

Towards MDP.jl: The Julia Library of MD Potentials

April 5, 2021

MDP.jl: the Julia library of MD potentials

Descriptor and force calculation

Next steps...

- ▶ **Molecular Dynamics (MD) simulations require a potential energy function** to describe the force field governing the interaction among atoms.

MDP.jl: the Julia library of MD potentials

- ▶ **Molecular Dynamics (MD) simulations require a potential energy function** to describe the force field governing the interaction among atoms.
- ▶ Force calculation often takes between 50% and 90% of the overall complexity, thus, **determining an adequate potential is a crucial task.**

MDP.jl: the Julia library of MD potentials

- ▶ **Molecular Dynamics (MD) simulations require a potential energy function** to describe the force field governing the interaction among atoms.
- ▶ Force calculation often takes between 50% and 90% of the overall complexity, thus, **determining an adequate potential is a crucial task.**
- ▶ MDP.jl will provide **fast and accurate potentials for classical MD simulations on exascale supercomputers.**
 - Coupling empirical and machine learning (ML) potentials
 - Quantifying uncertainties in trained ML potentials.
 - Identifying near-optimal configurations to include in the training data.

Potential energy functions

- ▶ Potential energy functions are determined based on **quantum information**, e.g. Density Functional Theory (DFT) data.

Potential energy functions

- ▶ Potential energy functions are determined based on **quantum information**, e.g. Density Functional Theory (DFT) data.
- ▶ Classical MD simulations use
 - **Empirical Potentials** (EP), such as Lennard-Jones or Tersoff.
 - **Machine Learning Potentials** (MLP), such as Neural Network Potentials (NNP) or Spectral Neighborhood Analysis Potentials (SNAP).

Potential energy functions

- ▶ Potential energy functions are determined based on **quantum information**, e.g. Density Functional Theory (DFT) data.
- ▶ Classical MD simulations use
 - **Empirical Potentials** (EP), such as Lennard-Jones or Tersoff.
 - **Machine Learning Potentials** (MLP), such as Neural Network Potentials (NNP) or Spectral Neighborhood Analysis Potentials (SNAP).
- ▶ **Present limitations** when solving complex study cases as ultrahigh temperature ceramics in hypersonic flows.
 - Existing EP, ReaxFF and COMB, do not produce satisfactory results since they **require retraining**.
 - MLP **demand a significant amount of DFT data** for the training process.

Coupling Empirical and ML potentials

- ▶ Coupling both potentials is a promising approach for
 - **Reducing the amount of training data**
 - **Leveraging physics information coded in the EP**

Coupling Empirical and ML potentials

- ▶ Coupling both potentials is a promising approach for
 - **Reducing the amount of training data**
 - **Leveraging physics information coded in the EP**
- ▶ Optimization problem

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

Coupling Empirical and ML potentials

- ▶ Coupling both potentials is a promising approach for
 - **Reducing the amount of training data**
 - **Leveraging physics information coded in the EP**
- ▶ Optimization problem

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

$$\mathbf{f}(\mathbf{r}^{N_j}, \mathbf{c}, j, i) = \mathbf{f}_E(\mathbf{r}^{N_j}, \mathbf{c}, j, i) + \mathbf{f}_{ML}(\mathbf{r}^{N_j}, \mathbf{c}, j, i)$$

Empirical component of the force

Empirical component of the force:

$$\mathbf{f}_E(\mathbf{r}^{N_j}, \mathbf{c}, j, i) = \mathbf{f}_{E,PS}(\mathbf{r}^{N_j}, \mathbf{c}, j, i) + \mathbf{f}_{E,BS}(\mathbf{r}^{N_j}, \mathbf{c}, j, i)$$

Empirical component of the force

Empirical component of the force:

$$\mathbf{f}_E(\mathbf{r}^{N_j}, \mathbf{c}, j, i) = \mathbf{f}_{E,PS}(\mathbf{r}^{N_j}, \mathbf{c}, j, i) + \mathbf{f}_{E,BS}(\mathbf{r}^{N_j}, \mathbf{c}, j, i)$$

The **power spectrum** component (long range interaction):

$$\mathbf{f}_{E,PS}(\mathbf{r}^{N_j}, \mathbf{c}, j, i) = \sum_{t=1}^{N_z} \sum_{k=1}^K \sum_{k'=k}^K \sum_{l=0}^L c_{tkk'l} \frac{\partial d_{tkk'l}^{PS}(\mathbf{r}^{N_j}, j, i)}{\partial \mathbf{r}_i^{N_j}}$$

Empirical component of the force

Empirical component of the force:

$$\mathbf{f}_E(\mathbf{r}^{N_j}, \mathbf{c}, j, i) = \mathbf{f}_{E,PS}(\mathbf{r}^{N_j}, \mathbf{c}, j, i) + \mathbf{f}_{E,BS}(\mathbf{r}^{N_j}, \mathbf{c}, j, i)$$

The **power spectrum** component (long range interaction):

$$\mathbf{f}_{E,PS}(\mathbf{r}^{N_j}, \mathbf{c}, j, i) = \sum_{t=1}^{N_z} \sum_{k=1}^K \sum_{k'=k}^K \sum_{l=0}^L c_{tkk'l} \frac{\partial d_{tkk'l}^{PS}(\mathbf{r}^{N_j}, j, i)}{\partial \mathbf{r}_i^{N_j}}$$

The **bispectrum** component (short range interactions):

$$\mathbf{f}_{E,BS}(\mathbf{r}^{N_j}, \mathbf{c}, j, i) = \sum_{t=1}^{N_z} \sum_{k=1}^K \sum_{k'=k}^K \sum_{l=0}^L \sum_{l_1=0}^L \sum_{l_2=0}^L c_{tkk'l l_1 l_2} \frac{\partial d_{tkk'l l_1 l_2}^{BS}(\mathbf{r}^{N_j}, j, i)}{\partial \mathbf{r}_i^{N_j}}$$

Empirical potential: PS basis functions

Derivative of the power spectrum basis functions

$$\frac{\partial d_{tkk'l}^{BS}(\mathbf{r}^{N_j}, j, i)}{\partial \mathbf{r}_i^{N_j}} = \sum_{s \in \Omega'_{jit}} p_{iskk'l}^{\partial}(\mathbf{r}^{N_j}, j) - \sum_{s \in \Omega''_{jit}} p_{sikk'l}^{\partial}(\mathbf{r}^{N_j}, j),$$

$$p_{i_0 i_1 k k' l}^{\partial}(\mathbf{r}^{N_j}, j) = \sum_{m=-l}^l \left(\frac{\partial u_{klm}(\mathbf{r}_{i_0}^{N_j} - \mathbf{r}_{i_1}^{N_j})}{\partial (\mathbf{r}_{i_0}^{N_j} - \mathbf{r}_{i_1}^{N_j})} \sum_{s \in \Omega_{j, i_1}} (u_{k'lm}(\mathbf{r}_s^{N_j} - \mathbf{r}_{i_1}^{N_j})) \right) +$$
$$\sum_{m=-l}^l \left(\frac{\partial u_{k'lm}(\mathbf{r}_{i_0}^{N_j} - \mathbf{r}_{i_1}^{N_j})}{\partial (\mathbf{r}_{i_0}^{N_j} - \mathbf{r}_{i_1}^{N_j})} \sum_{s \in \Omega_{j, i_1}} (u_{klm}(\mathbf{r}_s^{N_j} - \mathbf{r}_{i_1}^{N_j})) \right)$$

Empirical potential: PS basis functions

- The descriptors are based on products of radial basis functions $g_{lk}(r)$ and spherical harmonics $Y_{lm}(\theta, \phi)$.

$$u_{klm}(\mathbf{r}) = g_{lk}(r) Y_{lm}(\theta, \phi)$$

Empirical potential: PS basis functions

- ▶ The descriptors are based on products of radial basis functions $g_{lk}(r)$ and spherical harmonics $Y_{lm}(\theta, \phi)$.

$$u_{klm}(\mathbf{r}) = g_{lk}(r) Y_{lm}(\theta, \phi)$$

- ▶ $g_{lk}(r)$ can vary...
 - Spherical Bessel
 - Polynomials, Gaussian functions, etc.

Empirical potential: PS basis functions

- ▶ The descriptors are based on products of radial basis functions $g_{lk}(r)$ and spherical harmonics $Y_{lm}(\theta, \phi)$.

$$u_{klm}(\mathbf{r}) = g_{lk}(r) Y_{lm}(\theta, \phi)$$

- ▶ $g_{lk}(r)$ can vary...
 - Spherical Bessel
 - Polynomials, Gaussian functions, etc.
- ▶ Providing flexibility when calculating the derivative of $u_{klm}(\mathbf{r})$ using the finite difference method (FDM)

$$\frac{\partial u_{klm}(\mathbf{r})}{\partial(\mathbf{r})} = \left(\frac{u_{klm}(\mathbf{r} + \Delta\mathbf{x}) - u_{klm}(\mathbf{r} - \Delta\mathbf{x})}{2|\Delta\mathbf{x}|}, \dots, \dots \right)$$

Empirical potential: BS basis functions

Derivative of the bispectrum basis functions

$$\frac{\partial d_{tkk'lh_1l_2}^{BS}(\mathbf{r}^{N_j}, j, i)}{\partial \mathbf{r}_i^{N_j}} = \left(\frac{d_{tkk'lh_1l_2}(\mathbf{r}^{N_j} + \Delta \mathbf{X}_i, j, i) - d_{tkk'lh_1l_2}(\mathbf{r}^{N_j} - \Delta \mathbf{X}_i, j, i)}{2|\Delta \mathbf{X}_i|}, \dots, \dots \right)$$

Empirical potential: BS basis functions

Derivative of the bispectrum basis functions

$$\frac{\partial d_{tkk' l l_1 l_2}^{BS}(\mathbf{r}^{N_j}, j, i)}{\partial \mathbf{r}_i^{N_j}} = \left(\frac{d_{tkk' l l_1 l_2}^{BS}(\mathbf{r}^{N_j} + \Delta \mathbf{X}_i, j, i) - d_{tkk' l l_1 l_2}^{BS}(\mathbf{r}^{N_j} - \Delta \mathbf{X}_i, j, i)}{2|\Delta \mathbf{X}_i|}, \dots, \dots \right)$$

The bispectrum basis functions are formulated as:

$$d_{tkk' l l_1 l_2}^{BS}(\mathbf{r}^{N_j}, j, i) = \sum_{s \in \Omega_{jt}'''} b_{skk' l l_1 l_2}(\mathbf{r}^{N_j}, j, i)$$

$$b_{skk' l l_1 l_2}(\mathbf{r}^{N_j}, j, i) = \sum_{m=-l}^l \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} \bar{a}_{sklm}(\mathbf{r}^{N_j}, j) C_{m_1 m_2 m}^{l_1 l_2 l} a_{sk' l_1 m_1}(\mathbf{r}^{N_j}, j) a_{sk' l_2 m_2}(\mathbf{r}^{N_j}, j)$$

$$a_{iklm}(\mathbf{r}^{N_j}, j) = \sum_{s \in \Omega_{ji}} u_{klm}(\mathbf{r}_s^{N_j} - \mathbf{r}_i^{N_j})$$

Next steps

- ▶ Keep my summary of implemented equations up to date ✓

Next steps

- ▶ Keep my summary of implemented equations up to date ✓
- ▶ Optimization
 - Clebsch–Gordan coefficients
 - ▶ Fast calculation using "PartialWaveFunctions.jl" ✓
 - ▶ Stop calculation when a zero is found ✓
 - Calculate only in the neighbors
 - ▶ Precalculated neighbors information ✓
 - ▶ Optimize precalculation
 - Leverage spherical harmonics symetries
 - Threading and GPU

Next steps

- ▶ Keep my summary of implemented equations up to date ✓
- ▶ Optimization
 - Clebsch–Gordan coefficients
 - ▶ Fast calculation using "PartialWaveFunctions.jl" ✓
 - ▶ Stop calculation when a zero is found ✓
 - Calculate only in the neighbors
 - ▶ Precalculated neighbors information ✓
 - ▶ Optimize precalculation
 - Leverage spherical harmonics symetries
 - Threading and GPU
- ▶ Testing: the descriptors are invariant to
 - Rotation ✓
 - Permutation
 - Translation

Next steps

- ▶ Keep my summary of implemented equations up to date ✓
- ▶ Optimization
 - Clebsch–Gordan coefficients
 - ▶ Fast calculation using "PartialWaveFunctions.jl" ✓
 - ▶ Stop calculation when a zero is found ✓
 - Calculate only in the neighbors
 - ▶ Precalculated neighbors information ✓
 - ▶ Optimize precalculation
 - Leverage spherical harmonics symetries
 - Threading and GPU
- ▶ Testing: the descriptors are invariant to
 - Rotation ✓
 - Permutation
 - Translation
- ▶ ML potentials

Next steps

- ▶ Keep my summary of implemented equations up to date ✓
- ▶ Optimization
 - Clebsch–Gordan coefficients
 - ▶ Fast calculation using "PartialWaveFunctions.jl" ✓
 - ▶ Stop calculation when a zero is found ✓
 - Calculate only in the neighbors
 - ▶ Precalculated neighbors information ✓
 - ▶ Optimize precalculation
 - Leverage spherical harmonics symetries
 - Threading and GPU
- ▶ Testing: the descriptors are invariant to
 - Rotation ✓
 - Permutation
 - Translation
- ▶ ML potentials
- ▶ Integration with NBodySimulator and LAMMPS

MDP.jl + NBodySimulator.jl

Argon study case.

1. Run MDP.jl:

- Input (from the initial condition):
 - ▶ Force field (calculated through LJ, thus, the initial force field). Atomic positions.
- Output:
 - ▶ Force field (this fitted function will be used in the whole MD simulation)

2. Run NBodySimulator.jl

For each time step:

- Input (from the initial condition):
 - ▶ Force field from MDP.jl (the fitted force field expression is evaluated here). Atomic Velocities. Domain(box). Temperature. Mass. External forces.
- Output:
 - ▶ Atom positions. Atomic velocities. Temperature. Energy (kinetic, potential, and total). Radial distribution function

Argon study case.

1. Run NBodySimulator.jl

For each time step:

- Input: Quantum force from DFTK.jl
 - ▶ DFTK.jl input: Atomic positions from NBodySimulator.jl
 - ▶ DFTK.jl output: Quantum force (a single set of forces evaluated at one set of positions). Atomic positions. Atomic velocities. Domain (box). Temperature. Mass. External forces
- Output: Atom positions. Atomic velocities. Temperature. Energy (kinetic, potential, and total). Radial distribution function.

References

- ▶ “MDP.jl: The Julia Library of Molecular Dynamics Potentials”. MIT PSAAP-3 Team. 2021.
- ▶ “Accelerating Force Calculation in Molecular Dynamics with Transition to LAMMPS”. MIT PSAAP-3 Team. 2021.
- ▶ MDP. <https://github.com/cesmix-mit/MDP.jl>
- ▶ Fortran and Ar MD simulation.
https://ase.tufts.edu/chemistry/lin/outreach_FortranMD.html
- ▶ NBodySimulator. <https://github.com/SciML/NBodySimulator.jl>
- ▶ Molly. <https://github.com/JuliaMolSim/Molly.jl>
- ▶ Argon study case using NBodySimulator.
https://github.com/jrdegreff/cesmix-julia/blob/main/notebooks/nbodysimulator_argon_simulation.jl
- ▶ Draft DFTK+Molly / MDP+Molly. https://docs.google.com/document/d/1APQ1JjL0SoQTTsDam5j3lEKe3xCDwNzrbRIGNH_JzxU/edit?usp=sharing

Thanks :-)