Journal of Advances in Mathematics and Computer Science

35(5): 121-133, 2020; Article no.JAMCS.56632

ISSN: 2456-9968

(Past name: British Journal of Mathematics & Computer Science, Past ISSN: 2231-0851)

An Optimal Algorithm for the Solution of the Helmholtz Equation

Richard O. Akinola[1*], Blessing Okwudo Ogbeh[1] and Isaac Chukle[2]

[1] *Department of Mathematics, Faculty of Natural Sciences, Universiy of Jos, Nigeria.*

[2] *Department of Statistics and Mathematics, School of Science and Technology, Plateau State Polytechnic, Barkin Ladi, Nigeria.*

Authors' contributions

This work was carried out in collaboration among all authors. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/JAMCS/2020/v35i530286

Editor(s):

(1) Dr. Zhenkun Huang, Jimei University, China.

Reviewers:

(1) S. Kalaiselvi, Anna University, India.

(2) A. C. Wimal lalith De alwis, School of Four Incalculables and Tathagatha's Society No. 299, Sri Lanka.

(3) Pasupuleti Venkata Siva Kumar, Cse Department, India.

Complete Peer review History: http://www.sdiarticle4.com/review-history/56632

Received: 19 May 2020

Accepted: 23 July 2020

Published: 12 August 2020

Original Research Article

Abstract

Aims/Objectives: The Helmholtz equation is a partial differential equation which is used in numerical weather prediction. Angwenyi *et. al.,* used a five point finite difference stencil in discretizing the partial differential equation and solved the resulting square system of equations using eight iterative methods and concluded that the BICGSTAB was the most computationally efficient using just one example. However, based on a comparison of the norm of the residual and CPU time of four methods presented in this work on the same example in their paper and others; we not only discovered that the Gauss Seidel method out performed the BICGSTAB contradicting the claim of the authors but also the Thomas Block Tridiagonal Algorithm (TBTA) in the absence of round off errors.

Methodology: We compared the performance of the Gauss Seidel Method, BICGSTAB, Matlab backslash, and the Thomas Block Tridiagonal Algorithm (TBTA) for the numerical solution of the Helmholtz equation with different step sizes.

**Corresponding author: E-mail: roakinola@gmail.com;*

Results: We discovered that in the absence of round off errors, not only did the Gauss Seidel method but also the Thomas Block Tridiagonal Algorithm (TBTA) out performed the BICGSTAB contradicting the claim of Angwenyi *et. al.*
Conclusion: We do not recommend the BICGSTAB for the solution of the linear system of equations arising from the discretization of the Helmholtz equation as claimed by Angwenyi *et al.* Rather, the Thomas Block Tridiagonal Algorithm should be used and if one is thinking of an iterative method for the numerical solution of the Helmholtz equation, the Gauss-Seidel method should be the method of choice rather than the BICGSTAB.

Keywords: BICGSTAB; Helmholtz equation; Norm of residual; CPU-time.

2010 Mathematics Subject Classification: 65F15; 15A18; 93B60.

1 Introduction

The Helmholtz equation is an equation that describes the propagation of wave through a medium. It is a Partial Differential Equation (PDE) derived from the wave equation and often times called the reduced wave equation. The Helmholtz equation has many areas of application, some of which are in: waveguides, underwater acoustics [1], numerical weather prediction, electromagnetic radiation, seismology, and cloaking [2]. According to [2], tremendous advancements have been made in the area of cloaking for flexural waves in elastic plates. It arises naturally when one is looking for mono-frequency or time-harmonic solutions to the wave equation.

Consider the 2-Dimensional wave equation

$$\frac{\partial^2 U(x,y)}{\partial t^2} = c^2 \Delta U(x,y), \tag{1.1}$$

where Δ is the 2-dimensional Laplacian operator given by $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. The equation models the propagation of a wave travelling through a given medium at a constant speed c. Assuming a separable solution $U(x,y,t) = \psi(x,y)V(t)$ which satisfies (1.1), where ψ is independent of t and V is independent of x and y. This yields

$$\psi(x,y)\frac{\partial^2 V(t)}{\partial t^2} = c^2 V(t)\Delta\psi(x,y),$$

which can be re-expressed as

$$\frac{1}{c^2 V(t)}\frac{\partial^2 V(t)}{\partial t^2} = \frac{1}{\psi(x,y)}\Delta\psi(x,y).$$

The left hand side of the above equation is a function of only t while the right hand side is a function of x and y. Therefore, for the equation to be equal, both functions will have to be equal to some constant say $-s^2$. That is;

$$\frac{1}{c^2 V(t)}\frac{\partial^2 V(t)}{\partial t^2} = \frac{1}{\psi(x,y)}\Delta\psi(x,y) = -s^2.$$

Equating the first equation to the last yields,

$$\frac{\partial^2 V(t)}{\partial t^2} + c^2 s^2 V(t) = 0. \tag{1.2}$$

The solution of (1.2) is a linear combination of sine and cosine functions with angular frequency cs and from $\frac{1}{\psi(x,y)}\Delta\psi(x,y) = -s^2$, we have

$$\Delta\psi(x,y) + s^2\psi(x,y) = 0, \tag{1.3}$$

where ψ is a vector function and the quantity $s = \frac{2\pi f}{c_0}$ is the wave number, where f is the frequency and c_0 is a reference sound speed [1]. The solution of equation (1.3) will depend on the boundary conditions associated with the problem and it is known as the 2-dimensional Helmholtz equation. The wave number s, can be real or complex. When $s = 0$, the Helmholtz equation reduces to the Laplace equation $\Delta\psi = 0$. When $s^2 < 0$, that is for s imaginary, the equation becomes the space part of the diffusion equation. Solving the wave equation (1.1) then reduces to solving equations (1.2) and (1.3).

Several methods have been proposed for the numerical solution of the Helmholtz equation see [3, 4, 5, 6, 7, 8] and the references therein. As earlier mentioned, the Helmholtz equation can be used in cloaking; which is a process of making an object invisible to electromagnetic waves. Zhang in [2] vizualised the effect of theoretical cloaking devices have on traveling waves. He did this by using the Gauss-Seidel method in finding numerical solution to a particular generalized version of the Helmholtz equation. However, no comparison was made with other direct or iterative methods. In 2015, Ricardo *et al.*[9], defined a new Cauchy integral for domains with fractal boundary illustrating its usage to study the jump and Dirichlet type boundary value problems in a fractal domain. Recently, [10] considered the exterior Dirichlet problem for the heterogeneous Helmholtz equation, i.e. the equation $\nabla.(E\nabla\psi) + s^2 n\psi = -f$ where both E and n are functions of position. They proved a new priori bounds on the solution under conditions on E, n, and the domain that ensure nontrapping of rays. They also obtained new results about the well-posedness of such problems and the resonances of acoustic transmission problems.

Besides this, Serkh and Rokhlin [11] observed that when the Helmholtz equation is solved using integral equations, the solutions can be explicitly represented by a series of known Bessel functions of noninteger order. These explicit representations lead to highly accurate and efficient numerical algorithms for the solution of the Helmholtz equation on domains with corners. In addition, Huangxin and Weifeng in [12] presented a First Order System Least Squares (FOSLS) method for the Helmholtz equation at high wave number s, which leads to a Hermitian positive definite system of equations. They also gave an error analysis to the hp-version (the mesh size h, the approximation order p,) of the FOSLS method where the dependence on h, p, s is given explicitly and numerical experiments are given to verify theoretical results.

Angwenyi *et al.* [13], solved a single example of the Helmholtz equation using eight iterative methods viz-a-viz: Gauss-Seidel Method (GS), Jacobi Iterative Method(JI), Successive Over Relaxation (SOR), Conjugate Gradient(CG), Bi-conjugate Gradient (BICG), Bi-Conjugate Gradient Stabilized (BICGSTAB) see [14, 15], Quasi-Minimal Residual(QMR), and Generalized Minimal Residual (GMRES) methods. They concluded that the BICGSTAB is the most computationally efficient. On the contrary, we show by considering other examples that this is not the case because the Gauss-Seidel method and the direct Thomas Block Tridiagonal Algorithm out-performs the BICGSTAB in all numerical examples considered.

The plan of this paper is as follows: in Section 2.1, we revisit the well known five point finite difference scheme for discretizing the Helmholtz equation. This is then followed in Section 2.4 by a presentation on the Thomas Block Tridiagonal Algorithm (TBTA) and the computational cost. Finally, results of numerical experiments are presented by means of tables in Section 3 after which we conclude the paper.

2 Materials and Methods

2.1 The finite difference method

In this section, we restate the finite difference method [16] that solves the Helmholtz equation. The finite difference method involves replacing the partial derivatives by finite difference approximations,

thus converting the partial differential equation into a finite difference equation. The method works by replacing the region over which the independent variables in the PDE are defined by a finite grid of points at which the dependent variable is approximated. The partial derivatives in the PDE at each grid point are approximated from neighbouring values by using Taylor's theorem. Approximating the PDE at each grid point results in an equation involving neighbouring points as unknowns. This gives rise to a system of equations with unknowns, the number of grid points. This system can be represented in matrix form $A\mathbf{x} = \mathbf{b}$ where A is the coefficient matrix, $\mathbf{x}$ is the column vector of the unknowns and $\mathbf{b}$ is the constant vector.

2.2 Approximating the derivatives

If ψ is a function of x and y, then

$$\begin{aligned}
\frac{\partial\psi(x,y)}{\partial x} &\approx \frac{\psi(x+h,y)-\psi(x-h,y)}{2h},\\
\frac{\partial\psi(x,y)}{\partial y} &\approx \frac{\psi(x,y+h)-\psi(x,y-h)}{2k},\\
\frac{\partial^2\psi(x,y)}{\partial x^2} &\approx \frac{\psi(x+h,y)-2\psi(x,y)+\psi(x-h,y)}{h^2},\\
\frac{\partial^2\psi(x,y)}{\partial y^2} &\approx \frac{\psi(x,y+h)-2\psi(x,y)+\psi(x,y-h)}{k^2},
\end{aligned}$$

where h and k are step sizes on the x and y axes respectively. Let i be an arbitrary point associated with the x variable and j an arbitrary point associated with the y variable. Then using subscript notation, the equations above become respectively:

$$\begin{aligned}
\frac{\partial\psi_{ij}}{\partial x} &\approx \frac{\psi_{i+1,j}-\psi_{i-1,j}}{2h},\\
\frac{\partial\psi_{ij}}{\partial y} &\approx \frac{\psi_{i,j+1}-\psi_{i,j-1}}{2k},\\
\frac{\partial^2\psi_{ij}}{\partial x^2} &\approx \frac{\psi_{i-1,j}-2\psi_{ij}+\psi_{i+1,j}}{h^2}, \qquad (2.1)\\
\frac{\partial^2\psi_{ij}}{\partial y^2} &\approx \frac{\psi_{i,j-1}-2\psi_{ij}+\psi_{i,j+1}}{k^2}. \qquad (2.2)
\end{aligned}$$

2.3 The scheme

Let $R = [a,b]\times[c,d]$ be a rectangle in $\mathbb{R}^2$. Lets consider the boundary valued problem

$$\Delta\psi + s^2\psi = 0 \quad \text{in} \quad R, \qquad \psi(x,y) = f(x,y) \quad \text{on} \quad \partial R. \qquad (2.3)$$

Approximate R with an $n\times m$ lattice of the form (x_i, y_j), where n is the number of partitions on the x- axis and m is the number of partitions on the y- axis, such that

$$x_i = a+(i-1)h, \quad h = \frac{b-a}{n-1}, \qquad \text{and} \qquad y_j = c+(j-1)k, \quad k = \frac{d-c}{m-1}.$$

For each given mesh point (x_i, y_j), approximate the Helmholtz equation using (2.1) and (2.2) to arrive at

$$\frac{\psi_{i-1,j}-2\psi_{ij}+\psi_{i+1,j}}{h^2} + \frac{\psi_{i,j-1}-2\psi_{ij}+\psi_{i,j+1}}{k^2} + s^2\psi_{ij} = 0.$$

Multiplying through by k^2, taking $r = \frac{k^2}{h^2}$, then

$$r\psi_{i-1,j} + r\psi_{i+1,j} + \psi_{i,j-1} + \psi_{i,j+1} + (s^2k^2 - 2r - 2)\psi_{ij} = 0,$$

and assuming $h = k$, it is easy to see that

$$\psi_{i-1,j} + \psi_{i+1,j} + \psi_{i,j-1} + \psi_{i,j+1} + (s^2k^2 - 4)\psi_{ij} = 0. \tag{2.4}$$

The five point formula (2.4) is the finite difference scheme for solving the Helmholtz equation (1.3) numerically. This is applied at each mesh point (x_i, y_j), for $i = 0, 1, \cdots, n$ and $j = 0, 1, \cdots, m$. Since the values of ψ are known on the boundaries, applying the method to the interior mesh points gives us a square system of $(n-2)(m-2)$ equations in $(n-2)(m-2)$ unknowns to solve. This system in the form $A\mathbf{x} = \mathbf{b}$, where A is the coefficient matrix of dimension $(n-2)(m-2) \times (n-2)(m-2)$, $\mathbf{x}$ is the column vector of the unknowns and $\mathbf{b}$ is the constant vector. Solving this system provides the numerical solution to (1.3).

In the next section, we briefly discuss the Thomas Block Tridiagonal Algorithm for the solution of the linear system obtained from discretizing the Helmholtz equation.

2.4 Thomas block Tridiagonal algorithm

A tridiagonal matrix is a band (a sparse matrix whose non-zero entries are confined to a diagonal band, compromising of the leading diagonal and zero or more diagonal(s) on either side) matrix with non-zero elements only on the leading diagonal and slots horizontally or vertically adjacent the diagonal *i. e.,* along the sub and super diagonals.

Unlike most elliptic PDE's, which have the property of their coefficient matrix being tridiagonal, the Helmholtz equation does not have this. When solving the Helmholtz equation numerically, we arrive at a pentadiagonal coefficient matrix. There is an advantage in solving the system whose coefficient matrix is tridiagonal; matrix storage is less consumed and system computational time is reduced.

The Thomas Block tridiagonal algorithm can be applied in solving the sparse linear system of equations derived from the finite difference discretization of the Helmholtz equation, to reduce the pentadiagonal matrix A to a block tridiagonal one. The five point finite difference method generates a block tridiagonal SPD coefficient matrix so that a block version of the tridiagonal algorithm can be applied.

2.5 Thomas algorithm for solving block Tridiagonal systems

We consider the problem of solving a square linear system of equations, $\mathbf{Ax} = \mathbf{b}$ in which $\mathbf{A}$ is a large sparse, partitioned into blocks, each block is of size N by N and is either diagonal or tridiagonal. In practical applications, such matrices arise from a five-point finite difference discretization of partial differential equations as in Examples 3.1, 3.2 and 3.3. This section is structured as follows, we use a block LU-type factorization in factoring $\mathbf{A}$, after which block forward and backward substitutions are used in solving for the unknown vector-$\mathbf{x}$. We present the Thomas algorithm for solving block tridiagonal systems, and the computational cost in terms of the number of floating point operations required. The material in this section can be found in [18, pp. 58-61], [19, pp. 121-122] and [17].

Let $\mathbf{A}$ be partitioned as

$$\mathbf{A} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{C}_1 & & & & \\ \mathbf{A}_2 & \mathbf{B}_2 & \mathbf{C}_2 & & & \\ & \mathbf{A}_3 & \mathbf{B}_3 & \mathbf{C}_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \mathbf{A}_{N-1} & \mathbf{B}_{N-1} & \mathbf{C}_{N-1} \\ & & & & \mathbf{A}_N & \mathbf{B}_N \end{bmatrix}, \tag{2.5}$$

where the $\mathbf{A}_k, \mathbf{B}_k, \mathbf{C}_k$'s are of size N by N. Note that the $\mathbf{A}_k, \mathbf{B}_k, \mathbf{C}_k$'s do not have to be equal. The unknown vector $\mathbf{x}$ and corresponding right hand side is partitioned as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \vdots \\ \mathbf{b}_N \end{bmatrix}, \tag{2.6}$$

where each $\mathbf{x}_k$ and $\mathbf{b}_k$ are in $\mathbb{R}^N$. We factor $\mathbf{A}$ into a block LU type factorization of the form

$$\mathbf{A} = LU = \begin{bmatrix} \boldsymbol{\Gamma}_1 & & & & & \\ \boldsymbol{\Theta}_2 & \boldsymbol{\Gamma}_2 & & & & \\ & \boldsymbol{\Theta}_3 & \boldsymbol{\Gamma}_3 & & & \\ & & \ddots & \ddots & & \\ & & & \boldsymbol{\Theta}_{N-1} & \boldsymbol{\Gamma}_{N-1} & \\ & & & & \boldsymbol{\Theta}_N & \boldsymbol{\Gamma}_N \end{bmatrix} \begin{bmatrix} \mathbf{I} & \boldsymbol{\Delta}_1 & & & & \\ & \mathbf{I} & \boldsymbol{\Delta}_2 & & & \\ & & \mathbf{I} & \boldsymbol{\Delta}_3 & & \\ & & & \ddots & \ddots & \\ & & & & \mathbf{I} & \boldsymbol{\Delta}_{N-1} \\ & & & & & \mathbf{I} \end{bmatrix}, \tag{2.7}$$

where $\mathbf{I}$ is the N by N identity matrix, $\boldsymbol{\Theta}_k$, $\boldsymbol{\Delta}_k$ and $\boldsymbol{\Gamma}_k$ are square matrices. The 'L' and 'U' factors are block bidiagonal and the above factorization is not unique. This is because, we can also factor $\mathbf{A}$ as

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & & & & & \\ \boldsymbol{\Delta}_1 & \mathbf{I} & & & & \\ & \boldsymbol{\Delta}_2 & \mathbf{I} & & & \\ & & \ddots & \ddots & & \\ & & & \boldsymbol{\Delta}_{N-2} & \mathbf{I} & \\ & & & & \boldsymbol{\Delta}_{N-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Gamma}_1 & \boldsymbol{\Theta}_2 & & & & \\ & \boldsymbol{\Gamma}_2 & \boldsymbol{\Theta}_3 & & & \\ & & \boldsymbol{\Gamma}_3 & \boldsymbol{\Theta}_4 & & \\ & & & \ddots & \ddots & \\ & & & & \boldsymbol{\Gamma}_{N-1} & \boldsymbol{\Theta}_N \\ & & & & & \boldsymbol{\Gamma}_N \end{bmatrix}.$$

After expanding the right hand side of (2.7) and comparing with the entries of $\mathbf{A}$ in (2.5) blockwisely, we obtain

$$\boldsymbol{\Theta}_k = \mathbf{A}_k, \quad \text{for} \quad k = 2, 3, \ldots, N,$$
$$\boldsymbol{\Gamma}_1 = \mathbf{B}_1, \quad \text{and} \quad \boldsymbol{\Gamma}_1 \boldsymbol{\Delta}_1 = \mathbf{C}_1,$$

and the following recurrence $\boldsymbol{\Gamma}_k \Delta_k = \mathbf{C}_k$, where

$$\boldsymbol{\Gamma}_k = \mathbf{B}_k - \mathbf{A}_k \boldsymbol{\Delta}_{k-1},$$

for $k = 2, 3, \ldots N$. We first solve for $\boldsymbol{\Delta}_k$, use the previous $\boldsymbol{\Delta}_{k-1}$ and then substitute into $\boldsymbol{\Gamma}_k = \mathbf{B}_k - \mathbf{A}_k \boldsymbol{\Delta}_{k-1}$ to get the $\boldsymbol{\Gamma}_k$'s. This completes the block LU factorization of $\mathbf{A}$. The system $\mathbf{Ax} = \mathbf{b}$ now reduces to solving

$$L\mathbf{y} = \mathbf{b}, \quad \text{and} \quad U\mathbf{x} = \mathbf{y},$$

for $\mathbf{y}$ and $\mathbf{x}$ respectively. Now, using the L factor in (2.6), we can rewrite $L\mathbf{y} = \mathbf{b}$ as

$$L\mathbf{y} = \begin{bmatrix} \boldsymbol{\Gamma}_1 & & & & & \\ \mathbf{A}_2 & \boldsymbol{\Gamma}_2 & & & & \\ & \mathbf{A}_3 & \boldsymbol{\Gamma}_3 & & & \\ & & \ddots & \ddots & & \\ & & & \mathbf{A}_{N-1} & \boldsymbol{\Gamma}_{N-1} & \\ & & & & \mathbf{A}_N & \boldsymbol{\Gamma}_N \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \\ \vdots \\ \mathbf{y}_{N-1} \\ \mathbf{y}_N \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \vdots \\ \mathbf{b}_{N-1} \\ \mathbf{b}_N \end{bmatrix}.$$

Observe that one can solve for the $\mathbf{y}_k$'s, using forward substitution, beginning with $\boldsymbol{\Gamma}_1 \mathbf{y}_1 = \mathbf{b}_1$, for $\mathbf{y}_1$. Then for $k = 2, 3, \ldots, N$, we solve for the remaining $\mathbf{y}_k$'s from the relation

$$\boldsymbol{\Gamma}_k \mathbf{y}_k = \mathbf{b}_k - \mathbf{A}_k \mathbf{y}_{k-1}.$$

We now substitute the computed values of $\mathbf{y}_k$ into $U\mathbf{x} = \mathbf{y}$, that is,

$$\begin{bmatrix} \mathbf{I} & \boldsymbol{\Delta}_1 & & & & & \\ & \mathbf{I} & \boldsymbol{\Delta}_2 & & & & \\ & & \mathbf{I} & \boldsymbol{\Delta}_3 & & & \\ & & & \ddots & \ddots & & \\ & & & & \mathbf{I} & \boldsymbol{\Delta}_{N-1} & \\ & & & & & \mathbf{I} & \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_{N-1} \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \\ \vdots \\ \mathbf{y}_{N-1} \\ \mathbf{y}_N \end{bmatrix}.$$

It is easily seen from the last row above that $\mathbf{x}_N = \mathbf{y}_N$. Using backward substitution, we obtain the remaining $\mathbf{x}_k$'s from the recurrence relation

$$\mathbf{x}_k = \mathbf{y}_k - \boldsymbol{\Delta}_k \mathbf{x}_{k+1},$$

for $k = N-1, N-2, \ldots, 2, 1$. The above theory now leads to the following algorithm: Thomas Algorithm (see, for example, [18] and [19]).

Input $\mathbf{A}$, $\mathbf{b}$ and N.
Set $\Gamma_1 = \mathbf{B}_1$.
Solve $\boldsymbol{\Gamma}_1 \boldsymbol{\Delta}_1 = \mathbf{C}_1$ for $\boldsymbol{\Delta}_1$.
Solve $\boldsymbol{\Gamma}_1 \mathbf{y}_1 = \mathbf{b}_1$ for $\mathbf{y}_1$.
For $k = 2, 3 \ldots, N$
  Compute $\boldsymbol{\Gamma}_k = \mathbf{B}_k - \mathbf{A}_k \boldsymbol{\Delta}_{k-1}$.
  Solve $\boldsymbol{\Gamma}_k \boldsymbol{\Delta}_k = \mathbf{C}_k$ for $\boldsymbol{\Delta}_k$.
  Solve $\boldsymbol{\Gamma}_k \mathbf{y}_k = \mathbf{b}_k - \mathbf{A}_k \mathbf{y}_{k-1}$, for $\mathbf{y}_k$.
Set $\mathbf{x}_N = \mathbf{y}_N$.
For $k = N-1 : -1 : 1$
  Calculate $\mathbf{x}_k = \mathbf{y}_k - \boldsymbol{\Delta}_k \mathbf{x}_{k+1}$.

Next, we briefly describe the operation counts for the Thomas Algorithm. Note that since $\mathbf{A}$ has been partitioned into N blocks in N unknowns, this means that there are $n = N^2$ unknown vectors. Observe that the solution of $\mathbf{Ax} = \mathbf{b}$ by LU factorization costs $O(N^2)^3$ operations. Since the product of two N by N matrices requires N^3 operations [18], this implies that forward substitution which involves $(N-1)$ matrix-matrix multiplication at a cost of $2N^3$ operations and approximately $2(N^2)^2$ total operations. Similarly, there are $(N-1)$ LU factorizations at $\frac{2}{3}N^3$ operations which amounts to approximately $O(N^2)^2$ operations. Moreover, N^2 triangular solves are required at a cost of N^2 operations and $\sim O(N^2)^2$ floating point operations. Therefore, the total floating point operations required for Thomas block tridiagonal algorithm is approximately $O(N^2)^2$ operations. Hence, in solving $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is large and sparse, Thomas algorithm requires less number of operations and storage than the direct LU.

In the example studied by [13], the domain is a unit square with grid spacing $h = k = \frac{1}{n+1}$, therefore there are $N = n^2 = 9$ unknowns, and ψ is zero on the boundary.

To write this problem using matrix notation, we must select an order for the unknowns ψ_{ij}. The classical order is to start from the bottom grid row and then move up to the next grid row. So the above problem with $N = 9$ unknowns, the 9×1 column vector for the unknowns is given by

$$\mathbf{x} = [\psi_{11}, \psi_{21}, \psi_{31}, \psi_{12}, \psi_{22}, \psi_{32}, \psi_{13}, \psi_{23}, \psi_{33}]^T.$$

The coefficient matrix can be written down by considering each of the above equations. Each row in the matrix represents the coefficients in the equation at the (i, j)th grid, for example, the first row are the coefficients of the equation at the $(1, 1)$ grid. We arrive at a system of the form:

$$\begin{bmatrix} a & b & & e & & & & & \\ c & a & b & & e & & & & \\ & c & a & & & e & & & \\ d & & & a & b & & e & & \\ & d & & c & a & b & & e & \\ & & d & & c & a & & & e \\ & & & d & & & a & b & \\ & & & & d & & c & a & b \\ & & & & & d & & c & a \end{bmatrix} \begin{bmatrix} \psi_{11} \\ \psi_{21} \\ \psi_{31} \\ \psi_{12} \\ \psi_{22} \\ \psi_{32} \\ \psi_{13} \\ \psi_{23} \\ \psi_{33} \end{bmatrix} = \begin{bmatrix} F_{11} \\ F_{21} \\ F_{31} \\ F_{12} \\ F_{22} \\ F_{32} \\ F_{13} \\ F_{23} \\ F_{33} \end{bmatrix}.$$

This can be written compactly by grouping the grid rows into three; 3×1 vectors viz:

$$\mathbf{x}_1 = [\psi_{11}, \psi_{21}, \psi_{31}]^T, \quad \mathbf{x}_2 = [\psi_{12}, \psi_{22}, \psi_{32}]^T, \quad \mathbf{x}_3 = [\psi_{13}, \psi_{23}, \psi_{33}]^T.$$

In the same vein,

$$\mathbf{b}_1 = [F_{11}, F_{21}, F_{31}]^T, \quad \mathbf{b}_2 = [F_{12}, F_{22}, F_{32}]^T, \quad \mathbf{b}_3 = [F_{13}, F_{23}, F_{33}]^T.$$

The 9×9 linear system above then becomes a 3×3 block with each block a 3×3 matrix such that

$$\begin{bmatrix} \mathbf{B}_1 & \mathbf{C}_1 & \\ \mathbf{A}_2 & \mathbf{B}_2 & \mathbf{C}_2 \\ & \mathbf{A}_3 & \mathbf{B}_3 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix},$$

where $\mathbf{A}_2 = \mathbf{A}_3 = dI$, $\mathbf{C}_1 = \mathbf{C}_2 = eI$,

$$\mathbf{B}_1 = \mathbf{B}_2 = \mathbf{B}_3 = \begin{bmatrix} a & b & \\ c & a & b \\ & c & a \end{bmatrix}, \quad \text{and} \quad I = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}.$$

It is common for finite difference schemes in two space dimension to generate block tridiagonal matrices.

3 Results and Discussion

In this section, we discretize some partial differential equations using the five point stencil of the finite difference method in some defined domain. After the discretization, we obtain a linear system of equations. We then compared the solution of the linear system of equations using the Gauss-Seidel, BICGSTAB, Matlab Backslash and Thomas Block Tridiagonal Algorithm viz-a-viz their norms of residual, the number of iterations it takes for convergence and CPU runtime to ascertain which is the most computationally efficient. Throughout this section, we solved three examples numerically each with step sizes $h = k = 0.25$, $h = k = 0.1$ and $h = k = 0.05$ each corresponding to linear systems of $9 \times 9, 81 \times 81$ and 400×400 respectively.

Example 3.1.

We seek the numerical solution of the Helmholtz equation

$$\Delta\psi(x, y) + s^2\psi(x, y) = \Phi(x, y). \tag{3.1}$$

with boundary conditions $\psi(x, y) = \ln[(x + 1)^2 + y^2], \Phi(x, y) = \frac{1}{4}(x + y)$, $h = k = 0.25$ and $s = 1$. Let the problem be solved on the unit square, $0 \leq x \leq 1, 0 \leq y \leq 1$. This example was considered

in [13], but we solve this same problem using the above step size and later reduce it to $h = k = 0.1$ and $h = k = 0.05$. First we notice that this problem is a non-homogeneous case of the Helmholtz equation where the right hand side of the equation is not the zero vector. We apply the same procedure used in deriving the resulting linear system of equations in Section 2.3. But in this case, take note of the non-zero right hand side. With $h = k = 0.25$, we arrive at the finite difference scheme:

$$\psi_{i-1,j} + \psi_{i+1,j} + \psi_{i,j-1} + \psi_{i,j+1} + (s^2k^2 - 4)\psi_{i,j} = h^2\Phi(x_i, y_j).$$

But $\Phi(x, y) = \frac{1}{4}(x + y)$, therefore

$$(4 - s^2k^2)\psi_{i,j} - \psi_{i-1,j} - \psi_{i+1,j} - \psi_{i,j-1} - \psi_{i,j+1} = -\frac{h^2}{4}(x_i + y_j).$$

After substituting $h = k = 0.25$ and $s = 1$,

$$3.9375\psi_{i,j} - \psi_{i-1,j} - \psi_{i+1,j} - \psi_{i,j-1} - \psi_{i,j+1} = -0.0156(x_i + y_j).$$

The equation above is the finite difference scheme used in solving (3.1).

For want of space, if we let $a = 3.9375$, then

$$\begin{bmatrix} a & -1 & & -1 & & & & & \\ -1 & a & -1 & & -1 & & & & \\ & -1 & a & 0 & & -1 & & & \\ -1 & & 0 & a & -1 & & -1 & & \\ & -1 & & -1 & a & -1 & & -1 & \\ & & -1 & & -1 & a & 0 & & -1 \\ & & & -1 & & 0 & a & -1 & \\ & & & & -1 & & -1 & a & -1 \\ & & & & & -1 & & -1 & a \end{bmatrix} \begin{bmatrix} \psi_{11} \\ \psi_{21} \\ \psi_{31} \\ \psi_{12} \\ \psi_{22} \\ \psi_{32} \\ \psi_{13} \\ \psi_{23} \\ \psi_{33} \end{bmatrix} = \begin{bmatrix} 0.4991 \\ 0.7992 \\ 2.5053 \\ 0.2114 \\ -0.0156 \\ 1.4274 \\ 1.3715 \\ 1.1591 \\ 2.8962 \end{bmatrix},$$

where the blank spaces in the above coefficient matrix are zero entries. The above linear system is pentadiagonal which makes the Thomas Block Tridiagonal Algorithm also suitable. As stated earlier, we want to be sure that the conclusion reached by Angwenyi *et al.,* [13] is true by reducing the step size to $h = k = 0.1$. Hence, increasing the size of the linear system of equations from 9 by 9 above to 81 by 81. In addition, we reduced the step size to $h = k = 0.05$ and obtained a 400 by 400 system of equations. The results are as tabulated in Table 1.

Table 1. A comparison of the performance of the Thomas Block Tridiagonal Algorithm, Gauss Seidel, Matlab Backslash and BICGSTAB on Example 3.1.

Size of matrix	Thomas Block	Gauss Seidel	Matlab Backslash	BICGSTAB
Norm of Residual				
9 × 9	3.47e-16	3.85e-16	3.97e-16	4.45e-16
81 × 81	1.16e-15	1.07e-15	1.46e-15	1.81e-09
400 × 400	3.82e-14	4.06e-14	2.14e-14	2.76e-01
CPU TIME × 1000				
9 × 9	8.0000	0.0000	20.0001	4.0001
81 × 81	4.0001	0.0000	1028.1001	4.0001
400 × 400	36.0000	16.0000	9502.8000	52.0000

As shown in Table 1 for the 9 by 9 example, a ranking of the norm of the residual from the smallest to the largest shows that the Thomas Block Tridiagonal Algorithm (3.47×10^{-16}) gave the least followed by the Gauss Seidel method (3.85×10^{-16}) while the BICGSTAB (4.45×10^{-16}) method was the largest in the absence of round off errors. Since this is the same example considered by Angwenyi

et al. to arrive at their conclusion, this is a contradiction. In the same vein, as shown by the 81 by 81 example (row four of the table), it can be seen that the Gauss Seidel method had the smallest norm of residual (1.07×10^{-15}); followed by the Thomas Block Tridiagonal Algorithm (1.16×10^{-15}) while the acclaimed computationally efficient iterative method (albeit the BICGSTAB) had the largest (1.81×10^{-9}). In the 400 by 400 example, we see that Matlab Backslash had the smallest norm of residual (2.076×10^{-14}) which is followed by the TBTA (3.82×10^{-14}), Gauss-Seidel (4.06×10^{-14}) and BICGSTAB (2.76×10^{-1}). Judging by the large norm of residual, this example shows how vague their conclusions is because the BICGSTAB did not adequately solve the large system.

Next, using CPU time as a yardstick for comparison, we observed the following respectively for the $N=9, N=81$ and $N=400$: 8, 4.0001 and 36 (TBTA), 0, 0, 16 (Gauss-Seidel) and 4.0001, 4.0001 and 52 (BICGSTAB). As the size of the system increases, the CPU time increased drastically for Matlab Backslash. Hence, the use of Matlab Backslash is an academic exercise and should be discouraged for the numerical solution of the Helmholtz equation. Therefore, the TBTA, Gauss Seidel methods out-performs the BICGSTAB method both in CPU Time and norm of residual.

To further disclaim the claim of Angwenyi *et al.'s,* paper that the BICGSTAB is the most computationally efficient iterative method for solving the Helmholtz equation, we considered two more examples from a different author.

Example 3.2.

We solve [2]

$$\Delta\psi+\psi=0 \quad \text{in} \quad R=[0,1]\times[0,1], \quad h=k=0.25 \tag{3.2}$$
$$\text{with boundary conditions} \quad \psi(x,y)=\sin\left[\frac{\pi x}{6}\right].$$

Since $h=k$, we discretized the above PDE using the five point finite difference scheme (2.4). That is;

$$(4-s^2k^2)\psi_{i,j}-\psi_{i-1,j}-\psi_{i+1,j}-\psi_{i,j-1}-\psi_{i,j+1}=0. \tag{3.3}$$

But since $s=1$ and $h=k=0.25$, therefore

$$3.9375\psi_{i,j}-\psi_{i-1,j}-\psi_{i+1,j}-\psi_{i,j-1}-\psi_{i,j+1}=0. \tag{3.4}$$

This reduces to

$$\begin{bmatrix}
3.9375 & -1 & & -1 & & & & & \\
-1 & 3.9375 & -1 & & -1 & & & & \\
 & -1 & 3.9375 & & & -1 & & & \\
-1 & & & 3.9375 & -1 & & -1 & & \\
 & -1 & & -1 & 3.9375 & -1 & & -1 & \\
 & & -1 & & -1 & 3.9375 & & & -1 \\
 & & & -1 & & & 3.9375 & -1 & \\
 & & & & -1 & & -1 & 3.9375 & -1 \\
 & & & & & -1 & & -1 & 3.9375
\end{bmatrix}
\begin{bmatrix} \psi_{11} \\ \psi_{21} \\ \psi_{31} \\ \psi_{12} \\ \psi_{22} \\ \psi_{32} \\ \psi_{13} \\ \psi_{23} \\ \psi_{33} \end{bmatrix}
=
\begin{bmatrix} 0.1305 \\ 0.2588 \\ 0.8827 \\ 0 \\ 0 \\ 0.5 \\ 0.1305 \\ 0.2588 \\ 0.8827 \end{bmatrix}.$$

In the above linear system, the blank spaces in the coefficient matrix A are zeroes. Similarly, we reduced the step size in this example to $h=k=0.1$ and after imposing the necessary boundary conditions, we have a linear system of 81 equations in 81 unknowns. In addition, after reducing the step size to $h=k=0.05$, a 400 by 400 system is obtained. We solved the systems numerically using the four methods and the results are as tabulated in Table 2. The table shows that the four methods perform in the same fashion as those in Table 1. and the explanation that follows.

Example 3.3.

Consider finding the numerical solution of the following PDE [2]

$$\begin{cases} \Delta\psi+\psi=0 \quad \text{in R}, \\ \psi(x,y)=-\frac{1}{x^2+y^2} \quad \text{on} \quad \partial R, \end{cases} \qquad R=[0,1]\times[0,1], \quad \text{with} \quad h=k=0.25. \tag{3.5}$$

Table 2. A comparison of the performance of the Thomas Block Tridiagonal Algorithm, Gauss Seidel, Matlab Backslash and BICGSTAB on Example 3.2.

Size of matrix	Thomas Block	Gauss Seidel	Matlab Backslash	BICGSTAB
Norm of Residual				
9 × 9	1.10e-15	1.70e-15	3.90e-16	2.21e-15
81 × 81	5.40e-15	4.29e-15	3.41e-15	1.03e-08
400 × 400	1.18e-14	1.24e-14	5.84e-15	9.78e-02
CPU TIME × 1000				
9 × 9	8.0010	0.0000	20.0020	4.0001
81 × 81	4.0001	0.0000	1100.1001	4.0001
400 × 400	36.0000	12.0000	9382.0000	56.0000

The above PDE is the same as that in Example 2, except for the different boundary conditions. Hence, we obtained the same coefficient matrix but with a different right hand side

$$\mathbf{b} = [-32, -4, -2.719, -4, 0, -0.8, -2.719, -0.8, -1.28]^T .$$

Next, we reduced the step size to $h = k = 0.1$ and $h = k = 0.05$ and after imposing necessary boundary conditions we obtained respectively linear systems of sizes: 81 by 81 and 400 by 400.

The results of solving the corresponding linear systems using the above four mentioned methods is tabulated in Table 3.

Table 3. A comparison of the performance of the Thomas Block Tridiagonal Algorithm, Gauss Seidel, Matlab Backslash and BICGSTAB on Example 3.3.

Size of matrix	Thomas Block	Gauss Seidel	Matlab Backslash	BICGSTAB
Norm of Residual				
9 × 9	1.14e-14	4.94e-15	5.49e-15	9.94e-15
81 × 81	2.65e-12	2.59e-12	8.50e-13	5.09e-03
400 × 400	3.09e-13	2.24e-13	1.44e-13	1.93e+00
CPU TIME × 1000				
9 × 9	8.0001	0.0000	20.0001	4.0001
81 × 81	6.0001	0.0000	1104.1001	4.0001
400 × 400	24.0000	16.0000	9494.8000	56.0000

Besides the Matlab Backslash which was for academic purpose albeit it takes a long CPU time, for the (81 by 81 and 400 by 400) systems, Table 3. clearly shows that both TBTA (2.65×10^{-12} and 3.09×10^{-13}) and Gauss-Seidel method (2.59×10^{-12} and 2.24×10^{-13}) out-performs the BICGSTAB (5.09×10^{-3} and 1.93×10^{00}).

4 Conclusions

Since practical problems give rise to much bigger matrix systems, and from the results in this work, it is unsafe to recommend the BICGSTAB for the solution of the linear system of equations arising from the discretization of the Helmholtz equation as claimed by Angwenyi *et al.* Rather, the Thomas Block Tridiagonal Algorithm should be used and if one is thinking of an iterative method for the numerical solution of the Helmholtz equation, the Gauss-Seidel method should be the method of choice rather than the BICGSTAB.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Bayliss A, Goldstein CI, Turkel E. The numerical solution of the Helmholtz equation for wave propagation problems in underwater acoustics. Comp. & Maths. with Appls; 1985;11(7/8):655-665.

[2] Li Zhang. The finite difference method for the Helmholtz equation with applications to cloaking, International Journal of Applied Mathematical Sciences (JAMS). 2013;6:1-3.

[3] Hesteness MR, Stiefel E. Methods of conjugate gradients for solving linear system. 1952;49:409-436.

[4] Concus P, Golub GH. Use of fast direct methods for efficient numerical solution of non separable elliptic equation; 1973.

[5] Proskurowski W, Olof W. On the numerical solution of the Helmholtz's equation by the capacitance matrix method. Mathematics of Computation. 1976;30(135):433-436.

[6] Howard E. Elementary matrix theory (re printed Ed.). New York: Dover; 37.
ISBN 0-486-63946-0; 1980

[7] Goldstein CI. A finite element method for solving Helmholtz type equations in waveguides and other unbounded domains; 1982.

[8] Goldstein CI. The numerical solution of underwater acoustics propagation problems using finite difference and finite element methods; 1984.

[9] Ricardo AB, Juan BR., Rodrguez-Dagnino RM. Boundary value problems for hyperholomorphic solutions of two dimensional Helmholtz equation in a fractal domain. Applied Mathematics and Computation Journal. 2015;261:183-191.

[10] Graham I, Pembery O, Spence E. The Helmholtz equation in heterogeneous media: A priori bounds, well-posedness, and resonances. Journal of Differential Equations. 2019;266(6):2869-2923.
Available:https://doi.org/10.1016/j.jde.2018.08.048

[11] Serkh K, Rokhlin V. On the solution of the Helmholtz equation on regions with corners. Proceedings of the National Academy of Sciences. 2016;113(33):9171-9176.
DOI: 10.1073/pnas.1609578113

[12] Huangxin C, Weifeng Q. A first order system least squares method for the Helmholtz equation. Journal of Computational and Applied Mathematics. 2017;309:145-162.

[13] Angwenyi ND, Lawi G, Ojiema M, Owino M. On the computationally efficient numerical solution to the Helmholtz equation. International Mathematical Forum. 2014;9(6):259-266.

[14] Gerard LGS, Deiderik KF. BiCGSTAB(l) for linear equations involving unsymmetric matrices with complex spectrum, ETNA Kent state University. 1993;1:11-32.

[15] Van der Vorst HA BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of non-symmetric linear systems. SIAM J. SCI and Stat. Computation. 1992;13(2):631-644.

[16] Smith GD. Numerical solution of partial differential equations: Finite difference methods. Oxford University; 1985;67-68(3rd Ed.).

[17] Block Thomas Algorithm.
Available:www4.nscu.edu/eos/users/w/white/www/white/ma580/chap2.5.pdf

[18] Isaacson E, Keller HB. Analysis of numerical methods. John Wiley & Sons, Inc., London; 1996.

[19] Engeln-Mullges G, Uhlig F. Numerical Algorithms with C. Springer; 1996.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://www.sdiarticle4.com/review-history/56632