Stress and heat flux via automatic differentiation

Special Collection: Software for Atomistic Machine Learning

Marcel F. Langer; J. Thorben Frank; Florian Knoop

J. Chem. Phys. 159, 174105 (2023)
https://doi.org/10.1063/5.0155760
Stress and heat flux via automatic differentiation

Cite as: J. Chem. Phys. 159, 174105 (2023); doi: 10.1063/5.0155760
Submitted: 23 April 2023 • Accepted: 25 September 2023 • Published Online: 3 November 2023

Marcel F. Langer, J. Thorben Frank, and Florian Knoop

AFFILIATIONS
1 Machine Learning Group, Technische Universität Berlin, 10587 Berlin, Germany
2 BIFOLD–Berlin Institute for the Foundations of Learning and Data, Berlin, Germany
3 The NOMAD Laboratory at the Fritz Haber Institute of the Max Planck Society and Humboldt University, Berlin, Germany
4 Theoretical Physics Division, Department of Physics, Chemistry and Biology (IFM), Linköping University, SE-581 83 Linköping, Sweden

Note: This paper is part of the JCP Special Topic on Software for Atomistic Machine Learning.

ABSTRACT
Machine-learning potentials provide computationally efficient and accurate approximations of the Born–Oppenheimer potential energy surface. This potential determines many materials properties and simulation techniques usually require its gradients, in particular forces and stress for molecular dynamics, and heat flux for thermal transport properties. Recently developed potentials feature high body order and can include equivariant semi-local interactions through message-passing mechanisms. Due to their complex functional forms, they rely on automatic differentiation (AD), overcoming the need for manual implementations or finite-difference schemes to evaluate gradients. This study discusses how to use AD to efficiently obtain forces, stress, and heat flux for such potentials, and provides a model-independent implementation. The method is tested on the Lennard-Jones potential, and then applied to predict cohesive properties and thermal conductivity of tin selenide using an equivariant message-passing neural network potential.

© 2023 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/). https://doi.org/10.1063/5.0155760

I. INTRODUCTION

Molecular dynamics (MD) simulations enable computational prediction of thermodynamic quantities for a wide range of quantum systems, and constitute a cornerstone of modern computational science. In MD, systems are simulated by propagating Newton’s equations of motion – potentially modified to model statistical ensembles – numerically in time, based on the forces acting on each atom due to their movement on the Born–Oppenheimer potential-energy surface (PES). Therefore, the quality of the underlying PES is important for the predictive ability of this method. First-principles electronic structure methods such as density-functional theory (DFT) can be used to perform high-accuracy \textit{ab initio} molecular dynamics (aMD) simulations, provided the exchange-correlation approximation is reliable. Such approaches are, however, restricted by high computational cost, severely limiting accessible size and time scales. Computationally efficient approximations to the underlying PES are therefore required for the atomistic simulation of larger systems: Forcefields (FFs) are built on analytical functional forms that are often based on physical bonding principles, parametrized to reproduce quantities of interest for a given material. They are computationally cheap, but parametrizations for novel materials are not always available, and their flexibility is limited by their fixed functional form. Machine-learning interatomic potentials (MLIPs), where a potential is inferred based on a small set of reference calculations, aim to retain near first-principles accuracy while remaining linear scaling with the number of atoms. While MLIPs are limited, in principle, to modeling the physical mechanisms present in the training data, they have nevertheless emerged as an important tool for MD, often combined with active learning schemes. Modern MLIPs can include semi-local interactions through message-passing (MP) mechanisms, internal equivariant representations, and body-order expansions, which enable the efficient construction of flexible many-body interactions. In such complex architectures, the manual implementation of derivatives is often unfeasible. Finite-difference approaches require tuning of additional parameters, as well as repeated energy evaluations. Automatic differentiation (AD) presents an intriguing alternative: If the computation of the potential energy $U$ is implemented in a suitable framework, derivatives such as the forces $F$ or the stress...
\( \sigma \) can be computed with the same asymptotic computational cost as computing the energy. This is accomplished by decomposing the underlying “forward” computation into elementary operations with analytically tractable derivatives, computing local gradients for a given input, and then combining the resulting values using the chain rule.

This work provides a systematic discussion of and an introduction to the use of AD to compute forces, stress, and heat flux for MLIPs. While the calculation of forces with AD is common practice,\textsuperscript{22,29,51–58} stress and heat flux are not yet commonly available for many MLIPs. Both quantities have been the focus of much previous work due to the difficulty of defining and implementing them for many-body potentials and periodic boundary conditions.\textsuperscript{1–12} Introducing an abstract definition of MLIPs as functions of a graph representation of atomic systems, unified formulations of stress and heat flux are given, which can be implemented generally for any such graph-based machine-learning potential (GLP). An example implementation using \texttt{jax}\textsuperscript{43} is provided in the \texttt{glop} package.\textsuperscript{13} To validate the approach, different formulations of stress and heat flux are compared for the Lennard-Jones potential,\textsuperscript{22,29,51–58} where analytical derivatives are readily available for comparison, as well as a state-of-the-art message-passing neural network (MPNN), \texttt{so3Krates}.

Having established the correctness of the proposed implementation, the ability of \texttt{so3Krates} to reproduce first-principles cohesive properties and thermal conductivity of tin selenide (SnSe) is studied.

### II. AUTOMATIC DIFFERENTIATION

Automatic differentiation (AD) is a technique to obtain derivatives of functions implemented as computer programs.\textsuperscript{12,13} It is distinct from \textit{numerical differentiation}, where finite-difference schemes are employed, and \textit{symbolic differentiation}, where analytical derivatives are obtained manually or via computer algebra systems, and then implemented explicitly. Instead, AD relies on the observation that complex computations can often be split up into elementary steps, for which derivatives are readily implemented. If one can track those derivatives during the computation of the forward, or “primal,” function, the chain rule allows to obtain derivatives.

For this work, two properties of AD are particularly relevant: It allows the computation of derivatives with respect to quantities that are explicitly used in the forward computation, and it can do so at the same asymptotic computational cost as the forward function. In particular, AD can obtain two quantities efficiently: Given a differentiable function \( u: \mathbb{R}^n \rightarrow \mathbb{R}^m \), the Jacobian of \( u \) is defined as the \( M \times N \) matrix \( \partial u(x)/\partial x \), AD can then obtain Jacobian-vector and vector-Jacobian products, i.e., the multiplication and summation of factors over either the input or the output dimension. This corresponds to propagating derivatives from the inputs forwards, leading to forward-mode AD, or from the end result backwards, leading to reverse-mode AD. As many popular AD frameworks are primarily implemented to work with neural networks, where scalar loss functions must be differentiated with respect to many parameters, reverse-mode AD, also called “backpropagation,”\textsuperscript{17} is more generally available. More recent frameworks implement both approaches, for instance \texttt{jax}\textsuperscript{43} which is used in the present work. AD can also be leveraged to compute contractions of higher-order derivative operators.\textsuperscript{48}

### III. CONSTRUCTING GRAPH MLIPS

This work considers periodic systems,\textsuperscript{10} consisting of \( N \) atoms with atomic numbers \( Z_i \) placed in a simulation cell which is infinitely periodically tiled in space. We define

\[
R_{\text{sc}} := \{ r_i : i = 1 \ldots N \} \quad \text{positions in simulation cell}
\]

\[
Z := \{ Z_i : i = 1 \ldots N \} \quad \text{atomic numbers}
\]

\[
B := \{ b_a : a = 1, 2, 3 \} \quad \text{basis or lattice vectors}
\]

\[
r_{ia} := r_i + \sum_a n_a b_a \quad \text{replica position}
\]

\[
r_{ij} := r_i - r_j \quad \text{minimum image convention}
\]

In this setting, a MLIP is a function that maps the structure represented by its positions, lattice vectors, and atomic numbers, \((R_{\text{sc}}, B, Z)\), to a set of atomic potential energies \( U = \{ U_i : i = 1, \ldots N \} \). Each \( U_i \) is a many-body function of positions; in general, we do not assume that a further decomposition into body-ordered contributions is available. The total potential energy is then obtained by summing over atomic contributions: \( U = \sum_i U_i \). Since AD relies on the forward computation to calculate derivatives, it is sensitive to the exact implementation of this mapping. Care must be taken to construct the MLIP such that required derivatives are available.

This work considers MLIPs that scale linearly with \( N \). Therefore, the number of atoms contributing to a given \( U_i \) must be bounded, which is achieved by introducing a cutoff radius \( r_c \), restricting interactions to finite-sized atomic neighborhoods \( N(i) = \{ r_j : |r_{ij}| \leq r_c, r_j \in R_{\text{sc}} \} \). To ensure translational invariance, MLIPs do not rely on neighbor positions directly, but rather on atom-pair vectors centered on \( i \), from which atom-pair vectors between neighboring atoms can be constructed, for instance to determine angles.

The resulting structure can be seen as a graph \( G \). The vertices \( V \) of this graph are identified with atoms, labeled with their respective atomic numbers, and connected by edges \( E \) that are labeled from that representation using a learned function, such as a neural network or a kernel machine. Such models are conceptually simple, but cannot account for effects that extend beyond \( r_c \). Atomic potential energy contributions can therefore depend on positions up to a distance of \( r_c \).

Since interactions are confined to neighborhoods at each iteration, the computational cost of semi-local models nevertheless remains linear in \( N \) and \( M \).
We term this class of potentials, which act on sets of neighborhoods and use atom-pair vectors as input, graph-based machine-learning potentials (GLPs). Models with $M = 1$ are termed local, those with $M > 1$ semi-local. By construction, this framework does not include global interactions: GLPs are still local at thermodynamic scales. Nevertheless, the distinction between local and semi-local models is useful for the purpose of classifying the functional dependence of atomic potential energy contributions, which in turn determines the derivatives available via AD: In a local model, $U_i$ is a function only of atom-pair vectors within its neighborhood, and derivatives of $U$ with respect to a given $r_{ij}$ therefore uniquely map to one $U_i$. This is not the case for semi-local models and will be discussed further in Sec. IV.

We consider two strategies to construct GLPs: The “standard” way, which includes periodic boundary conditions via the edges in the graph, and an “unfolded” formulation, where periodicity is explicitly included via replica positions. Both concepts are illustrated in Fig. 1.

In the standard architecture, vertices in $G$ are identified with atoms in the simulation cell, using the minimum image convention (MIC) to include periodicity. Edges $E_{ij}$ exist between two atoms $i$ and $j$ in $R_s$ if they interact:

$$E_{ij}^{\text{std}} = \{ r_{ij}^{\text{MIC}} : |r_{ij}^{\text{MIC}}| \leq r_c \}. \quad (1)$$

We denote the graph constructed in this manner as $G^{\text{std}}$ and the set of edges $E_{ij}^{\text{std}}$.

Alternatively, we can first determine the total set of positions $R_{\text{unf}} \subset R_{\text{eff}}$ that can interact with atoms in the simulation cell, creating an unfolded system extracted from the bulk, consisting of $R_c$ and all replicas with up to $r_{\text{eff}}^c$ distance from the cell boundary. This construction can be performed efficiently, and adds only a number of positions that is proportional to the surface of the simulation cell, therefore becoming increasingly negligible as $N$ increases at constant density. However, the number of positions increases cubically with $M$, and this construction is therefore practical only for small $M$ (see supplementary material).

We proceed by constructing a correspondingly modified graph $G^{\text{unf}}$, and compute potential energies for vertices corresponding to atoms in the simulation cell only. By construction, since the same atom-pair vectors appear in the graph, this approach then reproduces the potential energy of the standard method. A similar construction appears in spatial decomposition approaches for multi-device parallelization, discussed for instance in Ref. 59: To contain the calculation of interactions in one device, all interacting positions outside a given domain must be considered as “ghost atoms.” The number of ghost atoms corresponds to the number of additional atoms due to the unfolding. As this approach becomes unfeasible for large values of $r_{\text{eff}}^c$, alternative schemes must be considered, for instance propagating message-passing updates, and later gradient information, between devices.

IV. DERIVATIVES

Having constructed the forward function for a given GLP, we can compute derivatives with respect to its inputs using AD. In this section, we discuss how forces, stress, and heat flux can be computed in this manner, and demonstrate the relationship between different formulations.

A. Forces

For MD, the most relevant quantities are the forces...
acting on the atoms in the simulation cell. Since \( R_{\text{c}} \) are an explicit input, they can be computed directly with AD: \( U \) is a scalar, and this therefore is a trivial Jacobian-vector product, which can be computed with the same asymptotic cost as \( U \).

An interesting situation arises if pairwise forces are desired. Strictly speaking, in a many-body MLIP, where interactions cannot be decomposed into pairwise contributions, such quantities are not well-defined and Newton’s third law is replaced by conservation of momentum, which requires \( \sum_{i=1}^{N} F_i = 0 \). Nevertheless, pairwise forces with an antisymmetric structure can be defined by exploiting the construction of GLPs in terms of atom-pair vectors. In the standard formulation, \( U \) is a function of all edges,

\[
U = U(\{ r_{ij} : ij \in E_{\text{std}} \}).
\]

Hence, by the chain rule,

\[
F_i = \sum_{j \in \mathcal{N}(i)} \frac{\partial U}{\partial r_{ij}} = \frac{\partial U}{\partial r_{ji}}
\]

The pairwise forces such defined exhibit anti-symmetry, and therefore fulfill Newton’s third law. For \( M = 1 \), the local case, this definition reduces to a more standard form:

\[
F_i = \frac{\partial U_i}{\partial r_{ij}} - \frac{\partial U_j}{\partial r_{ji}}.
\]

However, for general GLPs with \( M > 1 \), this definition includes a sum over all \( U_i \) that are influenced by a given edge

\[
F_{ij} = \sum_{k \in \mathcal{N}(i)} \frac{\partial U_k}{\partial r_{ij}} - \frac{\partial U_j}{\partial r_{ji}},
\]

subverting expectations connecting local potential energies to pairwise forces. We note that this seeming contradiction is a consequence of the combination of the peculiar construction of GLPs and AD: In principle, it is always possible to define extended neighborhoods up to \( r_{\text{eff}} \), obtaining \( U_i \) purely as a function of atom-pair vector originating from \( i \). However, to construct derivatives with respect to these atom-pair vectors with AD, these extended neighborhoods have to be constructed and included explicitly, therefore negating the computational efficiency gains of a GLP architecture.

### B. Stress

The definition of the (potential) stress is

\[
\sigma = \frac{1}{V} \frac{\partial U}{\partial \epsilon}_{|\epsilon = 0},
\]

with \( U \) denoting the potential energy after a strain transformation with the \( 3 \times 3 \) symmetric tensor \( \epsilon \)

\[
r \rightarrow (I + \epsilon) \cdot r,
\]

acting on \( R_{\text{eff}} \).

One approach, followed for instance by schnetpack,\textsuperscript{28,61} schnetpack 2.0,\textsuperscript{30} and nequip,\textsuperscript{34,66} is to inject the strain transformation explicitly into the construction of the GLP. This can be done at different points: One can transform \( R_{\text{c}} \) and \( B \) before constructing \( G \), directly transform atom-pair vectors, or transform all contributing positions \( R_{\text{neff}} \). Alternatively, as the inner derivative of inputs with respect to \( \epsilon \) is simply the input, the derivative of \( U \) with respect to inputs can be obtained with AD, and the stress can then be computed analytically from the results. This avoids modifying the forward computation of \( U \) entirely.

These approaches yield, with \( \otimes \) denoting an outer product,

\[
V \sigma = \frac{\partial U(R_{\text{c}}, B)}{\partial \epsilon} \quad (10)
\]

\[
= \frac{\partial U(E)}{\partial \epsilon} \quad (11)
\]

\[
= \frac{\partial U(R_{\text{neff}})}{\partial \epsilon} \quad (12)
\]

\[
= \sum_{i \in \mathcal{N}(i)} r_i \otimes \frac{\partial U}{\partial r_i} + \sum_{b \in \mathcal{B}} b \otimes \frac{\partial U}{\partial b} \quad (13)
\]

\[
= \sum_{i \in \mathcal{N}(i)} r_i \otimes \frac{\partial U}{\partial r_i} \quad (14)
\]

\[
= \sum_{i \in \mathcal{N}(i)} r_i \otimes \frac{\partial U}{\partial r_i} \quad (15)
\]

recovering previous formulations,\textsuperscript{31,33} Louwerse and Baerends\textsuperscript{31} considered the calculation of the stress for periodic systems, highlighting that contributions from replica positions must be considered, either by computing the elementary definition in Eq. (10) or the inclusion of the second term in Eq. (13). Both options are considered “much effort” in that work. Thompson \textit{et al.}\textsuperscript{33} discuss an alternative approach based on explicitly including replica positions, obtaining a result similar to the unfolded construction used for Eq. (15).

All these forms of the stress are equivalent and can be implemented directly with AD, obtaining numerically identical results as seen in Tables I and IV. In all cases, as the scalar \( U \) is differentiated with respect to its inputs, asymptotic cost remains linear.

However, as AD computes algorithmic derivatives, which reflect the particular implementation of the energy function exactly, care must be taken to ensure that the strain transformation in Eq. (9) is applied consistently. For example, obtaining a correct stress with Eqs. (10) and (13), where derivatives are obtained before the construction of the input graph, requires that the implementation of the MIC used matches the definition in Sec. III. If the MIC is implemented using fractional coordinates,\textsuperscript{63} this can be challenging. In such cases, it may be preferable to rely on approaches that compute derivatives on the basis of the graph input, such as Eqs. (11) and (14). This method is adopted, for instance, by \texttt{jax-md}.\textsuperscript{27} Many other MLIP packages use offsets to implement the MIC, and can therefore use Eq. (10), for instance schnetpack,\textsuperscript{28,61} schnetpack 2.0,\textsuperscript{30} nequip,\textsuperscript{34,66} and mace.\textsuperscript{34} The \texttt{matgl} package,\textsuperscript{28,61} on the other hand, uses Eq. (14), avoiding the introduction of a strain transformation.
The Journal of Chemical Physics

The fundamental definition of the heat flux for MLIPs was originally derived by Hardy for periodic quantum systems. It reads:

$$J_{\text{Full}} = \sum_{j \in R_{\text{all}}} r_j \left( \frac{\partial U_j}{\partial \mathbf{r}_j} \cdot \mathbf{v}_j \right) + \sum_{i \in R_{\text{sys}}} E_i \mathbf{v}_i$$

$$= J_{\text{pot}} + J_{\text{conv}},$$

where $\mathbf{v}_i$ denote velocities, $m_i$ masses, and $E_i = U_i + 1/2m_i v_i^2$ is the total energy per atom. Intuitively, the “potential” term $J_{\text{pot}}$ describes how the total instantaneous change in $U_i$ can be attributed to interactions with other atoms, with energy flowing between them, while the second, “convective,” term $J_{\text{conv}}$ describes energy being carried by individual atoms. In the present setting, $J_{\text{conv}}$ can be computed directly, as $E_i$ are available. $J_{\text{pot}}$, however, presents a challenge in an AD framework: In principle, $J_{\text{pot}}$ could be computed directly, obtaining the required partial derivatives with AD. However, as $J_{\text{pot}}$ is neither a Jacobian-vector nor a vector-Jacobian product, this requires repeated evaluations over the input or output dimension. Even when restricting $j \in R_{\text{sys}}$, which can be achieved by introducing the MIC for $r_j$ (see supplementary material for details),

$$J_{\text{pot}}^{\text{MIC}} = \sum_{j \in R_{\text{sys}}} r_j \left( \frac{\partial U_j}{\partial \mathbf{r}_j} \cdot \mathbf{v}_j \right),$$

the computational cost of a direct implementation with AD scales quadratically with $N$, rendering the system sizes and simulation times required for the GK method inaccessible. Based on previous work, we therefore survey approaches that restore linear scaling in the following.

For $M = 1$, edges can be uniquely assigned to atomic energy contributions as discussed for atomic stresses in Sec. IV A. In this case

$$\frac{\partial U}{\partial \mathbf{r}_j} = \sum_{k \in R_{\text{sys}}} \frac{\partial U_k}{\partial \mathbf{r}_k} \frac{\partial \mathbf{r}_k}{\partial \mathbf{r}_j},$$

so that

$$J_{\text{pot}}^{\text{MIC}} = \sum_{j \in R_{\text{sys}}} \left( r_j \frac{\partial U}{\partial \mathbf{r}_j} \cdot \mathbf{v}_j \right),$$

which requires a single evaluation of reverse-mode AD.

We note that the terms appearing in front of $v_j$ also appear in the stress in Eq. (14). However, for a given $j$, the pre-factor cannot be identified with the atomic stress as defined in Eq. (16) – the atomic energy being differentiated is not $U_j$ but $U_i$. The indices can only be exchanged for additive pairwise potentials; this inequivalence was recently corrected in the large-scale atomic/molecular massively parallel simulator (LAMMPS) code. This approach is not applicable for $M > 1$, since the relation in Eq. (20) no longer holds, and the mapping between stress contributions and heat flux contributions becomes invalid. Therefore, Eq. (19) must be used, incurring quadratic computational cost.

To restore linear scaling, one must find a way to rewrite Eq. (19) in terms of either Jacobian-vector or vector-Jacobian products by executing the sum over $i$ before taking the derivative with respect to $j$. The use of the MIC for the $r_j^{\text{MIC}}$ prefactors prevents this: An independent offset appears in $J_{\text{pot}}^{\text{MIC}} = r_j - \sum_{a} n_a b_a$. Adopting the unfolded construction avoids this issue, as replica positions are no longer used.

Introducing auxiliary positions $r_i^{\text{aux}}$, which are numerically identical to the positions $r_i$, but not used to compute $U$, and defining the energy barycenter $B = \sum_{r_i^{\text{aux}}} r_i^{\text{aux}} U_i$, the heat flux becomes

$$\frac{\partial U}{\partial \mathbf{r}_j} = \sum_{j \in R_{\text{sys}}} \frac{\partial B}{\partial \mathbf{r}_j} \cdot \mathbf{v}_j - \sum_{i \in R_{\text{sys}}} \left( r_j \frac{\partial U}{\partial \mathbf{r}_j} \cdot \mathbf{v}_j \right).$$

The first term requires three reverse-mode evaluations, or one forward-mode evaluation. Correspondingly, the second term requires one reverse-mode, or three forward-mode, evaluations. Since the overhead introduced by explicitly constructing $R_{\text{sys}}$ scales as $N^{\frac{3}{2}}$, overall linear scaling is restored, albeit with a pre-factor due to the higher number of positions to be considered.

To summarize, we have introduced two forms of the heat flux that can be implemented efficiently with AD: Eq. (21), which applies
to GLPs with $M = 1$, and Eq. (22), which applies for $M \geq 1$, but introduces some additional overhead. Both are equivalent to the general, quadratically-scaling, form given in Eq. (17), as seen in Tables II and VI.

### V. EXPERIMENTS

#### A. Lennard-Jones argon

The stress formulas in Eqs. (10)–(15) and heat flux formulas Eqs. (19), (21), and (22) have been implemented in the glp package\(^{43}\) using jax.\(^{44}\) As a first step, we numerically verify this implementation.

To this end, the Lennard-Jones potential\(^{45}\) is employed, where analytical derivatives including those required for the heat flux are readily available, and implementations are included in many packages, for example the atomic simulation environment (ASE).\(^{73}\) In the GLP framework, the Lennard-Jones potential can be seen as an extreme case of a $M = 1$ GLP, where $U_1$ is composed of a sum of pair terms:

$$U_1 = \frac{1}{2} \sum_{j \in \mathbb{N}(i)} 4\varepsilon \left( \frac{\sigma_{ij}^{12}}{r_{ij}^{12}} - \frac{\sigma_{ij}^{6}}{r_{ij}^{6}} \right).$$  

(23)

For this experiment, parameters approximating elemental argon are used.\(^{72}\) 100 randomly displaced and distorted geometries, based on the 512-atom $8 \times 8 \times 8$ supercell of the face-centered cubic primitive cell with lattice parameter 3.72 Å and angle 60° are used. Random velocities to evaluate a finite heat flux are sampled from the Boltzmann distribution corresponding to 10 K. Additional computational details are discussed in the supplementary material.

Table I compares the stress formulations in Eqs. (10)–(15) with finite differences. We report "best-case" results for finite differences, choosing the stepsize that minimizes the error. In the table, the mean absolute error (MAE) and mean absolute percentage error (MAPE) with respect to the analytical ground truth are reported. All given formulations are found to be equivalent. In single precision arithmetic, the AD-based implementations outperform finite differences, in double precision, errors are similar.

For the heat flux, finite difference approaches are not feasible. Therefore, only AD-based implementations are shown in Table II. In the case of the Lennard-Jones potential, where $M = 1$, Eqs. (19), (21), and (22) are found to be identical.

Having confirmed the numerical correctness of the presented implementations, we then proceed with a benchmark of the computation time of Eqs. (10)–(15). Here, we construct cubic supercells of varying sizes, generating 1000 randomly displaced samples; additional details are given in the supplementary material. The results, computed on one Nvidia V100 graphics processing unit (GPU), are given in Table III; Overall, computation times per atom are similar for AD-based approaches. Computation time per atom decreases with system size, as calculations are parallelized on the GPU. In this case, stress formulations based on the unfolded construction are on par with other approaches, indicating that in this setting, computational cost is dominated by graph construction, rather than the evaluation of the potential itself. For more complex potentials, as seen in Sec. VI, this is not the case, and the introduction of additional positions yields significant computational cost. The best performance in single precision is obtained for Eq. (14), which avoids the use of a strain transformation, and re-uses gradients computed for the forces. For double precision, Eq. (15) is optimal, which avoids the use of the MIC or a strain transformation, and also re-uses gradients.

#### TABLE II. Error in heat flux for Lennard-Jones argon, comparing different formulations to analytical derivatives. Results are shown for both single and double precision arithmetic.

<table>
<thead>
<tr>
<th>Equations</th>
<th>Single (eV Å/fs)</th>
<th>Single (%)</th>
<th>Double (eV Å/fs)</th>
<th>Double (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(19)</td>
<td>$2.81 \times 10^{-9}$</td>
<td>$1.71 \times 10^{-2}$</td>
<td>$1.47 \times 10^{-10}$</td>
<td>$6.81 \times 10^{-4}$</td>
</tr>
<tr>
<td>(21)</td>
<td>$2.84 \times 10^{-9}$</td>
<td>$1.67 \times 10^{-2}$</td>
<td>$1.47 \times 10^{-10}$</td>
<td>$6.81 \times 10^{-4}$</td>
</tr>
<tr>
<td>(22)</td>
<td>$2.44 \times 10^{-9}$</td>
<td>$1.54 \times 10^{-2}$</td>
<td>$1.47 \times 10^{-10}$</td>
<td>$6.81 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

#### TABLE III. Time per atom to compute stress with finite differences and different AD-based formulations for Lennard-Jones argon. Results are shown for both single and double precision, and for different simulation cell sizes $N$.

<table>
<thead>
<tr>
<th>Equations</th>
<th>Single</th>
<th>Double</th>
</tr>
</thead>
<tbody>
<tr>
<td>F.d.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(10)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(11)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(12)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(13)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(14)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(15)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$N = 864$</th>
<th>$N = 2048$</th>
<th>$N = 4000$</th>
<th>$N = 864$</th>
<th>$N = 2048$</th>
<th>$N = 4000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F.d.</td>
<td>$3.1 \times 10^{-5}$</td>
<td>$1.6 \times 10^{-5}$</td>
<td>$9.9 \times 10^{-6}$</td>
<td>$3.8 \times 10^{-5}$</td>
<td>$2.1 \times 10^{-5}$</td>
</tr>
<tr>
<td>(10)</td>
<td>$1.0 \times 10^{-6}$</td>
<td>$7.1 \times 10^{-7}$</td>
<td>$6.0 \times 10^{-7}$</td>
<td>$1.8 \times 10^{-6}$</td>
<td>$1.3 \times 10^{-6}$</td>
</tr>
<tr>
<td>(11)</td>
<td>$7.7 \times 10^{-7}$</td>
<td>$5.1 \times 10^{-7}$</td>
<td>$4.5 \times 10^{-7}$</td>
<td>$1.3 \times 10^{-6}$</td>
<td>$9.8 \times 10^{-7}$</td>
</tr>
<tr>
<td>(12)</td>
<td>$9.2 \times 10^{-7}$</td>
<td>$5.4 \times 10^{-7}$</td>
<td>$3.7 \times 10^{-7}$</td>
<td>$1.1 \times 10^{-6}$</td>
<td>$6.7 \times 10^{-7}$</td>
</tr>
<tr>
<td>(13)</td>
<td>$9.4 \times 10^{-7}$</td>
<td>$6.7 \times 10^{-7}$</td>
<td>$5.8 \times 10^{-7}$</td>
<td>$1.6 \times 10^{-6}$</td>
<td>$1.3 \times 10^{-6}$</td>
</tr>
<tr>
<td>(14)</td>
<td>$7.2 \times 10^{-7}$</td>
<td>$4.5 \times 10^{-7}$</td>
<td>$3.6 \times 10^{-7}$</td>
<td>$1.2 \times 10^{-6}$</td>
<td>$9.0 \times 10^{-7}$</td>
</tr>
<tr>
<td>(15)</td>
<td>$8.9 \times 10^{-7}$</td>
<td>$5.3 \times 10^{-7}$</td>
<td>$3.7 \times 10^{-7}$</td>
<td>$1.0 \times 10^{-6}$</td>
<td>$6.5 \times 10^{-7}$</td>
</tr>
</tbody>
</table>
VI. TIN SELENIDE WITH SO3KRATES

To investigate stress and heat flux in a practical setting, we now study tin selenide (SnSe) using the state-of-the-art SO3KRATES GLP. In contrast to other equivariant MLIPs, for instance NEQUIP,28 SO3KRATES replaces shared equivariant feature representations by separated branches for invariant and equivariant information, whose information exchange is handled using an equivariant self-attention mechanism. By doing so, one can achieve data efficiency and extrapolation quality competitive to state-of-the-art GLPs at reduced time and memory complexity. As non-local interactions are not modeled in the GLP framework introduced in this work, global interactions are disabled in the SO3KRATES models used at present.

The training data for this task is obtained from a large-scale ab initio Green–Kubo (aiGK) benchmark study by Knoop et al.,73,74 and consists of four aiMD trajectories in the NVT ensemble at 300 K and different volumes, from which a total of 3000 timesteps are used. These simulations were performed as the first steps of the aiGK workflow, first determining the equilibrium volume at 300 K and then generating thermalized starting geometries for the long-running NVE simulations required for the GK method. For this experiment, we repurpose these thermalization trajectories as training data, aiming to replace the subsequent, computationally expensive, aiMD with MLIP-accelerated MD. Additional details on the MLIP training can be found in the supplementary material.

A. Implementation of stress and heat flux

While no analytical derivatives are available for SO3KRATES, the implementation of the stress can be verified with finite differences, and the heat flux can be checked for consistency between

| Table IV. Error in stress for tin selenide, comparing different formulations to finite differences, for SO3KRATES with $M = 2$. Results are shown for both single and double precision arithmetic, and for $\sigma - V$ in place of $\sigma$. Results for other $M$, which are similar to the one shown here, can be found in the supplementary material. |
| --- | --- | --- | --- |
| Equations | Single MAE (eV) | MAPE (%) | Double MAE (eV) | MAPE (%) |
| (10) | $1.58 \times 10^{-2}$ | 4.40 | $1.45 \times 10^{-4}$ | 2.32 |
| (11) | $1.58 \times 10^{-2}$ | 4.38 | $1.45 \times 10^{-4}$ | 2.32 |
| (12) | $1.57 \times 10^{-2}$ | 4.33 | $1.45 \times 10^{-4}$ | 2.32 |
| (13) | $1.58 \times 10^{-2}$ | 4.40 | $1.45 \times 10^{-4}$ | 2.32 |
| (14) | $1.58 \times 10^{-2}$ | 4.38 | $1.45 \times 10^{-4}$ | 2.32 |
| (15) | $1.57 \times 10^{-2}$ | 4.33 | $1.45 \times 10^{-4}$ | 2.32 |

| Table V. Time per atom to compute stress with finite differences and different AD-based formulations, for tin selenide and SO3KRATES with $M = 2$. Results are shown for both single and double precision, and for different simulation cell sizes $N$. |
| --- | --- | --- | --- |
| Equations | Single Time per atom (s) | Double Time per atom (s) |
| $N = 864$ | $N = 2048$ | $N = 4000$ | $N = 864$ | $N = 2048$ | $N = 4000$ |
| F.d. | $1.2 \times 10^{-4}$ | $9.5 \times 10^{-5}$ | $8.8 \times 10^{-5}$ | $2.2 \times 10^{-4}$ | $1.9 \times 10^{-4}$ | $1.8 \times 10^{-4}$ |
| (10) | $8.2 \times 10^{-6}$ | $7.1 \times 10^{-6}$ | $6.9 \times 10^{-6}$ | $1.6 \times 10^{-5}$ | $1.5 \times 10^{-5}$ | $1.5 \times 10^{-5}$ |
| (11) | $8.0 \times 10^{-6}$ | $7.0 \times 10^{-6}$ | $6.9 \times 10^{-6}$ | $1.6 \times 10^{-5}$ | $1.5 \times 10^{-5}$ | $1.5 \times 10^{-5}$ |
| (12) | $3.5 \times 10^{-5}$ | $2.5 \times 10^{-5}$ | $2.0 \times 10^{-5}$ | $8.4 \times 10^{-5}$ | $5.7 \times 10^{-5}$ | $4.6 \times 10^{-5}$ |
| (13) | $8.0 \times 10^{-6}$ | $7.1 \times 10^{-6}$ | $6.9 \times 10^{-6}$ | $1.6 \times 10^{-5}$ | $1.5 \times 10^{-5}$ | $1.5 \times 10^{-5}$ |
| (14) | $8.0 \times 10^{-6}$ | $7.0 \times 10^{-6}$ | $6.8 \times 10^{-6}$ | $1.6 \times 10^{-5}$ | $1.5 \times 10^{-5}$ | $1.5 \times 10^{-5}$ |
| (15) | $3.5 \times 10^{-5}$ | $2.5 \times 10^{-5}$ | $2.0 \times 10^{-5}$ | $8.4 \times 10^{-5}$ | $5.7 \times 10^{-5}$ | $4.6 \times 10^{-5}$ |

| Table VI. Error in heat flux for tin selenide, comparing different formulations to the baseline implementation in Eq. (19) for SO3KRATES models with differing numbers of interaction steps $M$. Results are shown for both single and double precision arithmetic. |
| --- | --- | --- | --- |
| Equations | Single MAE (Å/fs) | MAPE (%) | Double MAE (Å/fs) | MAPE (%) |
| $M$ | $M$ | $M$ | $M$ | $M$ |
| (21) | 1 | $5.78 \times 10^{-9}$ | $9.74 \times 10^{-14}$ | $1.09 \times 10^{-17}$ | $1.73 \times 10^{-12}$ |
| (22) | 1 | $8.75 \times 10^{-8}$ | $2.65 \times 10^{-2}$ | $1.69 \times 10^{-16}$ | $4.31 \times 10^{-11}$ |
| (21) | 2 | $3.73 \times 10^{-3}$ | $5.84 \times 10^{2}$ | $3.73 \times 10^{-3}$ | $5.84 \times 10^{2}$ |
| (22) | 2 | $9.36 \times 10^{-8}$ | $1.00 \times 10^{-2}$ | $1.54 \times 10^{-16}$ | $1.60 \times 10^{-11}$ |
| (21) | 3 | $1.01 \times 10^{-3}$ | $3.27 \times 10^{2}$ | $1.01 \times 10^{-3}$ | $3.25 \times 10^{2}$ |
| (22) | 3 | $9.74 \times 10^{-8}$ | $3.04 \times 10^{2}$ | $1.65 \times 10^{-16}$ | $2.91 \times 10^{-11}$ |
different implementations. Similar to Sec. V A, we use 100 randomly displaced and distorted $4 \times 8 \times 8$ supercells of the 0 K primitive cell of SnSe for this experiment, sampling velocities from the Boltzmann distribution at 10 K to evaluate a finite heat flux.

Table IV compares the stress implementations in Eqs. (10)–(15) with finite differences, confirming the equivalence of all implemented formulations. Table V shows the computation times for the stress, for 3KRATES with $M = 2$; additional tables for $M = 1, 3$ can be found in the supplementary material. In contrast to the results for the Lennard-Jones potential, all AD-based approaches using the standard construction display similar performance, while the unfolded formulations, Eqs. (12) and (15), are significantly slower to compute, with the additional computational cost proportional to the number of unfolded positions. We conclude that in this setting, using a state-of-the-art MLIP, the computational cost of evaluating the potential dominates implementation differences in the stress, and all standard methods can be used.

Table VI compares the heat flux formulations in Eqs. (21) and (22) with the baseline in Eq. (19), implementing the quadratically-scaling “Hardy” heat flux with the MIC. For $M = 1$, all formulations are precisely equivalent. For $M > 1$, the semi-local case, only Eq. (22) is equivalent to the “Hardy” heat flux; Eq. (21) does not apply and consequently is not equivalent, displaying large deviations.

B. Equation of state and pressure

To assess the capability of 3KRATES to predict stress- and pressure-related materials properties, we calculate energy-volume and pressure-volume curves for SnSe to obtain an equation of state (EOS) of the Vinet form.\textsuperscript{27} The experiment is performed for unit cells that were homogeneously strained up to ±2% starting from the fully relaxed geometry, and relaxing the internal degrees of freedom afterwards. The energy vs volume curves for 3KRATES with $M = 1, 2, 3$ interaction steps are shown in Fig. 2 in comparison to the DFT reference using the PBEsol exchange-correlation functional with “light” default basis sets in FHI-aims.\textsuperscript{77,78} 3KRATES with three interaction steps ($M = 3$) yields the best visual agreement for the energy-volume curve in Fig. 2, and the best equilibrium volume. To quantify the agreement, we evaluate the Vinet EOS and extract the cohesive properties equilibrium volume $V_0$, isothermal bulk modulus $B_0$, and its volume derivative $B'_0$ (functional forms are given in the supplementary material). Results are listed in Table VII. For 3KRATES with three interaction steps ($M = 3$), the predicted volume deviates by –16% from the DFT reference, and the deviation of the bulk modulus is –3.14%. A larger error is seen for the volume derivative of the bulk modulus, $B'_0$, which deviates by 27.7%, indicating worse agreement further away from the training region. Overall, the agreement between DFT and 3KRATES when predicting cohesive properties can be considered satisfactory. The energy-volume predictions are very good, and transfer even to volumes that are larger or smaller than the ones seen during training.

Finally we check the internal consistency of energy and stress predictions with 3KRATES by fitting energy-volume and pressure-volume curves and verify that they yield identical parameters for the EOS. Results are shown in Fig. 3. As seen there, results are in perfect agreement, which verifies that stress and resulting pressure are consistent with the underlying energy function as expressed in Eq. (8).

C. Thermal conductivity

Finally, we proceed to GK calculations, following the approach outlined in our previous work.\textsuperscript{42,83} Finding that simulation cells

\begin{table}[h]
\centering
\caption{Cohesive properties of SnSe for PBEsol-DFT, and 3KRATES with different numbers of interaction steps $M$ obtained via the Vinet EOS. Best values are highlighted. Values are in good agreement with existing literature.\textsuperscript{29}}
\begin{tabular}{lccc}
\hline
 & DFT & $M = 1$ & $M = 2$ & $M = 3$ \\
\hline
$B_0$ (eV/Å$^3$) & 0.230 & 0.236 & 0.229 & 0.223 \\
$B'_0$ (eV/Å$^6$) & 5.576 & 2.830 & 4.815 & 7.118 \\
Error $B_0$ ($\%$) & ⋯ & 2.58 & −0.47 & −3.14 \\
Error $B'_0$ ($\%$) & ⋯ & −49.25 & −13.64 & 27.66 \\
Error $V_0$ ($\%$) & ⋯ & 0.22 & −0.40 & −0.16 \\
\hline
\end{tabular}
\end{table}
with \( N = 864 \) atoms and a simulation duration of \( t_0 = 1 \text{ ns} \) yield converged results (see supplementary material), we run 11 MD simulations using the glp package and so3krates, computing \( J \) at every step.

Table VIII shows the result: For \( M = 2, 3 \), so3krates in excellent agreement with the work by Brorsson et al.\(^{80} \) which uses a force constant potential (FCP). A larger difference is observed with the work by Liu et al.\(^{81} \) who, however, use the Perdew–Burke–Ernzerhof (PBE) exchange-correlation functional, as opposed to PBEsol, which was used for the present work. The observed thermal conductivity is consistent with experiments, as well as the size-extrapolated DFT result of Knoop et al.\(^{73} \) The anisotropy of \( \kappa \) in SnSe is captured as well.

Overall, we find that so3krates with more than one interaction step is able to predict the converged thermal conductivity of SnSe, using training data only from the thermalization step of a full aiGK workflow. With this approach, long-running equilibrium aiMD simulations, the bottleneck of the GK method, have been avoided.

VII. CONCLUSION

We explained how and demonstrated that the stress and heat flux can be computed efficiently with AD for potentials based on a graph of atom-pair vectors, which we termed GLPs, and provided example implementations in the glp package.\(^{44} \) In order to provide this functionality, glp also implements the necessary infrastructure to generate input graphs from structures, as well as the ability to compute forces. With this, it can be used as a lightweight driver for MD. An example implementation of the velocity Verlet integrator\(^{45} \) is therefore also provided. This functionality is extended in the gkx package,\(^{46} \) which implements the GK method using GLP.

Numerical experiments for Lennard-Jones argon and tin selenide with the so3krates GLP, verified that these quantities are computed correctly, efficiently, and consistently. The equivariant so3krates GLP was shown to predict cohesive properties and thermal conductivity of SnSe in good agreement with DFT, other MLIPs, and experiments, confirming the practical relevance of computational access to stress and heat flux.

This work enables the use of a large class of recently developed MLIPs, those which can be described in the GLP framework, in computational materials science, and in particular for the calculation of thermal conductivities using the GK method. For GLPs implemented with jax, the glp package is provided to enable the calculation of stress and heat flux without requiring further implementation efforts.

SUPPLEMENTARY MATERIAL

The supplementary material contains (1) an explanation of Eq. (19), (2) additional details and results for the numerical experiments in Sec. V, (3) information about the GK workflow and convergence, and (4) explicit forms for the equation of state.

ACKNOWLEDGMENTS

M.F.L. was supported by the German Ministry for Education and Research BIFOLD program (Ref. Nos. 01IS18025A and 01IS18037A), and by the TEC1p Project (ERC Horizon 2020 Grant No. 740233). M.F.L. would like to thank Samuel Schoenholz and Niklas Schmitz for constructive discussions, Shuo Zhao for feedback on the manuscript, and acknowledges contributions by Fabian Nagel and Adam Norris. J.T.F. acknowledges support from the Federal Ministry of Education and Research (BMBF) and BIFOLD program (Ref. Nos. 01IS18025A and 01IS18037A). F.K. acknowledges support from the Swedish Research Council (VR) Program No. 2020-04630, and the Swedish e-Science Research Centre (SeRC). The computations were partially enabled by the Berzelius resource provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre (NSC) and by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at NSC partially funded by the Swedish Research Council through Grant Agreement No. 2022-06725. Xuan Gu at NSC is acknowledged for technical assistance on the Berzelius resource.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Marcel F. Langer: Conceptualization (lead); Investigation (equal); Project administration (lead); Software (lead); Visualization (equal); Writing – original draft (lead); Writing – review & editing (lead).

J. Thorben Frank: Investigation (equal); Software (supporting); Writing – original draft (supporting); Writing – review & editing (supporting).
REFERENCES


44. See https://github.com/sirma/colp for the glp software library.


85 See https://github.com/sirmarcel/gkx/ for the gkx software library.