GOLF.jl: Generation and Optimization of Machine Learning Force Field

CESMIX Team

December 22, 2020

1 Notation

N	no. atoms	$\boldsymbol{r}$	(x, y, z)	3×1
N_i^{nb}	no. neighbors around atom i	r_{cut}	cut-off radius	3×1
$\mathcal{N}^{\text{nb}}$	$\sum N_i^{\text{nb}}$ total no. neighbors	Ω_i	neighbor list of atom i	$N_i^{\text{nb}} \times 1$
M	no. basis functions	$\boldsymbol{r}_i$	(x_i, y_i, z_i)	3×1
T	no. atom types	$\boldsymbol{r}_{ij}$	(x_{ij}, y_{ij}, z_{ij})	3×1
J	no. configurations	$\boldsymbol{r}^N$	$(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_N)$	$N \times 3$
N_j	no. atoms for configuration j	$\boldsymbol{r}^{N_j}$	$(\boldsymbol{r}_{1j}, \ldots, \boldsymbol{r}_{N_j j})$	$N_j \times 3$
$\mathcal{N}$	$\sum_{j=1}^{J} N_j$	$\boldsymbol{r}^{\mathcal{N}}$	$(\boldsymbol{r}^{N_1}, \ldots, \boldsymbol{r}^{N_J})$	$\mathcal{N} \times 3$
$\boldsymbol{r}$	position vector	$\boldsymbol{c}$	$(c_1, c_2, \ldots, c_M)$	$M \times 1$
$\boldsymbol{r}_i$	position vector of atom i	W_M	$\{d_m(\boldsymbol{r}^N)\}_{m=1}^{M}$	$M \times 1$
$\tilde{\boldsymbol{r}}_i$	minimum periodic image of atom i	$U(\boldsymbol{r}^N; \boldsymbol{c})$	$\sum_{m=1}^{M} c_m d_m(\boldsymbol{r}^N)$	1×1
U	potential energy	$\boldsymbol{f}_i(\boldsymbol{r}^N; \boldsymbol{c})$	$\sum_{m=1}^{M} c_m \frac{\partial d_m(\boldsymbol{r}^N)}{\partial \boldsymbol{r}_i}$	3×1
$\boldsymbol{f}_i$	forces acting on atom i	$\boldsymbol{f}_i(\boldsymbol{r}^{N_j}; \boldsymbol{c})$	$\sum_{m=1}^{M} c_m \frac{\partial d_m(\boldsymbol{r}^{N_j})}{\partial \boldsymbol{r}_i}$	3×1
W_M	basis set of M functions	$\boldsymbol{f}_i^{\text{qm}}(\boldsymbol{r}^{N_j}; \boldsymbol{c})$	QM forces	3×1
w_j	weight for configuration j	$\boldsymbol{w}$	$(w_1, \ldots, w_J)$	$J \times 1$

2 Force Field Generation

2.1 Formulation

1. For any given $\boldsymbol{r}^N = (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_N)$, we generate a set of M basis functions:

$$W_M = \text{span}\{d_m(\boldsymbol{r}^N),\ 1 \le m \le M\}. \tag{1}$$

2. For a given vector of M coefficients, $\boldsymbol{c} = (c_1, \ldots, c_M)$, we then compute the potential energy as

$$U(\boldsymbol{r}^N; \boldsymbol{c}) = \sum_{m=1}^{M} c_m d_m(\boldsymbol{r}^N) \tag{2}$$

and the atomic forces as

$$\boldsymbol{f}_i(\boldsymbol{r}^N; \boldsymbol{c}) = \sum_{m=1}^{M} c_m \frac{\partial d_m(\boldsymbol{r}^N)}{\partial \boldsymbol{r}_i}, \quad i = 1, \ldots, N. \tag{3}$$

2.2 Input files

- $\boldsymbol{Z} = (Z_1, \ldots, Z_T)$: 1D array of size T, where Z_t is the atomic number for atom type t.
- $\boldsymbol{c} = (c_1, \ldots, c_M)$: 1D array of M floats, where M is the number of basis functions. Note that M must be divisible by T.

The main input file contains $\boldsymbol{Z}$ and parameters related to the generation of the force field. The second input file contains $\boldsymbol{c}$.

2.3 Output files

The output of the software is a C++ code that computes MD potential energy and forces for any given $\boldsymbol{r}^N = (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_N)$.

- The software will produce C++ code for computing atomic forces.
- The resulting C++ code can be plugged into LAMMPS to enable MD simulations.
- The C++ code should be able to run on both CPUs and GPUs.

3 Force Field Optimization

3.1 Formulation

We want to find an optimal coefficient vector $\boldsymbol{c}^*$ to match the quantum mechanical (QM) forces. In particular, we assume that we are given the QM forces $\boldsymbol{f}_i^{\text{qm}}(\boldsymbol{r}^{N_j})$ for J different configurations $\{\boldsymbol{r}^{N_j}\}_{j=1}^{J}$, where each configuration j contains N_j atoms. The optimal coefficient vector is the solution of the least-squares problem:

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}\in\mathbb{R}^M} \sum_{j=1}^{J} w_j \sum_{i=1}^{N_j} \left| \boldsymbol{f}_i(\boldsymbol{r}^{N_j}; \boldsymbol{c}) - \boldsymbol{f}_i^{\text{qm}}(\boldsymbol{r}^{N_j}) \right|^2, \tag{4}$$

where $w_j, 1 \le j \le J$, are the user-defined weights with the default value $w_j = 1$. It thus follows that

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}\in\mathbb{R}^M} \sum_{j=1}^{J} w_j \sum_{i=1}^{N_j} \left| \sum_{m=1}^{M} c_m \frac{\partial d_m(\boldsymbol{r}^{N_j})}{\partial \boldsymbol{r}_i} - \boldsymbol{f}_i^{\text{qm}}(\boldsymbol{r}^{N_j}) \right|^2, \tag{5}$$

which can be written as

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}\in\mathbb{R}^M} \sum_{j=1}^{J} \sum_{i=1}^{N_j} \left| \sum_{m=1}^{M} \boldsymbol{d}_{ijm} c_m - \boldsymbol{b}_{ij} \right|^2. \tag{6}$$

Here $\boldsymbol{d}_{ijm} \in \mathbb{R}^3$ and $\boldsymbol{b}_{ij} \in \mathbb{R}^3$ are given by

$$\boldsymbol{d}_{ijm} = w_j^2 \frac{\partial d_m(\boldsymbol{r}^{N_j})}{\partial \boldsymbol{r}_i}, \qquad \boldsymbol{b}_{ij} = w_j^2 \boldsymbol{f}_i^{\text{qm}}(\boldsymbol{r}^{N_j}). \tag{7}$$

Let $\boldsymbol{D}$ be a matrix of size $3\mathcal{N} \times M$ obtained from $\boldsymbol{d}_{ijm}$, and $\boldsymbol{b}$ be a vector of dimension $3\mathcal{N}$ obtained from $\boldsymbol{b}_{ij}$. We can write the above least-squares problem in matrix form as

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}\in\mathbb{R}^M} (\boldsymbol{D}\boldsymbol{c} - \boldsymbol{b})^T (\boldsymbol{D}\boldsymbol{c} - \boldsymbol{b}) \tag{8}$$

which is equivalent to solving the following linear system

$$\boldsymbol{D}^T \boldsymbol{D} \boldsymbol{c}^* = \boldsymbol{D}^T \boldsymbol{b}. \tag{9}$$

This linear system is small enough to be solved using a direct solver.

3.2 Input files

- $\boldsymbol{Z} = (Z_1, \ldots, Z_T)$: 1D array of size T, where Z_t is the atomic number for atom type t.
- $\boldsymbol{N} = (N_j, w_j), j = 1, \ldots, J$: 1D array of size $J \times 2$, where N_j and w_j are the number of atoms and the weight for configuration j, respectively.

- $\boldsymbol{Q} = (q_{ij}, x_{ij}, y_{ij}, z_{ij}, f_{ij}, g_{ij}, h_{ij})$, $i = 1, \ldots, N_j, j = 1, \ldots, J$: 2D array of size $\mathcal{N} \times 7$.
  - $q_{ij} \in [1, T]$ is an integer indicating the type of atom i in configuration j
  - (x_{ij}, y_{ij}, z_{ij}) store the Cartesian coordinates of atom i in configuration j
  - (f_{ij}, g_{ij}, h_{ij}) store the QM forces acting on atom i in configuration j

There are three input files. The main input file contains $\boldsymbol{Z}$ and other parameters/data related to the optimization of the force field. The second input file contains $\boldsymbol{N}$, while the third input file contains $\boldsymbol{Q}$.

It is likely that we will couple the software with a DFT code to produce $\boldsymbol{N}$ and $\boldsymbol{Q}$ automatically. In this case, we won't need the second and third input files.

3.3 Output files

The output files are the same as those described in the previous section

4 Basis Functions

4.1 Power spectrum functions

4.1.1 Basis functions

Based on the quantum theory of angular momentum, we introduce the following functions

$$a_{nklm}(\boldsymbol{r}^N) := \sum_{i}^{\boldsymbol{r}_i \in \Omega_n} u_{klm}(\boldsymbol{r}_i - \boldsymbol{r}_n), \qquad m = -l, \ldots, l, \tag{10}$$

where u_{mlk} are the three-dimensional basic functions

$$u_{klm}(\boldsymbol{r}) = g_{lk}(r) Y_{lm}(\theta, \phi), \quad k \geq 1, l \geq 0, \text{and } m = -l, \ldots, l. \tag{11}$$

Here the spherical harmonics of degree l and order m are given by

$$Y_{lm}(\theta, \phi) = \sqrt{\frac{(2l+1)}{4\pi} \frac{(l-m)!}{(l+m)!}} P_{lm}(\cos(\theta)) e^{im\phi}, \qquad l \geq 0, m = -l, \ldots, l \tag{12}$$

with P_{lm} being the associated Legendre polynomials. Note that $g_{lk}(r)$ are radial basis functions. We will implement a number of different types of radial basis functions including polynomials, spherical Bessel functions, Gaussian functions, etc.

These functions allow us to construct a set of invariant functions as follows:

$$p_{nkk'l}(\boldsymbol{r}^N) = \left(\boldsymbol{a}_{nkl}(\boldsymbol{r}^N)\right)^{\mathrm{H}} \boldsymbol{a}_{nk'l}(\boldsymbol{r}^N), \qquad 1 \leq n \leq N, 1 \leq k \leq K, 0 \leq l \leq L, k' \geq k, \tag{13}$$

where $\boldsymbol{a}_{nkl}(\boldsymbol{r}^N)$ and $\boldsymbol{a}_{nk'l}(\boldsymbol{r}^N)$ are vectors of dimension $2l + 1$ from (10). Although they are invariant with respect to permutation, translation, and rotation, the number of such

functions can be too large to allow for an efficient construction of interatomic potentials. To reduce the number of functions, we perform a sum of $p_{nkk'l}(\boldsymbol{r}^N)$ over all atoms of the same type to obtain the following functions

$$d_{jkk'l}(\boldsymbol{r}^N) = \sum_{n}^{Z_n=j} p_{nkk'l}(\boldsymbol{r}^N), \qquad 1 \le j \le N_T, 1 \le k \le K, 0 \le l \le L, k' \ge k, \tag{14}$$

where the sum is performed for all atoms of type j. We recall that $Z_n \in [1, N_T]$ is an integer number indicating the type of atom n, where N_T is the number of atom types.

4.1.2 Partial derivatives of basis functions for single atom type

- i: atom i
- Ω_i: a list of neighbors of atom i
- N_i^{nb}: number of neighbors of atom i (the size of the list Ω_i)
- j: neighbor of atom i

For single atom type (i.e, $N_T = 1$), the basis functions are given by

$$d_{kk'l}(\boldsymbol{r}^N) = \sum_{i=1}^{N} \sum_{m=-l}^{l} \left(\left(\sum_{j \in \Omega_i} u_{klm}(\boldsymbol{r}_{ji}) \right) \left(\sum_{j \in \Omega_i} u_{k'lm}(\boldsymbol{r}_{ji}) \right) \right) \tag{15}$$

For any given atom n we note that

$$\partial \left(\sum_{j \in \Omega_i} u_{klm}(\boldsymbol{r}_{ji}) \right) / \partial \boldsymbol{r}_n = \begin{cases} u_{klm}(\boldsymbol{r}_{ni}) / \partial \boldsymbol{r}_{ni} & \text{if } n \in \Omega_i \quad \text{(neighbor)} \\ -u_{klm}(\boldsymbol{r}_{jn}) / \partial \boldsymbol{r}_{jn} & \text{if } n = i \quad \text{(self)} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

corresponding to three cases: (1) $n \in \Omega_i$ (i.e, n is a neighbor of atom i), (2) $n = i$ (i.e., n is self and j is its neighbor), (3) $n \notin \Omega_i$ and $n \neq i$. Hence, we have

$$\begin{aligned} &\partial d_{kk'l}(\boldsymbol{r}^N) / \partial \boldsymbol{r}_n = \\ &\underbrace{\sum_{i \in \Omega_n} \sum_{m=-l}^{l} \partial u_{klm}(\boldsymbol{r}_{ni}) / \partial \boldsymbol{r}_{ni} \left(\sum_{j \in \Omega_i} u_{k'lm}(\boldsymbol{r}_{ji}) \right) + \sum_{i \in \Omega_n} \sum_{m=-l}^{l} \left(\sum_{j \in \Omega_i} u_{klm}(\boldsymbol{r}_{ji}) \right) \partial u_{k'lm}(\boldsymbol{r}_{ni}) / \partial \boldsymbol{r}_{ni}}_{\text{atom } i \text{ is self and atom } n \text{ is its neighbor}} \\ &\underbrace{- \sum_{m=-l}^{l} \sum_{j \in \Omega_n} \partial u_{klm}(\boldsymbol{r}_{jn}) / \partial \boldsymbol{r}_{jn} \left(\sum_{j \in \Omega_n} u_{k'lm}(\boldsymbol{r}_{jn}) \right) - \sum_{m=-l}^{l} \left(\sum_{j \in \Omega_n} u_{klm}(\boldsymbol{r}_{jn}) \right) \sum_{j \in \Omega_n} \partial u_{k'lm}(\boldsymbol{r}_{jn}) / \partial \boldsymbol{r}_{jn}}_{\text{atom } n \text{ is self and atom } j \text{ is its neighbor}} \end{aligned} \tag{17}$$

This equation can be written as follows

$$\partial d_{kk'l}(\boldsymbol{r}^N)/\partial \boldsymbol{r}_n = \underbrace{\sum_{i\in\Omega_n} p^{\partial}_{nikk'l}(\boldsymbol{r}^N)}_{\text{atom } i \text{ is self and atom } n \text{ is its neighbor}} - \underbrace{\sum_{j\in\Omega_n} p^{\partial}_{jnkk'l}(\boldsymbol{r}^N)}_{\text{atom } n \text{ is self and atom } j \text{ is its neighbor}} \tag{18}$$

where

$$p^{\partial}_{nikk'l}(\boldsymbol{r}^N) = \sum_{m=-l}^{l} \partial u_{klm}(\boldsymbol{r}_{ni})/\partial \boldsymbol{r}_{ni} \left(\sum_{j\in\Omega_i} u_{k'lm}(\boldsymbol{r}_{ji})\right) + \sum_{m=-l}^{l} \left(\sum_{j\in\Omega_i} u_{klm}(\boldsymbol{r}_{ji})\right) \partial u_{k'lm}(\boldsymbol{r}_{ni})/\partial \boldsymbol{r}_{ni} \tag{19}$$

for $n \in \Omega_i$ and $i = 1, 2, \ldots, N$.

4.1.3 Partial derivatives of basis functions for multiple atom types

For multiple atom types (i.e, $N_T > 1$), the basis functions are given by

$$d_{tkk'l}(\boldsymbol{r}^N) = \sum_{i}^{Z_i=t} \sum_{m=-l}^{l} \left(\left(\sum_{j\in\Omega_i} u_{klm}(\boldsymbol{r}_{ji})\right) \left(\sum_{j\in\Omega_i} u_{k'lm}(\boldsymbol{r}_{ji})\right) \right), \qquad t = 1, \ldots, N_T. \tag{20}$$

For any given atom n we note that

$$\partial \left(\sum_{j\in\Omega_i} u_{klm}(\boldsymbol{r}_{ji})\right) /\partial \boldsymbol{r}_n = \begin{cases} u_{klm}(\boldsymbol{r}_{ni})/\partial \boldsymbol{r}_{ni} & \text{if } n \in \Omega_i \quad \text{(neighbor)} \\ -u_{klm}(\boldsymbol{r}_{jn})/\partial \boldsymbol{r}_{jn} & \text{if } n = i \quad \text{(self)} \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

corresponds to three different cases: (1) $n \in \Omega_i$ (i.e, n is a neighbor of atom i), (2) $n = i$ (i.e., n is self and j is its neighbor), (3) $n \notin \Omega_i$ and $n \neq i$. Hence, we have

$$\begin{aligned} &\partial d_{tkk'l}(\boldsymbol{r}^N)/\partial \boldsymbol{r}_n = \\ &\underbrace{\sum_{i\in\Omega_n}^{Z_i=t} \sum_{m=-l}^{l} \partial u_{klm}(\boldsymbol{r}_{ni})/\partial \boldsymbol{r}_{ni} \left(\sum_{j\in\Omega_i} u_{k'lm}(\boldsymbol{r}_{ji})\right) + \sum_{i\in\Omega_n}^{Z_i=t} \sum_{m=-l}^{l} \left(\sum_{j\in\Omega_i} u_{klm}(\boldsymbol{r}_{ji})\right) \partial u_{k'lm}(\boldsymbol{r}_{ni})/\partial \boldsymbol{r}_{ni}}_{\text{atom } i \text{ is self and of type } t\text{, while atom } n \text{ is its neighbor}} \\ &\underbrace{- \sum_{m=-l}^{l} \sum_{j\in\Omega_n} \partial u_{klm}(\boldsymbol{r}_{jn})/\partial \boldsymbol{r}_{jn} \left(\sum_{j\in\Omega_n} u_{k'lm}(\boldsymbol{r}_{jn})\right) - \sum_{m=-l}^{l} \left(\sum_{j\in\Omega_n} u_{klm}(\boldsymbol{r}_{jn})\right) \sum_{j\in\Omega_n} \partial u_{k'lm}(\boldsymbol{r}_{jn})/\partial \boldsymbol{r}_{jn}}_{\text{atom } n \text{ is self and of type } t\text{, while atom } j \text{ is its neighbor}} \end{aligned} \tag{22}$$

for $t = 1, \ldots, N_T$. This equation can be written as follows

$$\partial d_{tkk'l}(\boldsymbol{r}^N)/\partial \boldsymbol{r}_n = \underbrace{\sum_{i\in\Omega_n}^{Z_i=t} p^{\partial}_{nikk'l}(\boldsymbol{r}^N)}_{\text{atom } i \text{ is self and atom } n \text{ is its neighbor}} - \underbrace{\sum_{j\in\Omega_n}^{Z_n=t} p^{\partial}_{jnkk'l}(\boldsymbol{r}^N)}_{\text{atom } n \text{ is self and atom } j \text{ is its neighbor}} \tag{23}$$

for $t = 1, \ldots, N_T$, where

$$p^{\partial}_{nikk'l}(\boldsymbol{r}^N) = \sum_{m=-l}^{l} \partial u_{klm}(\boldsymbol{r}_{ni})/\partial \boldsymbol{r}_{ni} \left(\sum_{j \in \Omega_i} u_{k'lm}(\boldsymbol{r}_{ji}) \right) + \sum_{m=-l}^{l} \left(\sum_{j \in \Omega_i} u_{klm}(\boldsymbol{r}_{ji}) \right) \partial u_{k'lm}(\boldsymbol{r}_{ni})/\partial \boldsymbol{r}_{ni} \tag{24}$$

for $n \in \Omega_i$ and $i = 1, 2, \ldots, N$.

4.1.4 Implementation

Let Ω_i be a list of N_i^{nb} neighbors of atom i. Together $\Omega_i, 1 \le i \le N$, form the full neighbor lists for the MD system of N atoms. Given the atom positions $\{\boldsymbol{r}_i\}_{i=1}^N$ in the simulation domain, the cut-off radius r_{cut}, and periodic boundary conditions, the list Ω_i associated with an atom i can be constructed by finding all atoms j such that $\|\tilde{\boldsymbol{r}}_j - \boldsymbol{r}_i\| \le r_{\text{cut}}$, where $\tilde{\boldsymbol{r}}_j$ denotes the minimum image of $\boldsymbol{r}_j$ relative to atom i. (To find the minimum image $\tilde{\boldsymbol{r}}_j$, we find periodic images of $\boldsymbol{r}_j$ and calculate the distances between those periodic images and $\boldsymbol{r}_i$. Then the minimum image $\tilde{\boldsymbol{r}}_j$ is found as the image having the smallest distance.) Assume that there are N_i^{nb} indices $\{j_1^i, j_2^i, \ldots, j_{N_i}^i\}$ satisfying $\|\tilde{\boldsymbol{r}}_{j_q^i} - \boldsymbol{r}_i\| \le r_{\text{cut}}, 1 \le q \le N_i^{\text{nb}}$. Thus we obtain the full neighbor lists $\Omega_i = [j_1^i, j_2^i, \ldots, j_{N_i}^i], 1 \le i \le N$, and the following $\mathcal{N}^{\text{nb}} \times 3$ matrix

$$\boldsymbol{x}_{qi} = \tilde{\boldsymbol{r}}_{j_q^i} - \boldsymbol{r}_i, \quad 1 \le q \le N_i^{\text{nb}}, 1 \le i \le N, \tag{25}$$

where $\mathcal{N}^{\text{nb}} = \sum_{i=1}^N N_i^{\text{nb}}$ is the total number of neighbors in the full neighbor lists.

Next, we compute the basis functions

$$d_{tkk'l}(\boldsymbol{r}^N) = \sum_{i}^{Z_i=t} \sum_{m=-l}^{l} \left(\left(\sum_{q=1}^{N_i^{\text{nb}}} u_{klm}(\boldsymbol{x}_{qi}) \right) \left(\sum_{q=1}^{N_i^{\text{nb}}} u_{k'lm}(\boldsymbol{x}_{qi}) \right) \right) \tag{26}$$

and the partial derivatives of the power spectrum components

$$p^{\partial}_{qikk'l}(\boldsymbol{r}^N) = \sum_{m=-l}^{l} \partial u_{klm}(\boldsymbol{x}_{qi})/\partial \boldsymbol{x}_{qi} \left(\sum_{j=1}^{N_i^{\text{nb}}} u_{k'lm}(\boldsymbol{x}_{ji}) \right) + \sum_{m=-l}^{l} \left(\sum_{j=1}^{N_i^{\text{nb}}} u_{klm}(\boldsymbol{x}_{ji}) \right) \partial u_{k'lm}(\boldsymbol{x}_{ni})/\partial \boldsymbol{x}_{ni} \tag{27}$$

for $1 \le i \le N$ and $1 \le q \le N_i^{\text{nb}}$. Finally, we compute the partial derivatives of the basis functions as follows

$$\partial d_{tkk'l}(\boldsymbol{r}^N)/\partial \boldsymbol{r}_n = \underbrace{\sum_{q=1}^{N_n^{\text{nb}}, Z_{i'}=t} p^{\partial}_{q'i'kk'l}(\boldsymbol{r}^N)}_{\text{atom } n \text{ is neighbor}} - \underbrace{\sum_{q=1}^{N_n^{\text{nb}}, Z_n=t} p^{\partial}_{qnkk'l}(\boldsymbol{r}^N)}_{\text{atom } n \text{ is self}} \tag{28}$$

where $i' = j_q^n$ and q' is an index such that $j_{q'}^{i'} = n$. The index q' can be found by iterating the list $\Omega_{i'}$ as follows: For $q' = 1, 2, \ldots, N_{i'}^{\text{nb}}$ if $(j_{q'}^{i'} == n)$ then break.

4.2 Bispectrum functions

We begin by computing the bispectrum components as

$$b_{nkk'll_1l_2}(\boldsymbol{r}^N) = \sum_{m=-l_1}^{l} \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} \bar{a}_{nklm}(\boldsymbol{r}^N) C^{l_1l_2l}_{m_1m_2m} a_{nk'l_1m_1}(\boldsymbol{r}^N) a_{nk'l_2m_2}(\boldsymbol{r}^N), \tag{29}$$

where the functions $a_{nklm}(\boldsymbol{r}^N)$ are defined in (10). The bispectrum components are invariant with respect to permutation of atoms of the same type, translation and rotation of the reference frame. Therefore, the following functions

$$d_{jkk'll_1l_2}(\boldsymbol{r}^N) = \sum_{n}^{Z_n=j} b_{nkk'll_1l_2}(\boldsymbol{r}^N), \quad 1 \le j \le N_T, 1 \le k \le K, 0 \le l_1, l_2, l \le L, k' \ge k, \tag{30}$$

also have the same invariant properties. The number of functions for the bispectrum is $N_T K(K+1)(L+1)^3/2$, which is significantly higher than that for the power spectrum. However, due to the symmetry properties of the Clebsch–Gordan coefficients and the spherical harmonic functions, we can reduce the number of functions substantially by removing redundant bispectrum components.