JCTC Journal of Chemical Theory and Computation

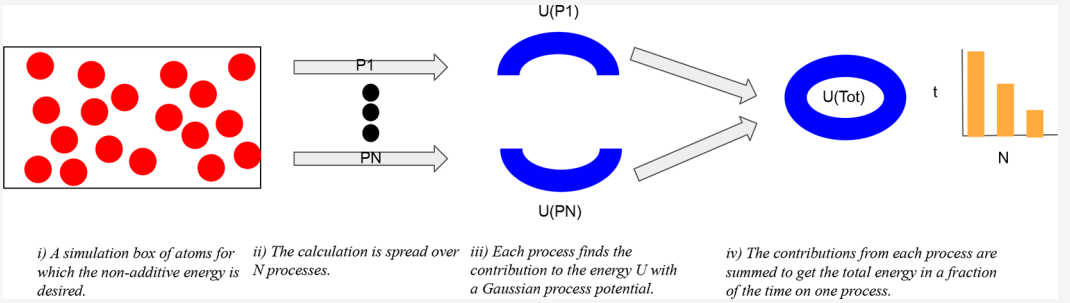
pubs.acs.org/JCTC Article

Parallel Implementation of Nonadditive Gaussian Process Potentials for Monte Carlo Simulations

Jack Broad,* Richard J. Wheatley, and Richard S. Graham

Cite This: https://doi.org/10.1021/acs.jctc.3c00113 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information



ABSTRACT: A strategy is presented to implement Gaussian process potentials in molecular simulations through parallel programming. Attention is focused on the three-body nonadditive energy, though all algorithms extend straightforwardly to the additive energy. The method to distribute pairs and triplets between processes is general to all potentials. Results are presented for a simulation box of argon, including full box and atom displacement calculations, which are relevant to Monte Carlo simulation. Data on speed-up are presented for up to 120 processes across four nodes. A 4-fold speed-up is observed over five processes, extending to 20-fold over 40 processes and 30-fold over 120 processes.

INTRODUCTION

First-principles predictions of macroscopic properties of fluids have the potential to improve models in a wide range of problems, from climate change[1,2] to water desalination.[3] Such predictions require a potential energy surface (PES), which is usually obtained by interpolating *ab initio* calculations of the molecular interactions and a method to convert this PES into macroscopic predictions. Though virial equations of state[4] can predict properties of gases, molecular simulation[5,6] is required for liquid and solid properties. For a simulation to achieve quantitative accuracy, it requires a potential that makes predictions of *ab initio* quality of the underlying microscopic interactions.

There are many strategies to fit potentials to *ab initio* data including parametric fits in which the parameters of the potential are postulated *a priori*. Recent examples include *ab initio* potentials,[7−12] which are so-called as they employ functional forms motivated by first principles. However, a parametric model cannot capture completely the high-dimensional PES. Furthermore, modeling mixtures with *ab initio* potentials is cumbersome, requiring fitting of many potentials with varying numbers of parameters to different data sets. These potentials also often fail to capture accurately three-body nonadditive effects, which are significant in predicting properties of liquids and solids.

Machine-learned potentials (MLPs)[13,14] offer a route to first-principles predictions in simulations through nonparametric interpolation of quantum-mechanical data. Nonparametric interpolation permits high-accuracy approximations of even a three-body PES without experimental data. MLPs therefore have the capacity to facilitate quantitatively accurate simulations, which could reduce the need for experiments when calculating gas, liquid, and coexistence properties.

MLPs employ different methods of prediction, with Gaussian processes[15] (GPs) and neural networks (NNs) being common choices. Recent developments in the training of NN potentials have reduced the number of training points they require to be comparable with that of GP potentials.[16,17] However, to avoid overfitting of a NN potential when the distribution or number of training points varies (such as when fitting to a different PES) may require alterations to the number of hidden layers or neurons per layer.[18] Overfitting is not a concern when training GPs, meaning the exact same algorithm used to train a GP potential on one system can be applied to other systems with no alterations.[19] Consequently, GP potentials are ideal for the simulation of mixtures.

Received: January 27, 2023

GP potentials have been produced for various systems[20−38] and achieve accuracies equivalent to NN potentials.[18,24,39] GP potentials have also been trained via transfer learning,[40,41] in which a training set is selected from a larger reference set of relatively cheap *ab initio* calculations and upgraded to a higher level of theory. Thus, a GP potential from coupled cluster calculations can be produced with a few expensive calculations.

GP potentials are unique in offering high accuracy interpolation with small training sets and flexible training algorithms. However, prediction with GPs is computationally intensive, scaling linearly with the training set size. As this can reach ∼1000 points for nonadditive PES of even small systems, effective parallelization algorithms for GP potentials are important. A general parallelization algorithm is presented here, with effective parallel speed-up demonstrated for nonadditive energy calculations of the type required in Monte Carlo (MC) simulations. This algorithm extends straightforwardly to additive energy calculations and comprises two parts: the precalculation of the covariance function using shared memory and the distribution of triplets over processes. Though the precalculation is GP-specific, the distribution of the triplets is not restricted to any potential. Moreover, it is effective in ensuring that the triplets are disseminated equitably across processes.

Efficient GP potentials would facilitate quantitatively accurate molecular simulations, which have wide-ranging applications. For example, the US government anticipates molecular simulations will be instrumental in carbon capture, utilization, and storage (CCUS).[42] CCUS pipelines contain small molecules such as N_2, O_2, Ar, and H_2 in addition to CO_2,[21] and GP potentials have modeled interactions between various small molecules reliably.[20,22,23] Thus, GP potentials are ideal for simulations to determine, with quantitative accuracy, properties of the mixtures in CCUS pipelines. This would be achieved without experiment and would enable liquid-phase predictions, which are impossible by current approaches.

The following sections discuss a method for efficient implementation of GP potentials, starting with the background theory in section 2. This is followed by an overview of the full box and atom displacement calculations in section 3. Thereafter a summary of the computational details is given in section 4, before the results are presented in section 5.

2. BACKGROUND THEORY

Gaussian Process Potentials. All GPs employ a kernel function when making predictions, which maps the problem being modeled to a feature space defined by the covariance between points. The GP potentials herein use a symmetric squared exponential kernel k_{Sym},[20,22,23]

$$k_{Sym}(\mathbf{x}, \mathbf{x}') = \sum_{p\in P} k_{SE}(\mathbf{x}, p\mathbf{x}'),$$

$$k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \prod_{k=1}^{N_d} \exp\left(-\frac{(x_k - x_k')^2}{2l_k^2}\right) \quad (1)$$

Here, k_{SE} is the squared exponential kernel, $\mathbf{x}$ and $\mathbf{x}'$ are vectors of inputs, N_d is the number of dimensions in these vectors, and P is the group of permutations of $\mathbf{x}'$ under which the output is invariant. The hyperparameters of the kernel shown in eq 1 are the signal variance σ_f^2 and the length scales l_k.

Hyperparameter values are obtained by maximizing the log-likelihood log(L) of the model on the training set, where

$$\mathcal{L} = (2\pi)^{-N_t/2} \left|\mathbf{K} + \mathbf{I}\sigma_n\right|^{-1/2} \exp\left(-\frac{1}{2}\mathbf{y}^T(\mathbf{K} + \mathbf{I}\sigma_n^2)^{-1}\mathbf{y}\right) \quad (2)$$

Here, N_t is the number of training points, $\mathbf{I}$ is the identity matrix, $\mathbf{K}$ is the covariance matrix obtained by evaluating k_{Sym} and $\mathbf{y}$ is a vector of observations. In this work, the observations are nonadditive energies for Ar_3 triplets, obtained at the CCSD(T) level with an aug-cc-pVTZ basis set. σ_n^2 is the Gaussian noise variance, which is an additional hyperparameter that is optimized. The process of developing the training set and obtaining the corresponding hyperparameters is referred to as training. Following training, predictions are made using the Woodbury vector $\boldsymbol{\lambda}$,

$$\boldsymbol{\lambda} = (\mathbf{K} + \mathbf{I}\sigma_n^2)^{-1}\mathbf{y} \quad (3)$$

(The Woodbury vector is denoted here as $\boldsymbol{\lambda}$ despite being typically represented as $\boldsymbol{\alpha}$, because α later denotes an atom in a triplet.)

When modeling PES, inverse interatomic distances are effective inputs for the kernel function.[20] The symmetry-equivalence of different intermolecular configurations is incorporated into the model with the symmetric kernel k_{Sym}, allowing training sets to be developed for the symmetry-distinct region of the phase space only. As in previous work,[20,22,23] all data sets used to train the GP here were built with Latin hypercube sampling.[43−45]

Using a GP that employs k_{Sym}, the nonadditive potential $U_{NA}(\mathbf{x})$ of a molecular triplet is

$$U_{NA}(\mathbf{x}) = \sigma_f^2 \sum_{i=1}^{N_t} \lambda_i \sum_{j=1}^{N_{perm}} \prod_{k=1}^{N_d} \exp\left(-\frac{(x_k - (x_{ik}')_j)^2}{2l_k^2}\right) \quad (4)$$

Here, i sums over training points, j sums over permutations, k runs over dimensions of $\mathbf{x}$, and $(x'_{ik})_j$ is the kth coordinate in the ith training point, after the latter has been subjected to the jth permutation in the group P.

A permutation is an interchange of two or more distances, under which the energy is invariant. For example, in a triplet of identical atoms the total number of permutations in P, N_{perm}, is six. These permutations are stored in a permutation matrix $\mathbf{P}$, which for the three-atom example is

$$\mathbf{P} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \\ 3 & 2 & 1 \end{pmatrix} \quad (5)$$

Each element in $\mathbf{P}$ corresponds to the index of an inverse interatomic distance. Though this matrix could become large for more complex systems, its inclusion reduces the number of expensive *ab initio* calculations. If a system was too complicated to include symmetry in this way, it is possible to ignore certain permutations to reduce the size of $\mathbf{P}$, though the effect of this is not explored herein.

https://doi.org/10.1021/acs.jctc.3c00113
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

3. PARALLEL CALCULATION OF POTENTIAL ENERGY IN SIMULATION

Here, a method is derived to calculate in parallel the total energy of a simulation box and the change therein following movement of an atom using a GP potential. The method exploits the functional form that describes the energy for such a potential, as shown in eq 4. Efficient parallel calculation of the total potential is achieved by distributing the molecular pairs across a number of processes and calculating the exponentials in eq 4 with shared memory. Though the exponential calculations are GP-specific, the strategy to distribute pairs is general to other potentials.

Simulation Box Energy. The total nonadditive energy of an entire simulation box of N_a molecules, U_{NA}^{tot}, is obtained by summing eq 4 over all triplets. The full range of triplets is described generally with the molecular numbers $\alpha\beta\gamma$, where $1 \leq \alpha < \beta < \gamma \leq N_a$. Hence the sum over molecular triplets is

$$\frac{U_{NA}^{tot}}{\sigma_f^2} = \sum_{\alpha<\beta<\gamma}^{N_a} \sum_{i=1}^{N_t} \lambda_i \sum_{j=1}^{N_{perm}} \prod_{k=1}^{N_d} \exp\left(-\frac{[(x_{\alpha\beta\gamma})_k - (x'_{ik})_j]^2}{2l_k^2}\right) \tag{6}$$

where $(x_{\alpha\beta\gamma})_k$ is the kth inverse intermolecular distance of the triplet $\alpha\beta\gamma$. Equation 6 therefore gives the total nonadditive energy when all triplets are calculated directly, *i.e.*, when no cutoff distance is used.

Equation 6 does not assume any specific type for the constituent molecules, but for simplicity, it is assumed henceforth that all particles are identical atoms. Hence, $N_d = 3$ and the length scale l_k is the same for all k. The smallest number of exponentials required to evaluate eq 6 is identified, and their values are stored in an array. The array

$$(E_{ik})_{\alpha\beta} = \exp\left(-\frac{[x_{\alpha\beta} - x'_{ik}]^2}{2l_k^2}\right) \tag{7}$$

where $i = 1...N_t$, $k = 1...N_d$ and α,β runs over all possible pairs, is sufficient. The loop over all training points i for all dimensions k is necessary because, for three identical atoms, all permutations of the dimensions are permitted (eq 5).

The use of a permutation matrix $\mathbf{P}$ allows one to write $(x'_{ik})_j = x'_{iP_{kj}}$ by taking the kth element of the jth row in $\mathbf{P}$ as the index of the relevant distance in the ith training point. Hence

$$\exp\left(-\frac{[x_{\alpha\beta} - (x'_{ik})_j]^2}{2l_k^2}\right) = (E_{iP_{kj}})_{\alpha\beta} \tag{8}$$

A mapping to identify which interatomic distance corresponds to $(x_{\alpha\beta\gamma})_k$, that is, the kth distance in the triplet $\alpha\beta\gamma$, is required. For a triplet of atoms, $\mathbf{x}_{\alpha\beta\gamma} = (x_{\alpha\beta}, x_{\alpha\gamma}, x_{\beta\gamma})$. As N_d is small it is possible to write out the product in eq 6 explicitly and substitute in eq 8 to obtain

$$\frac{U_{NA}^{tot}}{\sigma_f^2} = \sum_{\alpha<\beta<\gamma}^{N_a} \sum_{i=1}^{N_t} \lambda_i \sum_{j=1}^{N_{perm}} (E_{iP_{1j}})_{\alpha\beta} (E_{iP_{2j}})_{\alpha\gamma} (E_{iP_{3j}})_{\beta\gamma} \tag{9}$$

The algorithm in this work implements GP potentials efficiently by precomputing the exponentials in the exponential array and parallelizing the nested sum in eq 9 by distributing the triplets across processes. The algorithm to distribute pairs of atoms across processes for the sum over triplets is applicable to other three-body potentials, not just GP PES.

NONADDITIVE ENERGY CALCULATION FOR A SIMULATION BOX

Calculation of the total nonadditive energy of a simulation box U_{NA}^{tot} is necessary at the outset of a Monte Carlo (MC) simulation and during any move that affects all particles, such as a volume change. To calculate U_{NA}^{tot} necessitates evaluating $\frac{1}{6}(N_a^3 - 3N_a^2 + 2N_a) \approx N_a^3/6$ separate triplets. Though the method extends to mixtures of molecules, the subsequent discussion centers on atomic triplets of the same species such as Ar_3, for which results are presented later. Direct evaluation of eq 4 to calculate the nonadditive energy of an atomic triplet would involve $N_t N_{perm} N_d N_a^3/6$ exponentials, an expensive potential that scales unfavorably with N_a. However, each inverse distance is common to many triplets so it is useful to precalculate exponentials once. Afterward the exponentials are assembled to calculate U_{NA} for any triplet.

Under periodic boundary conditions (PBCs) and a minimum image convention (MIC), $x_{\alpha\beta}$ is the inverse minimum image (MI) distance between atoms α and β. For a simulation box of side length L, a cutoff $r_c = L/2$ is applied, such that if any distance in a triplet exceeds r_c then the nonadditive energy of the triplet is set to zero. The inverse MI distances are stored in array $\mathbf{X}$.

Precalculation of the exponentials is undertaken in parallel over a total of N_p processes, each with its own rank R, $R \in [1, N_p]$. The process with $R = 1$ is known as the root process. All processes may be on a single node or may be split across several nodes. Using the exponential array in parallel requires sharing large amounts of information, as all processes will likely need access to all exponentials. As the connections between nodes are not fast, the exponential array is calculated using shared memory, and each node has its own copy. Letting $\mathbf{E}_{\alpha\beta}$ be the segment of the exponential array pertaining to the $\alpha\beta$ distance, the calculation of the exponentials on a single process proceeds by Algorithm 1. During this calculation, exponentials are calculated between pairs of atoms separated by less than r_c only.

https://doi.org/10.1021/acs.jctc.3c00113
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

Algorithm 1 Pre-calculation of the exponential array on a process on a single node with shared memory, for three identical atoms. $N_{\mathrm{exp}}^{(R)}$ is the number of pairs assigned to a process and N_R is the number of processes on the node.

1: Order all atomic pairs $\alpha\beta$ from smallest to largest (*e.g.* 1,2; 1,3;... 1,N_{a}; 2,3;... 2,N_{a};... $N_{\mathrm{a}}-1$,N_{a}).
2: Determine $N_{\mathrm{exp}}^{(R)}$ for each process on the node such that the pairs are divided evenly.
3: Send the first $N_{\mathrm{exp}}^{(1)}$ pairs in the list from step 1 to the lowest ranked process on the node, send the $N_{\mathrm{exp}}^{(2)}$ to the next process and so on up to process N_R.
4: **for** $m = 1, N_{\mathrm{exp}}^{(R)}$ **do**
5: Take the mth $\alpha\beta$ pair assigned to process R.
6: **if** $r_{\alpha\beta} \leq r_{\mathrm{c}}$ **then**
7: **for** $i = 1, N_{\mathrm{t}}$ **do**
8: **for** $k = 1, N_{\mathrm{d}}$ **do**
9: Evaluate $\exp\left(\frac{\left[x_{\alpha\beta}-x'_{ik}\right]^2}{2l_k^2}\right)$.
10: Save this exponential to $(\mathrm{E}_{ik})_{\alpha\beta}$.
11: **end for**
12: **end for**
13: **else**
14: $\mathbf{E}_{\alpha\beta} = 0$.
15: **end if**
16: **end for**

Consequently, a speed-up of a factor of up to N_{p} is possible if all processes are on a single node. For processes split over several nodes, the speed-up is limited to a factor of N_R on the node with the fewest processes.

The calculation of $U_{\mathrm{NA}}^{\mathrm{tot}}$ is split over N_{p} processes, reducing the number of triplet evaluations on each process by up to a factor of N_{p}. As each process needs to only send back the total nonadditive energy of all assigned triplets (*i.e.*, a single number) the information sharing between processes is not a limiting factor. Consequently shared memory was not required, so there is no need to consider whether processes are on the same node when evaluating the triplet nonadditive energies. The work in this section is general and is not restricted to GPs.

To allocate triplets to processes, all atom pairs in the simulation box are first divided between the processes. The pairs assigned to each process are consistent throughout the calculation. The pairs cannot be assigned in the same manner as steps 1−3 of Algorithm 1 because all pairs containing a given atom must be distributed evenly; otherwise, moving that atom would leave a process to undertake an excessive share of the calculation. An even distribution is achieved by dividing the pairs via Algorithm 2.

Algorithm 2 Distribution of atomic pairs across processes for energy calculations. N_{p} is the total number of processes.

1: Order all atomic pairs $\alpha\beta$ from smallest to largest (*e.g.* 1,2; 1,3;... 1,N_{a}; 2,3;... 2,N_{a};... $N_{\mathrm{a}}-1$,N_{a}).
2: Begin from process $R = 1$.
3: **for** $i = 1, N_{\mathrm{a}} - 1$ **do**
4: **for** $j = i + 1, N_{\mathrm{a}}$ **do**
5: Assign pair (i, j) to process R.
6: Increase R by one.
7: **if** $R = N_{\mathrm{p}} + 1$ **then**
8: Reduce R by one so that $R = N_{\mathrm{p}}$.
9: Step 6 now reduces R by one.
10: **else if** $R = 0$ **then**
11: Increase R so that $R = 1$.
12: Step 6 now increases R by one.
13: **end if**
14: **end for**
15: **end for**

This approach for distributing the pairs over processes is general to all potentials. Furthermore, it is well-suited to the evaluation of the pair energy, as each process must calculate the energies of its assigned pairs only. Pairs are assigned in ascending order and then descending order so that when a single atom is moved, the triplets that must be re-evaluated are more equitably distributed. For every triplet, Algorithm 3 is used to verify that all MI distances in the triplet do not exceed the cutoff r_{c} before any calculation is undertaken.

https://doi.org/10.1021/acs.jctc.3c00113
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

Algorithm 3 Check the distances in a triplet $\alpha\beta\gamma$. The shortest distance in the triplet is r_{min}.

1: Verify that all distances in the triplet are below r_c.
2: Keep the atom with the lowest index in r_{min} in its current position (e.g. if $r_{min} = r_{\alpha\beta}$ then leave α unmoved).
3: Move the other two atoms to their minimum image positions relative to the unmoved atom.
4: Calculate the distance r' between the two moved atoms in their current positions.
5: **if** $r' > r_c$ **then**
6: A distance in $\alpha\beta\gamma$ exceeds r_c so its energy need not be calculated explicitly.
7: **else**
8: No distance in $\alpha\beta\gamma$ exceeds r_c so its energy must be calculated explicitly.
9: **end if**

Following assignment of the atom pairs, each process calculates the nonadditive energies of all triplets $\alpha < \beta < \gamma$ for which it owns the $\alpha\beta$ pairs. The individual nonadditive triplet energies on process R are stored in a vector $\mathbf{u}^{(R)}$, which has length equal to the number of triplets, $N_{tri}^{(R)}$, assigned to the process. The steps undertaken for the parallel calculation of U_{NA}^{tot} are given in Algorithm 4, in which $U_{NA}^{(R)}$ is the contribution to U_{NA}^{tot} from process R. Because the pairs assigned to a process are the same throughout the simulation, so are the triplets that it will evaluate.

Algorithm 4 Parallel calculation of U_{NA}^{tot}

1: A number of $\alpha\beta$ atom pairs $N_{pairs}^{(R)}$ are assigned to process R by algorithm 2.
2: **for** $i = 1, N_{pairs}^{(R)}$ **do**
3: Take the ith pair $(\alpha\beta)$ assigned to the process.
4: **for** $\gamma = \beta + 1, N_a$ **do**
5: Check that all distances in the triplet are MI distances within the cut-off r_c via algorithm 3.
6: If so, calculate non-additive energy for the $\alpha\beta\gamma$ triplet via the two innermost sums in equation 9; otherwise set the energy to 0.
7: Store the energy from the last step in $\mathbf{u}^{(R)}$.
8: **end for**
9: **end for**
10: Sum over the elements of $\mathbf{u}^{(R)}$ to find the contribution to $U_{NA}^{(R)}$ from the triplets on this process.
11: Send this contribution to the root process.
12: **if** on the root process **then**
13: Add all $U_{NA}^{(R)}$ to get U_{NA}^{tot}.
14: **end if**

Energy Change after Atom Displacement. Finding the new nonadditive energy of all triplets affected by moving an atom δ requires recalculation of the triplets that contain δ only, which renders it faster than the full box calculation. However, recalculation of the energy after an atom moves is undertaken more frequently during simulation. The nonadditive energy of triplets containing δ, $U_{NA}^{(\delta)}$, are found by modifying eq 9 to give

$$\frac{U_{NA}^{(\delta)}}{\sigma_f^2} = \sum_{\delta<\beta<\gamma}^{N_a} \sum_{i=1}^{N_t} \lambda_i \sum_{j=1}^{N_{perm}} (E_{iP_{1j}})_{\delta\beta} (E_{iP_{2j}})_{\delta\gamma} (E_{iP_{3j}})_{\beta\gamma} + \sum_{\alpha<\delta<\gamma}^{N_a} \sum_{i=1}^{N_t} \lambda_i \sum_{j=1}^{N_{perm}} (E_{iP_{1j}})_{\alpha\delta} (E_{iP_{2j}})_{\alpha\gamma} (E_{iP_{3j}})_{\delta\gamma} + \sum_{\alpha<\beta<\delta}^{\delta} \sum_{i=1}^{N_t} \lambda_i \sum_{j=1}^{N_{perm}} (E_{iP_{1j}})_{\alpha\beta} (E_{iP_{2j}})_{\alpha\delta} (E_{iP_{3j}})_{\beta\delta} \quad (10)$$

The change in nonadditive energy ΔU_{NA} is found by taking the difference between $U_{NA}^{(\delta)}$ before and after the move.

Displacement of δ alters the exponentials for all pairs in which it appears. Hence, calculating ΔU_{NA} for a displacement requires recalculation of the exponentials corresponding to $N_a - 1$ affected distances, where N_a is the number of atoms. Similarly, a Monte Carlo addition introduces N_a new pairs, necessitating a set of N_a new exponential calculations.

As Monte Carlo moves may be rejected, all old exponentials involving δ are stored. This is done separately on each process, as each process needs only save its own share of the exponentials before updating the exponential array. The exponentials on a single process are updated by Algorithm 5, where $\mathbf{O}_m$ contains all exponentials pertaining to the mth pair on process R.

https://doi.org/10.1021/acs.jctc.3c00113
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

Algorithm 5 Update exponentials involving atom δ on a node with shared memory. This algorithm is run for all processes on every node and $N_{\text{exp}}^{(R)}$ is the number of changed pairs assigned to process R.

1: Follow steps 1-3 of algorithm 1 for the changed distances to distribute them across the processes on the node.
2: **for** $m = 1, N_{\text{exp}}^{(R)}$ **do**
3: Take the mth pair $(\alpha\beta)$ assigned to process R.
4: Save the current exponentials for this pair in $\mathbf{E}_{\alpha\beta}$ to $\mathbf{O}_m$.
5: **if** $r_{\alpha\beta} \leq r_{\text{c}}$ **then**
6: **for** $i = 1, N_{\text{t}}$ **do**
7: **for** $k = 1, N_{\text{d}}$ **do**
8: Evaluate $\exp\left(\frac{\left[x_{\alpha\beta} - x'_{ik}\right]^2}{2l_k^2}\right)$.
9: Save the new exponential to $(\text{E}_{ik})_{\alpha\beta}$
10: **end for**
11: **end for**
12: **else**
13: $\mathbf{E}_{\alpha\beta} = 0$.
14: **end if**
15: **end for**

For process R, the change in nonadditive energy following an atom move is denoted $\Delta U_{\text{NA}}^{(R)}$. ΔU_{NA} is therefore the sum of $\Delta U_{\text{NA}}^{(R)}$ over all processes. For each process, $\Delta U_{\text{NA}}^{(R)}$ is calculated from the difference between the new and old nonadditive energies for each triplet containing the moved atom δ. The new nonadditive energies of each affected triplet on process R are stored in the vector $\mathbf{c}^{(R)}$, ready to be placed into $\mathbf{u}^{(R)}$ if the move is accepted. The length of $\mathbf{c}^{(R)}$ is equal to the number of affected triplets on the process, $N_{\text{changed}}^{(R)}$. The steps taken on each process to identify affected triplets and recalculate their energies are shown in Algorithm 6.

Algorithm 6 Parallel calculation of ΔU_{NA} after movement of atom δ. $N_{\text{pairs}}^{(R)}$ is the number of pairs assigned to process R.

1: **for** $i = 1, N_{\text{pairs}}^{(R)}$ **do**
2: The ith pair for process R is $\alpha\beta$, $\alpha < \beta$.
3: **if** $\alpha < \beta < \delta$ **then**
4: Check that all distances in the triplet are MI distances within the cut-off r_{c} via algorithm 3.
5: If so, calculate the new non-additive energy for $\alpha\beta\delta$; otherwise set this energy to 0.
6: Store the energy from the last step in $\mathbf{c}^{(R)}$.
7: Calculate the contribution of this triplet to $\Delta U_{\text{NA}}^{(R)}$ by finding the difference between its new and old energy.
8: Add the value from the last step to $\Delta U_{\text{NA}}^{(R)}$.
9: **else if** $\alpha = \delta$ or $\beta = \delta$ **then**
10: **for** $\gamma = \beta + 1, N_a$ **do**
11: Repeat steps 3-7 for the $\delta\beta\gamma$ triplet or $\alpha\delta\gamma$ triplet.
12: **end for**
13: **end if**[1]
14: **end for**
15: Send the total $\Delta U_{\text{NA}}^{(R)}$ to the root process.
16: **if** on the root process **then**
17: Evaluate ΔU_{NA} by summing all $\Delta U_{\text{NA}}^{(R)}$.
18: **end if**

[1]If $\alpha < \delta < \beta$ or $\delta < \alpha < \beta$ do nothing.

The combination of all algorithms discussed thus far gives rise to a general method for parallelizing the calculation of the nonadditive energy in a simulation with a GP potential. This method is in large part general to other methods of prediction too. After a brief discussion of what must be done on accepting or rejecting a move, the full algorithm is presented in this section.

When a move is accepted in a Monte Carlo simulation, the methods outlined above require the transfer of the triplet nonadditive energies from $\mathbf{c}^{(R)}$ into $\mathbf{u}^{(R)}$. $U_{\text{NA}}^{\text{tot}}$, the position of the moved atom, and the inverse distances must also be updated. Meanwhile, rejecting a move necessitates replacing the exponentials in $\mathbf{E}$ with the previous ones saved in $\mathbf{O}$ from each process.

The methods presented in the preceding discussion are summarized below in Algorithm 7, which describes how the nonadditive energies are calculated for an entire simulation over N_{moves} moves. This algorithm includes all considerations of the periodic boundary conditions and minimum image convention, and can be generalized to exchange and volume change moves. Furthermore, other than steps 3 and 8 it is general to all nonadditive potentials.

https://doi.org/10.1021/acs.jctc.3c00113
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

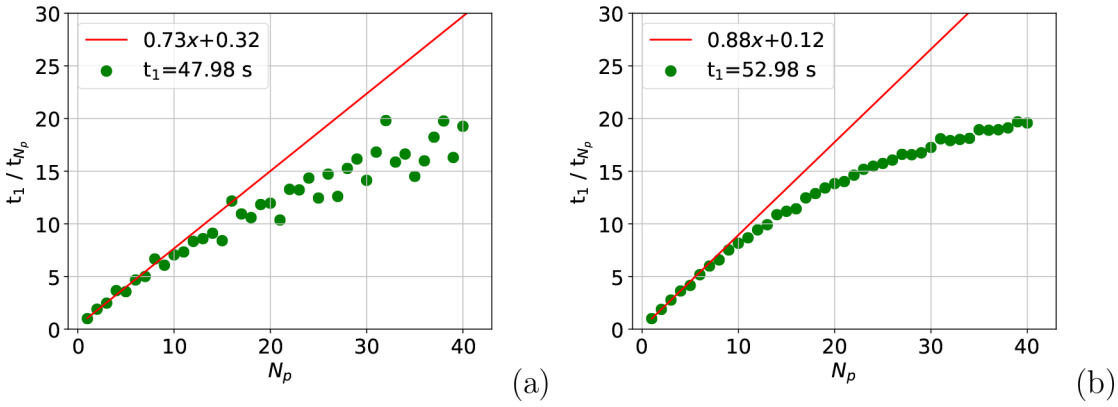


Figure 1. Speed-up on one node for the atom move (a) and full box (b) calculations. The move data are summed over 150 moves, of which half were accepted.

Algorithm 7 Full algorithm for non-additive energy calculations with a GP potential

1: Fill **X** with the minimum image distances.
2: Pre-calculate the exponential array using algorithm 1.
3: Evaluate U_{NA}^{tot} via algorithm 4.
4: **for** $i = 1, N_{moves}$ **do**
5: Propose a trial move.
6: Find the values of the changed minimum image distances.
7: Update the affected exponentials using algorithm 5.
8: Determine ΔU_{NA} via algorithm 6.
9: Accept or reject the move and update the data, as detailed in the text.
10: **end for**

4. COMPUTATIONAL DETAILS

All calculations were undertaken for a simulation box of argon atoms to assess the speed-up possible with the proposed parallelization. Much recent work implementing *ab initio* potentials in simulation[7−12] has focused on noble gases as exemplars due to the relative simplicity of their interactions. Argon was chosen here specifically due to the ability of GPs to model Ar_3 triplets[22] and their success in developing virial coefficients for CO_2−Ar mixtures.[19]

Augusta, a University of Nottingham high-performance computer, was used for all calculations. Each node on Augusta has 2 × 20 core processors (Intel Xeon Gold 6138 20C 2.0 GHz CPU). Code was written in Fortran, using the Fortran message passing interface for parallelization and openMP for shared memory. For all calculations, the O3 optimizer flag was enabled to minimize computational time prior to parallelization.

The three body nonadditive energy was calculated for a simulation box of N_a = 500 argon atoms, which had random starting positions. The side length L of the box was 29 Å, corresponding to a density of 1.4 kg dm^{-3}, which exceeded the critical density.

Periodic boundary conditions (PBCs) and a minimum image convention (MIC) were included, with a cutoff of $r_c = L/2$ = 14.5 Å applied to all calculations. The cutoff resulted in roughly 12% of nonadditive energy calculations being undertaken explicitly. 150 displacements were attempted, with atoms selected at random and moved up to 1.5 Å forward or backward along each of the *x*, *y*, and *z* axes. These moves were alternately accepted and rejected without the use of the Metropolis method.

The nonadditive GP potential was trained on a 999-point training set, via the method of Wheatley and Graham,[19] in which all energies were calculated at the CCSD(T) level of theory using an aug-cc-pVTZ basis set. This potential achieved a root-mean-square (RMS) error of <1% of the RMS value of the reference set and was used to make nonadditive energy predictions of first-principles quality for all explicit calculations. Training on this set took around 7 h. The nonadditive energy of a triplet $\alpha\beta\gamma$ was calculated as

$$U_{NA} = U(\alpha\beta\gamma) - U(\alpha\beta) - U(\alpha\gamma) - U(\beta\gamma) + U(\alpha) + U(\beta) + U(\gamma) \quad (11)$$

where $U(\alpha\beta\gamma)$ is the total triplet energy, the negative terms are the pairwise interaction energies, and the final three terms are the energies of the individual atoms.

5. RESULTS AND DISCUSSION

The factor by which the speed of a calculation increases when spread over a number of processes N_p is referred to here as the speed-up. This is defined as t_1/t_{N_p}, where t_i is the wall clock time in seconds for *i* processes. Speed-up data are presented for atom move and full box calculations.

The speed-up when completing the atom move and full box calculations in their entirety, t_{total}, is shown in Figure 1. This evidence a 7.5-fold speed-up on 10 processes for both calculations under realistic simulation conditions. On 40 processes, the speed-up is roughly 20-fold.

The speed-up on 40 processes means that 2.6 s is required to calculate the full simulation box energy and 2.4 s to displace 150 atoms. For a simulation comprising 100,000 cycles, each of which consists of one volume change and displacement of every atom, this corresponds to 12 days for a first-principles-quality simulation. This number is found by calculating the time for 100,000 volume changes on 40 processes (*i.e.* 2.6 s × 100, 000 ≈ 3 days) and adding it to the time taken to displace all 500 atoms 100,000 times. The time to displace one atom is 2.4/150 = 0.016 s, which means the total displacement time is 0.016 s × 500 × 100, 000 ≈ 9 days. The equivalent time to utilize a GP potential

https://doi.org/10.1021/acs.jctc.3c00113
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

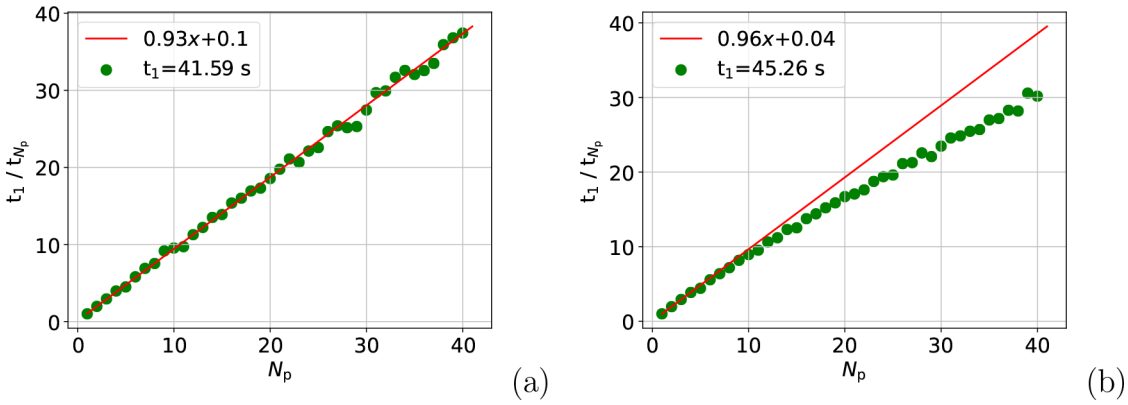


Figure 2. Speed-up on one node in calculating the nested triplet sum after an atom is moved (a) and for the full simulation box (b). The data in (a) are summed over 150 moves.

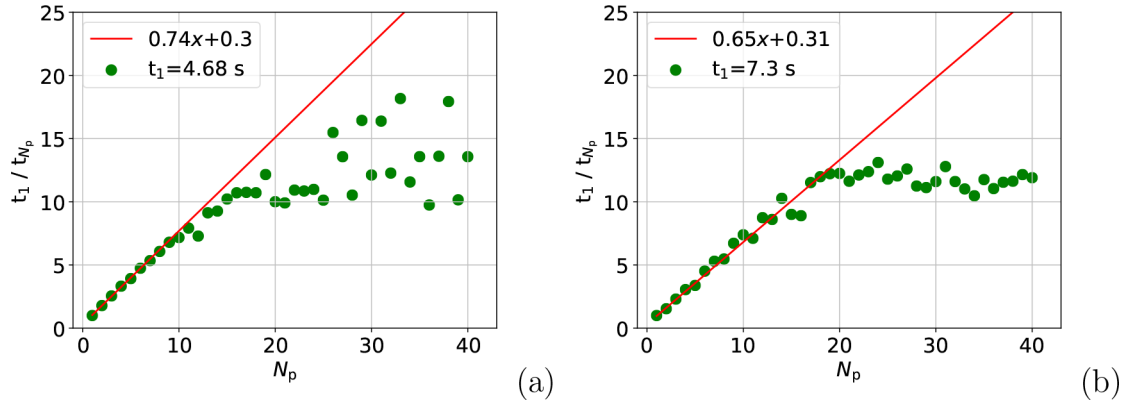


Figure 3. Speed-up on a single node in calculating the exponentials for the atom move (a) and the full box (b) calculations. The atom move data were summed over 150 moves.

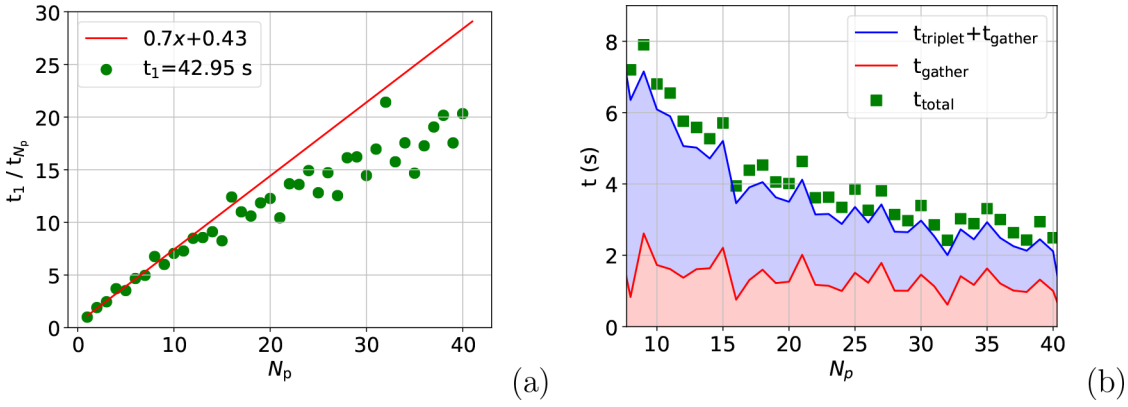


Figure 4. Speed-up in t_{wait}, which is the sum of $t_{triplet}$ and t_{gather} (a), and the values of $t_{triplet}$, t_{gather}, and t_{total} (b). The data are summed over 150 atom move calculations, and part (b) starts at $N_p = 8$.

on one process is 245 days. Calculating the additive energy requires a small fraction of the time for the nonadditive energy. Due to differences in the PES being considered and the number of calculations, direct comparison of these timings to those of other machine-learned potentials cannot be made without reservation. However, the times achieved here are comparable to those elsewhere,[46] and faster reported GP potentials[39] featured lower cutoff values and do not consider nonadditive intermolecular interactions.

An equivalent calculation using CCSD(T) with a complete basis set extrapolation would require around one h per triplet. When $N_a = 500$ there are ~20.7 million triplets, of which 12% must be evaluated due to the cutoff. This necessitates ~2.48 million calculations, which at one h each would take over 100,000 days. Similarly, the 500 atom displacements would require over 300,000 days. Thus, the time for a single cycle with direct CCSD(T) calculations would take in excess of 400,000 days or over 1000 years.

The time to undertake the nested sum over triplets, $t_{triplet}$, in eq 9 is the main contribution to t_{total} and is considered in Figure 2. Figure 2(a) shows that the speed-up in $t_{triplet}$ is excellent overall for the atom move calculation. Meanwhile, Figure 2(b)

https://doi.org/10.1021/acs.jctc.3c00113
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

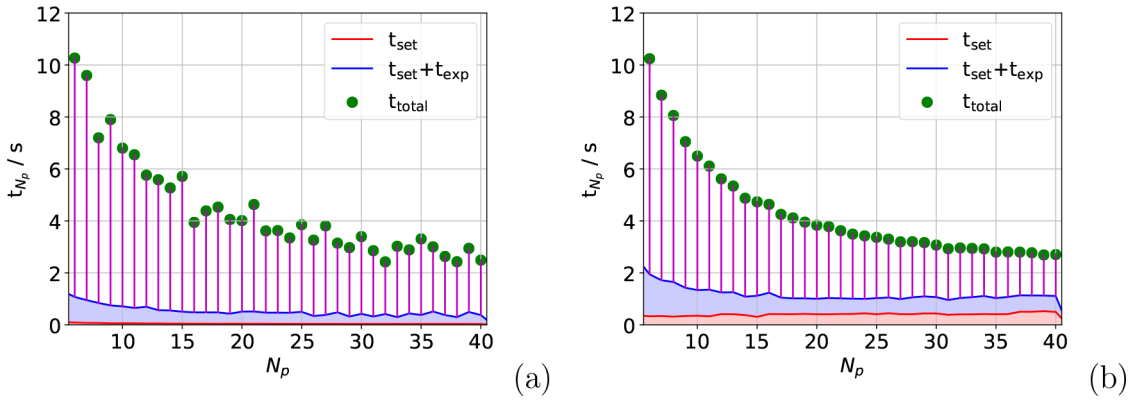


Figure 5. Plots showing the contributions to t_{total} against N_p for N_p = 6−40 for the atom move (a) and full box (b) calculations. The bottom, red section shows the contribution of the setup time t_{set}, while the area under the blue line shows t_{exp}. The vertical purple lines show the value of t_{wait} for each calculation.

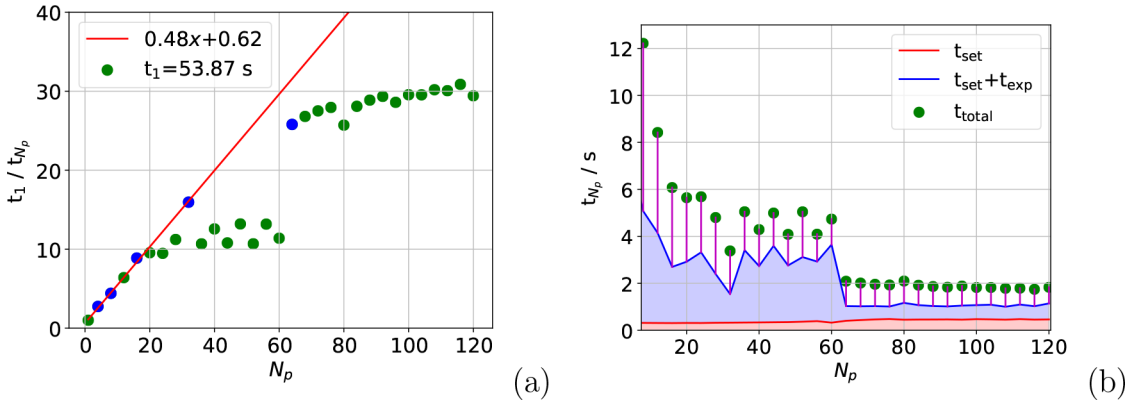


Figure 6. Speed-up in t_{total} (a) and the values of the main contributions to t_{total} (b) for the full box calculation on four nodes. Blue points in (a) correspond to N_p values of 2^x.

illustrates that the speed-up in the full box calculation is also impressive, if slightly lower. This is significant because the speed-up in $t_{triplet}$ facilitates the reduction in simulation time shown in Figure 1 for both calculations. For example, for the full box calculation on a single process, $t_{triplet}$ = 0.28 s, which is 85% of t_{total}. For the atom move calculations, $t_{triplet}$ is 87% of t_{total}.

The time required to calculate and store the exponentials in shared memory, t_{exp}, is also a notable contribution to t_{total}. For example, t_{exp} is ∼10% of t_{total} for the atom move calculation on a single process. For the full box calculation this rises to 14%. Figure 3 shows that an 11-fold speed-up in t_{exp} at $N_p \approx 15$ is not bettered reliably for the atom move calculation, while for the full box calculation a plateau at a 12-fold speed-up is achieved at $N_p \approx 17$. The plateaus in t_{exp} for both calculations partially explain the reduction in speed-up seen in Figure 1 relative to that in Figure 2. After $N_p \approx 15$, t_{exp} is effectively a fixed cost of 0.42 and 0.61 s for the atom move and full box calculations, respectively. This means when N_p = 40, t_{exp} accounts for 18% of the atom move calculation and 23% of the full box calculation. Alongside the greater fixed costs associated with the full box calculation, this is the reason for the reduction in speed-up between Figure 2(b) and Figure 1(b).

Meanwhile, for the atom move calculation, the reduced speed-up in Figure 1 compared with Figure 2(a) is caused by the "wait" for the busiest process to send its partial sum before the total energy can be calculated. The effect of the "wait" is proven by examining the speed-up on the root process in $t_{wait} = t_{triplet} + t_{gather}$, where $t_{triplet}$ is the time to calculate the nonadditive energies, and t_{gather} is the time to add them and gather the partial sums. Figure 4 shows the speed-up in t_{wait} for the atom move calculation, revealing that it is similar to the speed-up seen in Figure 1. This implies that waiting for the busiest process at the gather stage is the limiting factor on the total speed-up for the atom move calculation. This assertion is reinforced by Figure 4, which shows $t_{total} \approx t_{wait}$.

Figure 5 shows the t_{total} breakdown into its main contributions for both calculations. The setup time t_{set}, which is shown in red, includes all fixed costs such as instantiating arrays and process communication times. The figure shows that t_{wait} is the main contribution to t_{total} when N_p = 40 for both calculations and that the fixed costs for the atom move calculation are insignificant. Fixed costs for the full box calculation are slightly larger, however, as is the contribution of t_{exp}.

The preceding results indicate that neither calculation had plateaued when spread over 40 processes. As such, the speed-up in both calculations across 120 processes split evenly over four nodes was investigated. The even distribution of processes means that, for example, when N_p = 120, there are 120/4 = 30 processes on each node. When N_p = 1, the process is on a single node.

The speed-up over multiple nodes for the full box calculation is displayed in Figure 6(a). A similar trend is observed over the blue points in this figure as in the overall trend in Figure 1. For example, at N_p = 32 in Figure 6(a), a roughly 17-fold speed-up is seen, which matches closely the speed-up on a single node at equivalent N_p. Each blue point represents a calculation for which

https://doi.org/10.1021/acs.jctc.3c00113
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

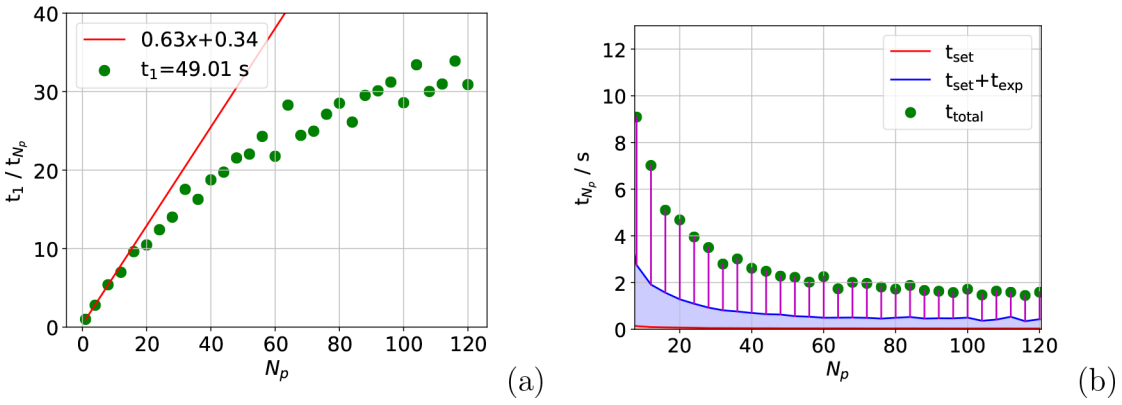


Figure 7. Speed-up in t_{total} (a) and the values of the main contributions to t_{total} (b) for the atom move calculations on four nodes. The calculations are spread evenly across four nodes, and the data in (a) were summed over 150 moves.

the total number of processes was a power of two (*i.e.* $N_p = 4$ for 2^2, $N_p = 8$ for 2^3, *etc.*). This suggests that the points on the plot where the calculation time increases with N_p are a product of the HPC architecture, rather than inherent to the algorithm. The similarity between the speed-up on four nodes and on one node for the unaffected points suggests that the algorithm extends to multiple nodes well.

The cause of the inconsistency of the parallelization of the full box calculation is evidenced in Figure 6(b), which shows that t_{exp} increased drastically for certain values of N_p. When this happened, t_{exp} became the dominant cost in the calculation, degrading speed-up. The figure also shows that when $N_p > 60$ the setup and exponential calculation times exceed t_{wait}. This explains the plateau in Figure 6(a).

For the atom move calculation, shown in Figure 7(a), a 20-fold speed-up is achieved across 40 processes, which matches the results seen for one node in Figure 1(a). This is evidence that the calculation can be parallelized across multiple nodes without loss of performance. In addition, the distribution across more processes gives rise to a further increase in speed-up. Meanwhile, Figure 7(b) shows that t_{wait} nears convergence when $N_p = 120$. This implies that spreading the calculation over more processes is unlikely to reduce calculation time further, with the costs associated with producing and communicating between further processes having the potential to degrade speed-up.

For any calculation in which $t_{triplet}$ constitutes a larger proportion of t_{total}, such as in a system containing molecules that require a more expensive potential with more training points, a further speed-up over multiple nodes would be expected. This would be most noticeable in the full box calculation as it would eliminate the plateau shown in Figure 6(a), but it would lead to enhanced speed-up in both. The 30-fold speed-up already observed for these calculations corresponds to an eight day calculation for the simulation outlined earlier, a massive reduction from the 245 days required if no parallelization was undertaken.

6. CONCLUSIONS

The parallelization strategy outlined reduces significantly the calculation time for the nonadditive energy of argon. The method is robust to different computational setups, with the parallelization conferring a 20-fold speed-up on 40 processes in both calculations on one or many nodes. This corresponds to a 12 day calculation time for a first-principles-quality nonadditive energy in a simulation comprising 100,000 cycles. Distribution over more processes leads to a further reduction, with 8 days required for the calculation rather than 245 if no parallelization is attempted.

Moreover, simulations requiring a larger number of triplet calculations, such as those with large simulation boxes, would see even better parallelization. This is also true if individual triplet evaluations were more costly, as would be the case for a molecular system with a potential that requires more training points. In addition, the method for distributing the triplet nonadditive energy calculations is applicable to different potentials.

To apply the strategy to Monte Carlo simulations, long-range corrections must be applied. Additive long-range corrections are described elsewhere,[6] while a method for implementing nonadditive corrections is given in the Supporting Information. This will allow a proof-of-concept calculation of phase coexistence in argon via a Monte Carlo simulation. Such simulations could be run for different properties or different monatomic systems using a suitable GP potential via the methods outlined above. Extension to molecular dynamics simulations is also possible by introducing forces, a method for which is also given in the Supporting Information.

Applications to systems of small molecules, such as those observed in CCUS pipelines, would require extension of the above methodology to mixtures of atomic and molecular species. This is relatively straightforward, requiring only the inclusion of an operator to identify the correct GP hyperparameters for any interaction and a method to order the interatomic distances in interactions between different species to construct the associated permutation matrices. Thus, the parallelization strategy discussed here represents a significant step toward quantitatively accurate simulations with GP potentials that give additive and nonadditive energies derived from first principles.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jctc.3c00113.

Methods to calculate forces and nonadditive long-range corrections for simulations of a single atomic species (PDF)

■ AUTHOR INFORMATION

Corresponding Author

Jack Broad − *Molecular Foundry, Lawrence Berkeley National Laboratory, Berkeley, California 94720, United States;*

https://doi.org/10.1021/acs.jctc.3c00113
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

orcid.org/0000-0002-4871-8367; Email: jwbroad@lbl.gov

Authors

Richard J. Wheatley − *School of Chemistry, University of Nottingham, Nottingham NG7 2RD, England*

Richard S. Graham − *School of Mathematical Sciences, University of Nottingham, Nottingham NG7 2RD, England;* orcid.org/0000-0002-5530-8120

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jctc.3c00113

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

The authors thank the University of Nottingham's Augusta HPC service, which provides a High Performance Computing service to the University's research community. See https://www.nottingham.ac.uk/it-services/research/hpc/ for more details. The authors are also thankful for funding provided by the Leverhulme Trust Doctoral Scholarship through Modelling and Analytics for a Sustainable Society (MASS) at the University of Nottingham.

■ REFERENCES

(1) Ilyina, T. Hidden trends in the ocean carbon sink. *Nature* **2016**, *530*, 426−427.

(2) Kim, H. W.; Yoon, H. W.; Yoon, S.-M.; Yoo, B. M.; Ahn, B. K.; Cho, Y. H.; Shin, H. J.; Yang, H.; Paik, U.; Kwon, S.; Choi, J.-Y.; Park, H. B. Selective gas transport through few-layered graphene and graphene oxide membranes. *Science* **2013**, *342*, 91−95.

(3) Lynch, C. I.; Rao, S.; Sansom, M. S. Water in nanopores and biological channels: A molecular simulation perspective. *Chem. Rev.* **2020**, *120*, 10298−10335.

(4) Dymond, J. H.; Marsh, K. N.; Wilhoit, R. C.; Frenkel, M. D. *Virial Coefficients of Mixtures*; Springer-Verlag: Berlin, 2021.

(5) Sadus, R. J. *Molecular Simulation of Fluids*; Elsevier: Amsterdam, 2002.

(6) Allen, M. P.; Tildesley, D. J. *Computer Simulation of Liquids*; Oxford university press: Oxford, 2017.

(7) Xu, P.; Guidez, E. B.; Bertoni, C.; Gordon, M. S. Perspective: Abinitioforce field methods derived from quantum mechanics. *J. Chem. Phys.* **2018**, *148*, No. 090901.

(8) Singh, A. N.; Dyre, J. C.; Pedersen, U. R. Solid−liquid coexistence of neon, argon, krypton, and xenon studied by simulations. *J. Chem. Phys.* **2021**, *154*, 134501.

(9) Deiters, U. K.; Sadus, R. J. Fully *a*prioriprediction of the vapor-liquid equilibria of Ar, Kr, and Xe from abinitiotwo-body plus three-body interatomic potentials. *J. Chem. Phys.* **2019**, *151*, No. 034509.

(10) Deiters, U. K.; Sadus, R. J. Interatomic Interactions Responsible for the Solid−Liquid and Vapor−Liquid Phase Equilibria of Neon. *Journal Phys. Chem. B* **2021**, *125*, 8522−8531.

(11) Deiters, U. K.; Sadus, R. J. First-principles determination of the solid-liquid-vapor triple point: The noble gases. *Phys. Rev. E* **2022**, *105*, No. 054128.

(12) Ströker, P.; Hellmann, R.; Meier, K. Thermodynamic properties of argon from Monte Carlo simulations using abinitiopotentials. *Phys. Rev. E* **2022**, *105*, No. 064129.

(13) Behler, J. Perspective: Machine learning potentials for atomistic simulations. *J. Chem. Phys.* **2016**, *145*, 170901.

(14) Mueller, T.; Hernandez, A.; Wang, C. Machine learning for interatomic potential models. *J. Chem. Phys.* **2020**, *152*, No. 050902.

(15) Rasmussen, C.; Williams, C. *Gaussian Processes for Machine Learning*; MIT Press, 2006.

(16) Unke, O. T.; Chmiela, S.; Gastegger, M.; Schütt, K. T.; Sauceda, H. E.; Müller, K.-R. Spookynet: Learning force fields with electronic degrees of freedom and nonlocal effects. *Nat. Commun.* **2021**, *12*, 7273.

(17) Zhang, Y.; Xia, J.; Jiang, B. REANN: A PyTorch-based end-to-end multi-functional deep neural network package for molecular, reactive, and periodic systems. *J. Chem. Phys.* **2022**, *156*, 114801.

(18) Kamath, A.; Vargas-Hernández, R. A.; Krems, R. V.; Carrington, T., Jr; Manzhos, S. Neural networks vs Gaussian process regression for representing potential energy surfaces: A comparative study of fit quality and vibrational spectrum accuracy. *J. Chem. Phys.* **2018**, *148*, 241702.

(19) Graham, R. S.; Wheatley, R. J. Machine learning for non-additive intermolecular potentials: from quantum chemistry to first-principles predictions. *ChemComm* **2022**, *58*, 6898−6901.

(20) Uteva, E.; Wheatley, R. J.; Wilkinson, R. D.; Graham, R. S. Interpolation of intermolecular potentials using Gaussian processes. *J. Chem. Phys.* **2017**, *147*, 161706.

(21) Cresswell, A. J.; Wheatley, R. J.; Wilkinson, R. D.; Graham, R. S. Molecular simulation of the thermophysical properties and phase behaviour of impure CO_2 relevant to CCS. *Faraday Discuss.* **2016**, *192*, 415−436.

(22) Uteva, E.; Wheatley, R. J.; Wilkinson, R. D.; Graham, R. S. Active learning in Gaussian process interpolation of potential energy surfaces. *J. Chem. Phys.* **2018**, *149*, 174114.

(23) Broad, J.; Preston, S.; Wheatley, R. J.; Graham, R. S. Gaussian process models of potential energy surfaces with boundary optimization. *J. Chem. Phys.* **2021**, *155*, 144106.

(24) Handley, C. M.; Hawe, G. I.; Kell, D. B.; Popelier, P. L. A. Optimal construction of a fast and accurate polarisable water potential based on multipole moments trained by machine learning. *Phys. Chem. Chem. Phys.* **2009**, *11*, 6365−6376.

(25) Mills, M. J. L.; Popelier, P. L. A. Intramolecular polarisable multipolar electrostatics from the machine learning method Kriging. *Comput. Theor. Chem.* **2011**, *975*, 42−51.

(26) Mills, M. J. L.; Popelier, P. L. A. Polarisable multipolar electrostatics from the machine learning method Kriging: an application to alanine. *Theor. Chem. Acc.* **2012**, *131*, 1137.

(27) Kandathil, S. M.; Fletcher, T. L.; Yuan, Y.; Knowles, J.; Popelier, P. L. A. Accuracy and tractability of a Kriging model of intramolecular polarizable multipolar electrostatics and its application to histidine. *J. Comput. Chem.* **2013**, *34*, 1850−1861.

(28) Dai, J.; Krems, R. V. Interpolation and extrapolation of global potential energy surfaces for polyatomic systems by Gaussian processes with composite kernels. *J. Chem. Theory Comput.* **2020**, *16*, 1386−1395.

(29) Sugisawa, H.; Ida, T.; Krems, R. V. Gaussian process model of 51-dimensional potential energy surface for protonated imidazole dimer. *J. Chem. Phys.* **2020**, *153*, 114101.

(30) Schmitz, G.; Klinting, E. L.; Christiansen, O. A Gaussian process regression adaptive density guided approach for potential energy surface construction. *J. Chem. Phys.* **2020**, *153*, No. 064105.

(31) Burn, M. J.; Popelier, P. L. A. Creating Gaussian process regression models for molecular simulations using adaptive sampling. *J. Chem. Phys.* **2020**, *153*, No. 054111.

(32) Boussaidi, M. A.; Ren, O.; Voytsekhovsky, D.; Manzhos, S. Random Sampling High Dimensional Model Representation Gaussian Process Regression (RS-HDMR-GPR) for multivariate function representation: application to molecular potential energy surfaces. *J. Phys. Chem. A* **2020**, *124*, 7598−7607.

(33) Sivaraman, G.; Krishnamoorthy, A. N.; Baur, M.; Holm, C.; Stan, M.; Csányi, G.; Benmore, C.; Vázquez-Mayagoitia, Á. Machine-learned interatomic potentials by active learning: amorphous and liquid hafnium dioxide. *Npj Comput. Mater.* **2020**, *6*, 1−8.

(34) Caro, M. A.; Csanyi, G.; Laurila, T.; Deringer, V. L. Machine learning driven simulated deposition of carbon films: From low-density to diamondlike amorphous carbon. *Phys. Rev. B* **2020**, *102*, 174201.

(35) Liu, Y. B.; Yang, J. Y.; Xin, G. M.; Liu, L. H.; Csányi, G.; Cao, B. Y. Machine learning interatomic potential developed for molecular simulations on thermal properties of β-Ga_2 O_3. *J. Chem. Phys.* **2020**, *153*, 144501.

(36) Glielmo, A.; Sollich, P.; De Vita, A. Accurate interatomic force fields via machine learning with covariant kernels. *Phys. Rev. B* **2017**, *95*, 214302.
(37) Glielmo, A.; Zeni, C.; De Vita, A. Efficient nonparametric n-body force fields from machine learning. *Phys. Rev. B* **2018**, *97*, 184307.
(38) Zeni, C.; Rossi, K.; Glielmo, A.; Fekete, Á.; Gaston, N.; Baletto, F.; De Vita, A. Building machine learning force fields for nanoclusters. *J. Chem. Phys.* **2018**, *148*, 241739.
(39) Zuo, Y.; Chen, C.; Li, X.; Deng, Z.; Chen, Y.; Behler, J.; Csanyi, G.; Shapeev, A. V.; Thompson, A. P.; Wood, M. A.; Ong, S. P. Performance and Cost Assessment of Machine Learning Interatomic Potentials. *J. Phys. Chem. A* **2020**, *124*, 731−745.
(40) Torrey, L.; Shavlik, J. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*; IGI Global: Hershey, PA, 2010; pp 242−264.
(41) Pan, S. J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345−1359.
(42) Accelerating Breakthrough Innovation in Carbon Capture, Utilization and Storage. https://www.energy.gov/fecm/downloads/accelerating-breakthroughinnovation-carbon-capture-utilization-and-storage (accessed 27/06/2022).
(43) McKay, M. D.; Beckman, R. J.; Conover, W. J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* **2000**, *42*, 55−61.
(44) Stein, M. Large sample properties of simulations using Latin hypercube sampling. *Technometrics* **1987**, *29*, 143−151.
(45) Loh, W.-L. On Latin hypercube sampling. *Ann. Stat.* **1996**, *24*, 2058−2080.
(46) Houston, P. L.; Qu, C.; Nandi, A.; Conte, R.; Yu, Q.; Bowman, J. M. Permutationally invariant polynomial regression for energies and gradients, using reverse differentiation, achieves orders of magnitude speed-up with high precision compared to other machine learning methods. *J. Chem. Phys.* **2022**, *156*, No. 044120.

https://doi.org/10.1021/acs.jctc.3c00113
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX