

Performance of ILU Preconditioning Techniques in Simulating Anisotropic Diffusion in the Human Brain

Ning Kang^{a,1}, Jun Zhang^{a,2}, Eric S. Carlson^{b,1}

^a*Laboratory for High Performance Scientific Computing and Computer Simulation,
Department of Computer Science, University of Kentucky,
Lexington, KY 40506-0046, USA*

^b*Department of Chemical Engineering, University of Alabama,
P. O. Box 870203, Tuscaloosa,
AL 35487-0203, USA*

Abstract

We conduct simulations for the unsteady state anisotropic diffusion process in the human brain by discretizing the governing diffusion equation on a face-centered cubic grid and adopting a high performance differential-algebraic equation solver, IDA, to deal with the resulting large scale system of DAEs. Incomplete LU preconditioning techniques are used with the GMRES method to accelerate the convergence rate of the iterative solution. We then investigate and compare the efficiency and effectiveness of a number of ILU preconditioners, and find out that the ILUT with a dual dropping strategy gives the best overall performance when it is provided with the optimum choices of the fill-in parameter and the threshold dropping tolerance.

Key words: anisotropic diffusion simulation, diffusion tensor imaging, face-centered cubic grid, differential-algebraic equation, preconditioning techniques

Email addresses: nkang2@csr.uky.edu (Ning Kang), jzhang@cs.uky.edu (Jun Zhang), ecarlson@bama.ua.edu (Eric S. Carlson).

¹ The research work of this author was supported by the U.S. Department of Energy Office of Science under grant DE-FG02-02ER45961.

² The research work of this author was supported in part by the U.S. National Science Foundation under grants CCR-9988165, CCR-0092532, and ACR-0202934, in part by the U.S. Department of Energy Office of Science under grant DE-FG02-02ER45961, in part by the Kentucky Science and Engineering Foundation under grant KSEF-02-264-RED-002, in part by the Japan Research Organization for Information Science and Technology (RIST), and in part by the University of Kentucky Research Committee.

1 Introduction

The solution of the general unsteady state anisotropic diffusion equation could be of critical importance in the development of improved approaches for the analysis of diffusion tensor magnetic resonance imaging (DT-MRI). The DT-MRI technique can be exploited to visualize and extract information about the brain white matter and nerve fibers by using fiber traces, which has raised promises for a better understanding of the fiber tract anatomy of the human brain. In combination with functional MRI, it might also open a window for the crucial issue of connectivity between different parts of the brain, which is useful for functional and morphological research on the brain [2].

As diffusion is truly a three dimensional process which can be modeled by a second order tensor, the molecular mobility in tissues may be anisotropic, as in the brain white matter. It is known that anisotropic diffusion in the white matter reveals microscopic properties of the anatomy of the nerve fibers by the fact that water tends to diffuse predominantly along the fibers, because tightly packed myelin membranes restricts diffusion perpendicular to the axons [19]. The basic principal of the diffusion tensor imaging (DTI) stems from the orientation information provided by the phenomenon of water diffusion anisotropy in the white matter [14]. The diffusion tensor imaging dictates the diffusion behavior of water in tissue on a voxel by voxel basis. For each voxel, the diffusion tensor yields the diffusion coefficient corresponding to any direction in space. Given this information, diffusion anisotropy may become the reason to explain the fact that the diffusion across fiber axes encounters greater obstacle or restriction than along them. The direction of the highest diffusion coefficients is therefore believed to point along a putative fiber bundle traversing the voxel. Thus, the panoramic view of the fastest diffusion direction can be generated to provide a visualization of the white matter pathways and their orientation.

A number of fiber tracking algorithms have been developed since the appearance of DT-MRI. In [3] a variety of these algorithms are described and reviewed. As the measured quantity in DT-MRI is water diffusion, an intuitive way to understand the diffusion data is to spread a virtual concentration peak of water [7], or to specify a starting point for tractography where a seed is diffused [4]. This approach makes use of the full information contained in the diffusion tensor and it is not dependent upon a point to point eigenvalue/eigenvector computation along a trajectory, thus in that sense hopefully is more robust. It is also intuitively related to underlying physio-chemical process [13,18]. The diffusion process and related transport mechanisms in the brain are discussed in detail in [12].

Simply stated, anisotropic systems are those that exhibit a preferential flow

direction while isotropic systems are those that have no preference. According to Fick's first law, the flux, J , has magnitude proportional to the concentration gradient, ∇C , and is directed opposite to ∇C , i.e.,

$$J = -d\nabla C, \quad (1)$$

where the proportionality constant d is the so-called diffusion coefficient. In the presence of anisotropy, the flow field does not follow the concentration gradient directly, for the material properties also affect diffusion. Therefore, the diffusion tensor, D , is introduced to fully describe the molecular mobility along each direction and the correlation between these directions. Thus, the flux is given as

$$J = -D\nabla C, \quad (2)$$

and the diffusion tensor is

$$D = \begin{pmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{pmatrix}.$$

For the brain system on which we are focusing and other typical systems, the tensor is symmetric. In a reference frame $[x', y', z']$ that coincides with the principal or self directions of diffusivity, the offdiagonal terms do not exist and the tensor is reduced only to its diagonal terms. In practice, however, measurements are made in the reference frame $[x, y, z]$ of the MRI scanner gradients, which usually do not coincide with the diffusion frame of the tissue [2]. Hence, it is important to note that each component of the flux vector J may include contributions from all components of the concentration gradient.

Essentially, we are seeking to solve an unsteady state diffusion equation in an anisotropic medium based on the measured diffusion tensor D . The anisotropic diffusion process, due to conservation of mass, is governed by

$$\frac{\partial C}{\partial t} = \nabla \cdot (D\nabla C), \quad (3)$$

where t is the independent time variable. This equation says that over the time, the rate of change in concentration is proportional to the divergence of the flux.

In a Cartesian coordinate system, the Equation (3) is expressed as

$$\begin{aligned} \frac{\partial C}{\partial t} = & \frac{\partial}{\partial x} \left(D_{xx} \frac{\partial C}{\partial x} + D_{xy} \frac{\partial C}{\partial y} + D_{xz} \frac{\partial C}{\partial z} \right) \\ & + \frac{\partial}{\partial y} \left(D_{yx} \frac{\partial C}{\partial x} + D_{yy} \frac{\partial C}{\partial y} + D_{yz} \frac{\partial C}{\partial z} \right) \\ & + \frac{\partial}{\partial z} \left(D_{zx} \frac{\partial C}{\partial x} + D_{zy} \frac{\partial C}{\partial y} + D_{zz} \frac{\partial C}{\partial z} \right). \end{aligned} \quad (4)$$

This equation could be very difficult to solve under the circumstance of the human brain for a variety of reasons. First, since the brain structure is heterogeneous where anisotropy requires full tensor representation, the second order cross derivatives must be calculated. The Cartesian grids are inefficient for estimation of these. Second, the diffusion tensor changes drastically between adjacent small regions in the brain tissues. Thus, fine gridding must be used and this leads to large systems of equations. The third challenge which we are concerned with most in the current paper is how to perform necessary diffusion simulations over the whole brain with sufficient accuracy and acceptable computational time and memory cost.

We adopt several strategies in this article to tackle these difficulties. For efficient gridding scheme, the face-centered cubic (FCC) grid is employed for the discretization, with advantages over the Cartesian grid discussed in the next section. For issues involved in handling the large heterogeneous system, we exploit general purpose modules from the ACTS Toolkit [1], which includes high performance differential-algebraic-equation (DAE) system solvers, as the primary integration tools. The ACTS Toolkit is a set of tools mostly developed at the national laboratories of the U.S. Department of Energy (DOE) and universities to facilitate the development of high performance computing applications in scientific and complex research areas. ACTS stands for Advanced Computational Testing and Simulation. *One aim of this study is to take advantages of these general purpose ACTS tools and to evaluate their usefulness in this particular new application.* For the large scale sparse linear systems arising from each integration step, the iterative methods based on the Krylov subspaces are used and a number of highly efficient and robust preconditioners based on the incomplete LU factorization of the Jacobian matrix are applied as well to achieve fast solutions, since the total CPU time and memory usage are our major concerns in conducting simulations.

The remainder of the paper is organized as follows. A more detailed description of our gridding approach is presented in the next section. Also in the next section, a high performance DAE integration solver is briefly described. Section 3 outlines the GMRES iterative method and discusses in detail the incomplete LU factorization based preconditioning techniques. In Section 4, we conduct a series of numerical experiments and compare the performance of several incomplete LU preconditioners. The final concluding remarks are given in Section 5.

2 Gridding Scheme and Differential-Algebraic-Equation Solver

Among the initial steps of numerical simulations on anisotropic diffusion process is to determine a suitable and efficient gridding scheme to discretize the governing partial differential equation. In this paper, we are concerned with discretizing the 3D diffusion equation (4) on a face-centered cubic (FCC) grid using finite difference approximation. One easiest way [20] of generating the FCC grid is to color the nodes of a Cartesian grid with two colors so that no two neighbors in the directions of the coordinate axes have the same colors, and then remove all of the nodes of one of the colors. The remaining nodes form the FCC grid. They are highly structured. Figures 1 and 2 show the two dimensional case of the FCC grid with a uniform grid spacing, which is the classical triangular mesh and has a seven point hexagonal stencil with one point at the center and six points surrounding it. In the case of 3D, Figure 3 displays the three dimensional FCC mesh and the grid has a thirteen point cuboctahedral stencil, as shown in Figure 4. The node at the center of Figure 4 is symmetrically surrounded by 12 equidistant nodes. The external 12 nodes are the 12 vertices of a cuboctahedron.

A similar 2D FCC grid was discussed by Kantorovich and Krylov [11] in 1958. Boghosian [5] used the FCC grid for lattice gas modeling. Xu [20] used the FCC grid for the numerical solution of partial differential equations and referred to it as the hypersphere-close-pack grid because the location of the computational nodes corresponds to the location at the centers of densely packed equidiameter hyperspheres (at least in 2D, 3D and 4D). Sun et al. [17] investigated the FCC grid as well and presented a fourth order compact finite difference scheme on it for the numerical solution of the 2D convection diffusion equation, based on the strategy of directional derivatives. So far, the FCC grid has received relatively little attention and its potential benefits for many types of applications are not well known.

Since we use the finite difference method for the discretization, under such circumstance, all of the second order cross derivatives of the concentration variable C in Equation (4) need to be approximated at each computational node. If a Cartesian grid is used, the second order cross derivative estimations need at least nine points in 2D and 19 points in 3D. However, the FCC grid offers advantages over the Cartesian grid in the solution of Equation (4). For the 3D FCC grid in which we are now interested, the symmetry of the stencil allows for easy approximation of the second order cross derivatives by using only 13 points, six less than that required by the Cartesian grid. Thus, it achieves the same order of accuracy with far fewer grid points compared with the Cartesian grid, which makes the computation more efficient. In addition to its usefulness for assessing the cross derivative behavior, the distribution of the FCC grid is suitable for efficient multivariate interpolation. This suitability

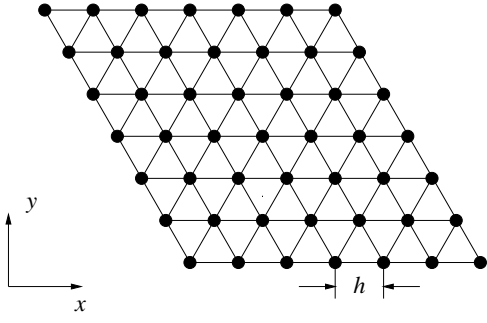


Fig. 1. The 2D FCC grid with a mesh size h [20].

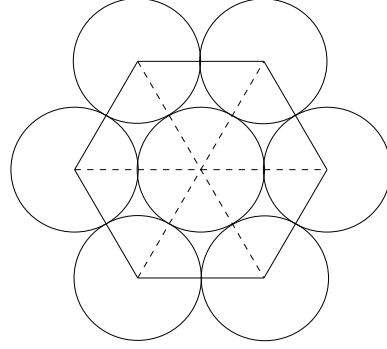


Fig. 2. The seven point finite difference scheme in the 2D FCC grid [20].

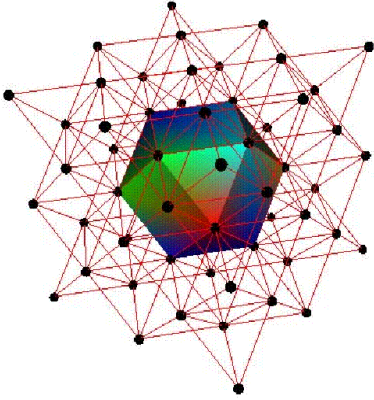


Fig. 3. The 3D FCC grid (the center unit is a cuboctahedron) [20].

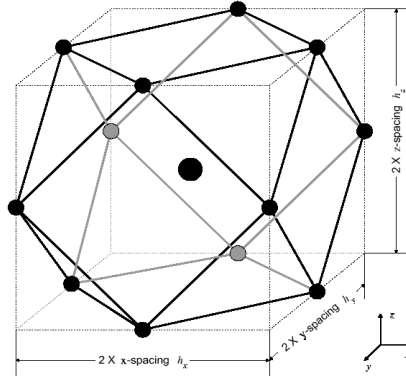


Fig. 4. The 13 point cuboctahedral stencil in a 3D FCC grid [20].

makes the FCC grid particularly appropriate for efficient, stable, and robust implementation of the multidimensional finite differencing method (MDFD), which is a general methodology for the numerical solution of partial differential equations defined on irregular domains [20].

In our present simulation, we apply the central difference in space and backward differentiation formula in time to approximate the spatial derivative and time derivative terms in Equation (4), respectively. On the boundaries of the heterogeneous system, we assume that it is insulated, i.e., $(D\nabla C) \cdot \mathbf{n} = 0$, which corresponds to the Neumann condition. This condition means that the normal part of the gradient of the concentration on the boundary is zero, in other words, nothing escapes. A 3D Gaussian function with an equal standard deviation of $\sigma = 0.2$ is selected to be the initial distribution profile of the water concentration in the brain

$$C \Big|_{t=0} = \frac{1}{\sigma^3 \sqrt{8\pi^3}} e^{-[(x-\mu_x)^2 + (y-\mu_y)^2 + (z-\mu_z)^2] / 2\sigma^2}, \quad (5)$$

where μ_x , μ_y , and μ_z are the mean in the x , y , and z direction, respectively.

The discretization of the Equation (4) and its boundary conditions on the FCC grid generates a large scale system of semi-explicit differential-algebraic equations (DAEs) with the form

$$F(t, f, f') = 0, \quad (6)$$

where f and f' are N -dimensional vectors corresponding to the discretized values of C and $\partial C/\partial t$, and the initial condition given in (5). Detailed explanations of the DAEs theories and its numerical solution methods on initial-value problems are given in [6]. In this paper, we consider using a high performance DAE solver, the IDA solver in the SUNDIALS suite, one of the software packages contained in the ACTS Toolkit. The name IDA stands for Implicit Differential-Algebraic solver, which is a general purpose solver for the initial value problem for systems of DAEs. SUNDIALS stands for SUite of Nonlinear and Differential/ALgebraic equation Solvers. More detailed information about the SUNDIALS suite and the IDA solver can be found in [9,10]. In the following, we briefly overview the algorithms used in IDA for solving DAEs. See [6,8,9] for more details.

The IDA solver uses the backward differentiation formula (BDF) method to approximate the time derivative in (6), implemented in a variable order, variable step form. It means that at every step, IDA selects the order and step size based on the behavior of the solution, in an attempt to get a solution with the minimum number of steps. The default orders of BDF range from 1 to 5. The application of BDF to the DAE system (6) leads to a nonlinear algebraic system to be solved at each time step, which is

$$G(f_n) \equiv F\left(t_n, f_n, h_n^{-1} \sum_{i=0}^k \alpha_{n,i} f_{n-i}\right) = 0, \quad (7)$$

where f_n is the calculated approximation to $f(t_n)$ and the step size is $h_n = t_n - t_{n-1}$. $\alpha_{n,i}$, $i = 0, 1, \dots, k$, are the coefficients of the BDF method, which are uniquely determined by the order of k and the step size at the previous times. IDA solves the nonlinear system (7) by a modified version of Newton iteration method regardless of the integration method options. This results in a linear system for each Newton correction, given by

$$f_n^{(m+1)} = f_n^{(m)} - cJ^{-1}G[f_n^{(m)}], \quad (8)$$

where $f_n^{(m)}$ is the m th approximation to f_n , c is a constant chosen to speed up the rate of convergence of the iteration, and J is some approximation to the system Jacobian

$$J = \frac{\partial G}{\partial f} = \frac{\partial F}{\partial f} + \alpha \frac{\partial F}{\partial f'}, \quad (9)$$

where $\alpha = \alpha_{n,0}/h_n$, which changes whenever the step size or the BDF method order changes.

During the course of integrating the system, the IDA solver imposes tolerances on the computed local truncation errors at the n th time step by using the weighted root-mean-square (wrms) norm, and requires it to satisfy the inequality

$$\|E_n\|_{wrms} = \left[\frac{1}{N} \sum_{i=1}^N (E_n^i/w^i)^2 \right]^{1/2} < 1, \quad (10)$$

where the superscript i denotes the i th component, and the i th weight is given by

$$w^i = rtol|f^i| + atol^i \quad \text{or} \quad w^i = rtol|f^i| + atol, \quad (11)$$

where $atol$ and $rtol$ stand for absolute and relative error tolerances, respectively. $rtol$ is a scalar while $atol$ may be either an N -dimensional vector or a scalar.

For the solution of the linear system (8), the IDA solver includes both direct and iterative methods. In the direct method case, the system Jacobian J expressed in (9) can be treated as either dense or banded, and in each case, the user can either provide an approximation to J or have the IDA solver compute one internally by difference quotients.

For iterative method option, the only iterative method included in the IDA solver so far is the scaled preconditioned GMRES method, denoted as SPGMR. When solving the linear system (8), a preconditioner matrix P must be supplied and need be constructed to approximate J , which leads to a cheap linear system solution, and then factored and used for as many time steps as possible. Because any nontrivial DAE needs a preconditioner, the Newton iteration test and hence the code reliability is not justified without a reasonable preconditioner. The IDA solver only allows left preconditioning. Also, it is known that the iteration matrix for any nontrivial DAE becomes more and more ill-conditioned as the step size is increased [6]. Therefore, the preconditioner may need to be rescaled. So, scaling is included explicitly in the SPGMR algorithm, using a diagonal scaling matrix whose diagonal elements are the weights w^i of (11). We will discuss more about the iterative methods and the preconditioning in the next section.

3 Iterative Methods and Preconditioning Techniques

We can rewrite the linear system (8) as

$$J[f_n^{(m+1)} - f_n^{(m)}] = -cG[f_n^{(m)}], \quad (12)$$

which needs to be solved at each Newton iteration. To simplify notation, (12) can be abstracted as

$$Ax = b, \quad (13)$$

where A is the $N \times N$ system Jacobian matrix in (9), $x = f_n^{(m+1)} - f_n^{(m)}$ and $b = -cG[f_n^{(m)}]$ are both N -dimensional vectors.

Due to the nature of the 3D problem under our consideration, direct solution methods based on Gaussian elimination are not suitable for their prohibitive CPU and memory expenses. Another thing about solving the time-dependent system is that it is often possible to use a preconditioner over more time steps when using iterative methods than it would be possible to keep an iteration matrix in the direct method, because the iterative methods do the rest of the work in solving the system. Therefore, we stick on the preconditioned iterative methods based on the Krylov subspaces.

We solve the sparse linear system (13) by using the scaled preconditioned GMRES method. GMRES is one of a class of Krylov subspace projection methods, which applies the Arnoldi process to construct an orthonormal basis of the Krylov subspace. The idea behind the method and its algorithms can be found in [15]. For any realistic and nontrivial DAE problem, due to the highly possible ill-conditioning of the coefficient matrix A in (13), the GMRES method needs to be enhanced to include the scale factors, so that all vector norms become weighted norms in the problem variables. And for robustness and efficiency, preconditioning of the linear iteration is essential, and it is beneficial to be combined into the iterative method as well. The following algorithm of the scaled preconditioned version of the GMRES method with a preconditioner matrix P and a diagonal scale matrix S is extracted from [8,15].

In Algorithm 3, the initial guess is x_0 . l_{max} is the maximum dimension of the Krylov subspace. δ is the convergence test constant. e_1 is the first standard unit vector in \mathbb{R}^{m+1} . The $\sin \theta$ elements of the Givens rotations are denoted as $s_j (j = 1, \dots, m)$. The algorithm makes use of the tricks discussed in [16] that this QR factorization can be done progressively as each column appears, and the residual norm $\|b - Ax_m\|_{wrms}$ can be obtained without calculating x_m at each substep. Hence, the program might exit earlier once this norm is small enough.

Scaled Preconditioned GMRES.

1. $r_0 = b - Ax_0$; stop if $\|r_0\|_{wrms} < \delta$
2. $\bar{r}_0 = S^{-1}P^{-1}r_0$, compute $\beta = \|\bar{r}_0\|_2 = \|P^{-1}r_0\|_{wrms}$, $v_1 = \bar{r}_0/\beta$
3. For $m = 1, \dots, l_{max}$, do:
4. Compute $w = S^{-1}P^{-1}ASv_m$
5. For $k = 1, \dots, m$, do:
6. $h_{k,m} = (w, v_k)$
7. $w = w - h_{k,m}v_k$
8. End For
9. Compute $h_{m+1,m} = \|w\|_2$ and $v_{m+1} = w/h_{m+1,m}$
10. Update QR factorization of $\bar{H}_m = (h_{ij})_{1 \leq i \leq m+1, 1 \leq j \leq m}$
11. Compute residual ρ_m indirectly by
12. $\rho_m = \|b - Ax_m\|_{wrms} = \beta \cdot |s_1 \cdots s_m|$
13. If $\rho_m < \delta$, go to 14; otherwise go to 4
14. End For
15. Define $V_m = [v_1, \dots, v_m]$, then compute y_m ,
the minimizer of $\|\beta e_1 - \bar{H}_m y\|_2$, and $x_m = x_0 + SV_m y_m$
16. If convergence criterion is satisfied, then stop;
otherwise set $x_0 = x_m$ and go to 1

The basic idea of preconditioning is as follows. Let P be a sparse matrix which approximates the coefficient matrix A in some way. Preconditioning in an iterative method for solving the linear system (13) means applying the method instead to the equivalent system $P^{-1}Ax = P^{-1}b$. We hope that the preconditioned system would be easier to solve than the original problem, i.e., the matrix $P^{-1}A$ is better conditioned than A or $P^{-1}A$ has a more favorable eigenvalue distribution than A does. In order for preconditioning to be efficient, P should be in some sense close to A and its construction should be inexpensive.

The current preconditioning techniques under our investigation are a class of preconditioners which are constructed based on the incomplete LU (ILU) factorization of the coefficient matrix A . In these cases, P is constructed in a factored form as $P = LU$, where L is a sparse lower triangular matrix with a unit main diagonal, and U a sparse upper triangular matrix. Various ILU factorizations can be derived by performing Gaussian elimination and dropping elements in some nondiagonal positions [15]. Their main differences lie in the dropping strategy applied to certain fill-in elements. Roughly speaking, based on the dropping approach used in the ILU factorizations, the ILU preconditioners can be classified into two categories. The first category uses what we call static pattern scheme, typically including ILU(0), ILU(1), etc., in which the elements are dropped only dependent on the nonzero structure of A , i.e., the locations of the nonzero entries in A . The static pattern scheme relies on the levels of fill which are known before the construction of the preconditioner matrix by using a heuristic algorithm or a symbolic factorization process, and it is blind to numerical values of the elements. So, the ILU preconditioners

using this scheme sometimes are called level-based preconditioners. Another category employs the so-called dynamic pattern scheme, where the elements are dropped based on some threshold parameters. In other words, the matrix entries produced during the Gaussian elimination process which are smaller than the predetermined threshold values would be discarded. A representative example of using the dynamic scheme is ILUT, which utilizes a dual dropping strategy. The preconditioners in this category are sensitive to their magnitudes rather than their positions.

As to the ILU preconditioners based on the static pattern scheme, the amount of fill-in and computational efforts for obtaining the $ILU(k)$ factorization cannot be predicted for $k > 0$ and the cost of updating the levels could be quite high. This scheme may also suffer from dropping large elements for indefinite matrices and lead to an inaccurate incomplete factorization [15]. However, the dynamic scheme based ILUT preconditioner which makes use of a dual dropping strategy is equipped with two parameters, τ , the threshold dropping tolerance, and p , the fill-in elements, to control the computational and memory cost, respectively. A small value of τ implies a more accurate preconditioner and a faster convergence rate but needs more time to construct. A larger value of τ has the opposite effect. The parameter p plays a major role in controlling the memory cost. By way of carefully choosing τ and p , we may be able to compute an ILU factorization efficiently without costing too much memory space.

4 Numerical Experiments

In this section, we present a number of numerical results for the performance of different ILU preconditioners on the simulation of the anisotropic diffusion in the human brain. The codes are implemented with the IDA solver package as the primary integration tool. The diffusion tensor data set used in our simulation is provided by Dr. Daniel Gembris at Institut für Medizin im Forschungszentrum Jülich (Institute for Medicine, Jülich Research Center, Jülich, Germany). The resolution of the tensor data set is $128 \times 128 \times 16$ with each voxel size being $2.5 \times 2.5 \times 7.5 \text{ mm}^3$ defined on the Cartesian mesh. Thus, the actual number of computational nodes in the FCC grid is half the size of that in the Cartesian grid, in terms of the way of generating the FCC grid as described in Section 2. The 3D domain is discretized with a 13 point difference stencil such that there are 131,072 unknowns in the resulting linear systems and 1,622,560 nonzeros in the system Jacobian matrix. Natural ordering of the FCC grid points is used. The numerical tests are conducted on a Sun Blade 100 workstation with a single 500 MHz UltraSPARC-IIe processor and 128 MB memory.

The following testing parameters are set fixed for all numerical experiments that we carry out. The total integration time is 1.017×10^5 seconds. The sparse linear systems are solved by GMRES(20) with the maximum number of restarts being five. Thus GMRES gets restarted for every 20 iterations to avoid large memory cost. The maximum order in the BDF method is set at 10. In order to have an acceptable convergence rate, the maximum time step size is not allowed to exceed 5×10^3 . We select both the relative tolerance and absolute tolerance to be 1×10^{-4} , and the linear and nonlinear iteration convergence factors are chosen to be 1×10^{-5} and 0.5, respectively, in hope of achieving reasonable accuracy without taking too much computing time.

In the tables containing numerical results, the entry of “preconditioner” specifies the ILU preconditioner used; “sparsity” means the sparsity ratio, which is derived from the number of nonzero entries in the preconditioner matrix divided by that of the Jacobian matrix; “iteration” shows the total number of scaled preconditioned GMRES iterations during the course of integration; “setup” is the total CPU time in seconds to construct the preconditioners; “solution” is the total CPU time in seconds to solve the given DAE system; “ratio” stands for the percentage ratio of the total preconditioner construction time over the total solution time; τ and p are the two parameters of the ILUT preconditioner. The total number of nonlinear iterations is 42 for all testing cases, while the preconditioner construction routine is totally called six times during the integration (in the case of using ILU(0) but with the ILUT data pattern, the number of called times is seven), which means it is reused over a number of time steps to approximate the Jacobians without changing itself.

Table 1 shows the performance results obtained by using the ILUT(p, τ) preconditioner, with a fill-in parameter p and a threshold dropping tolerance τ . We can see that the ILUT preconditioner converges in all the cases. When p is fixed, with the decrease of τ , it takes more time to construct the preconditioner and the percentage ratio of the preconditioner setup time over the total solution time expense also gets increased. Since the preconditioner becomes more accurate with the reduction of τ , the convergence performance of the GMRES method gets better with the trend of having less and less iterations, but with the price of more time to obtain the solution. We notice that in the column of “iteration”, the number of total linear iterations do not manifest a monotonic decrease along with τ . The column of “solution”, the total CPU cost, displays the similar behavior as well. The reason may exist in the magnitudes of the matrix entries in that it is highly possible that the convergence rate of the GMRES method thus the solution is very sensitive to some small elements in the preconditioner matrix. If the threshold dropping tolerance τ remains constant, the number of iterations declines as the fill-in parameter p grows larger, and at the same time the corresponding preconditioner setup time and the total CPU time go up. Therefore, the quality and accuracy of the ILUT preconditioner could be adjusted by selecting appropriate values for

the parameters τ and p .

We may come up with the idea as well from the sparsity ratio diagrams of the preconditioners depicted in Figure 5, that the ILUT preconditioner supplies a flexible computational tool which is desirable for realistic applications with varying memory requirements. The definition of sparsity ratio tells us that it is an indicator of the nonzero density and the storage cost of a preconditioner. Figure 5 illustrates the curves of the sparsity ratio of the ILUT preconditioners constructed at each call by the IDA solver during the evolution of the time integration. The sequence number that the ILUT is called to reconstruct is shown on the horizontal coordinate of the two subfigures. It is clear from Figure 5 that when the number of fill-in elements is fixed, the sparsity ratios of ILUT grow larger along with the reduction of the dropping tolerance τ , shown in the left panel. On the other hand, if τ keeps unchanged, the ILUT sparsity ratios increase with the fill-in parameter p , as given in the right panel of Figure 5. Roughly speaking, the ILUT preconditioner with a larger average sparsity ratio achieves a better convergence behavior, as revealed by the smaller number of iterations in the third column of Table 1. However, larger average sparsity ratio also means more computational efforts, illustrated in the “setup” and “solution” columns in Table 1. So, we need to take into consideration the tradeoff between the effects made by the fill-in parameter p and the dropping tolerance τ for our problem, and try to find some compromised values for keeping the sparsity ratio relatively small and at the same time maintaining the ILUT preconditioners to be effective on convergence.

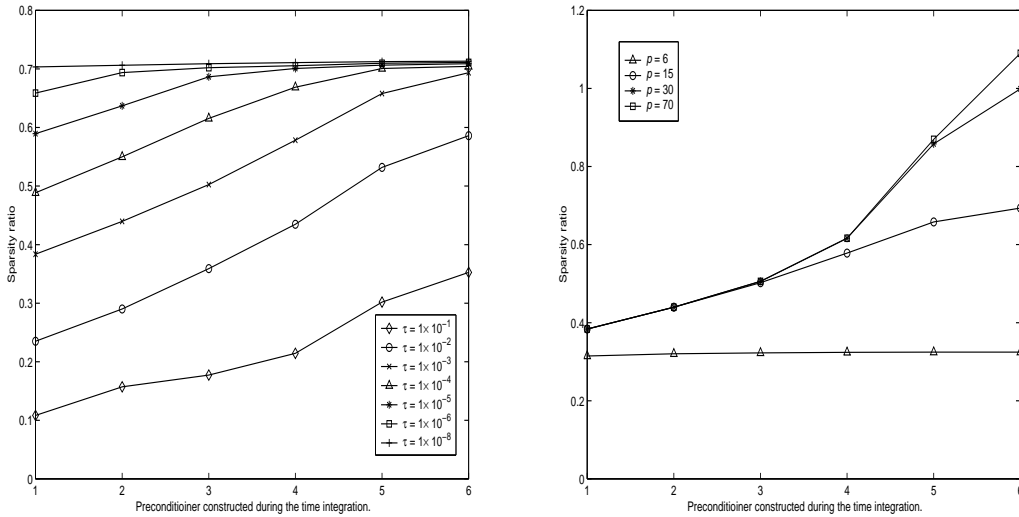


Fig. 5. The diagram of the sparsity ratio of the ILUT preconditioners constructed during the time integration. Left: ILUT with $p = 15$ and variable τ . Right: ILUT with $\tau = 1 \times 10^{-3}$ and variable p .

The numerical results of applying the static pattern scheme based $ILU(k)$ preconditioners are contained in Table 2. For the purpose of comparison, a set of ILUT data with $p = 15$ and $\tau = 10^{-2}$ is included as well. Table 2

p	τ	iterations	setup	solution	ratio%
6	10^{-1}	192	4.12	366.66	1.1
	10^{-2}	161	6.34	394.40	1.6
	10^{-3}	174	8.19	431.58	1.9
	10^{-4}	182	10.65	428.48	2.5
	10^{-5}	172	13.51	420.46	3.2
	10^{-6}	170	17.28	404.44	4.3
	10^{-8}	166	24.97	407.65	6.1
15	10^{-1}	158	4.14	337.36	1.2
	10^{-2}	137	7.93	367.73	2.2
	10^{-3}	148	12.25	437.83	2.8
	10^{-4}	155	19.57	520.96	3.8
	10^{-5}	138	28.35	529.80	5.4
	10^{-6}	152	38.60	527.05	7.3
	10^{-8}	128	74.77	649.50	11.5
30	10^{-1}	158	4.53	358.90	1.3
	10^{-2}	132	8.24	380.42	2.2
	10^{-3}	143	15.13	482.74	3.1
	10^{-4}	117	25.37	478.56	5.3
	10^{-5}	134	41.90	680.08	6.2
	10^{-6}	120	63.89	734.75	8.7
	10^{-8}	117	119.19	770.54	15.5
70	10^{-1}	158	4.29	352.55	1.2
	10^{-2}	152	8.32	389.37	2.1
	10^{-3}	129	15.35	452.74	3.4
	10^{-4}	110	29.70	481.99	6.2
	10^{-5}	128	58.31	791.26	7.4
	10^{-6}	102	94.96	770.18	12.3
	10^{-8}	98	208.93	1131.77	18.5

Table 1
Performance data of ILUT with varying fill-in parameter p and threshold dropping tolerance τ .

preconditioner	sparsity	iterations	setup	solution	ratio%
ILU(0)	0.35	184	8.46	504.36	1.7
ILU(1)	0.60	144	20.03	463.37	4.3
ILU(2)	1.12	124	64.46	613.32	10.5
ILU(3)	1.79	116	160.25	964.85	16.6
ILUT	0.41	137	7.93	367.73	2.2

Table 2

Performance data of $ILU(k)$, $k = 0, 1, 2, 3$, compared with $ILUT(p = 15, \tau = 10^{-2})$. The sparsity for ILUT is an average value over the whole time integration period.

p	τ	iterations	setup	solution	ratio%
6	10^{-4}	159	10.96	368.39	3.0
15	10^{-3}	175	11.30	449.91	2.5
30	10^{-4}	165	11.97	417.60	2.9
70	10^{-4}	184	12.22	418.06	2.9

Table 3

Performance data of $ILU(0)$ with the ILUT data pattern applied.

demonstrates the performance of $ILU(k)$ with $k = 0, 1, 2, 3$.³ Among all the preconditioners compared, ILUT is the most efficient one, which is reflected by both the setup time and the total solution time. Although $ILU(2)$ and $ILU(3)$ have very good convergence rate, they are not desirable in our problem because of their high computational cost. It seems that $ILU(1)$ should have a smaller number of iterations compared to ILUT since its sparsity ratio is larger than ILUT, but in fact it suffers from the fact that the Jacobian matrix has too many small magnitude entries, which takes $ILU(1)$ more time to do computation. In contrast, the ILUT based on the dynamic pattern scheme might avoid the suffering haunted in $ILU(k)$ by removing the small magnitude elements during the construction.

In Table 3, the performance data are given under the case of using the preconditioner of $ILU(0)$ but with the ILUT data pattern applied. The actual calls for the preconditioner construction are seven in this scenario because in the implementation, an ILUT is called first and only called once for preprocessing to extract its data pattern before the six successive $ILU(0)$ subroutines are called. The results in Table 3 are not very competitive compared to the ILUT with optimum choices, although the composite preconditioner does have less total CPU cost and improved convergence rate than the original $ILU(0)$.

³ The original Jacobian matrix contains many numerical zero entries which are removed during the ILU factorization. This results in some $ILU(k)$ preconditioners with a sparsity ratio less than 1.

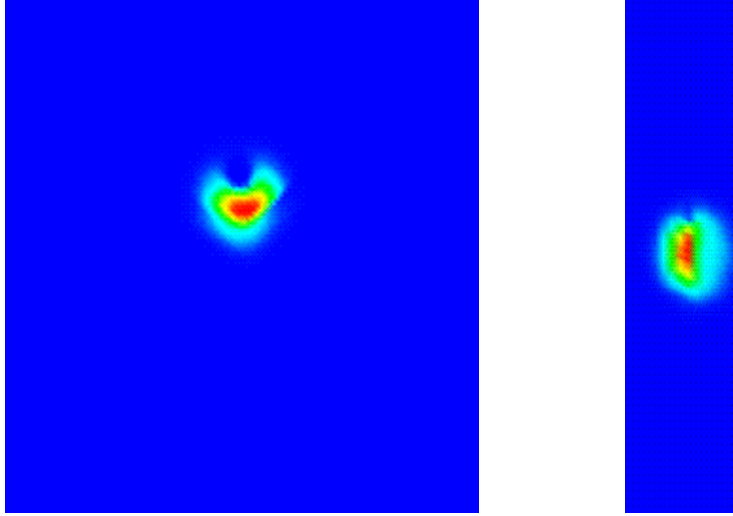


Fig. 6. The concentration distribution profiles of the anisotropic diffusion in the human brain. Left: the profile on the cutting plane of $z = 8$. Right: the profile on the cutting plane of $y = 75$.

For completeness, shown in Figure 6 are the simulated concentration distribution profiles of the anisotropic diffusion in the human brain. The simulated result on the 3D Cartesian grid is derived from interpolation when we map the solution on the FCC grid back to the Cartesian grid.

5 Concluding Remarks

In the current work, we simulate the anisotropic diffusion process in the human brain, which could be of vital importance for the analysis of DT-MRI. The face-centered cubic (FCC) grid is utilized to discretize the anisotropic diffusion equation for its efficient approximation of the second order cross derivatives and the multivariate interpolation. A high performance DAE solver, IDA, with a scaled preconditioned GMRES iterative method, is employed to solve the resulting large scale system of DAEs. By applying a number of different incomplete LU preconditioning techniques for the GMRES method, we perform numerical experiments to investigate the efficiency and effectiveness of the preconditioners in solving the linear systems arising at each Newton iteration.

The ILU preconditioners based on the static pattern scheme, $ILU(k)$, are found to be inefficient for our problem. The best choice among all the preconditioners that we have investigated is the dynamic pattern scheme based ILUT, with optimum choices of its two thresholding parameters p and τ . The composite preconditioner in which $ILU(0)$ uses the ILUT data pattern shows better performance than $ILU(0)$, but no better than ILUT. Our numerical tests illustrate that regardless of the thresholding strategies applied in the ILU precon-

ditioning, the choice of the corresponding parameters has direct and distinct influences on the accuracy and the construction cost of the preconditioner, the convergence rate of the iterative solution, and the total computational efforts. In general, the more entries kept in the factorization, the higher quality of the ILU preconditioner, which makes the iterative solution more robust but with the price of more construction time.

Acknowledgments

This study was funded by the U.S. Department of Energy Office of Science under the project “Development of a High Performance Anisotropic Diffusion Equation Solver Using the ACTS Toolkit” (DE-FG02-02ER45961). The first two author would also like to acknowledge DOE’s support of their attending the ACTS Workshop, organized by Drs. Tony Drummond and Osni Marques, at the Lawrence Berkeley National Laboratory, in September 2002. We would also like to thank Dr. Daniel Gembris, at Institute for Medicine, Jülich Research Center, Jülich, Germany, for providing the diffusion tensor data set.

References

- [1] <http://acts.nersc.gov/>.
- [2] D. L. Bihan, J. F. Mangin, C. Poupon, C. A. Clark, S. Pappata, N. Molko, and H. Chabriat, Diffusion tensor imaging: concepts and application, *J. Magnetic Resonance Imaging*, 13:534-546, 2001.
- [3] A. B. M. Bjornemo, *White Matter Fiber Tracking Using Diffusion Tensor MRI*, Master’s Thesis, Linkoping University, Sweden, 2002.
- [4] P. G. Batchelor, D. L. G. Hill, F. Calamante, and D. Atkinson, Study of connectivity in the brain using the full diffusion tensor from MRI, *Information Processing in Medical Imaging*, 17th International Conference, IPMI’01, June 2001, UC Davies, USA. Published by Springer, *Lecture Notes in Computer Science 2082*, pp. 121-133.
- [5] B. M. Boghosian, Lattice gases and cellular automata, *Future Generation Computer Systems*, 16:171–185, 1999.
- [6] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, Philadelphia, PA, 1996.

- [7] D. Gembris, H. Schumacher, and D. Suter, Solving the diffusion equation for fiber tracking in the living human brain, *Proc. of the International Society for Magnetic Resonance Medicine (ISMRM)*, 9:1529, Glasgow, Scotland, April 2001.
- [8] P. N. Brown, A. C. Hindmarsh, and L. R. Petzold, Using Krylov methods in the solution of large-scale differential-algebraic systems, *SIAM J. Sci. Comput.*, 15:1467-1488, 1994.
- [9] A. C. Hindmarsh and A. G. Taylor, *User Documentation for IDA, a Differential-Algebraic Equation Solver for Sequential and Parallel Computers*, LLNL Report UCRL-MA-136910, Center for Applied Scientific Computing, LLNL, Livermore, CA, 1999.
- [10] <http://acts.nersc.gov/sundials/main.html>.
- [11] L. V. Kantorovich and V. I. Krylov, *Approximate Methods of Higher Analysis*, New York, Interscience Publishers, 1958.
- [12] C. Nicholson, Diffusion and related transport mechanism in brain tissue, *Reports on Progress in Physics*, 64:815-884, 2001.
- [13] C. Nicholson and E. Syková, Extracellular space structure revealed by diffusion analysis, *Trends in Neurosciences*, 21(5):207-215, 1998.
- [14] C. Poupon, J. F. Mangin, C. A. Clark, V. Frouin, J. Régis, D. L. Bihan, and I. Bloch, Towards inference of human brain connectivity from MR diffusion tensor data, *Medical Image Analysis*, 5:1-15, 2001.
- [15] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, MA, 1996.
- [16] Y. Saad and M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.*, 7:856-869, 1986.
- [17] H. Sun, N. Kang, J. Zhang, and E. S. Carlson, A fourth order compact difference scheme on face centered cubic grids with multigrid method for solving 2D convection diffusion equation, Technical Report, No.341-02, Department of Computer Science, University of Kentucky, Lexington, KY, 2002.
- [18] I. Vorisek and E. Sykova, Evolution of anisotropic diffusion in the developing rat corpus callosum, *J. Neurophysiol.*, 78:912-919, 1997.
- [19] C. F. Westin, S. E. Maier, B. Khidir, P. Everett, F. A. Jolesz, and R. Kikinis, Image processing for diffusion tensor magnetic resonance imaging, *Proceedings of Second International Conference on Medical Image Computing and Computer-assisted Interventions (MICCAI)*, 441-452, Cambridge, England, July 1999.
- [20] J. Xu, *Multidimensional Finite Differencing (MDFD) with Hypersphere-Close-Pack Grids for Numerical Solution of PDE Defined on Irregular Domains*, Ph.D. Thesis, University of Alabama, Tuscaloosa, AL, 2001.