DL_FFLUX: a parallel, quantum chemical topology force field

Benjamin C. B. Symons[1, 2], Michael Bane[3, 4] and Paul L. A. Popelier[1, 2]

[1] Manchester Institute of Biotechnology (MIB), 131 Princess Street, Manchester M1 7DN, Great Britain

[2] Department of Chemistry, University of Manchester, Oxford Road, Manchester M13 9PL, Great Britain

[3] High End Compute LTD, https://highendcompute.co.uk

[4] Department of Computer Science, University of Liverpool, Ashton Street, Liverpool L39 3BX, Great Britain

Abstract

DL_FFLUX is a force field based on quantum chemical topology that can perform molecular dynamics for flexible molecules endowed with polarisable atomic multipole moments (up to hexadecapole). Using the machine learning method kriging (aka Gaussian Process Regression) DL_FFLUX has access to atomic properties (energy, charge, dipole moment, …) with quantum mechanical accuracy. Newly optimised and parallelised using domain-decomposition MPI, DL_FFLUX is now able to deliver this rigorous methodology at scale while still in reasonable time frames. DL_FFLUX is delivered as an add-on to the widely distributed molecular dynamics code DL_POLY 4.08. For the systems studied here (10^3-10^5 atoms), DL_FFLUX is shown to add minimal computational cost to the standard DL_POLY package. In fact, the optimisation of the electrostatics in DL_FFLUX means that, when high rank multipole moments are enabled, DL_FFLUX is up to 1.25x faster than standard DL_POLY. The parallel DL_FFLUX preserves the quality of the scaling of the MPI implementation in standard DL_POLY. For the first time it is feasible to use the full capability of DL_FFLUX to study systems that are large enough to be of real world interest. For example, a fully flexible, high-rank polarised (up to and including quadrupole moments) 1 ns simulation of a system of 10,125 atoms (3,375 water molecules) takes 30 hours (wall time) on 18 cores.

1. Introduction

A computational chemistry code needs to compete along two axes: accuracy and speed. Generally, an improvement in one comes at the expense of the other. Classical force fields are well known to be fast but often lack accuracy[1,2]. *Ab initio* methods typically achieve greater accuracy than classical force fields but this accuracy comes at a high computational cost. This cost limits *ab initio* methods to relatively small systems or short simulation time scales. Thus, a holy grail of computational chemistry is finding a method[3-17], often involving machine learning nowadays, that is fast and accurate, the equivalent of "having one's cake and eat it".

DL_FFLUX is a force field that aims to perform accurate Molecular Dynamics (MD) calculations without sacrificing speed too much. The accuracy of the FFLUX methodology has been demonstrated at various levels of analysis from single molecules[18-21] to small clusters of molecules[22] and even ions[23]. However, before the advances presented in the current paper, the prohibitive computational cost of DL_FFLUX made 'bulk' simulations impossible. The current work describes, in detail, a step change towards making DL_FFLUX a practical but more reliable alternative to well-known and popular force fields.

The accomplishment of DL_FFLUX up to now can be attributed to its rigorous theoretical foundation[24] and design from scratch. At the heart of DL_FFLUX is the quantum topological atom offered by the Quantum Theory of Atoms in Molecules (QTAIM)[25], which led to the energy partitioning scheme called Interacting Quantum Atoms (IQA)[26]. Both are part of Quantum Chemical Topology (QCT), a term coined[27] in 2003 to refer to the collection of approaches[28] sharing the idea of a (gradient) vector field partitioning a quantum mechanical function. If this function is the electron density, then the (quantum) topological atom emerges.

The computational cost associated with QCT is mitigated in two ways. Firstly, the results of QCT calculations are used to train machine learning models, specifically, kriging or Gaussian process regression[29,30] method of machine learning. After compute-intensive training, these kriging models can then quickly predict atomic energies and multipole moments, using only the nuclear coordinates of the given atom's environment. The predictions occur in real time, during the course of a MD simulation, at considerably less cost than performing quantum mechanical calculations. Moreover, kriging needs fewer data to reach a given accuracy compared[11,31] compared to neural nets. It should also be mentioned here that polarisation is taken care of without the need for on-the-fly dipole moment iterations using polarisabilities. Secondly, DL_FFLUX is now parallelised with Message Passing Interface (MPI)[32] so it can take advantage of High Performance Computing hardware.

In summary, DL_FFLUX can perform MD simulations with fully flexible molecules, and polarisable atomic multipole moments (up to hexadecapole moment). Molecular flexibility and polarisation are achieved without recourse to the usual means of harmonic potentials[33], Drude oscillators[34] or other methods[35] [36]. However, in DL_FFLUX, predicted (intramolecular) potential energy surfaces and atomic multipole moments handle molecular flexibility, polarisation and intramolecular charge transfer. All this information is captured by the kriging models trained on an effective paucity of computationally expensive QCT data. The advance presented here is that DL_FFLUX is currently optimised and parallelised such that QCT data are now available during a MD simulation at relatively low extra cost.

2. Theoretical Background

2.1. QTAIM

The gradient of the electron density, $\nabla\rho(\vec{r})$, defines a topological atom as a naturally emerging subspace. Each atom comprises an attractor (the nucleus) and a particular portion of the total electron density. Each nucleus attracts a bundle of trajectories of the gradient of the electron density, which are commonly referred to as gradient paths. The subspace spanned by this bundle is the topological atom. The boundaries of an atom are interatomic surfaces, which are surfaces of zero-flux of $\nabla\rho(\vec{r})$. In other words, each atom inside a molecule is bounded by surfaces that obey eq 1,

$$\nabla\rho(\vec{r})\cdot\vec{n}(\vec{r})=0 \quad \forall\vec{r}\in S \tag{1}$$

The constraint in eq 1 defines a surface that is not crossed by any gradient paths, which is clear from Figure 1 where a water dimer complex illustrates various topological objects.

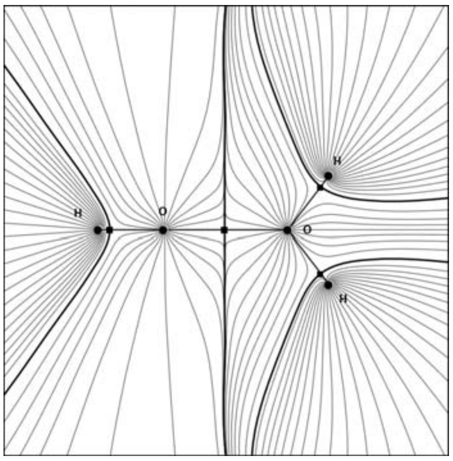

Figure 1. Gradient paths for a water dimer. The circles represent nuclei (attractors). Thick black lines are interatomic surfaces, and the squares that lie on these lines are so-called bond critical points. The second hydrogen of the left water is perpendicular to the plotting plane and not shown.

Where an atom is not bordered by another atom, its electron density extends out to infinity. However, in order to have a practical boundary to an atom at the edge of a molecule, the atom can be capped by an arbitrary constant electron density surface. We also note that topological atoms are space-filling: they leave no gaps and do not overlap.

2.2. IQA

This relatively recent scheme decomposes atomic energies in a chemically meaningful way[37], without the problems[38] of older energy decomposition analyses such as NEDA[39]. Inspired by early calculations[40] of the electrostatic energy between topological atoms, IQA extends QTAIM's original virial-based energy partitioning scheme such that atomic energies are also valid for non-equilibrium geometries. IQA is a reference-free and exhaustive partitioning scheme, two properties shown to be favourable[41]. The IQA energy decomposition is achieved by partitioning the one-and two-electron density matrices. However, the derivation[26] is not discussed here, only the results.

Atomic IQA energies can be broken down into intra-atomic and interatomic contributions,

$$E_{IQA}^{A} = E_{Intra}^{A} + \frac{1}{2}\sum_{B \neq A} V_{Inter}^{AB} \tag{2}$$

Both terms in eq 2 can be broken down further as follows,

$$E_{Intra}^{A} = T^{A} + V_{ne}^{AA} + V_{ee}^{AA} \tag{3}$$

$$V_{Inter}^{AB} = V_{ne}^{AB} + V_{en}^{AB} + V_{ee}^{AB} + V_{nn}^{AB} \tag{4}$$

The superscripts in eqs 3 and 4 indicate the two atoms that are interacting with each other, while the subscripts indicate the type of interaction. The intra-atomic energy in eq 3 is comprised of the kinetic energy, T^{A}, nuclear-electron and electron-electron interactions. The interatomic term in eq 4 also contains the nuclear-nuclear interaction. Note that the interatomic electron-nuclear interaction needs to be accounted for both ways: the nucleus of one atom must be paired with the electrons of the other atom and *vice versa*. DL_FFLUX does not see the finer details outlined in eqs 2, 3 and 4. Rather, DL_FFLUX works with E_{IQA}^{A} values, which is the only level of resolution necessary for MD.

2.3. Kriging

Kriging is a machine learning technique that takes inputs (features) and maps them to a single output. In this case the outputs are IQA energies and atomic multipole moments. The features for a given atom are geometric and defined in an atomic local frame (ALF). Three atoms are required to construct the axes of the ALF. The features of atoms that are not used to construct the ALF are comprised of spherical polar coordinates defined with respect to this ALF. Each atom is the center of its own ALF. Full details of this framework can be found in our previous work[19]. The choice of a local frame is significant as it means that all multipole moments are invariant with respect to global rotations and translations.

Kriging makes predictions based on some training set containing N_{train} training points. The training of kriging models is a topic unto itself[20] and is not covered here as it is not relevant. Predictions of a quantity of interest, $\hat{Y}^A$, relating to (topological) atom *A*, are made according to

$$\hat{Y}^A = \mu^A + \sum_{j=1}^{N_{train}} a_j{}^A \exp\left[-\sum_{k=1}^{N_{feat}} \theta_k{}^A \left| f_{k,j}{}^A - f_k{}^A \right|^{p_k{}^A} \right] \tag{5}$$

where μ^A is the average value of the output for all training points, $a_j{}^A$ is the "weight" of the j^{th} training point, $\theta_k{}^A$ and $p_k{}^A$ are hyperparameters optimised during training (although the latter are typically set to "2" without much loss of accuracy). The exponent's argument contains N_{feat}-dimensional vectors of features where the number of features is equal to the dimensionality of the system. Equation 5 essentially finds the correlation between a set of (previously unseen) input features $\vec{f}^A$ and the features of the training data $\vec{f}_j{}^A$ and predicts based on this correlation. From the form of eq 5 follows that, if a set of inputs is 'far away' from any of the training points, the exponent's argument approaches zero and the predicted output returns to the average, μ^A. This is a potential weakness if the feature space has been trained for a narrow sample of configuration space that does not cover the geometries encountered during a simulation.

For the purposes of DL_FFLUX, $\hat{Y}^A$ is either $\hat{E}_{IQA}^A$ or one of the multipole moments, $\{\hat{Q}_{l,m}\}$. In the case of the IQA energy, DL_FFLUX also calculates the forces[42] resulting from these energies. These are the intra-model forces, which for our purposes are synonymous with intramolecular energies. However, it is possible to have models describe and predict quantum system than single molecules; molecular complexes can also be treated by this methodology. The i^{th} component of the force on atom *B* is calculated as

$$F_i^B = -\sum_{A=1}^{N_{atoms}} \sum_{k=1}^{N_{feat}} \frac{d\hat{E}_{IQA}^A}{df_k{}^A} \frac{df_k{}^A}{d\alpha_i{}^B} . \tag{6}$$

Equation 6 is the sum of derivatives of the IQA energy of every atom *A* with respect to the global Cartesian coordinates of atom *B*, $\alpha_i{}^B$. This derivative is obtained via a chain rule involving the features $df_k{}^A$. Note that eq 6 replaces the use of traditional harmonic bond and angle potentials in DL_FFLUX, a hallmark of DL_FFLUX's fresh approach to force field design.

2.4. Electrostatics

Electrostatics in DL_FFLUX is worthy of a brief discussion. Short-range electrostatics are captured by the kriging models and are wrapped up inside E_{IQA}^{A}. Short-range for our purposes means intramolecular. On the other hand, long-range electrostatics (intermolecular) is then calculated using smooth particle mesh Ewald[43] with polarisable moments predicted by kriging models. The rank of the electrostatic interaction is denoted L'. This quantity refers to the highest rank multipole moments present in a given simulation. For example, L'=2 means that monopole (l=0), dipole moments (l=1) and quadrupole moments (l=2) are present and that monopole-monopole, monopole-dipole, monopole-quadrupole, dipole-dipole, dipole-quadrupole and quadrupole-quadrupole interactions are computed. At present FFLUX supports moments up to and including hexadecapole moments (hence L'=4). It has been shown previously that point charge only does not suffice for the convergence of long-range electrostatic interactions in water[44]. The need for high-rank multipolar electrostatics has been observed in the small protein crambin as well[45].

Overall and in summary, DL_FFLUX is a truly new force field, which is much closer to the underlying quantum reality. DL_FFLUX "sees the electrons" and exploits a parameter-free definition of an atom inside a system. Using machine learning, DL_FFLUX learns how atomic energies, charges and multipole moments vary with the surrounding atoms' geometry. As such it captures all polarisation and many-body effects, as well as charge transfer, in one streamlined scheme. The approach avoids perturbation theory and thus benefits from a clear treatment of short-range interactions. Moreover, DL_FFLUX breaks free from the rigid-body constraints of advanced polarisable force fields. The well-defined atom at the heart of DL_FFLUX enables physics-based machine learning. It uses kriging instead of neural nets, thereby reducing the training data size. Finally, we point out DL_FFLUX's modularity: each energy term represents only the physical phenomenon it describes and nothing else e.g. the electrostatic energy is well defined. If this contribution is improved, then the other non-electrostatic terms will not be affected. Hence, DL_FFLUX is systematically improvable; energy reliability does not depend on fortuitous cancellation or compensation between different force field terms, as is the case with classic force fields.

2.5. Domain decomposition

There are several approaches one can take to implementing MPI in a given program. Discussed here is the Domain Decomposition (DD) method[46]. This method was chosen for DL_FFLUX primarily because DD is already implemented in DL_POLY 4.08. The domain decomposition method is relatively simple in principle. In DD MPI, the simulation cell is partitioned into smaller cells called domains such that each of the N_p processes computes some subset of the total system in parallel. Figure 2 demonstrates this concept for a simple 2D system: the space is split in half as there are 2 MPI processes, where *P=0* and *P=1* denote the first and second process, respectively.

In the ideal case, N_p domains would lead to a speed-up given by

$$t_{parallel} = \frac{t_1}{N_p} \tag{7}$$

where t_1 is the time on one single process (i.e. serial). However, in practice eq 7 is complicated by the need for communications between domains. It is also possible that not all of the code is parallelised and thus a more realistic equation for speed-up is

$$t_{parallel} = \frac{t_1 - t_{serial}}{N_p} + t_{serial} + t_{comms}(N_p) \tag{8}$$

where t_{serial} and $t_{comms}(N_p)$ respectively represent the time spent in serial code and in MPI communications, the latter being some possibly non-trivial function of N_p. Serial code is a hard limiting factor on speed-up. For example, if 10 % of the runtime is spent in serial code, then the maximum possible speed-up is a factor of 10 according to Amdahl's Law. This situation can only be improved by parallelising any serial code.

MPI communications are typically necessary when calculations in a given domain require information from neighbouring domain(s) or for synchronisation purposes, among other reasons. In MD simulations any pairwise (or many-body) interactions between particles located on different domains will require some degree of communication. In practice, some particle *i*, located in a given domain, will interact with all particles *j* that satisfy the condition $r_{ij} < r_{cut}$ where $r_{ij} = |\vec{r}_{ij}|$ is the distance between particles *i* and *j*, and r_{cut} is the cut-off radius. Any particle *j* that satisfies the cut-off condition but is not in the same domain as *i* will require a communication between the processes associated with the domains of *j* and *i.* Information such as *j*'s coordinates and multipole moments will have to be communicated. Note that communication is also required in the DD method when a particle crosses the boundary between domains during a simulation.

The way to deal with these communications is to construct a so-called *halo* for each domain, which is shown in Figure 2. A domain's halo contains all the information it needs about neighbouring domains. The data required by a process can then be split into two parts: local and halo. The local part is the information about the particles that reside in a process' domain and the halo is all the extra information needed to compute any interactions between particles in different domains. A halo need only extend a distance r_{cut} beyond a domain's borders because interactions between particles separated by a larger distance are not computed. In practice, a halo typically extends slightly further than r_{cut}.

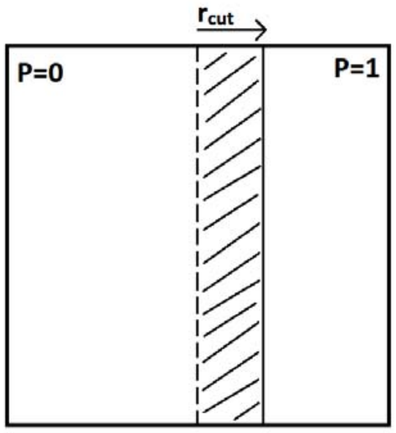

Figure 2. A 2D simulation cell split into two domains. The dashed line represents the divide between the domains. The halo of P=0 is shown by the shaded area.

It is evident from Figure that, the smaller r_{cut}, the smaller the halo. A useful ratio is given by eq 9,

$$\frac{r_{cut}^3}{V_{Domain}} \qquad (9)$$

where V_{Domain} is the volume of a domain. As this ratio decreases, the proportion of particles in a given domain that require communications also decreases. This ratio can be decreased by reducing r_{cut} but this may reduce the accuracy of the simulation. Instead, the size of a domain can be increased. This can be done by increasing the overall system size with a fixed N_p or by reducing N_p for a given system size. For a given system there will come a point when increasing N_p starts to slow down the code due to the increased cost of communications (note the optimal N_p is highly system dependent). Another important factor to consider is load balancing. If the density of the system is highly inhomogeneous then different processes may have significantly different amounts of computation to carry out, i.e. the computational load is imbalanced. If this is the case, then some processes will have to wait for others to finish, which negatively impacts parallel performance. Provided the density is relatively homogeneous then this issue should not occur when using domain decomposition.

It is worth noting that the DD method must be extended if it is combined with periodic boundary conditions (PBCs). In this case the DD halo must be combined with a PBC halo. Instead of containing just particles from neighbouring domains the halo may also include image particles from neighbouring image cells. Figure shows the full halo of the domain associated with *P=0* in the case of MPI and PBCs. The portion of the halo that extends beyond the main simulation cell is the part that contains image particles.

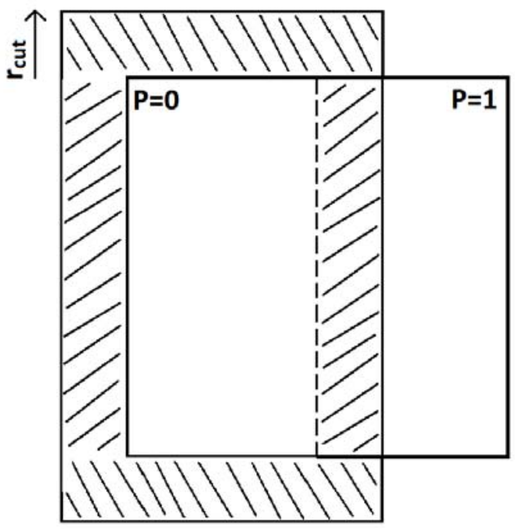

Figure 3. A 2D simulation cell split into two domains. The full halo of P=0 is shown by the shaded area, accounting for PBCs.

3. Code structure

3.1. Overview

DL_FFLUX is a FORTRAN90 code, written as a modular attachment to the code DL_POLY 4.08 [47,48]. Note that some bug fixes from DL_POLY 4.09 have been integrated into DL_FFLUX. DL_FFLUX implements the FFLUX methodology and comprises 17 subroutines as well as some changes to the DL_POLY source code in order to interface the two. DL_FFLUX options can be set in the same way as typical DL_POLY options using the CONTROL file. DL_POLY provides the framework for the MD part of the code, i.e. the parts of an MD simulation that are not changed by DL_FFLUX. The role of DL_FFLUX can be split into two parts. The first part makes predictions using kriging models that have been trained prior to running the MD simulation. In every time step atomic IQA energies and multipole moments (potentially up to hexadecapole moment) are predicted. The multipole moments are fed into Ewald summation subroutines, DL_FFLUX specific, as well as heavily modified versions of the DL_POLY Ewald subroutines. The second part computes the intramolecular forces that result from the gradient of the predicted E_{IQA} potential energy surface. Figure demonstrates a typical time step.

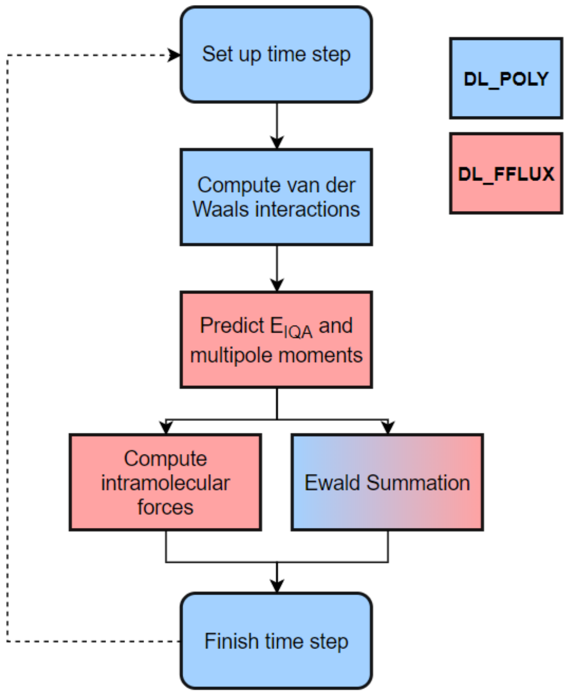
Set up time step
DL_POLY
DL_FFLUX
Compute van der Waals interactions
Predict E_{IQA} and multipole moments
Compute intramolecular forces
Ewald Summation
Finish time step

Figure 4. A flow diagram demonstrating the stages of a typical DL_FFLUX MD time step.

The flow diagram in Figure does not imply that the computation of intramolecular forces and the Ewald summation is carried out simultaneously but rather that the DL_FFLUX predictions feed into both sections of the code. However, the two routines are independent and so they could run concurrently, in principle. Concurrent computation is not explored here but will be the subject of future work. Note that the Ewald summation is a blend of colours as it involves a mixture of DL_FFLUX, and modified DL_POLY, subroutines.

3.2. MPI

3.2.1. DL_POLY

DL_POLY already implements the DD method of MPI. However, the challenge remained of suitably modifying the DL_FFLUX code to be compatible with DL_POLY's existing implementation. For a given time step in a DL_POLY MD simulation, there are several places where calls to MPI routines are present. Most importantly, at the start of a time step the halos must be constructed, and at the end of a time step any particles crossing a domain boundary must be relocated. MPI calls are also needed for the fast Fourier transforms required in the reciprocal space part of the Ewald summation as well as in I/O routines.

Full details of DL_POLY’s parallelisation strategy can be found in the user manual[49] as well as in reference[50]. However, a brief overview is given here. At the beginning of a time step, each domain calculates which of its particles each of its neighbours will require in their halos. The information on these particles is packed into a buffer and sent to the relevant neighbouring domain. Each domain can then build up its own halo from the data it receives from its neighbours. The cost of these communications is minimised by sending data in three waves, that is, in the $\pm x$, $\pm y$ and $\pm z$ directions. Note that the order of these waves does not matter[51]. It is important to mention here that DL_POLY actually divides domains further into link cells. Each link cell has side lengths, l, which is as close as possible to (but always greater than) r_{cut}. Link cells enable the construction of a link list for every atom. The link list is a list of every atom inside a given atom's cut-off radius. The main benefit of the link list method over, for example, the Verlet neighbour list relates to computational cost. The Verlet method computes all pairs of distances, i.e. N(N-1), to construct the list of all pairs of atoms that interact. However, in the link list approach one divides space into so-called link cells approximately of size R_{cut} such that, when constructing the list of atoms that a given atom can interact with, one only has to inspect the neighbouring link cells (26 neighbours in 3D). Computing all pairwise distances for the Verlet method is an $O(N^2)$ operation whereas the link list method turns out to be an approximately $O(N)$ calculation. However, the link cell method only becomes more efficient[51] than the Verlet neighbour list when the number of link cells per dimension is greater than 3 as it is only when this condition is met that the method computes fewer than all pairwise distances.

The final aspect of the DL_POLY DD implementation to consider is the choice of indexing. Each process has a subset of all the relevant property arrays such as positions, velocities, forces etc. Approximately, the subsets will be of size $natms/N_p$. These arrays can either retain global indices or be indexed locally.

In the global case the indices of atoms do not change. However, all loops must be altered so that each process operates only on the indices corresponding to atoms that reside in its domain. DL_POLY uses local indexing instead. In this case, the arrays on each process are indexed from 1 to $nlast$, which is the number of local atoms plus the number of halo atoms on a given process. The variable $natms$, which stores the total number of atoms in the system in the serial case, is redefined on each process to be the number of local atoms in that domain. This means that loops in the main code do not need to be altered when going from serial to MPI, thus preserving a single source code. However, this complicates matters slightly as there needs to be a mapping from local to global indices. It is also useful to have the reverse mapping. This mapping is non-trivial, especially in the case of PBCs because a global atom can appear multiple times in a domain's halo, which means that there may not be a unique mapping from a global index to a single local index. This issue is solved in DL_POLY by storing a list of the local indices and a local list of global indices that has been sorted. A binary search algorithm is then used to search the sorted list.

3.2.2. DL_FFLUX

One of the main parts of DL_FFLUX is the IQA energy prediction and intramolecular force calculation loop. This loop had to be modified to work with DD MPI. The original structure of the loop is shown in Figure 5. There is a loop over all $natms$ local atoms. The IQA energy of atom i, E_{IQA}^{i} is predicted and then the force that is exerted by i on all other atoms is computed in a second, interior loop. The force calculation is subject to the constraint that atom *j* , the atom that atom *i* exerts a force on, is part of the same kriging model as *i*. For our current purposes, the term model is synonymous with molecule, i.e. an atom can exert an IQA force on another atom only if both atoms belong to the same molecule.

```
Do i=1,natms
  Predict $E^{i}_{IQA}$
  Do j=1,natms
    If (j $\in$ MODEL(i)) Then
      Compute the IQA force exerted on j by i
    End If
  End Do
End Do
```

Figure 5. Pseudo code for DL_FFLUX IQA energy and force calculation loop before modification.

The routine has been changed in several ways as demonstrated in Figure 6. Prior to any computation of energies and forces, the globally indexed model arrays that keep track of which atoms share a kriging model need to be converted to local indices. Note that Figures 5 and 6 show only the MODEL array for the sake of simplicity but, in reality, there are two arrays needed for this purpose. If this were not done, then local indices would be fed into these arrays and incorrect global indices would be returned. In other words, DL_FFLUX would not keep proper track of which atoms belong to the same model leading to incorrect predictions and resulting forces. Note that while only the first "natms" entries (the local part) of the MODEL array are needed for the force calculation in Figure 6, the rest of the array is required at a later point and so it is necessary to convert to local indices for the full array.

```
Do i=1,nlast
  Construct locallly indexed MODEL(i)
End Do

Do i=1,natms
  Predict $E^{i}_{IQA}$
  Do j=1,Size(MODEL(i))
    If (j $\in$ MODEL(i)) Then
      Compute the IQA force exerted on i by j
    End If
  End Do
End Do
```

Figure 6. Pseudo code for DL_FFLUX IQA energy and force calculation loop after modification.

For the DD method to work properly, a given process must compute all forces exerted on its local atoms by all local and halo atoms. The original code in Figure 5 did not do this as both loops were indexed from 1 to *natms* . Given the outer loop corresponded to the atom exerting the force, only the forces exerted on local atoms by local atoms would be computed. This would pose an issue for any molecules that do not reside entirely in a single domain. To resolve this issue, the inner loop now occurs explicitly over all model atoms which, thanks to the localisation of the model arrays, could be local atoms or part of the halo.

The other major change to the DL_FFLUX code concerns the prediction of the multipole moments. In the case of point charges this is done in a routine called fflux_ewald. This routine predicts atomic charges at every time step and then feeds them into DL_FFLUX and modified versions of DL_POLY's Ewald routines to compute the long-range electrostatics. In the serial case, moments are predicted in a simple loop, as shown in Figure 7.

```
Do i=1,natms
  Predict Charge on atom i
End Do
```

Figure 7. Original charge prediction loop.

When PBCs are also included there is a slight complication as the image atoms also need charges. However, these are all copies of main cell atoms, which means that the charges have already been predicted. As such, the image atoms in the PBC halo are just assigned the same charges as their main cell counterparts without requiring extra predictions.

In the case of MPI the modification to the loop is very simple and is shown in Figure 8. The only difference here is that the loop now runs to *nlast* rather than *natms* . This means that charges are predicted for all local and halo atoms. This is correct but involves some redundant work. The halo consists of main cell atoms that reside in a neighbouring domain, which do require new charge predictions. However, it also consists of image atoms, which may not require new charge predictions. The impact of this redundant work is discussed in the Results Section.

```
Do i=1,nlast
  Predict Charge on atom i
End Do
```

Figure 8. Modified charge prediction loop.

In the case of multipole moments of higher rank than point charge, the construction of the multipole component of the halo is done differently. Rather than re-predicting the halo moments as in the case of point charges, the halo moments are instead filled in by MPI communications. This is done in exactly the same way as the rest of the halo is constructed. The only difference is that it is done mid time step. Arguably this is a less elegant solution than constructing every aspect of the halo at the start of a time step. However, it does not appear to impact performance. In the future this could, with some considerable restructuring of code, be integrated into the DL_POLY halo construction. However, this would involve considerable alteration of DL_POLY routines thus compromising the modularity of DL_FFLUX.

4. Results

Four systems were considered, which consist of water boxes of different sizes: 5,184 atoms; 10,125 atoms; 46,875 atoms and 107,811 atoms. The dimensions of the boxes were chosen to give approximately correct liquid densities at 300 K. All simulations were carried out with in the NVT ensemble using the Nosé-Hoover thermostat with a 1 fs time step and a 9 Å cut-off radius unless stated otherwise. An MPI process always binds to a single processor core. These 4 test cases provide a reasonable range of system sizes. Unless stated otherwise, all benchmarks were performed on a single node comprising 2 Intel Xeon Gold 6152 "Skylake" chips, each with 22 cores and a nominal clock speed of 2.10 GHz. The node has 768 Gb of RAM.

The Supporting Information (SI) provides extra information such as Figures S1 to S4, which examine how the run time scales as a function of system size for a given value of the number of processes N_p. Figures S1 to S4 show the relationships for $N_p = 1$, 2, 4 and 8, respectively, with a range extending to more than 100,000 atoms. Secondly, Table S1 shows how the code scales with cut-off radius (for the 107,811 atom system).

4.1. Optimisation

Alongside parallelisation, the DL_FFLUX code has been extensively rewritten to be faster and more memory-efficient when running in serial. This section compares the serial un-optimised (original) and optimised codes, that is, without any MPI. The optimised code now uses up to a factor of 10^4 less memory for the systems studied (the larger the system, the larger the saving). The effects of optimisation on the runtime are best demonstrated by comparing timings and profiles of the serial code before and after optimisation. Time and memory constraints of the un-optimised code[22] mean that only the smallest system (5,184 atoms) could be studied prior to optimisation. More detail about the optimisation is given in Section 6 of the SI ("Technical details on optimisation").

The profiles in Figure 9 are completely different. The un-optimised profile is dominated entirely by DL_FFLUX, taking 99.8% of the runtime. After optimisation, DL_FFLUX takes a considerably more modest 32.2% of the runtime.

FFLUX
DL_POLY
FFLUX
DL_POLY

Figure 92. Profiles of the un-optimised (left) and optimised (right) code broken down into DL_FFLUX and DL_POLY contributions. The charts are not to scale, the right hand chart represents a computation that is ~75,000x faster than the left hand chart.

The timings in Table 1 also demonstrate large speed-ups when running in serial. Results are reported in time taken for a 1 ns simulation with 1 fs time steps, and a 9 Å electrostatic and van der Waals cut-off radius. Note that the original DL_FFLUX code uses a different definition of the rank of electrostatic interaction. Rather than L’, the original code uses $L = l_A + l_B + 1$ where l_A and l_B are the rank of the multipoles on atoms *A* and *B* , respectively. In other words, in the (l_A, l_B) matrix of possible interacting multipole moments, *L* refers to a triangle while *L’* refers to a square. This means that, for L=2 for example, dipole moments are enabled but only the monopole-monopole and dipole-monopole (or dipole-monopole) interactions are computed. Note that dipole-dipole interactions are not computed because then *L* would be 1+1+1=3. However, for L’ we can say that the closest equivalent L’ value to L=2 is L’=1, which computes also the dipole-dipole interaction. As such, when comparing the old and new DL_FFLUX it is important to recognise that the new code is calculating many more electrostatic interactions as L (or L’) becomes larger. Figure S5 in the SI examines the computational cost of increasing L’ in more detail.

	Un-optimised	**Optimised**	**Speed-up**
L=1/L’=0	250 years	29.0 hours	x 75,569
L=2/L’=1	1,709 years	4.7 days	x 132,811
L=3/L’=2	6,573 years	6.5 days	x 369,352

Table 1. Comparison of time taken to perform a 1 ns simulation of the 5,184 atom water box using the original (un-optimised) and optimised codes. Note that the un-optimised timings are estimates based on single time step simulations.

4.2. Strong scaling

Strong scaling is the scaling of runtime with the number of processes (note that Section 1 in the SI shows weak scaling, i.e. scaling of runtime with problem size at fixed N_p). All four systems were run at each of the 3 levels of electrostatic interaction (L'=0, 1 and 2) at various values of N_p in order to study the strong scaling.

The strong scaling data for all 4 systems are shown in Figure 10. It is evident that all systems behave in a broadly similar way although the overall quality of the scaling tends to be better as system size increases. Larger systems are also able to utilise more processes (as the cell can be divided into more domains) and so often achieve a greater overall speed-up. From now on data will not necessarily be shown for all systems to avoid repetition. Any data not given in the main paper can be found in the SI. We will now focus on the largest system (107,811 atoms) in order to analyse the scaling in more depth. Any discussion generalises to all 4 systems unless stated otherwise.

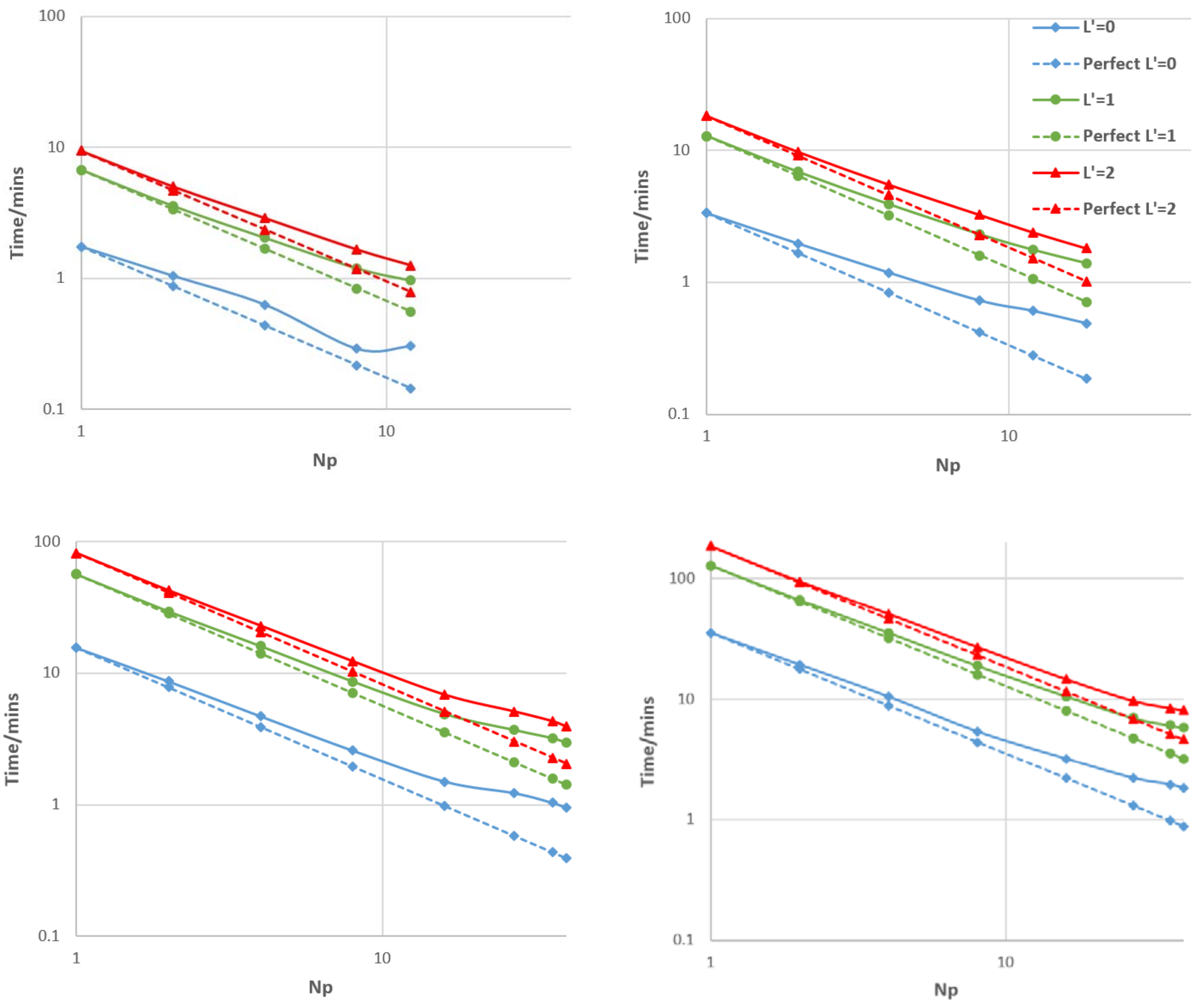

Figure 30. Strong scaling, both actual and perfect, on log-log axes for all 4 water boxes: (top left) 5,184 atoms in a (38 Å)^3 box; (top right) 10,125 atoms in a (47 Å)^3 box; (bottom left) 46,875 atoms in a (78 Å)^3 box; and (bottom right) 107,811 atoms in a (103.5 Å)^3 box.

Figure 11 provides a different measure of the quality of the scaling, plotting t_1/t_{N_p} (the time when run on a single process compared to that on N_p processes) rather than time itself. This ratio is the factor by which the code has been sped-up by going from serial to N_p processes. In the perfect case this will simply be equal to N_p .

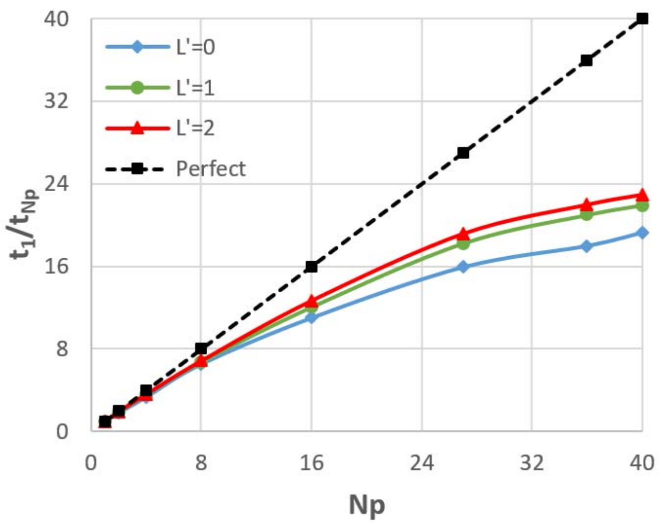

Figure 41. Speed-up of the whole code relative to serial for the 107,811 atom water box.

It is clear from Figure 11 that L'=0 scales worse than L'=1 or L'=2. This is generally the case for all 4 systems as can be seen in Figure 10 and Figures S6 to S8, which report on the remaining 3 systems. Profiling revealed the culprit is the aforementioned redundant prediction work being done in the case of L'=0, which is avoided when implementing MPI communications for higher rank multipole moments. In the case of L'=0 the predictions scale poorly with N_p whereas, for L'=2, they scale considerably better.

The poor scaling of predictions for L'=0 shown in Figure 12 is exacerbated by the fact that the predictions take on average 26 % of the runtime, while for L'=2 the predictions take on average 9 % of the runtime. Note L'=1 is not considered here as initial profiles showed performance essentially identical to L'=2. There is more actual work to be done by the prediction routine for L'=2 because there are considerably more multipole moments to predict per atom. However, the removal of the redundant work and the massively increased cost of the electrostatics routines means that predictions take up a relatively small fraction of the overall work. This analysis suggests that it would be beneficial to remove the redundant work associated with the point charge halo in the same way as is done for higher rank multipole moments. This will be the subject of future development of DL_FFLUX.

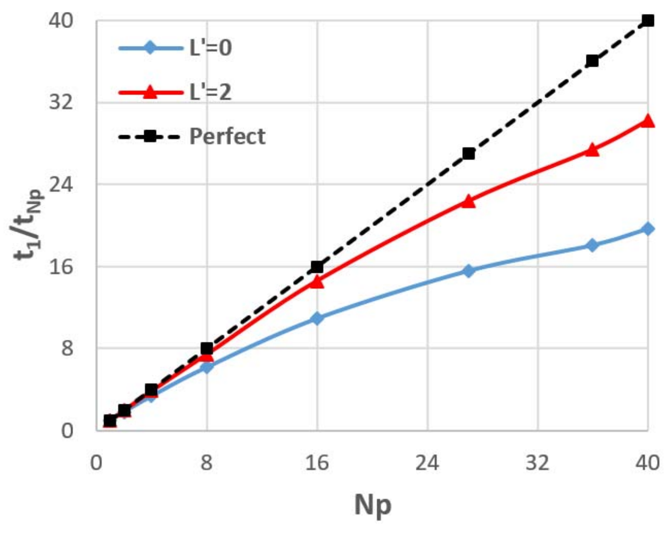

Figure 52. Speed-up of the DL_FFLUX prediction routine only relative to serial for the 107,811 atom water box.

Until now, all tests have been confined to a single node. However, the performance of MPI across multiple nodes is a key metric. As such, the largest system (107,811 atoms) was studied on a set of nodes, each with two 12-core Intel Xeon E5-2690 v3 "Haswell" chips with a nominal clock speed of 2.6 GHz and 128 Gb RAM per node as well as Mellanox InfiniBand. All timings in Figure 13 (apart from serial) have been obtained on at least 2 nodes, e.g. N_p = 2 means 1 process per node where each process still corresponds to a single core. The scaling in the right hand graph of Figure 13 is comparable to that in Figure 11 up to N_p = 40 (the maximum in Figure 11) despite the processes being split over multiple nodes in Figure 13. This shows that the internode communications in DL_FFLUX are not overly costly and do not prohibit scaling across multiple nodes, at least for small numbers of nodes. For L'=2, the best speed-up increases from a factor of 23x (Figure 11) to a factor of 44x using 4 nodes.

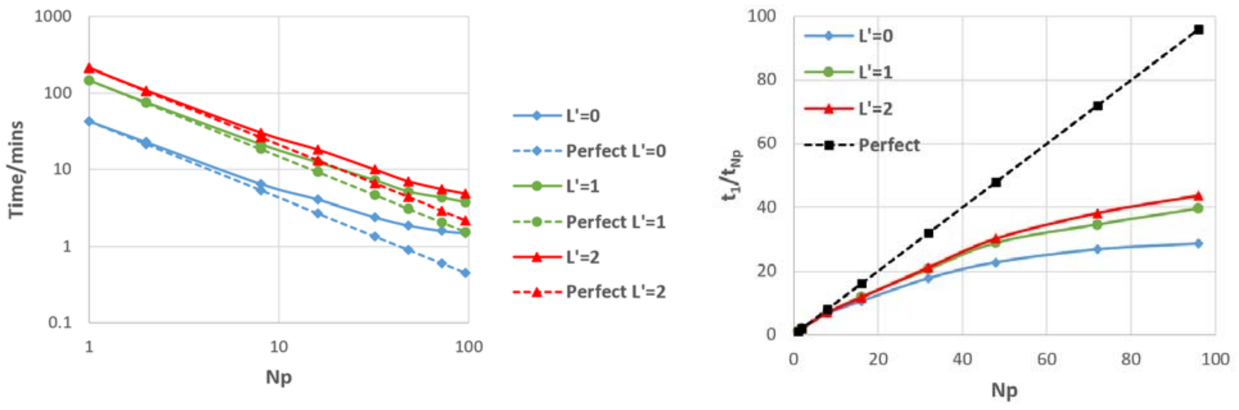

Figure 13. Strong scaling on multiple nodes for the 107,811 atom system. Log-log scaling (left) and speed-up relative to serial (right).

4.3. The computational cost of DL_FFLUX

Perhaps the most relevant question pertaining to DL_FFLUX's performance is how much extra computational cost the methodology incurs over more traditional MD techniques. DL_FFLUX is based on DL_POLY, so a comparison between the two codes is a natural one to draw. There are multiple ways to explore this question. Firstly, a 'direct' comparison can be drawn between "plain vanilla" DL_POLY with a commonly used water potential and DL_FFLUX. To this end, we performed simulations using DL_POLY 4.09 and a flexible SPC water potential[33] for the 107,811 atom water box at L'=0, 1 and 2. For the L'=1 and L'=2 comparisons, the SPC potential was modified to include fixed dipole and quadrupole moments (based on average DL_FFLUX values). The SPC and DL_FFLUX models have in common that they are flexible. However, the flexibility in DL_FFLUX arises from N-body intramolecular interactions[52] (comprised of intra- and interatomic interactions). DL_FFLUX also has fully polarisable multipole moments. We mention here that kriging takes care of predicting the outcome of the polarisation process (i.e. a multipole moment corresponding to a given geometry) rather than the polarisation process itself (i.e. polarisability). We note that there are more recent, improved water potentials other than SPC that include polarisation in various ways[53,54].

Figure 14 shows the ratio of runtime in DL_FFLUX compared to DL_POLY, the latter using the modified SPC potential. The ratio is taken at all values of N_p (N_p = 1, 2, 4, …) and then averaged to produce the values in Figure 14. For example, a value of 2 means that DL_FFLUX is on average 2x slower than DL_POLY. Figure 14 shows that, despite a slowdown at L'=0, DL_FFLUX is in fact quicker on average than plain DL_POLY, once higher rank multipole moments are enabled. The speed-up with higher rank multipole moments is largely due to some optimisation of the Ewald routines in DL_FFLUX. Figure 14 is compelling evidence that DL_FFLUX can (and has been) integrated into a traditional MD package while incurring minimal extra computational overhead. This means that DL_FFLUX is now competitive in terms of speed whilst offering a methodology with a rigorous theoretical grounding.

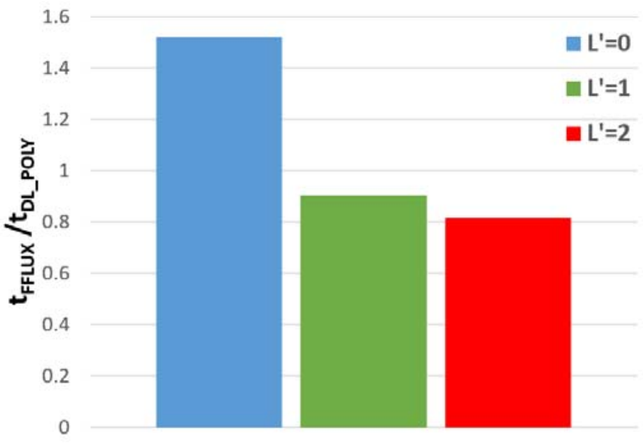

Figure 14. The time for a DL_FFLUX simulation divided by the time for a DL_POLY simulation with a flexible SPC water potential using the 107,811 atom water box. Averaged over all values of N_p tested.

The performance of DL_FFLUX relative to DL_POLY can be analysed further by breaking the code into the DL_FFLUX and DL_POLY components. This is done by profiling the code and comparing the performance of the DL_FFLUX subroutines to the DL_POLY subroutines. This analysis allows examination of the strong scaling of the MPI in both parts of the code separately. This is shown for L'=0 and L'=2 for the 107,811 atom example. Figure 15 shows that in both cases, DL_FFLUX and DL_POLY scale in a similar way, i.e. the DD MPI implementations in both sets of routines are well integrated with each other. The reason that DL_FFLUX scales worse than DL_POLY in the L'=0 case is once again due to the redundant work when predicting point charges. Note that the scalability of the MPI in various versions of DL_POLY has been tested extensively in the past[55,56].

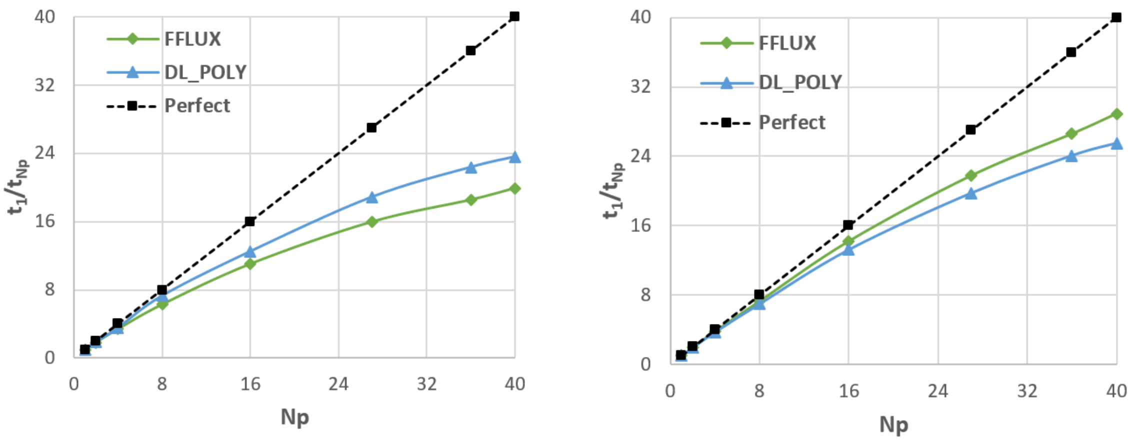

Figure 65. Speed-up, relative to serial, of DL_FFLUX and DL_POLY routines at L'=0 (left) and L'=2 (right).

Profiling also gives us some insight into the percentage of the total runtime spent in DL_FFLUX and DL_POLY routines (as well as MPI specific routines), which is presented in Figure 16 as pie charts. For the sake of clarity, just the profiles for N_p = 1, 8 and 36 are shown; the rest of the pie charts appear in Figure S9 as well as a more detailed "sample" breakdown in Figure S10 showing the relative timings of the top 5 most costly subroutines. Figure 16 gives a little more context to the numbers presented in Figure 14. The fraction of time spent in DL_FFLUX becomes much less important as L' increases and the electrostatics starts to dominate, consistent with the trend in Figure 14. In all cases, DL_FFLUX represents a minority of the overall runtime.

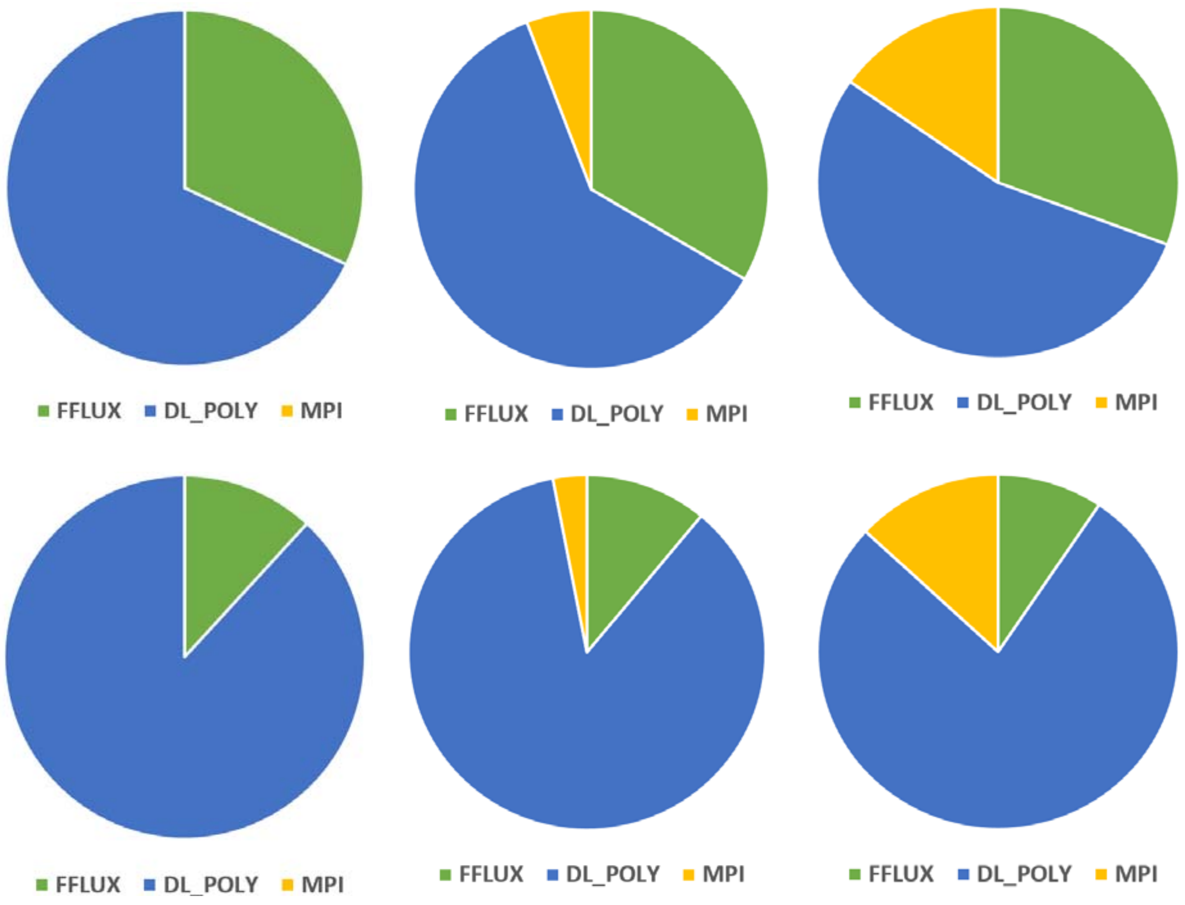

Figure 16. Profiles of the code at L'=0 (top row) and L'=2 (bottom row). Breakdowns are shown for N_p = 1, 8 and 36 going from left to right. The charts are not to scale, as N_p increases the total time taken decreases.

Figure 16 also provides some insight into the deterioration of the strong scaling seen in Figure 10 as N_p increases. As expected, the fraction of time spent in MPI routines increases as N_p increases. This is because, as N_p increases for a fixed system size, the domain size decreases, which means that the ratio in eq 9 increases. This in turn means that each process has to communicate information about a greater fraction of the atoms in its domain. There is also the issue of the number of link cells per domain. DL_POLY has a built-in warning system that tells the user when the number of link cells per domain is smaller than 3, i.e. when the link cell method is not efficient. For the 107,811 atom system this warning is given only when N_p = 16. Note that the warning is given for 16 and but not for 27 because, in the latter case, the space can be broken into 3 domains per dimension (3x3x3) whereas for N_p = 16, the decomposition is 2, 2, 4, i.e. there is an asymmetry in the dimensions leading to fewer link cells per domain in some. For the smaller systems this warning is given considerably more frequently. In other words there is a non-linear relationship between the cost of communications, t_{comms} (the final term in eq 8) and N_p. This is demonstrated more explicitly in Figure S11.

4.4. Best case timings

Presented in Figure 17 are the best case timings for each of the four systems in nanoseconds per day (ns/day). This is an informative, practical metric as the behaviour of water is well reproduced and studied at the scale of nanoseconds. More complex systems may require simulations on the order of tens or even hundreds of nanoseconds. Between a day and a week is a realistic time frame for a simulation. It is therefore clear from Figure 17 that DL_FFLUX is now capable of studying the behaviour of systems up to approximately 10^5 atoms in a reasonable time frame.

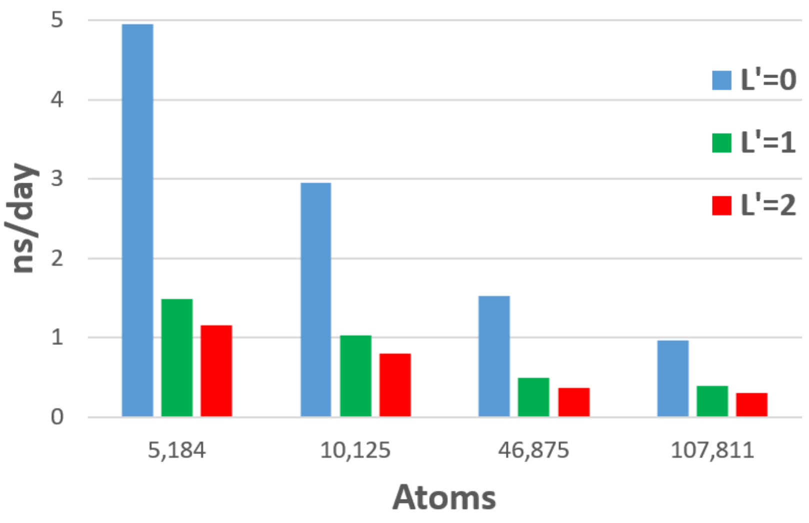

Figure 17. Timings at the optimum number of MPI processes for each system in nanoseconds per day.

5. Conclusion

The advance presented here is that of feasibility. The rigorous methodology of FFLUX based on quantum chemical topology has already been shown to be accurate for small systems. However, the practicality of applying the method to large scale molecular dynamics has been hampered by computational cost. The newly optimised and parallelised DL_FFLUX suffers from no such issues. DL_FFLUX can now comfortably and competitively operate on systems comprised of up to 10^5 atoms. With flexible molecules and high-rank long-range electrostatics based on polarisable atomic multipole moments, DL_FFLUX is now poised to tackle problems of real world significance accurately and in reasonable time frames. We have also demonstrated that DL_FFLUX is a light-weight add-on to the standard DL_POLY package, incurring little computational overhead. DL_FFLUX is also well integrated into the existing DL_POLY MPI, preserving the quality of the scaling with respect to number of processes.

Supporting Information

The Supporting Information is free of charge on the ACS publications website.

Section 1 on weak scaling contains Figures S1-S4 showing runtime as a function of the number of atoms for varying L' and N_p values. Section 2 on the cut-off radius gives timings in Table S1. Section 3 on L' shows Figure S5 with timings as a function of L' and the number of atoms. Section 4 on scaling contains Figures S6, S7 and S8, with relative speed-ups as a function of L' and an increasing number of atoms. Section 5 on profiling shows Figures S9 and S10 with pie charts, and Figure S11 with the fraction of total runtime spent in MPI communications as a function of N_p. Section 6 provides technical details on optimisation.

AUTHOR INFORMATION

Corresponding Author

*Phone: +44 161 3064511. E-mail: pla@manchester.ac.uk

Funding Sources

P.L.A.P. acknowledges the EPSRC for funding through the award of an Established Career Fellowship (grant EP/K005472).

Acknowledgments

The authors would like to acknowledge the assistance given by Research IT and the use of the Computational Shared Facility at The University of Manchester.

References

(1) Rauscher, S.; Gapsys, V.; Gajda, M. J.; Zweckstetter, M.; de Groot, B. L.; Grubmüller, H., Structural Ensembles of Intrinsically Disordered Proteins Depend Strongly on Force Field: A Comparison to Experiment. *J.Chem.Theor.Comput.* **2015**, *11*, 5513.

(2) Lindorff-Larsen, K.; Maragakis, P.; Piana, S.; Eastwood, M. P.; Dror, R. O.; Shaw, D. E., Systematic Validation of Protein Force Fields against Experimental Data. *PLOS ONE* **2012**, *7*, e32131.

(3) Smith, J. S.; Isayev, O.; Roitberg, A. E., ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chem.Sci.* **2017**, *8*, 3192.

(4) Smith, J. S.; Nebgen, B. T.; Zubatyuk, R.; Lubbers, N.; Devereu, C.; Barros, K.; Tretiak, S.; Isayev, O.; Roitberg, A. E., Approaching coupled cluster accuracy with a general-purpose neural network potential through transfer learning. *Nat. Commun.* **2019**, *10*, 2903.

(5) Ko, T. W.; Finkler, J. A.; Goedecker, S.; Behler, J., General-Purpose Machine Learning Potentials Capturing Nonlocal Charge Transfer. *Accs.Chem.Res.* **2021**, *54*, 808.

(6) Wu, X.; Clavaguera, C.; Lagardère, L.; Piquemal, J.-P.; de la Lande, A., AMOEBA Polarizable Force Field Parameters of the Heme Cofactor in Its Ferrous and Ferric Forms. *J.Chem.Theor.Comput.* **2018**, *14*, 2705.

(7) Rackers, J. A.; Wang, Z.; Lu, C.; Laury, M. L.; Lagardère, L.; Schnieders, M. J.; Piquemal, J.-P.; Ren, P.; Ponder, J. W., Tinker 8: Software Tools for Molecular Design. *J.Chem. Theory Comp* **2018**, *14*, 5273.

(8) Jolly, L.-H.; Duran, A.; Lagardère, L.; Ponder, J. W.; Ren, P.; Piquemal, J.-P., Raising the Performance of the Tinker-HP Molecular Modeling Package [Article v1.0]. *Living J. Comp. Mol. Sci* **2019**, *1*, 10409.

(9) Nguyen, T. T.; Szekely, E.; Imbalzano, G.; Behler, J.; Csanyi, G.; Ceriotti, M.; Gotz, A. W.; Paesani, F., Comparison of permutationally invariant polynomials, neural networks, and Gaussian approximation potentials in representing water interactions through many-body expansions. *J.Chem.Phys.* **2018**, *148*, 241725.

(10) Deringer, V. L.; Bernstein, N.; Csányi, G.; Ben Mahmoud, C.; Ceriotti, M.; Wilson, M.; Drabold, D. A.; Elliott, S. R., Origins of structural and electronic transitions in disordered silicon. *Nature* **2021**, *589*, 59.

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

(12) Huang, B.; von Lilienfeld, O. A., Quantum machine learning using atom-in-molecule-based fragments selected on the fly. *Nat.Chem.* **2020**, *12*, 945.

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

(14) Cole, D. J.; Vilseck, J. Z.; Tirado-Rives, J.; Payne, M. C.; Jorgensen, W. L., Biomolecular Force Field Parameterization via Atoms-in-Molecule Electron Density Partitioning. *J.Chem.Theor.Comput.* **2016**, *12*, 2312.

(15) Li, W.; Dong, H.; Ma, J.; Li, S., Structures and Spectroscopic Properties of Large Molecules and Condensed-Phase Systems Predicted by Generalized Energy-Based Fragmentation Approach. *Accts Chem Res* **2021**, *54*, 169.

(16) Schriber, J. B.; Nascimento, D. R.; Koutsoukas, A.; Spronk, S. A.; Cheney, D. L.; Sherrill, C. D., CLIFF: A component-based, machine-learned, intermolecular force field. *J.Chem.Phys.* **2021**, *154*, 184110.

(17) Leven, I.; Hao, H.; Tan, S.; Guan, X.; Penrod, K. A.; Akbarian, D.; Evangelisti, B.; Hossain, M. J.; Islam, M. M.; Koski, J. P.; Moore, S.; Aktulga, H. M.; van Duin, A. C. T.; Head-Gordon, T., Recent Advances for Improving the Accuracy, Transferability, and Efficiency of Reactive Force Fields. *J.Chem.Theory Comput.* **2021**, *17*, 3237.

(18) Hughes, Z. E.; Thacker, J. C. R.; Wilson, A. L.; Popelier, P. L. A., Description of Potential Energy Surfaces of Molecules Using FFLUX Machine Learning Models. *J.Chem.Theor.Comp.* **2019**, *15*, 116.

(19) Thacker, J. C. R.; Wilson, A. L.; Hughes, Z. E.; Burn, M. J.; Maxwell, P. I.; Popelier, P. L. A., Towards the simulation of biomolecules: optimisation of peptide-capped glycine using FFLUX. *Mol.Simul.* **2018**, *44*, 881.

(20) Burn, M. J.; Popelier, P. L. A., Creating Gaussian Process Regression Models for Molecular Simulations Using Adaptive Sampling *J.Chem.Phys.* **2020**, *153*, 054111.

(21) Fletcher, T. L.; Popelier, P. L. A., Multipolar Electrostatic Energy Prediction for all 20 Natural Amino Acids Using Kriging Machine Learning. *J.Chem.Theor.Comput.* **2016**, *12*, 2742.

(22) Hughes, Z. E.; Ren, E.; Thacker, J. C. R.; Symons, B. C. B.; Silva, A. F.; Popelier, P. L. A., A FFLUX Water Model: Flexible, Polarizable and with a Multipolar Description of Electrostatics. *J.Comput.Chem.* **2019**, *41*, 619.

(23) Di Pasquale, N.; Davie, S. J.; Popelier, P. L. A., The accuracy of ab initio calculations without ab initio calculations for charged systems: Kriging predictions of atomistic properties for ions in aqueous solutions. *J.Chem.Phys.* **2018**, *148*, 241724.

(24) Popelier, P. L. A., Molecular Simulation by Knowledgeable Quantum Atoms. *Phys.Scr.* **2016**, *91*, 033007.

(25) Bader, R. F. W. *Atoms in Molecules. A Quantum Theory.*; Oxford Univ. Press: Oxford, Great Britain, 1990.

(26) Blanco, M. A.; Martín Pendás, A.; Francisco, E., Interacting Quantum Atoms: A Correlated Energy Decomposition Scheme Based on the Quantum Theory of Atoms in Molecules. *J.Chem.Theor. Comp.* **2005**, *1*, 1096.

(27) Popelier, P. L. A.; Aicken, F. M., Atomic Properties of Amino Acids: computed Atom Types as a Guide for future Force Field Design. *ChemPhysChem* **2003**, *4*, 824.

(28) Popelier, P. L. A. In *The Nature of the Chemical Bond Revisited*; Frenking, G., Shaik, S., Eds.; Wiley-VCH, Chapter 8: 2014, p 271.

(29) Jones, D. R.; Schonlau, M.; Welch, W. J., Efficient Global Optimization of Expensive Black-Box Functions. *J.Global Optim.* **1998**, *13*, 455.

(30) Rasmussen, C. E.; Williams, C. K. I. *Gaussian Processes for Machine Learning.*; The MIT Press: Cambridge, USA, 2006.

(31) Handley, C. M.; Hawe, G. I.; Kell, D. B.; Popelier, P. L. A., Optimal Construction of a Fast and Accurate Polarisable Water Potential based on Multipole Moments trained by Machine Learning. *Phys.Chem.Chem.Phys.* **2009**, *11*, 6365.

(32) Gabriel, E.; Fagg, G. E.; Bosilica, G.; Angskun, T.; Dongarra, J. J.; Squyres, J. M.; Sahay, V.; Kambadur, P.; Barrett, B.; Lumsdaine, A.; Castain, R. H.; Daniel, D. J.; Graham, R. L.; Woodall, T. S. In *Proceedings, 11th European PVM/MPI Users' Group Meeting* Budapest, Hungary, 2004.

(33) Amira, S.; Spangberg, D.; Hermansson, K., Derivation and evaluation of a flexible SPC model for liquid water *Chem.Phys.* **2004**, *303*, 327.

(34) Lemkul, J. A.; Huang, J.; Roux, B.; MacKerell, A. D. J., An Empirical Polarizable Force Field Based on the Classical Drude Oscillator Model: Development History and Recent Applications. *Chem.Rev.* **2016**, *116*, 4983−5013.

(35) Rick, S. W.; Stuart, S. J.; Berne, B. J., Dynamical Fluctuating Charge Force Fields: Application to Liquid Water. *J Chem.Phys.* **1994**, *101*, 6141.

(36) Poier, P. P.; Jensen, F., Describing Molecular Polarizability by a Bond Capacity Model *J. Chem. Theory Comput.* **2019**, *15*, 3093.

(37) Guevara-Vela, J. M.; Francisco, E.; Rocha-Rinza, T.; Martín Pendás, A., Interacting Quantum Atoms - A Review. *Molecules* **2020**, *25*, 4028.

(38) Phipps, M. J. S.; Fox, T.; Tautermann, C. S.; Skylaris, C.-K., Energy decomposition analysis approaches and their evaluation on prototypical protein–drug interaction patterns. *Chem.Soc.Rev.* **2015**, *44*, 3177.

(39) Glendening, E. F.; Streitwieser, A., Natural energy decomposition analysis: An energy partitioning procedure for molecular Interactions with application to weak hydrogen bonding, strong ionic, and moderate donor-acceptor interactions. *J.Chem.Phys.* **1994**, *100*, 2900.

(40) Popelier, P. L. A.; Kosov, D. S., Atom-atom partitioning of intramolecular and intermolecular Coulomb energy. *J.Chem.Phys.* **2001**, *114*, 6539.

(41) Martin Pendas, A.; Blanco, M. A.; Francisco, E., Chemical Fragments in Real Space: Definitions, Properties, and Energetic Decompositions. *J.Comp.Chem.* **2007**, *28*, 161.

(42) Mills, M. J. L.; Popelier, P. L. A., Electrostatic Forces: formulae for the first derivatives of a polarisable, anisotropic electrostatic potential energy function based on machine learning. *J.Chem.Theory Comput.* **2014**, *10*, 3840−3856.

(43) Essmann, U.; Perera, L.; Berkowitz, M. L.; Darden, T.; Lee, H.; Pedersen, L. G., A smooth particle mesh Ewald method. *J.Chem.Phys.* **1995**, *103*, 8577.

(44) Rafat, M.; Popelier, P. L. A., A convergent multipole expansion for 1,3 and 1,4 Coulomb interactions. *J.Chem.Phys.* **2006**, *124*, 144102.

(45) Yuan, Y.; Mills, M. J. L.; Popelier, P. L. A., Multipolar Electrostatics for Proteins: Atom-Atom Electrostatic Energies in Crambin. *J.Comput.Chem.* **2014**, *35*, 343.

(46) Smith, W., Molecular Dynamics on hypercube parallel computers. *Comp.Phys.Commun.* **1991**, *62*, 229.

(47) Todorov, I. T.; Smith, W.; Trachenko, K.; Dove, M. T., DL_POLY_3: new dimensions in molecular dynamics simulations via massive parallelism. *J.Mater.Chem.* **2006**, *16*, 1911.

(48) Boateng, H. A.; Todorov, I. T., Arbitrary order permanent Cartesian multipolar electrostatic interactions. *J.Chem.Phys.* **2015**, *142*, 034117.

(49) Todorov, I. T.; Smith, W.; CCLRC Daresbury Laboratory: Warrington, Great Britain, 2018.

(50) Todorov, I. T.; Smith, W., DL POLY 3: the CCP5 national UK code for molecular-dynamics simulations. *Phil.Trans.R.Soc.Lond. A* **2004**, *362*, 1835.

(51) Pinches, M. R. S.; Tildesley, D.; Smith, W., Large Scale Molecular Dynamics on Parallel Computers using the Link-cell Algorithm *Molec.Simul.* **1991**, *6*, 51.

(52) Konovalov, A.; Symons, B. C. B.; Popelier, P. L. A., On the many-body nature of intramolecular forces in FFLUX and its implications. *J.Comp.Chem.* **2021**, *42*, 107.

(53) Naserifar, S.; Goddard, W. A., The quantum mechanics-based polarizable force field for water simulations. *J.Chem.Phys.* **2018**, *149*, 174502.

(54) Liu, C.; Piquemal, J.-P.; Ren, P., Implementation of Geometry-Dependent Charge Flux into the Polarizable AMOEBA+ Potential. *J.PhysChem.Lett.* **2020**, *11*, 419.

(55) Guest, M. F.; Elena, A. M.; Chalk, A. B. G., DL_POLY - a performance overview analysing, understanding and exploiting available HPC technology. *Molec.Simul.* **2019**, 1603380.

(56) Mabakane, M. S.; Moeketsi, D. M.; Lopis, A. S., Scalability of DL_POLY on High Performance Computing Platform. *South African Computer Journal* **2017**, *29*, 81.