JOURNAL OF COMPUTATIONAL PHYSICS 59, 324–330 (1985)

Optimizing a Lattice QCD Simulation Program

PH. DE FORCRAND, D. LELLOUCH,* AND C. ROIESNEL†

Centre de Physique Théorique de l'Ecole Polytechnique,‡
Plateau de Palaiseau, 91128 Palaiseau, Cedex, France

Received April 10, 1984; revised August 28, 1984

Three computer-time saving techniques for lattice QCD programs are presented. They concern the link updating order, the compacting of the data and the pseudo-heatbath algorithm.

LATTICE QCD PROGRAM OPTIMIZATION

With the advent and the by now widespread use of numerical simulations of Quantum Chromo Dynamics (QCD) on a lattice, the share of computer resources used by High Energy theoretical physicists has been steadily increasing these last few years. Their skill in using these resources may not have followed at the same space. In this paper we want to describe only those specific features of the program we use to generate field configurations on a lattice which could improve the efficiency of any similar program. The general Monte Carlo method of lattice QCD simulations has been described elsewhere [1–3], and the reader is referred to these papers for more details on the algorithm. Since lattice QCD simulations these days use typically 100 h of CDC 7600 equivalent CPU time or more, we think it worthwhile to describe three computer time saving techniques which have not yet been used to our knowledge. Optimized in this fashion, our program, which runs on a CRAY-1S computer, takes approximately 85 μs/link update (75 μs with the standard 10 hits Metropolis algorithm), with provision for accommodating a 12^4 lattice in the 850K words available memory.

Section I describes the updating order of the links, Section II the storage of the data, and Section III the optimization of the updating algorithm proper, after Ref. [4].

1. LINK UPDATING ORDER

We choose to update sequentially all 8 links radiating from a lattice site, in the arbitrary order 1 to 4 for links in the positive 1–4 directions, then 5 to 8 for links in

* Laboratoire de Physique Nucléaire et des Hautes Energies, Ecole Polytechnique, Palaiseau, France.
† CERN, Theory Division, Geneva, Switzerland.
‡ Groupe de Recherche du CNRS No. 48.

0021-9991/85 $3.00

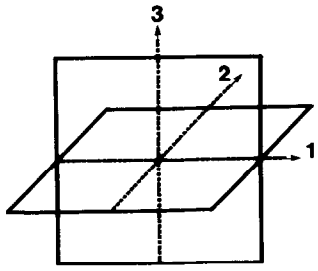

FIG. 1. "Corner" links (—) and "axial" links (···) in three dimensions.

the negative directions. (See Fig. 1, in 3 dimensions.) After all 8 such links have been updated, another lattice site is explored, and only half of the sites need be so considered to complete a sweep of the whole lattice.

This updating order is different from the one commonly used, where links in the positive directions only are changed before one proceeds to the next site. We believe it presents several advantages:

(i) one can make maximal use of intermediate results common to the neighbourhood of several links. With the Wilson action $S = \sum_{\square}(1 - \frac{1}{3}\,\mathrm{Re\,Tr}\,\square)$, the updating procedure requires the knowledge of the sum of 6 matrices associated with the "incomplete plaquettes," which would be completed by the link being updated. Such incomplete plaquettes are made up (see Fig. 1) of one "corner" times an "axial" link. The same corner can be reused later on for the updating of this axial link, so that the number of matrix products necessary to calculate the set of "incomplete plaquettes" for each link is reduced from

$$6 \times 2 = 12 \quad \text{with the conventional method, to}$$

$$6 \times (1 + \tfrac{1}{2}) = 9.$$

This essentially represents a 25% savings in computer time for this part of the program, since matrix multiplication is by far the most time consuming task in the algorithm.

Of course the savings is greater for actions involving larger planar loops too: it amounts to a factor ~3 for the Symanzik tree-level-improved action (TI action) [5] involving 2×1 plaquettes, and a factor >4 for an action proposed by Mütter and Schilling [6] including 8-shaped loops as well.

(ii) Given the precalculated set of "corners," one can include measurements of small planar and nonplanar loops at very little cost in CPU time in the Monte Carlo procedure for the "axial" links. With the TI action we could thus measure all planar loops up to size 4×2 at each sweep for an increase ~5% in CPU time. Of course this possibility should be used with caution if one wishes to extract loop–loop correlations because a given, "frozen" configuration is usually required in that case. But we think it is a nice opportunity to accumulate, without using any tem-

porary storage device, high statistics on small loop averages, for which more sophisticated methods [7] only give moderate improvements.

(iii) Our choice of link updating order lends itself well to vectorization. All sites on a hypercubic sublattice of spacing $2a$ (or face-centered sublattice of spacing $4a$ for the TI action) can be treated independently, yielding a uniform density of heat-exchange centers throughout the lattice. Then this hypercubic sublattice is moved around into another of 8 different positions until one sweep through the lattice is completed. Remarking that links to be paired into a corner always belong to the same sublattice, it would be possible to store the links as 8 distinct sublattices and thus minimize the time spent fetching links in the main lattice. This has not been implemented, in order to keep the data structure and analysis simple.

(iv) This updating scheme seems to us well suited for future developments of lattice simulations. We tried to perform multiple sequences of updates of the 8 axial links, and simultaneous updates of these 8 links as well, in both cases with the hope of decreasing sweep-to-sweep correlation, and with somewhat disappointing results. But it is also quite easy to implement gauge fixing, by multiplying all 8 links radiating from a site by some arbitrary $SU(3)$ matrix. And our way of grouping links might be quite appropriate for blocking procedures.

2. Data Compaction

The challenge to data compaction was to store $4 \times 12^4 = 82944$ $SU(3)$ matrices into a 1M word 64-bit computer memory, practically reduced to ~850K words for code and data. The "natural" way to represent a 3×3 matrix with complex elements was obviously unsatisfactory.

We reduced the amount of required memory to one third of this, i.e., 6 words per matrix. One should note that, in theory, an $SU(3)$ matrix can be represented by $(N^2 - 1) = 8$ real numbers, but this leads to very inconvenient computations.

The way we did it is a compromise between size, computing speed and accuracy. We only store the first 2 rows of each matrix; the third one is then computed as the cofactors of the previous rows; an extra factor of 2 in storage is obtained using a half precision 32-bit representation. Since the Cray machine instructions do not provide hardware half-precision operations, we had the choice of the representation. We used a fixed point representation, where each real matrix element x is represented by

$$I(x) = \mathrm{INT}[(x+1)(2^{31}-1) + \tfrac{1}{2}].$$

This is possible because each element of any $SU(N)$ matrix is less than 1 in modulus. The accuracy of such a method is $\sim 4.6 \times 10^{-10}$, better than a standard 32-bit floating point half-precision representation, and worse than the full precision representation by a factor $3 \times 10^{-5}/4 \times 10^{-10} \sim 10^5$. To keep rounding errors under

control, we reunitarize the link matrices every fifth iteration in general, so that the unitarity defect, defined as

$$CABS[U(1,1)\cdot U^*(1,2)+U(2,1)\cdot U^*(2,2)+U(3,1)\cdot U^*(3,2)]$$
$$+ABS[U(1,1)\cdot U^*(1,1)+U(2,1)\cdot U^*(2,1)+U(3,1)\cdot U^*(3,1)]$$

stays at the 10^{-8} level. The unitarization routine then takes 2.5 μsec/link.

The routines to pack and unpack matrices, written in CRAY Assembly Language (CAL), are extremely fast: their asymptotic speeds are 2 Clock-Periods = 25 ns/word to pack (or unpack). We reach such a speed by "chaining" together up to 6 functional units, yielding a top instruction-rate of $6\times 80 = 480$ Mops (mega-operations per second)!

3. Pseudo-heatbath Algorithm

We first tried a simple Metropolis algorithm [8], where one obtains a new link candidate by multiplying the old link by some random $SU(3)$ matrix. Our random sample had a centered quasi-Gaussian distribution in the Lie algebra, with a spread governed by a simple feedback mechanism which maintained the acceptance rate around 45%. The resulting auto-correlation on the plaquette average, defined as

$$C(n) \equiv$$
$$\frac{\langle \square(\text{sweep } n+i)\,\square(\text{sweep } i)\rangle - \langle \square(\text{sweep } n+i)\rangle\langle \square(\text{sweep } i)\rangle}{\sqrt{[\langle \square(\text{sweep } n+i)^2\rangle - \langle \square(\text{sweep } n+i)\rangle^2]\cdot[\langle \square(\text{sweep } i)^2\rangle - \langle \square(\text{sweep } i)\rangle^2]}}$$

is shown on Fig. 2, for the TI action on a 12^4 lattice at $\beta = 3.6$ (analogous to $\beta \sim 5.3$ for the Wilson action). On the same plot appear results obtained with the pseudo-heatbath algorithm of Ref. [4], using 2, 3, and 4 successive $SU(2)$ heatbaths for each $SU(3)$ matrix. The improvement brought by the latter algorithm is less drastic than in Ref. [4], where the action and β-value were different. The choice obviously depends on the CPU time needed for each algorithm. We eventually decided for the pseudo-heatbath algorithm with 3 $SU(2)$ subgroups, because we could reduce its execution time even on our vector-machine, and because, due to our extended TI action, $\sim\frac{2}{3}$ of the CPU time was spent computing the "incomplete plaquettes" anyway.

One practical difficulty to implement the pseudo-heatbath algorithm on a vector machine consists in generating efficiently a random variable with distribution $e^{k_i x}\sqrt{1-x^2}$ between -1 and $+1$, where k_i varies with the link i being updated. We could not generate such a distribution directly, so we implemented a filter which would reject some of the input values to ensure the correct output distribution after enough passes. Unfortunately (see Ref. [9]) the rejection rate of the filter is not predetermined, so that it is not possible to come out with the exact distribution after some fixed number of attempts through the filter. We discovered systematic

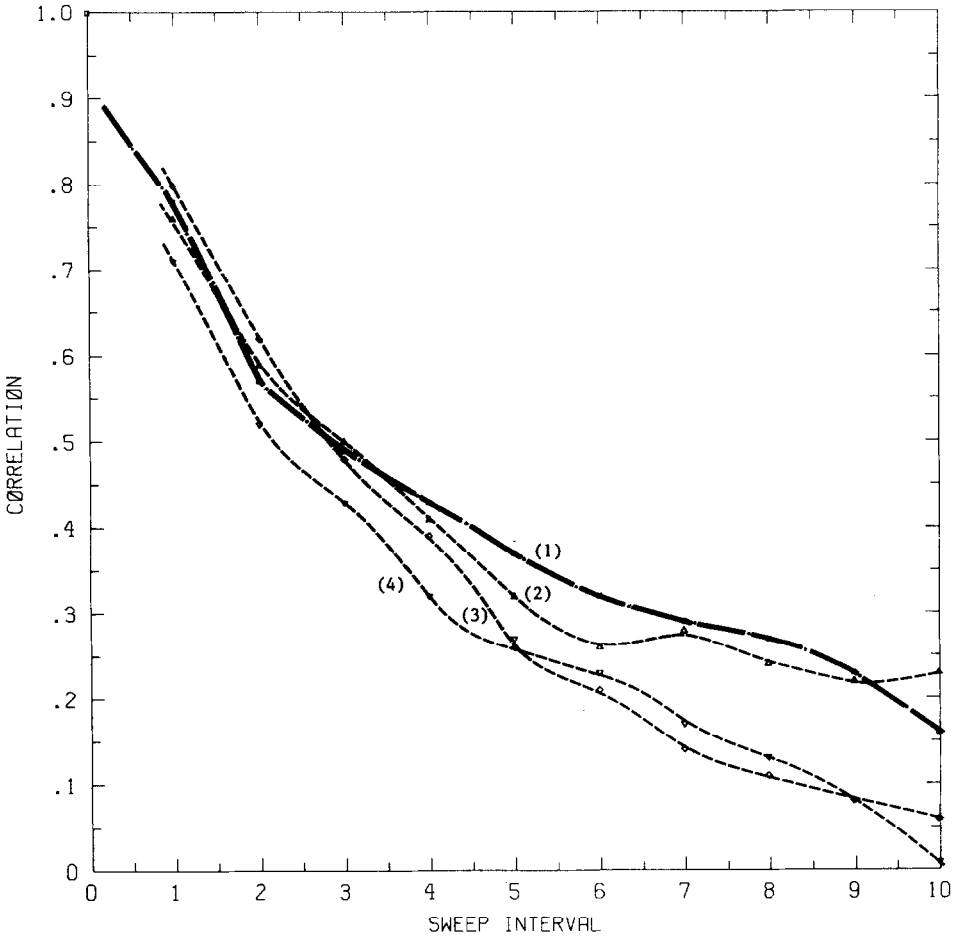

FIG. 2. Auto-correlation on the plaquette average for a 12^4 lattice at $\beta = 3.6$ with TI action. Curves correspond to the Metropolis algorithm with 10 hits/link (– · –) and the pseudo-heatbath algorithm with 2, 3, and 4 $SU(2)$ subgroups (---). For each algorithm, $C(n)$ was measured over a sample of 300 sweeps through the thermalized lattice.

effects upon trying to set to 1 the random variables which had not been generated after a fixed number of passes. So we decided instead to leave the filter loop untouched (and not vectorized) but to prepare, by vectorized operations, the input distribution so that it matches the final one as well as possible. This is feasible through the use of tables [10], which take some memory space and are not practical on our CRAY computer. Instead we found that the average number of attempts through the filter could be reduced from 8 or more (starting from an exponential distribution $e^{k_i x}$ when k_i is large) to less than 1.3 by starting from $e^{\cdot 7 k_i x}$ (see Fig. 3). A still better input distribution, which we chose, is that plotted in Fig. 3 where the turnover point between exponential and uniform distribution can be

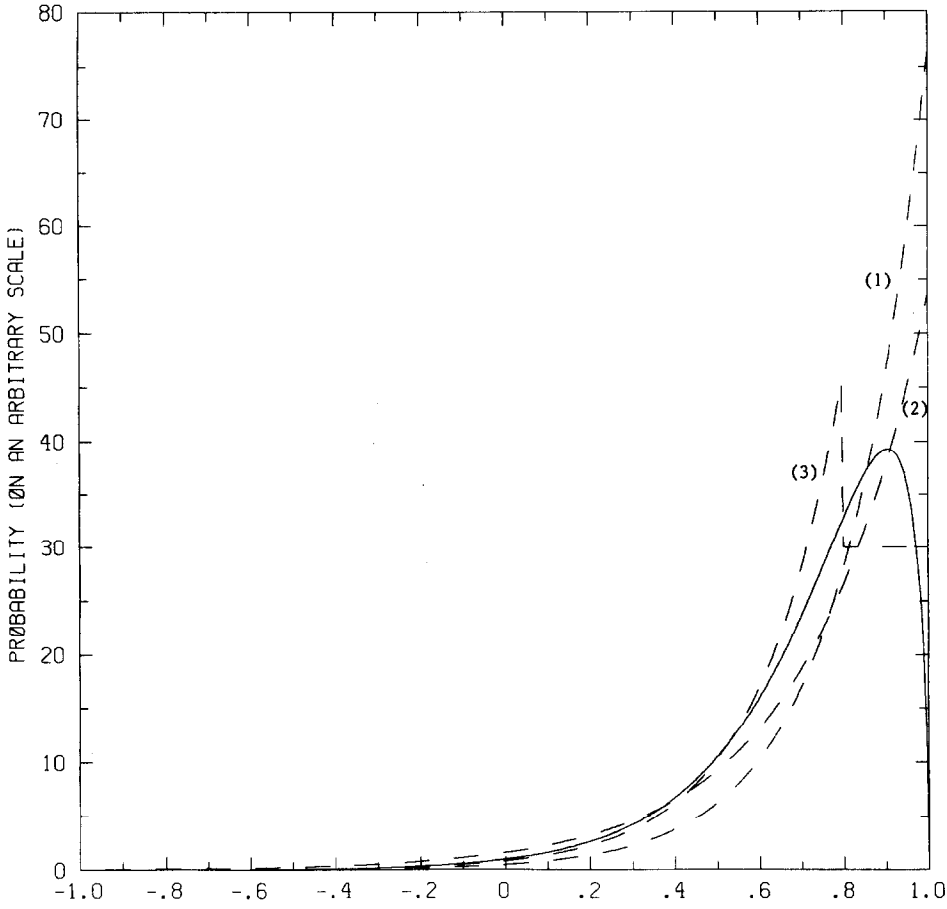

FIG. 3. Typical form of the distribution $e^{kx}\sqrt{1-x^2}$ (—). Various input distributions are e^{kx} (1), e^{-7kx} (2), exponential + uniform (3).

adjusted as a function of k_i. It requires more preparation than the simple exponential distribution, and yet executes faster on our machine because all preparatory operations are performed vectorially (about 10 times as fast as the nonvectorized filter operations).

CONCLUSION

We have been using for our simulations the CRAY-1S computer of the CCVR. For an 8^4 lattice or larger, the times spent in each part of our Monte Carlo program, with the Wilson or the TI action, are given in Table I, in microseconds per link. The main routines in our program have been written in CRAY Assembly

TABLE I

Time[a] Spent in Each Part of Our Monte Carlo Program, for the Wilson or the Tree-Level Improved (TI) Action

	Wilson	TI
Incomplete plaquettes	26	73
Data compaction	7	18
Metropolis 10 hits	42	42
Pseudo-heatbath 3 subgroups	52	52

[a] In μs/link.

Language by one of us (D.L.); standard Fortran code runs 0 (30%) slower. Depending on the size of the lattice and the available memory, data compaction may or may not be necessary. The speed of the rest of the program does not depend on that option, since the arithmetic is always performed with 64-bit accuracy. Some degradation does occur for small lattices, because the vector of independent links which can be updated together is shorter; but the update time per link on a 4^4 lattice still falls below 100 μs. (without data compaction). In any case, our main point is that the three techniques described above can be used with profit on *any* machine.

Note added in proof. Following the recent suggestion of Kennedy *et al.* [11], we have now vectorized the first two attempts at passing through our pseudo-heatbath filter. With the initial probability distribution e^{-7k_jx} [(2) on Fig. 3], our streamlined routine now takes 32 μs/link instead of 52 for 3 $SU(2)$ subgroups.

ACKNOWLEDGMENTS

We thank H. Lipps for discussions, and E. Marinari for giving us his original pseudo-heatbath routine. We gratefully acknowledge the technical support of P. Herchuelz and the CCVR CRAY assistance team.

REFERENCES

1. M. CREUTZ, *Phys. Rev. D* **21** (1980), 2308.
2. D. BARKAI AND K. J. M. MORIARTY, *Comput. Phys. Commun.* **25** (1982), 57; **26** (1982), 477; **27** (1982), 105.
3. D. BARKAI, K. J. M. MORIARTY, AND C. REBBI, BNL preprint BNL-33953.
4. N. CABIBBO AND E. MARINARI, *Phys. Lett. B* **119** (1982), 387.
5. PH. DE FORCRAND AND C. ROIESNEL, *Phys. Lett. B* **137** (1984), 213; CERN preprint TH-3858.
6. K. H. MÜTTER AND K. SCHILLING, Wuppertal preprint WUB 83-24.
7. G. PARISI, R. PETRONZIO, AND F. RAPUANO, *Phys. Lett. B* **126** (1983), 250.
8. N. METROPOLIS, A. W. ROSENBLUTH, M. N. ROSENBLUTH, A. H. TELLER, AND E. TELLER, *J. Chem. Phys.* **21** (1953), 1087.
9. K. C. BOWLER AND B. J. PENDLETON, *Nucl. Phys. B* **230** (FS 10) (1984), 109.
10. F. GUTBROD, communication at the Lattice Coordinating Meeting, CERN, Dec. 1983.
11. A. D. KENNEDY, J. KUTI, S. MEYER, AND B. J. PENDLETON, Santa Barbara preprint NSF-ITP-84-62.