PHYSICAL REVIEW D VOLUME 40, NUMBER 6 15 SEPTEMBER 1989

Rapid Communications

The Rapid Communications section is intended for the accelerated publication of important new results. Since manuscripts submitted to this section are given priority treatment both in the editorial office and in production, authors should explain in their submittal letter why the work justifies this special handling. A Rapid Communication in **Physical Review D** *should be no longer than five printed pages and must be accompanied by an abstract. Page proofs are sent to authors, but because of the accelerated schedule, publication is not delayed for receipt of corrections unless requested by the author or noted by the editor.*

Block-conjugate-gradient method

J. F. McCarthy
Department of Physics, Indiana University, Bloomington, Indiana 47405
(Received 20 April 1989)

It is shown that by using the block-conjugate-gradient method several, say s, columns of the inverse Kogut-Susskind fermion matrix can be found simultaneously, in less time than it would take to run the standard conjugate-gradient algorithm s times. The method improves in efficiency relative to the standard conjugate-gradient algorithm as the fermion mass is decreased and as the value of the coupling is pushed to its limit before the finite-size effects become important. Thus it is potentially useful for measuring propagators in large lattice-gauge-theory calculations of the particle spectrum.

In a recent paper,[1] we reported an ultimately unsuccessful attempt to improve the computational efficiency of the inversion of the Kogut-Susskind (KS) fermion matrix in lattice-gauge-theory computations by the method of incomplete Cholesky decomposition. In this paper, we are able to report a type of preconditioning for the KS matrix that can save a factor of more than 2 computationally in interesting cases. We hasten to add, however, that it is only useful in the restricted case in which several systems are required to be solved simultaneously, as in finding several columns of the inverse KS matrix at one time in propagator calculations. Hence, it would be of no use in generating gauge configurations in algorithms which include dynamical fermions. Nevertheless, it could conceivably save some computing time in the measurement part of spectral calculations, especially as these simulations are pushed to ever-increasing lattice sizes and decreasing quark masses.

The method that we use is the block-conjugate-gradient (BCG) method.[2,3] This is a simple extension of the commonly used conjugate-gradient algorithm to block form, in which several systems are solved simultaneously. If we let A be the $n\times n$ matrix to be inverted, B be the $n\times s$ array of source vectors, and X be the $n\times s$ array of solution vectors, then the set of equations $AX=B$ can be solved iteratively by the BCG algorithm as follows.[2]

Given the initial vector $X^{(0)}$, let $R^{(0)}=B-AX^{(0)}$, define $P^{(0)}=R^{(0)}\gamma_0$, and for $k=0,1,\ldots,$ update the iterates, residuals, and directions:

$$X^{(k+1)}=X^{(k)}+P^{(k)}\alpha_k\,,$$
$$R^{(k+1)}=R^{(k)}-AP^{(k)}\alpha_k\,,$$
$$P^{(k+1)}=(R^{(k+1)}+P^{(k)}\beta_k)\gamma_{k+1}\,,$$
$$\alpha_k=(P^{(k)T}AP^{(k)})^{-1}\gamma_k^T R^{(k)T}R^{(k)}\,,$$
$$\beta_k=\gamma_k^{-1}(R^{(k)T}R^{(k)})^{-1}R^{(k+1)T}R^{(k+1)}\,.$$

Here, $P^{(j)}$ is an $n\times s$ array of A-conjugate direction vectors; $R^{(j)}$ is an $n\times s$ array of residual vectors; and α_j, β_j, and γ_j are $s\times s$ matrices.

The algorithm terminates when the residuals $R^{(k)}$ become acceptably small. This will happen provided that the algorithm does not lose stability. Loss of stability can occur if the matrices $P^{(k)}$ and $R^{(k)}$ lose full rank. The nonsingular matrices γ_k are parameter matrices used to monitor the stability of the algorithm. The iterates $R^{(k)}$ and $X^{(k)}$ are invariant with respect to the choice of γ_k. In practice, we found that problems with stability did not arise in our implementation of the BCG algorithm so that we could set $\gamma_k=I$ in all cases.

What advantages do we gain by using the block form of the conjugate-gradient algorithm? In Ref. 2 it is shown that after k steps of the scaled block-conjugate-gradient algorithm, the error in component m, $m=1,2,\ldots,s$, is bounded by

$$e_m^{(k)T}Ae_m^{(k)}\le\left(\frac{1-\sqrt{\kappa^{-1}}}{1+\sqrt{\kappa^{-1}}}\right)^{2k}c_1\,,\qquad(1)$$

where $\kappa=\lambda_n/\lambda_s$, λ_i is the ith eigenvalue of A in increasing order, and c_1 is a complicated constant depending on m but independent of k. [There are, in addition, some technical conditions stated in the theorem and required for the proof which we do not wish to go into here and we urge any reader interested in the details to consult Ref. 2. In particular, the constant c_1 contains terms with the factor $1/(\lambda_n-\lambda_s)$ so that the proof does not hold for the case $s=n$.] The block method cannot require more iterations than the standard CG method in which all components of the error obey a similar bound to (1) with $\kappa=\lambda_n/\lambda_1$ (i.e., the "condition number"). The extent of its advantage depends sensitively on the eigenvalue distribution of A. The block method is particularly useful when A has several small eigenvalues widely separated from the others. In

such cases, it can give a significant reduction in the total number of iterations required for convergence. In our work, we found that solving $s=12$ systems at once (e.g., finding 12 columns of the inverse KS matrix) using the BCG algorithm required as many as four times fewer iterations as applying the standard CG algorithm 12 times. The actual factor involved depends on the quark mass (it improves as the mass is decreased), on the coupling (it is best to choose a value of β just below the onset of the finite-size phase transition), and on the block size s (it improves as s increases and $s=12$ is the largest value that we considered). Another advantage of the BCG algorithm is that it forms the product of the matrix A with several vectors at once so that it requires a smaller number of accesses to A. This could increase efficiency on some vector computers. The BCG algorithm is easily vectorized, in the same way as the standard CG algorithm, so it has none of the problems of the incomplete Cholesky decomposition in this respect.

Analogously and, indeed, historically antecedent to the BCG algorithm, a block Lanczos algorithm was developed by Cullum and Donath[4] and Golub and Underwood[5] for the solution of linear eigenvalue problems. The use of the Lanczos algorithm in lattice gauge theory has been advocated by Barbour.[6,7] In his papers, certain claims are made concerning the preferability of using the Lanczos algorithm over the conjugate-gradient algorithm, specifically that convergence is much better at quark masses less than 0.05. Also, it is claimed that at small quark masses the block inversion is about $s/2$ times faster than the equivalent single-row inversion for Susskind fermions (where s, as before, is the number of vectors in a block). These claims led us to investigate using the Lanczos algorithm, both in its standard and block forms. It is well known that the Lanczos and CG algorithms are mathematically equivalent in that the residual and the solution vector calculated at each iteration are equal. However, there is some question as to whether the two are computationally equivalent—a matter that is determined by the way in which rounding errors arise in the separate algorithms. We found the standard Lanczos alogrithm and CG algorithm to be computationally equivalent for masses as low as 0.005, which is as low as we tried. For the block algorithms, we found the block-conjugate-gradient algorithm to be far preferable to Lanczos. This is because it is simpler to program, it requires fewer operations per iteration, and, most importantly, it gives rise to no stability problems whereas the block Lanczos algorithm develops stability problems for low masses and large s. Hence, all our further computations were done using the BCG algorithm. This has the added advantage of being a straightforward generalization of the most commonly used standard algorithm.

We can see two situations in which the Lanczos algorithm is very useful: (i) in computing eigenvalues; (ii) when the matrix to be inverted is not positive definite. The second situation does not concern us here. The first was thoroughly investigated by Barbour in computing the low-eigenvalue spectrum of the KS matrix[6,7] We are indebted to these papers for the insight that they have given us into the nature of the spectrum of the KS matrix,

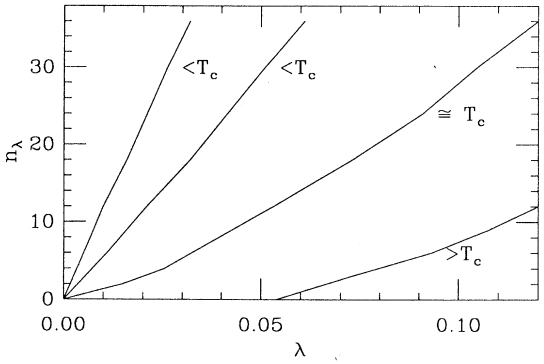

FIG. 1. Schematic diagram showing how the spectrum of low eigenvalues of the KS matrix evolves as the coupling crosses the finite-temperature phase transition.

which is important for understanding the way in which the CG algorithm converges as the coupling β is changed and also for understanding the onset of the confinement-deconfinement phase transition in numerical simulations of finite-temperature QCD. We shall include a brief summary of their findings.

Let M be the KS matrix. The eigenvalues of M are of the form $2m \pm i\lambda_i$ for $i=1,\ldots,n$, where $n\times n$ is the size of the matrix. The λ_i are the eigenvalues of the anti-Hermitian, zero-mass matrix. Barbour calculated the low-eigenvalue distribution of the λ_i for various values of β in connection with studies of finite-temperature QCD. From these it would appear that the low-eigenvalue spectrum evolves in the manner shown in Fig. 1 as β is increased. The density of eigenvalues near zero decreases as one approaches the finite-temperature phase transition and vanishes as the transition point is crossed via a rapid increase in the magnitude of the lowest eigenvalue for a small increase in β. This ties in with calculations of the chiral order parameter $\langle\bar{\psi}\psi(m)\rangle$ of finite-temperature QCD. In the spectral representation we have (see Ref. 6)

$$\langle\bar{\psi}\psi(m)\rangle = \frac{3}{N}\langle \mathrm{Tr}(M^{-1})\rangle = \frac{3}{N}\left\langle \sum_{i=1}^{N} \frac{2m}{\lambda_i^2+(2m)^2}\right\rangle \xrightarrow{V\to\infty} 3\left\langle \int_{-\infty}^{\infty} d\lambda \frac{2m\rho(\lambda)}{\lambda^2+(2m)^2}\right\rangle,$$

where $\rho(\lambda)$ is the normalized spectral density and so

$$\lim_{m\to 0}\lim_{V\to\infty}\langle\bar{\psi}\psi(m)\rangle = 3\pi\rho(0).$$

The value of $\langle\bar{\psi}\psi(0)\rangle$ is determined by the density of eigenvalues near zero. It is finite below the transition point and zero above it. If a small mass is introduced, then $\langle\bar{\psi}\psi(m)\rangle$ never vanishes but it decreases sharply at the transition point with a gap whose size is determined by the gap between the lowest eigenvalue in the high- and low-temperature phases.

What effect does the nature of the eigenvalue spectrum have on the rate of convergence of the CG algorithm at different β? The algorithm converges faster as one approaches the transition point from below, initially because the eigenvalues are becoming more clustered together due to the decrease in density around zero of the λ_i, and then because the lowest eigenvalue rapidly increases, reducing

the condition number. For the block-conjugate-gradient algorithm we can expect that there will be some region of β, just before the onset of the phase transition, where it will be at its most effective. This will be a situation in which there will be a tail of low eigenvalues adjoining the main cluster. For values of β above the phase transition region we would not expect the BCG method to be of any use at all and far below the phase transition region its usefulness would decrease. We shall see that, in fact, our work confirms these expectations nicely.

We used lattices of size $8^3\times12$ with four flavors of dynamical fermions. They were generated using the hybrid-molecular-dynamics algorithm.[8] Periodic boundary conditions were used for the gauge fields in all directions while, for the fermions, we used periodic boundary conditions in space and antiperiodic boundary conditions in imaginary time. A dynamical fermion mass of 0.1 was always used. These specifications are the same as those in our previous paper[1] and so we could make use of some of our previous results—namely, that the phase transition occurs around $\beta=5.5\pm0.1$.

For every iteration we computed the residual $R^{(k)}=B-AX^{(k)}$ and our condition for convergence was

$$\epsilon_k\equiv\sqrt{R^{(k)\dagger}R^{(k)}}<0.00005\,.$$

Of course, in the block algorithm we are dealing with an *array* of solution vectors. When the residual corresponding to one of the solution vectors reached the convergence criterion, we deleted that column and continued the algorithm with $s-1$ vectors; and so on until all the columns had converged.

The array of source vectors B in our calculations consisted of unit vectors chosen from the $n\times n$ identity matrix. In fact, we usually just chose the first s columns of the identity matrix, but the selection could have been made at random and given similar results. When we present our results we compare the convergence of one particular column in the course of computations using the BCG algorithm with different block sizes. To have been complete we would have had to compare the results of, say, one run of the $s=12$ BCG algorithm with 12 separate runs of the standard CG algorithm. We did do a complete analysis in a couple of cases and convinced ourselves that all columns are treated in the same way by the block algorithm and that to present the results of one column is representative.

Figure 2 shows graphs of the residual versus number of iterations for four different masses at $\beta=5.375$. It illustrates two general trends which hold for all values of β—that the number of iterations required for convergence decreases as the block size s is increased and that this effect is enhanced at smaller masses.

That the effect should be enhanced at smaller masses is quite easy to explain since, as mentioned before, the eigenvalues of the KS matrix are of the form $2m+\lambda_i$ and as m is increased it dominates the ratio $\kappa=(2m+\lambda_n)/(2m+\lambda_s)$ which controls the rate of convergence.

The trend with increasing s, however, is harder to explain satisfactorily and is very β dependent as can be seen from Table I. Table I shows the number of iterations required for convergence by the BCG algorithm as a function of β, s, and m. From our previous work[1] we believe

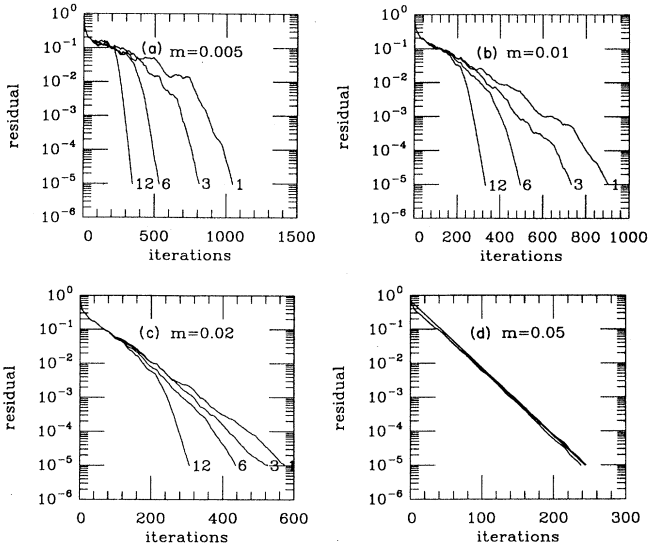

FIG. 2. Residual vs number of iterations for BCG with $s=1$, 3,6,12 at $\beta=5.375$ and four different masses.

the finite-size phase transition occurs at $\beta=5.5\pm0.1$. Notice that for $\beta=5.725$, well above the phase transition, very little is gained by increasing the block size s, even at the smallest mass considered (i.e., $m=0.005$). However, at $\beta=5.300$, the optimum value from Table I, a factor of close to 4 can be gained at $s=12$ and $m=0.005$. The value of $\beta=5.300$ is just below the region where the effects of the phase transition begin to be felt. This is best illustrated by looking at how the ratio of the number of iterations required by the $s=1$ algorithm at $m=0.005$ and $m=0.01$ varies as β is increased. At $\beta=5.300$ the ratio is $1962/1198\approx1.64$ but at 5.325 it is $1553/1194\approx1.30$ and by 5.375 it has decreased to $1047/903\approx1.16$. This ratio is supposed to vary as the ratio of the lowest eigenvalues [i.e., $(\lambda_{\min}+2m_1)/(\lambda_{\min}+2m_2)$] which should be approximately equal to the ratio of masses (i.e., $0.01/0.005=2$) provided that m dominates $\lambda_{\min}$, but we conjecture that as the phase transition is approached $\lambda_{\min}$ increases rapidly for small increases in β, giving rise to deviations at masses whose size is of the same order as $\lambda_{\min}$. We see this ultimately as being a finite-size effect.

For $5.200<\beta<5.300$ the BCG algorithm still gives great savings for small mass and large s, although the benefits decrease as β decreases.

We now must address the problem of the overhead involved in using the BCG algorithm. We will obtain a rough estimate of the overhead by considering the number of floating point operations (FPO's) required. In the BCG algorithm there are $(s+1)s$ $3n$-vector inner products requiring $(24n-2)(s+1)s$ FPO's, s matrix-vector multiplications requiring $2\times582sn$ FPO's, three multiplications of $3n\times s$ and $s\times s$ matrices requiring $3\times3(8s-2)sn$ FPO's, and three $3n$-vector additions requiring $3\times2\times3n$ FPO's. [The factor of 582 in the matrix-vector multiplication is arrived at in the following way. The matrix A in Kogut-Susskind fermion calculations is given by $M^\dagger M$ where M is the KS fermion matrix mentioned before. M has 24 complex off-diagonal terms, i.e., eight

TABLE I. Total number of iterations required for convergence by BCG with different block sizes s, on gauge configurations at various β and m on an $8^3\times 12$ lattice. The values in parentheses are the ratios of the number of iterations required for $s=3,6,12$ to the number required for $s=1$.

β	Mass	$s=1$	$s=3$	$s=6$	$s=12$
5.200	0.005	2473	2398 (1.03)	1484 (1.67)	806 (3.07)
	0.01	1248	1261 (0.99)	1241 (1.01)	722 (1.61)
5.250	0.005	2280	2062 (1.11)	1133 (2.01)	636 (3.58)
	0.01	1171	1248 (0.93)	1048 (1.12)	611 (1.92)
5.275	0.005	2136	1648 (1.30)	962 (2.22)	556 (3.84)
	0.01	1236	1203 (1.03)	906 (1.36)	532 (2.32)
5.300	0.005	1962	1431 (1.37)	851 (2.31)	499 (3.93)
	0.01	1198	1143 (1.04)	794 (1.51)	475 (2.52)
5.325	0.005	1553	1217 (1.28)	728 (2.13)	448 (3.47)
	0.01	1194	1093 (1.09)	675 (1.77)	431 (2.77)
5.375	0.005	1047	854 (1.23)	550 (1.90)	373 (2.81)
	0.01	903	767 (1.18)	518 (1.74)	356 (2.54)
	0.02	575	557 (1.03)	454 (1.27)	325 (1.77)
	0.05	245	252 (0.97)	243 (1.01)	
5.400	0.005	787	663 (1.19)	443 (1.78)	311 (2.53)
	0.01	724	621 (1.17)	426 (1.70)	300 (2.41)
5.475	0.005	545	474 (1.15)	349 (1.56)	268 (2.03)
	0.01	505	444 (1.13)	329 (1.53)	256 (1.97)
	0.02	430	388 (1.11)	298 (1.44)	235 (1.83)
5.725	0.005	157		148 (1.06)	136 (1.15)

directions by three colors, and a scalar diagonal term depending on the fermion mass. So the matrix-vector multiplication $M^\dagger(Mx)$ requires $2\times 3\times(24\times 6+2+24\times 2)$ FPO's.] In the standard CG algorithm there are two $3n$-vector inner products, one matrix-vector multiplication, three scalar-vector multiplications, and three $3n$-vector additions. Hence the ratio of the number of FPO's required by the s-block BCG algorithm to the number required by s implementations of the standard CG algorithm is

$$\frac{2\times 582+24(s+1)+9(8s-2)+18/s}{2\times 582+48+18+18}. \qquad (2)$$

The value of this ratio depends on s. For $s=12$ it is 1.86. This means that in the best case, when $\beta=5.300$, $m=0.005$, and $s=12$, the actual speed-up is a factor of about 2.2 since the ratio of iterations required is about four. We measured the overhead in computations by comparing the CPU time taken by scalar implementations of the BCG and standard CG algorithms and it was well estimated by (2).

In conclusion, we believe that the block-conjugate-gradient algorithm could be useful in future spectral computations in lattice gauge theory. It has the nice feature of improving in efficiency relative to the standard conjugate-gradient algorithm as the mass is decreased and as the value of the coupling is pushed to its limit before the finite-size effects become important.

I would like to thank S. Gottlieb, W. Liu, R. Renken, R. Sugar, and D. Toussaint for the use of their program for generating gauge configurations. This research was supported in part by the U.S. Department of Energy under Contract No. DE-AC02-84ER40125.

[1] J. F. McCarthy, Phys. Rev. D **39**, 3167 (1989).
[2] D. P. O'Leary, Linear Algebra Appl. **29**, 293 (1980).
[3] D. P. O'Leary, Parallel Computing **5**, 127 (1987).
[4] J. Cullum and W. E. Donath, in Proceedings of the IEEE Conference on Decision and Control, Phoenix, Arizona, 1974 (unpublished).
[5] G. H. Golub and R. R. Underwood, in *Mathematical Software III*, edited by J. R. Rice (Academic, New York, 1977).
[6] I. M. Barbour, in *Gauge Theory on a Lattice: 1984*, proceedings of the ANL Workshop, Argonne, Illinois, 1984, edited by C. Zachos, W. Celmaster, E. Kovacs, and D. Sivers (Argonne National Laboratory, Argonne, 1984).
[7] I. M. Barbour, in *Advances in Lattice Gauge Theory*, proceedings of the Conference, Tallahassee, Florida, 1985, edited by D. W. Duke and J. F. Owens (World Scientific, Singapore, 1985).
[8] S. A. Gottlieb, W. Liu, R. L. Renken, R. L. Sugar, and D. Toussaint, Phys. Rev. D **35**, 2531 (1987).