GMIN moves revamp

Working document

Contents

Introduction

This is the working document for discussion of revamping GMIN's step taking routines and implementation. It is very much a work in progress and focusses on two aspects - the development of a new MOVES module (*GMIN/source/moves.f90*) and an associated mechanism for calling these moves in a flexible way.

The group wiki pages for the MOVES and SANITY modules can be found here:

MOVES: http://goo.gl/uOvRsZ

SANITY: http://goo.gl/Hy0hVj

Chris' initial presentation from AVT on Tuesday 15th of April discussing the need for these changes can be found here: http://goo.gl/3Vlgzc

The problems

- Moves are often potential specific - more than they need to be
- The code is very fragmented - we have moves everywhere
- As a result - the logical flow of loops in *mc.F* is tortuous to decipher
- It is hard or impossible to produce move sequences or move blocks
- There is a lot of repeated code - how many times have we coded cartesian moves?!

How can we address this?

We need a major rethink as to how GMIN changes the coordinates before quenching - while retaining backwards compatibility as much as possible. At the moment, the plan is to:

1. Move all step taking routines to a MOVES module (moves.90)

We need to decide if this is to also include ‘driver’ routines - for example group rotation steps use two subroutines *GROUPROTSTEP* (the driver routine, selects which groups to be rotated during each step and the rotation angle) and *GROUPROTATION* (performs the actual rotation given the atoms in the group).

2. Develop a new flexible framework for steptaking

We need a new way to specify which moves we’d like to apply to the system when during a GMIN run. We currently rely on potential specific implementations (i.e. Birgit’s CHARMM moves using the *CHMOVE* keyword...which is undocumented) or people coding moves in a sensible way so that some can work together (e.g. in AMBER, we do MD steps first, then dihedral/group rotations and finally cartesian displacements). There are many cases where more flexibility would be hugely valuable - and at the moment people just hard code in special cases which makes the problem worse!

Any new framework needs to include the ability to:

- Perform a (repeated?) sequence of steps
- Construct ‘groups’ or ‘blocks’ of steps to be taken at the same time
- Apply steps to a subset of atoms/particles only
- Support MPI (*BHPT*) runs - allowing different steps for each replica
- Allow periodic/frequency based steps
- Strictly define the order in which steps are taken
- Test steps!
- Prevent inappropriate steps from being taken

Please add anything you think is missing to the above wish list!

Progress (17/04/2014)

This project is currently in the early planning and implementation stage:

- *MOVES* exists (*moves.f90*), but currently only contains three step taking routines:
    - *CARTESIAN_SPHERE*
    - *CARTESIAN_SIMPLE*
    - *ROTATION_ABOUT_AXIS* *(tested and it replicates group rotation! :D)*
- The NEWMOVES keyword is currently required in *data* to bypass all old step taking routines in mc.F
- Steps need to be manually commented/uncommented to use them in mc.F - search for *NEWMOVEST*.

Coding guidelines

As discussed on the wiki page, the moves in the new module need to conform to the following guidelines to keep us on the straight and narrow:

- code blocks must be indented in a manner consistent with the rest of the module (3 spaces)
- variable names must be understandable e.g. MAX_STEP rather than S
- avoid using GOTOs!
- all subroutines must be fully commented (in English)
- all required and optional arguments must be fully explained
- every move routine must have an optional final argument ATOM_LIST to apply the move to a subset of atoms. See the example on the wiki page
- only utility modules (e.g. VEC3) may be USE'd from MOVES. This does **NOT** include COMMONS
- no potential specific information may be required by any routine in MOVES - this should be dealt with in the driver routine
- move routines should **NOT** print anything outside of STOP messages - printing should come from the appropriate driver routines
- where appropriate, sanity checks (GMIN SANITY module) and tests (GMIN TESTS module) should be included in move routines

Additional thoughts:

- conditional moves (e.g. restart)
- sanity check moves (e.g. moving two atoms apart if they underwent cold fusion)
- definable blocks of moves
- think about changing step size based on acceptance ratio and other quantities we can measure
- functional form for step size

Please add anything you think is missing to the above list!

To-do list

Chris

- Add rigid translation moves to the MOVES module
- Make a test driver routine for AMBER - requires some planning

Kyle

- Finalise move file input format (preferably something standard)
- Convert Python parser into Fortran (should be fun!)