JCTC Journal of Chemical Theory and Computation

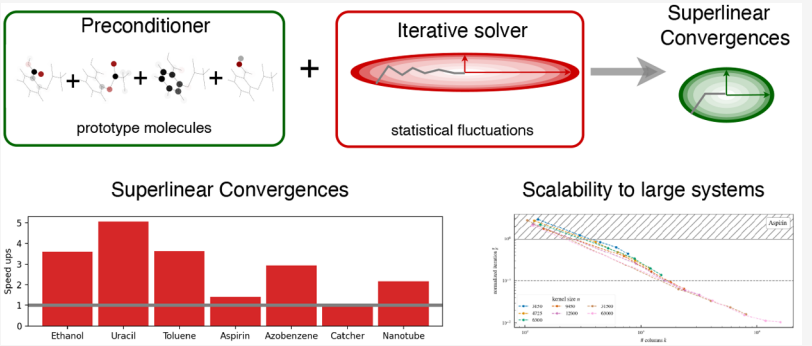
pubs.acs.org/JCTC Article

Reconstructing Kernel-Based Machine Learning Force Fields with Superlinear Convergence

Stefan Blücher, Klaus-Robert Müller,* and Stefan Chmiela*

Cite This: https://doi.org/10.1021/acs.jctc.2c01304 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information



ABSTRACT: Kernel machines have sustained continuous progress in the field of quantum chemistry. In particular, they have proven to be successful in the low-data regime of force field reconstruction. This is because many equivariances and invariances due to physical symmetries can be incorporated into the kernel function to compensate for much larger data sets. So far, the scalability of kernel machines has however been hindered by its quadratic memory and cubical runtime complexity in the number of training points. While it is known that iterative Krylov subspace solvers can overcome these burdens, their convergence crucially relies on effective preconditioners, which are elusive in practice. Effective preconditioners need to partially presolve the learning problem in a computationally cheap and numerically robust manner. Here, we consider the broad class of Nyström-type methods to construct preconditioners based on successively more sophisticated low-rank approximations of the original kernel matrix, each of which provides a different set of computational trade-offs. All considered methods aim to identify a representative subset of inducing (kernel) columns to approximate the dominant kernel spectrum.

1. INTRODUCTION

In recent years, machine learning force fields (MLFF) have emerged as a valuable modeling tool in quantum chemistry.[1−4] The promise of MLFFs is to combine the performance of classical FFs with the accuracy of computationally expensive high-level ab initio methods. In this setting, both neural networks[5−13] and kernel-based approaches[14−22] have been successfully applied. Kernel machines are generally considered to be more data efficient in modeling high-quality MLFFs,[2] but yield large linear optimization problems when many training samples and/or large molecule sizes are involved[23] (e.g., in materials[22,24,25] or biomolecules[26]). Linear systems are generally solved in closed-form, which has quadratic memory complexity. For large kernel systems, closed-form solutions are therefore not feasible, and it is necessary to invoke iterative solvers, which do not require to store the full optimization problem at once.[27−29] Alas, iterative solvers are highly sensitive to the numerical properties of the kernel matrix: strong correlations within the data yield ill-conditioned optimization problems that are hard to converge, because small changes in the model parameters lead to vastly different responses.[23] MLFFs are exposed to this issue due to predominantly stable atomic bonding patterns. This issue can also occur in models that incorporate differential constraints as inductive bias.[30] Preconditioning techniques aim to diminish the strongest linear dependencies, by presolving parts of the system using a numerically more stable algorithm and thereby separate the effects of strong correlations from the iterative solver. While many standard preconditioning approaches exist, effective solutions often rely on domain expertise.

In this study, we discuss the relevant theoretical and practical considerations necessary to develop the appropriate preconditioner for systems of many interacting atoms, in order to construct accurate MLFFs. An effective preconditioner has to represent all atomic cross-correlations faithfully, while being

Special Issue: Machine Learning for Molecular Simulation

Received: December 23, 2022

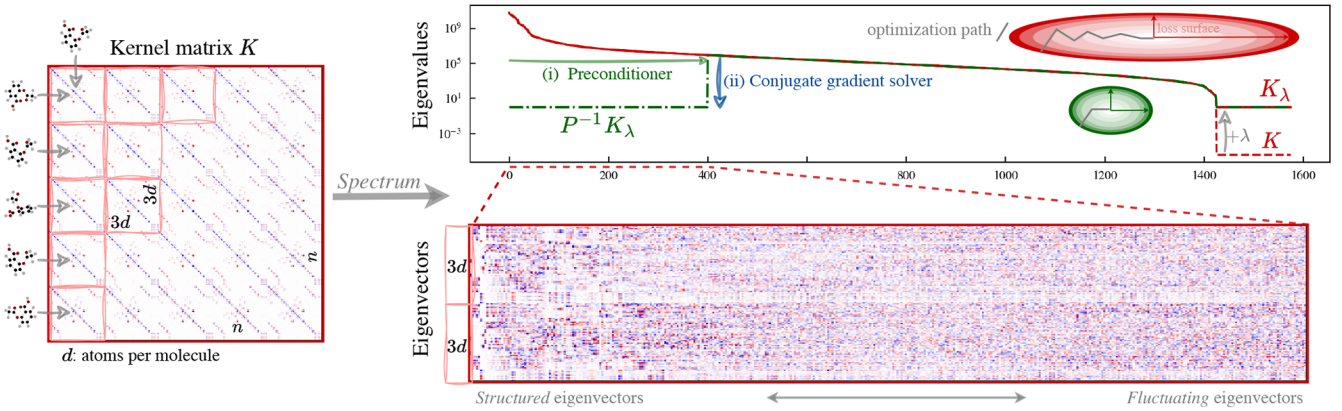


Figure 1. (Left) sGMDL kernel matrix for aspirin as a representative example for correlated molecular systems.[35] (Right) A preconditioned iterative solver first removes the dominant dimensions of the eigenspectrum and then approximates the remaining spectrum increasingly accurately via a #-dimensional Krylov subspace (# iteration steps). Large magnitude differences across eigenvalues lead to narrow valleys in the loss landscape (red ellipse), which obfuscates the optimization path. Preconditioning can remedy this by flattening the spectrum, leading to wider valleys (green roundish ellipse), which are easier to descent. To show the dominant spectral structure, we focus on the first entries (specifying the contributions of the first two training molecules) of the dominant ∼400 eigenvectors.

computationally cheap to construct. Basic preconditioning approaches, such as Jacobi or sparse preconditioning, remove potentially important entries (correlations) from the kernel matrix and can therefore not reliably describe complex many-body interactions. The alternative is to use low-rank approximations that compress, but retain all entries in the kernel matrix. The optimal solution would be to find the most relevant subspace spanned by the kernel matrix via SVD decomposition, which is however prohibitively expensive in practice. A more economical approach is to approximate the kernel matrix using a subset of its columns as inducing columns. This is the key idea behind the Nyström method[31] and the incomplete Cholesky decomposition,[32] which differ in the way how inducing columns are sampled. These two methods are representative for a broad class of inducing point methods[32−34] and offer opposing trade-offs with regard to construction cost and convergence speed.

We demonstrate in various numerical experiments how these different preconditioning approaches affect the spectrum of the kernel matrix and hence the convergence of iterative solvers. This allows us to derive a heuristic that helps practitioners to balance computational resources efficiently when reconstructing kernel-based MLFFs with iterative training algorithms.

2. SCALABLE KERNEL SOLVERS FOR QUANTUM CHEMISTRY

2.1. Kernel Machines for MLFF. Kernel machines offer a powerful way to model complex functions and have successfully been applied to reconstruct MLFFs.[14−22] Here, we focus on the symmetric gradient-domain machine learning (sGDML)[16] model, which incorporates all important equivariances and invariances of molecular FF to be particularly data efficient.[15] The atomic positions are encoded via a descriptor, to obtain a positive semidefinite (PSD) kernel matrix $K \in \mathbb{R}^{n\times n}$ with size $n = 3\ d\ n_{\text{train}}$ (n_{train} training molecules with d atoms). The resulting learning problem amounts to solving the regularized linear system

$$K_\lambda \cdot \boldsymbol{\alpha} = \boldsymbol{y} \tag{1}$$

with the kernel $K_\lambda = K + \lambda I_n$ and atomic force labels y. Using the solution $\boldsymbol{\alpha}$, force predictions for new input conformations are queried at linear computational cost in the number of training points n_{train}. This makes kernel-based MLFFs cheaper than deep neural network architectures with comparable accuracy in many cases.[23,36] For example, consider the number of parameters for models trained with 1k aspirin examples in ref 7: 500k (NewtonNet[12]) to 3M (SpookyNet,[11] NequIP[13]) in contrast to sGDML, which only used 63k parameters.

Equation 1 is typically solved analytically in closed-form, which requires storage of the full matrix at a memory complexity of $O(n^2)$ and cubic runtime cost $O(n^3)$. Following this approach, memory complexity is the bottleneck in kernel-based methods, which prevents scaling to large system sizes or numbers of training points. Iterative solvers can overcome these limitations by solving the optimization problem numerically, using gradient descent. Gradient descent only relies on evaluations of matrix−vector products[23,27,29,37−41] and thus remedies the memory constraint, as no matrix needs to be stored explicitly.[42,43] Since K is PSD, the linear system can be solved using the conjugate gradient (CG) algorithm, which is more efficient than plain gradient descent optimization.[38] The CG algorithm enforces conjugacy between optimization steps and thereby iteratively constructs a basis for the Krylov subspace of the kernel matrix with respect to the labels y. This leads to a rapid progression toward the solution $\boldsymbol{\alpha}$, with a theoretically guaranteed convergence in (at most) linear time $O(\#n^2)$, with $\# \leq n$. Depending on the spectral properties, i.e., the effective numerical rank of the kernel matrix, superlinear convergence in $\# \ll n$ is often achievable in practice.[44−46] In particular, the condition number (given by the ratio of largest to smallest singular value) and the eigenvalue decay rate are the decisive properties that determine the overall performance of this algorithm.

To illustrate this, we discuss these spectral properties for a specific example that is representative for the molecular data sets considered in this work (see Figure 1, left panel) and most correlated systems in general. The kernel matrix consists of ($3d \times 3d$) block matrices, which describe correlations between pairs of training molecules. Each block represents the correlation structure between the d atoms within each example molecule. The molecular structures are restricted by the laws of quantum chemistry, which yield similar spectral characteristics

https://doi.org/10.1021/acs.jctc.2c01304
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

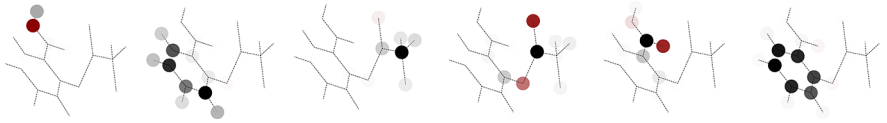


Figure 2. Atomic contributions per eigenvector (see Appendix A). The kernel matrix allows a low-rank decomposition in terms of prototype molecules.

across different systems. This enables us to derive transferable numerical insights about the kernel-based FF reconstruction problem. The eigenvectors and eigenvalues are shown in red (right panel of Figure 1). For clarity, we only show the dominant 400 eigenvectors (columns) and their entries corresponding to two molecule (rows). The dominant eigenvectors exhibit a periodic pattern, which can be interpreted as a small set of prototypical molecular block structure elements. These patterns represent atom-wise contributions by each eigenvector (see Appendix A for details), which we visualize in Figure 2. Intuitively, the structured eigenvectors form a low-rank basis for the full molecule, as they capture recurring substructures across all training points, e.g., strong correlations between interacting atoms. On the other hand, the remaining low-magnitude eigendirections represent high-frequency fluctuations from the prototypical geometries. These eigenvectors contribute the individual geometric features of the training data.[47] This part of the spectrum decreases less rapidly, compared to the dominant part. Lastly, the spectrum also includes a (regularized) null space, which is unavoidable, due to a roto-translationally invariant representation of the molecule in terms of pairwise distances.

2.2. Theoretical Preconditioning for MLFF. Large differences in magnitude across eigenvalues (large condition number) induce long and narrow multidimensional valleys along the loss surface. These narrow valleys hamper gradient descent solvers, since the optimization path bounces back and forth on the way to the solution (as illustrated by the ellipses in Figure 1). A suitable preconditioner is a full-rank matrix $P \approx K_\lambda$ that captures the dominant spectral components. It is used to normalize the matrix spectrum, which has the same effect as smoothening the underlying loss landscape.[40,48−53] With P, the linear system can equivalently be reformulated as

$$P^{-1}K_\lambda \cdot \boldsymbol{\alpha} = P^{-1}\boldsymbol{y} \tag{2}$$

which preserves the original solution $\boldsymbol{\alpha}$. After applying the preconditioner, the remaining spectrum is then iteratively approximated within the Krylov subspace generated by the CG algorithm (blue arrow in Figure 1). This two-step procedure can be regarded as a systematic low-rank decomposition of the kernel matrix (see Figure 3 (top row) and Appendix A).

To retain the original PSD structure, we require that a symmetric decomposition $P = LL^T$ ($L \in \mathbb{R}^{n\times n}$) exists.[23,48,50] Another key requirement is that the construction of the preconditioner P should not dominate the overall computational costs. Generally, there are two principled ways to achieve this:[54] either via sparsification of the kernel matrix (e.g., zero-out entries to obtain a block-diagonal form) or via a compression into a low-rank representation. Only the latter approach is able to faithfully represent the complex correlation structure of quantum chemical systems. In this work, we therefore focus on symmetric low-rank approximations $K \approx K_k = L_k L_k^T$ ($L_k \in \mathbb{R}^{n\times k}$, $k \ll n$), which are expanded into invertible full-rank preconditioners after the regularization term is added:

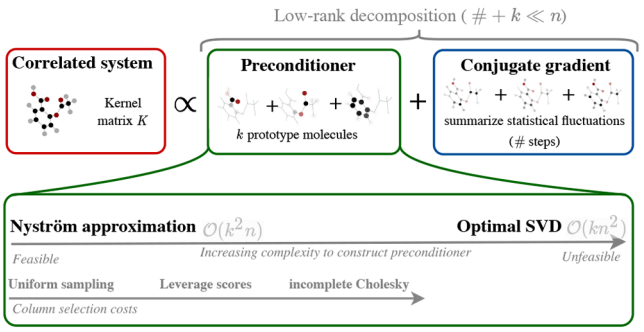


Figure 3. The preconditioned CG solver perform a numerically exact low-rank decomposition of the full kernel matrix K (see Appendix A for details). Ideally, the preconditioner accounts for the correlations (molecular prototypes) whereas the CG solver captures the remaining statistical fluctuations. When choosing the preconditioner, we have to balance the trade-off between numerical costs and accuracy with respect to capturing more correlations.

$$P = L_k L_k^T + \lambda I_n \tag{3}$$

The inverse P^{-1} can then be computed economically via the Woodbury formula,

$$P^{-1} = \lambda^{-1}[I_n - L_k(\lambda I_k + L_k^T L_k)^{-1}L_k^T] \tag{4}$$

under some numerical considerations.[23] Overall, the runtime of this approach is $O(k^2 n)$, and it can be implemented at a memory complexity of only $O(kn)$, by exploiting the symmetry of the matrix decomposition above.[31]

The optimal, but also most expensive, rank-k approximation K_k is given by the singular-value decomposition (SVD), which is identical to the eigenvalue decomposition in this PSD matrix case. A *SVD preconditioner* can directly remove the k dominant eigenvalues and hence optimally normalize the spectrum (green arrow in Figure 1). Since every eigenvector represents a unique linear combination of kernel columns, access to all columns is required to construct the SVD. Consequently, the computational cost of an SVD scales quadratically with kernel size, as $O(kn^2)$. This scaling is prohibitive for large-scale MLFF reconstruction tasks, and thus, SVD is only considered as a theoretical bound for the optimal performance of rank-k preconditioners in this work.

3. NYSTÖM-TYPE PRECONDITIONER

A low-rank approximation of the kernel matrix can also be constructed as a projection onto a *subset* of its columns. This basic idea is underlying the Nyström method, which is traditionally used to approximate large scale kernel machines[31,34,55−59] and recently also as a preconditioner.[50,60−62]

https://doi.org/10.1021/acs.jctc.2c01304
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

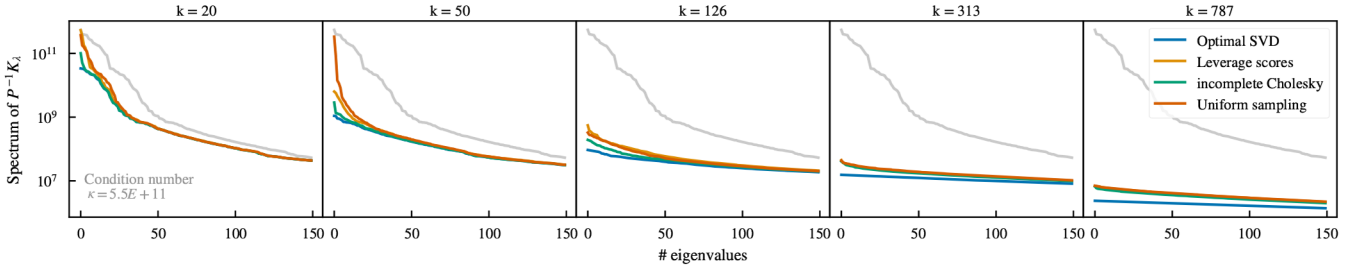


Figure 4. Preconditioned spectrum of $P^{-1}K_\lambda$ for varying preconditioner size k, restricted to the dominant 150 eigenvalues. Results for aspirin ($n_{\text{train}} = 250, n \approx 15\text{k}$); more molecules in Figure A2. The gray curve corresponds to the original spectrum of K_λ. All Nyström-type approaches are lower bounded by the optimal SVD preconditioner.

Given $k \ll n$ inducing columns $K_{\text{nk}} = \begin{pmatrix} K_{\text{kk}} \\ K_{(n-k)k} \end{pmatrix}$, the Nyström approximation represents the full kernel matrix according to

$$K \approx K_k = K_{nk} K_{kk}^{-1} K_{nk}^{\text{T}} = \begin{pmatrix} K_{kk} & K_{(n-k)k}^T \\ K_{(n-k)k} & K_{(n-k)k} K_{kk}^{-1} K_{(n-k)k}^T \end{pmatrix} \tag{5}$$

In this expression, the matrix dimensionality is implicitly encoded in the subscript, i.e., $K_{(n-k)k} \in \mathbb{R}^{(n-k)\times k}$. K_{kk} can be symmetrically factorized to obtain the form in eq 3 (see refs 23 and 63 for implementation details). The approximation error is given by the Schur complement $S = K/K_{kk} = K_{(n-k)(n-k)} - K_{(n-k)k} K_{kk}^{-1} K_{(n-k)k}^T$. Intuitively, all $(n - k)$ remaining columns are expressed in the basis of the inducing columns. Therefore, the accuracy is highly dependent on selecting the most representative set of inducing columns. We now set out to find a set of expressive inducing columns that represent the dominant part of the spectrum and thus capture reoccurring structural patterns in the data, as depicted in Figure 2.

3.1. Column Selection Strategies. *3.1.1. Uniform Sampling.* The simplest approach is to draw a random sample of columns.[31,33,64,65] This strategy is computationally inexpensive, but it essentially ignores the spectral properties of the kernel matrix and will therefore only serve as a baseline in our study.

3.1.2. Leverage Score Sampling. Not all columns contribute equally to the composition of the kernel matrix. To get a good approximation of the column space spanned by the kernel matrix, leverage scores can be used to measure how uniformly information is distributed among all columns.[34,66−68] Columns that are strongly correlated get assigned a low leverage score, while linearly independent columns mostly represent themselves and yield large leverage scores. Sampling according to leverage scores thus gives a higher chance of recovering a more representative set of inducing columns for the kernel matrix spectrum,[47,69] although linear independence is not guaranteed.

In our ridge regression setting, it is appropriate to consider *ridge leverage scores* $\tau_i(K) = (K(K + \lambda I_n)^{-1})_{ii}$,[69,70] which use a diagonal regularization parameter λ to diminish small components of the kernel spectrum. Here, each kernel column is projected onto the regularized kernel to determine its overlap with the columns of the regularized kernel matrix. Since evaluating this expression incurs the same computational cost as the original linear system, approximations are needed to make this computation feasible in practice.[23,69]

3.1.3. Incomplete Cholesky. Lastly, we discuss the *incomplete Cholesky*[27,29,63,71] decomposition as a deterministic column selection strategy. The algorithm constructs a low-rank representation $K_k = L_k L_k^T$, by iteratively selecting inducing columns based on the Schur complement $S \in \mathbb{R}^{(n-1)\times(n-1)}$.

In contrast to the two previous probabilistic approaches, the set of inducing columns is not selected at once, to avoid correlations. The Cholesky algorithm computes the residual Schur complement at each step and uses it as input for the next iteration, which then operates on the projected matrix with all previously selected inducing columns removed. In that sense, the Cholesky method uses successive Nyström approximations for each inducing column. In the first iteration, we have (after permutation $K_{n1} = \begin{pmatrix} K_{11} \\ \boldsymbol{b} \end{pmatrix}$):

$$S = K/K_{11} = K_{(n-1)(n-1)} - \frac{1}{K_{11}} \boldsymbol{b}\boldsymbol{b}^T \tag{6}$$

The largest diagonal element of S indicates the next inducing column, by which the approximation error is minimized according to the trace norm.[72] Since the trace is equal to the sum of all eigenvalues, this pivot rule focuses on the columns which are most representative for the remaining dominant spectral components in each Cholesky iteration. Despite being a greedy approach, the incomplete Cholesky systematically orthogonalizes correlated columns and thereby ensures linear independence, which enhances the representative power of the preconditioner.[63] This leads close to optimal accuracy (comparable to SVD) in practice and theoretically guaranteed exponential convergence for exponentially decaying eigenvalue spectra.[72] Note that if identical inducing columns are chosen for the incomplete Cholesky algorithm and the Nyström method, both methods are equivalent.[32,33] The incomplete Cholesky algorithm is as costly as the overall Nyström preconditioner $O(k^2 n)$[72] (i.e., it doubles the runtime costs). In contrast, the cost for the leverage scores estimation can be readily adapted by the fidelity of the approximation scheme.[65] While all three approaches are in the same complexity class $O(k^2 n)$, their computational prefactors are ordered according to (i) uniform sampling, (ii) leverage score sampling, and (iii) incomplete Cholesky, also see Figure 3 (bottom panel). We present an efficient incomplete Choleskyimplementation in Appendix A2.

https://doi.org/10.1021/acs.jctc.2c01304
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

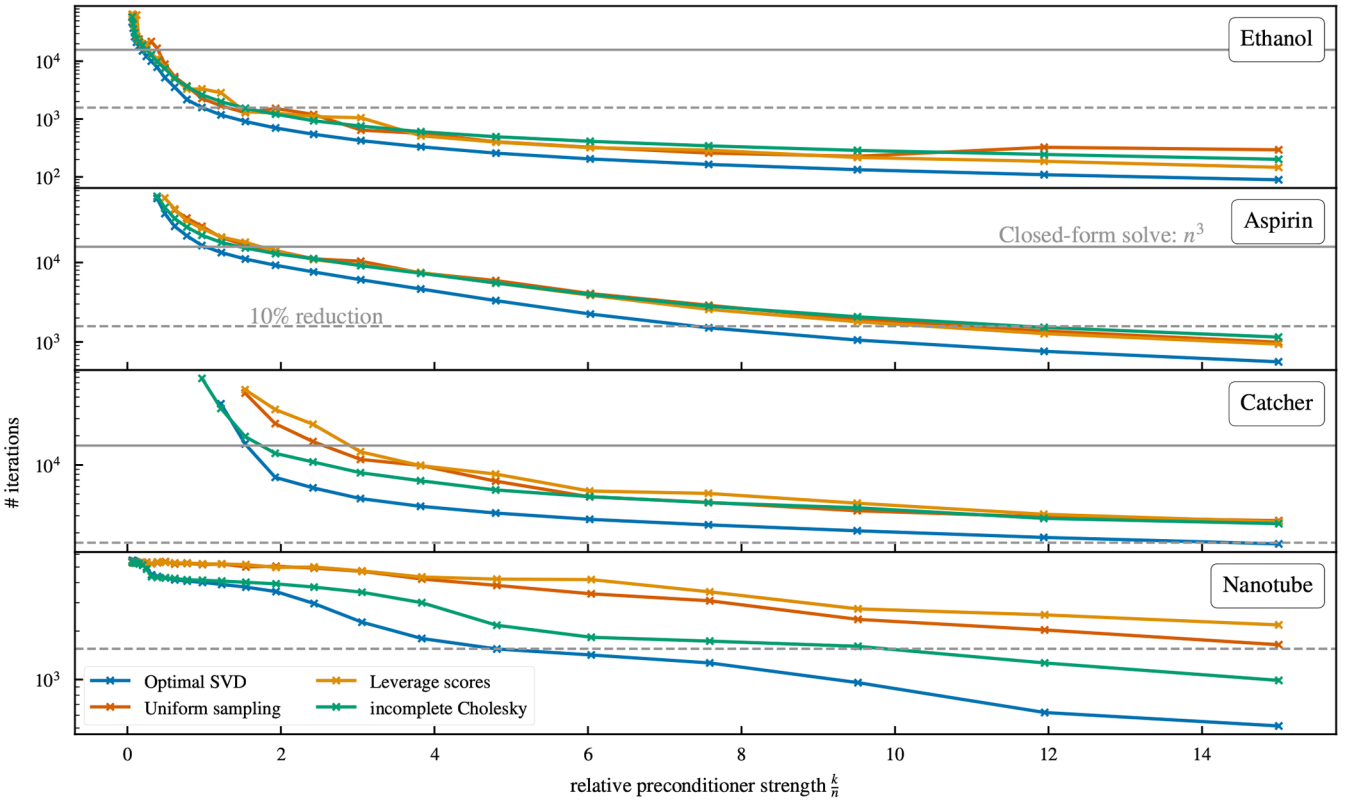


Figure 5. Iterative convergence speed for varying preconditioner strength for fixed kernel size $n \approx 15\text{k}$ ($n_{\text{train}} = 250$ for aspirin). Only minor performance differences are visible for ethanol. In contrast, for more complex molecules (catcher, nanotube), the elaborate incomplete Cholesky preconditioner is superior to probabilistic sampling of inducing columns.

4. EXPERIMENTAL RESULTS

We use the MD17 (plus azobenzene)[35] and MD22[23] benchmark data sets (available at www.sgdml.org) for our numerical experiments. Furthermore, we include a new catcher data set, which consists of the buckyball batcher data set in MD22, but without the fullerene (see Appendix B for details). We refer to these data sets in abbreviated form, by their molecule name, i.e., aspirin refers to the MD17-aspirin data set.

To establish a baseline, we first consider a small learning task ($n_{\text{train}} = 250$ for aspirin), for which a comparison with the computationally expensive, optimal SVD preconditioner is still possible within hardware limitations (Sections 4.1 and 4.2). In the following Section 4.3, we analyze the scaling behavior of all preconditioning approaches to large systems and derive a heuristic to find preconditioner sizes that perform well in practice. Throughout our numerical experiments, we use the sGDML kernel[16] with fixed length scale $\sigma = 10$ and regularization $\lambda = 10^{-10}$.

4.1. Preconditioned Spectrum. We begin by analyzing the eigenvalue spectrum of the preconditioned kernel matrix $P^{-1} K_\lambda$, since the convergence rate of the iterative solver is directly connected to its condition number.[44−46] Figure 4 shows the 150 dominant eigenvalues for aspirin for various numbers of inducing columns k. As expected from our theoretical considerations, SVD preconditioning is the most effective approach, leading to the smallest condition number, as compared to all Nyström-type preconditioning approaches considered in this work. It is worth noting that the optimal SVD preconditioner is never exceedingly better in this test (below 1 order of magnitude), independently of the number of inducing columns. This is a first reassuring confirmation that Nyström-type preconditioning is effective in this kind of application and that our preconditioner construction procedure is numerically robust.

For small numbers of inducing columns, the incomplete Cholesky preconditioner shows a significantly better performance, compared to the two more basic probabilistic column sampling approaches. The incomplete Cholesky decomposition removes the contributions of already sampled columns via the Schur complement and is therefore able to construct a more representative approximation of the spectrum using the same number of inducing columns. In contrast, the inducing columns returned by the uniform or leverage score sampling approaches are potentially correlated, which can limit their effectiveness. As expected, the inducing columns generated by the incomplete Cholesky decomposition are not guaranteed to match the most important eigendirections, which potentially makes it less effective than a SVD, yet cheaper to construct.

Since the dominant eigenvectors are partially aligned with the kernel columns in this experiment, the incomplete Cholesky preconditioner performs nearly optimally for a small number of inducing columns (up to $k \approx 3d$ in Figure 4). For larger preconditioners, this advantage diminishes. Here, the corresponding eigenvectors are not structured and do not significantly align with individual kernel columns anymore (see lower right panel in Figure 1). Hence, we observe a constant performance gap with respect to the optimal SVD preconditioner with growing k. In this scenario, the particular column selection is less relevant, and the preconditioner size k alone determines the approximation quality, despite a vastly varying

https://doi.org/10.1021/acs.jctc.2c01304
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

construction cost between the different preconditioning approaches. We observe a similar pattern for other molecular data sets in Figure A2. For the sake of completeness, we also demonstrate the limitations of local Jacobi preconditioning in the same experimental setup (see refs 73−77 and Figure A1).

4.2. Convergence Speed of Conjugate Gradient. Next, we quantify the iterative solver runtime by tracking the number of CG steps # until convergence, as this performance metric is insensitive to implementation and hardware details. Figure 5 shows the relationship between convergence speed and relative preconditioner strength k/n. We consider molecules of different sizes, including ethanol (d = 9), aspirin (d = 21), a catcher molecule (d = 88, see Appendix B), and a large nanotube (d = 370). Additional systems are shown in Figure A3. The preconditioner strength is varied between $k \sim 50$ up to 14% of all columns.

The gray solid horizontal line corresponds to the cubic costs of the closed-form solution, and the dashed line gives a visual cue for a $10\% \cdot n^3$ reduction, to provide a scale for relative computational complexity. For example, ethanol is simpler (easier to solve), compared to larger or more complex molecules, such as aspirin. We observe that the functional dependency between preconditioner strength and computational cost differs between all molecules. In particular, the number of inducing columns necessary to achieve superlinear convergence depends on the complexity of the molecule. These different complexities are embodied within the characteristic shape of dominant part of the spectrum (see Figure A2 for explicit spectra). For example, the catcher molecule requires the most inducing columns, since its spectrum contains many nearly degenerate, dominant eigenvalues. Similarly, it is difficult to accelerate convergence for the nanotube data set, since the dominant part of the spectrum is only weakly decaying. In contrast, ethanol is simple to solve, and more than 2% preconditioning already leads to rapid convergence (1 order of magnitude faster). Generally, undersized preconditioners lead to more CG steps, which incur a computational cost that is similar to that of analytic closed-form solutions. In constrast, more inducing columns generally lead to faster convergence, but at increased preconditioner construction cost (see Section 4.3 for practical guidance on this trade-off).

Having discussed commonalities between all approaches, we now analyze their differences with respect to their applicability and numerical performance when applied to different molecules. From Figure 5, it is clear that the SVD preconditioner is the most effective approach. As expected and already observed in Section 4.1, it lower bounds all other Nyström-type preconditioners. We emphasize again that the SVD preconditioner is too expensive in practice and only used here as the theoretical optimum. The performance gap between the optimal SVD preconditioner and all three Nyström-type preconditioners is largely determined by how well the inducing columns align with the eigenvectors. All three Nyström-type preconditioners perform similarly for most molecules, e.g., consider ethanol in Figure 5, for which barely any difference is visible. This behavior is similar for to all additional molecules presented in Figure A3. Note that the structural changes within the eigenvectors when applying larger preconditioners also degrades the differences observed for the weakly preconditioned aspirin molecule (e.g., for $k < 100$ in Figure 4).

This indicates that a uniform subset of columns is sufficiently expressive for the kernel spectrum, and therefore, no significantly better set of inducing columns can be found to further improve preconditioner performance. However, both probabilistic column selection approaches introduce fluctuation into the preconditioner performance, because any potential correlations between inducing columns are disregarded (i.e., they can draw a *bad* set of inducing columns by chance). In contrast, the incomplete Cholesky performance is monotonously improving and does not fluctuate. This is reminiscent of the deterministic pivoting rule, which systematically improves the approximation error via repeated rank-1 updates at each iteration.

Further, we observe noteworthy differences for the larger and more structured catcher and nanotube molecules (two bottom panels in Figure 5). Here, the incomplete Cholesky preconditioner significantly outperforms both probabilistic column sampling approaches. For small preconditioner sizes, the incomplete Choleksy's performance is close the optimal SVD baseline. For the catcher molecule, this advantage diminishes, and for larger preconditioners, all Nyström-type approaches perform equivalently. For the nanotube molecule, there are persistent differences between all preconditioner approaches. Here, the performance gap between the SVD preconditioner and both probabilistic approaches increases with the number of inducing columns. This indicates that both approaches do not construct an expressive set of columns. We attribute this to the symmetric, repetitive structure of the large nanotube, which creates many correlated (i.e., redundant) kernel columns. As outlined in Section 3, such a scenario is problematic for probabilistic sampling, as it inevitably incorporates correlated inducing columns.[78] Here, the systematic selection of independent inducing columns by the incomplete Cholesky approach appears to play a crucial role to solely incorporate the relevant molecular structure.

Lastly, we note that the leverage score sampling approach does not appear to significantly outperform the uniform sampling baseline, even for small preconditioners (~2% or $k \sim 300$) (see Figure 5). This is somewhat counterintuitive as leverage scores are intended to improve on basic uniform sampling. We speculate that this is related to the fact that the regularization λ for the ridge leverage scores is fixed and not adapted to changing number of inducing columns k. Hence, ridge leverage scores are indicative for the spectral properties of the kernel matrix (K_λ), instead of directly targeting the dominant k-dimensional eigenspace (K_k), which is relevant for preconditioning. We experimentally investigate this intuition in Figure A3, via an improved (however computationally costly) notion of leverage scores. These improved leverages scores indeed remove the gap with respect to the uniform sampling baseline. However, they do not improve the preconditioner performance significantly. Therefore, we conclude that the incomplete Cholesky preconditioner is the most capable approach, as it is highly effective in selecting a representative set of inducing columns. This leads to nearly optimal preconditioning for a large, structured nanotube molecule. However, this relative advantage generally degrades with increasing number of inducing columns. For larger preconditioners, it is sufficient to uniformly sample columns, since small eigendirections are less structured as they obtain support from most atoms simultaneously. To arrive at the right preconditioner choice, we additionally need to account for the different computational costs associated with each preconditioner (see

https://doi.org/10.1021/acs.jctc.2c01304
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

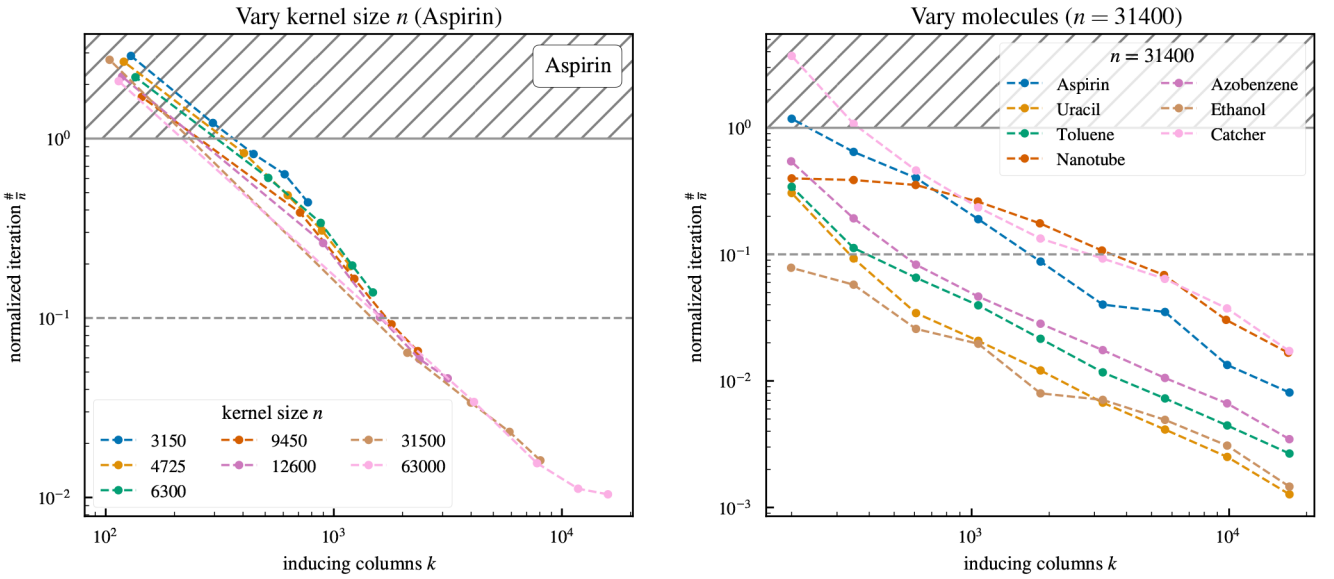


Figure 6. The superlinear convergence ($\#/n < 1$) of iterative solvers is related to the preconditioning strength k via a monomial power law (linear dependence in the log−log space). For increasing kernel size n, relatively less columns k are needed to obtain an equivalent superlinear advantage. Hence, preconditioning is increasingly valuable (left panel). The monomial trade-off proportionality is verified for any molecule but the exact relation depends on the molecule type (right panel).

also Section 3). Since the probabilistic preconditioners are generally cheaper, in many situations (e.g., large preconditioners or unstructured molecules such ethanol/uracil) random sampling should be preferred over the more capable but also more costly incomplete Cholesky approach.

4.3. Optimal Number of Inducing Columns. Next, we investigate the iterative convergence behavior, when scaling from smaller to larger kernel systems. To this end, we consider the normalized iteration $\#/n$, which measures the relative advantage of the preconditioned iterative solver compared to the closed-form solution. A normalized iteration of one indicates equivalent runtimes between both methods. For simplicity, we restrict this analysis to leverage score-based preconditioning in the following, which lower bounds the performance of all other Nyström-type preconditioners. Here, the lower bound refers to the worst performance (a better preconditioner should result in smaller k for the optimal trade-off).

In Figure 6 (left panel), we consider the aspirin data set and vary the absolute kernel size n. Interestingly, we find that the convergence does not depend on the kernel size n. It is solely determined by the absolute number of inducing columns k. We observe a trade-off between iterative solver runtime and memory demand, i.e., for doubling the preconditioner the solver runtime halves, as can be seen by the linear relation in log−log space $\#/n \propto k^{-m}$ (see Figure 6).

Next, in Figure 6 (right panel), we analyze this trade-off for different molecules and fixed kernel size n = 31k. We observe that the same power-law relation persists for all molecules. However, offsets and slopes m now depend on the molecule type, as expected based on their different spectra (see Appendix B1). We characterize the offset by the minimal preconditioner size $k_{\min}$, which indicates parity between iterative and closed-form solver runtime (intersection with solid line). As a consequence, more complex molecules require a larger minimal preconditioner size $k_{\min}$ (see Table 1).

Table 1. Rule of Thumb Hyperparameters (m and $k_{\min}$)[a]

	d	m	$k_{\min}$
Default	—	1	100
Ethanol	9	0.87	10
Uracil	12	1.07	32
Toluene	15	1.01	44
Aspirin	21	1.14	236
Azobenzene	24	1.02	62
Catcher	88	1.02	316
Nanotube	370	0.73	89

[a]Reusing experiments from Figure 6. with a kernel size of n = 31k. Here, d is the number of atoms per molecule; m represents the reduction in normalized iteration $\#/n$ associated with an increasing preconditioner size k. The minimal preconditioner size $k_{\min}$ indicates equal runtimes between iterative and closed-form solve. The molecular conformation are taken from the MD17[35] and MD22[23] data sets.

4.3.1. Deriving a Rule of Thumb. We now discuss the subtleties of choosing the ideal number of inducing columns k, to minimize the overall runtime of the solver. Here, we need to balance two aspects: Too few inducing columns lead to exceedingly long (effectively nonconvergent) CG runtimes. Second, too many inducing columns are prohibitively costly due to the quadratic scaling in k. Hence, the optimal preconditioner size k scales with the kernel size n, which means that using a fixed absolute (small or large) preconditioner size is clearly not a viable strategy. A simple heuristic is to use a fixed percentage of columns for preconditioning (e.g., 1%-baseline, $k = \frac{n}{100}$).[23]

We derive a more elaborate heuristic based on the overall computational costs of the preconditioned linear solver. These are given by the preconditioner costs $O(k^2 n)$ plus the estimated iterative solver runtime (as formalized by the monomial relation), i.e.,

https://doi.org/10.1021/acs.jctc.2c01304
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

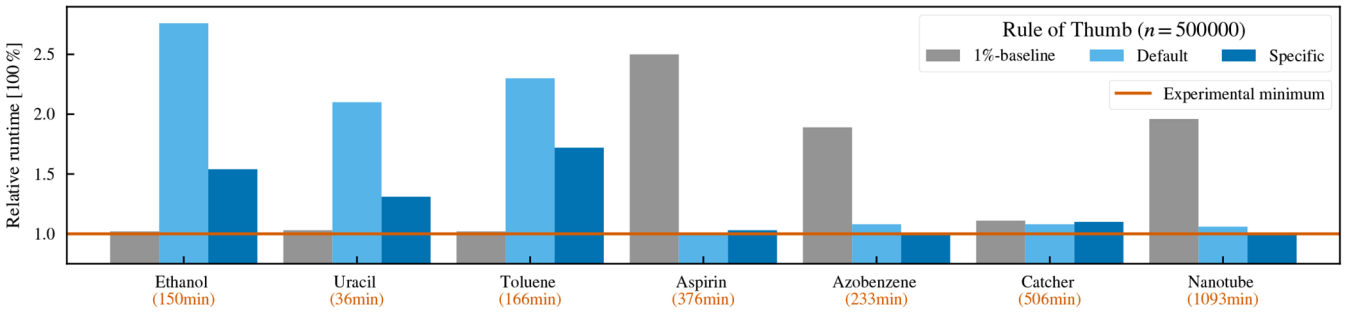


Figure 7. Minimizing the overall runtime by estimating the ideal number of inducing columns k in advance. All runtimes are given as relative percentage of experimental optimal runtime (orange), which is stated in parentheses. Fixed kernel size n = 500k (see Figure A5 for smaller kernel sizes). The rule of thumb performs best for complex and difficult to converge molecules, e.g., catcher or nanotube. For simpler and fast converging molecules (ethanol or uracil), a molecule specific rule of thumb can reduce the gap with respect to the 1%-baseline estimate ($k = \frac{n}{100}$).

Table 2. Comparing Accuracy and Runtime between the Analytic (Closed-Form) and Preconditioned CG (PCG) Solver[a]

	Accuracy	Runtime [min]		
Molecule	$\lvert MSE_{PCG} - MSE_{Analytic} \rvert$	Analytic	PCG	Speed up
Ethanol	0.00003	38 ± 4	10.5 ± 1.1	3.6×
Uracil	0.00007	33 ± 6	6.5 ± 0.3	5.0×
Toluene	0.00002	41 ± 4	11.2 ± 0.8	3.6×
Aspirin	0.00006	35 ± 4	25.3 ± 1.6	1.4×
Azobenzene	0.00012	35 ± 4	12.0 ± 0.9	2.9×
Catcher	0.00009	50 ± 5	50.4 ± 2.4	1.0×
Nanotube	0.00002	151 ± 17	70 ± 5	2.2×

[a]All experiments were performed on a NVIDIA A100 (40GB) a fixed kernel size n = 63k. All runtimes are averaged across seven runs. Stating the mean-absolute error (MAE) of force prediction in mol kcal^{-1} Å^{-1}. The PCG performs indistinguishably at faster runtimes and lower memory demand.

$$\text{runtime}[k] \propto n^3\left[\left(\frac{k_{\min}}{k}\right)^m + \left(\frac{k}{n}\right)^2\right] \tag{7}$$

Note that we recover the original cubical runtime for $k \to k_{\min}$ and $k \to n$. We then obtain a rule of thumb (RoT) to *predict* the ideal number of columns k^{RoT} for preconditioning via minimizing the overall cost:

$$k^{\text{RoT}} = \min_{k\in[1,n]}\{\text{runtime}[k]\} = \left(\frac{(k_{\min})^m m n^2}{2}\right)^{1/2+m} \tag{8}$$

As discussed in the beginning of this section, both hyperparameters ($k_{\min}$ and m) are connected to the spectral properties of the kernel matrix. Thus, they can be determined for smaller systems and later reused for larger systems. In Table 1, we provide estimates for various molecules from the MD17 and MD22 data sets.[23,35] We observe that the power-law trade-off coefficient m is consistently close to unity, whereas the minimal preconditioner size $k_{\min}$ is associated with the size of the molecule and its complexity (dominant spectral properties). We define default hyperparameters (m = 1, $k_{\min}$ = 100) and distinguish between a default vs a (molecule)-specific RoT. Here, the default RoT is directly applicable to new molecules/data sets, whereas the specific RoT showcases the potential improvements via pretraining the hyperparameters.

4.3.2. Experimental Validation. We validate our RoT by measuring the overall runtime depending on the number of inducing columns for various molecules. Thereby, we obtain empirical estimates for the optimal number of inducing columns and the corresponding minimal runtime. We linearly interpolate all measurements to obtain runtime estimates for the default/specific RoT.

In Figure 7, we show the relative runtime between the RoT prediction and the empirical minimal runtime. Additionally, we also compare to the simple 1%-baseline (i.e., $k = \frac{n}{100}$).[23] Interestingly, there are two distinct behaviors across our selection of data sets. First, the small molecules (ethanol, uracil, and toluene) for which the 1%-baseline outperforms our RoT. Second, for larger and more complex molecules (aspirin, azobenzene, catcher, and nanotube), our RoT is better than the 1%-baseline. Here, the 1%-baseline underestimates the preconditioner strength, which leads to significantly longer (potentially nonconvergent) runtimes (plus 18 h for the nanotube). Additionally, the RoT can be targeted to a specific molecule type. This further improves the performance of the specific RoT, as compared to its default counterpart, see Figure 7. In particular, for all simpler molecules, the gap between 1%-baseline and specific RoT is reduced. Simultaneously, it retains the high performance for the more complex molecules.

All findings are further verified by additional experiments for two smaller kernel sizes (n = 75k and n = 158k) in Figure A5. Here, the specific RoT is even closer to the experimental minimum, since its molecule-specific parameters (Table 1) were estimated using a similar system size (n = 31k). This indicates that the specific RoT can be further improved through measuring the hyperparameters for a larger system, i.e., by reducing the gap between estimate and inference kernel size. However, obtaining these more accurate molecule-specific hyperparameters increases the computational overhead.

4.3.3. Remarks on Runtime and Predictive Accuracy. So far, we have focused on algorithmic complexities, iterations

https://doi.org/10.1021/acs.jctc.2c01304
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

times, and relative solver runtimes in our theoretical considerations and experimental validation. We now investigate how transferable these consideration are to the overall solver, as this is what practitioners ultimately are interested in. To this end, we state the overall solver runtime (preconditioner construction + iterative solve) in the molecule label in orange in Figure 7. For example, the training runtime of uracil is 1 order of magnitude lower in comparison to the nanotube. This roughly agrees with their relative difference (based on CG iteration steps) in Figure 6 (right panel). In general, the computational ordering deduced from the number of iteration steps # in Figure 6 (right panel) aligns with the experimentally measured training runtimes (orange labels in Figure 7 and Table 2). This validates our previous analysis, which was mostly based on the iteration time. For example, consider uracil and ethanol, which are consistently the simplest molecules to solve. Alternatively, the most expensive molecules are the catcher and nanotube molecules. Note the experimental nanotube runtime is twice as long as for the catcher molecule, which is, judging from their similar iterative runtimes in Figure 6, somewhat unexpected. This effect is closely related to the costs of evaluating the kernel forward pass. Since the sGDML kernel integrates over all relevant symmetries,[16] more symmetries increase the runtime complexity (28 [nanotube] > 4 [catcher]). This is also apparent when comparing the longer runtimes of ethanol (6 symmetries) with the faster training time of uracil (no symmetry).

Lastly, we compare the experimental runtimes and accuracies between a standard analytic (closed-form) solver and the preconditioned CG solver using the ideal number of inducing columns as predicted by our specific RoT. The results in Table 2 show no significant differences in the accuracy of the force prediction. However, the iterative solver is up to 5 times faster compared to its analytic (closed-form) counterpart, and it uses significantly less memory. We emphasize that the kernel size was limited to n = 63k (matching 1k aspirin points) in our experiments, constrained by the analytic solver, which needs access to the complete kernel matrix (∼30 GB). In contrast, the preconditioned CG only requires to store a smaller preconditioner (∼3 GB) and thus allows one to scale to much larger problem sizes with the same hardware (as done in Figure 7). Note that the cost for computing the preconditioner is negligible for a small number of inducing columns, and hence, preconditioned iterative solvers are effectively always superior to closed-form solvers for large systems.[27]

Overall, we conclude that our proposed heuristic provides an accurate and robust estimate for a practically well-performing number of inducing columns. This circumvents unnecessary large or too small preconditioners and can thus substantially speed up training. Our default RoT provides reasonable estimates irrespective of molecule type and kernel size. Moreover, we have demonstrated that our approach can be targeted to a specific molecule type and is therefore extendable to new molecules and data sets.

5. CONCLUSION

To make large-scale kernel learning for MLFF reconstruction more widely accessible, this study reviews the combination of Nyström-type preconditioners and the iterative CG solver for model training. This approach is motivated by the following observation: Strong correlations between atoms give rise to exceedingly ill-conditioned loss surfaces, which can be readily addressed by selecting the appropriate inducing columns for a preconditioning step. The remaining stochasticity of the learning problem can then be effectively targeted with the Krylov subspace spanned by the iterative CG solver. In combination, both steps can be viewed as a low-rank expansion of the kernel matrix, which also allows for a chemically interpretation in terms of molecular fragments.

We have demonstrated, how to construct preconditioners that enable superlinear convergence of the CG solver. This represents a significant speed-up over traditional closed-form approaches, in addition to the removal of their high memory demand that we originally set out to alleviate. Remarkably, the effectiveness of the preconditioner scales with the number inducing columns k and is largely independent of the kernel size n. This favorable scaling behavior is also a consequence of the aforementioned reoccurring correlation patterns between atoms that stay largely the same across training points.

We have furthermore discussed important considerations when sampling inducing columns. For simpler molecules, it suffices to sample uniformly, due to the strong correlation between individual training points (molecular geometries). In contrast, more involved physical systems require a careful selection of inducing columns, as the relevant information is unevenly distributed. In that setting, basic sampling approaches are not sufficient as they lack the essential physical ingredients. This was the case in our catcher or nanotube examples, where the incomplete Cholesky decomposition was considerably more effective in finding a representative set of inducing columns, compared to the other approaches (except SVD).

Overall, we have shown how physical insights about the problem at hand can be leveraged to build a more efficient training algorithm. Moving forward, our findings can be combined with tools from automatic differentiation,[30] which allow one to readily adapt models to new applications. This paves the way toward more challenging domains, such as biomedical molecules or materials, where it is even more instrumental to appropriately make use of powerful inductive biases.

Furthermore, our findings can be applied in the context of other recent advances in the field, such as the construction of more efficient descriptors[79] or self-attention based learning.[7] Both approaches allow global interactions, which can be potentially costly with a growing number of interacting atoms. Here, our understanding can help to mitigate some of the intrinsic scalability limitations toward larger system sizes.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jctc.2c01304.

Additional technical details and experimental results for more molecules (PDF)

■ AUTHOR INFORMATION

Corresponding Authors

Klaus-Robert Müller − *BIFOLD−Berlin Institue for the Foundations of Learning and Data, 10587 Berlin, Germany; Technische Universität Berlin, Machine Learning Group, 10587 Berlin, Germany; Department of Artificial Intelligence, Korea University, Seoul 136-713, Korea; Max Planck Institute for Informatics, 66123 Saarbrücken, Germany; Google Research, Brain Team, 10117 Berlin, Germany;*

https://doi.org/10.1021/acs.jctc.2c01304
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

orcid.org/0000-0002-3861-7685; Email: klaus-robert.mueller@tu-berlin.de

Stefan Chmiela − *BIFOLD−Berlin Institue for the Foundations of Learning and Data, 10587 Berlin, Germany; Technische Universität Berlin, Machine Learning Group, 10587 Berlin, Germany;* orcid.org/0000-0003-0892-952X; Email: stefan@chmiela.com

Author

Stefan Blücher − *BIFOLD−Berlin Institue for the Foundations of Learning and Data, 10587 Berlin, Germany; Technische Universität Berlin, Machine Learning Group, 10587 Berlin, Germany;* orcid.org/0000-0002-6330-7996

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jctc.2c01304

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

S.B., K.-R.M., and S.C. acknowledge support by the German Federal Ministry of Education and Research (BMBF) for BIFOLD (01IS18037A). K.-R.M. was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the government (MSIT) (No. 2019-0-00079, Artificial Intelligence Graduate School Program, Korea University and No. 2022-0-00984, Development of Artificial Intelligence Technology for Personalized Plug-and-Play Explanation and Verification of Explanation) and by the German Federal Ministry for Education and Research (BMBF) under Grants 01IS14013B-E and 01GQ1115. We thank Valentin Vassilev-Galindo for generating the catcher molecule dataset.

■ REFERENCES

(1) Noé, F.; Tkatchenko, A.; Müller, K.-R.; Clementi, C. Machine learning for molecular simulation. *Annu. Rev. Phys. Chem.* **2020**, *71*, 361−390.

(2) Unke, O. T.; Chmiela, S.; Sauceda, H. E.; Gastegger, M.; Poltavsky, I.; Schutt, K. T.; Tkatchenko, A.; Muller, K.-R. Machine Learning Force Fields. *Chem. Rev.* **2021**, *121*, 10142−10186.

(3) Keith, J. A.; Vassilev-Galindo, V.; Cheng, B.; Chmiela, S.; Gastegger, M.; Müller, K.-R.; Tkatchenko, A. Combining machine learning and computational chemistry for predictive insights into chemical systems. *Chem. Rev.* **2021**, *121*, 9816−9872.

(4) Pinheiro, M.; Ge, F.; Ferré, N.; Dral, P. O.; Barbatti, M. Choosing the right molecular machine learning potential. *Chem. Sci.* **2021**, *12*, 14396−14413.

(5) Behler, J.; Parrinello, M. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.* **2007**, *98*, 146401.

(6) Behler, J. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *J. Chem. Phys.* **2011**, *134*, 074106.

(7) Frank, J. T.; Unke, O. T.; Müller, K.-R. So3krates: Equivariant attention for interactions on arbitrary length-scales in molecular systems. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, 2022

(8) Lubbers, N.; Smith, J. S.; Barros, K. Hierarchical modeling of molecular energies using a deep neural network. *J. Chem. Phys.* **2018**, *148*, 241715.

(9) Schütt, K. T.; Arbabzadah, F.; Chmiela, S.; Müller, K. R.; Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. *Nat. Commun.* **2017**, *8*, 13890.

(10) Unke, O. T.; Meuwly, M. PhysNet: A neural network for predicting energies, forces, dipole moments, and partial charges. *J. Chem. Theory Comput.* **2019**, *15*, 3678−3693.

(11) 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.

(12) Haghighatlari, M.; Li, J.; Guan, X.; Zhang, O.; Das, A.; Stein, C. J.; Heidar-Zadeh, F.; Liu, M.; Head-Gordon, M.; Bertels, L.; et al. Newtonnet: A newtonian message passing network for deep learning of interatomic potentials and forces. *Digital Discovery* **2022**, *1*, 333−343.

(13) Batzner, S.; Musaelian, A.; Sun, L.; Geiger, M.; Mailoa, J. P.; Kornbluth, M.; Molinari, N.; Smidt, T. E.; Kozinsky, B. E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nat. Commun.* **2022**, *13*, 2453.

(14) Rupp, M.; Tkatchenko, A.; Müller, K.-R.; Von Lilienfeld, O. A. Fast and accurate modeling of molecular atomization energies with machine learning. *Phys. Rev. Lett.* **2012**, *108*, 058301.

(15) Chmiela, S.; Sauceda, H. E.; Müller, K.-R.; Tkatchenko, A. Towards exact molecular dynamics simulations with machine-learned force fields. *Nat. Commun.* **2018**, *9*, 3887.

(16) Chmiela, S.; Sauceda, H. E.; Poltavsky, I.; Müller, K.-R.; Tkatchenko, A. sGDML: Constructing accurate and data efficient molecular force fields using machine learning. *Comput. Phys. Commun.* **2019**, *240*, 38−45.

(17) Christensen, A. S.; Bratholm, L. A.; Faber, F. A.; Anatole von Lilienfeld, O. FCHL revisited: Faster and more accurate quantum machine learning. *J. Chem. Phys.* **2020**, *152*, 044107.

(18) Deringer, V. L.; Bartók, A. P.; Bernstein, N.; Wilkins, D. M.; Ceriotti, M.; Csányi, G. Gaussian process regression for materials and molecules. *Chem. Rev.* **2021**, *121*, 10073−10141.

(19) Faber, F. A.; Christensen, A. S.; Huang, B.; Von Lilienfeld, O. A. Alchemical and structural distribution based representation for universal quantum machine learning. *J. Chem. Phys.* **2018**, *148*, 241717.

(20) Glielmo, A.; Sollich, P.; De Vita, A. Accurate interatomic force fields via machine learning with covariant kernels. *Phys. Rev. B* **2017**, *95*, 214302.

(21) Li, H.; Zhou, M.; Sebastian, J.; Wu, J.; Gu, M. Efficient force field and energy emulation through partition of permutationally equivalent atoms. *J. Chem. Phys.* **2022**, *156*, 184304.

(22) Sauceda, H. E.; Gálvez-González, L. E.; Chmiela, S.; Paz-Borbón, L. O.; Müller, K.-R.; Tkatchenko, A. BIGDML—Towards accurate quantum machine learning force fields for materials. *Nat. Commun.* **2022**, *13*, 3733.

(23) Chmiela, S.; Vassilev-Galindo, V.; Unke, O. T.; Kabylda, A.; Sauceda, H. E.; Tkatchenko, A.; Müller, K.-R. Accurate global machine learning force fields for molecules with hundreds of atoms. *Sci. Adv.* **2023**, *9*, eadf0873.

(24) Chen, C.; Deng, Z.; Tran, R.; Tang, H.; Chu, I.-H.; Ong, S. P. Accurate force field for molybdenum by machine learning large materials data. *Phys. Rev. Mater.* **2017**, *1*, 043603.

(25) Zhang, Y.; Ling, C. A strategy to apply machine learning to small datasets in materials science. *npj Comput. Mater.* **2018**, *4*, 25.

(26) Unke, O. T.; Stöhr, M.; Ganscha, S.; Unterthiner, T.; Maennel, H.; Kashubin, S.; Ahlin, D.; Gastegger, M.; Sandonas, L. M.; Tkatchenko, A. et al. Accurate Machine Learned Quantum-Mechanical Force Fields for Biomolecular Simulations. *arXiv Preprint*, arXiv:2205.08306, 2022.

(27) Gardner, J.; Pleiss, G.; Weinberger, K. Q.; Bindel, D.; Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018.

(28) Liu, H.; Ong, Y.-S.; Shen, X.; Cai, J. When Gaussian process meets big data: A review of scalable GPs. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 4405−4423.

(29) Wang, K.; Pleiss, G.; Gardner, J.; Tyree, S.; Weinberger, K. Q.; Wilson, A. G. Exact Gaussian processes on a million data points. In

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), 2019.
(30) Schmitz, N. F.; Müller, K.-R.; Chmiela, S. Algorithmic Differentiation for Automated Modeling of Machine Learned Force Fields. *J. Phys. Chem. Lett.* **2022**, *13*, 10183−10189.
(31) Williams, C.; Seeger, M. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13 (NIPS 2000)*, 2000.
(32) Patel, R.; Goldstein, T.; Dyer, E.; Mirhoseini, A.; Baraniuk, R. Deterministic column sampling for low-rank matrix approximation: Nyström vs. incomplete Cholesky decomposition. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, 2016; pp 594−602.
(33) Kumar, S.; Mohri, M.; Talwalkar, A. Sampling techniques for the Nystrom method. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, 2009; pp 304−311.
(34) Musco, C.; Musco, C. Recursive sampling for the nystrom method. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017/
(35) Chmiela, S.; Tkatchenko, A.; Sauceda, H. E.; Poltavsky, I.; Schütt, K. T.; Müller, K.-R. Machine learning of accurate energy-conserving molecular force fields. *Sci. Adv.* **2017**, *3*, e1603015.
(36) 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*, 044120.
(37) Freitas, N.; Wang, Y.; Mahdaviani, M.; Lang, D. Fast Krylov methods for N-body learning. In *Advances in Neural Information Processing Systems 18 (NIPS 2005)*, 2005.
(38) Hestenes, M. R.; Stiefel, E. Methods of Conjugate Gradients for Solving Linear Systems. *Res. Natl. Inst. Stand. Technol.* **1952**, *49*, 409.
(39) Kim, K. I.; Franz, M. O.; Schölkopf, B. Iterative kernel principal component analysis for image modeling. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1351−1366.
(40) Srinivasan, B. V.; Hu, Q.; Gumerov, N. A.; Murtugudde, R.; Duraiswami, R. Preconditioned Krylov solvers for kernel regression. *arXiv Preprint*, arXiv:1408.1237, 2014.
(41) Rudi, A.; Carratino, L.; Rosasco, L. Falkon: An optimal large scale kernel method. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017.
(42) Hackbusch, W. *Iterative Solution of Large Sparse Systems of Equations*; Springer, 1994; Vol. *95*.
(43) Saad, Y. *Iterative Methods for Sparse Linear Systems*; SIAM, 2003.
(44) Van der Sluis, A.; van der Vorst, H. A. The rate of convergence of conjugate gradients. *Numer. Math.* **1986**, *48*, 543−560.
(45) Axelsson, O.; Kaporin, I. On the sublinear and superlinear rate of convergence of conjugate gradient methods. *Numer. Algorithms* **2000**, *25*, 1−22.
(46) Beckermann, B.; Kuijlaars, A. B. Superlinear convergence of conjugate gradients. *SIAM J. Numer. Anal.* **2001**, *39*, 300−329.
(47) Braun, M. L.; Buhmann, J. M.; Müller, K.-R. On relevant dimensions in kernel feature spaces. *J. Mach. Learn. Res.* **2008**, *9*, 1875−1908.
(48) Avron, H.; Clarkson, K. L.; Woodruff, D. P. Faster kernel ridge regression using sketching and preconditioning. *SIAM J. Matrix Anal. Appl.* **2017**, *38*, 1116−1138.
(49) Benzi, M. Preconditioning techniques for large linear systems: a survey. *J. Comput. Phys.* **2002**, *182*, 418−477.
(50) Cutajar, K.; Osborne, M.; Cunningham, J.; Filippone, M. Preconditioning kernel matrices. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016; pp 2529−2538.
(51) Kaasschieter, E. F. Preconditioned conjugate gradients for solving singular systems. *J. Comput. Appl. Math.* **1988**, *24*, 265−275.
(52) Ma, S.; Belkin, M. Diving into the shallows: a computational perspective on large-scale shallow learning. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017.
(53) Shabat, G.; Choshen, E.; Or, D. B.; Carmel, N. Fast and accurate Gaussian kernel ridge regression using matrix decompositions for preconditioning. *SIAM J. Matrix Anal. Appl.* **2021**, *42*, 1073−1095.
(54) Zhou, T.; Tao, D.Godec: Randomized low-rank & sparse matrix decomposition in noisy case. In *ICML'11: Proceedings of the 28th International Conference on International Conference on Machine Learning*, 2011; pp 33−40.
(55) Li, C.; Jegelka, S.; Sra, S.Fast DPP sampling for nystrom with application to kernel methods. In *Proceedings of the 33 rd International Conference on Machine Learning*, 2016; pp 2061−2070.
(56) Sun, S.; Zhao, J.; Zhu, J. A review of Nyström methods for large-scale machine learning. *Inf. Fusion.* **2015**, *26*, 36−48.
(57) Smola, A. J.; Schökopf, B. Sparse Greedy Matrix Approximation for Machine Learning. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, 2000; pp 911−918.
(58) Drineas, P.; Mahoney, M. W.; Cristianini, N. On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. *J. Mach. Learn. Res.* **2005**, *6*, 2153−2175.
(59) Zhang, K.; Tsang, I. W.; Kwok, J. T. Improved Nyström low-rank approximation and error analysis. *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008)*, 2008; pp 1232−1239.
(60) Al Daas, H.; Rees, T.; Scott, J. Two-Level Nyström- -Schur Preconditioner for Sparse Symmetric Positive Definite Matrices. *SIAM J. Sci. Comput.* **2021**, *43*, A3837−A3861.
(61) Frangella, Z.; Tropp, J. A.; Udell, M. Randomized Nyström Preconditioning. *arXiv Preprint*, arXiv:2110.02820, 2021.
(62) Kim, H.; Teh, Y. W.Scaling up the Automatic Statistician: Scalable structure discovery using Gaussian processes. In *Practical Bayesian Nonparametrics Workshop, NIPS 2016*, 2018; pp 575−584.
(63) Foster, L.; Waagen, A.; Aijaz, N.; Hurley, M.; Luis, A.; Rinsky, J.; Satyavolu, C.; Way, M. J.; Gazis, P.; Srivastava, A. Stable and Efficient Gaussian Process Calculations. *J. Mach. Learn. Res.* **2009**, *10*, 857−882.
(64) Bach, F. Sharp analysis of low-rank kernel matrix approximations. *JMLR: Workshop and Conference Proceeding*, 2013, *30*; pp 1−25.
(65) Cohen, M. B.; Lee, Y. T.; Musco, C.; Musco, C.; Peng, R.; Sidford, A. Uniform sampling for matrix approximation. In *ITCS '15: Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, 2015; pp 181−190.
(66) Gittens, A.; Mahoney, M. W. Revisiting the Nyström method for improved large-scale machine learning. *J. Mach. Learn. Res.* **2016**, *17*, 3977−4041.
(67) Ipsen, I. C.; Wentworth, T. Sensitivity of leverage scores. *arXiv Preprint*, arXiv:1402.0957, 2014.
(68) Wang, S.; Zhang, Z. Improving CUR matrix decomposition and the Nyström approximation via adaptive sampling. *J. Mach. Learn. Res.* **2013**, *14*, 2729−2769.
(69) Alaoui, A.; Mahoney, M. W. Fast randomized kernel ridge regression with statistical guarantees. *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2015.
(70) McCurdy, S. Ridge regression and provable deterministic ridge leverage score sampling. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018*, 2018.
(71) Fine, S.; Scheinberg, K. Efficient SVM training using low-rank kernel representations. *J. Mach. Learn. Res.* **2001**, *2*, 243−264.
(72) Harbrecht, H.; Peters, M.; Schneider, R. On the low-rank approximation by the pivoted Cholesky decomposition. *Appl. Numer. Math.* **2012**, *62*, 428−440.
(73) Anzt, H.; Dongarra, J.; Flegar, G.; Higham, N. J.; Quintana-Ortí, E. S. Adaptive precision in block-Jacobi preconditioning for iterative sparse linear system solvers. *Concurr. Comput.* **2019**, *31*, e4460.
(74) Concus, P.; Golub, G. H.; Meurant, G. Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Comput.* **1985**, *6*, 220−252.

https://doi.org/10.1021/acs.jctc.2c01304
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

(75) Knyazev, A. V. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput.* **2001**, *23*, 517−541.
(76) Lee, F.-H.; Phoon, K.-K.; Lim, K.; Chan, S. Performance of Jacobi preconditioning in Krylov subspace solution of finite element equations. *Int. J. Numer. Anal. Methods. Geomech.* **2002**, *26*, 341−372.
(77) Si, S.; Hsieh, C.-J.; Dhillon, I. Memory efficient kernel approximation. *Proceedings of the 31st International Conference on Machine Learning*, 2014; pp 701−709.
(78) Chen, Y.; Epperly, E. N.; Tropp, J. A.; Webber, R. J. Randomly pivoted Cholesky: Practical approximation of a kernel matrix with few entry evaluations. *arXiv Preprint*, arXiv:2207.06503, 2022.
(79) Kabylda, A.; Vassilev-Galindo, V.; Chmiela, S.; Poltavsky, I.; Tkatchenko, A. Towards linearly scaling and chemically accurate global machine learning force fields for large molecules. *arXiv Preprint*, arXiv:2209.03985, 2022.

https://doi.org/10.1021/acs.jctc.2c01304
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX