Molecular Integration Simulation Toolkit:
Interfacing novel integrators with Molecular Dynamics codes

Iain Bethune[1]*, Elena Breitmoser[1], Anton Martinsson[2], Zofia Trstanova[2], Ben Leimkuhler[2]
*i.bethune@epcc.ed.ac.uk
[1]EPCC, The University of Edinburgh [2]School of Mathematics, The University of Edinburgh

The ExTASY project was funded by NSF and EPSRC [EP/K039490/1].
This work used the ARCHER UK National Supercomputing Service (www.archer.ac.uk).

Background

Sampling the conformational space of complex macromolecules with Molecular Dynamics is hard! Despite significant effort in simulation software (e.g. GROMACS, NAMD, AMBER...) able to take advantage of hardware such as GPUs and parallel supercomputers, access to timescales of biological relevance – milliseconds to seconds – is still far from routine.

The ExTASY project proposes a set of Extensible Tools for Advanced Sampling and analYsis which take a three-pronged approach to the sampling problem:

- **More sampling** by using massive 'ensemble' simulations. 100s – 1000s of loosely coupled MD simulations executed automatically on HPC systems.
- **Smarter sampling** by coupling simulation with analysis tools to determine which regions of conformational space are well-sampled and promote exploration of under-sampled regions.

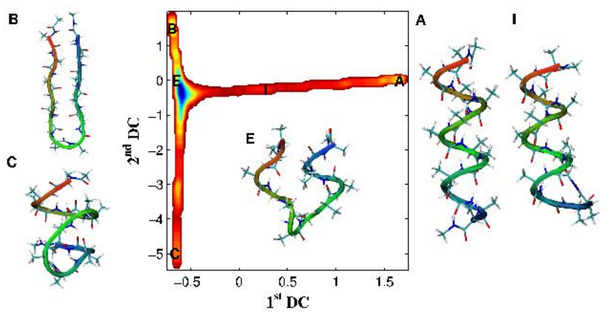

Figure 1: Free energy landscape of alanine-12, computed using Diffusion-map-directed-MD
Image from www.extasy-project.org

- **Faster sampling** by better integration algorithms, coupling the latest numerical schemes to existing highly-optimised MD codes to increase the base simulation time tep, and incorporate collective variable data on-the-fly.

The Molecular Integration Simulation Toolkit (MIST) library tackles the last of these objectives. The library is designed to meet the need of two user groups - for developers it provides a simple but powerful API for developing and testing new integration schemes; for users a suite of integrators is available which may be coupled to codes including GROMACS and AMBER with negligible loss of performance. Thus the complexity barrier associated with 'production' MD codes is overcome and new algorithms can quickly get into the hands of users.

MIST Library Architecture

MIST is implemented as a C++ library which provides two key abstractions, each implemented as an abstract class:

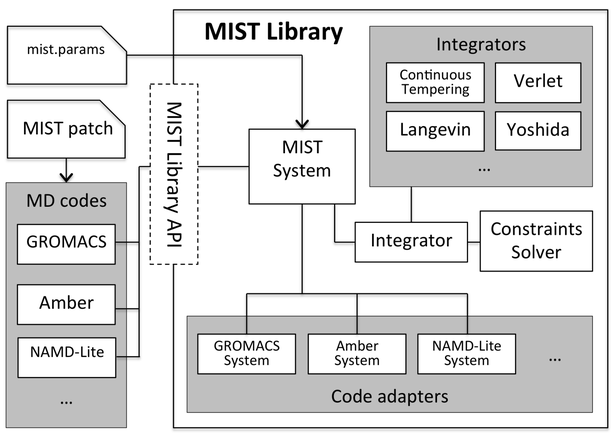

Figure 2: Schematic representation of MIST library structure

- The `System` represents the current simulation state, and a simple API provides access to 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. Adapters are provided for each supported MD code.
- An `Integrator` has a single method, which integrates the system from time `t` to `t+dt` and is implemented using only the System API, so completely independent of any particular MD code. Individual integrators are implemented as subclasses and so can be used interchangeably and have access to a set of convenience function for common tasks such as integrating positions and velocities.

MD codes are interfaced to MIST via calls to the MIST library C or Fortran90 API, inserted into the main MD loop by means of source-code patches. API calls give MIST all of the required information about the state of the system, for example the total number of particles, and pointers to the positions, velocities and masses. A callback function to compute updated forces is registered, which will be later used by MIST during integration. Once the library has been initialised, the usual contents of the MD loop are replaced by a single call to `MIST_Step(double dt)`, which advances the state of the system by a single timestep.

The MIST architecture ensures a clear separation of concerns – the modifications to the MD code are independent of which actual MIST integrator is selected by the user, thus once MIST support is added to a code it enables use of any of the set of integrators in the library; due to the `System` abstraction, the integrators are agnostic to the choice of MD code, so integrators need only be implemented once and can be used with any of the supported MD codes.

Using MIST

Once an MD code has been patched and built with the MIST library, to use MIST rather than a native integrator requires only a single modification to the MD code input file. For example, in a GROMACS mdp file, set `integrator = mist`. Selecting and configuring an integrator from within MIST is done through an additional file `mist.params`:

```
integrator langevin   # Select Langevin dynamics (BAOAB scheme)

langtemp 300          # Target temperature (K)
langfriction 1.0      # Friction constant (ps^-1)

constraints all-bonds # Apply constraints to all bonds
```

The file format consists of key/value pairs. At a minimum, the file should select an integrator, and any further parameters are applied by that integrator. A list of possible integrators and their parameters is on the MIST wiki: https://bitbucket.org/extasy-project/mist/wiki/MIST%20Integrators.

Constraints may be applied either to all bonds in the system or only bonds to hydrogen atoms. By default, a Symmetric Newton solver is used, although RATTLE is also available as an option.

Performance Tests

MIST is designed to simplify integrator development without incurring a significant performance overhead. This is achieved by inserting MIST library calls into the 'host' MD code via source-code patches. To measure the impact of using MIST compared to standard GROMACS, we simulated a 50Å^3 box of water molecules using a flexible TIP3P model from the CHARMM forcefield for a total of over 11,000 atoms. We used Verlet integration and an NVE ensemble with a time step of 1 fs. The input files for this system can be found in the examples directory of the MIST distribution.

CPU performance is measured on ARCHER, a Cray XC30 with Intel Ivy Bridge processors, and GPU performance on Hydra, a machine with Intel Sandy Bridge processors and an Nvidia Tesla K20m GPU. When using CPUs only, MIST matches the performance of native GROMACS to within 2%, and when using the GPU, at worst the overhead of using MIST was around 5%.

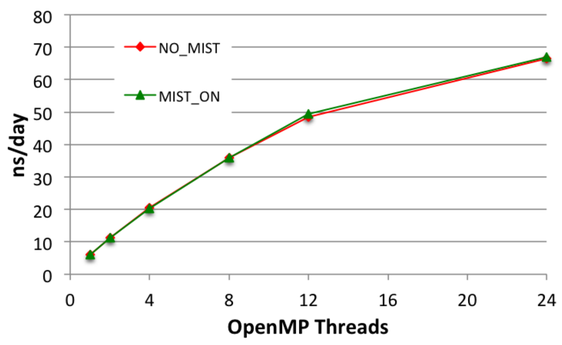

Figure 3: GROMACS performance on ARCHER using OpenMP

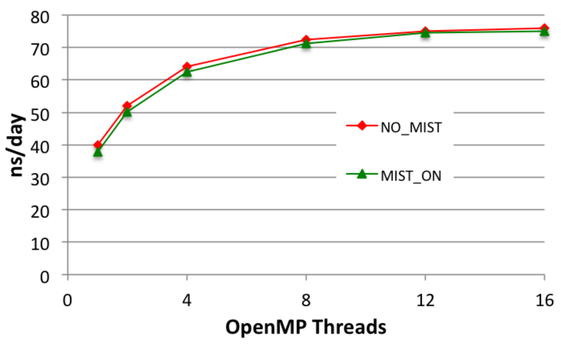

Figure 4: GROMACS performance on Hydra using CUDA + OpenMP

Download MIST

The MIST library source code and documentation is freely available (BSD Licence) from:

www.extasy-project.org/mist

The following features are currently available in the MIST library:

- Amber, GROMACS and NAMD-Lite plugins (LAMMPS under development)
- GPU support with GROMACS and Amber
- MPI and OpenMP parallelisation with GROMACS
- NVE integrators: Velocity Verlet, Verlet Leapfrog, Yoshida
- NVT integrators: Langevin Dynamics based on a BAOAB splitting
- Rapid Sampling schemes: Continuous Tempering, Temperature Accelerated MD
- Access to individual force-field components
- Constraint Solver (RATTLE and Symmetric Newton's method)

Please download and experiment with the code - we welcome feedback, contributions and suggestions!