Interfacing novel integrators and existing Molecular Dynamics codes with the MIST library

Iain Bethune¹, Elena Breitmoser¹, Antonia B K Collis¹, Gianpaolo Gobbo² and Benedict J Leimkuhler²

* Corresponding Author: ibethune@epcc.ed.ac.uk

¹ EPCC, The University of Edinburgh, Peter Guthrie Tait Road, Edinburgh EH9 3FD, UK
² Maxwell Institute for Mathematical Sciences and School of Mathematics, The University of Edinburgh,
  Peter Guthrie Tait Road, Edinburgh EH9 3FD, UK

Background

The need for extremely long simulation timescales to fully sample the conformational space of complex macromolecules with molecular dynamics is well known [1] and one of the key drivers of the UK-US ‘ExTASY’ project. In addition to established advanced sampling techniques such as Metadynamics [2], Replica Exchange MD [3] or Diffusion-Map-directed-MD [4] and the use of optimised Molecular Dynamics software (e.g. NAMD [5] or GROMACS [6]) and specialised hardware [7], there is also a growing demand for novel integration algorithms to either extend the simulation timestep, or to incorporate collective variable information on-the-fly in order to increase the sampling rate. Most often, new algorithms are first implemented in research codes which lack the performance or additional features to be used for production simulations. For example the rigid body rotational scheme of Dullweber et al [8] was first implemented in the ORIENT package, and is still not widely available, despite having provably better energy conservation properties in long simulations due to its symplectic and time-reversible nature. Likewise, a Langevin Dynamics integrator based on a BAOAB splitting scheme [9] is available only in the NAMD-Lite package, which is too slow for practical use on large systems.

The root cause of this problem is a software complexity barrier that comes from the highly optimised performance of production MD codes, which creates a catch-22 for integrator developers: if new algorithms cannot easily be incorporated into widely used MD packages, it may be impossible to demonstrate the benefits of the new algorithms at scale and enable uptake by the MD user community, and without demand from the user community the MD package developers will not implement new integration algorithms in their codes themselves.

The Molecular Integration Software Tools Library

To overcome this barrier we have developed the Molecular Integration Software Tools (MIST) library - a lightweight abstraction of the functionality of a Molecular Dynamics code that allows integrators to be developed using a simple Application Programming Interface (API) without exposing the complexity of the underlying software while still maintaining excellent performance. MIST is implemented as a C++ library, and is freely available from http://www.extasy-project.org/mist under a BSD licence. A schematic representation of the MIST library is shown in figure 1. The library provides two key abstractions which are of use to integrator developers:

- MIST represents the state of an atomistic simulation as a System class, consisting of a set of particles, with various properties: position, velocity, mass, species, and per-particle forces. A simple API is provided to get and set the properties of each particle. In addition, accessors are provided for global quantities such as the potential energy, and a single function call updates the forces on the particles given the current state of the system. Adaptor subclasses are provided for several supported MD codes, which implement the System API using the data structures present in that code.
An Integrator in MIST is a class which provides only a single method, which integrates the system over a timestep from \( t \) to \( t + dt \). Normally, particle positions and velocities are defined at both the start and the end of the step, although the library also supports schemes where the positions and velocities are offset in time (for example positions at \( t \) and velocities at \( t + dt/2 \)). Integrators are implemented using only the System API, completely independent of any particular MD code. Several examples are provided, including Velocity Verlet and Verlet Leapfrog integrators, 4th and 8th order Yoshida symplectic integrators [10], Langevin dynamics using the BAOAB scheme [9], and a ‘Continuous Tempering’ extended Hamiltonian scheme [11].

![MIST library architecture](image)

**Figure 1: MIST library architecture**

Using the System API, the implementation of the Velocity Verlet algorithm is as simple as:

```c
void VerletIntegrator::Step(double dt)
{
    VelocityStep(0.5 * dt);
    PositionStep(dt);
    system->UpdateForces();
    VelocityStep(0.5 * dt);
}
```

The MIST library may be interfaced to several production MD codes, enabling the benefits of the new integration algorithms to be combined with the highly optimised force evaluation, existing system setup and I/O schemes with minimal overhead. MIST also provides a C and FORTRAN API to enable integration with ‘host’ MD codes. MIST API calls are inserted in place of the usual MD step via source-code patches which are distributed with the MIST library and applied during the build process. At present, MIST has plug-ins for GROMACS, NAMD-Lite and AMBER. The main simplifying assumption that MIST makes in the current release is that all of the particles are local to a single process - so distributed memory parallelism such as MPI is not supported. Nevertheless, both shared memory parallelism (OpenMP) and/or the use of GPUs is allowed, depending on what is implemented in the host code, so high performance can be achieved for moderate sized systems.

To use MIST integrators with a particular MD code requires first patching and building the code, linked with the MIST library. At runtime, a single configuration flag in the input file to the MD code enables MIST, and all further configuration of the library is done via a separate `mist.params` file, for example
selecting which integrator to use and setting additional integrator-specific parameters such as a temperature for Langevin dynamics. All configuration of the force-field, simulation parameters like the timestep, initial coordinates and I/O is done as normal in the input file for the MD code.

**Performance**

The MIST library provides an abstract MD interface to make it easier to develop integrators, and so a single implementation can be portable to several MD codes. However, this is only useful if it does not come at too high a performance cost. The MIST library is designed to minimise the overhead associated with the interface between the MD code and library by giving the library direct access to the MD code’s data structures, rather than explicitly passing individual positions and momenta repeatedly across the interface. Likewise, updated forces are computed via a callback to the MD code’s existing optimised force evaluation routines.

To measure the impact of using a MIST integrator compared to the native integrator implemented in each MD code, we have simulated a 50Å³ box of water molecules using the flexible TIP3P model from the CHARMM forcefield. In all cases we used the Velocity Verlet scheme and an NVE ensemble with a timestep of 1 fs. Input files are available in the examples directory of the MIST distribution. In Figure 2, NO_MIST refers to a normal build of GROMACS, MIST_OFF refers to GROMACS built with MIST, but still using the native Verlet integrator, and MIST_ON is GROMACS using the Verlet integrator provided by MIST.

Tests were carried on ARCHER, a Cray XC30 with 2 12-core Intel Ivy Bridge 2.7 GHz processors per node. Figure 2(a) shows that for a serial calculation the overhead of performing the integration using MIST is negligible, since the vast majority of time is spent in GROMACS’ optimised force evaluation routines. GROMACS can take advantage of multiple OpenMP threads, and so as the number of threads in increased from 1 to 12 GROMACS achieves a speedup of 7.5x (NO_MIST). When MIST is used for integrating the equations of motion (MIST_ON), the overhead grows to 35% with 12 OpenMP threads due to the fact that in this test, OpenMP is not used within MIST, thus only the force evaluation part of the calculation is sped up. There is also a small overhead on the order of 10% or less that is observed when MIST is linked in but not used (MIST_OFF), due to the extra logic which is added to the GROMACS MD loop by the MIST patch.

![Figure 2(a): GROMACS performance on ARCHER using OpenMP for force computation only](image1)

![Figure 2(b): GROMACS performance on Hydra using OpenMP + Nvidia K20m GPU for force evaluation and OpenMP within MIST](image2)

Figure 2(b) shows data from Hydra, a Linux system with 2 16-core Intel Sandy Bridge 2.6 GHz processors per node, augmented with an Nvidia Tesla K20m GPU. In these tests as well as enabling
OpenMP and GPU acceleration within GROMACS, we also take advantage of OpenMP within the MIST integrators. Here we observe an overhead of around 5% for using MIST, which decreases to around 1% as the number of OpenMP threads increases. Since the majority of the force calculation is carried out by the GPU, GROMACS only obtains a speedup of 1.87x when running on 12 threads compared with 1. However, the integration step within MIST is more efficient, and so the overhead decreases on larger numbers of threads. It is worth noting that even on with the GPU plus a single CPU core, we can achieve a throughput of 40ns per day, compared to the maximum of 37.5ns per day obtained using an entire 12-core CPU on ARCHER.

We argue that the overheads of 5% or less that we have measure here show that MIST does indeed meet its dual aim of providing an abstraction with ease-of-use suitable for development of new integrators, while maintaining the excellent performance expected from highly-optimised produce MD codes.

Roadmap

Our vision is to extend MIST to interface with a wide range of MD codes including NAMD and LAMMPS, as well as provide additional common functionality such as constraint solvers, extended system approaches to constant temperature and pressure ensembles. MIST is intended as a vehicle both for the integrator development community to easily implement and test new algorithms on realistic production-scale systems, and also for a wide community of users to access the latest integration schemes. Already, MIST has been used for the first implementation of aforementioned Continuous Tempering scheme, where MD trajectories of Alanine-10 in vacuo for about 1 microsecond have been carried out using MIST with NAMD-Lite [11]. Simulations in explicit water using GROMACS with MIST are currently underway.

References