Fourier Acceleration of Relaxation Processes in Disordered Systems

Ghassan George Batrouni[a]
Newman Laboratory of Nuclear Studies, Cornell University, Ithaca, New York 14853

Alex Hansen[b]
Department of Physics, Cornell University, Ithaca, New York 14853

and

Mark Nelkin
School of Applied and Engineering Physics, Cornell University, Ithaca, New York 14853
(Received 14 April 1986)

Critical slowing down poses a major obstacle to reaching the steady-state distribution in large-scale numerical simulations. We demonstrate how to alleviate this problem by means of Fourier acceleration, a method consisting of updating in k space with a k-dependent time step. The method is general and applicable to a wide range of problems. We demonstrate its use by numerical experiments on random resistor networks at the percolation threshold.

PACS numbers: 64.60.Ak, 02.70.+d, 66.30.Dn

Large-scale numerical simulations of physical systems often have to face the major obstacle of "critical slowing down." By this we mean that the number of iterations needed for the system to relax to its steady state grows faster than the volume. For example, to obtain the current distribution on a random resistor network at the percolation threshold,[1] the number of iterations for the Jacobi relaxation method grows like $L^{d_f+d_w}$, where d_f is the fractal dimension of the current-carrying backbone and d_w is the random-walk dimension. For[2] $d=2$, $d_f\approx 1.62$ and $d_w\approx 2.87$, making the growth in computational requirements quite severe. This is a rather general difficulty which has been studied in other contexts.[3,4] The methods of Ref. 3, i.e., Fourier acceleration, can alleviate this difficulty in many problems, ranging from lattice field theory to spin glasses. In this paper we apply the method to the random resistor network at the percolation threshold. This relatively simple example is of current physical interest, and gives a useful benchmark for the method.

The random resistor network at the percolation threshold in $d=2$ is constructed as follows: Imagine a square lattice of size L (times the lattice constant). There are two types of links connecting neighboring nodes: They either have unit resistance with probability p, or infinite resistance (having been cut) with probability $1-p$. At the percolation threshold $p_c=\frac{1}{2}$, clusters of connected finite resistors appear that have no length scale associated with them other than the lattice constant and L, and are thus fractal within this range. We pick a cluster large enough to connect a node close to the lower left corner to a node close to the upper right corner of the lattice (the "infinite" cluster). Current is then fed into the cluster through these two nodes (ports) from an external source, and we calculate the resulting currents through each finite resistor belonging to this cluster. The Kirchhhoff equation for each node is then

$$-\sum_j (V_i - V_j) + I_i \equiv D^2 V_i + I_i = 0, \tag{1}$$

where V_i is the voltage on the ith node, and I_i is the external current fed into the network when i denotes either of the two ports and is zero otherwise. The sum, which we denote by D^2, is over nearest-neighbor nodes connected to the ith node by a unit resistor. The simplest way to solve Eq. (1) is to write the diffusion equation

$$dV_i/dt = D^2 V_i + I_i . \tag{2}$$

Clearly, the steady-state solution of this diffusion (or relaxation) process is given by $dV_i/dt=0$, and is therefore the solution to Eq. (1).

As is typical of lattice simulations, the above relaxation method suffers from critical slowing down. To see this, consider the Fourier transform of the diffusion equation in discrete time on a Euclidean lattice

$$\tilde{P}(\mathbf{k},t+\epsilon) = \tilde{P}(\mathbf{k},t) - \epsilon k^2 \tilde{P}(\mathbf{k},t), \tag{3}$$

where $\tilde{P}$ is some density, $\epsilon = t/n$ is the discrete time step, and n is the number of iterations. The solution to this equation is

$$\tilde{P}(\mathbf{k},t) = (1-k^2\epsilon)^n \tilde{P}(\mathbf{k},0) \sim \tilde{P}(\mathbf{k},0) e^{-k^2 t}. \tag{4}$$

The relaxation time for a mode $\mathbf{k}$ is proportional to k^{-2}. The overall relaxation time T is therefore dominated by the large-scale structure (small k) of the lattice. This is critical slowing down. If we now choose $\epsilon \propto k^{-2}$ in Eq. (4), $\tilde{P}(\mathbf{k},t)$ will evolve at a constant rate for all $\mathbf{k}$. Critical slowing down is completely eliminated. This exact removal is not possible for interacting theories, although it can be reduced drastically. For more details, see Ref. 3.

This critical slowing down is more severe on fractal

lattices because diffusion is much slower there than on ordinary Euclidean lattices.[5] Consider a backbone with $N \sim L^{d_f}$ nodes. To implement the algorithm of Eq. (2) we need N operations per sweep, while the correlation length ξ, i.e., the "radius of influence" of a node at a time t, is given by $\xi \sim t^{1/d_w}$. The overall relaxation time T therefore must scale as

$$T \sim L^{d_w}. \tag{5}$$

Multiplying this with the number of operations needed per sweep, N, we find that computing time scales as $L^{d_f+d_w}$, as already quoted. For Euclidean lattices, $d_w=2$ independent of d, and on fractals $d_w>2$. For bond percolation $d_f+d_w\approx 4.5$ so that critical slowing down is very severe.

Use of a k-dependent time step in k space translates into use of a coordinate-dependent time step in real space. So, Eq. (2) becomes

$$V_i(n+1)=V_i(n)+\textstyle\sum_j'\epsilon(i,j)[D^2V_j(n)+I_j]. \tag{6}$$

ϵ is now a function of i and j, and $\sum_j'$ means that the summation runs only over the backbone since the potentials and the dynamics are only defined there. Equivalently, Eq. (6) can be understood from another point of view. It is the evolution equation of a system with coordinate-dependent and long-range couplings, but where the steady-state configuration is *identical* to that of Eq. (2). The advantage of Eq. (6) is that the presence of long-range interactions offers the possibility, through the right choice of $\epsilon(i,j)$, of faster convergence for the large-scale structure, since it is no longer being mediated only by nearest-neighbor interactions.

The question is now how to choose the site-dependent ϵ. On Euclidean lattices it is the zero-frequency (time^{-1}) Green's function for the evolution equation, D^{-2}. On fractal lattices we use the ensemble average[6] of D^{-2}. Thus,

$$\epsilon(i,j)\propto r^{d_w-d_f}, \tag{7}$$

where $r=|i-j|$. Notice that ϵ is translationally invariant, since it is an average, even though each realization of the backbone is far from being so.

To define Fourier transforms we need the entire lattice. So, instead of a restricted sum in Eq. (6), extend the summation to include all sites on the lattice, and define

$$D^2V_i(n)=0, \tag{8a}$$

$$V_i(n)=0, \tag{8b}$$

for i not belonging to the backbone. Then the algorithm works as follows: We have the configuration at the n th time step, and we use Eq. (6), with the summation extended over all lattice sites, to obtain the configuration at the $(n+1)$th step. For i on the backbone, this gives with Eq. (8a) an evolution identical to that of Eq. (6), with the restricted summation, and for i not belonging to the backbone,

$$V_i(n+1)=\textstyle\sum_j'\epsilon(i,j)[D^2V_j(n)+I_j]. \tag{9}$$

We see that nodes belonging to the backbone are updated correctly. However, the dynamics on the backbone has leaked out and affected points on the outside. Up to this point, this leakage is inconsequential since its effect has not fed back onto the backbone, but it will if we go ahead with the next update. To prevent this, we reinitialize $D^2V_i(n+1)$ and $V_i(n+1)$ to zero for all i not belonging to the backbone. Then we iterate again. Thus, the reinitialization between updates allows us to use the entire lattice instead of being confined to the backbone where we cannot define the Fourier transforms. The Fourier acceleration is implemented by writing in Eq. (6) $\epsilon(i,j)=\hat{F}^{-1}\tilde{\epsilon}\hat{F}$ to obtain

$$V_i(n+1) = V_i(n)+\textstyle\sum_j\hat{F}^{-1}\tilde{\epsilon}\hat{F}[D^2V_j(n)+I_j], \tag{10}$$

where $\tilde{\epsilon}$ is the Fourier transform of ϵ,

$$\tilde{\epsilon}\propto k^{-d+d_f-d_w}, \tag{11}$$

$\hat{F}$ is a fast Fourier transform (FFT) from real to k space, and $\hat{F}^{-1}$ is its inverse. In words, the Fourier accelerated algorithm, Eq. (10), works as follows. Given the configuration at the n th time step, compute the quantity $[D^2V_i(n)+I_i]$ at every site, and then perform a Fourier transformation on it (represented symbolically by the multiplication with $\hat{F}$). Then multiply by $\tilde{\epsilon}$ and perform a transformation back into coordinate space (the multiplication by $\hat{F}^{-1}$), and add the result to the old configuration to find the updated one. Since ϵ is translationally invariant, its Fourier transform is diagonal. Thus the matrix product between $\tilde{\epsilon}$ and $\hat{F}[D^2V_i+I_i]$ involves only L^d operations and not $L^{d_f}\times L^{d_f}$. This is the reason for doing the Fourier transforms.

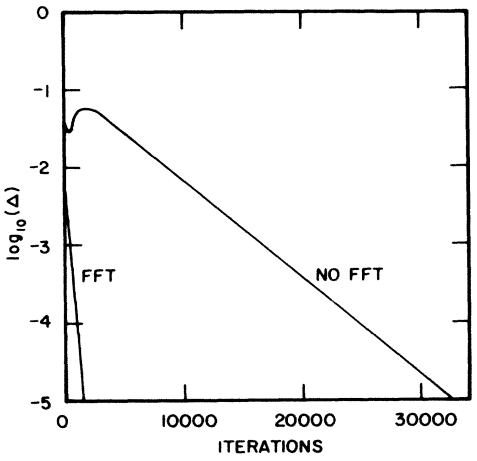

FIG. 1. A plot of $\log_{10}\Delta$, as defined by Eq. (17), vs number of iterations, for one realization on a 32×32 lattice.

Now, we discuss the stopping condition. Figure 1 shows that the convergence of the algorithm is geometric, i.e., the solution at the n th time step can be written as

$$V_i(n) = V_i(\infty) + C_i\lambda^n, \tag{12}$$

where $V_i(\infty)$ is the exact solution. λ (<1) is the largest eigenvalue of D^2, it is of the form $\exp(-\epsilon k_{\min}^2)$, and it controls the rate of convergence as discussed earlier. Clearly, if $\lambda \ll 1$, convergence is very fast, and there is no critical slowing down. The problem is that when $\lambda \approx 1$ a very large n is needed for convergence.

It might be tempting to define the "error," Δ', by

$$\Delta' = \delta^{n+1}/V, \tag{13}$$

$$\delta^{n+1} = \{\textstyle\sum_i [V_i(n+1) - V_i(n)]^2\}^{1/2}$$
$$= [\textstyle\sum_i C_i^2]^{1/2}|\lambda^n(1-\lambda)|/V$$
$$\equiv C\lambda^n(1-\lambda)/V, \tag{14}$$

$$V = [\textstyle\sum_i V_i^2(n)]^{1/2}, \tag{15}$$

and stop the iteration when Δ' is less than the desired accuracy. It is easy to show that this is a reasonable estimate of the error *only* when $\lambda \ll 1$, i.e., no critical slowing down. To see this, remember that the true error is the difference between the exact answer $[V_i(\infty)]$ and the approximate one $[V_i(n)]$; i.e., by Eq. (13)

$$\Delta = \{\textstyle\sum_i [V_i(n) - V_i(\infty)]^2\}^{1/2}/V \tag{16a}$$
$$= [\textstyle\sum_i C_i^2]^{1/2}\lambda/V = C\lambda^n/V. \tag{16b}$$

Notice that the difference between the true error Δ and the estimate Δ' is a factor of $(1-\lambda)$. If $\lambda \ll 1$, i.e., no critical slowing down, $\Delta \approx \Delta'$, and either quantity can be used as an error estimate.

However, in the case of critical slowing down, $\lambda \approx 1$, Δ' is much smaller than Δ, and even though in principle, one can take Δ' arbitrarily small, resulting in an arbitrarily high accuracy, this is not enough because this accuracy is not known. Without this knowledge a proper interpretation of the data is not possible. Δ is therefore a more accurate error estimate, and to calculate it we need to calculate $C\lambda^n$. By substitution from Eq. (13), we see that the error at the $(n+1)$th step, $C\lambda^{n+1}/V$, is given by

$$\Delta = \delta^{n+1}/V|1 - (\delta^n/\delta^{n+1})|. \tag{17}$$

We tested the above results numerically, and found that specifying, for example, $\Delta' = 10^{-6}$ for $L = 32$, resulted in accuracy of only 10^{-4}. On the other hand, specifying $\Delta = 10^{-6}$ resulted in an accuracy of order 10^{-6}. We emphasize that since our starting point was Eq. (12), the above analysis holds only for geometric convergence, which occurs here.

We tested the Fourier acceleration and compared it to unaccelerated runs on two-dimensional lattices of size 8, 16, 32, 64, and 128. It worked very well for all of them. We noticed that for the lattices of size 8, 16, and to a lesser extent 32, choosing $\tilde{\epsilon} \sim k^{-3}$, corresponding to a one-dimensional structure, worked better than the fractal form. We interpret this as being caused by the "red" bonds (links) dominating the backbone for these small lattices.[7]

In Fig. 1 we show plots of $\log_{10}\Delta$ versus number of iterations for a typical run on a lattice of size 32. Clearly the accelerated algorithm is much faster than the unaccelerated one. We checked the configurations produced by these runs and they agree to within the specified tolerance. The linear form of the plots simply means that the system reached its asymptotic regime where relaxation follows an exponential decay law, i.e., geometric convergence.

Table I shows the number of iterations and time (on an FPS 264 computer) needed on lattices of size 32, 64, and 128. These numbers are averages over twenty realizations (different backbones). For $L = 64$ and 128, the unaccelerated runs were too long to do for 20 realizations. We did a single run for $L = 64$ that took more than 10^5 iterations to reach $\Delta = 10^{-4}$, while the corresponding accelerated run took less than 10^3 iterations.

We now estimate how the number of operations is expected to scale with L. Each (fast) Fourier transform costs $L^d \log L^d$ while the multiplication by $\tilde{\epsilon}$ costs L^d operations. So, the algorithm scales as $L^d + 2dL^d \log L \sim 2dL^d \log L$ per sweep. If we assume that critical slowing down has been totally eliminated, this is, within a constant factor, the entire computational cost for convergence since the total number of sweeps will be independent of lattice size. This is to be compared to the $L^{d_f + d_w}$ operations for the unaccelerated algorithm. Clearly the gain is large and gets larger with lattice size. Table I shows that increasing L increases the number of iterations by a small factor. This means that critical slowing down has not been totally removed although its effect is drastically reduced.

TABLE I. For the accelerated and unaccelerated algorithms, the average number of sweeps and central processing unit (CPU) seconds per realization needed to reach an accuracy $\leq \Delta = 10^{-4}$. These values are averages over twenty realizations for each algorithm at each L. For $L = 64$ and 128, the unaccelerated algorithm was much too slow to run.

	FFT		No FFT	
L	Sweeps	Time (CPU sec)	Sweeps	Time (CPU sec)
32	930	30	17000	590
64	3100	310	. . .	. . .
128	6950	2750	. . .	. . .

The Fourier-acceleration ideas discussed and tested in this paper can be applied to any algorithm in which the entire lattice is updated at the same time, such as the above Jacobi method and the conjugate-gradient method.[8] The Metropolis[9] and Gauss-Seidel[8] method are not such algorithms, since updating is done site by site. This would require the calling of two FFT's per site, which is too time consuming, whereas for the Jacobi and conjugate-gradient methods, we only call two FFT's per sweep through the whole lattice. We tested our Fourier-acceleration ideas on the conjugate-gradient (CG) method[8,10] for $L=128$. We found that the unaccelerated CG method took 200 sec per backbone (averaging over 20 backbones) to solve for the voltage distribution at machine precision. The Fourier-accelerated CG method took 40 sec for the same run. We found the optimal $\epsilon(k)$ for this algorithm to be $k^{-2.3}$ whereas for the accelerated Jacobi method it was $k^{-3.2}$. The origin of this difference is that the CG method suffers less critical slowing down than the Jacobi method. Clearly, Fourier acceleration also works for the CG method.

The fact that the CG method is so much faster, for this problem, than the accelerated Jacobi method does not mean that it is always superior. The CG method does not work for systems with many local minima, such as spin glasses. For such systems, CG will find a minimum, which is more likely to be one of many local ones, instead of the desired global minimum. One efficient method for obtaining global minima is Kirkpatrick's heating-annealing method,[11] but being based on the Metropolis scheme, it cannot be Fourier accelerated. On the other hand, the heating-annealing idea can be combined with our Fourier-accelerated Jacobi method to yield a very efficient optimization algorithm. This may be done as follows: When the accelerated Jacobi algorithm finds a minimum, give the system a thermal kick by adding a random number to the right-hand side of Eq. (10). If the system were in a global minimum, it is likely to relax back to it. If the minimum were local, the thermal kick can help the system overcome the local potential barrier and relax into another minimum. As in Kirkpatrick's method, this is repeated until we find the global minimum, except that here we have Fourier acceleration working for us, and as we have shown this can lead to very substantial gains in speed and efficiency.

The authors thank C. T. H. Davies, P. Duxbury, G. H. Gunaratne, G. R. Katz, G. P. Lepage, K. Runge, and A. D. Sokal for useful discussions. This work was supported by the National Science Foundation through Grants No. PHY 82-09011 (for G.G.B.), DMR 81-08328-A (for A.H. and M.N.), and DMR 81-17011 (for A.H.), and by the Fulbright Foundation (for A.H.). Computations supporting this research were performed on the Cornell Production Supercomputer Facility, which is supported in part by the National Science Foundation, New York State, and the IBM Corporation.

---

[a]Present address: Department of Physics, Boston University, Boston, MA 02215.

[b]Address after October 1, 1986: Groupe de Physique des Solides, Ecole Normale Supérieure, 24 rue Lhomond, 75231 Paris Cedex 05, France.

[1]R. Rammal, C. Tannous, P. Breton, and A.-M. S. Tremblay, Phys. Rev. Lett. **54**, 1718 (1985); L. de Arcangelis, S. Redner, and A. Coniglio, Phys. Rev. B **31**, 4725 (1985).

[2]H. J. Herrman and H. E. Stanley, Phys. Rev. Lett. **53**, 1121 (1984); S. Havlin, D. Movshovitz, B. Trus, and G. H. Weiss, J. Phys. A **18**, L719 (1985).

[3]G. G. Batrouni, G. R. Katz, A. S. Kronfeld, G. P. Lepage, B. Svetitsky, and K. G. Wilson, Phys. Rev. D **32**, 2736 (1985).

[4]J. Goodman and A. D. Sokal, Phys. Rev. Lett. **56**, 1015 (1986).

[5]R. Rammal and G. Toulouse, J. Phys. (Paris), Lett. **44**, L13 (1983).

[6]B. O'Shaughnessy and I. Procaccia, Phys. Rev. Lett. **54**, 455 (1985), and Phys. Rev. A **32**, 3073 (1985).

[7]A. Coniglio, J. Phys. A **15**, 3829 (1982).

[8]J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis* (Springer-Verlag, New York, 1980).

[9]W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes* (Cambridge Univ. Press, Cambridge, United Kingdom, 1986).

[10]G. R. Katz, Ph.D. thesis, Cornell University, 1986 (unpublished).

[11]S. Kirkpatrick, J. Stat. Phys. **34**, 975 (1984).