well motivated, but future research can explore many different possibilities for surrogate modeling. We implement the following neural network scheme using the neural network library Pytorch [56].

The inputs and the outputs of the problem are as follows: Given the $n_{\rm in} = 5$ inputs to MD (temperature, two densities, and two ionization states), provide $n_{\rm out} = 3$ diffusion coefficients (two self and one mutual). For this, we train a multitask regression model. A neural network consists of a series of *layers*, each of which has many *neurons*. Each layer takes a vector of inputs, x, and produces a vector of *activations*, x', one for each neuron, via the following equation:

$$x' = f(Wx + b). \tag{13}$$

Here W is a *weight matrix*, b is a *bias vector*, and f is an *activation function*, a differentiable, nonlinear function that is applied element-wise over the input vector. We choose the Rectified Linear Unit (ReLU) function $f(a) = \max(0, a)$. The network is built from $n_{\rm layers} = 6$ layers connected in series, such that the output of each layer is fed as input into the next. We choose the number of neurons in intermediate layers, also called *hidden* layers, to be $n_{\rm hidden} = 64$, while the input and output sizes of $n_{\rm in} = 5$ and $n_{\rm out} = 3$ for the first and last layers, respectively, are fixed to match the size of the MD data space.

A network must be trained to capture the behavior of a dataset; this consists of adjusting the parameters θ (weights and biases) of each layer to produce a satisfactory model. We use a typical formulation based on the minimization of a *loss function* $\mathcal{L}$ based on the mean-squared error:

$$\mathcal{L} = \frac{1}{n_{\rm batch}} \sum_{i,t} \left(\hat{D}_i^t - D_i^t\right)^2, \tag{14}$$

where index i runs over the training examples and t runs over training targets. D_i^t represents the t-th diffusion coefficient for example i and $\hat{D}_i^t$ the prediction of the network, that is, the output of the t-th neuron in the last layer of the network. This output is minimized by accelerated gradient descent with the Adam optimizer [57] in a series of *epochs*, that is, passes over the entire training dataset, using batches of size $n_{\rm batch} = 50$; for each batch, the loss function is evaluated, the gradients $\partial\mathcal{L}/\partial\theta$ computed, and the optimizer invoked to adjust the parameters of the network. Gradients are computed using automatic differentiation, which is built into the PyTorch [56] library used to implement the network. Ten percent of a training dataset is withheld from gradient descent for validation with early stopping. We use a patience-based learning scheduler that halves the learning rate if the validation loss does not improve for $n_{\rm patience} = 20$ epochs and stops training and reverts to the last best network if the validation loss does not improve for $2n_{\rm patience}$ epochs, with a hard cap of $n_{\rm epochs} = 2000$ at maximum. In practice, networks terminate within a few hundred epochs. Both the input to and output from the network are empirically normalized using the mean and standard deviation of the training data and converted from and to physical units when a request is received from the GLUE code.

To build an ensemble for query-by-committee uncertainty, train each network to a random subselection of the database; we withhold a random 10% of the data from the network for calibration. Between the random split of the data and the random initialization of network weights, each member of the ensemble produces a different model. For each model, we check performance of the network on its calibration data to ensure that the coefficient of determination $R^2 >= 0.7$ to ensure the model is reasonable and reject it if not. The process is continued until $n_{\rm ensemble} = 5$ networks have been successfully trained.

We flag model outputs as requiring MD information based on a threshold on the standard deviation of ensemble predictions. To establish this threshold, we compare the typical ensemble deviation to the typical network error. The average RMS error of the networks applied to their calibration folds is measured and recorded as a typical error scale $E_{\rm cal}$. We also record the mean absolute error of the ensemble on the entire dataset as $E_{\rm ens}$ and the average standard deviation of the networks as σ. We form a typical disagreement S and per-prediction quality flag s_i for any example i based on the ensemble standard deviation of predictions σ_i:

$$S = \sigma \frac{E_{\rm cal}}{E_{\rm ens}}, \tag{15}$$

$$s_i = k\frac{\sigma_i}{S}. \tag{16}$$

If $s_i \geqslant 1$, then a prediction is marked for simulation; predictions with $s_i < 1$ are trusted. Since $\bar{\sigma}$ is applied to the entire set of training points, the calibration factor $\bar{E}_{\rm cal}/\bar{E}_{\rm ens}$ establishes a scale linking the performance of the ensemble on the entire training data to the performance individual models on unseen calibration data; S represents the expected model fluctuations on unseen data similar to the training set. The constant $k = 3$ controls how stringently we require the ensemble to agree in order to defer to MD; setting $k = 1$ would request MD for any sample above typical model disagreement. We select $k = 3$, requiring MD when model fluctuations are significant. This procedure is applied for each target t independently.

We tested the training algorithm on a database of 1100 points generated from MD. Each point in the database is tested once by an ensemble that had no access to that point for training by using 10-fold cross-validation procedure. The network performance is shown in Fig. 1. For each target, the coefficient of determination R^2 is greater than 0.98, indicating that the training strategy produces good-quality models. Points are colored by the quality flag s_i, and points that would be selected for MD verification ($s_i > 1$) are circled in red. In total across all targets, this algorithm selects 31 points, or less than 3% of the dataset. The network is less confident about regions where the diffusion coefficient is large, and data are only sparsely available. These results demonstrate that the approach is successful, but we emphasize that our modular

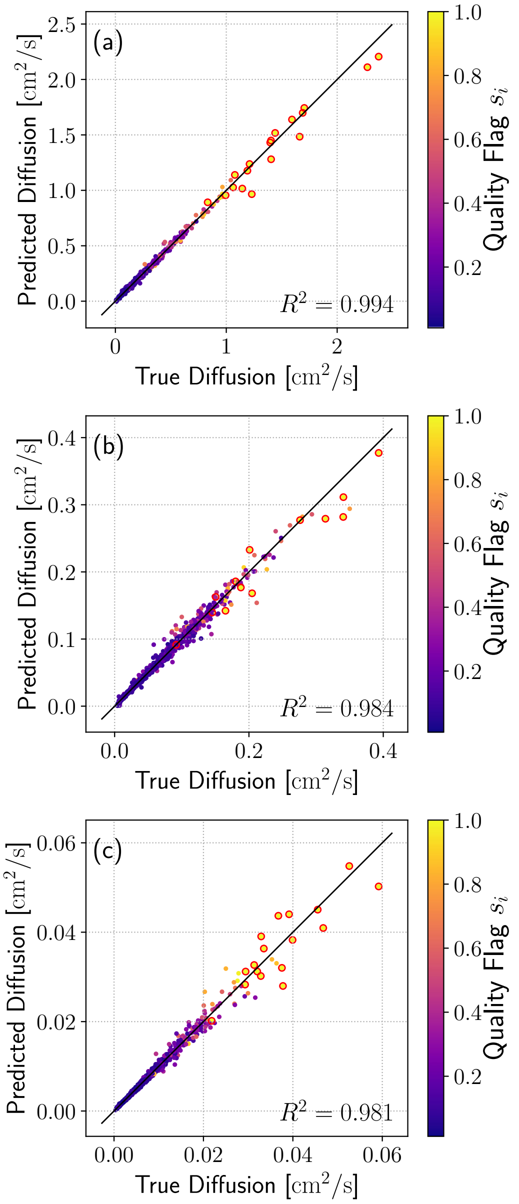

FIG. 1. Test of ML surrogate model performance for all the mutual diffusion points in the test dataset which were generated by randomly sampling 1100 points from a five-dimensional input parameter space. (a) Deuterium-deuterium mutual diffusion, (b) argon-deuterium mutual diffusion, and (c) argon-argon mutual diffusion. Points selected for MD verification are circled in red.

GLUE code framework allows future work to improve on this strategy or to use different surrogate models entirely.

The communication between the KT, MD, and the surrogate models is done using GLUE, which is described in the next section.

C. GLUE workflow

To facilitate this coupling, we built on previous work [16,58,59] on multiscale coupling of scientific codes. At its simplest, we determine what physical properties need to be exchanged between the various scales and derive application programming interfaces (APIs) from these. We then provide a lightweight infrastructure to communicate these physical properties between the various scientific codes as shown in Fig. 2.

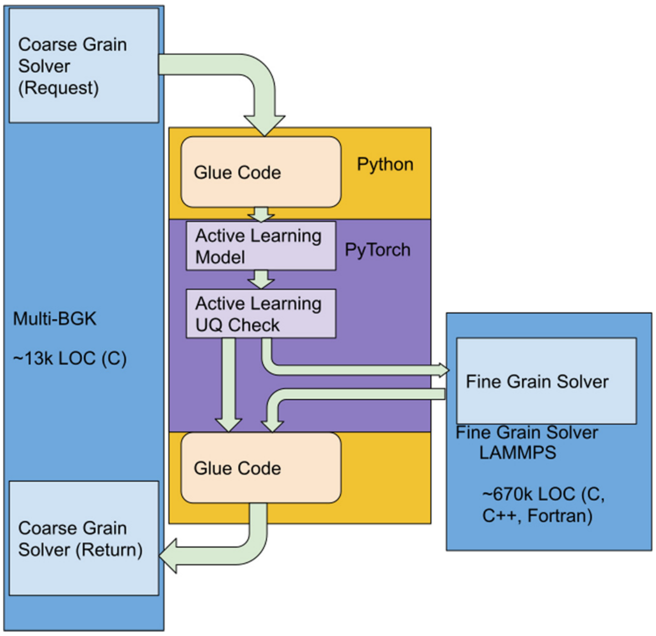

FIG. 2. Sample of the GLUE code implementation of our microscale-macroscale coupling. On the macroscale simulation (left) sending a request, the GLUE code (center) uses the active learning algorithms (center purple) to determine if the model's uncertainty quantification is such that a new fine grain simulation (right) needs to be called. Then either the result of the fine grain simulation or the model's prediction are returned to the macroscale simulation.

With these APIs we are able to create a very modular system where each component can be swapped out for a different implementation. Currently, our infrastructure is written with a collection of commodity software, including SQLite, but we are in the process of evaluating tools such as Lawrence Livermore National Laboratory's Flux scheduler [60] as a more exascale oriented workflow manager. Similarly, we rely on LAMMPS [27] for our fine-scale calls but we could easily switch to a different MD solution so long as it had similar scientific capabilities. And for the purposes of the machine learning and uncertainty quantification we use the popular PyTorch library [56] but, again, interface in such a way that different solutions can be slotted in and out.

This modularity is largely the key to allowing our approach to replace a scientific code with a machine learning solution. By treating the scale bridging infrastructures as an explicitly defined contract we are able to swap out these implementations. If a neural net is capable of accepting the same inputs and yielding the same outputs as an MD simulation, then it is functionally identical and we are able to switch between solutions at will.

Since the effectiveness of machine learning methods vary depending on the available training data and problem, we add in an additional check for any machine learning solution. In our infrastructure code we query the models provided by the machine learning solution for the output associated with our inputs. Our surrogate model then provides both the expected

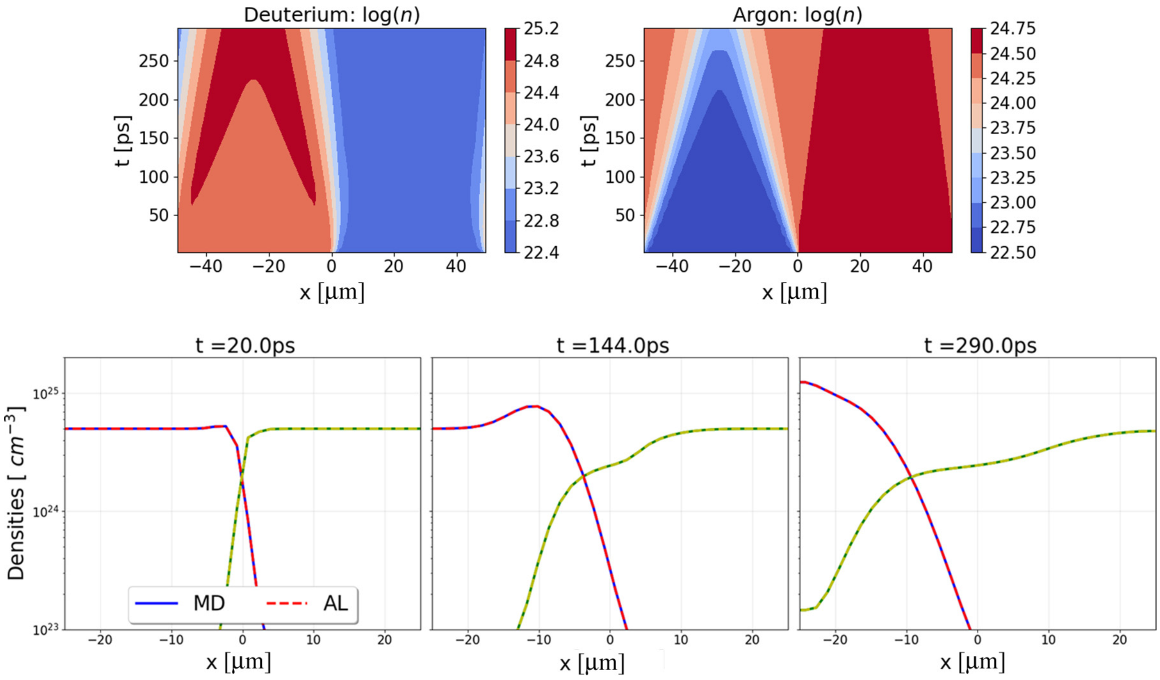

FIG. 3. Top: predicted densities $n(x, t)$ at time $t = 290$ ps. We are initially using an SQL database containing 1100 MD points generated by randomly sampling the parameter space. Bottom: Comparison of the active learning (dashed lines) and the MD (solid lines) at three different times. An MD run using LAMMPS took approximately 7 min and 50 s per call on a 36 core Skylake. Surrogate model training took around 82 s, and surrogate model evaluation takes milliseconds or less. The model error in the case is 10^{-5}.

outputs as well as the uncertainty quantification to indicate its confidence that this specific model gave a valid answer. If the confidence is too low, then we fall back to calling the actual MD simulation and provide the data for the machine learning solution to retrain and generate a new surrogate model for future use.

III. RESULTS

In this section we will demonstrate the performance of our multiscale coupling by investigating interfacial mixing of a light species (deuterium) and heavy species (argon) in the warm dense matter regime. These two species were chosen because they are found in the foam and gas, respectively, of the Marble experiment [21–23]. While the full experiment consists of four (or possibly more) species, we begin with this relatively simple binary mixture of the lightest and heaviest element in the capsule.

We first consider the case of materials atomically mixing due to a uniform, constant background electron temperature. Next, we further test the robustness of the active learning component by considering a linearly increasing background electron temperature, which will move the physical conditions outside the realm of our initial training dataset. This will lead the machine learning algorithm into a parameter space region with high uncertainty and will therefore require more fine-scale calculations and on-the-fly updates of the surrogate model.

A. Test problem 1 (instantaneous heating)

In this problem, we consider a one-dimensional deuterium-argon interface case with equal concentrations, initially at rest. The material number densities are $n = 10^{25}$ cm^{-3}, with a material temperature of 100 eV. A background electron temperature of 100 eV is kept constant over the simulation time, which is meant to model the effects of preheat radiation on the interface. The kinetic model is discretized with 64 points in space and a 40^3 velocity grid, with periodic boundary conditions. We use a time step of 2 ps and we ran the simulation up to 290 ps.

To verify the correctness of the model, we compare with a direct (brute-force) coupling of BGK and MD, which in principle requires 64 MD calculations at each time step. However, as the simulation proceeds, some of the MD requests are exact duplicates of those which already exist in the database, which somewhat reduces the number of fine-scale requests.

The results of this problem are shown in Fig. 3. We observe that the active learner correctly predicted the MD results for all requests in this test case, as our training dataset provided good coverage of the dynamically evolving simulation data. The model error estimated with root-mean-squared error in the case 10^{-5}. Figure 4 illustrates the distribution of requests executed to obtained microscopic information needed by the macroscale code. We observed that the number of MD tasks required represented less than 1% of the workload required in the verification run.

B. Test problem 2 (gradual heating)

Next, we consider the same initial condition as in Test Problem 1 and linearly increase the background electron temperature from 100 to 600 eV over the course of the simulation. This temperature ramp ensures that the simulation dynamically travels through a much larger portion of the n_i, T

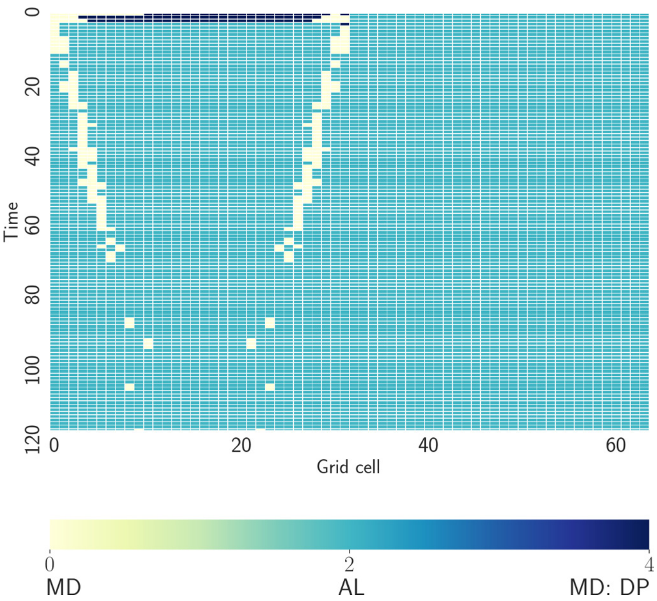

FIG. 4. Plot of the provenance of the fine-scale data delivered to the coarse-scale model for the first problem, shown for each time step and every grid cell. Almost all of the requests are handled by active learning surrogate model, while a small number of more expensive MD calls are made in the dynamically evolving mixing region. The GLUE code also detects a few requests that are exact matches to calls stored in the database. These are marked as duplicates (DP).

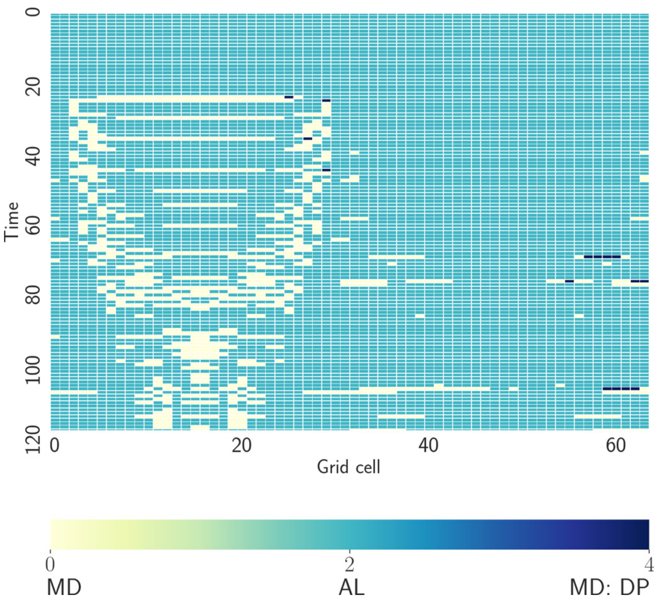

FIG. 5. Plot of the provenance of the fine-scale data delivered to the coarse-scale model for the second problem (temperature ramp), shown for each time step and every grid cell. Most of the requests are handled by active learning surrogate model, but more MD calls are required due to the changing temperatures.

parameter space, which is likely to touch on regions with higher uncertainty. Furthermore, the conditions at late time in the simulation will leave the initial training set entirely, which will challenge the framework to dynamically find a new model as it pushes into conditions that it has not seen before. This pushes the active learning framework finds extra region of high uncertainty and consequently to request more MD points as shown in Fig. 5. Not surprisingly, most of the MD calls are focused on the initial deuterium region rather than the argon region, as more mixing is occurring there.

Our multiscale scheme enables the use of MD-accurate transport coefficients valid across coupling regimes to inform hydrodynamics and kinetic codes. Transport coefficients are generally computed using the Chapman-Enskog (CE) [61] solution of the Boltzmann or Fokker-Planck equation through an expansion of the distribution function. The key terms that arise in this expansion are the so-called collision integrals, which capture key properties of the particle interactions in the hydrodynamic limit and are the inputs into the formulas for the transport coefficients [9]. For weakly coupled plasmas, the cross sections have been evaluated using a screened-Coulomb potential and are available in the form of tables [55] and analytical fits [62] for low-dimensional problems with few species.

A recent theory, the effective potential theory [63], which models many-body correlation effects by treating binary interactions as arising through the potential of mean force, rather than a screened Coulomb potential, provides transport coefficients that cover the weakly and moderately coupled regimes. The evolution of the coupling parameter defined as $\Gamma = \langle Z\rangle^2/aT$, where a is the Wigner-Seitz radius, T is the average temperature, and $\langle Z\rangle$ is the mean average ionization [28], is shown in Fig. 6. As the simulations continue, transport coefficients can be dynamically evolved to capture the changing coupling regimes which are present in Marble-like experiments [21–23] during preheat.

Finally, Table I shows a comparison of total computation time for each of the two test cases. Since our workload is embarrassingly parallel during a given batch, we give the cost of each case in terms of how many MD calls are required. The training and retraining cost of AL (a few seconds) is negligible with respect to an MD run (7 min). The total number of requests for each of the test problem is 8640.

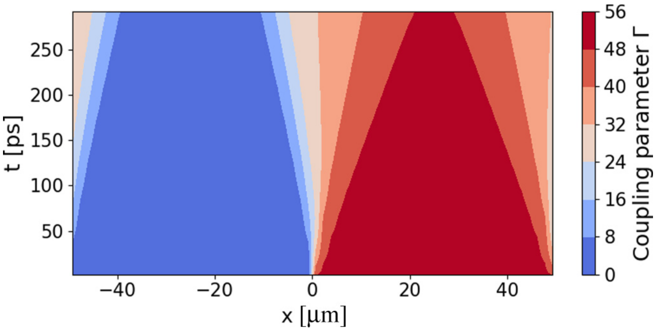

FIG. 6. This figure illustrates the evolution of the coupling parameter of the plasma in time and space for the second problem. We estimated the mean ionization state using a Thomas-Fermi model. Initially, the region containing the deuterium is weakly coupled while the argon region is strongly coupled due to its high charge state (around ∼12 here). As the interface evolves, the mixed argon-deuterium mixing region is in the moderately to strongly coupled regime.

TABLE I. We present the number of MD simulations needed for each of the test cases. Simulations were performed on the Darwin cluster at Los Alamos National Laboratory on a 36 core Skylake microarchitecture. For the training purposes, we used a table of 1100 points. Ninety percent of the dataset was used to train the network while withholding 10% for testing.

	MD calls of Hero run	MD calls during offline training	MD calls during AL run
Case 1	6786 (∼113 h)	994	218 (∼20 h)
Case 2	7690 (∼128 h)	994	858 (∼31 h)

We show both the MD requests for the hero (MD-KT) and the upscaling run which uses AL. These results showed that this approach can produce high-quality results along with considerable computational gains.

IV. CONCLUSIONS AND PERSPECTIVES

We have presented a multiscale framework to bridge atomic to continuum scales. Our framework uses deep neural networks to produce valid models of MD data on the fly and communicates this information to our fine-scale model. The performance of this method has been examined for two test problems showing that the number of MD calls required can be significantly reduced even for highly transient system with large variations in local properties. Furthermore, we have shown through this multiscale scheme that one can dynamically evolve the transport coefficients to cover the physical regimes (i.e., weakly and strongly coupled) visited during the early stages of high-energy-density experiments.

Our future work will apply this framework to the four-species multidimensional MARBLE pore collapse problem using both the kinetic formulation and a direct numerical simulation (DNS) code [64]. Molecular dynamics will be used for the generation of mutual diffusion as in this present work along with the additional transport coefficients (viscosity, thermal conductivity, thermal diffusion) needed by the DNS code.

We also plan to explore different machine learning workflows. In particular, while the query-by-committee method is simple and effective, it is one of many available strategies for uncertainty quantification. Future work may focus on evaluating possible substitutes, such as dropout-based uncertainty [65] or heteroskedastic loss functions [66]. Besides this, a useful change to the ML workflow would be to allow running MD simulations of subthreshold uncertainty in a priority queue, with priority given by the uncertainty of the data—this would not delay the coarse-scale simulation but would better take advantage of any available computational resources to deliver maximal information to the ML-based surrogate model. In addition, parallel methods for active learning might improve the workflow. By considering the full covariance of the ensemble, rather than just the ensemble variance, it may be possible to select points for MD in a more optimal way. By estimating the relationship between different candidate MD points using the model covariance, the active learning procedure could attempt to determine if all fine-grained points on a given time step that fail the uncertainty check need to be run in MD, or if they are correlated with each other, in which case a only a maximal covariance subset of points could be dispatched.

Finally, job loads on clusters tend to vary heavily. Under minimal load, spawning jobs as needed is a very effective way to only use as many resources as are required by the simulation at a given time. Under heavy load this results in long stalls and failed runs as the spawned jobs spend the majority of their time in the job queue. One such way to resolve this is to take advantage of resource pools. By utilizing tools like Lawrence Livermore National Laboratory's Flux [60], we can launch fine-grain simulations in a controlled job pool and increase or decrease the size of the resource pool based on the current needs of the overall simulation. This allows us to take advantage of patterns commonly associated with task based runtimes [58] while continuing to avoid invasive changes to scientific codes.

ACKNOWLEDGMENTS

Research presented in this article was supported by the Laboratory Directed Research and Development program of Los Alamos National Laboratory (LANL) under project number 20190005DR and used resources provided by the LANL Institutional Computing Program. LANL is operated by Triad National Security, LLC, for the National Nuclear Security Administration of U.S. Department of Energy (Contract No. 89233218CNA000001). This document is LA-UR-20-23628.

[1] L. Collins, I. Kwon, J. Kress, N. Troullier, and D. Lynch, Phys. Rev. E **52**, 6202 (1995).

[2] S. Root, K. R. Cochrane, J. H. Carpenter, and T. R. Mattsson, Phys. Rev. B **87**, 224102 (2013).

[3] L. A. Collins, S. R. Bickham, J. D. Kress, S. Mazevet, T. J. Lenosky, N. J. Troullier, and W. Windl, Phys. Rev. B **63**, 184110 (2001).

[4] D. C. Wilson, C. W. Cranfill, C. Christensen, R. A. Forster, R. R. Peterson, N. M. Hoffman, G. D. Pollak, C. K. Li, F. H. Séguin, J. A. Frenje, R. D. Petrasso, P. W. McKenty, F. J. Marshall, V. Y. Glebov, C. Stoeckl, G. J. Schmid, N. Izumi, and P. Amendt, Phys. Plasmas **11**, 2723 (2004).

[5] C. Fryer, A. Diaw, C. Fontes, A. Hungerford, J. Kline, H. Johns, N. Lanier, S. Wood, and T. Urbatsch, High Energy Density Phys. **35**, 100738 (2020).

[6] A. Le, T. J. T. Kwan, M. J. Schmitt, H. W. Herrmann, and S. H. Batha, Phys. Plasmas **23**, 102705 (2016).

[7] O. Larroche, H. G. Rinderknecht, M. J. Rosenberg, N. M. Hoffman, S. Atzeni, R. D. Petrasso, P. A. Amendt, and F. H. Séguin, Phys. Plasmas **23**, 012701 (2016).

[8] W. T. Taitano, A. N. Simakov, L. Chacón, and B. Keenan, Phys. Plasmas **25**, 056310 (2018).

[9] R. L. Liboff, Phys. Fluids **2**, 40 (1959).

[10] M. J. Rosenberg, F. H. Séguin, P. A. Amendt, S. Atzeni, H. G. Rinderknecht, N. M. Hoffman, A. B. Zylstra, C. K. Li, H. Sio, M. Gatu Johnson, J. A. Frenje, R. D. Petrasso, V. Y. Glebov, C. Stoeckl, W. Seka, F. J. Marshall, J. A. Delettrez, T. C. Sangster, R. Betti, S. C. Wilks, J. Pino, G. Kagan, K. Molvig, and A. Nikroo, Phys. Plasmas **22**, 062702 (2015).

[11] H. G. Rinderknecht, H. Sio, C. K. Li, A. B. Zylstra, M. J. Rosenberg, P. Amendt, J. Delettrez, C. Bellei, J. A. Frenje, M. Gatu Johnson, F. H. Séguin, R. D. Petrasso, R. Betti, V. Y. Glebov, D. D. Meyerhofer, T. C. Sangster, C. Stoeckl, O. Landen, V. A. Smalyuk, S. Wilks, A. Greenwood, and A. Nikroo, Phys. Rev. Lett. **112**, 135001 (2014).

[12] J. S. Ross, D. P. Higginson, D. Ryutov, F. Fiuza, R. Hatarik, C. M. Huntington, D. H. Kalantar, A. Link, B. B. Pollock, B. A. Remington, H. G. Rinderknecht, G. F. Swadling, D. P. Turnbull, S. Weber, S. Wilks, D. H. Froula, M. J. Rosenberg, T. Morita, Y. Sakawa, H. Takabe, R. P. Drake, C. Kuranz, G. Gregori, J. Meinecke, M. C. Levy, M. Koenig, A. Spitkovsky, R. D. Petrasso, C. K. Li, H. Sio, B. Lahmann, A. B. Zylstra, and H.-S. Park, Phys. Rev. Lett. **118**, 185003 (2017).

[13] M. M. Marinak, S. W. Haan, T. R. Dittrich, R. E. Tipton, and G. B. Zimmerman, Phys. Plasmas **5**, 1125 (1998).

[14] J. N. Glosli, D. F. Richards, K. J. Caspersen, R. E. Rudd, J. A. Gunnels, and F. H. Streitz, in *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing (SC'07)* (Association for Computing Machinery, New York, NY, USA, 2007), pp. 1–11.

[15] Weinan E, B. Engquist, and Z. Huang, Phys. Rev. B **67**, 092101 (2003).

[16] D. Roehm, R. S. Pavel, K. Barros, B. Rouet-Leduc, A. L. McPherson, T. C. Germann, and C. Junghans, Comput. Phys. Commun. **192**, 138 (2015).

[17] T. Dornheim, J. Vorberger, S. Groth, N. Hoffmann, Z. A. Moldabekov, and M. Bonitz, J. Chem. Phys. **151**, 194104 (2019).

[18] D. Cohn, L. Atlas, and R. Ladner, Mach. Learn. **15**, 201 (1994).

[19] B. Settles, Active Learning Literature Survey, Technical Report, Department of Computer Sciences, University of Wisconsin—Madison (2009).

[20] H. S. Seung, M. Opper, and H. Sompolinsky, in *Proceedings of the 5th Annual Workshop on Computational Learning Theory(COLT'92)* (Association for Computing Machinery, New York, 1992), pp. 287–294.

[21] T. J. Murphy, M. R. Douglas, J. R. Fincke, R. E. Olson, J. A. Cobble, B. M. Haines, C. E. Hamilton, M. N. Lee, J. A. Oertel, N. A. G. Parra-Vasquez, R. B. Randolph, D. W. Schmidt, R. C. Shah, J. M. Smidt, and I. L. Tregillis, J. Phys.: Conf. Ser. **717**, 012072 (2016).

[22] C. E. Hamilton, M. N. Lee, and A. N. G. Parra-Vasquez, Fusion Sci. Technol. **70**, 226 (2016).

[23] R. B. Randolph, J. A. Oertel, D. W. Schmidt, M. N. Lee, B. M. Patterson, K. C. Henderson, and C. E. Hamilton, Fus. Sci. Technol. **70**, 230 (2016).

[24] L. Yin, B. J. Albright, E. L. Vold, W. D. Nystrom, R. F. Bird, and K. J. Bowers, Phys. Plasmas **26**, 062302 (2019).

[25] P. L. Bhatnagar, E. P. Gross, and M. Krook, Phys. Rev. **94**, 511 (1954).

[26] J. R. Haack, C. D. Hauck, and M. S. Murillo, Phys. Rev. E **96**, 063310 (2017).

[27] S. Plimpton, J. Comput. Phys. **117**, 1 (1995).

[28] R. More, in *Pressure Ionization, Resonances, and the Continuity of Bound and Free States*, edited by D. R. Bates and B. Bederson, Advances in Atomic and Molecular Physics, Vol. 21 (Academic Press, New York, 1985), pp. 305–356.

[29] D. O. Gericke, J. Vorberger, K. Wünsch, and G. Gregori, Phys. Rev. E **81**, 065401(R) (2010).

[30] T. Haxhimali, R. E. Rudd, W. H. Cabot, and F. R. Graziani, Phys. Rev. E **90**, 023104 (2014).

[31] J. P. Hansen and I. R. McDonald, Phys. Rev. A **23**, 2041 (1981).

[32] F. Perrot and M. W. C. Dharma-wardana, Phys. Rev. B **62**, 16536 (2000).

[33] C. S. Jones and M. S. Murillo, High Energy Density Phys. **3**, 379 (2007).

[34] S. Dutta and J. Dufty, Europhys. Lett. **102**, 67005 (2013).

[35] G. Kelbg, Ann. Phys. **467**, 219 (1963).

[36] D. M. Ceperley and B. J. Alder, Phys. Rev. Lett. **45**, 566 (1980).

[37] E. W. Brown, B. K. Clark, J. L. DuBois, and D. M. Ceperley, Phys. Rev. Lett. **110**, 146405 (2013).

[38] C. E. Starrett and D. Saumon, Phys. Rev. E **87**, 013104 (2013).

[39] C. Starrett and D. Saumon, High Energy Dens. Phys. **10**, 35 (2014).

[40] M. Green, J. Chem. Phys. **22**, 398 (1954), cited By 811.

[41] R. Kubo, J. Phys. Soc. Jpn. **12**, 570 (1957).

[42] L. P. Kadanoff and P. C. Martin, Ann. Phys. **24**, 419 (1963).

[43] A. J. White, C. Ticknor, E. R. Meyer, J. D. Kress, and L. A. Collins, Phys. Rev. E **100**, 033213 (2019).

[44] J. Daligault, Phys. Rev. Lett. **108**, 225004 (2012).

[45] N. R. Shaffer, S. D. Baalrud, and J. Daligault, Phys. Rev. E **95**, 013206 (2017).

[46] I.-C. Yeh and G. Hummer, J. Phys. Chem. B **108**, 15873 (2004).

[47] B. L. Holian and D. J. Evans, J. Chem. Phys. **78**, 5147 (1983).

[48] A. Grossfield, P. N. Patrone, D. R. Roe, A. J. Schultz, D. Siderius, and D. M. Zuckerman, Living J. Comput. Mol. Sci. **1**, 5067 (2018).

[49] V. Garzó, A. Santos, and J. J. Brey, Phys. Fluids A **1**, 380 (1989).

[50] J. M. Burgers, *low Equations for Composite Gases* (Academic Press, New York, 1969).

[51] J. R. Haack, C. D. Hauck, and M. S. Murillo, J. Stat. Phys. **168**, 826 (2017).

[52] J. P. Hansen, F. Joly, and I. R. McDonald, Physica A **132**, 472 (1985).

[53] D. B. Boercker and E. L. Pollock, Phys. Rev. A **36**, 1779 (1987).

[54] C. Ticknor, J. D. Kress, L. A. Collins, J. Clérouin, P. Arnault, and A. Decoster, Phys. Rev. E **93**, 063208 (2016).

[55] C. Paquette, C. Pelletier, G. Fontaine, and G. Michaud, Astron. J. Suppl. Ser. **61**, 177 (1986).

[56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, in *Advances in Neural Information Processing Systems 32*, edited

by H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché-Buc, E. Fox, and R. Garnett (Curran Associates, Red Hook, NY, 2019), pp. 8024–8035.

[57] D. P. Kingma and J. Ba, in *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*, edited by Y. Bengio and Y. LeCun (2015).

[58] R. Pavel, C. Junghans, S. M. Mniszewski, and T. C. Germann, Using charm++ to support multiscale multiphysics, in *Proceedings of the Programming Models and Co-Design Meeting* (2017).

[59] R. S. Pavel, A. L. McPherson, T. C. Germann, and C. Junghans, in *Proceedings of the 2nd International Workshop on Hardware-Software Co-Design for High Performance Computing (Co-HPC '15)* (Association for Computing Machinery, New York, 2015).

[60] D. H. Ahn, N. Bass, A. Chu, J. Garlick, M. Grondona, S. Herbein, J. Koning, T. Patki, T. R. Scogland, B. Springmeyer *et al.*, in *Proceedings of the 2018 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS'18)* (IEEE, Los Alamitos, CA, 2018), pp. 10–19.

[61] S. Chapman and T. G. Cowling, *The Mathematical Theory of Nonuniform Gases,* 3rd ed. (Cambridge University Press, Cambridge, UK, 1970).

[62] L. G. Stanton and M. S. Murillo, Phys. Rev. E **93**, 043203 (2016).

[63] S. D. Baalrud and J. Daligault, Phys. Rev. Lett. **110**, 235001 (2013).

[64] Z. Li and D. Livescu, Phys. Plasmas **26**, 012109 (2019).

[65] Y. Gal and Z. Ghahramani, in *Proceedings of the International Conference on Machine Learning* (Proceedings of Machine Learning Research, New York, 2016), pp. 1050–1059.

[66] B. Lakshminarayanan, A. Pritzel, and C. Blundell, in *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Red Hook, NY 12571, 2017), pp. 6402–6413.