KMC Simulation Trace File Format (Proposed Revision)

Charlie Thornton

February 27, 2005

1 The Trace

The simulation trace file[1] logs data for every Monte-Carlo step in the simulation. There are many steps, so this file tends to get very big (gigabytes). Because of the size concerns, we want to minimize the amount of data we write out every time step. We also want to be able to understand the exact state of the simulation at a given step. Using this file, we need to be able to get at the following data:

- The change in time represented by each event[2].
- The energy threshold (ΔE) for each event.
- The locations of all the active atoms in the simulation following each event.
- The database process represented by each event.

This information encapsulates the state of the simulation at any given time from a variety of perspectives. The fine-grained nature allows us to perform an analysis at any time granularity and over any time window in the simulation.

In the formats below, I mention *int* and *real* datatypes. For the purposes of my analysis, the *int* type occupies 4 bytes and the *real* type occupies 8 bytes (double precision). There are obvious places to shave these numbers down, but that will keep until we're sure what we want to output.

[1] Currently referred to as the "stack" file.
[2] "Event" and "process" are synonymous.

2 Current Format

The current format of this file is only suitable certain kinds of analysis. It provides Δt, a non-reversable[3] representation of the cental atom's position, and the motion the atoms underwent during the current process. Notably, it can't tell us what database processes have occured or the positions of the atoms in the simulation. There may be a way to get ΔE from this data, but I am unconvinced.

2.1 Header

This record is written out once at the beginning of the file:

fn

Where:
fn (string) Name of the initial mesh configuration (abk) file.

2.2 Body

$20 + 8n$ bytes
This record is written out for each Monte-Carlo step:

Δt	k_{atom}	n	pos_1^{init}	pos_1^{final}	...	pos_n^{init}	pos_n^{final}

Where:

$\mathbf{\Delta t}$	(real)	Elapsed time during this Monte-Carlo step
$\mathbf{k_{atom}}$	(real)	Central atom's position in k-space
$\mathbf{n}$	(int)	Number of moving atoms
$\mathbf{atom_i}$	(int)	Initial position of the i^{th} moving atom
$\mathbf{pos_i}$	(int)	New position of the i^{th} moving atom.

As I mentioned above, I don't know how to go from the k-space representation of the central atom to an atom in R^2. From my perspective, this is crippling.

[3] I would happily be wrong. Does anyone have an equation?

3 Proposed Format

I'm proposing this revision of the event trace file. I believe it would give us the ability to track all of the interesting information in the simulation at any given MC step. It's also 4 bytes smaller per step[4].

3.1 Header

This record is written out once at the beginning of the file:

version

Where:

version (int) File format version number

The only element we would need in the header for the moment is a version number. That way if we needed to make changes we'll still be able to read all of the old files. With a few extra pieces of information in the header, we could actually make all of the results (except the data in the process database) for the simulation self-contained in this file.

3.2 Body

$16 + 8n$ bytes
This record is written out for each Monte-Carlo step:

Δt	*pid*	*n*	$atom_1$	pos_1	...	$atom_n$	pos_n

Where:

Δt	(real)	Elapsed time
pid	(int)	Database process ID
n	(int)	Number of moving atoms
$\mathbf{atom_i}$	(int)	Index of the i^{th} moving atom
$\mathbf{pos_i}$	(int)	New position of the i^{th} moving atom. Assuming that the i^{th} atom is in "center" position, pos_i tells us where in the configuration representation to translate it to. This allows us to represent the new location as a single int rather than two reals.

It turns out, because of how symetries are used by the process database, knowing the process ID is not enough information to know where atoms moved to. It only tells us what kind of move occured. However, it is sufficient to give us ΔE. We must still dump the atom movements in the smallest possible way. I believe this can be done by taking advantage of the known translation vectors. The only downside is, the simulation code has to be smart enough to convert between a movement between two non-central positions as though it is a movement from the central position.

[4] A savings of about 12.5% assuming we have an average of 2 moving atoms