Coarse-Grained Simulation of DNA using LAMMPS
An implementation of the oxDNA model

Oliver Henrich*
*Department of Physics, SUPA, University of Strathclyde, Glasgow G4 0NG, Scotland, UK**
(Dated: September 6, 2021)

During the last decade coarse-grained nucleotide models have emerged that allow us to study DNA and RNA on unprecedented time and length scales. Among them is oxDNA, a coarse-grained, sequence-specific model that captures the hybridisation transition of DNA and many structural properties of single- and double-stranded DNA. oxDNA was previously only available as standalone software, but has now been implemented into the popular LAMMPS molecular dynamics code. This article describes the new implementation and analyses its parallel performance. The LAMMPS implementation of oxDNA lowers the entry barrier for using the oxDNA model significantly, facilitates future code development and interfacing with existing LAMMPS functionality as well as other coarse-grained and atomistic DNA models.

I. INTRODUCTION

DNA is one of the most important bio-polymers, as its sequence encodes the genetic instructions needed in the development and functioning of many living organisms. While we know now the sequence of many genomes, we still know little as to how DNA is organised in 3D inside a living cell, and of how gene regulation and DNA function are coupled to this structure. The complexity of the DNA molecule can be brought to mind by highlighting a few of its quantitative aspects. The entire DNA within a single human cell is about 2 m long, but only 2 nm wide and organised at different hierarchical levels. If compressed into a spherical ball, this ball would have a diameter of about 2 μm [1].

Computational modelling of DNA appears as the only avenue to understanding its intricacies in sufficient detail and has been an important field in biophysics for decades. Traditionally, most of the available simulation techniques have worked at the atomistic level of detail [2]. Existing atomistic force fields can capture fast conformational fluctuations and protein-DNA binding, but cannot deliver the necessary temporal and spatial resolution to describe phenomena that occur on larger time and length scales as they are often limited to a few hundred base pairs and (at most) microsecond time scales. Recent years have therefore witnessed a rapid increase of a new research effort at a different, coarse-grained level [3]. Coarse-grained (CG) models of DNA can provide significant computational and conceptual advantages over atomistic models leading often to three or more orders of magnitude greater efficiency. The challenge consists in retaining the right degrees of freedom so that the CG model reproduces relevant emergent structural features and thermodynamic properties of DNA. CG modelling of DNA is not only an efficient alternative to atomistic approaches. It is indispensable for the modelling of DNA on timescales in the millisecond range and beyond, or when long DNA strands of tens of thousands of base pairs or more have to be considered, e.g. to study the dynamics of DNA supercoiling (i.e. the local over- or under-twisting of the double helix, which is also important for gene expression in bacteria), of genomic DNA loops and of chromatin or chromosome fragments.

A small number of very promising CG DNA models have emerged to date. Conceptually they can be categorised into top-down approaches, which use empirical interactions that are parameterised to match experimental observables, or bottom-up approaches, which eliminate dispensable degrees of freedom systematically starting from atomistic force fields. They may also target different applications depending on their capabilities, such as single versus double stranded DNA (ssDNA and dsDNA), or nanotechnological versus biological applications. We refer to [4] for a comprehensive overview of the capabilities of individual models and recent activities in this field.

From a software point of view these models are often based on standalone software [5–7], which has a somewhat limiting effect on uptake and user communities growth. Others models use popular MD-codes as computational platforms, such as GROMACS [8] in case of the SIRAH [9] and the MARTINI force field [10], or NAMD [11, 12]. Another suitable platform for CG simulation of DNA has emerged in form of the powerful Large-Scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) for molecular dynamics [13], including the widely used 3SPN.2 model [14, 15] and others that target even larger length scales [16, 17].

This article reports the latest effort of implementing the popular oxDNA model [18, 19] into the LAMMPS code. Until recently this model was only available as bespoke and standalone software [20]. Through the efficient parallelisation of LAMMPS it is now possible to run oxDNA in parallel on multi-core CPU-architectures, extending its capabilities to unprecedented time and length scales. The largest system that could be studied by oxDNA was previously limited by the size of system that can be fitted onto a single GPU.

This paper is organised as follows: In Section II we briefly introduce the details of the oxDNA

* Corresponding email address: oliver.henrich@strath.ac.uk

and oxDNA2 models. Section III explains how the LAMMPS implementation of the oxDNA models can be invoked and provides further information on the code distribution and documentation. In Section IV we describe the LAMMPS implementation of novel Langevin-type rigid body integrators which feature improved stability and accuracy. Section V gives details of the scaling performance of parallel implementation.

II. THE OXDNA MODEL

The oxDNA model consists of rigid nucleotides with three interaction sites for the effective interactions between the nucleotides. These pairwise-additive forces arise due to the excluded volume, the connectivity of the phosphate backbone, the stacking, cross-stacking and coaxial stacking as a consequence of the hydrophobicity of the bases, as well as hydrogen bonding between complementary base pairs. Fig. 1 illustrates these interactions schematically for the original version of the model, to which we refer as oxDNA [19]. In this version all three interaction sites are co-

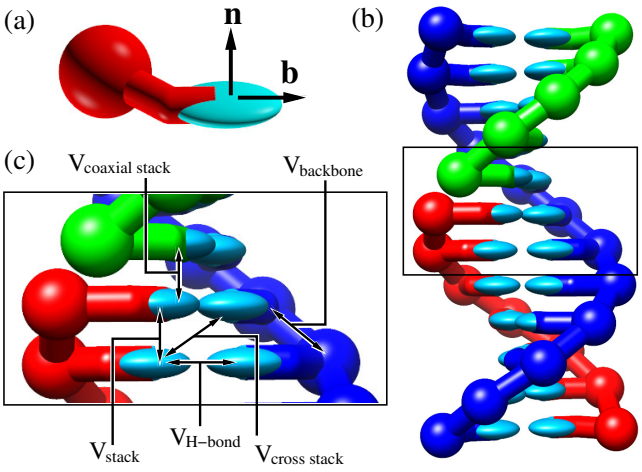

FIG. 1. Overview of oxDNA2 model: (a) Nucleotide geometry with base vector $\boldsymbol{b}$ and base normal $\boldsymbol{n}$. The backbone interaction site is situated approximately at the centre of the red sphere, whereas stacking and hydrogen-bonding interaction sites are on the cyan ellipsoid. (b) A short duplex featuring major and minor grooves. (c) Bonded interaction for phosphate backbone connectivity, pair interactions for hydrogen-bonding, stacking, cross-stacking and coaxial stacking. oxDNA2 contains additional interactions for excluded volume and implicit electrostatics in form of a Debye-Hückel potential.

linear. The hydrogen bonding/excluded volume site and the stacking site are separated from the backbone/electrostatic interaction site by 0.74 length units (6.3 Å) and 0.8 length units (6.8 Å), respectively. The orientation of the bases is specified by a base normal vector, which defines the notional plane of the base and the vector between the interaction sites. Together with the relative distance vectors between the interaction sites, the base vector and base normal vector are used to modulate the stacking, cross-stacking, coaxial stacking and hydrogen bonding interaction between two consecutive nucleotides.

The simplest interaction is the backbone connectivity, which is modelled with FENE (finitely extensible nonlinear elastic) springs acting between the backbone interaction sites. The excluded volume interaction is modelled with truncated and smoothed Lennard-Jones potentials between backbone sites, base sites and between the backbone and base sites. The hydrogen bonding interaction consists of smoothed, truncated and modulated Morse potentials between the hydrogen bonding site. The stacking interaction falls into three individual sub-interactions: the stacking interaction between consecutive nucleotides on the same strand as well as cross-stacking and coaxial stacking between any nucleotide in the appropriate relative position. It is worth emphasising that the duplex structure is not specified or imposed in any other way, but emerges naturally through this choice of interactions and their parameterisation. This is another strength of the oxDNA model and permits an accurate description of both ssDNA and dsDNA. The stacking interactions are modelled with a combination of smoothed, truncated and modulated Morse, harmonic angle and harmonic distance potentials. All interactions have been parameterised to match key thermodynamic properties of ssDNA and dsDNA such as the longitudinal and torsional persistence length or the melting temperature of the duplex [18, 21, 22].

A short schematic overview of various interactions involved in the definition of oxDNA2 model is given in Fig. 1. More details can be found in the original publications [18, 19] and in a recent publication on the use and concept of oxDNA [23].

The original model (oxDNA) has been further developed to include sequence-specific stacking and hydrogen bonding interaction strengths [25] (oxDNA1.5) and implicit ions, which are modelled by means of a Debye-Hückel potential [24] (oxDNA2). A major improvement of the latest version is also the fact that it shows the correct structure with major and minor grooves (see Fig. 2 (b)). This is achieved through a modification of the relative position of the backbone and stacking/hydrogen bonding interaction sites, as schematically depicted in Fig. 2 (a).

III. THE LAMMPS IMPLEMENTATION OF OXDNA

A. Code Distribution, Force Fields and Compilation

The software is open source and distributed under GNU General Public License (GPL). It is available for download as LAMMPS CG-DNA package from the central LAMMPS repository at Sandia National Laboratories, USA [13]. This includes a detailed online documentation, examples and utility scripts. We re-

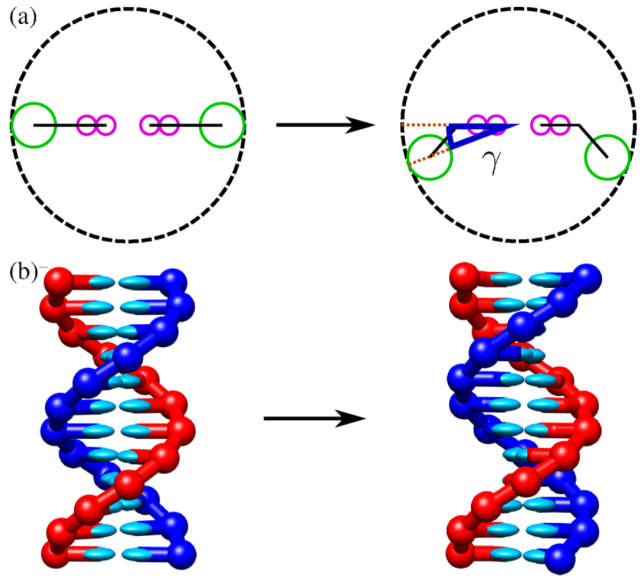

FIG. 2. (a): Schematic distinction between oxDNA (left) and oxDNA2 (right). In oxDNA all interaction sites are co-linear whereas in oxDNA2 the backbone interaction site and the stacking and hydrogen-bonding interaction sites are oriented at an angle. (b): The non-co-linear arrangement of the interaction sites leads to the formation of the major and minor groove, an important structural feature of DNA. Reproduced from [24], with the permission of AIP Publishing.

fer also to these materials for a general introduction into the usage of LAMMPS.

To compile the code, load the LAMMPS standard packages `MOLECULE` and `ASPHERE` and the `CG-DNA` package by issuing
`make yes-molecule yes-asphere yes-cg-dna`
in the main source code directory and compile as usual.

All three versions oxDNA, oxDNA1.5 and oxDNA2 are implemented in the LAMMPS code and can be invoked through appropriate keywords in the input file. This allows for instance to run without sequence-specific interactions and without implicit ions (oxDNA force field and keyword `seqav` ≡ oxDNA), with sequence-specific interactions and without implicit ions (oxDNA force field and keyword `seqdep` ≡ oxDNA1.5) or with implicit ions and with or without sequence-specific interactions (oxDNA2 force field and keywords `seqdep` or `seqav`, respectively).

The source code is also distributed via our own GitHub repository at [26] under the project name *Coarse-Grained DNA Simulation (cgdna)*. Please send a request to join the project for full access that includes permission to browse the repository and commit changes.

B. Force and Torque Calculation

Integrating the equations of motion of rigid bodies requires accurate information of their relative orientations. In simple situations this can be achieved through Euler angles, which describe the orientation of a rigid body and its local reference frame with respect to the laboratory system. Euler angles have the disadvantage that they are not unambiguously defined as a singularity arises when two rotation axes fall parallel. This situation, usually referred to as gimbal lock, arises easily in a system that contains a large number of rigid bodies. Unsurprisingly, it triggers numerical instabilities, which is why rigid body problems are best formulated by means of quaternions [27] instead of Euler angles.

Computationally it is most efficient to integrate the quaternion degrees of freedom directly via a generalised 4-component quaternion torque (see [19] for a detailed derivation of the oxDNA forces and generalised 4-torques using quaternion dynamics). Unfortunately such an interface for generalised quaternion torques and momenta is not provided in LAMMPS. It expects for its rigid body integrators 3-component torques and angular momenta as input quantities (besides the Newtonian force for the integration of the coordinate degrees of freedom). To be consistent and simplify interfacing with existing functionality, we decided to adhere to this convention. This, however, entails conversion of the unit quaternions into Cartesian unit vectors of a body frame before forces and torques can be calculated for the integration step, thus leading to a computational overhead (see Appendix A).

Once this choice has been made, the calculation of the forces and torques is most conveniently formulated following Ref. [28]. If $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{b}}$ are the principal axes of two rigid bodies A and B and r is the norm of the relative distance vector $\boldsymbol{r} = \boldsymbol{r}_A - \boldsymbol{r}_B$ from B to A, then the pair potential depends on a combination of these quantities,

$$U = U(r, \hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}) = U(r, \{\hat{\boldsymbol{a}}_m \cdot \hat{\boldsymbol{r}}\}, \{\hat{\boldsymbol{b}}_n \cdot \hat{\boldsymbol{r}}\}, \{\hat{\boldsymbol{a}}_m \cdot \hat{\boldsymbol{b}}_n\}) \quad (1)$$

where $\hat{\boldsymbol{r}}, \hat{\boldsymbol{a}}_m$ and $\hat{\boldsymbol{b}}_n$ are the normalised relative distance and orthonormal principal axes vectors. From this definition the force on A due to B are straightforwardly written as

$$\boldsymbol{F}_A = -\boldsymbol{F}_B = -\frac{\partial U}{\partial \boldsymbol{r}} = -\frac{\partial U}{\partial r}\hat{\boldsymbol{r}} - r^{-1}\sum_m \left[\frac{\partial U}{\partial(\hat{\boldsymbol{a}}_m \cdot \boldsymbol{r})}\hat{\boldsymbol{a}}_m^{\perp} + \frac{\partial U}{\partial(\hat{\boldsymbol{b}}_m \cdot \boldsymbol{r})}\hat{\boldsymbol{b}}_m^{\perp}\right] \quad (2)$$

Here $\hat{\boldsymbol{a}}_m^{\perp} = \hat{\boldsymbol{a}}_m - (\hat{\boldsymbol{a}}_m \cdot \hat{\boldsymbol{r}})\hat{\boldsymbol{r}}$ denotes the component of $\hat{\boldsymbol{a}}_m$ which is perpendicular to $\hat{\boldsymbol{r}}$. The torques are slightly more involved:

$$\boldsymbol{\tau}_A = \sum_m \frac{\partial U}{\partial(\hat{\boldsymbol{a}}_m \cdot \boldsymbol{r})}(\hat{\boldsymbol{r}} \times \hat{\boldsymbol{a}}_m)$$

$$-\sum_{mn}\frac{\partial U}{\partial(\hat{\boldsymbol{a}}_m\cdot\hat{\boldsymbol{b}_n})}(\hat{\boldsymbol{a}}_m\times\hat{\boldsymbol{b}}_n) \quad (3)$$

$$\boldsymbol{\tau}_B=\sum_n\frac{\partial U}{\partial(\hat{\boldsymbol{b}}_n\cdot\boldsymbol{r})}(\hat{\boldsymbol{r}}\times\hat{\boldsymbol{b}}_n) +\sum_{mn}\frac{\partial U}{\partial(\hat{\boldsymbol{a}}_m\cdot\hat{\boldsymbol{b}_n})}(\hat{\boldsymbol{a}}_m\times\hat{\boldsymbol{b}}_n). \quad (4)$$

The fact that local angular momentum conservation requires

$$\boldsymbol{\tau}_A+\boldsymbol{\tau}_B+\boldsymbol{r}\times\boldsymbol{f}=0 \quad (5)$$

can be conveniently utilised for debugging and verification purposes. The implementation was verified against two independent implementations, namely Ouldridge's own code, which is based on quaternion dynamics [19] as well as the standalone oxDNA code [20], which makes also use of the same scheme for the force and torque calculation. To this end two benchmarks were studied, a 5-base-pair duplex and a 8-base pair nicked duplex, which are both provided as examples in the CG-DNA package.

C. Input File

In the following we discuss the structure of the input file and how the newly introduced oxDNA2 classes are invoked.
We work with Lennard-Jones reduced units, which are invoked in LAMMPS via

```
units lj
```

The system is three-dimensional.

```
dimension 3
```

In LAMMPS, an oxDNA nucleotide is represented as a bonded-ellipsoidal hybrid particle with the associated degrees of freedom of bonded particles in a bead-spring polymer (backbone connectivity) and aspherical particles with shape (moment of inertia), quaternion (orientation) and angular momentum.

```
atom_style hybrid bond ellipsoid oxdna
```

Users are required to suppress the atom sorting algorithm as this can lead to problems in the bond topology of the DNA.

```
atom_modify sort 0 1.0
```

It is important to set the skin size correctly, which controls the extent of the neighbour lists. Too large a skin size and neighbour lists become unnecessarily long, leading to superfluous communication. Too short and partners in the pair interactions will be lost.

```
neighbor 3.8 bin
```

A good way to fine-tune this parameter is to run an NVE simulation with constant energy before applying Langevin integrators. We recommend `neighbor 3.8 bin` as a safe starting point, but this may be reduced. Likewise, frequent update of the neighbour lists can lead to an undue performance degradation. This parameter should be tuned as well so that no dangerous builds (as reported in the standard output of LAMMPS) occur.

```
neigh_modify every 1 delay 0 check yes
```

The initial configuration and topology is created by means of an external setup tool (see Sec. III D) and read in.

```
read_data data_file_name
```

All masses are set to 3.1575 in LJ units.

```
set atom * mass 3.1575
```

Note that the moment of inertia is determined through the shape parameter in the data file (see below Sec. III D). There are four types of nucleotides (A=1, C=2, G=3, T=4), which are grouped together into a group named `all` for the integration.

```
group all type 1 4
```

The oxDNA2 classes with its parameters are invoked as follows:

```
bond_style oxdna2/fene
bond_coeff * 2.0 0.25 0.7564
pair_style hybrid/overlay oxdna2/excv &
   oxdna2/stk oxdna2/hbond oxdna2/xstk &
   oxdna2/coaxstk oxdna2/dh
pair_coeff * * oxdna2/excv 2.0 0.7 0.675 2.0
&
   0.515 0.5 2.0 0.33 0.32
pair_coeff * * oxdna2/stk seqdep T &
   1.3523 2.6717 6.0 0.4 0.9 0.32 0.6 1.3 &
   0 0.8 0.9 0 0.95 0.9 0 0.95 2.0 0.65 &
   2.0 0.65
pair_coeff * * oxdna2/hbond seqdep 0.0 8.0 &
   0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 0.7 &
   1.5 0 0.7 0.46 3.141592653589793 0.7 &
   4.0 1.5707963267948966 0.45 &
   4.0 1.5707963267948966 0.45
pair_coeff 1 4 oxdna2/hbond seqdep 1.0678 &
   8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 &
   0.7 1.5 0 0.7 0.46 3.141592653589793 &
   0.7 4.0 1.5707963267948966 0.45 4.0 &
   1.5707963267948966 0.45
pair_coeff 2 3 oxdna2/hbond seqdep 1.0678 &
   8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 &
   0.7 1.5 0 0.7 0.46 3.141592653589793 &
   0.7 4.0 1.5707963267948966 0.45 4.0 &
   1.5707963267948966 0.45
pair_coeff * * oxdna2/xstk 47.5 0.575 &
   0.675 0.495 0.655 2.25 &
   0.791592653589793 0.58 1.7 1.0 0.68 1.7 &
   1.0 0.68 1.5 0 0.65 1.7 0.875 0.68 1.7 &
   0.875 0.68
pair_coeff * * oxdna2/coaxstk 58.5 0.4 0.6 &
   0.22 0.58 2.0 2.891592653589793 0.65 &
   1.3 0 0.8 0.9 0 0.95 0.9 0 0.95 40.0 &
   3.116592653589793
pair_coeff * * oxdna2/dh T rhos 0.815
```

The variables `T` and `rhos` stand here for the temperature and salt concentration in oxDNA units [20].

Please note that according to the LAMMPS parsing rules the ampersands (&) represent line breaks.
Visit the LAMMPS online documentation and manual for more information on oxDNA2.

D. Data File and Setup Tool

The data file contains all relevant structural parameters for the simulation, i.e. details about the number of atoms, the topology of the molecules, the size of the simulation box, initial velocities, etc. The LAMMPS implementation of oxDNA follows the standard form as discussed in the LAMMPS user manual. We outline the relevant parts below.

At the beginning of the data file the total number of particles and bonds has to be given. As we are using hybrid particles, we need to set the same number of ellipsoids. For a standard DNA duplex consisting of 8 complementary base pairs we need 16 atoms, 16 ellipsoids and 14 bonds, 7 on each of the two single strands. If the strands are nicked, which we do not assume here, the number of bonds would be reduced.

```
16 atoms
16 ellipsoids
14 bonds
```

We use four atom types to represent the four different nucleotides in DNA (A=1, C=2, G=3, T=4). We use only one bond type.

```
4 atom types
1 bond types
```

The dimensions of the simulation box are defined as follows:

```
-20.0 20.0 xlo xhi
-20.0 20.0 ylo yhi
-20.0 20.0 zlo zhi
```

Although already stated in the input file, we need to provide again the masses of the nucleotides.

```
Masses
1 3.1575
2 3.1575
3 3.1575
4 3.1575
```

The nucleotides are defined after the keyword `Atoms`. Each row contains the atom-ID (1,2,3 in the example below), the atom type (1,1,4), the position (x,y,z), the molecule ID (all 1 in this case), an ellipsoidal flag (1) and a density (1).

```
Atoms
1 1 0.00000  0.00000 0.00000 1 1 1
2 1 0.13274 -0.42913 0.37506 1 1 1
3 4 0.48461 -0.70835 0.75012 1 1 1
⋮
```

Next we set the initial velocities to the desired value, here all equal to 0. The first column contains the atom-ID (1,2,3), the following three columns the translational velocity and the last three columns the angular momentum.

```
Velocities
1 0.0 0.0 0.0 0.0 0.0 0.0
2 0.0 0.0 0.0 0.0 0.0 0.0
3 0.0 0.0 0.0 0.0 0.0 0.0
⋮
```

Note that this is our special choice in the setup tool. The velocities can be generally initialised to any value. Large values will lead to the FENE springs becoming overstretched and may provoke an early abortion of the run.
The ellipsoids are defined with atom-ID, shape (1.17398 to produce the correct moment of inertia) and initial quaternion (last four columns).

```
Ellipsoids
1 1.17398 1.17398 1.17398 1.00000 0.  0.  0.
2 1.17398 1.17398 1.17398 0.95534 0.  0.
0.29552
3 1.17398 1.17398 1.17398 0.82534 0.  0.
0.56464
⋮
```

Finally, we specify the bond topology. The first column contains the bond-ID (1,2,3), the second one the bond type (1) and the third and fourth the IDs of the two bond partners.

```
Bonds
1 1 1 2
2 1 2 3
3 1 3 4
⋮
```

The polarity of the DNA strand is encoded through the sequence in which atom IDs appear in the bond definition. For instance atom '1' in the first bond lies in 3'-direction, whereas atom '2' is in 5'-direction. It is advisable that the atom IDs are mostly consecutive and that lower atom IDs are used on the 3'-end. This convention is also used in the oxDNA standalone code.

Note that the `replicate` command in LAMMPS, which allows to replicate entities one or more times in each dimension, does not reproduce this information directly. In order to create a configuration that is based on copies of the original configuration, a run with 0 time steps with a `write_data` command at the end has to be performed first. This bypasses the Verlet loop and creates a new data file, which contains all entities (original and replicas) with the full bond information and will set the correct 3'-to-5' polarity correctly when read in.

To simplify the setup procedure we provide a simple python tool with the example and utility files of the CG-DNA package. The script allows the user to create single- and double-stranded DNA from an input file that specifies the sequence and requires an installation of `numpy`.

The syntax is very straightforward, but the system size has to be specified in the following way:

```
$> python generate.py <box_offset> \
     <cubic_box_length> <sequence_file_name>
```

The output is written directly into a data file in LAMMPS format. This has to be given in the LAMMPS input file. `<sequence_file_name>` is an ASCII input file that contains keywords and the sequence of one ssDNA strand. Two options are available. For a single, helical strand consisting of ssDNA, the sequence file contains a single line:

```
ACGTA
```

If the sequence is prepended by the keyword `DOUBLE`, then a single, helical DNA duplex is created. The bases on the second strand are complementary to those on the first strand, which is given in the sequence input file:

```
DOUBLE ACGTA
```

Consecutive strands are positioned and oriented randomly without creating any overlap in case of more than one ssDNA or dsDNA strand. Note that the procedure works only below a critical density as this simple script does not feature cell lists. Besides these setup tools, the CG-DNA package contains as well example input, data and standard output files of short benchmark runs of dsDNA duplexes.

E. Output and Visualisation

LAMMPS offers a multitude of possible output formats, including parallel HDF5 and NetCDF formats, VTK format or very basic standard trajectory data. We will summarise here how output of basic observables of the oxDNA model can be invoked in the input file.

The xyz style writes XYZ files, which is a simple text-based coordinate format that many codes can read, which has one line per atom with the atom type and the x-, y-, and z-coordinate of that atom. This style is invoked via

```
dump 1 all xyz Nint trajectory.xyz
```

where `Nint` is the output frequency in timesteps. Additional output of e.g. velocity, force and torque on a per-atom basis makes some customisation necessary,

```
dump 2 all custom Nint filename.dat id x y z
&
    vx vy vz fx fy fz tqx tqy tqz
```

where `id` is the unique atom-ID. The output of quaternions requires a so-called `compute` style. The result of the `compute` style can then be retrieved in the following way:

```
compute quat all property/atom quatw quati &
    quatj quatk
dump 3 all custom Nint filename.dat id &
    c_quat[1] c_quat[2] c_quat[3] c_quat[4]
```

Another observable that may be of interest is the energy, or more specifically broken down into rotational, kinetic and potential energy. This is also done through a `compute` style.

```
compute erot all erotate/asphere
compute ekin all ke
compute epot all pe
variable erot equal c_erot
variable ekin equal c_ekin
variable epot equal c_epot
variable etot equal c_erot+c_ekin+c_epot
```

Note that the somewhat simpler `thermo_style` command for output discards the kinetic energy of rotation when the kinetic energy is requested.

LAMMPS does not contain a direct visualisation toolkit and there are multiple ways how snapshots can be visualised. ParaView [29], Ovito [30] or VMD (Visual Molecular Dynamics) [31] are open source, multiplatform data analysis and visualisation applications (see LAMMPS online manual [13] for more information about possible visualisation pipelines). We want to highlight *oxView* [32, 33], the bespoke visualisation tool for oxDNA. It offers a convenient way to show the base orientations. Using *oxView* requires transformation of the LAMMPS data format into the native oxDNA format. This can be achieved through *TacoxDNA* [34, 35], the suite of tools and converters, but requires a specific LAMMPS output format that contains the quaternions and angular momenta. The command for the LAMMPS input script is given here below:

```
dump 4 all custom Nint filename.dat &
    id mol type x y z ix iy iz vx vy vz &
    c_quat[1] c_quat[2] c_quat[3] c_quat[4] &
    angmomx angmomy angmomz
dump_modify out sort id
dump_modify 4 format line "&
    %d %d %d %22.15le %22.15le %22.15le &
    %d %d %d %22.15le %22.15le %22.15le &
    %22.15le %22.15le %22.15le %22.15le &
    %22.15le %22.15le %22.15le"
```

IV. LANGEVIN-TYPE RIGID-BODY INTEGRATORS

Together with the CG-DNA package comes also an implementation of novel Langevin-type rigid-body integrators that were developed by Davidchack, Ouldridge and Tretyakov [36]. The motivation for this was that previously only a limited choice of suitable Langevin integrators for rigid bodies was available in LAMMPS. Without noise all integrators A, B and C in the above reference are identical and basically equivalent to the integrator presented by Miller et al. [37]. Nevertheless, we refer to this case as the "DOT integrator" (the other implementation of the Miller integrator is only available when using the `fix rigid` command in LAMMPS). The DOT integrator is an alternative to the standard LAMMPS NVE integrator

for aspherical particles, and can be invoked by replacing the standard choice

```
fix 1 all nve/asphere
```

with

```
fix 1 all nve/dot
```

in the input file. This energy-conserving integrator is useful for an analysis of the accuracy of this family of integrators or the integrity of the pair interactions at a given timestep size Δt.

The C integrator in Ref. [36], to which we refer as "DOT-C integrator", is invoked by replacing the standard NVE integrator for aspherical particles and the fix for Langevin dynamics

```
fix 1 all nve/asphere
fix 2 all langevin 0.1 0.1 0.03 457145
angmom 10
```

with one single fix

```
fix 1 all nve/dotc/langevin 0.1 0.1 0.03 &
     457145 angmom 10
```

To measure the accuracy of the new integrators, we run a test case consisting of a short, nicked duplex with 8 base pairs (16 nucleotides). Fig. 3 shows the accuracy measured through the normalised difference between the total energy E_{tot} for this particular benchmark and the total energy at the beginning of the run E^*_{tot}. We compared the standard `fix nve/asphere` integrator, which is based on a Richardson iteration in the update of the quaternion degrees of freedom, to the new DOT integrator, which uses a rotation sequence to update the quaternions. Shown are results for two different timestep sizes $\Delta t = 10^{-3}$ and $\Delta t = 10^{-4}$. Both simulations were run for the same physical simulation time to allow direct comparison of the deviations of a dynamical run. As this is done in the NVE ensemble and without noise, the energy should be exactly conserved. This corresponds to a straight, horizontal line at 0.

It is obvious that above a certain timestep size the accuracy of the new DOT integrator is slightly inferior compared to the standard integrator. Up to a certain point the DOT integrator actually seems to deviate further from the correct result, whereas the standard integrator fluctuates more around the correct value. This, however, is more or less a transient effect as longer runs show there is no permanent drift away from the correct result. For Langevin dynamics, it is not possible to evaluate the accuracy and stability in the same way. We opted instead for an estimate based on the average kinetic, rotational, potential and total energy of the benchmark. Again, we performed runs of $\tau = 10000$ Lennard-Jones time units length, this time thermalised, and averaged the results over the time interval. The number of MD-timesteps and the output frequency for each timestep size were adapted so that the total physical simulation time and the statistical basis of the error calculations were consistent. The temperature in reduced LJ-units was set to $T = 0.1$, whereas the translational and rotational friction or damping coefficients were set to

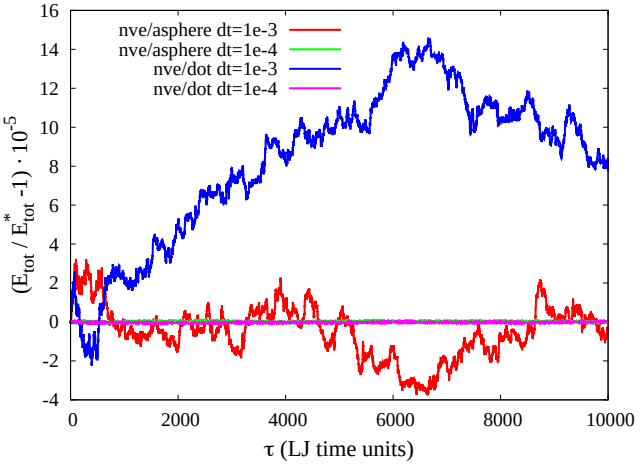

FIG. 3. Relative normalised accuracy $(E_{tot} - E^*_{tot})/E^*_{tot}$ of the standard LAMMPS NVE integrator for aspherical particles and the NVE DOT integrator from Ref. [36]. E^*_{tot} is the total free energy at the beginning of the simulation runs.

$\gamma = 1/0.03$ and $\Gamma = 1/0.3$, respectively. The results are summarised in Tab. I. These values were used during the verification of the LAMMPS implementation because they produced relatively smooth trajectories that could be easily followed. For actual production runs it may be more appropriate to use different values to allow a better and more efficient sampling of the configuration space.

Based on three translational and three rotational degrees of freedom per nucleotide and 8 base pairs we expect kinetic and rotational energies $E_{kin} = E_{rot} = 2.4$ for a temperature settings $T = 0.1$. This is very well achieved for all timestep sizes and both integrators, the standard LAMMPS integrator `fix nve/asphere & fix langevin` and the DOT-C integrator `fix nve/dotc/langevin`. However, there appears to be a slight decrease in the DOT-C integrator for very large step sizes ($\Delta t = 2 \cdot 10^{-2}$). The deviation of the total energy between all timestep sizes, admittedly an *ad hoc* criterion to quantify the stability of the integrators, but one that is rather hard for the integrators to get exactly right, is in the sub-percent range. It is actually slightly better for the DOT-C integrator than for the standard LAMMPS integrator. The statistical errors, reported in Tab.I, are the standard deviations of a linear least square fit and show that the deviations are well above the uncertainty of the fits.

Remarkably, for the DOT-C integrator the limit for a stable integration is $\Delta t = 2 \cdot 10^{-2}$, which represents a very large timestep size. This is about 4 times larger than the maximum timestep size for which the standard LAMMPS Langevin integrator produces sound results. Because of the more complex rotations in quaternion space and various additional transformations that the DOT-C integrator requires there is a

	Δt	E_{kin}	E_{rot}	E_{pot}	E_{tot}	standard error of E_{tot} fit
fix nve/asphere & fix langevin						
	10^{-4}	2.3999	2.4001	-21.4512	-16.6513	± 0.00377 (0.0227%)
	10^{-3}	2.4015	2.4021	-21.5564	-16.7582	± 0.00349 (0.0208%)
	$5 \cdot 10^{-3}$	2.4012	2.3999	-21.6352	-16.8315	± 0.00322 (0.0191%)
nve/dotc/langevin						
	10^{-4}	2.3989	2.3997	-21.5278	-16.7292	± 0.00362 (0.0216%)
	10^{-3}	2.3998	2.4008	-21.6631	-16.8624	± 0.00335 (0.0199%)
	10^{-2}	2.3959	2.3941	-21.6151	-16.8251	± 0.00318 (0.0189%)
	$2 \cdot 10^{-2}$	2.3895	2.3752	-21.6266	-16.8619	± 0.00313 (0.0185%)

TABLE I. Average kinetic, rotational, potential and total energy for the standard LAMMPS integrator `fix nve/asphere` & `fix langevin` and the DOT-C integrator `nve/dotc/langevin` for different timestep sizes.

small overhead of about 15% compared to the standard LAMMPS integrator. Nevertheless, this small overhead of the DOT-C integrator is very well compensated by the computational efficiency and possibility to increase the timestep size by 400% (from a maximum of $\Delta t = 5 \cdot 10^{-3}$ for the standard LAMMPS integrator to $\Delta t = 2 \cdot 10^{-2}$ for the DOT-C integrator).

V. PERFORMANCE ANALYSIS

A detailed analysis of the performance of oxDNA in LAMMPS and in the standalone code was recently carried out [23]. Here, we show results of two simple benchmarks to study the parallel performance of the LAMMPS implementation. The size of each benchmark is well beyond the current capabilities of the standalone CPU version, so each demonstrates as well a minimal performance requirement. The benchmarks

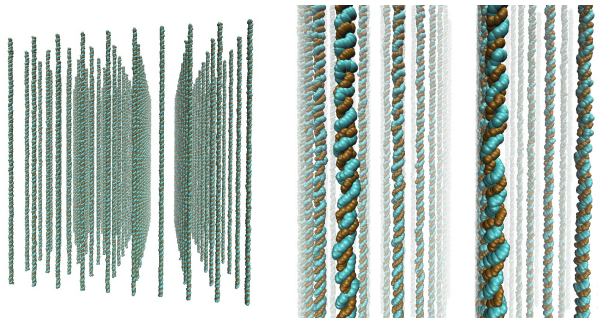

FIG. 4. The low-density benchmark consisting of a 10×10 array of DNA duplexes with A-T base pairs and a length of 600 base pairs each, in total 60 kbp. The high-density benchmark (not shown) consisted of a similar 40×40 array of duplexes with 960 kbp in total. The pictures show the final configuration the end of a performance run and were produced with VMD. The centre of mass of each nucleotide is represented through a sphere.

consisted of arrays of double-stranded, regularly arranged DNA duplexes, each with a length of 600 base pairs. The low-density (LD) benchmark was formed by a 10×10 array of duplexes, giving a total of 60 kbp, and is shown in Fig. 4. The high-density (HD) benchmark was formed by a 40×40 array of duplexes with a density 16 times larger than the LD case and a total number of 960 kbp. Whilst a regular array of double-stranded DNA strands appears perhaps somewhat artificial, it creates a reasonably load-balanced situation and facilitates the performance analysis. The obtained densities of DNA, are however very well comparable to those of DNA gels [38] and high density states of DNA which form liquid-crystalline phases [39].

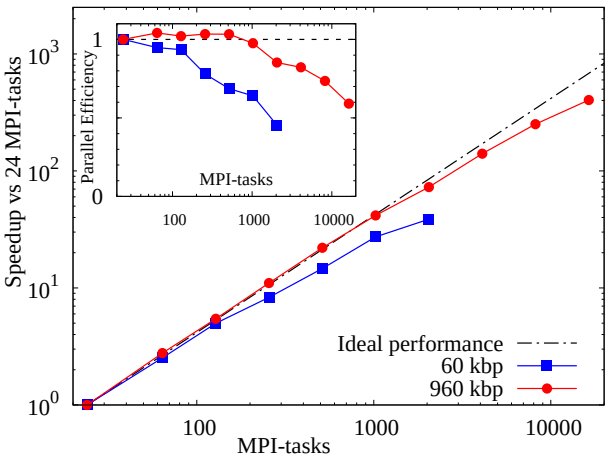

FIG. 5. Strong scaling behaviour: Speedup of the low and high density benchmarks of 60 kbp and 960 kbp, respectively, compared to the single node performance with 24 MPI-tasks. The inset shows the parallel efficiency relative to the single node case with 24 MPI-tasks.

Strong scaling tests were performed on ARCHER on up to 86 nodes (LD) and 683 nodes (HD), respectively. The benchmark cases were run for 30,000 (LD) and 10,000 (HD) MD-timesteps with a timestep size of $\Delta t = 5 \times 10^{-3}$. We used the standard LAMMPS integrators for Langevin dynamics, although the scal-

ing behaviour was found to be virtually identical when using the above described rigid body integrator DOT-C. The primary reason for this was that the wallclock time for runs with the standard integrator was still a few percent shorter, although the improved efficiency of the DOT-C integrator would mean these runs were shorter in physical time. The temperature in reduced LJ-units was $T = 0.1$, whereas the translational and rotational friction coefficients were set to $\gamma = 1/0.03$ and $\Gamma = 1/0.3$, respectively.

Fig. 5 shows the parallel speedup for both benchmarks relative to the single node performance with 24 MPI-tasks. The code performs well for the LD benchmark up to about 128 MPI-tasks with a parallel efficiency around 95% (see the inset). Beyond several hundred MPI-tasks a gradual performance degradation is observed. At 2048 MPI-tasks the parallel efficiency has decreased to about 45% and the total speedup is roughly 930-fold compared to the single core performance (39-fold compared to the single node performance).
A look at the ratio of the number of local atoms, i.e. those that are inside a process boundary, to the number of ghost atoms, i.e. those which need to be communicated via neighbour lists, proves that the observed performance degradation is due to the comparably small size of the problem. At the largest core counts there are on average only about 60 local atoms present on each process, whereas the number of ghost atoms is with about 225 atoms almost four times larger. LAMMPS is known to require at least a few hundred local atoms or more for a good parallel performance [40]. The speedup is still relatively good because the fraction of time that the algorithm spends in the force calculation is still comparably large. For the HD benchmark, 16 times larger than the LD case, the performance degradation is more or less mirrored at core counts that are about 16 times larger. For the HD benchmark the total speedup at 16384 MPI-tasks is 9680-fold with respect to the single core performance (400-fold compared to the single node performance) and the parallel efficiency is still at around 60%.

These two examples are of course slightly idealised in the sense that both benchmarks fulfil easily the requirement of good load-balancing, which is necessary to obtain a good scaling performance. LAMMPS, however, features sophisticated load-balancing algorithms which permit good scaling behaviour also for very inhomogeneous systems. We are planning to extend the existing implementation to benefit further from recent developments pertaining to threaded parallelisation on shared memory architectures such as many-core chips and general purpose graphical processing units (GPGPUs).

One of the major advantages of the new LAMMPS implementation is that it can be directly compared with other coarse-grained models that are also based on the LAMMPS code. To this end, we compared the single core performance of oxDNA2 with that of 3SPN.2 [14]. The benchmark consisted of two complementary dsDNA duplexes of 8 bps with implicit ions. In order to compare both models we set the translational friction coefficient γ to about $(300\ \text{fs})^{-1}$. We opted for the maximum timestep size that provided a stable integration, which was $\Delta t = 35$ fs (3SPN.2) and $\Delta t = 48$ fs (oxDNA2 + DOT-C integrator), respectively.

On a single Intel Core i7 2.8 GHz processor using the latest version of LAMMPS (16 March 2018) 3SPN.2 delivered a performance of about 60 μs per day. oxDNA2 was able to surpass this by about a factor 1.6 with a performance of roughly 100 μs per day. Note that comparing the wall times is only an approximate way to compare the performance as there is no guarantee that similar processes take a similar simulation time in the two models.

Apart from the enhanced stability of the rigid body integrator, this difference in performance will be caused by the different number of degrees of freedom that both models require: oxDNA/oxDNA2 uses only 13 degrees of freedom per nucleotide (3 coordinate positions, 3 translational momenta, 3 angular momenta and 4 quaternion degrees of freedom), whereas 3SPN.2 uses 18 degrees of freedom per nucleotide (3 particles with each 3 coordinate positions and translational momenta).

Unfortunately, we could not measure the parallel performance of 3SPN.2. But this conceptual difference between the two models is very likely to entail further detrimental effects when running in parallel. With the larger number of degrees of freedom per nucleotide in 3SPN.2, communication overheads are likely to build up more quickly and neighbour lists are longer and probably have to be rebuilt more frequently. On the other hand, the current LAMMPS implementation of oxDNA offers further potential for optimisation as it spends a good part its time computing the inverse cosine (around 12%, see Appendix A). This could be alleviated for instance through the introduction of appropriate lookup tables for trigonometric functions.

ACKNOWLEDGMENTS

This work was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service (eCSE05-10). OH acknowledges support from the EPSRC Early Career Fellowship Scheme (EP/N019180/2).

[1] Calladine, C., Drew, H. R., Luisi, B. F. & Travers, A. A. *Understanding DNA* (Elsevier Academic Press, London, UK, 2004).

[2] Laughton, C. A. & Harris, S. A. *WIREs Comput. Mol. Sci.* **1**, 590 (2011).

[3] Potoyan, D. A., Savelyev, A. & Papoian, G. A. *WIREs Comput. Mol. Sci.* **3**, 69 (2013).

[4] Dans, P. D., Walther, J., Gómez, H. & Orozco, M. *Curr. Opin. Struct. Biol.* **37**, 29 (2016).

[5] Maffeo, C., Ngo, T. T. M., Ha, T. & Aksimentiev, A. *J. Chem. Theory Comput.* **10**, 2891 (2014).

[6] Korolev, N., Luo, D., Lyubartsev, A. & Nordenskiöld, L. *Polymers* **6**, 1655 (2014).

[7] Maciejczyk, M., Spasic, A., Liwo, A. & Scheraga, H. *J. Chem. Theory Comput.* **10**, 5020 (2014).

[8] Pronk, S. *et al. Bioinformatics* **29**, 845 (2013).

[9] Machado, M. R. & Pantano, S. *J. Chem. Theory Comput.* **32**, 1568 (2016).

[10] Uusitalo, J. J., Ingolfsson, H. I., Akhshi, P., Tieleman, D. P. & Marrink, S. J. *J. Chem. Theory Comput.* **11**, 3932 (2015).

[11] Phillips, J. C. *et al. J. Comput. Chem.* **26**, 1781 (2005).

[12] Markegard, C., Fu, I., Reddy, K. & Nguyen, H. *J. Chem. Phys. B* **119**, 1823 (2015).

[13] `https://www.lammps.org`.

[14] Hinckley, D. M., Freeman, G. S., Whitmer, J. K. & de Pablo, J. J. *J. Chem. Phys.* **139**, 144903 (2013).

[15] Hinckley, D. M. & de Pablo, J. J. *J. Chem. Theory Comput.* **11**, 5436 (2015).

[16] Brackley, C. A., Morozov, A. N. & Marenduzzo, D. *J. Chem. Phys.* **140**, 135103 (2014).

[17] Fosado, Y. A. G., Michieletto, D. & J. Allan, e. a. *Soft Matter* **12**, 9458 (2016).

[18] Ouldridge, T. E., Louis, A. A. & Doye, J. P. K. *J. Chem. Phys.* **134**, 085101 (2011).

[19] Ouldridge, T. *Coarse-grained modelling of DNA and DNA self-assembly.* Ph.D. thesis, University of Oxford, 2011 (2011).

[20] `https://dna.physics.ox.ac.uk`.

[21] Holbrook, J., Capp, M., Saecker, R. & Record, M. *Biochemistry* **38**, 8409 (1999).

[22] SantaLucia Jr, J. & Hicks, D. *Annu. Rev. Biophys. Biomol. Struct.* **33**, 415 (2004).

[23] Sengar, A., Ouldridge, T. E., Henrich, O., Rovigatti, L. & Sulc, P. A primer on the oxDNA model of DNA: When to use it, how to simulate it and how to interpret the results. *Front. Mol. Biosci.* **8**, 693710 (2021). 2104.11567.

[24] Snodin, B. *et al. J. Chem. Phys.* **142**, 234901 (2015).

[25] Sulc, P. *et al. J. Chem. Phys.* **137**, 135101 (2012).

[26] `https://github.com/ohenrich/cgdna`.

[27] Allen, M. P. & Tildesley, D. J. *Computer Simulation of Liquids* (Oxford University Press, Oxford, UK, 1989).

[28] Allen, M. & Germano, G. *Mol. Phys.* **104**, 3225 (2006).

[29] `https://www.paraview.org`.

[30] `https://www.ovito.org`.

[31] Humphrey, W., Dalke, A. & Schulten, K. VMD – Visual Molecular Dynamics. *J. Mol. Graphics* **14**, 33–38 (1996).

[32] Poppleton, E. *et al.* Design, optimization and analysis of large DNA and RNA nanostructures through interactive visualization, editing and molecular simulation. *Nucleic Acids Res.* 1–12 (2020).

[33] `https://github.com/sulcgroup/oxdna-viewer`.

[34] Suma, A. *et al.* TacoxDNA: A user-friendly web server for simulations of complex DNA structures, from single strands to origami. *J. Comput. Chem.* **40**, 2586–2595 (2019).

[35] `https://github.com/lorenzo-rovigatti/tacoxDNA`.

[36] Davidchack, R. L., Ouldridge, T. E. & Tretyakov, M. V. *J. Chem. Phys.* **142**, 144114 (2015).

[37] Miller, T. F., Eleftheriou, M., Pattnaik, P., Ndirango, A. & Newns, D. *J. Chem. Phys.* **116**, 8649 (2002).

[38] Rovigatti, L., Smallenburg, F., Romano, F. & Sciortino†, F. *ACS Nano* **8**, 3567 (2014).

[39] Michele, C. D., Rovigatti, L., Bellinic, T. & Sciortino, F. *Soft Matter* **8**, 8388 (2012).

[40] LAMMPS Documentation. `https://docs.lammps.org/Manual.html` (2021). Chapter 7.1: Benchmarks.

APPENDIX

Appendix A: Profiling

Profiling allows a detailed analysis of the implementation and gives an overview of how much time the code spends in each individual subroutine. We used the Craypat Performance Tools on the ARCHER UK National Supercomputing Service to conduct sampling experiments of the high and low density benchmarks. Although the experiments where actually performed with the oxDNA model, the results are representative as well for oxDNA2 as the only difference between the two is a different local geometry of the interaction sites and an additional pair interaction in form of a Debye-Hückel potential.

Fig. 6 shows a pie chart of the low density (LD) run. The image on the left shows the results on a single node with 24 MPI-tasks, whereas the image on the right is for 2048 MPI-tasks. Focussing first on a single node, calls to the MPI-library are below 5% and do not appear with an individual pie section. The total time spent in the force calculation is around 86% (according to the LAMMPS breakdown). Interestingly, a significant fraction of the time is spent on calculating the local body coordinate system of the nucleotide from the quaternion degrees of freedom (MathExtra::q_to_exy, 11.3%).

A significant portion falls also on the calculation of the inverse cosine (acos, 12.1%). The conversion from quaternions to 3-vectors is done separately in every single interaction. This has been done for simplicity, but represents a 6-fold overhead as it could be optimised by calculating the 3-vectors only once per timestep, then saving the for later use by the interactions. This optimisation would come at increased communication as the additional nine components of the three unit vectors would have to be communicated across the process boundaries. Another pos-

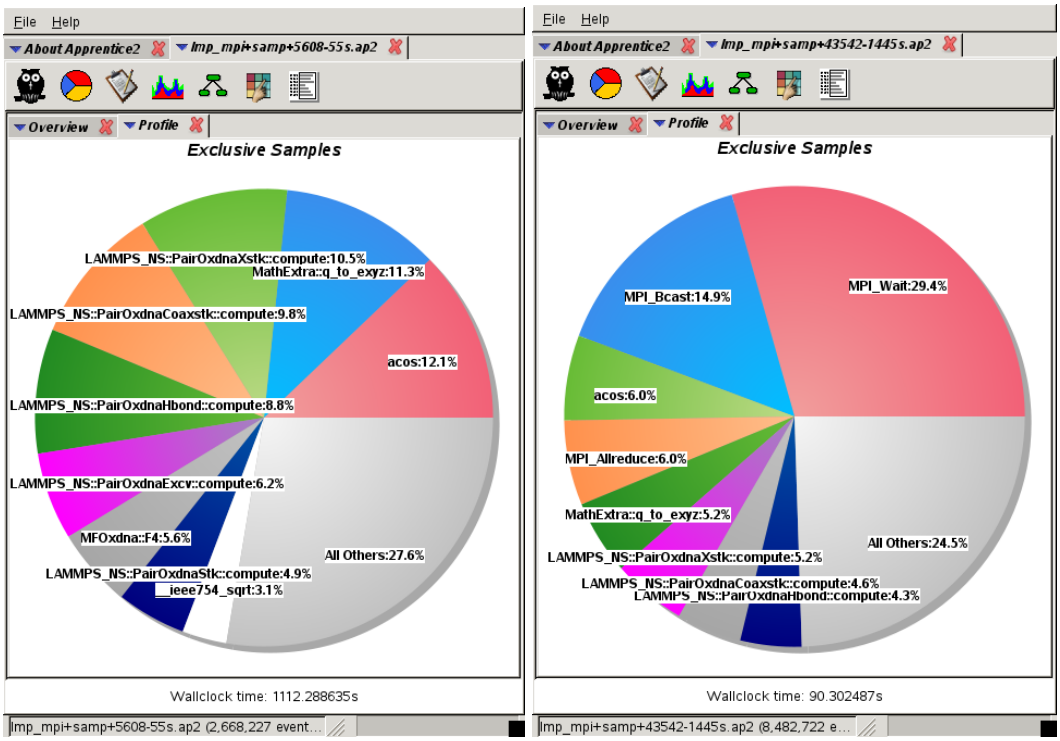

FIG. 6. Craypat performance analysis of a sampling experiment for the low density benchmark (60 kbp) on a single node (left, 24 MPI-tasks) and for 2048 MPI-tasks (right). Note that the assigned colour code for the functions is different in both cases.

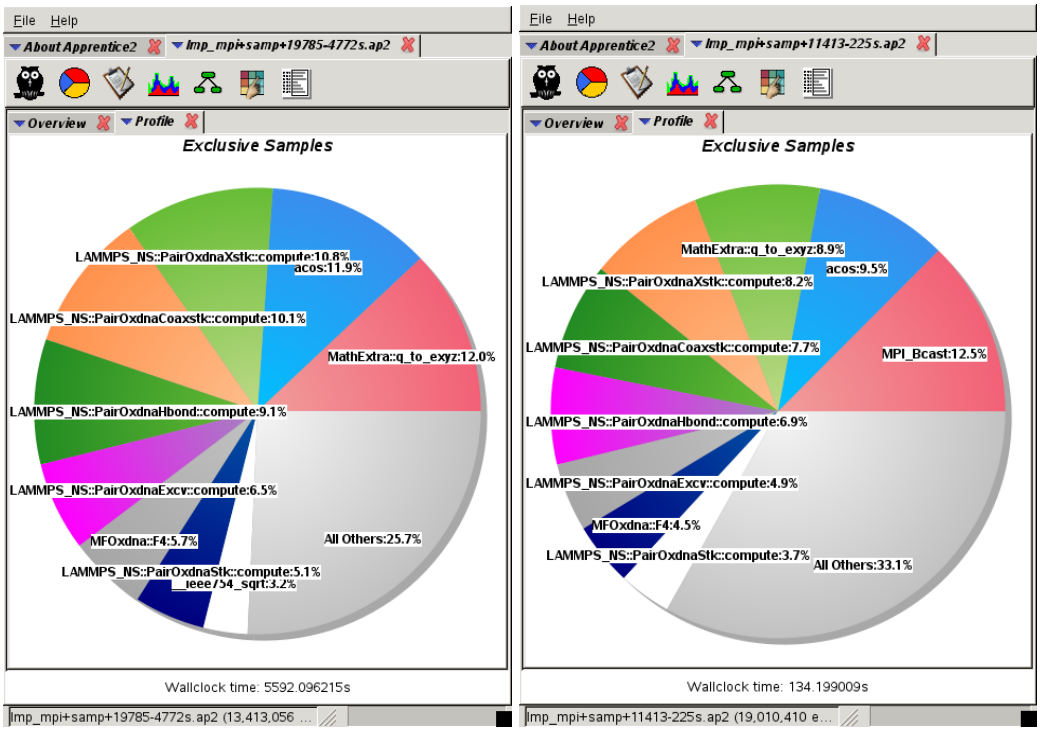

FIG. 7. Craypat performance analysis of a sampling experiment for the high density benchmark (960 kbp) on a single node (left, 24 MPI-tasks) and for 2048 MPI-tasks (right). Note that the assigned colour code for the functions is different in both cases.

sibility, and a major adaptation, would be to formulate the entire force calculation in generalised quaternion forces and torques, therefore avoiding the transformation in the first place. We decided deliberately against this possibility as this would require calculation of four force and torque components in quaternion space. The calculation with 3-vectors on the other hand, as currently implemented, requires only three force and torque components. Perhaps most importantly, they can be made available directly to the other LAMMPS routines. It is thus very likely that a performance gain from avoiding the transformation would be outweighed either by the larger number of additional components and generalised quaternion forces and torques which also had to be communicated across the process boundaries or by disadvantages from a software engineering point of view.

The large fraction of the inverse cosine is more difficult to optimise. It emerges in the stacking, cross- and coaxial stacking and hydrogen bonding interactions through a partial derivative with respect to the relative distances. A previous version of the implementation spent a whopping 29% of its time calculating the inverse cosine. This prohibitively large figure could be cut down to the current 12% by introducing appropriate early-rejection criteria in each force calculation. Further improvements might be possible through small-argument approximations of the inverse cosine. This will be tested in a future version of the code (e.g. for the upgrade to oxDNA 2.0).

At 2048 MPI-tasks, shown on the right of Fig. 6, the code spends more than 50% of its time in call to the MPI-library. The percentage of time in the force calculation has fallen to about 43%. As stated above, this is primarily the consequence of an insufficient number of local atoms with respect to the number of ghost atoms, and does not reflect a problem with the parallel performance of the implementation.

For the HD benchmark on a single node, shown on the left in Fig. 7, calls to the MPI-library are below 3%. The conversion of quaternions to 3-vectors (MathExtra::q_to_exyz) and the calculation of the inverse cosine (acos) are constant at about 12%. At 2048 MPI-tasks we observe a parallel efficiency of about 85%. The time spent in the force calculation is still about 82% (according to the LAMMPS breakdown) with calls to the MPI-library amounting to just below 13%. The CPU time of the quaternion conversion to the local body frame of the nucleotide and the inverse cosine each at are around 9% due to the larger share of the calls to the MPI-library.