Efficient Learning of Accurate Surrogates for Simulations of Complex Systems

A. Diaw,[1] M. McKerns,[2] I. Sagert,[2] L. G. Stanton,[3] and M. S. Murillo[4]

[1] *RadiaSoft LLC, Colorado 80301 USA*
[2] *Los Alamos National Laboratory, Los Alamos, New Mexico 87545 USA*
[3] *Department of Mathematics and Statistics, San José State University, San José, California 95192 USA*
[4] *Department of Computational Mathematics, Science and Engineering, Michigan State University, East Lansing, Michigan 48824 USA*

(Dated: February 3, 2022)

CONTENTS

I. INTRODUCTION

In this supplementary report, we apply our methodology to finding accurate surrogates for RDFs from expensive MD simulations. For this type of problem, we will consider three different systems with the increasing complexity of the parameter space: the one-component plasma (OCP) model with a one-parameter space, the Lennard-Jones (LJ) fluid with a two-parameter space, and finally an extension of the OCP, the binary-ionic mixture (BIM) model with a four parameter space.

II. SUPPLEMENTARY NOTE S1: RADIAL DISTRIBUTION FUNCTIONS

Here, we are interested in building accurate surrogates for radial distribution functions (RDF) for systems of neutral and charged liquids, where the data for the RDFs were computed with large-scale molecular dynamics (MD). RDF describes interparticle correlations within isotropic materials by averaging over relevant atomic or molecular coordinates; it is an important fluid characteristic, necessary for analyzing x-ray Thomson scattering experiments[1], investigating protein interactions with cell membranes[2], and examining shock-induced phase transitions[3]. In addition, the RDF plays a key role in perturbative fluids theories, as many aspects of thermodynamic properties in fluids can be expressed in terms of it. The RDF is also useful in investigating some inhomogeneous fluid structures[4], for example in dense plasmas, mixing phenomena[5], and enhancement of nuclear reaction rates[6]. Furthermore, recent developments in kinetic theory[7] and hydrodynamics[8] models for dense plasmas highlight the importance for RDF models. Because of the fundamental role of RDFs in our understanding of physical phenomena in fluids, it has been the focus of numerous studies.

The Lennard-Jones (LJ) model is among the most widely used models for neutral liquids, and numerous fits to its RDF have been given. Early work by Goldman[9] argues for as few parameters as possible for ease of use and to avoid overfitting to noise in the data; fits were made to 87 tables with 108 parameters. Later, Matteoli and Mansoori[10] suggested a considerably simplified form with only 21 parameters and two forms (small and large separation r), allowing for the possibility of extend-

ing the fits to mixtures. Later, Morsali *et al.*[11], reiterate that fits should not have too many parameters, and with improved simulation data, provide a fit with only 11 parameters, again with two functional forms valid at small and large separations.

For strongly coupled plasma studies, the one-component plasma (OCP) is the simplest model used. The OCP is a system of particles with charge Ze interacting through a repulsive Coulomb potential $rV(r) = Z^2e^2$, and a uniform, neutralizing background. Here, Z is the charge number of a given particle, and e is the fundamental charge. The statistical properties of the OCP can be described in terms of a single parameter, the Coulomb coupling parameter $\Gamma = (Z^2e^2)/(aT)$, where $a = (4\pi n/3)^{-1/3}$ is the Wigner-Seitz radius with n being the total ionic number density, and T is the temperature in energy units (note that the charge q is represented here in Gaussian-cgs units). Rogers *et al.*[12] employed Monte Carlo data to generate the OCP structure factor in tabular form, which requires a special request (and cost) for distribution. Brettonet and Derouiche[13] provided a fit to the OCP structure factor in terms of only two parameters, one of which was found to be a constant. However, this fit lacked the accuracy of other approaches. Desbiens *et al.*[14] have parametrized the RDF of the OCP using MD data, where their fit model was motivated by the Matteoli and Mansoori form[10]. To achieve high accuracy, it was necessary to fit weak and strong coupling functional forms separately, with four and nine parameters, respectively. Because there is a Fisher-Widom (FW) line[15–18], which delineates the transition to oscillatory behavior in the RDF, different functional forms are needed. These examples illustrate the challenges with tabulating or fitting data to high accuracy, and because these challenges are for a model system with a single parameter Γ, it is not clear that they generalize to more complex systems such as binary ionic mixtures (BIM), ternary ionic mixtures (TIM) or Lennard-Jones mixtures where we have additional parameter space.

RDFs were computed with MD for three systems to explore both the type (*i.e.*, range and repulsive/attractive) of interaction and the role of dimensionality. In all cases, the MD code LAMMPS[19] was used to produce the data with standard techniques. The equations of motion were first integrated in the canonical ensemble, with constant particle number, volume, and temperature maintained using a Nosé-Hoover thermostat, over 10^5 timesteps, to establish thermodynamic equilibrium at the desired temperature. Then, the production runs were carried out in the micro-canonical ensemble with $t = 1000\,\omega_p^{-1}$, where ω_p^{-1} is a characteristic oscillation period for the system. To help improve the quality of our data, we reduced statistical fluctuations through the use of large particle numbers ($N = 2048$) and long runs. The RDFs $g(r;\mathbf{x})$ were calculated with 1024 bins in the range $0 < r < L/2$, where L is the simulation system size, and $\mathbf{x}$ are the input parameters of system. Note that for binary mixtures, we have three different RDFs $g_{ij}(r;\mathbf{x})$ where i and j are

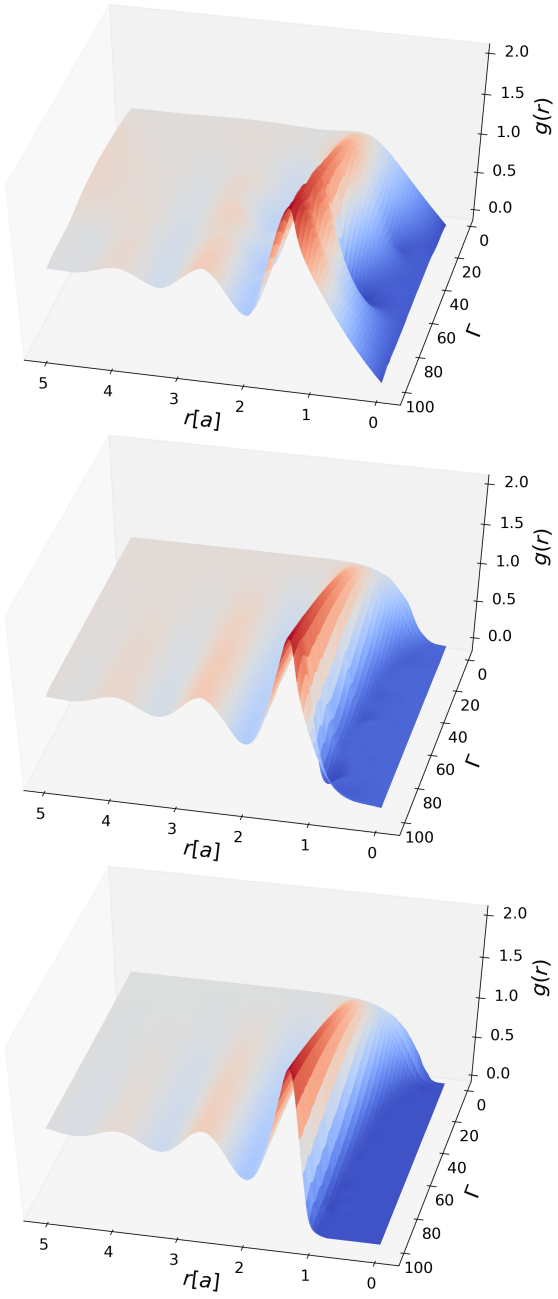

FIG. 1. Illustration of using online learning to refine a surrogate for a radial distribution of OCP obtained from MD using an ensemble of N_s Nelder-Mead optimizers. The optimizers search for critical points, while an estimator learns the surface from the sampled points. Validity of the surrogate is defined as in Eq. (7), with $tol_{max} = 10^{-6}$ and $tol_{sum} = 10^{-3}$, and *train*, *converged*, *data*, and *metric* as defined in III H. The surrogates shown were produced using ensembles of N_s of 4, 8, and 16, respectively. Note that $N_s = 4$ will produce a similar surrogate to $N_s = 16$, either by performing multiple iterations to *test* validity or by setting *warm* (the required number of model evaluations per iteration) to a large enough number that each of the four optimizers is respawned roughly three times.

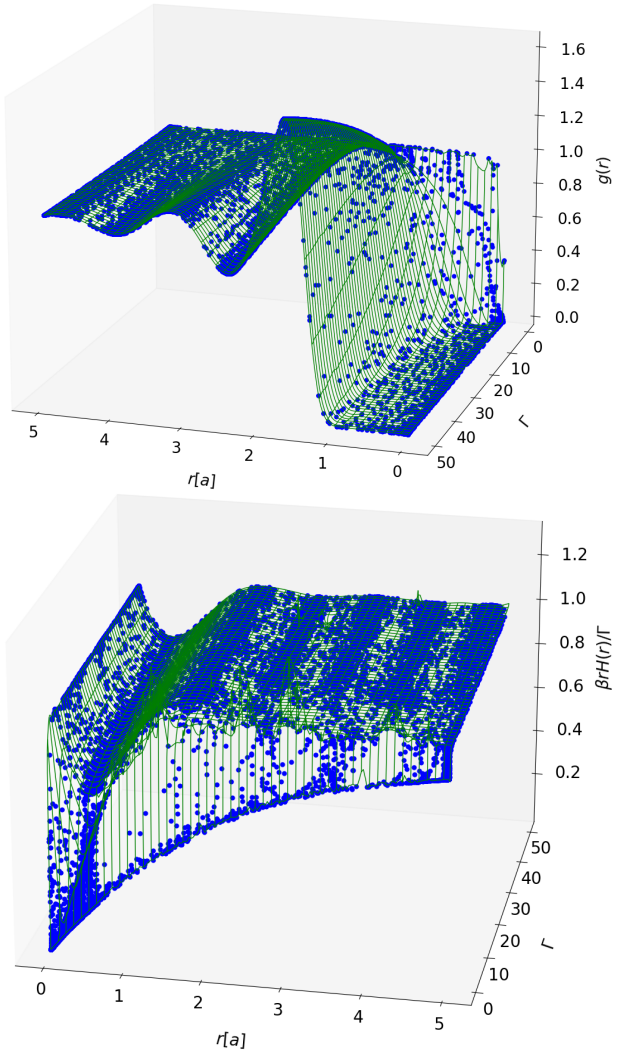

FIG. 2. MD data (blue dots) compared to the surrogate (green surface) for the OCP system in (r, Γ) space, for (a) RDF $g(r)$ (b) screening potential $H(r)$. The initial search domain was defined as $\Gamma \in [0.1, 50]$ and $r[a] \in [0, 5]$, covering both the weakly and strongly coupled regimes. We used a lattice sampler with an ensemble of 40 Nelder-Mead solvers at the default configuration, a definition of *test* validity as in Eq. (7) with $tol_{max} = 10^{-6}$ and $tol_{sum} = 10^{-3}$, and *train*, *converged*, *data*, and *metric* as defined in III H. The resulting surrogate for $g(r)$, $H(r)$, and the associated model evaluations, are stored to disk (see III E) for programmatic access by coarse-grained codes.

the species indexes.

In each of the three cases below, all LAMMPS model evaluations are archived to a model evaluation database, as in III A. We used "lattice" sampling with an ensemble of 40 Nelder-Mead solvers at the default configuration to generate the model evaluations, and interpolation with a thin-plate RBF to generate the surrogate. We defined *test* validity as in Eq. (7) with $tol_{max} = 10^{-6}$ and $tol_{sum} = 10^{-3}$, and *train*, *converged*, *data*, and *metric* as defined in III H. Learned surrogates were saved to a surrogate database as in III B.

a. **One-Component Plasma.** The one-component plasma (OCP) is a model that assumes a system of point ions with charge q at a temperature T embedded in a uniform, neutralizing background. The thermodynamic state of the OCP system is entirely determined by the coupling parameter Γ, so each MD simulation computes the RDF with the form $g(r) = g(r; \Gamma)$. Since the RDF enters into the calculations of many quantities in many forms, we also choose to calculate the related Salpeter screening potential, which is defined in terms of the RDF as:

$$\beta H(r) = \frac{\Gamma}{r} + \log\big[g(r)\big], \tag{1}$$

where $\beta = 1/T$, and r is in units of the Wigner-Seitz radius a. The screening potential $H(r)$ is used to estimate the enhancement factor of nuclear reaction rates in dense plasmas.

For the OCP, the radial coordinate r was defined to be in the range $[0, L/2]$, where L is the simulation box size, and the coupling parameter was taken be in the range of $[0.1, 100]$, which spans both weakly and strongly coupled regimes. Figure 1 illustrates the sampling process of the framework with the surface of the OCP RDF $g(r, \Gamma)$, mapped using an ensemble of local optimizers. The surrogate surface is interpolated on a uniform grid, and as we increase the sampling, the surface resolution improves considerably, and the surrogate captures details very accurately across physical regimes. In Figure 2, we show the predicted RDF and screening potential in (Γ, r) space. The blue dots are directly calculated with MD, while the green lines are calculated with the learned surrogates. We observe excellent agreement between the results produced with the surrogates, and MD, across the defined parameter space.

Next, we examined the predictive accuracy of surrogates learned with the procedure above, with regard to thermodynamic quantities, such as the internal energy and pressure. For an OCP, the internal energy and pressure are given by:

$$\frac{\beta U}{N} = \frac{3}{2}\Gamma \int_0^\infty r\big[g(r) - 1\big]dr, \tag{2}$$

$$\beta\frac{P}{n} = 1 + \frac{\Gamma}{2}\int_0^\infty r\big[g(r) - 1\big]\,dr, \tag{3}$$

where N is the number of particles, and n the density. We trained surrogates for the OCP as above, and repeated the procedure for several different combinations of N_s and tol_{sum}. Note that surrogates were not trained on energy and pressure, but were trained and scored as in Eq. (7). We generated surrogates using all possible combinations of $N_s = \{4, 10\}$ and $tol_{sum} = \{10^{-4}, 10^{-3}\}$. We then used Eqs. (2)-(3) to calculate the pressure and energy using our surrogates, and then compared to similar results produced with MD. We found that the surrogates show an increase in predictive accuracy when $N_s = 10$, as can be seen in Figure 3. Note that strong oscillations

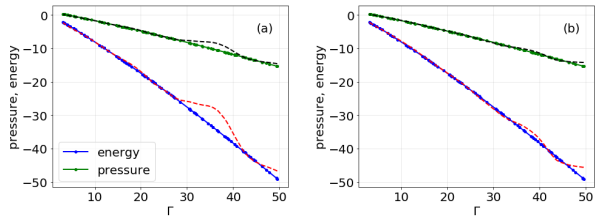

FIG. 3. Energy ($\beta U/N$) and Pressure ($\beta P/\rho$) predicted by a learned surrogate (dashed lines) for an OCP, as compared to results of MD simulations (dotted lines). Surrogates were trained using thin-plate RBF interpolation, with lattice sampling directed by an ensemble of N_s Nelder-Mead optimizers. Note that surrogates were not trained on energy and pressure, but used a definition of *test* validity as in Eq. (7), with *data* and *metric* as defined in III H, and $tol_{max} = 10^{-6}$. We define *train* and *converged* identically to *test*, and (a) $N_s = 4$, $tol_{sum} = 10^{-3}$; (b) $N_s = 4$, $tol_{sum} = 10^{-4}$; When comparing energy and pressure calculated by MD with that produced by the surrogates, the surrogates showed an increase in predictive accuracy when $N_s = 10$, as opposed to 4. This improvement can be attributed to the higher volume of training data near the critical points.

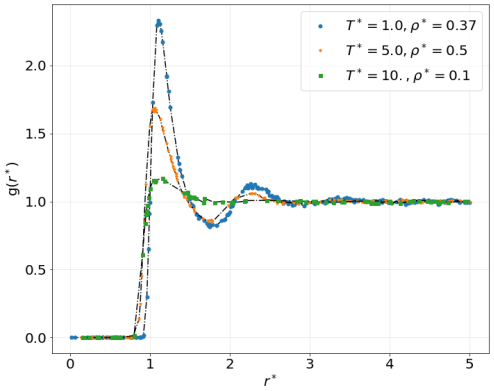

FIG. 4. RDF $g(r)$ for a 2-D Lennard-Jones (LJ) liquid. The markers (squares, etc) indicate data generated from MD. The lines show the surrogate-predicted results, with the surrogates trained. The physical quantities are in LJ units. For the one-fluid LJ system, we observed the same high level of predictive accuracy, with regard to various thermodynamic state points, as we found for the surrogates for OCP and BIM.

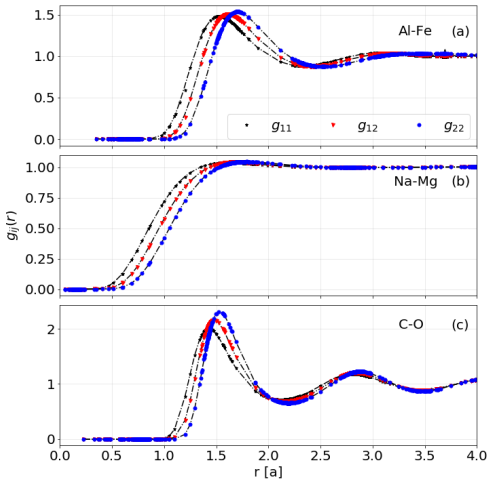

FIG. 5. RDFs $g_{ij}(r)$ of various binary ionic mixtures: Na-Mg, Al-Fe and C-O. The markers (dots) correspond to data generated by MD. The lines show the surrogate-predicted results. In each case: (a) $\Gamma = 0.1, c = 0.8, Z_1 = 11, Z_2 = 12$; (b) $\Gamma = 1, c = 0.6, Z_1 = 13, Z_2 = 2$; (c) $\Gamma = 2.5, c = 0.5, Z_1 = 6, Z_2 = 8$, the predicted results accurately reproduce the results from MD.

are observed in the surrogate's predictions of both the pressure and energy when $N_s = 4$, and that the number of peaks and their amplitude are more substantial at high values of Γ. Indeed, with $N_s = 4$, the surrogates struggle for predictive accuracy in the region of high coupling. When we increase to N_s to 10, we have a higher volume of training data near the critical points, and thus surrogate quality improves around the critical points. In this case, the predicted pressure and internal energy agree very well with MD, where the relative deviation from ground truth (MD) is less than 10^{-4}. That this trend holds across the full range of coupling parameter examined.

b. **Lennard-Jones fluid.** We now consider a LJ liquid in 2 spatial dimensions, where the system is composed of N particles of mass m randomly distributed in the simulation box and interacting through the LJ potential given by:

$$V(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right], \tag{4}$$

with depth of the potential, ϵ, and σ the relative distance at which the interaction between pairs of two particles is zero. The LJ potential is known to give a reasonable description of interactions of atoms in rare-gas systems, which is the limit to which the attractive component is derived. For the LJ potential, we used the reduced units given by: $T^* = T/\epsilon$, $r^* = r/\sigma$ and $\rho^* = \rho\sigma^3$, where ρ^* is the dimensionless density. The resulting RDF for the LJ system then has the form: $g(r^*; T^*, \rho^*)$.

We trained surrogates for the LJ system on $g(r^*; T^*, \rho^*)$, as opposed to $g(r; \Gamma)$ in the previous case for the OCP. We compared our results to calculations performed with MD for different values of T^* and ρ^*. Figure 4 compares the LJ RDF generated with MD with that of the surrogate, for three different values of temperature and density. As can be seen from the figure, the surrogates very accurately predict the results from MD across the entire range, and for different T^* and ρ^*.

c. **Binary Ionic Mixture.** Finally, we consider a BIM model, which is an extension of the OCP allowing for two separate species embedded in a neutralizing electronic background. The ion-ion interaction potentials are modeled with the Coulomb potential

$$V_{ij}(r) = \frac{Z_i Z_j e^2}{r}, \tag{5}$$

where the Z_i is the charge number of the i^{th} species. As with the OCP simulations, the long-range Coulomb forces are handled through an Ewald summation algorithm. For the binary mixture, we introduce the more general coupling parameter

$$\Gamma_{ij} = \frac{Z_i Z_j e^2}{aT}. \tag{6}$$

Letting $\Gamma = e^2/(aT)$ be the proton-proton coupling parameter, the RDFs of the BIM model will subsequently have the form: $g_{ij}(r; \Gamma, c, Z_i, Z_j)$, where $c = n_1/n$ is the concentration of one of the species.

Figure 5 plots the RDFs for three different mixtures of chemical elements present in white dwarfs[20–22], Al-Fe, Na-Mg, C-O, as produced with MD. We also generated surrogates and compared the results to the MD data. We observed excellent agreement between the surrogates and MD, despite the higher dimensions of the BIM RDF parameter space.

III. SUPPLEMENTARY NOTES S2: *MYSTIC* IMPLEMENTATION

We present here some of the components of the method and implementation to *mystic*.

A. Cached model evaluations

Given an expensive `model`, we augment the model by linking the model to a database of model evaluations:

```
model = mystic.cache.cached('model')(model)
```

where the resulting database is named `'model'`, and the model is now augmented by with programmatic access to the database:

```
cache = model.__cache__()
```

and an inverted model

```
inverted_model = model.__inverse__
```

which also hooks into the database, and can be used in solving for maxima. Alternately, we can augment the model with:

```
model = WrapModel('model', model, cached=True, rnd=False)
```

which additionally augments the model with sampling and distance methods, as detailed in III B and III D, respectively. We can also access the database of model evaluations directly from disk

```
cache = mystic.cache.archive.read('model')
```

and, regardless of how the cache was accessed, the data can be loaded into a `mystic.math.legacy.dataset` with:

```
data = as_dataset(cache)
```

or a `mystic.monitor` with:

```
monitor = as_monitor(cache)
```

B. Sampling model evaluations

The samplers available in `mystic.samplers` provide an interface to optimizer-directed and more traditional sampling methods. Here we combine a `SparsitySampler` with a `NelderMeadSimplexSolver` to produce an ensemble of Nelder-Mead optimizers that start where the database is the most sparse, and run to termination:

```
kwds = {solver:NelderMeadSimplexSolver, dist:None}
sampler = SparsitySampler(bounds, model, pts, **kwds)
sampler.sample_until(terminated=all)
```

where, `bounds` indicates the lower and upper bounds for each input parameter, `dist` is a distribution object or similar random number generator that provides additional noise in the sampling, and `pts` is the integer number of optimizers to use *in each direction*. Hence, `pts = 4` will spawn 4 optimizers to search for minima, and 4 optimizers to search for maxima. Here, `model` is an expensive model as generated in III A. For `solver`, we are using a Nelder-Mead solver at the default configuration (see III F), as opposed to passing a configured solver instance. A sampler also provides traditional (not optimizer-directed) sampling, with:

```
sampler.sample()
```

so, a `SparsitySampler` with `pts = 4` will sample 8 data points in total, located at the 8 most sparse coordinates in the database. Alternately, if a `WrapModel` is used,

```
model.sample(bounds, -pts, sampler=SparsitySampler, **kwds)
```

then setting `pts = 4` will spawn 4 minima-seeking and 4 maxima-seeking optimizers, while

```
model.sample(bounds, pts, sampler=SparsitySampler, **kwds)
```

will use traditional sampling to sample 8 total points.

C. Learning a surrogate

A surrogate is generated with a learning strategy, based on an estimation method such as interpolation or machine learning. We learn a surrogate with interpolation by using `InterpModel`, essentially a `WrapModel` which uses `mystic.math.interpolate.interpf` to produce an interpolated surrogate. For example:

```
kwds = {method:'thin_plate', noise:1e-8, smooth:0}
surrogate = InterpModel('surrogate', model, **kwds)
```

generates a `surrogate` for `model` that attempts to best satisfy using a 'thin-plate' RBF. Here, `smooth:0` forces the interpolated function to go through the nodal points, and `noise:1e-8` adds Gaussian noise with an amplitude of `1e-8` to remove duplicate inputs (and thus avoid a singular matrix. Thus once `surrogate.fit()` is called, or the surrogate is evaluated, `surrogate(x)` closely approximates `model(x)` for all data in the `'model'` database. Whenever `model` is evaluated, and thus

more data is added to the model database, we can call `surrogate.fit()` to trigger an update to the interpolated surrogate. We can save the surrogate to disk for later use with:

```
mystic.cache.function.write(surrogate, 'surrogate.db')
```

Alternately, we can use machine learning to generate a surrogate. For example, we use a `MLPRegressor` and a `StandardScaler` from *scikit-learn* and `MLData`, `Estimator`, and `LearnedModel` from *mystic* generate a surrogate from a neural network (NN). We generate a training set that includes all the data, as we will be testing on yet-to-be-acquired data:

```
x, y = data.coords, data.values
mld = MLData(x, x, y, y).
```

We next build an estimator, and as training a NN is non-deterministic, we use `improve_score` from *mystic* to iteratively improve the best-fit estimator:

```
args = {hidden_layer_sizes:(100,75,50), max_iter:1000,
        n_iter_no_change:5, solver:'lbfgs'}
est = Estimator(MLPRegressor(**args), StandardScaler())
est = improve_score(est, mld, tries=10, verbose=True)
```

Here, `args` are configuration hyperparameters for the `MLPRegressor`. `StandardScaler` will use the default configuration. The estimator `est` is considered best-fit to the data when `tries=10` successive iterations fail to yield improvement. Similarly to how an `InterpModel` uses interpolation to learn a surrogate for an expesive model, a `LearnedModel` uses the above-configured estimator to produce a surrogate from a NN

```
kwds = {estimator:est.estimator, transform:est.transform}
surrogate = LearnedModel('surrogate', model, **kwds)
```

D. Validity metrics

Surrogate validity is defined in terms of a metric, which measures the distance between the surrogate and the model (or model evaluations). One of the most common metrics is the square of the pointwise difference between the outputs:

```
dist = [(surrogate(x) - model(x))**2 for x in data.coords]
```

and this is indeed the default metric in an `ErrorModel`

```
error = ErrorModel('error', model, surrogate, metric=None)
dist = [error(x) for x in data.coords]
```

where, alternate metrics can be provided of the form `metric(y, y')`. A less common but more robust metric is the graphical distance, which is essentially the *shortest normal line* from the model evaluated at `x` to the surrogate at the point of tangency. The graphical distance is available at `mystic.math.distance.graphical_distance`

```
dist = graphical_distance(surrogate, data, hausdorff=True)
```

or directly from a surrogate built as in III C

```
dist = surrogate.distance(data)
```

where `hausdorff` indicates whether or not we include Δ_x. Once a metric has been defined, we can find the surrogate that is optimally valid, in terms of the surrogate hyperparameters, by using an optimizer to minimize `dist`. Alternately, we can define a validity threshold, such as

```
valid = max(dist) <= 1e-4 and mean(dist) <= 1e-5
```

that validates the surrogate when `valid = True`.

E. Using stored surrogates

Learned surrogates are stored to disk (as in III C), and can be accessed programmatically. When the surrogates are evaluated for given inputs, they provide a reasonable approximation for results generated by the more expensive model. For example, we load the surrogate for the binary mixture from disk with:

```
g = mystic.cache.function.read("BIM.db")
```

and then evaluate the surrogate:

```
g11, g12, g22 = g(r, Gamma, c, Z1, Z2)
```

where `r` is the radial coordinate in Wigner-Seitz radius, `Gamma` is the proton-proton coupling parameter, `c` is the concentration of species 1 with nuclear charge `Z1`, and `Z2` is the nuclear charge of species 2.

F. Default optimizer configuration

The defaults for the `NelderMeadSimplexSolver` are:

```
defaults = {maxiter: None, maxfun: None, xtol: 1e-4,
            ftol: 1e-4, radius: 0.05, adaptive: False}
```

where `maxiter` is the maximum number of iterations, `maxfun` is the maximum number of function evaluations, `xtol` is the acceptable absolute change in each parameter for convergence, `ftol` is the acceptable absolute change in the objective for convergence, `radius` is the percent change for initial simplex values, and `adaptive` is true if adaptive parameters should be used. The Nelder-Mead optimizer uses a candidate relative tolerance (CRT) termination, specifically `CRT(xtol, ftol)`[23].

The defaults for the `PowellDirectionalSolver` are:

```
defaults = {maxiter: None, maxfun: None, ftol: 1e-4,
            gtol: 2, imax: 500, xtol: 1e-4, direc: None}
```

where `maxiter`, `maxfun`, and `ftol` are defined as above, `gtol` is the maximum iterations to run without improvement, `imax` is the line-search maximum iterations, `xtol` is the line-search error tolerance, and `direc` is the the initial direction set (which, if not provided, will use the identity matrix). Powell's optimizer uses a normalized

change over generation (NCG) termination, specifically `NCG(ftol, gtol)`[23].

Thus, the Nelder-Mead solver will stop when the absolute difference in both $\mathbf{x}$ and $f(\mathbf{x})$ over one iteration is less than 10^{-4}, while Powell's solver will stop when the normalized absolute difference of $f(\mathbf{x})$ over `gtol` iterations is less than 10^{-4}.

G. Hartmann's function

The coefficients of the 6-D Hartmann's function are given by

$$\alpha = \begin{vmatrix} 1.0 \\ 1.2 \\ 3.0 \\ 3.2 \end{vmatrix}, A = \begin{vmatrix} 10 & 3 & 17 & 3.50 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{vmatrix}$$

and

$$P = 10^{-4} \begin{vmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{vmatrix}$$

H. Sampling for Training Validity

We assess the impact on the efficiency of optimizer-directed sampling due to the configuration of the optimizer. Our optimizer-directed sampler uses a "lattice" sampling strategy with an ensemble of 40 `NelderMeadSimplexSolver` instances. We define our *test* for validity as:

$$\sum_y \Delta_y \leq tol_{sum} \ \wedge \ \max(\Delta_y) \leq tol_{max}, \tag{7}$$

where $tol_{sum} = 1 \cdot 10^{-3}$ and $tol_{max} = 1 \cdot 10^{-6}$. We use a graphical distance with $\Delta_x \neq 0$, and *data* defined as all existing model evaluations (i.e. prior plus newly sampled). We define *train* as in Eq. (7), again with $tol_{sum} = 1 \cdot 10^{-3}$ and $tol_{max} = 1 \cdot 10^{-6}$. We use a quality *metric* for training, defined by $\delta = \sum_y \Delta_y$, and define *converged* identically to *test* in Eq. (7).

[1] S. H. Glenzer and R. Redmer, X-ray Thomson scattering in high energy density plasmas, Reviews of Modern Physics **81**, 1625 (2009).

[2] F. Di Natale, H. Bhatia, T. S. Carpenter, C. Neale, S. Kokkila-Schumacher, T. Oppelstrup, L. Stanton, X. Zhang, S. Sundram, T. R. W. Scogland, G. Dharuman, M. P. Surh, Y. Yang, C. Misale, L. Schneidenbach, C. Costa, C. Kim, B. D'Amora, S. Gnanakaran, D. V. Nissley, F. Streitz, F. C. Lightstone, P.-T. Bremer, J. N. Glosli, and H. I. Ingólfsson, A massively parallel infrastructure for adaptive multiscale simulations: Modeling ras initiation pathway for cancer, in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '19 (Association for Computing Machinery, New York, NY, USA, 2019).

[3] K. Kadau, T. C. Germann, P. S. Lomdahl, and B. L. Holian, Microscopic View of Structural Phase Transitions Induced by Shock Waves, Science **296**, 1681 (2002).

[4] R. Evans, The nature of the liquid-vapour interface and other topics in the statistical mechanics of non-uniform, classical fluids, Advances in Physics **28**, 143 (1979).

[5] L. G. Stanton, J. N. Glosli, and M. S. Murillo, Multiscale molecular dynamics model for heterogeneous charged systems, Phys. Rev. X **8**, 021044 (2018).

[6] S. Ichimaru, Nuclear fusion in dense plasmas, Rev. Mod. Phys. **65**, 255 (1993).

[7] S. D. Baalrud and J. Daligault, Mean force kinetic theory: A convergent kinetic theory for weakly and strongly coupled plasmas, Physics of Plasmas **26**, 082106 (2019), arXiv:1904.09208 [physics.plasm-ph].

[8] A. Diaw and M. S. Murillo, Excess pressure and electric fields in nonideal plasma hydrodynamics, Phys. Rev. E **99**, 063207 (2019).

[9] S. Goldman, An explicit equation for the radial distribution function of a dense lennard-jones fluid, The Journal of Physical Chemistry **83**, 3033 (1979), https://doi.org/10.1021/j100486a020.

[10] E. Matteoli and G. A. Mansoori, A simple expression for radial distribution functions of pure fluids and mixtures, J. Chem. Phys. **103**, 4672 (1995).

[11] A. Morsali, E. K. Goharshadi, G. Ali Mansoori, and M. Abbaspour, An accurate expression for radial distribution function of the Lennard-Jones fluid, Chemical Physics **310**, 11 (2005).

[12] F. J. Rogers, D. A. Young, H. E. Dewitt, and M. Ross, One-component plasma structure factor in tabular form, Phys. Rev. A **28**, 2990 (1983).

[13] J.-L. Bretonnet and A. Derouiche, Analytic form for the one-component plasma structure factor, Phys. Rev. B **38**, 9255 (1988).

[14] N. Desbiens, P. Arnault, and J. Clérouin, Parametrization of pair correlation function and static structure factor of the one component plasma across coupling regimes, Physics of Plasmas **23**, 092120 (2016), arXiv:1606.04675 [physics.plasm-ph].

[15] J. G. Kirkwood, Statistical mechanics of liquid solutions, Chem. Rev. **19**, 275–307 (1936).

[16] M. E. Fisher and B. Wiodm, Decay of Correlations in Linear Systems, J. Chem. Phys. **50**, 3756 (1969).

[17] C. Vega, L. F. Rull, and S. Lago, Location of the fisher-widom line for systems interacting through short-ranged potentials, Phys. Rev. E **51**, 3146 (1995).

[18] P. Hopkins, A. J. Archer, and R. Evans, Asymptotic decay of pair correlations in a yukawa fluid, Phys. Rev. E **71**, 027401 (2005).

[19] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comput. Phys. **117**, 1 (1995).

[20] B. Paxton, J. Schwab, E. B. Bauer, L. Bildsten, S. Blinnikov, P. Duffell, R. Farmer, J. A. Goldberg, P. Marchant, E. Sorokina, A. Thoul, R. H. D. Townsend, and F. X. Timmes, Modules for Experiments in Stellar Astrophysics (MESA): Convective Boundaries, Element Diffusion, and Massive Star Explosions, ApJS **234**, 34 (2018).

[21] A. Diaw and M. S. Murillo, A DYNAMIC DENSITY FUNCTIONAL THEORY APPROACH TO DIFFUSION IN WHITE DWARFS AND NEUTRON STAR ENVELOPES, The Astrophysical Journal **829**, 16 (2016).

[22] E. B. Bauer and L. Bildsten, Polluted White Dwarfs: Mixing Regions and Diffusion Timescales, Astrophys. J. **872**, 96 (2019).

[23] M. McKerns, P. Hung, and M. Aivazis, mystic: highly-constrained non-convex optimization and UQ (2009), http://pypi.python.org/pypi/mystic.