Daresbury Laboratory
INFORMATION QUARTERLY for COMPUTER SIMULATION
of CONDENSED PHASES
Number 23 1986

A comment on debugging Monte Carlo programs.

Mihaly Mezei
Department of Chemistry,
Hunter College of the CUNY,
New York, NY 10021, USA.

The production of correct computer code is a major difficulty in any computational project. For a long code, extensive testing is required, special cases where the result is known should be tried. A particular difficulty arises in the debugging of a Monte Carlo program: since the results are legitimately subject to statistical uncertainties, no precise match with existing results can be expected.

This note will describe an approach to the debugging of Monte Carlo programs for simulating an assembly of particles. The key idea is the extensive use of consistency checks as follows. In most problems, there are several expressions or algorithms that compute a given quantity and usually the one that is (thought to be) optimal is chosen. The other ones, however, can still be used for a consistency check. In the rest of this note the self tests employed in a Metropolis Monte Carlo [1] program using pairwise additive energy and single particle moves with force biased displacements [2] will be described (see also Ref. 3).

The fundamental test is on the calculation and updating of the total energy of a system of N molecules, E:

$$E = \sum_{\mathrm{i<j}}^{\mathrm{N}} e_{\mathrm{ij}} \tag{1}$$

where e_{ij} is the pair interaction energy between molecules i and j. E is updated whenever a particle i' is moved. For this update only the terms involving the moved particle i' are to be considered since the rest remain unchanged:

$$E_{\mathrm{new}} = E^{\mathrm{old}} - B_{\mathrm{i'}}^{\mathrm{old}} + B_{\mathrm{i'}}^{\mathrm{new}} \tag{2}$$

where

$$B_{\mathrm{i}} = \sum_{\mathrm{i\neq j}}^{\mathrm{N}} e_{\mathrm{ij}} \tag{3}$$

is the binding energy of molecule i and the superscripts old and new refer to the status before and after the move, respectively. The $B_{\rm i}$ values are also carried and kept updated whenever a move is accepted (for all molecules $i \neq i'$):

$$B_{\rm i}^{\rm new} = B_{\rm i}^{\rm old} - e_{\rm i'j}^{\rm old} + e_{\rm i'j}^{\rm new}. \tag{4}$$

This way (since the $e_{\rm ij}$ values are not stored), Eq.(2) can be used without recomputing the $e_{\rm i'j}$ values before the move and the recomputation is only needed for accepted moves (to update the $B_{\rm i}$s)[4]. The energy E is related to the $B_{\rm i}$s

$$E = \sum_{\rm i=1}^{\rm N} B_{\rm i}/2 \tag{5}$$

Therefore, the first test compares the result of Eq.(4) with the energy carred. The second test looks at the binding energy $B_{\rm i'}$ carried by the program and compares it with the value directly computed from Eq.(3). The third test recomputes E using Eq. (1) and compares it with the energy carried. Notice that the first test requires no energy calculation at all, the second requires the computation of $\sim N$ energies and the third requires the computation of $\sim N^2/2$ energies. All of these test should give equality up to the precision of the computer.

If the program calculates forces (torques) and the virial sum then analogous tests can be performed on them. The virial sum is updated in analogy of the total energy, and thus the third test is applicable. The components of the forces, torques acting on a molecule are obtained and updated in analogy to the binding energy $B_{\rm i}$ and thus the second test can be applied to them.

The tests described above do not check the calculation of $e_{\rm ij}$ and its derivatives but rather the operations involved in composing the various molecular and supermolecular sums. For programs calculating forces, there is a possibility to test simultaneously the calculation of $e_{\rm ij}$ and its derivatives by performing a numerical differentiation and comparing the derivatives computed. This test, however, is not as strong as the ones described above since in general when the difference quotient is a good approximation to the derivative the increment is very small. But a too small increment will give imprecise quotient due to the limited precision of the machine. The test is more powerful when performed on a system consisting of a few molecules only. In our experience, on a 32-bit word-length machine only 2-3 digit agreement can be obtained.

Condensed phase simulations are usually performed under periodic boundary conditions. While most applications choose the rectangular unit cell (that is very simple to implement), there are advantages to use other shapes: truncated octahedron, hexagonal prism or the Wiegner-Seitz cell of the face-centered cubic close packing. The efficient implementation will use an algorithm for the determination of the image cell a given point is in that is specifically developed for the boundary condition chosen. An inefficient, but general algorithm would simply find the cell whose center is the nearest to the point considered. This general algorithm, however, can be used to check any of the specific ones.

Quite often, a Monte Carlo program maintains a list of neighbours (to reduce energy calculation) or the number of neighbours (for calculation of coordination numbers). Again, this data is computed from scratch at the beginning only and is updated thereafter. In the self-testing mode, however, they can be recomputed from scratch and compared with the lists carried. For the neighbour lists the symmetry should also be checked: if i is a neighbour of j than j must be a neighbour of i and vice versa.

In the actual realization a subroutine is maintained in the program for these tests. This subroutine is called only optionally, primarily during testing period but periodically thereafter as well (since some bugs may manifest themselves only after longer runs or under some special input parameter combinations). Any time the program is modified, the self tests are called.

In concluding, I would like to apologize to the reader for the obviousness of the material described. The reason for the presentation is not so much to point out the existence of these identities or alternative algorithms but rather to report that they proved to be extremely useful in detecting and locating bugs during the development of our Monte Carlo programs. In several instances, prior to performing the consistency checks, the program apparently "worked" and there was no obvious indication of malfunctioning. When a failure occured, the combination of failed tests usually gave a strong indication as to the location of the error in the program and quite often the point of break- down in the calculation singled out the special circumstances under which the error occured, further narrowing the search. It is suggested therefore that these tests (and possibly any other that is applicable to the problem at hand) should be routinely incorporated into Monte Carlo programs as a debugging aid for the developper and as a confidence builder for the user. As a final warning, however, one should add that all consistency checks serve only as necessary conditions for the correctness of the program and other means of testing should be employed as well.

Acknowledgements.

This research was supported by NIH grant GM 24914 and NSF grant CHE-8293501. Several useful discussions with Prof. K. Bencsath are gratefully acknowledged.

References:

1. N.A. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller, 1953, J. Chem. Phys., **21**, 1087.

2. M. Rao, C.S. Pangali and B.J. Berne, Mol. Phys., **37**, 1779 (1079).

3. D.L. Beveridge, M. Mezei, P.K. Mehrotra, F.T. Marchese, G. Ravi-Shanker, T. Vasu and S. Swaminathan, in "Molecular-Based Study of Fluids", J.M. Haile and G.A. Mansoori, eds., Advances in Chemistry, Vol 204, American Chemical Society, Washington (1983).

4. P.K. Mehrotra, unpublished result.