Hierarchical approach to diffusive flow in heterogeneous systems

Yüksel Günal* and P. B. Visscher†
Department of Physics, University of Alabama, Tuscaloosa, Alabama 35487-0324
(Received 25 July 1996)

Conventional methods for the simulation of diffusive systems are quite slow when applied to strongly inhomogeneous systems. We present a hierarchical approach motivated by dynamic renormalization-group ideas and based on the Walsh transform (or Haar wavelet) of signal-processing theory. The method is very efficient for simulation of petroleum reservoirs or other strongly inhomogeneous diffusive or pressure-driven flow systems. As we increase the number of cells N into which a sample inhomogeneous two dimensional test system is divided, the method becomes faster than conventional finite-difference methods by $N=16\times16$, and is roughly 40 times faster at $N=64\times64$; we argue that the speedup factor should asymptotically increase at least proportionally to N^2. [S1063-651X(97)07505-3]

PACS number(s): 47.11.+j, 02.60.Pn, 47.55.Mh

Traditional finite-element and finite-difference methods for numerical solution of differential equations have a discretization error that depends on a power of the time or space increment, Δt or Δr. In the case of a diffusive system, stability usually requires Δt to be of order $\Delta r^2/D$ (where D is the diffusivity), and the discretization error is proportional to a power of Δr. In highly inhomogeneous systems, parts of which require very small Δr and/or very large diffusivity D, this requires a very small Δt.

The instability for large Δt arises when material moves more than one cell diameter during the time interval Δt; in the usual explicit finite-difference (EFD) algorithm, material is allowed to move from a cell only into its nearest neighbors. Our approach solves this problem by allowing material to move across as many cells as necessary. We describe the motion of the material by a discrete Green function or influence function $G_{\Delta t}(d,s)$ such that $G_{\Delta t}(d,s)c(s)$ is the amount of material that moves from a source cell s [whose original material content is $c(s)$] to a destination cell d during the time interval Δt. If $G_{\Delta t}(d,s)$ is nonzero only when d and s are nearest neighbor cells, this is equivalent to a conventional EFD algorithm. This is the case for sufficiently small Δt, so we may begin with such an algorithm and coarsen the time scale by doubling Δt. The influence function for the interval $2\Delta t$ is obtained from that for Δt by self-convolution [see Eq. (8) below].

Of course, this repeated convolution process increases the spatial range of the influence function, which is stored in our computer implementation as a linked list; soon the number of destination cells d that can be reached from each source cell s becomes large and the method becomes very time-consuming. However, we can coarsen the space as well as the time scale, by lumping cells together into larger cells. This decreases the number of source cells, as well as the number of destination cells reached from each source cell, and hence the size of the influence-function list. This scheme is motivated by the renormalization-group method which has been so useful in the theory of critical phenomena [1], although our present description does not require prior knowledge of renormalization-group theory. It has been shown [2] that the diffusion problem in a *homogeneous* system has a fixed point with respect to a combined space-and-time renormalization transformation. That is, we can continue coarsening the space and time scales indefinitely without indefinitely increasing the size of the influence-function list.

Such cell coarsening is even more useful in an inhomogeneous system, because we can use physical information to choose which cells to lump together. An inhomogeneous system such as an oil reservoir tends to be compartmentalized into compartments within which oil flows fairly freely, separated by relatively impermeable regions. If we lump cells between which there is relatively free flow (high effective diffusivity), when we reach a large scale the cells will *be* the compartments.

We can visualize the hierarchical lumping of cells by placing the cells on a binary tree as in Fig. 1.

The disadvantage of this coarsening process, of course, is that we lose spatial resolution in our description of the system. To some extent such a contraction of the description is desired, but we would like to maintain control of our approximations and limit the loss of information. When we replace contents $c(l)$ and $c(l')$ of two cells l and l' by the coarse-grained content $c(L)\equiv c(l)+c(l')$ where the larger cell L is the union of l and l', we can avoid losing any

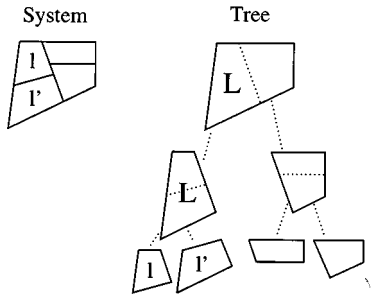

FIG. 1. Sketch of a hierarchically subdivided system (left) and its representation as a binary tree (right). Cell L is subdivided into l and l' as described in the text.

*Electronic address: gunal@pi.ph.ua.edu
†Electronic address: pv@pi.ph.ua.edu

information if we also include as a variable the difference $c(L,1)\equiv c(l)-c(l')$ as well as the sum $c(L,0)$ (the arguments 0 and 1 simply indicate whether a sum or a difference is intended).

We can do this at any level of the tree shown in Fig. 1—the difference between the two halves of the small cell l can be denoted by $c(l,1)$. We can even define a difference of differences $c(L,1,1)\equiv c(l,1)-c(l',1)$. Each 1 in this expression can be regarded as one bit of a ''Walsh sequency index'' w. (The Walsh transform is a discrete signal transform used in electrical engineering [3,4], which we here generalize to a hierarchical system.) We will lump the bits into a single binary integer (Walsh index) w, so $c(L,1,1)$ becomes $c(L,w)$ with $w=3$. We define a general ''Walsh variable'' recursively by

$$c(L,w0)\equiv c(l,w)+c(l',w),$$
$$c(L,w1)\equiv c(l,w)-c(l',w), \tag{1}$$

whenever l and l' are the two halves of the cell L. Here the Walsh index $w0$ is w with a zero bit appended at the right, i.e., $w0\equiv 2w$; similarly $w1\equiv 2w+1$. In this notation, our original $c(l)$ is written $c(l,0)$ and serves to start the recursion. The Walsh index plays a role similar to that of the wave number in the Fourier transform, in that variables with a small Walsh index describe large-scale structure, whereas large Walsh indices describe short wavelength structure within a cell L. For each 1 bit in the binary representation of w, there is one subtraction in the construction of $c(l,w)$.

If each cell lumping replaces two cell contents by two Walsh variables, the number of variables remains the same and we have gained nothing yet. However, our algorithm drops terms from the influence function if they are less than some preset tolerance δ. Typically, these are terms involving differences (i.e., having large Walsh indices), which are much smaller than those involving sums. This is the virtue of the hierarchical description: terms describing the effects of a cell content are never negligible compared to the terms for nearby cells, whereas terms describing the effects of differences may be negligible compared to terms for sums. These advantages are similar to those of spectral (Fourier transform) methods in homogeneous systems; in a sense one can regard the Walsh transform as the proper generalization of the Fourier transform to inhomogeneous systems.

Commercial reservoir simulation programs usually treat a reservoir as a three-component system [5] (oil, gas, water) in which flow is governed by Darcy's law. However, to provide a simple test of the hierarchical method described above, we will consider only one component (oil), in which case Darcy's law reduces to the diffusion equation. To see this, begin with Darcy's law for the velocity $\mathbf{v}$:

$$\mathbf{v}=-\frac{K}{\mu}\nabla P, \tag{2}$$

where K is the permeability, μ is the viscosity, and P is the pressure. Then the continuity equation for the density ρ is

$$\frac{d\rho}{dt}=-\nabla\cdot(\rho\mathbf{v}). \tag{3}$$

Linearizing in the small quantity v (and in the corresponding small variations in ρ and P) and defining the bulk modulus $B=\rho dP/d\rho$, we can express everything in terms of ρ:

$$\frac{d\rho(\mathbf{r},t)}{dt}=\nabla\cdot[D(\mathbf{r})\nabla\rho(\mathbf{r},t)], \tag{4}$$

where $D(\mathbf{r})=BK(\mathbf{r})/\mu$ is an effective diffusivity, very heterogeneous in a typical carbonaceous oil reservoir because it is proportional to the permeability K. So the problem we will actually solve is that of diffusion in a very inhomogeneous system.

To describe the evolution of the cell content $c_t(d)$ of a cell labeled d (proportional to the density ρ), the most straightforward discretization of Eq. (4) is

$$\frac{c_{t+\Delta t}(d)-c_t(d)}{\Delta t}=\frac{1}{\Delta r^2}\sum_f D(f)[c_t(d_+(f))-c_t(d)], \tag{5}$$

where the sum is over faces f of the cell d, and $d_+(f)$ is the cell in front of the (directed) face f (the cell behind it is always d). When we lump cells, so some of our variables are Walsh variables $c(d,w)$, we can write the discretization [Eq. (5)] in the form

$$c_{t+\Delta t}(d,w)=\sum_{s,v} G_{\Delta t}(d,w;s,v)c_t(s,v), \tag{6}$$

where $G_{\Delta t}(d,w;s,v)$ is an influence function describing the influence of a Walsh variable in the source cell s on one in the destination cell d. Before any lumping has occurred, all Walsh variables have $w=0$, and $G_{\Delta t}(d,0;s,0)=(\Delta t/\Delta r^2)D(f)$ if d and s are neighbors separated by the face f. The diagonal influence function $G_{\Delta t}(d,0;d,0)=1-\Sigma(\Delta t/\Delta r^2)D(f)$, where the sum is over the faces f of the cell d, and all other G's are zero. Equation (6) is thus equivalent to an EFD algorithm, which requires a small Δt; we have used the practical limit of stability [6],

$$\Delta t=\Delta r^2/4D_{\max}, \tag{7}$$

where $D_{\max}$ is the maximum diffusivity in the system.

We now increase Δt to $2\Delta t$; the new influence function is the convolution

$$G_{2\Delta t}(d,w;s,v)=\sum_{e,u} G_{\Delta t}(d,w;e,u)G_{\Delta t}(e,u;s,v). \tag{8}$$

After several such convolutions, the spatial range of the influence function is increased, especially in regions of high diffusivity. Here the pressure (i.e., density) in nearby cells equalizes rapidly—the contents $c(l)$ and $c(l')$ of two nearby cells contribute nearly equally to future contents $c(d)$: $G(d,0;l,0)\approx G(d,0;l',0)$ (the zeros here are the Walsh indices). To decide whether two cells l and l' should be lumped together, we look at the ratio

$$r=\frac{G(l,0;l',0)}{G(l,0;l,0)}. \tag{9}$$

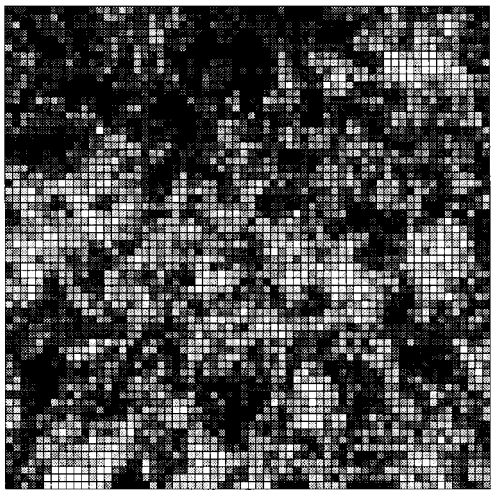

FIG. 2. The permeability distribution used for the 64×64 system [7]. Halftone density varies linearly with $\ln(K)$ or $\ln(D)$, in such a way that the mean halftone density is 0.5 and the minimum permeability displayed (as black) differs from the maximum (white) by a factor of 4×10^4, for the case of inhomogeneity $I=20$. (In case of difficulty in reproducing the figures in print, they are also presented at the web site http://www.mint.ua.edu/geo/flow/.)

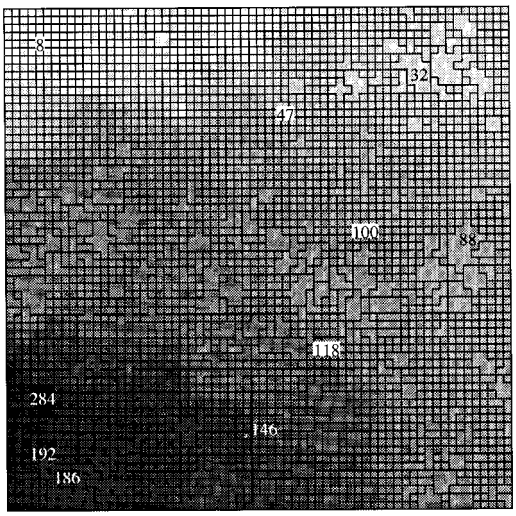

FIG. 3. The final density distribution in the 64×64 system with inhomogeneity $I=20$, showing boundaries of lumped cells. Density values are printed at several points, scaled so $\rho_\infty=100$. Note that cell lumping occurs in regions of high permeability.

We lump l and l' if r exceeds a lumping threshold $r_{\max}$, which we will tune to maximize the speed of the algorithm (see Fig. 5). When we decide to lump cells l and l' into a large cell L (as in Fig. 1), we can calculate the new influence function in two stages. In the first stage we calculate elements $G'(d,w;s,v)$ in which the *destination* cell d takes coarse values (including L) but s takes values including l and l'. These are the same as the old G's unless d is L, in which case we obtain from Eq. (1)

$$G'(L,w0;s,v)=G(l,w;s,v)+G(l',w;s,v),$$

$$G'(L,w1;s,v)=G(l,w;s,v)-G(l',w;s,v), \qquad (10)$$

where as before $w0$ means w with a zero bit appended. In the second stage, we calculate $G''(d,w;s,v)$ where both d and s take values L and not l or l': we coarsen the source cell. Again, $G''(d,w;s,v)=G'(d,w;s,v)$ unless $s=L$, in which case

$$G''(d,w;L,v0)=\tfrac{1}{2}[G'(d,w;l,v)+G'(d,w;l',v)],$$

$$G''(d,w;L,v1)=\tfrac{1}{2}[G'(d,w;l,v)-G'(d,w;l',v)]. \qquad (11)$$

Although we have developed both three dimensional (3D) and 2D programs, we have done test calculations on a 2D system to simplify the display (Figs. 2 and 3). The test system has impenetrable barriers and four control parameters: N,I,δ, and r. The first two describe the complexity of the system: N is the system size (16×16 to 64×64) and the inhomogeneity parameter I is a normalized standard deviation: the standard deviation of the permeability divided by the mean permeability.

The other two parameters are the error tolerance δ, which we choose to give an acceptable overall truncation error, and the lumping threshold $r_{\max}$ [Eq. (9), Fig. 5].

The diffusivity (i.e., permeability) distribution we have used is shown in Fig. 2. It is a realization of a log-normal distribution with fractal spatial correlations, obtained by exponentiating a correlated Gaussian distribution described elsewhere [7]. We adjust the normalized standard deviation I by scaling the Gaussian distribution before exponentiating it. The prefactor that governs the overall scale of the diffusivity or permeability can be removed from the problem by rescaling time.

The system shown in Fig. 2 is 64×64; we specify the permeability in the smaller-N systems by coarse graining (averaging over 2×2 or 4×4 cells). The permeability at a face is taken to be the average of that in the adjoining cells. As a test initial condition, we use a δ-function density concentrated in the lower left cell of the system. After an infinite time, the density takes a uniform value ρ_∞; we evolve the system until the density in the source cell is $2\rho_\infty$, and show the result in Fig. 3.

To compare our scheme with an EFD algorithm, we vary the remaining parameter, the tolerance δ. We plot in Fig. 4 the required CPU time (on a Silicon Graphics R4000PC Indy, 133 MHz) against the accuracy achieved, defined by the fractional rms error

$$(\text{error})^2\equiv N^{-1}\sum_c\left(\frac{\rho(c)-\rho_{\text{exact}}(c)}{\rho_\infty}\right)^2, \qquad (12)$$

where $\rho_{\text{exact}}(c)$ is the EFD result to which our result converges as $\delta\to0$.

Note that the speedup factor of our algorithm compared to the EFD algorithm increases rapidly as the allowable error is increased. It is indicated by an arrow at the error value of 1.5%, where it is about 25. Using this factor as a figure of merit for our algorithm, we plot it in Fig. 5 as a function of the lumping parameter $r_{\max l}$ [Eq. (9)], and choose $r_{\max}=0.9$ as an optimal value.

The virtue of our hierarchical algorithm is that it frees us from having to choose a uniform cell size Δx. The conver-

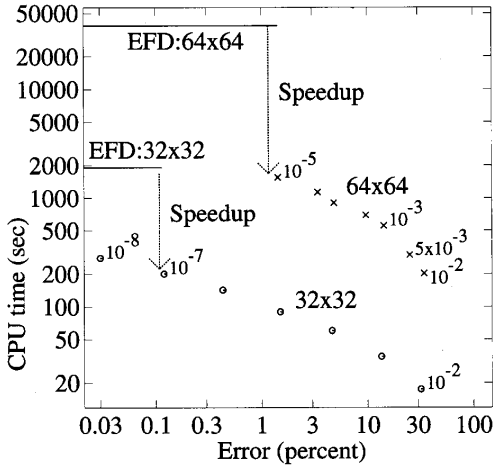

FIG. 4. Logarithmic plot of the CPU time required by our hierarchical influence function algorithm, compared to that required by an explicit finite-difference (EFD) algorithm, for 64×64 and 32×32 systems with inhomogeneity $I=20$. Smaller numbers by some data points indicate the tolerance δ (10^{-8}–10^{-2}).

gence of the method is controlled by the error tolerance δ rather than Δx or the number of cells N. We must still choose a value for N in order to specify the diffusivity at each face. However, for large N we will lump each cell many times before starting the actual evolution [Eq. (6)], so the CPU time for the evolution will become independent of the spatial resolution of the permeability distribution (i.e., of N). Figure 6 shows that we have not yet reached this asymptotic region [this is also apparent from Fig. 3, which shows that many of the original (smallest) cells remain after cell lumping]. Nonetheless, our algorithm is already much faster than the EFD, whose CPU time increases as $N^2 D_{\max}$ (because $\Delta t \propto \Delta x^2/D_{\max} \propto 1/ND_{\max}$, so the number of time steps $\propto ND_{\max}$, and the CPU time per time step also $\propto N$). This rapid increase of the CPU time of the EFD algorithm

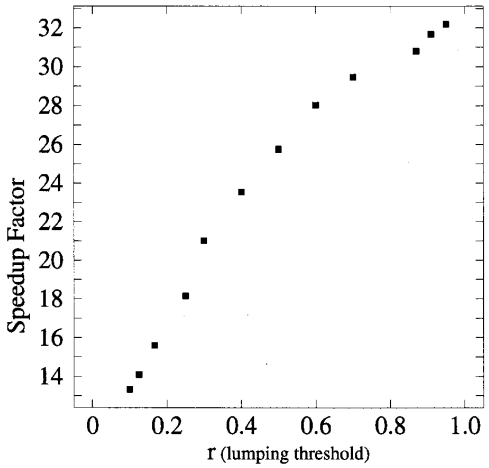

FIG. 5. Speedup factor as a function of the lumping threshold r [Eq. (9)], in a 32×32 system with inhomogeneity $I=20$ and tolerance $\delta=10^{-4}$. Evidently performance improves as we increase r toward 1.0. Placing r very close to 1.0 risks magnifying the effects of small numerical errors, so we used $r=0.9$ in the other figures.

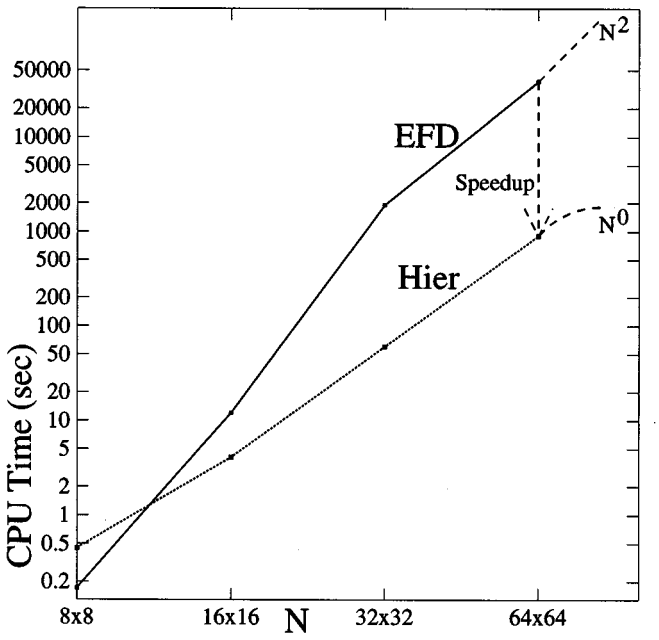

FIG. 6. Logarithmic plot of the CPU time required by the EFD algorithm and our hierarchical algorithm (labeled ''Hier''), versus the spatial resolution (the number of cells N). The dashed extensions to the right of the data points indicate the expected asymptotic behavior (N^2 or N^0, respectively); the EFD curve is less smooth because of large statistical fluctuations in the extremal value $D_{\max}$.

with N is apparent in Fig. 6. Note that the dependence of this EFD CPU time on the diffusivity $D_{\max}$ [Eq. (7)] at the single point of highest diffusivity gives it very large statistical fluctuations. (For example, there is an upward fluctuation at $N=32\times32$ which exaggerates the speedup factor for that N.) The average $D_{\max}$ clearly will increase with N (the extreme value in a large sample is likely to be larger than in a small sample) but the increase can be shown to be slower than N itself, so we have ignored it in extrapolating the data of Fig. 6 to the right with a dashed line of slope 2. The curve for our hierarchical algorithm has no such large statistical fluctuations. We are not yet in the asymptotic region where it becomes independent of N; this is also apparent from Fig. 3, which shows that many of the original (smallest) cells remain after cell lumping. However, the advantage over the EFD algorithm is already substantial.

We are using the explicit finite-difference algorithm as our point of comparison, not because it is the most efficient

TABLE I. Speedup factor as a function of system size N, for three values of the inhomogeneity I. Tolerance is $\delta=10^{-4}$. The large values at $N=32\times32$ are due to a fluctuation of $D_{\max}$ as described in the text.

	System size			
I	8×8	16×16	32×32	64×64
10	0.2	1.6	15	18
20	0.3	3.0	32	43
40	0.3	5.2	63	71

existing algorithm, but because it is the simplest. Implicit algorithms can be stable for larger Δt, but incur errors of order 100% when Δt exceeds ours by more than a factor of order 2, so this does not affect our conclusions.

The dependence of the speedup factor on the inhomogeneity is shown in Table I. Evidently the influence-function algorithm is most advantageous in highly inhomogeneous systems.

Although the method described here has some features in common with methods already in common use in grid-based numerical simulation, none of these older methods approaches its efficiency for inhomogeneous systems. The idea of coarsening cells is used in ''multigrid'' methods [8]. For example, the solution of Laplace's equation by the relaxation method is very slow on a fine grid. It can be speeded up by doing a few iterations of relaxation on a larger grid to get the coarse features of the solution correct, and then returning to the fine grid to improve the finer features. Computational fluid dynamics codes often use an ''adaptive grid'' method [9], wherein larger grid sizes are used in regions where fields do not vary rapidly in space, and finer grids are used where there are fine-scale variations in the fields. These methods do not retain subgrid-scale information such as is contained in our Walsh variables. Unlike in our approach, the time increment cannot be increased above what is stable on the finest grid.

In conclusion, we have shown that a hierarchical algorithm based on the dynamic renormalization group and the Walsh transform can simulate diffusive flow in an inhomogeneous system much more efficiently than conventional finite-difference algorithms. This occurs because the rapid power-law dependence of CPU time on the fundamental scales Δr and Δt is replaced in the hierarchical method by a logarithmic dependence.

The work described here was partially supported by the U.S. Department of Energy under Cooperative Agreement No. DE-FC02-91ER75678.

[1] K. G. Wilson and J. Kogut, Phys. Rep. **12C**, 75 (1974).
[2] P. B. Visscher, J. Stat. Phys. **25**, 211 (1981).
[3] K. G. Beauchamp, *Applications of Walsh and Related Functions, with an Introduction to Sequency Theory* (Academic Press, London, 1984).
[4] M. Holschneider, *Wavelets* (Clarendon Press, Oxford, 1995).
[5] Khalid Aziz, *Petroleum Reservoir Simulation* (Applied Science Publishers, London, 1979).
[6] E. S. Oran and J. P. Boris, *Numerical Simulation of Reactive Flow* (Elsevier Science Publishing, New York, 1987), p. 102.
[7] P. B. Visscher, Judy Dye, and Jen-Ho Fang, Comput. Phys. **7**, 217 (1993).
[8] W. L. Briggs, *A Multigrid Tutorial* (SIAM, Philadelphia, 1987).
[9] B. A. Boyett, M. S. El-Mandouh, and R. E. Ewing, in *Computational Methods in the Geosciences*, edited by W. E. Fitzgibbon and M. F. Wheeler (SIAM, Philadelphia, 1992).