Commun. Comput. Phys.
doi: 10.4208/cicp.101110.061211a

Vol. **12**, No. 4, pp. 1148-1162
October 2012

A Fast Direct Solver for a Class of 3-D Elliptic Partial Differential Equation with Variable Coefficient

Beibei Huang[1,2,*], Bin Tu[1] and Benzhuo Lu[1]

[1] *Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and System Sciences, Academy of Science, Beijing 100190, P.R. China.*
[2] *Departament d'Enginyeria Química, Universitat Rovira i Virgili, Av. dels Països Catalans, 26 Tarragona 43007, Spain.*

Received 10 November 2010; Accepted (in revised version) 6 December 2011

Communicated by Bo Li

Available online 17 April 2012

Abstract. We propose a direct solver for the three-dimensional Poisson equation with a variable coefficient, and an algorithm to directly solve the associated sparse linear systems that exploits the sparsity pattern of the coefficient matrix. Introducing some appropriate finite difference operators, we derive a second-order scheme for the solver, and then two suitable high-order compact schemes are also discussed. For a cube containing N nodes, the solver requires $\mathcal{O}(N^{3/2}\log^2 N)$ arithmetic operations and $\mathcal{O}(N\log N)$ memory to store the necessary information. Its efficiency is illustrated with examples, and the numerical results are analysed.

AMS subject classifications: 15A15, 15A09, 15A23

Key words: Fast solver, direct method, discrete Laplace operator, fast matrix inversion.

1 Introduction

In dealing with sparse linear algebraic systems that arise in the discretization of elliptic partial differential equations, iterative solvers require the pre-handling of ill-conditioned matrices, e.g., the Conjugate Gradient and Generalized Minimum Residual methods. On the other hand, compared with the iterative solvers using direct elimination can always get result easier when dealing with poorly conditioned coefficient matrices usually.

In a typical direct solver, there is usually an initial ordering step to reorder the rows and columns, so that the transformed coefficient matrix has some special structure such

*Corresponding author. *Email addresses:* hbb21st@lsec.cc.ac.cn (B. Huang), tubin@lsec.cc.ac.cn (B. Tu), bzlu@lsec.cc.ac.cn (B. Lu) as block-triangular form. The internal structure of the dense matrices may also be exploited, to reduce the computational cost [4,5]. For example, a spiral pattern of the orderings that arise from a 2-D elliptic PDE can render the linear system in a block-tridiagonal form [1]; and it has also been shown that a sweeping ordering efficiently solves a 2-D discrete system arising from a moving perfectly matching layer (PML), using banded LU-factorization [2]. Another technique for dimension-reduction, with a much simpler format to deal with such structured matrices [1], has influenced us in designing our direct solver. It combines a similar dimension-reduction technique with fast algorithms for the spiral pattern, to solve the resulting sequence of sparse coefficient matrices.

Our main application is to a Poisson equation with variable coefficient, which arises in many areas including electric or electromagnetic field theory and heat conduction, i.e.,

$$\nabla\cdot\rho\nabla u=f. \tag{1.1}$$

For example, Eq. (1.1) applies in the theory of electrolyte solutions, where the distribution of counterion density strongly depends on the dielectric coefficients [7,8]. Changes in the dielectric coefficient for the electrolyte solution (from 10 to 25, 40, 60, and 78.5 within the first 7.4 Angströms at the surface of DNA) substantially increase the calculated surface concentration of counterions of all sizes. In a contoured lattice model involving a dielectric boundary and Boltzmann equation for the charge density, the Poisson equation

$$-\nabla\cdot[\varepsilon(r)\nabla\phi(r)]=\rho(r)/\varepsilon_0$$

of form similar to (1.1) can be approximated by the finite element representation

$$\sum_j[(\phi_i-\phi_j)\varepsilon_{ij}]=\rho_i h^2/\varepsilon_0,$$

where ε_{ij} is the arithmetic average of the dielectric coefficients (for the elements i and j) and ρ_i denotes the relevant value of the charge distribution. Yet another example arises in diffusion-reaction processes [9], where

$$\begin{aligned}&\frac{\partial p^i(r,t)}{\partial t}=\nabla\cdot\left\{D^i(r)e^{-\beta V^i(r,t)}\nabla(e^{\beta V^i(r)}p^i(r,t))\right\}+\alpha^i(r)p^i(r,t),\\&\nabla\cdot\epsilon\nabla\phi(r,t)=-\rho^f(r)-\sum_i q^i p^i(r,t)\end{aligned}$$

involves the density distribution function $p^i(r,t)$ of the diffusing particles of the ith species with diffusion coefficient $D^i(r)$ and charge q^i, the fixed source charge distribution ρ^f, the inverse Boltzmann energy β, the dielectric coefficient ϵ, the potential V^i that imposes driving forces on the ith diffusing species, and the intrinsic reaction rate $\alpha^i(r)$. The dielectric coefficient actually depends in a complicated way on the pressure, temperature and material density, but for simplicity it was argued that one may adopt the linear form

$$\epsilon=\epsilon_p+\frac{p^w}{p_0^w}*(\epsilon_w-\epsilon_p), \tag{1.2}$$

where ϵ_p and ϵ_w denote the dielectric coefficients of solute and solvent respectively, p^w the water density, and p_0^w the bulk density (55.5 M in the standard state). The justification given for the simple form (1.2) is that the induced dipole moment in the water is approximately proportional to its density. Following our approach to numerically solve the Poisson equation with variable coefficient, the discretization would treat $\varepsilon(r)$ as a continuous function of r and one of two finite difference formats (FDF) with different precision would be invoked. For a system matrix arising from a cube containing N nodes, the schemes for both of these FDF require $N^{\frac{3}{2}}(\log N)^2$ arithmetic operations and $\mathcal{O}(N\log N)$ memory.

This paper is structured as follows. In Section 2, we solve the standard 3-D Poisson equation, as a natural extension of earlier calculations for the corresponding 2-D equation [1]. The adjustment to solve the elliptic PDE with variable coefficient (1.1) is described in Section 3.1. The high-order compact finite difference schemes for the solver are discussed in Section 3.2, where it is explained why they are suitable for our solver. The results of the numerical experiments and associated problems are then presented in Section 4, before a final summary in Section 5.

2 Solver for standard 3-D Poisson equation

2.1 Seven-point stencil

The direct solver used for the linear system is associated with a seven-point stencil (cf. Fig. 1), which can be regarded as an extension of the standard five-point stencil.

In this section, we consider the boundary value problem consisting of the standard 3-D linear Poisson equation in Cartesian coordinates

$$\nabla^2 u = f, \quad (x,y,z) \in \Omega, \tag{2.1}$$

where

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2},$$

subject to the Dirichlet boundary condition $u = u_0$ where $(x,y,z) \in \partial\Omega$. Assuming an equal grid size in all three Cartesian directions (x,y,z) denoted by h, we proceed to calculate the solution values at the inner nodes that approximate $u(x,y,z)$ for $(x,y,z) \in \Omega$. In the x-direction we take

$$\begin{aligned}
\frac{\partial^2 u}{\partial x^2} &= \frac{1}{h^2}\Delta^2_{0,x} u_{i,j,k} + \mathcal{O}(h^2),\\
\frac{\partial^2 u}{\partial y^2} &= \frac{1}{h^2}\Delta^2_{0,y} u_{i,j,k} + \mathcal{O}(h^2),\\
\frac{\partial^2 u}{\partial z^2} &= \frac{1}{h^2}\Delta^2_{0,z} u_{i,j,k} + \mathcal{O}(h^2),
\end{aligned}$$

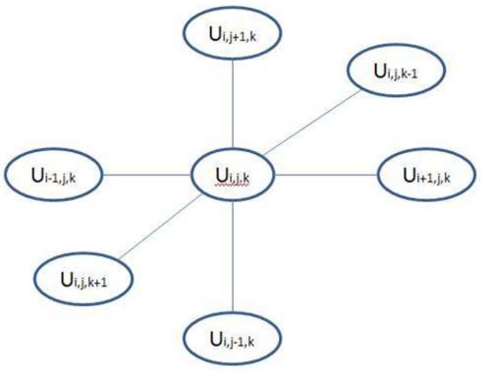

Figure 1: The 7-point stencil.

using the relevant central difference operator such that

$$\Delta_{0,x}^2 u_{i,j,k} = u_{i-1,j,k} + u_{i+1,j,k} - 2u_{i,j,k}.$$

On similarly involving central difference operators $\Delta_{0,y}$ and $\Delta_{0,z}$ in the other two directions y and z, we obtain the following FDF:

$$\begin{aligned}\nabla^2 u|_{i,j,k} &= \frac{1}{h^2}\left(u_{i-1,j,k}+u_{i+1,j,k}+u_{i,j-1,k}+u_{i,j+1,k}+u_{i,j,k-1}+u_{i,j,k+1}-6u_{i,j,k}\right)\\ &= f_{i,j,k}.\end{aligned} \tag{2.2}$$

2.2 Direct method for solving the standard 3-D Poisson equation

Let m denote the number of layers consisting of $(2m+1)^3$ uniform cubes, as illustrated for $m=0$,1 and 2 in Fig. 2 (where only 3 layers are shown for the case $m=2$).

The outer cube defines the boundary of the model, and every cube other than an outer cube is completely enclosed by other cubes.

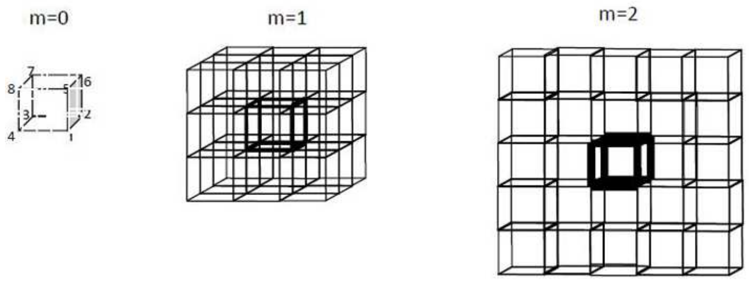

Figure 2: Uniform cubes with spiral orderings.

We collect the nodes into index sets $J_1, J_2, \cdots, J_m$, and number the nodes in each layer in order. Thus

$$\begin{aligned}
&J_1 = \{1,2,3,4,5,6,7,8\},\\
&J_2 = \{9,10,11,12,\cdots,64\},\\
&J_3 = \{65,66,67,\cdots,216\},\\
&\vdots\\
&J_m = \{(2m-2)^3+1,(2m-2)^3+2,(2m-2)^3+3,\cdots,(2m)^3\},
\end{aligned}$$

where J_m denotes the set of identifiers of the nodes in the m-th layer, and each identifier denotes a row number or column number of an element in the coefficient stiffness matrix. Thus each block in this matrix is a coefficient submatrix, which is generated from the nodes in the nearby layers. The associated linear system

$$\begin{bmatrix} A_{11} & A_{12} & 0 & 0 & \dots & 0 \\ A_{21} & A_{22} & A_{23} & 0 & \dots & 0 \\ 0 & A_{32} & A_{33} & A_{34} & \dots & 0 \\ 0 & 0 & A_{43} & A_{44} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & A_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ \vdots \\ f_m \end{bmatrix} \tag{2.3}$$

takes on block-tridiagonal form, where the matrix sparsity pattern is shown in Fig. 3. Nonzero submatrices are composed from nearby layers, and the block A_{mn} is such that $|m-n| \leq 1$, so for example A_{23} denotes the coefficient block from the second and the third layer.

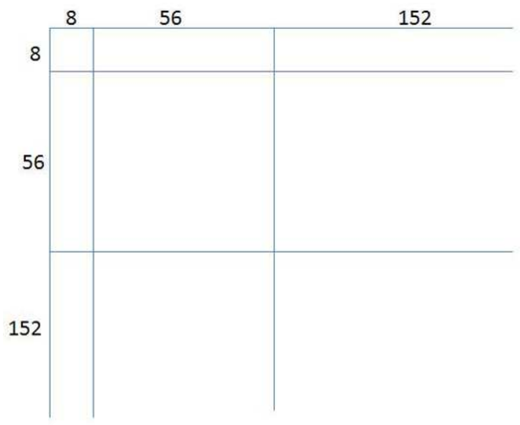

Figure 3: The sparsity pattern of the coefficient matrix.

We consequently partition the coefficient matrix into four sub-matrices, so the original matrix in Eq. (2.3) is partitioned as

$$M_0=\left[\begin{array}{c:cccccc} A_{11} & A_{12} & 0 & 0 & \ldots & 0 \\ \hdashline A_{21} & A_{22} & A_{23} & 0 & \ldots & 0 \\ 0 & A_{32} & A_{33} & A_{34} & \ldots & 0 \\ 0 & 0 & A_{43} & A_{44} & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & A_{mm} \end{array}\right]. \tag{2.4}$$

We proceed analogously to Gaussian elimination for solving a system of linear equations, except that here *sub-matrices* are the pivots. Thus we first eliminate A_{21} by post-multiplying the first row by $-A_{11}^{-1}A_{21}$ and adding it to the second row. Then we ignore the first row and first column in the result, to proceed to consider the reduced algebraic system

$$\begin{bmatrix} \widetilde{A}_{22} & A_{23} & 0 & \ldots & 0 \\ A_{32} & A_{33} & A_{34} & \ldots & 0 \\ 0 & A_{43} & A_{44} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & A_{mm} \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} \widetilde{f}_2 \\ f_3 \\ f_4 \\ \vdots \\ f_m \end{bmatrix}, \tag{2.5}$$

where $\widetilde{A}_{22}=A_{22}-A_{21}A_{11}^{-1}A_{12}$, $\widetilde{f}_2=f_2-A_{21}A_{11}^{-1}f_1$ and the new coefficient matrix here may be denoted by M_1. (The difference between M_1 and the corresponding sub-matrix of M_0 is that the first block A_{22} becomes $\widetilde{A}_{22}$, where $\widetilde{A}_{22}$ is the Schur complement of A_{22} in the matrix M_0.) Further such elimination row by row successively produces new coefficient matrices M_2, M_3, $\cdots$ and new right hand sides f_2, f_3, $\cdots$, until we have the last matrix M_{m-1} corresponding to the linear algebraic system: $\widetilde{A}_{m,m}x_m=\widetilde{f}_m$. In pseudocode, this procedure is as follows:

```
(1) $\widetilde{A}_{11}=A_{11}$, $\widetilde{f}_1=f_1$
(2) For $k=2$ to $m$
(3) $\widetilde{A}_{k,k}=A_{k,k}-A_{k,k-1}\widetilde{A}_{k-1,k-1}^{-1}A_{k-1,k}$
(4) $\widetilde{f}_k=f_k-A_{k,k-1}\widetilde{A}_{k-1,k-1}^{-1}\widetilde{f}_{k-1}$
(5) End
```

We then get x_m from the last linear system and proceed by back substitution, similar to the process following Gaussian elimination but now of course involving matrices. Thus the value of x_{m-1} is derived by solving the linear system

$$\widetilde{A}_{m-1,m-1}x_{m-1}=\widetilde{f}_{m-1}-A_{m-1,m}x_m,$$

and subsequently the values $x_{m-2},x_{m-3},\cdots,x_2,x_1$ are obtained iteratively. Thus this process in pseudocode is as follows:

```
(1) $x_m = \widetilde{A}_{mm}^{-1}\widetilde{f}_m$
(2) For $k=m-1$ to 1
(3) $x_k = \widetilde{A}_{kk}^{-1}(\widetilde{f}_k - A_{k,k+1}x_{k+1})$
(4) End
```

As one would expect, the most time-consuming step here is the matrix inversion. The $\widetilde{A}_{k,k}^{-1}$ is a $[(2k)^3-(2k-2)^3]\times[(2k)^3-(2k-2)^3]$ dense matrix and hence, since the cost to invert $\widetilde{A}_{k,k}$ is $\mathcal{O}(k^6)$ by methods such as the Gauss algorithm, the total cost can be as large as $\mathcal{O}(k^7)$. In order to reduce the time cost as far as possible, we adopted a recursive fast inversion scheme, a classical divide-and-conquer strategy (D&C) that recursively breaks down a large matrix into four sub-matrices until all the sub-matrix sizes become less than a threshold predefined in the direct inversion – viz.

$$M^{-1}=\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1}=\begin{bmatrix} (shurD) & -(shurD)BD^{-1} \\ -D^{-1}C(shurD) & D^{-1}+D^{-1}C(shurD)(BD^{-1}) \end{bmatrix}. \tag{2.6}$$

To further reduce the computational complexity, compressible matrices may be used in the matrix inversion [1]. An ε-rank-k matrix is defined to be such that most of its k singular values are larger than ε. If the matrix M can be partitioned into four equal pieces as in (2.6) and the off-blocks have ε-rank-k at most p, we say M is compressible – and accordingly, the matrices B and C are both compressible for certain ε. The references [14] and [15] provide an efficient way to obtain the low-rank approximation of the matrices, and [16] describes the randomized singular value decomposition algorithm for computing a low-rank approximation of a given numerically low-rank matrix. With the numerically low rank off-diagonal block B for example, we adopt the algorithm developed in [14] to construct a low-rank approximation $B\simeq U_1MU_2^T$ for given rank p. For a low-rank matrix M, the rank p is far less than the order of B, hence

$$-(shurD)BD^{-1}=-(shurD)U_1M(U_2^TD^{-1})$$

greatly reduces the number of multiplications between compressible matrices and vectors.

As the off-diagonal blocks all have low rank, the matrix multiplications in *shurD* and $D^{-1}C(shurD)(BD^{-1})$ involve only a small number of multiplications between compressible matrices and vectors. Indeed, if the ranks of the off-diagonal blocks do not grow larger than p in the recursive procedure, the computational complexity of the scheme is $\mathcal{O}(p^2k^3\log^2 k)$. In our test discussed in Section 4, we set $\varepsilon=1.0^{-10}$ for all blocks. That achieves a very high solution accuracy but leads to high rank in most of the off-diagonal blocks, so it can be a challenge to strike a balance between solution accuracy and the number of multiplications involved.

3 Direct method for solving the 3-D Poisson equation with variable coefficient

3.1 Constructing an FDF

Let us now consider the Poisson equation (1.1) and assume the variable coefficient ρ is a continuous and differentiable function, although in many physical environments there are usually only discrete tabular values for ρ. In passing, one may recall the well known conservative FDF

$$\sum_{s\in i,j,k}\Big(\rho_{s+\frac{1}{2}}u_{s+1}+\rho_{s-\frac{1}{2}}u_{s-1}-(\rho_{s+\frac{1}{2}}+\rho_{s-\frac{1}{2}})u_s\Big)=f,$$

where for brevity $u_{i+1}\equiv u_{i+1,j,k}$, $u_{i-1}\equiv u_{i-1,j,k}$, $u_i\equiv u_{i,j,k}$ and $\rho_{i+\frac{1}{2}}\equiv\rho_{i+\frac{1}{2},j,k}$, and one similarly deals with $s=j$ and $s=k$. However, we have derived a new FDF that exploits several difference operators, and although the corresponding matrix is not symmetric it has been found to preserve the efficiency and accuracy of the numerical simulations.

If ρ is constant, the Poisson equation (1.1) of course reduces to the form discussed in Section 2.1 – viz.

$$\rho\nabla^2u=f. \tag{3.1}$$

However, when ρ is not constant Eq. (1.1) may be expanded as

$$\nabla\rho\cdot\nabla u+\rho\nabla^2u=f, \tag{3.2}$$

the form from which we proceed to derive the new FDF.

The appropriate FDF for the second term on the left-hand side of Eq. (3.2) is that used in Section 2.1, when Eq. (2.1) is represented by

$$\rho_{i,j,k}\frac{\Delta^2_{0,x}+\Delta^2_{0,y}+\Delta^2_{0,z}}{h^2}u_{i,j,k}=f_{i,j,k}, \tag{3.3}$$

for an equidistant spatial meshed and cube with side length h. In proceeding to derive an FDF for the term $\nabla\rho\cdot\nabla u$, some useful symbols to denote linear finite difference operators are as follows:

(1) ε: $\varepsilon u_k=u_{k+1}$

(2) Δ: $\Delta u_k=u_{k+\frac{1}{2}}-u_{k-\frac{1}{2}}=\varepsilon^{\frac{1}{2}}-\varepsilon^{-\frac{1}{2}}$

(3) Γ: $\Gamma u_k=\frac{1}{2}\Big(u_{k+\frac{1}{2}}+u_{k-\frac{1}{2}}\Big)=\frac{1}{2}\Big(\varepsilon^{\frac{1}{2}}+\varepsilon^{-\frac{1}{2}}\Big)$

(4) Θ: $\Theta u_k=u_k$

(5) D^j: $(Dh)^ju=\frac{d^ju}{dx^j}h^j$

A Taylor expansion yields

$$\varepsilon u(x) = u(x+h) = u(x) + u'(x)h + \frac{u''}{2!}h^2 + \cdots = \sum_{j=0}^{\infty} \frac{u^j h^j}{j!},$$

such that

$$D^j u h^j = (hD)^j u = \frac{d^j u}{dx^j} h^j = u^j h^j$$

and hence $\varepsilon u(x) = e^{hD} u(x)$, so a new representation of the differential operator D is obtained – viz.

$$D = \frac{\ln \varepsilon}{h}.$$

Now ε can be represented by Δ and Θ, i.e., $\varepsilon = (\Delta/2 + \sqrt{\Theta + \Delta^2/4})^2$, so yet another new representation of D is obtained:

$$D = \frac{2}{h} \ln\left(\frac{\Delta}{2} + \sqrt{\Theta + \frac{1}{4}\Delta^2}\right). \tag{3.4}$$

Then setting $\Theta = 1$ and expanding the right-hand side by the Binomial Theorem, we have

$$D = \frac{1}{h} \sum_{j=0}^{\infty} \frac{(-1)^j}{2j+1} \binom{2j}{j} \left(\frac{\Delta}{4}\right)^{2j+1}. \tag{3.5}$$

The power of the operator Δ in each term in this infinite series is odd and cannot be used to construct mappings in R^2, so the operator Γ is now introduced. Since $4\Gamma^2 = \varepsilon + \varepsilon^{-1} + 2\Theta$ and $\Delta^2 = \varepsilon + \varepsilon^{-1} - 2\Theta$, we then obtain

$$\Theta = \Gamma\Gamma^{-1} = \Gamma\left(\Theta + \frac{1}{4}\Delta^2\right)^{-\frac{1}{2}} = \Gamma \sum_{i=0}^{\infty} (-1)^i \binom{2i}{i} \left(\frac{\Delta^2}{16}\right)^i, \tag{3.6}$$

so Θ can be represented by Γ times an even power series in Δ. If we now recall that $\Gamma\Delta u_k = (u_{k+1} - u_{k-1})/2$, the differential operator D may finally be rewritten

$$D = \Theta D = \frac{\Gamma\Delta}{h} \left[\left(\sum_{j=0}^{\infty} \frac{(-1)^j}{2j+1} \binom{2j}{j} \left(\frac{\Delta}{4}\right)^{2j}\right)\right] \left[\sum_{i=0}^{\infty} (-1)^i \binom{2i}{i} \left(\frac{\Delta^2}{16}\right)^i\right] = \frac{\Gamma\Delta}{h} + \mathcal{O}(\Delta), \tag{3.7}$$

so that when truncated at $\mathcal{O}(\Delta)$ we have a representation for D that uses the fewest nodes in the corresponding FDF. A more precise representation for D is of course always possible by adopting a higher order truncation, but that brings in more node layers and hence more calculation. Alternative and possibly better ways to provide a higher precision FDF will be discussed in Section 3.2. However, on the implicit assumption of continuity and differentiability we will now adopt

$$\begin{aligned}\frac{\partial u_{i,j,k}}{\partial x}&=\frac{1}{2h}(u_{i+1,j,k}-u_{i-1,j,k})+\mathcal{O}(h^2),\\ \frac{\partial u_{i,j,k}}{\partial y}&=\frac{1}{2h}(u_{i,j+1,k}-u_{i,j-1,k})+\mathcal{O}(h^2),\\ \frac{\partial u_{i,j,k}}{\partial z}&=\frac{1}{2h}(u_{i,j,k+1}-u_{i,j,k-1})+\mathcal{O}(h^2),\end{aligned}$$

and hence the FDF

$$\begin{aligned}\nabla\rho\cdot\nabla u=&\frac{\partial\rho}{\partial x}\frac{\partial u_{i,j,k}}{\partial x}+\frac{\partial\rho}{\partial y}\frac{\partial u_{i,j,k}}{\partial y}+\frac{\partial\rho}{\partial z}\frac{\partial u_{i,j,k}}{\partial z}\\ =&\frac{1}{2h}(u_{i+1,j,k}-u_{i-1,j,k})\frac{\partial\rho}{\partial x}+\frac{1}{2h}(u_{i,j+1,k}-u_{i,j-1,k})\frac{\partial\rho}{\partial y}\\ &+\frac{1}{2h}(u_{i,j,k+1}-u_{i,j,k-1})\frac{\partial\rho}{\partial z},\end{aligned}\tag{3.8}$$

so we have the following composite FDF for Eq. (3.2):

$$\begin{aligned}\nabla\rho\cdot\nabla u+\rho\nabla^2u=&u_{i+1,j,k}\left(\frac{\rho}{h^2}+\frac{1}{2h}\frac{\partial\rho}{\partial x}\right)+u_{i-1,j,k}\left(\frac{\rho}{h^2}-\frac{1}{2h}\frac{\partial\rho}{\partial x}\right)\\ &+u_{i,j+1,k}\left(\frac{\rho}{h^2}+\frac{1}{2h}\frac{\partial\rho}{\partial y}\right)+u_{i,j-1,k}\left(\frac{\rho}{h^2}-\frac{1}{2h}\frac{\partial\rho}{\partial y}\right)\\ &+u_{i,j,k+1}\left(\frac{\rho}{h^2}+\frac{1}{2h}\frac{\partial\rho}{\partial z}\right)+u_{i,j,k-1}\left(\frac{\rho}{h^2}-\frac{1}{2h}\frac{\partial\rho}{\partial z}\right)-\frac{6\rho u_{i,j,k}}{h^2}\\ =&f_{i,j,k}.\end{aligned}\tag{3.9}$$

The FDF (2.2) and (3.9) only differ in their coefficients, so Eq. (3.2) can be solved by merely altering the coefficients of the stiffness matrix described in Section 2.2. Thus the computational cost of the scheme using (3.9) remains $\mathcal{O}(N^{3/2}\log^2 N)$ and requires $\mathcal{O}(N\log N)$ memory to store the information used, as for (2.2). The numerical test and results are given in Section 4. Usually, evaluation of $\nabla\rho$ at the nodes is by interpolation from tabular values for ρ.

3.2 High-order compact scheme

In this subsection, we introduce a higher precision method with truncation error $\mathcal{O}(h^4)$. The spiral pattern over the entire cube and the sparsity pattern of the corresponding matrix indicates that a most suitable approach is a compact difference scheme restricted to a patch of cells adjacent to any given node in the mesh.

A high-order compact FDF has previously been obtained through a process depending on the desired approximate order [17, 18], where the leading error term in an initial central difference approximation for a particular equation is expanded continuously until the desired approximate order is reached. Thus in principle one may derive any high-order compact difference scheme, although the derivations typically involve long and tedious algebraic manipulations. The symbolic software package Mathematica has been used to help obtain several compact schemes for the three-dimensional Poisson equation, where truncated Taylor series expansions and a computer-aided tool to automatically generate high order discretization are invoked [19,21]. Inter alia, this approach has yielded an explicit fourth-order compact finite difference scheme for approximating the 3-D convection-diffusion equation

$$\Delta u(x,y,z)+(\lambda(x,y,z),\mu(x,y,z),\phi(x,y,z))\cdot\nabla u(x,y,z)=f(x,y,z),$$

which has three parameters with definite physical meaning and is very important in computational fluid dynamics to describe transport phenomena.

Now provided $\rho\neq 0$ the Poisson equation (3.2) can be converted into such a 3-D convection-diffusion form – viz.

$$\Delta u(x,y,z)+\frac{\nabla\rho}{\rho}\cdot\nabla u(x,y,z)=\frac{f(x,y,z)}{\rho}, \tag{3.10}$$

so the envisaged discretization scheme yields a 19-point formula

$$\sum_{l=0}^{18} c_l u_l = F_1,$$

where u_l denotes the value of the function $u(x,y,z)$ at the node labeled l in Fig. 4 and c_l denotes the corresponding coefficient [21]. In addition, There is also a 6th-order accurate finite difference scheme that employs the 27 points of the cubic grid [20] as shown in Fig. 5, which yields a 27-point formula that can be obtained through a method similar to that discussed in Reference [21]. The nodes that both schemes employ are located in adjacent cubes, so the corresponding matrices have the same form as shown in Fig. 3. Thus the algorithm of the direct solver described in Section 2.2 remains effective in both cases, except that the blocks $\widetilde{A}_{ii}$ and A_{ij} become density matrices and whether or not the off-diagonal blocks still have low rank depends on the coefficients.

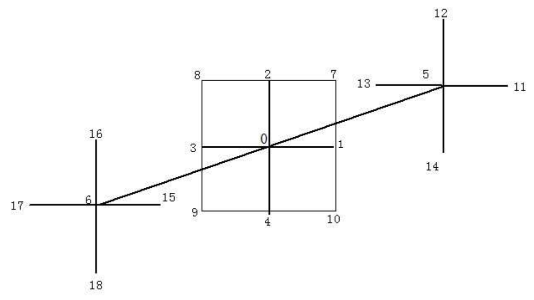

Figure 4: 19-point stencil.

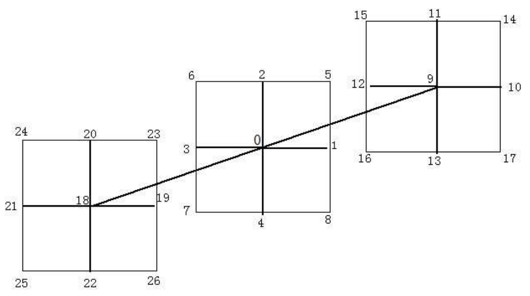

Figure 5: 27-point stencil.

4 Numerical tests and analysis

Let us now describe the results of our numerical experiments and analyze the performance of our proposed methods. All programs were written in Matlab but not optimized, and the computer used had a 3.0GHz CPU and 1GByte RAM. As described in Section 2, we implemented the direct solver using the FDF (2.2) with order of accuracy $\mathcal{O}(h^2)$. Our test case was a series of nested cubes, where the outermost cube is a fixed boundary with unit length.

Table 1: Numerical results for standard Poisson equation.

Hierarchy	N	$T_{Solve}(Sec)$	Mem(M)	$e_1(relative)$	$e_2(absolute)$
2	64×64	0.0571	1.536e-002	0	0.0504
3	216×216	0.0398	3.226e-001	4.1397e-017	0.0247
4	512×512	0.1326	1.809e+000	1.2478e-016	0.0147
5	1000×1000	0.4284	6.024e+000	1.7200e-016	0.0098
6	1728×1728	1.2473	1.518e+001	1.6615e-016	0.0070
7	2744×2744	5.4488	3.214e+001	2.3144e-016	0.0052
8	4096×4096	18.7720	6.047e+001	2.8569e-016	0.0040
9	5832×5832	62.6291	1.043e+002	3.7982e-016	0.0032
10	8000×8000	132.7445	1.687e+002	5.7511e-016	0.0026
* 20	64000×64000	55.4674	2.989e+003	-	-
* 30	216000×216000	10595.4768	-	-	-

Table 1 illustrates the numerical results of the test described in Section 2.2, for the given function $v=\sin(2\pi x)\sin(2\pi y)\sin(2\pi z)$ that describes a periodic electrostatic potential, and the periodic variable coefficient $\rho=\sin x+\sin y+\sin z$. Thus we numerically solved the elliptic partial differential equation

$$\nabla\cdot\rho\nabla u=\nabla\rho\cdot\nabla v+\rho\nabla^2 v$$

with that given variable coefficient and the boundary condition $u=v$. In the table, "Hierarchy" denotes the number of layers of nested cubes, N is the size of the matrix arising from the corresponding discretization, T_{Solve} and Mem denote the corresponding time and memory cost in constructing the matrix $\widetilde{A}_{mm}^{-1}$, and $e1$ and $e2$ are the relative and absolute errors generated by the difference formula (2.2) defined by

$$e1=\frac{||ShurA_{k,k}||-||\widetilde{A}_{k,k}||}{||\widetilde{A}_{k,k}||} \qquad \text{and} \qquad e2=\sqrt{||u-u_{real}||_2}.$$

Note that $e1$ can be used to compare the computed Schur complement against the result from an iterative solver, and $e2$ is the difference between the computed and actual values of u. In the last two tests listed in the table, the cubes were divided into 20 and 30 layers respectively, but in these two cases we did not calculate the corresponding Schur complement due to the limited memory (in order to reduce the otherwise considerable time and memory cost).

Table 2: Results of numerical test with $\rho=\sin(2\pi x)\sin(2\pi y)\sin(2\pi z)$.

Hierarchy	N	$T_{Solve}(Sec)$	$e_1(symmetry)$	$e_2(asymmetry)$	$e_3(10^{-12})$	$e_3(10^{-16})$
2	64×64	0.0470	0.0511	0.0504	2.2180e-006	2.2180e-006
3	216×216	0.2572	0.0249	0.0246	2.9738e-010	2.9738e-010
4	512×512	0.2934	0.0148	0.0146	1.0697e-012	4.8439e-015
5	1000×1000	0.7425	0.0098	0.0097	1.3160e-012	3.6814e-016
6	1728×1728	2.0181	0.0070	0.0069	1.2255e-012	4.4100e-016
7	2744×2744	5.7910	0.0052	0.0052	1.3810e-012	4.7388e-016
8	4096×4096	16.3040	0.0041	0.0040	1.4068e-012	7.0017e-016
9	5832×5832	36.7263	0.0033	0.0032	1.4102e-012	1.0201e-015
10	8000×8000	74.5259	0.0027	0.0026	1.3052e-012	1.4787e-015
11	10648×10648	142.4446	0.0022	0.0022	1.2639e-012	2.0806e-015
12	13824×13824	280.6617	0.0019	0.0019	1.2741e-012	3.3745e-015
13	17576×17576	487.4073	0.0016	0.0016	1.2578e-012	4.5457e-015
14	21952×21952	833.4443	0.0014	0.0014	1.2050e-012	1.1086e-011
15	27000×27000	1419.8219	0.0012	0.0012	1.2343e-012	1.0371e-014

Table 2 illustrates the numerical results of the test described in Section 3.1. Since the structure of the stiffness matrix arising from FDF (3.9) is the same as for the FDF (2.2), the memory required is identical so *Mem* is omitted from Table 2. In this test $e1$ denotes the error generated from the discrete formula in conservative form where the corresponding matrix is symmetric, whereas e_2 denotes the error generated from FDF (3.9) and

$$e_3=\sqrt{||u-u_{GMRES}||_2}$$

denotes the difference between the solution obtained by the proposed solver and the solution obtained by the GMRES with the tolerance $r(r=10^{-12},10^{-16})$. Our proposed direct solver evidently achieved high accuracy, to at least the same precision for the GMRES with tolerance 10^{-12}.

Table 3 shows the rank of some off-diagonal blocks arising in the matrix inversion process in the previous test, when the matrix size is 27000×27000. The results indicate that the rank of the off-diagonal blocks increases with the block size, and increasing the computational accuracy ε does not significantly reduce the ranks of the off-diagonal blocks. However, the increased rank does not match the increased size of the blocks, so the method remains effective for compressible matrices and vectors.

5 Conclusion

We have presented a direct solver for a 3-D elliptic partial differential equation, with corresponding FDF and the direct solution of linear algebraic system that arise from a cube with nodes in a spiral pattern. The proposed solver adopts fast algorithms for compressible matrices arising from the elliptic 3-D partial differential equation with variable coefficient considered. Suitable high-order compact schemes for the solver were discussed.

Table 3: Ranks of the off-diagonal blocks in numerical test with ρ.

Off-Diagonal Blocks	$\varepsilon = 1.0^{-10}$	$\varepsilon = 1.0^{-5}$
120×120	98	94
122×122	100	95
127×127	87	77
127×127	115	106
127×127	103	80
148×148	64	64
182×182	143	133
242×242	100	99
254×254	194	175
254×254	193	176
364×364	144	123
338×338	248	220
508×508	196	147
508×508	198	149
676×676	246	175

It must be admitted that some real performance gaps remain between the solution for 2-D and the 3-D partial differential equations. In particular, the complexity increases from $\mathcal{O}(n)$ to $\mathcal{O}(n^2)$ where n denotes the number of nodes; and secondly, the ranks of the off-diagonal blocks increase as the size of the blocks increases, whereas in 2-D they remain almost the same. Our proposed scheme is nevertheless quite promising, given its success on the test problem involving the 3-D Poisson equation with variable coefficient, although of course its application to further concrete problems (with associated analysis) is warranted.

Acknowledgments

This work, supported by NFS No. 11001257, was stimulated by Per-Gunnar Martinsson's paper "A Fast Direct Solver for a Class of Elliptic Partial Differential Equations". Professor Jingfang Huang suggested solving the Poisson equation with variable coefficient as a test case. We are very grateful to both of them for their selfless help.

References

[1] P.-G. Martinsson, A fast direct solver for a class of elliptic partial differential equations, J. Sci. Comput. 38 (2009), pp. 316–330.

[2] B. Engquist and L. Ying, Sweeping preconditioner for the Helmholtz equation: Moving perfectly matched layers, Multiscale Model. Sim. 9(2) (2011), 686-710.

[3] S. Chandrasekaran, M. Gu, and W. Lyons, A fast adaptive solver for hierarchically semiseparable representations, Calcolo 42(3-4) (2005), pp. 171–185.

[4] S. Chandrasekaran, M. Gu, and W. Lyons, Superfast multifrontal method for structured linear systems of equations, Private Communication (2007).

[5] L. Grasedyck, R. Kriemann, and S. Le Borne, Domain-decomposition based H-matrix preconditioners, Proceedings of DD16. LNSCE, vol. 55, Springer, Berlin (2006), pp. 661–668, pp. 471–476.

[6] P. Le Tallec, Domain decomposition methods in computational mechanical, Computational Mechanics Advances (1986), pp. 121–220.

[7] G.R. Pack, G.A. Garrett, L. Wong, and G. Lamm, The Effect of a variable dielectric coefficient and finite Ion size on Poisson-Boltzmann calculations of DNA-electrolyte systems, Biophysical J. 65 (1993), pp. 1363-1370.

[8] J. Warwicker and H.C. Watson, Calculation of the electric potential in the active site cleft due to α-helical dipoles, J. Mol. Biol. 157 (1982), pp. 671-679.

[9] B.Z. Lu and J.A. McCammon, Molecular surface-free continuum model for electrodiffusion processes, Chem. Phys. Lett. 451(4-6) 2008, pp. 282-286.

[10] R.F. Warming and B.J. Hyett, The modified equation approach to the stability and accuracy analysis of finite-difference methods, J. Comput. Phys. 14(2) (1974), pp. 159–179.

[11] W. Hackbusch, Elliptic Differential Equations Theory and Numerical Treatment, Springer-Verlag Berlin Heidelberg, Berlin, 1992, pp. 40–72. (Electronic).

[12] A. Iserles, A First Course in the Numerical Analysis of Differential Equations, Tinghua University Press, 2005, pp. 101–127. (Chinese version).

[13] D.H. Yu and H.Z. Tang, Numerical Solution of Differential Equations, Science Press, 2007, pp. 300–307. (Chinese version).

[14] E. Liberty, F. Woolfe, P. Martinsson, V. Rokhlin, and M. Tygert, Randomized algorithms for the low-rank approximation of matrices, Proc. Natl. Acad. Sci. USA 104 (2007), pp. 20167–0172.

[15] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, A fast randomized algorithm for the approximation of matrices, Appl. Comput. Harmon. Anal. 25 (2008), pp. 335–366.

[16] L. Lin, J. Lu, and L. Ying, Fast construction of hierarchical matrix representation from matrix-vector multiplication, J. Comput. Phys. 230(10) (2010), pp. 4071-4087.

[17] W.F. Spotz and G.F. Carey, A high-order compact formulation for the 3-D Poisson equation, Numer. Methods Part. Diff. Eqs. 12 (1996), pp. 235-243.

[18] W. F. Spotz and G. F. Carey, High-order compact scheme for the steady stream-function vorticity equations, Int. J. Numer. Methods Eng. 38 (1995), pp. 3497-3512.

[19] M.M. Gupta and J. Kouatchou, Symbolic derivation of finite difference approximations for three dimensional Poisson equation, Int. J. Numer. Methods Part. Diff. Eqs. 14(5) (1998), pp. 593-606.

[20] U. Ananthakrishnaiah, R. Manohar, and J. W. Stephenson, Fourth-order finite difference for three-dimensional general elliptic problems with variable coefficients, Numer. Methods Part. Diff. Eqs. 3 (1987), pp. 229-240.

[21] J. Zhang, An explicit fourth-order compact finite difference scheme for three dimensional convection-diffusion equation, Commun. Numer. Methods Eng. 14 (1997), pp. 263-280.