Iterative Solvers for the Heat Conduction Equation

Method · February 2020

DOI: 10.13140/RG.2.2.21705.49765

1 author:

Douglas Vinson Nance
Air Force Research Laboratory
22 PUBLICATIONS **70** CITATIONS

SEE PROFILE

Computational Physics Notes

Iterative Solvers for the Heat Conduction Equation

Douglas V. Nance

February 2020

Iterative Solvers for the Heat Conduction Equation

Douglas V. Nance
Independent Research Scientist

February 2020

DISCLAIMER

The author of this report has used his best effort in preparing this document. The author makes no warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed or represents that its use would not infringe privately owned rights. Reference herein to any commercial products, processes, trade name, trademark, manufacturer or otherwise, does not necessarily constitute or imply its recommendation, or favoring, by the author.

Abstract

This report addresses an implicit scheme for the Heat Conduction equation and the linear system solver routines required to compute the numerical solution for this equation at each time step. The Crank-Nicolson implicit difference scheme is the canonical scheme that serves as the test bed for the Gauss-Seidel, standard Conjugate Gradient, Three Terms Recurrence Conjugate Gradient and the Biconjugate Gradient iterative methods. Exact solutions are constructed for three test problems and used to validate numerical solutions. The test problems are constructed in 1D, 2D and 3D with uniform grids. Non-uniform grids are also constructed for the 1D and 2D problems to generate non-symmetric matrices for testing with the solvers. Numerical solutions are shown graphically in comparison with exact solutions, and direct calculations of error are performed and tabulated. Some comments are made concerning the background and motivation for this effort.

Contents

1 Introduction

This report discusses numerical solution techniques for the Heat Equation, a parabolic partial differential equation (PDE) found in various sectors of applied mathematics and physics. In addition, this PDE is well known, and frequently used for didactic purposes.[1] A reason underlying its utility is that this PDE incorporates time variation with the steady nature of the Laplace operator. It is also a classic equation used in researching numerical techniques.[2] That application is of primary interest in this report.

1.1 Background

In 2010, I completed a technical report on numerical solutions of the heat equation via the finite volume (FV) method.[3] That work departs from the more standard finite difference (FD) technique. The FV discretization method is usually applied for a different set of applications, namely fluid dynamics. As it happens, the old report now resides on the ResearchGate website, so I was very surprised (and pleasingly so) to see how many times this report has been downloaded over many months. The level of interest exhibited by the research community in this topic is both encouraging and appreciated. To be a bit more specific, my motivation for using the finite volume approach is the desire to combine a heat equation solver with a computational fluid dynamics (CFD) code. Why? It is desirable to capture the heat transfer into an aerodynamic body under the heating caused by hypersonic flight. The CFD solver is discretized via the finite volume technique, and it makes sense to apply the same discretization procedure to the heat equation. It makes for easier interface coding. All in all, this effort was successful, and I tested both explicit and implicit discretization methods. For the implicit discretization, I chose the Crank-Nicolson method cast in three dimensions. To simplify the linear system solution procedure, I applied the Alternating Direction Implicit (ADI) method to integrate each space direction separately. Of course, the Thomas algorithm is employed in each of the three directions as a tridiagonal matrix solver.[2] Unfortunately, the accuracy of the ADI numerical solution suffered in comparison with the explicit time integration scheme. I remain unhappy with this result.

Nearly ten years have passed since I have had time to work on this problem and resolve my personal dissatisfaction with the loss of solution accuracy associated with the ADI method. My angst is one of the reasons for the present work. Another reason is my interest in developing numerical solutions for the time dependent Schrödinger Equation and other computational physics problems. Although this equation is cast in the complex field, the equation is a parabolic PDE and may be solved by a similar set of numerical techniques.[4] Of course, no solution approach, either analytical or numerical, is likely to be successful unless the requisite techniques are well understood and competently coded. The notes that follow document the author's thoughts on this problem.

1.2 The Heat Equation

The heat equation (HE) is classified as a parabolic PDE.[1] The model HE used in this report is written as

$$\frac{\partial u}{\partial t} = \nabla^2 u, \;\; u = u(\vec{x}, t), \;\; t > 0, \;\; \vec{x} \in D \tag{1}$$

while the Laplacian in three Cartesian dimensions is given as

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

The quantity u is a temperature-like or internal energy-like quantity, and t is the time-like coordinate while $\vec{x}$ is the position vector in the domain D. To create a solvable system, boundary conditions are required. For example, we can enforce Dirichlet conditions by specifying u on the boundary of D (∂D).[5] On the other hand, if we specify $\partial u/\partial n$ on the boundary, where n is the coordinate normal to the boundary, we have described Von Neumann conditions. To complete the problem, we must also specify initial conditions, i.e., prescribe u at all space points in the domain at t equal zero. There are other forms of the HE that incorporate properties like thermal diffusivity; these forms are still similar to Equation (1). It is interesting to note that the structure of the HE is not the same if $-t$ is substituted in place of t. This fact indicates that the conduction of heat, or more generally, a diffusion, is an irreversible process.[6] Therefore, it is not time reversible and mathematically discerns between the past and the future. This PDE also satisifies a maximum principle on a bounded domain in space-like and time-like coordinates.

The Heat Equation is analytically solvable for many problems in one, two and three dimensions. Common tools used for obtaining analytical solutions are separation of variables and Fourier series.[7] A set of test problems for the heat equation have been selected and analytically solved in Section 3 of this report. These exact solutions are developed to provide validation data for numerical solutions described later. The numerical solutions are computed by using the Crank-Nicolson discretization method, but the Alternating Direction Implicit (ADI) is not used in this work. Instead, the full implicit matrix resulting from the Crank-Nicolson method is solved by using iterative solution techniques. In one dimension, all three diagonals for the implicit matrix are included in the iterative solution as is typical for the direct solution. However, for problems in two and three dimensions, the implicit matrices have five and seven diagonals, respectively. In this work, all of the diagonals are included in the iterative solution. Moreover, iterative schemes are investigated here in lieu of direct solution methods such as LU-Decomposition in the interest of increasing computational speed, conserving storage and decreasing error in the linear solution procedure.[8]

The heat conduction problems of interest in this manuscript have non-trivial boundary conditions. For this type of problem, a uniform temperature (or internal energy) distribution is specified as the initial condition. The variation or unsteadiness in the field is provided by non-zero boundary conditions. The first two test problems have asymmetric boundary conditions while the final test case may be characterized as a three-dimensional immersion

problem. In this case a body, originally at a uniform temperature, is immersed in a large reservoir set at a different uniform temperature. This scenario is "cartoon-like" because the reservior does not respond to heat transfer.

In the sections of the report that follow, different iterative solution methods for linear systems are described. The exposition is kept as simple as possible for pedagogical purposes. At least, that is the intent; the author's success or failure in this endeavor is to be judged by the reader. This report can be mapped out as follows. Section 2 describes the numerical methods that are applied including the Crank-Nicolson discretization procedure, the Gauss-Seidel linear system solver, the standard Conjugate Gradient and Three Terms Recurrence Conjugate Gradient methods. Also, the Biconjugate Gradient method is discussed for non-symmetric linear systems. Section 3 presents the test problems complete with analytical solution procedures. The focus of this work is to examine the accuracy of the implicit method and to compare the iterative performance of the different linear solution schemes. Finally, Section 4 provides computational results and an overall analysis of the performance and accuracy of our numerical schemes. Section 5 contains an exposition of the conclusions drawn from the results of the calculations. Moreover, some recommendations are made regarding which numerical schemes show greater promise for solving the Time-Dependent Schrödinger equation.

2 Methods

In this section, the numerical methods used in solving the heat equation are discussed. In deference to my earlier work, I concentrate on implicit discretization procedures in order to (i) maximize a permissible discrete time step and (ii) increase solution accuracy. The performance of numerical schemes in item (ii) is of particular interest.

2.1 The Crank-Nicolson Implicit Discretization

Crank and Nicolson derived a long-lasting, well documented discretization method for the Heat Equation.[2] The Finite Difference Method is employed by using Newton divided differences.[10] For the purpose of simplicity, assume that the solvable field is represented by a uniform mesh with fixed spatial and temporal step sizes Δx and Δt, respectively. For simplicity, we may consider the PDE cast in one Cartesian dimension, i.e.,

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \quad u = u(x,t) \tag{2}$$

The space term occurring in (2) is discretized by using Newton divided differences as follows.

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} + O(\Delta x^2) \tag{3}$$

The subscript "i" is indicative of the field point in space at x_i where

$$x_i = i\Delta x, \;\; i = 1, 2, \ldots \tag{4}$$

We assume that the space field starts at $x = 0$. It follows that

$$u_i = u(x_i) \tag{5}$$

Note that in (3), we have discretized only in space. Time has not yet been considered. To provide a basis for comparison, consider an explicit scheme; let the time-like variable t be written as

$$t = n\Delta t, \;\; n = 1, 2, \ldots \tag{6}$$

In the notation above, n is the index chosen for the time level. It is represented as a superscript in the expression below, i.e.,

$$\left.\frac{\partial u}{\partial t}\right|_i^n = \frac{u_i^{n+1} - u_i^n}{\Delta t} + O(\Delta t) \tag{7}$$

If we substitute (7) and (3) into (2) without the error specifiers, we obtain a difference equation. Specifically,

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \tag{8}$$

Note that the space terms in (8) are now evaluated at time level n, the present time. A simple rearrangement of (8) yields the explicit Euler difference equation wherein

$$u_i^{n+1} = u_i^n + r\left(u_{i+1}^{n*} - 2u_i^{n*} + u_{i-1}^{n*}\right) \tag{9}$$

while $r = \Delta t/\Delta x^2$ is denoted as the mesh ratio.[11] Equation (9) is really an elementary difference equation, a point where the understanding of this topic really begins. This form is easy to code and play with, yet it has a number of disadvantages such as numerical instability (for certain mesh ratios), an issue we do not explore here. Instead, we now delve into an implicit form. Let us represent the terms at time n tagged with a * in (9) as an average of the present and future time levels.[2] For example,

$$u_i^{n*} = \frac{1}{2}\left(u_i^{n+1} + u_i^n\right) \tag{10}$$

With this substitution in (9), we obtain

$$u_i^{n+1} = u_i^n + r\left[\frac{1}{2}\left(u_{i+1}^{n+1} + u_{i+1}^n\right) - 2\cdot\frac{1}{2}\left(u_i^{n+1} + u_i^n\right) + \frac{1}{2}\left(u_{i-1}^{n+1} + u_{i-1}^n\right)\right] \tag{11}$$

By sequestering the advanced time level $(n+1)$ terms on the left of the equality, we obtain

$$u_i^{n+1} - \frac{r}{2}u_{i+1}^{n+1} + ru_i^{n+1} - \frac{r}{2}u_{i-1}^{n+1} = u_i^n + \frac{r}{2}u_{i+1}^n - ru_i^n + \frac{r}{2}u_{i-1}^n \tag{12}$$

By multiplying by two and collecting terms, we arrive at the Crank-Nicolson equation in one dimension.[2] Moreover,

$$\begin{aligned} -ru_{i+1}^{n+1} + 2(1+r)u_i^{n+1} - ru_{i-1}^{n+1} = ru_{i+1}^n + 2(1-r)u_i^n + ru_{i-1}^n, \\ 2 \le i \le \text{imax} - 1, \quad n = 1, 2, \ldots \end{aligned} \tag{13}$$

The limits on index i are as shown because boundary values are located at i = 1 and imax. Equation (13) is the classic Crank-Nicolson difference formula. An examination of its convergence properties is beyond the scope of this report, but interested readers are referred to Mitchell et al. and to Smith.[2, 11] Still, it important to state that the Crank-Nicolson formula is stable for $r > 0$. That having been written, we will further qualify this result in the results section of this report.

The Crank-Nicolson difference formula is readily generalizable to both two and three dimensions for uniform meshes.[2] For two Cartesian dimensions, the heat equation is written as

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{14}$$

By recalling (3) and including a fixed index j for the y coordinate held constant for the partial derivative in x, we can similarly note that

$$\left.\frac{\partial^2 u}{\partial y^2}\right|_j = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} + O(\Delta y^2) \tag{15}$$

the difference equation for two dimensions may be written as

$$\begin{aligned} -r_y u_{i,j-1}^{n+1} - r_x u_{i-1,j}^{n+1} + 2(1 + r_x + r_y)u_{i,j}^{n+1} - r_x u_{i+1,j}^{n+1} - r_y u_{i,j+1}^{n+1} \\ = r_y u_{i,j-1}^{n} + r_x u_{i-1,j}^{n} + 2(1 - r_x - r_y)u_{i,j}^{n} + r_x u_{i+1,j}^{n} + r_y u_{i,j+1}^{n}, \end{aligned} \tag{16}$$

$$2 \leq i \leq \text{imax} - 1, \quad 2 \leq j \leq \text{jmax} - 1 \tag{17}$$

In (16), there are two Cartesian mesh ratios, i.e.,

$$r_x = \frac{\Delta t}{\Delta x^2}; \quad r_y = \frac{\Delta t}{\Delta y^2} \tag{18}$$

This difference equation is stable for r_x and r_y greater than zero. For completeness, it is worthwhile to present the difference equation for our heat conduction equation in three dimensions. It is no surprise that the PDE is written as

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \tag{19}$$

The stencil or computational molecule for the 1D model (9) comprises three space nodes while that for the 2D model (16) utilizes five space nodes and has a cruciform configuration. The stencil for the 3D model involves seven space nodes in what might be referred to as a diamond configuration. The Crank-Nicolson difference formula for (19) is written as

$$\begin{aligned} -r_z u_{i,j,k-1}^{n+1} - r_y u_{i,j-1,k}^{n+1} - r_x u_{i-1,j,k}^{n+1} + 2(1 + r_x + r_y + r_z)u_{i,j,k}^{n+1} - r_x u_{i+1,j,k}^{n+1} \\ - r_y u_{i,j+1,k}^{n+1} - r_z u_{i,j,k+1}^{n+1} \\ = r_z u_{i,j,k-1}^{n} + r_y u_{i,j-1,k}^{n} + r_x u_{i-1,j,k}^{n} + 2(1 - r_x - r_y - r_z)u_{i,j,k}^{n} + r_x u_{i+1,j,k}^{n} \\ + r_y u_{i,j+1,k}^{n} + r_z u_{i,j,k+1}^{n}, \end{aligned} \tag{20}$$

for $2 \leq i \leq \text{imax} - 1$, $2 \leq j \leq \text{jmax} - 1$ and $2 \leq k \leq \text{kmax} - 1$. It follows from this discussion that each of the implicit matrices is sparse containing few non-zero entries. The only non-zero entries exist along matrix diagonals. The 1D problem's matrix is tridiagonal and can be easily solved through the use of the Thomas algorithm.[11] The 2D test problem matrix is not quite so easy to solve since it has a pentadiagonal structure (five diagonals). An easy counting exercise shows that the 3D test case entails seven diagonals forming a heptadiagonal matrix. This matrix structure is particularly advantageous since only the coefficients of the difference equation need to be stored. Also, one may observe that the largest coefficients exist on the main diagonal for each matrix lending to a measure of diagonal dominance that helps in the numerical solution.[8]

2.2 Discretized Forms on Non-Uniform Grids

This section of the report is motivated by a desire to test our linear system solver numerics on non-symmetric matrices. If you look at the difference equations developed in the previous subsection, you will notice that the resulting matrices are symmetric. As it happens, if

we cast difference equations for a non-uniform grid, then the matrix symmetry disappears. The occurrence of the non-uniform grid and hence a lack of matrix symmetry adds another measure of realism to the problem.

Let us begin by examining the 1D system for a non-uniform grid. The impact of non-uniformity on this system is that the distance between grid nodes or points is not constant and can vary through the extent of the grid. The 1D stencil for the second partial derivative of u with respect to x may be written as

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i = a\, u_{i-1} + b\, u_i + c\, u_{i+1} \tag{21}$$

Difference coefficients a, b and c may be dervied by using the same procedure applied for the uniform grid except the changing spatial step size must be tracked. The Taylor series used for this purpose may be written as

$$u_{i+1} = u_i + (\Delta_i^+ x)u_i' + \frac{1}{2}(\Delta_i^+ x)^2 u_i'' + \frac{1}{3!}(\Delta_i^+ x)^3 u_i''' + \frac{1}{4!}(\Delta_i^+ x)^4 u_i^{(4)} + \ldots \tag{22}$$

$$u_{i-1} = u_i - (\Delta_i^- x)u_i' + \frac{1}{2}(\Delta_i^- x)^2 u_i'' - \frac{1}{3!}(\Delta_i^- x)^3 u_i''' + \frac{1}{4!}(\Delta_i^- x)^4 u_i^{(4)} + \ldots \tag{23}$$

where

$$\Delta_i^+ x = x_{i+1} - x_i; \quad \Delta_i^- x = x_i - x_{i-1}$$
$$\Delta_i^c x = x_{i+1} - x_{i-1}$$

By substituting (22) and (23) into (21), one obtains the following system of equations that must solved to obtain a, b and c.

$$a + b + c = 0 \tag{24}$$

$$a\,(\Delta_i^+ x) - c\,(\Delta_i^- x) = 0 \tag{25}$$

$$a\,(\Delta_i^+ x)^2 - c\,(\Delta_i^- x)^2 = 2 \tag{26}$$

With a bit of algebra, it can be shown that

$$a = \frac{2}{(\Delta_i^+ x)(\Delta_i^c x)} \tag{27}$$

$$b = \frac{-2}{(\Delta_i^+ x)(\Delta_i^- x)} \tag{28}$$

$$c = \frac{2}{(\Delta_i^- x)(\Delta_i^c x)} \tag{29}$$

Equations (24), (25) and (26) insure that the partial derivative is represented to the third order. On a uniform grid, the same stencil would yield a higher order of accuracy, but a

higher price is required to achieve the same order on a non-uniform grid. These coefficients can be applied to create a Crank-Nicolson implicit difference formula for the non-uniform mesh. The mechanics of the derivation are essentially the same as those shown earlier for the uniform grid. Here, we simple state the result.

$$-r_{i+1}u_{i+1}^{n+1} + (1+r_i)u_i^{n+1} - r_{i-1}u_{i-1}^{n+1} = r_{i+1}u_{i+1}^{n} + (1-r_i)u_i^{n} + r_{i-1}u_{i-1}^{n} \tag{30}$$

for i= 2, 3, ..., imax-1 where

$$r_{i+1} = \frac{\Delta t}{(\Delta_i^+ x)(\Delta_i^c x)} \tag{31}$$

$$r_i = \frac{\Delta t}{(\Delta_i^+ x)(\Delta_i^- x)} \tag{32}$$

$$r_{i-1} = \frac{\Delta t}{(\Delta_i^- x)(\Delta_i^c x)} \tag{33}$$

By using the same procedure, a Crank-Nicolson formula can be produced for 2D problems on the non-uniform mesh. This formula may be written as

$$\begin{aligned} -r_{j-1}u_{i,j-1}^{n+1} - r_{i-1}u_{i-1,j}^{n+1} + (1+r_i+r_j)u_{i,j}^{n+1} - r_{i+1}u_{i+1,j}^{n+1} - r_{j+1}u_{i,j+1}^{n+1} \\ = r_{j-1}u_{i,j-1}^{n} + r_{i-1}u_{i-1,j}^{n} + (1-r_i-r_j)u_{i,j}^{n} + r_{i+1}u_{i+1,j}^{n} + r_{j+1}u_{i,j+1}^{n} \end{aligned} \tag{34}$$

where, in addition to (31 - 33), the mesh ratios in the y direction are

$$r_{j+1} = \frac{\Delta t}{(\Delta_j^+ y)(\Delta_j^c y)} \tag{35}$$

$$r_j = \frac{\Delta t}{(\Delta_j^+ y)(\Delta_j^- y)} \tag{36}$$

$$r_{j-1} = \frac{\Delta t}{(\Delta_j^- y)(\Delta_j^c y)} \tag{37}$$

while the y mesh increments are

$$\Delta_j^+ x = y_{j+1} - y_j; \quad \Delta_j^- y = y_j - y_{j-1}$$
$$\Delta_j^c x = y_{j+1} - y_{j-1}$$

2.3 The Gauss-Seidel Iterative Solver for Linear Systems

Gauss-Seidel iteration is a prominent method for solving a system of linear equations.[10] This system is commonly written as

$$A\vec{x} = \vec{b} \tag{38}$$

A is an $M \times M$ square matrix of coefficients while $\vec{x}$ is the $M \times 1$ vector of unknowns. The $M \times 1$ vector $\vec{b}$ consists of known quantities. This system is easily adapted for the heat conduction equation in one, two or three Cartesian dimensions. For purposes of illustration, consider heat conduction in one dimension (the x coordinate with index i), and let imax = 5. For this simple problem, $i = 1$ and 5 correspond to known boundary values (assuming Dirichlet boundaries [1]), so the solvable field is indexed by $i = 2, 3, 4$. As mentioned earlier, heat conduction is an initial boundary value problem, so u_i^n is known for the previous time level n at $i = 1, 2, 3, 4, 5$. For a uniform grid, this system may be written as follows:

$$\begin{bmatrix} 2(1+r) & -r & 0 \\ -r & 2(1+r) & -r \\ 0 & -r & 2(1+r) \end{bmatrix} \times \begin{bmatrix} u_2^{n+1} \\ u_3^{n+1} \\ u_4^{n+1} \end{bmatrix} = \begin{bmatrix} \text{RHS}_2 + ru_1 \\ \text{RHS}_3 \\ \text{RHS}_4 + ru_5 \end{bmatrix} \tag{39}$$

The entries on the right hand side of the above equation ($\text{RHS}_{2,3,4}$) are easily computed from the right hand side of (13), i.e.,

$$\begin{aligned} \text{RHS}_2 &= ru_1 + 2(1-r)u_2^n + ru_3^n \\ \text{RHS}_3 &= ru_2^n + 2(1-r)u_3^n + ru_4^n \\ \text{RHS}_4 &= ru_3^n + 2(1-r)u_4^n + ru_5 \end{aligned} \tag{40}$$

Entries u_1 and u_5 in (39) and (40) are fixed, unchanging boundary values, so the temporal index n has been dropped from them. The remaining u_i^n are known at time level n, so we may elicit the time $n+1$ values u_i^{n+1} by solving this linear system. The 1D system is tridiagonal and easily solved by the Thomas algorithm [11], but 2D and 3D have more diagonals and cannot be solved without greater complication. There are direct solution algorithms such as Gaussian Elimination and LU Decomposition, but for large systems, these algorithms can get expensive and may admit significant computational errors.[10] For this reason, iterative schemes are examined here. The first algorithm of interest is Gauss-Seidel iteration.

Gauss-Seidel iteration is a scheme that is well suited for symmetric, positive definite matrices.[8] In fact, for these matrices, it converges for any estimate of the solution vector $\vec{x}$. In some instances, it also works for matrices that are nearly symmetric. Still, an important requirement is that all main diagonal entries must be non-zero. To illustrate Gauss-Seidel iteration, we split the coefficient matrix into the sum of lower, main and upper diagonal matrices.[12] Observe that

$$\begin{bmatrix} 2(1+r) & -r & 0 \\ -r & 2(1+r) & -r \\ 0 & -r & 2(1+r) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ -r & 0 & 0 \\ 0 & -r & 0 \end{bmatrix} + \begin{bmatrix} 2(1+r) & 0 & 0 \\ 0 & 2(1+r) & 0 \\ 0 & 0 & 2(1+r) \end{bmatrix} + \begin{bmatrix} 0 & -r & 0 \\ 0 & 0 & -r \\ 0 & 0 & 0 \end{bmatrix} \tag{41}$$

This splitting method lends to a convenient method for updating elements of the solution vector in time. The lower diagonal matrix invokes the use of time level $n+1$ entries as they become available. The upper diagonal matrix rolls time level n elements into the computation. For the 1D heat equation, the Gauss-Seidel scheme takes the following form.[10]

$$\left(\begin{bmatrix} 0 & 0 & 0 \\ -r & 0 & 0 \\ 0 & -r & 0 \end{bmatrix} + \begin{bmatrix} 2(1+r) & 0 & 0 \\ 0 & 2(1+r) & 0 \\ 0 & 0 & 2(1+r) \end{bmatrix} + \begin{bmatrix} 0 & -r & 0 \\ 0 & 0 & -r \\ 0 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} \text{RHS}_2 \\ \text{RHS}_3 \\ \text{RHS}_4 \end{bmatrix} \tag{42}$$

In the above equation, boundary values have been absorbed into $\text{RHS}_{2,4}$. By solving for the main diagonal's matrix and assigning the time order for updating $u_{2,3,4}$, we have that

$$\begin{aligned} 2(1+r) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}^{n+1} &= \begin{bmatrix} \text{RHS}_2 \\ \text{RHS}_3 \\ \text{RHS}_4 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ -r & 0 & 0 \\ 0 & -r & 0 \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}^{n+1} \\ &\quad - \begin{bmatrix} 0 & -r & 0 \\ 0 & 0 & -r \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}^{n} \end{aligned} \tag{43}$$

The Gauss-Seidel equations for this elementary system are given by solving for u_i^{n+1} and simplifying. The end result is

$$\begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}^{n+1} = \frac{1}{2(1+r)} \begin{bmatrix} \text{RHS}_2 + ru_3^n \\ \text{RHS}_3 + ru_2^{n+1} + ru_4^n \\ \text{RHS}_4 + ru_3^{n+1} \end{bmatrix} \tag{44}$$

As one can see, this system of equations allows the use of newly computed values of u_i^{n+1} as soon as they become available. If more grid points are added to the system, then it is only necessary to add one more Gauss-Seidel equation to the system for each grid point. A full forward iteration of the system entails evaluating the three equations in (44) in top to bottom order.

2.4 Standard Conjugate Gradient Method - Theoretical Aspects

Gauss-Seidel and related iterative solution methods rely upon properties of the coefficient matrix for the linear system being solved. Solutions involving symmetric, positive definite matrices converge from any starting solution. Conjugate Gradient methods, on the other hand, operate by minimizing a quadratic functional.[8, 9] There are several different manifestations of the conjugate method. For purposes of illustration, consider minimization of the functional below.[13]

$$f(\vec{x}) = \frac{1}{2}(\vec{r}, A^{-1}\vec{r}) \tag{45}$$

where $\vec{r}$ is the residual for the linear system, i.e..

$$\vec{r} = A\vec{x} - \vec{b} \tag{46}$$

We can conduct the minimization by representing (45) with indicial notation. Let $\vec{x} = x_i$. Then consider the term on the right side of the inner product in (45).

$$A^{-1}\vec{r} = A^{-1}(A\vec{x} - \vec{b}) = A_{li}^{-1}(A_{lm}x_m - b_l) \tag{47}$$

By expanding this product, we see that

$$A^{-1}\vec{r} = A_{li}^{-1}A_{lm}x_m - A_{li}^{-1}b_l \tag{48}$$

If we exploit a property of the matrix inverse, we obtain

$$A^{-1}\vec{r} = \delta_{im}x_m - A_{li}^{-1}b_l = x_i - A_{li}^{-1}b_l \tag{49}$$

With this substitution, our quadratic functional becomes

$$f(\vec{x}) = \frac{1}{2}(A_{ij}x_j - b_i)(x_i - A_{il}^{-1}b_l) \tag{50}$$

By expanding this product and simplifying, we obtain

$$f(\vec{x}) = \frac{1}{2}(A_{ij}x_jx_i - 2b_ix_i + A_{il}^{-1}b_ib_l) \tag{51}$$

The next step in the minimization procedure is to compute the gradient of the functional, i.e.,

$$\nabla f(\vec{x}) = \frac{\partial f(\vec{x})}{\partial x_k} = \frac{1}{2}\left[A_{ij}\left(\frac{\partial x_j}{\partial x_k}x_i + x_j\frac{\partial x_i}{\partial x_k}\right) - 2b_i\frac{\partial x_i}{\partial x_k}\right] \tag{52}$$

Note that the last term in (51) is independent of x_k; thus, it does not contribute to the gradient. Further simplification yields

$$\nabla f(\vec{x}) = \frac{1}{2}\left[A_{ij}(\delta_{jk}x_i + x_j\delta_{ik}) - 2b_i\delta_{ik}\right] \tag{53}$$

and

$$\nabla f(\vec{x}) = \frac{1}{2}\left[A_{ik}x_i + A_{ki}x_i - 2b_k\right] \tag{54}$$

Recall that matrix A is symmetric, so by applying symmetry to the first term and reindexing the second term, we obtain

$$\nabla f(\vec{x}) = \frac{1}{2}\left[A_{ki}x_i + A_{ki}x_i - 2b_k\right] \tag{55}$$

The end result is

$$\nabla f(\vec{x}) = A_{ki}x_i - b_k \tag{56}$$

A minimum value for $f(\vec{x})$, if it exists, occurs at a point $\vec{x}$ where the gradient vanishes.[8, 13] Hence,

$$\nabla f(\vec{x}) = A_{ki}x_i - b_k = 0 \tag{57}$$

Therefore, at the critical value

$$A\vec{x} = \vec{b} \tag{58}$$

The solution for the linear system clearly defines a critical value, but does it constitute a minimum? We can apply the second derivative test to the prior result to determine the concavity of the functional at the critical value. We compute the Hessian matrix [14], i.e.,

$$\frac{\partial^2 f(\vec{x})}{\partial x_k} = A_{ki}\frac{\partial x_i}{\partial x_l} = A_{ki}\delta_{il} = A_{kl} \tag{59}$$

What we have shown is that the matrix of second partial derivatives (or Hessian) of the functional is the same as the matrix A. At the critical value, the determinant of the Hessian yields the concavity.[15] This determinant is the same as the determinant of A. Recall that A is a positive definite matrix and that the determinant of a positive definite matrix is always positive.[12] Therefore,

$$\left|\frac{\partial^2 f(\vec{x})}{\partial x_k}\right| = |A| > 0. \tag{60}$$

so (58) minimizes the functional. We have shown that the solution of the linear system minimizes the functional; now, all that is needed is to prescribe an algorithm for the minimization elucidating the Conjugate Gradient procedure.

The aspect of the Conjugate Gradient class of methods that I find very interesting is that the minimization (or solution) procedure is conducted in a special vector space denoted as a Krylov space.[13, 16, 17] This algorithm entails structuring a succession of search directions that are closely related to the residual vector. We can elucidate a procedure for this method by considering details of the minimization of the functional $f(\vec{x})$. In the remainder of this section, we follow the exposition of Axelsson[13] very closely since it is an excellent reference. At each iteration, we increment the argument of $f(\vec{x})$ in proportion to a vector search direction $\vec{d}$, i.e., at the k^{th} iteration,

$$\text{Minimize } f(\vec{x}^{\,k} + \tau_k \vec{d}^{\,k}), \quad \text{for } \tau_k \text{ finite} \tag{61}$$

In the iteration, the new estimate of the minimizing value of $\vec{x}$ is denoted as

$$\vec{x}^{\,k+1} = \vec{x}^{\,k} + \tau_k\, \vec{d}^{\,k} \tag{62}$$

The associated residual vector is

$$\vec{r}^{\,k} = A\vec{x}^{\,k} - \vec{b} \tag{63}$$

Consider the argument used in deriving the gradient as a function of τ. Let

$$h(\tau) = f(\vec{x} + \tau\vec{d}) - f(\vec{x}) \tag{64}$$

By substituting (63) and simplifying, we find that

$$h(\tau) = \tau(\vec{r}^{\,k}, \vec{d}^{\,k}) + \frac{1}{2}\tau^2(\vec{d}^{\,k}, A\vec{d}^{\,k}) \tag{65}$$

By differentiating this functional, we can determine τ for critical values.

$$h'(\tau) = (\vec{r}^{\,k}, \vec{d}^{\,k}) + \tau(\vec{d}^{\,k}, A\vec{d}^{\,k}) = 0 \tag{66}$$

The critical value is revealed as $\tau = \tau_k$;

$$\tau_k = -\frac{(\vec{r}^{\,k}, \vec{d}^{\,k})}{(\vec{d}^{\,k}, A\vec{d}^{\,k})} \tag{67}$$

Now we use (63) to transform (62) into the corresponding residual equation by first multiplying by A and then subtracting $\vec{b}$.

$$\vec{r}^{\,k+1} = \vec{r}^{\,k} + \tau_k A\vec{d}^{\,k} \tag{68}$$

The inner product formed with $\vec{d}^{\,k}$ yields a core result, i.e.,

$$(\vec{r}^{\,k+1}, \vec{d}^{\,k}) = (\vec{r}^{\,k}, \vec{d}^{\,k}) + \tau_k(A\vec{d}^{\,k}, \vec{d}^{\,k}) \tag{69}$$

After substituting (67) into (69), we have that

$$(\vec{r}^{\,k+1}, \vec{d}^{\,k}) = 0 \tag{70}$$

showing that the new residual is orthogonal to the preceding search direction. That result is a key property associated with the standard Conjugate Gradient method.[13]

Since we have a new value of $\vec{x}$ and a new residual $\vec{d}$, we are ready for the next iteration. To perform the new iteration, a new search direction is needed. It is denoted as $\vec{d}^{\,k+1}$, and we require that it be *conjugately* orthogonal to previous search directions. Therefore, we restrict $\vec{d}^{\,k+1}$ so that

$$(\vec{d}^{\,k+1}, A\vec{d}^{\,k}) = 0 \tag{71}$$

The sequence of search directions forms a Krylov Space denoted V_k; specifically,

$$V_k = \text{Span}(\vec{d}^{\,0}, \vec{d}^{\,1}, \ldots, \vec{d}^{\,k}) \tag{72}$$

In addition, mutual orthogonality exists between the current residual and preceding search directions; that is,

$$(\vec{r}^{\,k}, \vec{d}^{\,j}) = 0 \tag{73}$$

where j takes on values $0 \leq j \leq k-1$ for k strictly greater than one. It is desirable to inductively demonstrate (73) for a unit increase in k. Assume (73), and form the inner product of (69) with $\vec{d}^{\,j}$.

$$(\vec{r}^{\,k+1}, \vec{d}^{\,j}) = (\vec{r}^{\,k}, \vec{d}^{\,j}) + \tau_k (A\vec{d}^{\,k}, \vec{d}^{\,j}) \tag{74}$$

Suppose that $j = k-1$ and evaluate (74).

$$(\vec{r}^{\,k+1}, \vec{d}^{\,k-1}) = (\vec{r}^{\,k}, \vec{d}^{\,k-1}) + \tau_k (A\vec{d}^{\,k}, \vec{d}^{\,k-1}) \tag{75}$$

The first term on the right of (75) is zero by (73) while the second term is zero by (71), conjugate orthogonality of the search directions. So in this case, the assertion is shown. Now suppose that $j = k$.

$$(\vec{r}^{\,k+1}, \vec{d}^{\,k}) = (\vec{r}^{\,k}, \vec{d}^{\,k}) + \tau_k (A\vec{d}^{\,k}, \vec{d}^{\,k}) \tag{76}$$

For this case, it is necessary to apply (67). On substitution in (76), the desired inner product vanishes, so we can conclude that

$$(\vec{r}^{\,k+1}, \vec{d}^{\,j}) = 0, \quad 0 \leq j \leq k \tag{77}$$

which inductively demonstrates the desired result.[13] Now let us consider the selection of a new search direction. Given the orthogonality of the residuals to the search directions, let us write the new search direction as

$$\vec{d}^{\,k+1} = -\vec{r}^{\,k+1} + \beta_k \vec{d}^{\,k}, \quad k = 0, 1, 2, \ldots \tag{78}$$

In the above equation, the β_k are, in this work, real, yet undetermined, coefficients. Apply the conjugate orthogonality assumption to (78); it also acts as an induction hypothesis. That is

$$(\vec{d}^{\,k+1}, A\vec{d}^{\,k}) = 0 \tag{79}$$

Take the inner product of (78) with $\vec{d}^{\,k}$.

$$(\vec{d}^{\,k+1}, A\vec{d}^{\,k}) = -(\vec{r}^{\,k+1}, A\vec{d}^{\,k}) + \beta_k (\vec{d}^{\,k}, A\vec{d}^{\,k}) \tag{80}$$

Because of (79), the left side of (80) vanishes, and we may easily solve for β_k, i.e.,

$$\beta_k = \frac{(\vec{r}^{\,k+1}, A\vec{d}^{\,k})}{(\vec{d}^{\,k}, A\vec{d}^{\,k})} \tag{81}$$

In a similar manner, we can form an expression using (78) and the search direction $\vec{d}^{\,j}$ for $j \leq k-1$. We obtain

$$(\vec{d}^{\,k+1}, A\vec{d}^{\,j}) = -(\vec{r}^{\,k+1}, A\vec{d}^{\,j}) + \beta_k (\vec{d}^{\,k}, A\vec{d}^{\,j}) \tag{82}$$

Because of the way we select search directions, we know that $(\vec{d}^{\,k}, A\vec{d}^{\,j}) = 0$ for $0 \leq j \leq k-1$, so

$$(\vec{d}^{k+1}, A\vec{d}^{j}) = -(\vec{r}^{k+1}, A\vec{d}^{j}) \tag{83}$$

To evaluate (83), it is instructive to examine V_k, the Krylov space of search directions. Recall that

$$V_k = \text{Span}\{\vec{d}^{0}, \vec{d}^{1}, \ldots, \vec{d}^{k}\} \tag{84}$$

The Conjugate Gradient method allows some latitude in selecting the initial search direction, so it advantageous to choose $\vec{d}^{0} = -\vec{r}^{0}$.[13] As is specified by (78), a linear combination, we can also define the Krylov space as

$$V_k = \text{Span}\{\vec{r}^{0}, \vec{r}^{1}, \ldots, \vec{r}^{k}\} \tag{85}$$

Since $j \leq k-1$, the linear combination $A\vec{d}^{j}$ certainly falls within V_k, yet $\vec{r}^{k+1}$ is linearly independent of V_k, so

$$(\vec{r}^{k+1}, A\vec{d}^{j}) = 0, \quad 0 \leq j \leq k-1 \tag{86}$$

and we can conclude from (83) that

$$(\vec{d}^{k+1}, A\vec{d}^{j}) = 0, \quad 0 \leq j \leq k \tag{87}$$

which inductively demonstrates conjugate orthogonality of the sequence of search directions. Admittedly, the analysis associated with this topic is a bit more difficult; you might say tedious.

2.5 Standard Conjugate Gradient Method - Algorithm

The theory behind the standard Conjugate Gradient method (presented in part here) is somewhat lengthy and is presented in pieces and parts. This fact lends to the complicated nature of the theory, but the implementation of this method is remarkably simple. For that reason, it is important to derive a few additional, yet key, results. Then it is necessary to present the algorithm in a straightforward manner. Again, we follow Axelsson[13]; this reference is the lighthouse that guides my study of this method while Golub [8] is also a reasonably good resource. Recall the slope factors τ_k and β_k occurring in the solution/residual and search direction update formulas.

$$\tau_k = -\frac{(\vec{r}^{k}, \vec{d}^{k})}{(\vec{d}^{k}, A\vec{d}^{k})} \tag{88}$$

$$\beta_k = \frac{(\vec{r}^{k+1}, A\vec{d}^{k})}{(\vec{d}^{k}, A\vec{d}^{k})} \tag{89}$$

The numerator of (88) can be rewritten into a more convenient form. This inner product is reworked as follows. We begin by substituting for $\vec{d}^{k}$ from (78) evaluated for k instead of $k+1$.

$$-(\vec{r}^k, \vec{d}^k) = -(\vec{r}^k, -\vec{r}^k + \beta_k \vec{d}^{k-1}) \tag{90}$$

Exploiting linearity of the inner product, we have that

$$-(\vec{r}^k, \vec{d}^k) = -(\vec{r}^k, -\vec{r}^k) - \beta_k(\vec{r}^k, \vec{d}^{k-1}) \tag{91}$$

The second inner product on the right hand side is zero by (77). Hence,

$$-(\vec{r}^k, \vec{d}^k) = (\vec{r}^k, \vec{r}^k) \tag{92}$$

Although it is not to be used in the formula for τ_k, we need an alternative form for the inner product in the denominator of both τ_k and β_k. We proceed by substituting for $A\vec{d}^k$ from (69).

$$(\vec{d}^k, A\vec{d}^k) = \left(\vec{d}^k, \frac{1}{\tau_k}\left(\vec{r}^{k+1} - \vec{r}^k\right)\right) \tag{93}$$

Again, we apply linearity to obtain

$$(\vec{d}^k, A\vec{d}^k) = \frac{1}{\tau_k}(\vec{d}^k, \vec{r}^{k+1}) - \frac{1}{\tau_k}(\vec{d}^k, \vec{r}^k) \tag{94}$$

The first term on the right hand side of this equation is zero by (77), so

$$(\vec{d}^k, A\vec{d}^k) = -\frac{1}{\tau_k}(\vec{d}^k, \vec{r}^k) = \frac{1}{\tau_k}(\vec{r}^k, \vec{r}^k) \tag{95}$$

where (92) has been applied. Now consider the inner product in the numerator of β. As earlier, we substitute for $A\vec{d}^k$. Observe that

$$(\vec{r}^{k+1}, A\vec{d}^k) = \left(\vec{r}^{k+1}, \frac{1}{\tau_k}\left(\vec{r}^{k+1} - \vec{r}^k\right)\right) \tag{96}$$

By using linearity again,

$$(\vec{r}^{k+1}, A\vec{d}^k) = \frac{1}{\tau_k}(\vec{r}^{k+1}, \vec{r}^{k+1}) - \frac{1}{\tau}(\vec{r}^{k+1}, \vec{r}^k) \tag{97}$$

The second term on the right hand side of this equation is zero due to orthogonality of the residual vectors, so

$$(\vec{r}^{k+1}, A\vec{d}^k) = \frac{1}{\tau_k}(\vec{r}^{k+1}, \vec{r}^{k+1}) \tag{98}$$

With the use of results of (92),(95) and (98), we can rewrite τ_k and β_k as

$$\tau_k = \frac{(\vec{r}^k, \vec{r}^k)}{(\vec{d}^k, A\vec{d}^k)} \tag{99}$$

$$\beta_k = \frac{(\vec{r}^{k+1}, \vec{r}^{k+1})}{(\vec{r}^k, \vec{r}^k)} \tag{100}$$

With the advent of (99) and (100), we can present the standard Conjugate Gradient algorithm.

Step (1): Set the initial values for the residual and for the search direction, i.e.,

$$\vec{d}^{\,0} = -\vec{r}^{\,0} \tag{101}$$

$$\vec{r}^{\,0} = A\vec{x}^{\,0} - \vec{b} \tag{102}$$

Step (2): Compute τ_k with (99).
Step (3): Compute the updated solution vector, i.e.,

$$\vec{x}^{\,k+1} = \vec{x}^{\,k} + \tau_k \vec{d}^{\,k} \tag{103}$$

Step (4): Compute the updated residual vector, i.e.,

$$\vec{r}^{\,k+1} = \vec{r}^{\,k} + \tau_k A\vec{d}^{\,k} \tag{104}$$

Step (5): Compute β_k with (100).
Step (6): Compute the updated search direction vector, i.e.,

$$\vec{d}^{\,k+1} = -\vec{r}^{\,k+1} + \beta_k \vec{d}^{\,k} \tag{105}$$

Step (7): Check the level of convergence by examining the change in $\vec{x}^{\,k}$ and $\vec{r}^{\,k}$. There are different ways to accomplish this step.
Step (8): If the level of convergence is insufficient, update k as $k+1$ and go to Step (2). Otherwise, terminate the loop.

2.6 Three-Term Recurrence Conjugate Gradient Method - Theoretical Aspects

An alternate form of the Conjugate Gradient method is due to Lanczos and was developed in the early 1950s.[18] For the sake of comparison and simply to see how well it works, I have selected this three-term recurrence form for testing here.[13] We still solve the same linear system $A\vec{x} = \vec{b}$. To derive this method, we again follow Axelsson and begin by defining a slightly different inner product. That is, let $\vec{v}$ and $\vec{w}$ be two vectors in $\Re^n$, and let $W \in \mathrm{M}(n \times n)$ is a symmetric, positive definite matrix. Then the inner product is defined as

$$(\vec{v}, \vec{w}) = \vec{v}^{\,T} W \vec{w} \tag{106}$$

To begin the solution process, let $\vec{x}^{\,0}$ be an initial solution vector, e.g., an initial conditions vector for the heat conduction problem. The estimate for the vector is computed as[13]

$$\vec{x}^{\,1} = \vec{x}^{\,0} - \beta_0 \vec{r}^{\,0} \tag{107}$$

multiplication from the left by A and then subtracting $\vec{b}$ on both sides of the above equation leads to an equation for the up to date residual.

$$A\vec{x}^{\,1} - \vec{b} = A\vec{x}^{\,0} - \vec{b} - \beta_0 A\vec{r}^{\,0} \tag{108}$$

Hence,

$$\vec{r}^{\,1} = \vec{r}^{\,0} - \beta_0 A\vec{r}^{\,0} \tag{109}$$

By taking the inner product with $\vec{r}^{\,0}$, we have that

$$(\vec{r}^{\,0}, \vec{r}^{\,1}) = (\vec{r}^{\,0}, \vec{r}^{\,0}) - \beta_0(\vec{r}^{\,0}, A\vec{r}^{\,0}) \tag{110}$$

In this case, we assume that different residual vectors are mutually orthogonal, so $(\vec{r}^{\,0}, \vec{r}^{\,1}) = 0$, so we can solve for β_0, i.e.,

$$\beta_0 = \frac{(\vec{r}^{\,0}, \vec{r}^{\,0})}{(\vec{r}^{\,0}, A\vec{r}^{\,0})} \tag{111}$$

At this point, post initialization, the recursion formula is introduced. For $k = 1, 2, \ldots$,

$$\vec{x}^{\,k+1} = \alpha_k \vec{x}^{\,k} + (1 - \alpha_k)\vec{x}^{\,k-1} - \beta_k \vec{r}^{\,k} \tag{112}$$

As performed in earlier analyses, if we multiply (112) by the matrix A and subtract $\vec{b}$, the residual update equation is elucidated, i.e.,

$$\vec{r}^{\,k+1} = \alpha_k \vec{r}^{\,k} + (1 - \alpha_k)\vec{r}^{\,k-1} - \beta_k A\vec{r}^{\,k} \tag{113}$$

where again, $k = 1, 2, \ldots$. What do we do with these formulas? We must determine α_k and β_k for $k = 0, 1, \ldots$. As before, we have to depend on inductive arguments where we first assume that the $\vec{r}^{\,k}$ are mutually orthogonal, i.e., $(\vec{r}^{\,k}, \vec{r}^{\,k-1-j}) = 0$ for $j = 0, 1, \ldots$. More specifically, we directly address the cases $(\vec{r}^{\,k+1}, \vec{r}^{\,k-j}) = 0$ for $j = 0, 1$. If we first consider $j = 0$, then we have that

$$(\vec{r}^{\,k+1}, \vec{r}^{\,k}) = 0 \tag{114}$$

Substitute (113) into the above inner product. We obtain

$$(\vec{r}^{\,k+1}, \vec{r}^{\,k}) = \alpha_k(\vec{r}^{\,k}, \vec{r}^{\,k}) + (1 - \alpha_k)(\vec{r}^{\,k-1}, \vec{r}^{\,k}) - \beta_k(A\vec{r}^{\,k}, \vec{r}^{\,k}) = 0 \tag{115}$$

By applying orthogonality, we may solve the above equation for α_k.

$$\alpha_k = \beta_k \frac{(A\vec{r}^{\,k}, \vec{r}^{\,k})}{(\vec{r}^{\,k}, \vec{r}^{\,k})} \tag{116}$$

Now we may examine the case where $j = 1$. The inner product in question is $(\vec{r}^{\,k+1}, \vec{r}^{\,k-1})$, and again, we substitute for $\vec{r}^{\,k+1}$.

$$(\vec{r}^{\,k+1}, \vec{r}^{\,k-1}) = \alpha_k(\vec{r}^{\,k}, \vec{r}^{\,k-1}) + (1 - \alpha_k)(\vec{r}^{\,k-1}, \vec{r}^{\,k-1}) - \beta_k(A\vec{r}^{\,k}, \vec{r}^{\,k-1}) = 0 \tag{117}$$

By invoking orthogonality and the distributive law, we obtain

$$(\vec{r}^{k-1}, \vec{r}^{k-1}) - \alpha_k(\vec{r}^{k-1}, \vec{r}^{k-1}) - \beta_k(A\vec{r}^{k}, \vec{r}^{k-1}) = 0 \tag{118}$$

By reorganizing the equation, we achieve the desired result

$$\alpha_k(\vec{r}^{k-1}, \vec{r}^{k-1}) + \beta_k(A\vec{r}^{k}, \vec{r}^{k-1}) = (\vec{r}^{k-1}, \vec{r}^{k-1}) \tag{119}$$

An additional equation required for the iterative scheme is given as

$$(\vec{r}^{k-1}, A\vec{r}^{k}) = (A\vec{r}^{k-1}, \vec{r}^{k}) \tag{120}$$

where we have posited that A is a self-adjoint matrix.[13] If we evaluate (113) at k, we have that

$$\vec{r}^{k} = \alpha_{k-1}\vec{r}^{k-1} + (1 - \alpha_{k-1})\vec{r}^{k-2} - \beta_{k-1}A\vec{r}^{k-1} \tag{121}$$

By solving this equation for $A\vec{r}^{k-1}$, we have that

$$A\vec{r}^{k-1} = \frac{1}{\beta_{k-1}}[\alpha_{k-1}\vec{r}^{k-1} + (1 - \alpha_{k-1})\vec{r}^{k-2} - \vec{r}^{k}] \tag{122}$$

We now form the inner product on the right side of (120).

$$(A\vec{r}^{k-1}, \vec{r}^{k}) = \frac{1}{\beta_{k-1}}[\alpha_{k-1}(\vec{r}^{k-1}, \vec{r}^{k}) + (1 - \alpha_{k-1})(\vec{r}^{k-2}, \vec{r}^{k}) - (\vec{r}^{k}, \vec{r}^{k})] \tag{123}$$

After applying orthogonality, the desired result is

$$(A\vec{r}^{k-1}, \vec{r}^{k}) = -\frac{(\vec{r}^{k}, \vec{r}^{k})}{\beta_{k-1}} \tag{124}$$

The above relation can be used to rewrite (119). Recall that

$$\alpha_k(\vec{r}^{k-1}, \vec{r}^{k-1}) + \beta_k(A\vec{r}^{k}, \vec{r}^{k-1}) = (\vec{r}^{k-1}, \vec{r}^{k-1}) \tag{125}$$

To promote brevity of the derivation, let us invoke Axelsson's notation. Let

$$\delta_k = (\vec{r}^{k}, \vec{r}^{k}) \text{ and } \mu_k = \frac{(\vec{r}^{k}, A\vec{r}^{k})}{(\vec{r}^{k}, \vec{r}^{k})} \tag{126}$$

So (126) can be used to simply rewrite (119) as

$$\alpha_k\delta_{k-1} + \beta_k(A\vec{r}^{k}, \vec{r}^{k-1}) = \delta_{k-1} \tag{127}$$

By substituting (124) into the above equation, we obtain

$$\alpha_k\delta_{k-1} + \beta_k\left(-\frac{\delta_k}{\beta_{k-1}}\right) = \delta_{k-1} \tag{128}$$

With the use of (126), (116) can be written as

$$\alpha_k = \beta_k\mu_k \tag{129}$$

So (128) then becomes

$$\beta_k \mu_k \delta_{k-1} + \beta_k \left(-\frac{\delta_k}{\beta_{k-1}} \right) = \delta_{k-1} \tag{130}$$

Algebraic rearrangement of this expression yields the final form

$$\beta_k^{-1} = \mu_k - \frac{\delta_k}{\delta_{k-1}} \beta_{k-1}^{-1}, \quad k = 1, 2, \ldots \tag{131}$$

Prior to presentation of the three-term recurrence algorithm, the recurrence relations for $\vec{x}^{\,k}$ and $\vec{r}^{\,k}$ are rewritten for convenience. For $k > 1$, recall that

$$\vec{x}^{\,k+1} = \alpha_k \vec{x}^{\,k} + (1 - \alpha_k) \vec{x}^{\,k-1} - \beta_k \vec{r}^{\,k} \tag{132}$$

In the above expression, let $\tilde{\alpha}_k = \alpha_k - 1$; we obtain the expression

$$\vec{x}^{\,k+1} = \alpha_k \vec{x}^{\,k} - \tilde{\alpha}_k \vec{x}^{\,k-1} - \beta_k \vec{r}^{\,k} \tag{133}$$

If we subtract then add $\vec{x}^{\,k}$, we have that

$$\vec{x}^{\,k+1} = \alpha_k \vec{x}^{\,k} - \vec{x}^{\,k} + \vec{x}^{\,k} - \tilde{\alpha}_k \vec{x}^{\,k-1} - \beta_k \vec{r}^{\,k} \tag{134}$$

Simplify.

$$\vec{x}^{\,k+1} = (\alpha_k - 1) \vec{x}^{\,k} + \vec{x}^{\,k} - \tilde{\alpha}_k \vec{x}^{\,k-1} - \beta_k \vec{r}^{\,k} \tag{135}$$

$$\vec{x}^{\,k+1} = \tilde{\alpha}_k \vec{x}^{\,k} + \vec{x}^{\,k} - \tilde{\alpha}_k \vec{x}^{\,k-1} - \beta_k \vec{r}^{\,k} \tag{136}$$

$$\vec{x}^{\,k+1} = \tilde{\alpha}_k (\vec{x}^{\,k} - \vec{x}^{\,k-1}) + \vec{x}^{\,k} - \beta_k \vec{r}^{\,k} \tag{137}$$

Let

$$\Delta \vec{x}^{\,k} = \vec{x}^{\,k} - \vec{x}^{\,k-1} \tag{138}$$

so

$$\vec{x}^{\,k+1} = \vec{x}^{\,k} + \tilde{\alpha}_k \Delta \vec{x}^{\,k} - \beta_k \vec{r}^{\,k} \tag{139}$$

Finally,

$$\vec{x}^{\,k+1} = \vec{x}^{\,k} + \Delta \vec{x}^{\,k+1}, \quad k = 1, 2, \ldots \tag{140}$$

where

$$\Delta \vec{x}^{\,k+1} = \tilde{\alpha}_k \Delta \vec{x}^{\,k} - \beta_k \vec{r}^{\,k} \tag{141}$$

The recurrence relation for the residual vector may be rewritten in a similar manner. We begin with (113)

$$\vec{r}^{\,k+1} = \alpha_k \vec{r}^{\,k} + (1-\alpha_k)\vec{r}^{\,k-1} - \beta_k A\vec{r}^{\,k} \tag{142}$$

or with the substitution for $\tilde{\alpha}_k$,

$$\vec{r}^{\,k+1} = \alpha_k \vec{r}^{\,k} - \tilde{\alpha}_k \vec{r}^{\,k-1} - \beta_k A\vec{r}^{\,k} \tag{143}$$

The remainder of the derivation is very much the same as with $\vec{x}^{\,k+1}$. Observe.

$$\vec{r}^{\,k+1} = \alpha_k \vec{r}^{\,k} - \vec{r}^{\,k} + \vec{r}^{\,k} - \tilde{\alpha}_k \vec{r}^{\,k-1} - \beta_k A\vec{r}^{\,k} \tag{144}$$

$$\vec{r}^{\,k+1} = (\alpha_k - 1)\vec{r}^{\,k} + \vec{r}^{\,k} - \tilde{\alpha}_k \vec{r}^{\,k-1} - \beta_k A\vec{r}^{\,k} \tag{145}$$

$$\vec{r}^{\,k+1} = \vec{r}^{\,k} + \tilde{\alpha}_k \vec{r}^{\,k} - \tilde{\alpha}_k \vec{r}^{\,k-1} - \beta_k A\vec{r}^{\,k} \tag{146}$$

$$\vec{r}^{\,k+1} = \vec{r}^{\,k} + \tilde{\alpha}_k (\vec{r}^{\,k} - \vec{r}^{\,k-1}) - \beta_k A\vec{r}^{\,k} \tag{147}$$

Let $\Delta\vec{r}^{\,k} = \vec{r}^{\,k} - \vec{r}^{\,k-1}$, and we have that

$$\vec{r}^{\,k+1} = \vec{r}^{\,k} + \tilde{\alpha}_k \Delta\vec{r}^{\,k} - \beta_k A\vec{r}^{\,k} \tag{148}$$

The desired result is

$$\vec{r}^{\,k+1} = \vec{r}^{\,k} + \Delta\vec{r}^{\,k+1}, \quad k = 1, 2, \ldots \tag{149}$$

where

$$\Delta\vec{r}^{\,k+1} = \tilde{\alpha}_k \Delta\vec{r}^{\,k} - \beta_k A\vec{r}^{\,k} \tag{150}$$

2.7 Three-Term Recurrence Conjugate Gradient Method - Algorithm

For preliminaries, the coefficient matrix A is symmetric and positive definite. We must choose an initial, approximate solution $\vec{x}_0$. For the heat conduction problem, this step is not difficult since such a solution is given either by the initial conditions or by the solution computed at the previous time level. Therefore,

Step 1: Compute the initial residual vector.

$$\vec{r}_{\,0} = A\vec{x}_{\,0} - \vec{b} \tag{151}$$

Step 2: Set $\Delta\vec{x}^{\,0} = 0$ and $\Delta\vec{r}^{\,0} = 0$.

Step 3: Begin the main loop; set $k = 1$.

Step 4: Compute μ_k (126), β_k (131) and α_k (129).

Step 5: Compute $\Delta\vec{x}^{\,k}$ (141), $\vec{x}^{\,k}$ (140), $\Delta\vec{r}^{\,k}$ (150) and $\vec{r}^{\,k}$ (149).

Step 6: Reset storage locations.

Step 7: Increment k.

Step 8: Return to Step 2; continue the loop.

2.8 Biconjugate Gradient Method - Theoretical Aspects

When I first began to study Conjugate Gradient methods, I was taken aback when I learned that the standard Conjugate Gradient method is restricted to symmetric, positive definite matrices. Implicit matrices associated with Computational Fluid Dynamics may not satisfy these requirements, so at that time, some twenty-four years ago, I encountered this roadblock and simply ran out of time. A college quarter proved to be all too short for this directed individual study topic. Now retired, I have the luxury of a bit more time. As it happens, the idea behind the Conjugate Gradient method can be adapted for non-symmetric matrices. The scheme considered here is the Biconjugate Gradient (BCG) method that generates biorthogonal pairs of residual vectors.[19] As in the preceding sections of this report, we follow Axelsson's discussion.[13] In that reference, the three-term recurrence scheme is adapted in theory for this purpose.[13] Without alteration, the regular three-term scheme cannot generate a biorthogonal set of vectors. One way to think of the BCG method is that one vector is associated with the non-symmetric matrix A while the other vector is associated with the matrix transpose A^T. In the interest of full disclosure, Axelsson associates the second vector with the adjoint matrix instead of the transpose, but the matrices used here are real, so we use the transpose with respect to the inner product. As in Axelsson, we consider two linear systems, i.e.,

$$A\vec{x} = \vec{b} \tag{152}$$

and

$$A^T\vec{\tilde{x}} = \vec{\tilde{b}} \tag{153}$$

Associated with these two systems are two residual vectors: $\vec{r}$ and $\vec{\tilde{r}}$. Since we are only interested in solving (152), we need not compute $\vec{\tilde{x}}$. On the other hand, the biorthogonal residual vector $\vec{\tilde{r}}$ is needed. Therefore, the recurrence scheme becomes

$$\begin{aligned}
\vec{x}^{\,k+1} &= \alpha_k\vec{x}^{\,k} + (1-\alpha_k)\vec{x}^{\,k-1} - \beta_k\vec{r}^{\,k} \\
\vec{r}^{\,k+1} &= \alpha_k\vec{r}^{\,k} + (1-\alpha_k)\vec{r}^{\,k-1} - \beta_k A\vec{r}^{\,k} \\
\vec{\tilde{r}}^{\,k+1} &= \alpha_k\vec{\tilde{r}}^{\,k} + (1-\alpha_k)\vec{\tilde{r}}^{\,k-1} - \beta_k A^T\vec{\tilde{r}}^{\,k}
\end{aligned} \tag{154}$$

The BCG algorithm is initiated with the typical zero level residual

$$\vec{r}^{\,0} = A\vec{x}^{\,0} - \vec{b} \tag{155}$$

For purposes of this algorithm, $\vec{x}^{\,0}$ is arbitrary, but for the heat equation, it is the initial conditions vector. Since we are not solving the auxiliary system (the linear system associated with A^T instead of A), $\vec{\tilde{r}}^{\,0}$ is arbitrary, yet, it must not be orthogonal to $\vec{r}^{\,0}$. Axelsson specifies that a good choice is to set $\vec{\tilde{r}}^{\,0}$ equal to $\vec{r}^{\,0}$. As in the common three-term recurrence algorithm, it is necessary to determine the unknown parameters α_k and β_k in a manner that the two sets of residual vectors $\{\vec{r}^{\,0}, \vec{r}^{\,1}, \ldots\}$ and $\{\vec{\tilde{r}}^{\,0}, \vec{\tilde{r}}^{\,1}, \ldots\}$ are biorthogonal.[13] That is to say,

$$(\vec{r}^{\,i}, \vec{\tilde{r}}^{\,j}) = 0, \quad i \neq j \tag{156}$$

The orthogonality relation $(\vec{r}^{\,k+1}, \vec{\tilde{r}}^{\,k}) = 0$ can be used to show that

$$\alpha_k = \mu_k \beta_k \tag{157}$$

where

$$\mu_k = \frac{(A\vec{r}^{\,k}, \vec{\tilde{r}}^{\,k})}{(\vec{r}^{\,k}, \vec{\tilde{r}}^{\,k})} \tag{158}$$

The above relation is easy to derive by direct computation of the inner product.[13] Secondly, the orthogonality relation $(\vec{r}^{\,k+1}, \vec{\tilde{r}}^{\,k-1}) = 0$ can be used to derive a second relation. This derivation is performed here because of a probable typographical error on page 492 in Axelsson.[13] Observe that

$$(\vec{r}^{\,k+1}, \vec{\tilde{r}}^{\,k-1}) = (\alpha_k \vec{r}^{\,k} + (1 - \alpha_k)\vec{r}^{\,k-1} - \beta_k A\vec{r}^{\,k}, \vec{\tilde{r}}^{\,k-1}) \tag{159}$$

By using orthogonality, we have that

$$(1 - \alpha_k)(\vec{r}^{\,k-1}, \vec{\tilde{r}}^{\,k-1}) - \beta_k (A\vec{r}^{\,k}, \vec{\tilde{r}}^{\,k-1}) = 0 \tag{160}$$

We can rewrite α_k via (157), and we may rewrite "1" using in multiplicative inverse. So the above equation becomes

$$(\beta_k \beta_k^{-1} - \beta_k \mu_k)(\vec{r}^{\,k-1}, \vec{\tilde{r}}^{\,k-1}) - \beta_k (A\vec{r}^{\,k}, \vec{\tilde{r}}^{\,k-1}) = 0 \tag{161}$$

Factoring produces

$$\beta_k (\beta_k^{-1} - \mu_k)(\vec{r}^{\,k-1}, \vec{\tilde{r}}^{\,k-1}) - \beta_k (A\vec{r}^{\,k}, \vec{\tilde{r}}^{\,k-1}) = 0 \tag{162}$$

or by using properties of matrix A in the inner product,

$$(\beta_k^{-1} - \mu_k)(\vec{r}^{\,k-1}, \vec{\tilde{r}}^{\,k-1}) - (\vec{r}^{\,k}, A^T \vec{\tilde{r}}^{\,k-1}) = 0 \tag{163}$$

Now it necessary to write the last equation in (154) for k instead of $k + 1$, i.e.,

$$\vec{\tilde{r}}^{\,k} = \alpha_{k-1} \vec{\tilde{r}}^{\,k-1} + (1 - \alpha_{k-1})\vec{\tilde{r}}^{\,k-2} - \beta_{k-1} A^T \vec{\tilde{r}}^{\,k-1} \tag{164}$$

Solving for the matrix-vector product, we have

$$A^T\vec{\tilde{r}}^{k-1} = \beta_{k-1}^{-1}(\alpha_{k-1}\vec{\tilde{r}}^{k-1} + (1-\alpha_{k-1})\vec{\tilde{r}}^{k-2} - \vec{\tilde{r}}^{k}) \tag{165}$$

This result we substitute into (163). All inner products except for the final one vanish to yield

$$(\beta_k^{-1} - \mu_k)(\vec{r}^{k-1}, \vec{\tilde{r}}^{k-1}) = -\beta_{k-1}^{-1}(\vec{r}^{k}, \vec{\tilde{r}}^{k}) \tag{166}$$

If Axelsson's notation is employed, e.g.,

$$\delta_k = (\vec{r}^{k}, \vec{\tilde{r}}^{k}) \tag{167}$$

(166) can be algebraically rearranged to provide the final result

$$\beta_k^{-1} = \mu_k - \frac{\delta_k}{\delta_{k-1}}\beta_{k-1}^{-1}, \quad k = 1, 2, \ldots \tag{168}$$

For the case where $k = 0$, the value of β_0 is given by (157) and (158). Specifically,

$$\beta_0 = \frac{(\vec{r}^{0}, \vec{\tilde{r}}^{0})}{(A\vec{r}^{0}, \vec{\tilde{r}}^{0})} \tag{169}$$

It is interesting to note a special property of this algorithm. Equation (160) can be written in a more revealing manner. That is

$$(1-\alpha_k)(\vec{r}^{k-1}, \vec{\tilde{r}}^{k-1}) = \beta_k(A\vec{r}^{k}, \vec{\tilde{r}}^{k-1}) \tag{170}$$

Without calling any attention to it, we have shown in equations (163) through (166) that

$$(A\vec{r}^{k}, \vec{\tilde{r}}^{k-1}) = -\beta_{k-1}^{-1}(\vec{r}^{k}, \vec{\tilde{r}}^{k}) \tag{171}$$

Thus

$$(1-\alpha_k)(\vec{r}^{k-1}, \vec{\tilde{r}}^{k-1}) = -\beta_k\beta_{k-1}^{-1}(\vec{r}^{k}, \vec{\tilde{r}}^{k}) \tag{172}$$

In (172), we require that $(\vec{r}^{k-1}, \vec{\tilde{r}}^{k-1}) \neq 0$ and that $\alpha_k \neq 0$. Otherwise, $(\vec{r}^{k}, \vec{\tilde{r}}^{k}) = 0$. This zero inner product indicates that a BCG solution has already been found and the process should stop. This event portends a breakdown of the algorithm.[13] Generally, this difficulty implies that a poor choice has been made for $\vec{r}^{0}$. For other properties or idiosyncracies of this algorithm, the interested reader may refer to Axelsson.[13]

2.9 Biconjugate Gradient Method - Algorithm

The theory presented above is practically identical to that for the three-term recurrence standard Conjugate Gradient method. Due to the clear similarities for the biconjugate gradient method, we can adapt the equations for the standard Conjugate Gradient method to the same end. The major change is that we must compute the biconjugate residual $\vec{\tilde{r}}^{k}$. The inner products are computed in the form of (167). A concise sketch of this algorithm follows

as it is used to compute the results shown in this report.[13, 19]

For preliminaries, the matrix A is non-symmetric since the assumption of symmetry is relaxed for this procedure. A initial guess at the solution for $A\vec{x} = \vec{b}$ is needed in order to compute the starting residual vector. The procedure is shown below.

Step 1: Compute the zero level residual based upon the solution guess. $\vec{r}^{\,0} = A\vec{x}^{\,0} - \vec{b}$.

Step 2: Set the zero level conjugate residual equal to residual computed in Step 1, i.e., $\vec{\tilde{r}}^{\,0} = \vec{r}^{\,0}$.

Step 3: Compute the zero level biconjugate search direction vectors, Specifically, $\vec{d}^{\,0} = -C\vec{r}^{\,0}$ and $\vec{\tilde{d}}^{\,0} = -C\vec{\tilde{r}}^{\,0}$ for a choice of constant C. In my programming, it is necessary to set $C = 1.5$ For other problems, a different value of C may be necessary.

Step 4: Set the counter $k = 0$. Start the main iteration loop.

Step 5: Compute τ_k where

$$\tau_k = \frac{(\vec{r}^{\,k}, \vec{\tilde{r}}^{\,k})}{(A\vec{d}^{\,k}, \vec{\tilde{d}}^{\,k})} \tag{173}$$

Step 6: Compute the updated solution as

$$\vec{x}^{\,k+1} = \vec{x}^{\,k} + \tau_k \vec{d}^{\,k} \tag{174}$$

Step 7: Compute the biconjugate residuals via

$$\begin{aligned} \vec{r}^{\,k+1} &= \vec{r}^{\,k} + \tau_k A\vec{d}^{\,k} \\ \vec{\tilde{r}}^{\,k+1} &= \vec{\tilde{r}}^{\,k} + \tau_k A^T \vec{\tilde{d}}^{\,k} \end{aligned} \tag{175}$$

Step 8: Compute β_k as

$$\beta_k = \frac{(\vec{r}^{\,k+1}, \vec{\tilde{r}}^{\,k+1})}{(\vec{r}^{\,k}, \vec{\tilde{r}}^{\,k})} \tag{176}$$

Step 9: Compute the new biconjugate search directions via

$$\begin{aligned} \vec{d}^{\,k+1} &= -\vec{r}^{\,k+1} + \beta_k \vec{d}^{\,k} \\ \vec{\tilde{d}}^{\,k+1} &= -\vec{\tilde{r}}^{\,k+1} + \beta_k \vec{\tilde{d}}^{\,k} \end{aligned} \tag{177}$$

Step 10: Check convergence. Update k; return to Step 5.

3 Test Problems

Every computer code I have developed in support of this research is written new line by line. No pre-packaged mathematics software is employed in any capacity or in any manner of association with the work described in this report. For this reason, my computer codes must be thoroughly tested through validation. The discretization schemes are archived within the public literature, so it is comfortable in assuming that these schemes are consistent and stable (with some restrictions). Still, the numerical results must be validated against a battery of exact solutions. The requisite exact solutions are developed in this section of the report.

3.1 Solution of the Heat Equation in One Dimension

The first test problem addressed by the numerical methods described in this report is a one-dimensional heat conduction problem with asymmetric boundary conditions. This initial boundary value problem [7] (IBVP) is given as

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2},\ 0 < x < 1,\ \ t > 0 \tag{178}$$

$$u(0,t) = a;\ u(1,t) = b \tag{179}$$

$$u(x,0) = u_0(x) = g \tag{180}$$

where a and b are permitted to be non-zero. Moreover, a need not equal b. In a standard approach, we tend to seek an exact solution may be obtained for this problem by using Separation of Variables.[1] We assume a solution of the form

$$u(x,t) = T(t)X(t) \tag{181}$$

over the same domain as the IBVP. By substituting (181) into (178), we obtain

$$\frac{T_t(t)}{T(t)} = \frac{X_{xx}(x)}{X(x)} = -\lambda^2 \tag{182}$$

where the separation constant λ has been chosen. Two ordinary differential equations can be extracted from (182), i.e.,

$$T_t(t) + \lambda^2 T(t) = 0 \tag{183}$$

$$X_{xx}(x) + \lambda^2 X(x) = 0 \tag{184}$$

The first equation (183) is governs time-like behavior while (184) describes the behavior of the system in space. The time-like equation is easily solved by substituting into (183) the trial solution

$$T(t) = A\exp(\omega t) \tag{185}$$

Note that A is an arbitrary constant. Simplification reveals that

$$\omega = -\lambda^2 \tag{186}$$

and we have that

$$T(t) = A\exp(-\lambda^2 t), \quad t > 0 \tag{187}$$

By substituting a similar trial solution into (184), we obtain two linearly independent solutions. These functions may be summed to form a particular solution for (184), i.e.,

$$X(x) = B\cos(\lambda x) + C\sin(\lambda x), \quad 0 < x < 1 \tag{188}$$

B and C are arbitrary constants. In many textbooks, trivial boundary conditions, i.e., $a = b = 0$, are applied to (188) to extract an eigenvalue equation for the parameter λ. As stated, this problem is more complicated since since a and b are generally non-zero. In this case, a different procedure must be used to arrive at an eigenvalue equation. We can reduce this system to a problem with zero boundary conditions by writing $u(x,t)$ as follows.[5]

$$u(x,t) = u_{ss}(x) + u_{us}(x,t) \tag{189}$$

where $u_{ss}(x)$ is a steady state solution in space that satisfies (178) with the non-trivial boundary conditions (179). Note that $u_{ss}(x)$ is not dependent on the time-like coordinate t. The second component of (189) $u_{us}(x,t)$ is a solution of the time-dependent heat equation with trivial boundary conditions. Note that (178) is linear, so we may conclude that (189) is an acceptable solution. Also, this procedure is alluded to in Zachmanaglou and Thou.[5] The steady solution $u_{ss}(x)$ is a solution to the Boundary Value Problem

$$\frac{\partial^2 u_{ss}(x)}{\partial x^2} = \frac{d^2 u_{ss}(x)}{dx^2} = 0, \qquad a \le x \le b \tag{190}$$

We can solve (190) by simple integration. It is easily shown that

$$u_{ss}(x) = C_0\, x + C_1 \tag{191}$$

where C_0 and C_1 are arbitrary constants. The boundary conditions $u_{ss}(0) = a$ and $u_{ss}(1) = b$ allow these constants to be identified. The resulting solution is

$$u_{ss}(x) = (b-a)x + a \tag{192}$$

Considering (192), we have that

$$u_{ss}(0) = a \;\; ; u_{ss}(1) = b \tag{193}$$

In the light of (179) and (189), we can deduce that

$$\frac{\partial u_{us}(x,t)}{\partial t} = \frac{\partial^2 u_{us}(x,t)}{\partial x^2}, \quad 0 \le x \le 1, \quad t > 0 \tag{194}$$

$$u_{us}(0,t) = u_{us}(1,t) = 0, \qquad t \ge 0 \tag{195}$$

forming an IBVP for the unsteady $u_{us}(x,t)$. The PDE (194) for $u_{us}(x,t)$ has the same form as our heat equation, so the form (181) derived in the Separation of Variables is valid. Hence, we are free to apply the particular solution form (188) to space dependent part of $u_{us}(x,t)$. That is to say,

$$X(x) = B\cos(\lambda x) + C\sin(\lambda x), \quad 0 \leq x \leq 1 \tag{196}$$

If we apply the boundary condition (195) at $x = 0$, it is clear that coefficient B must be zero, and

$$X(x) = C\sin(\lambda x), \quad 0 \leq x \leq 1 \tag{197}$$

By applying the boundary condition (195) at $x = 1$, we obtain the eigenvalue equation since C cannot be zero. We have

$$x(1) = C\sin(\lambda x) = 0, \quad 0 \leq x \leq 1 \tag{198}$$

Hence,

$$\lambda = n\pi, \quad n = 1, 2, \ldots \tag{199}$$

Combining this result with (198) and (187), we have the particular solution

$$u_{us}(x,t) = A' \exp(-n^2\pi^2 t)\sin(n\pi x), \quad n = 1, 2, \ldots \tag{200}$$

where A' is an arbitrary constant that must be obtained from application of the initial conditions. Recall the form of $u(x,t)$.

$$u(x,t) = u_{ss}(x) + u_{us}(x,0) \tag{201}$$

The initial condition is defined as $u(x,0) = u_0(x)$; hence,

$$u(x,0) = u_0(x) = u_{ss}(x) + u_{us}(x,0) \tag{202}$$

and

$$u_{us}(x,0) = u_0(x) - u_{ss}(x) \tag{203}$$

By substituting (192), we have

$$u_{us}(x,0) = u_0(x) - [(b-a)x + a] \tag{204}$$

A Fourier series can now be used to represent $u_{us}(x,t)$, i.e.,

$$u_{us}(x,t) = \sum_{n=1}^{\infty} A'_n \exp(-n^2\pi^2 t)\sin(n\pi x) \tag{205}$$

If we evaluate (205) for $t = 0$ and integrate over the range of x, the A'_n are revealed.

$$\int_0^1 u_{us}(x,0)\sin(m\pi x)dx = \sum_{n=1}^{\infty} A'_n \int_0^1 \sin(n\pi x)\sin(m\pi x)dx \tag{206}$$

The integral on the right hand side of (206) can be evaluated as shown below.

$$\sum_{n=1}^{\infty} A'_n \int_0^1 \sin(n\pi x)\sin(m\pi x)dx = A'_n \frac{1}{2}\delta_{mn} = A'_m/2 \tag{207}$$

where δ_{mn} is the Dirac Delta function. By using this result in (206), we obtain

$$A'_m = 2\int_0^1 u_{us}(x,0)\sin(m\pi x)dx \tag{208}$$

By substituting (204), we have that

$$A'_m = 2\int_0^1 (u_0(x) - [(b-a)x + a])\sin(m\pi x)dx \tag{209}$$

Evaluation of the elementary integrals produces a formula for the A'_m as a function of the initial value $u_0(x)$, i.e.,

$$A'_m = 2\int_0^1 u_0(x)\sin(m\pi x)dx + \frac{2(b-a)}{m\pi}\cos(m\pi) - \frac{2a(1-\cos(m\pi))}{m\pi}, \quad m = 1, 2, \ldots \tag{210}$$

For the purposes of this report, the initial condition is a constant function $u_0(x) = u_0$. Hence,

$$A'_m = \frac{2u_0(1-\cos(m\pi))}{m\pi} + \frac{2(b-a)}{m\pi}\cos(m\pi) - \frac{2a(1-\cos(m\pi))}{m\pi}, \quad m = 1, 2, \ldots \tag{211}$$

or changing m for n,

$$A'_n = \frac{2(u_0-a)(1-\cos(n\pi))}{n\pi} + \frac{2(b-a)}{n\pi}\cos(n\pi), \quad n = 1, 2, \ldots \tag{212}$$

The general solution for this problem is given by (189), (192), (205) and (212).

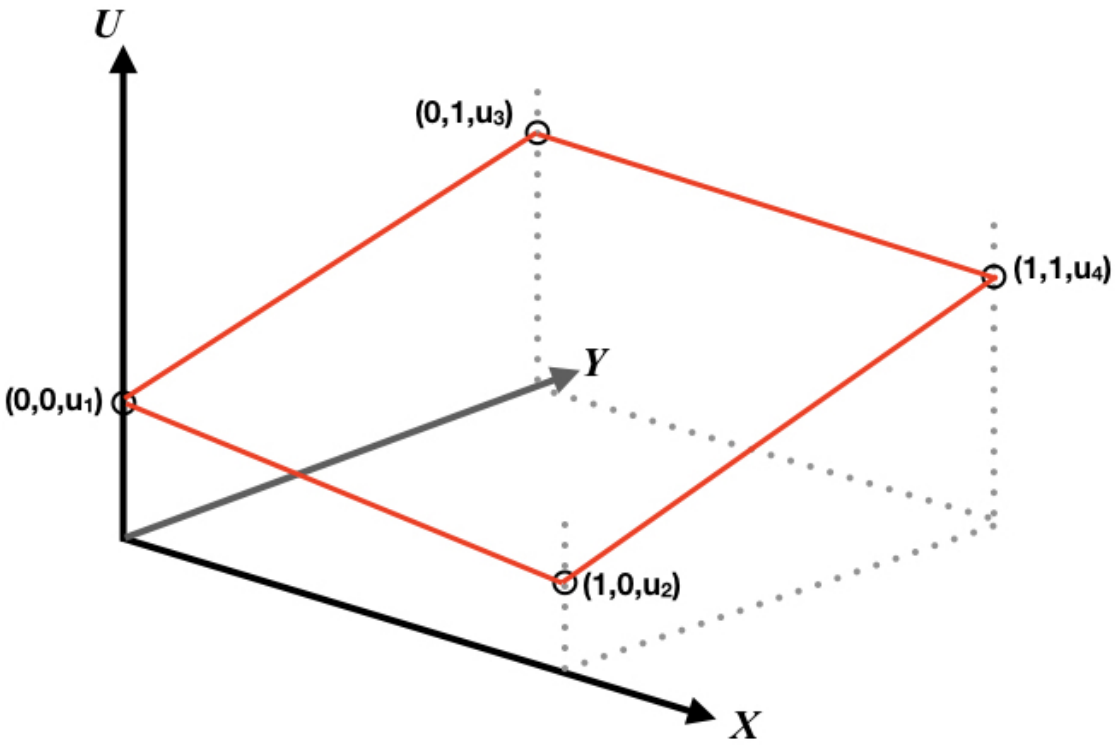

Figure 1: Boundary Configuration for the 2D Heat Conduction Test Problem

3.2 Solution of the Heat Equation in Two Dimensions

This problem is designed to form a more complicated test case for our numerical methods. In this case, the solution $u(x, y, t)$ is a surface existing above the xy-plane as shown in Figure 1. The boundary conditions are enforced along the red lines shown in Figure 1. The boundary locus for $u(x, y, t)$ forms a skewed plane above the xy-plane. The associated IBVP is expressed as follows.

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \quad 0 < x < 1,\ 0 < y < 1,\ t > 0 \tag{213}$$

$$u(0, y, t) = u_1 + y(u_3 - u_1), \quad 0 < y < 1 \tag{214}$$

$$u(1, y, t) = u_2 + y(u_4 - u_2), \quad 0 < y < 1 \tag{215}$$

$$u(x, 0, t) = u_1 + x(u_2 - u_1), \quad 0 < x < 1 \tag{216}$$

$$u(x, 1, t) = u_3 + x(u_4 - u_3), \quad 0 < x < 1 \tag{217}$$

$$u(x, y, 0) = u_0(x, y) \tag{218}$$

In the design of this problem, u_1, u_2 and u_3 are free parameters for the user to select. The final boundary parameter u_4 is given by

$$u_4 = u_2 + u_3 - u_1 \tag{219}$$

in order to maintain a planar boundary configuration. As you may recall, three non-collinear points in Euclidean space define a plane. It is obvious that these boundary conditions are non-trivial, so as with the preceding test problem, the solution for this case is expressed as

$$u(x,y,t) = u_{ss}(x,y) + u_{us}(x,y,t) \tag{220}$$

where $u_{ss}(x,y)$ is a steady state solution of the heat equation that enforces the non-trivial boundary conditions delineated in (218). On the other hand, $u_{us}(x,y,t)$ is an unsteady solution of the heat equation that enforces trivial Dirichlet boundary conditions on the square, i.e.,

$$\begin{aligned} u(0,y,t) &= 0, \quad 0 < y < 1 \\ u(1,y,t) &= 0, \quad 0 < y < 1 \\ u(x,0,t) &= 0, \quad 0 < x < 1 \\ u(x,1,t) &= 0, \quad 0 < x < 1 \end{aligned} \tag{221}$$

The solution $u_{ss}(x,y)$ for the steady state problem is simply a solution for Laplace's equation respective of the boundary conditions (221). We can determine this solution by using a trial equation, e.g.,

$$u_{ss}(x,y) = a + bx + cy + dxy \tag{222}$$

It is easy to show that this equation satisfies the 2D Laplace equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \quad 0 < x < 1,\ 0 < y < 1 \tag{223}$$

Coefficients a, b, c and d can be determined by writing (222) at each of the four points u_1, u_2, u_3 and u_4. This action produces a system of four equations in four unknowns. Specifically,

$$\begin{aligned} a + b(0) + c(0) + d(0)(0) &= u_1 \\ a + b(1) + c(0) + d(1)(0) &= u_2 \\ a + b(0) + c(1) + d(0)(1) &= u_3 \\ a + b(1) + c(1) + d(1)(1) &= u_4 \end{aligned}$$

The above system can be reduced as follows.

$$\begin{aligned} a = u_1 \Rightarrow \qquad & a = u_1 \\ a + b = u_2 \Rightarrow \qquad & b = u_2 - u_1 \\ a + c = u_3 \Rightarrow \qquad & c = u_3 - u_1 \\ a + b + c + d = u_4 \Rightarrow \qquad & d = 0 \end{aligned} \tag{224}$$

It follows that the steady state solution is given as

$$u_{ss}(x,y) = u_1 + (u_2 - u_1)x + (u_3 - u_1)y \tag{225}$$

The remaining function u_{us} is a solution to the IBVP below.

$$\frac{\partial u_{us}}{\partial t} = \frac{\partial^2 u_{us}}{\partial x^2} + \frac{\partial^2 u_{us}}{\partial y^2}, \quad 0 < x < 1,\ 0 < y < 1,\ t > 0 \tag{226}$$

$$\begin{aligned} u_{ss}(0,y,t) &= 0,\ 0 < y < 1 \\ u_{us}(1,y,t) &= 0,\ 0 < y < 1 \\ u_{us}(x,0,t) &= 0,\ 0 < x < 1 \\ u_{us}(x,1,t) &= 0,\ 0 < x < 1 \end{aligned} \tag{227}$$

The system above may be solved by Separation of Variables. Assume that

$$u_{us}(x,y,t) = X(x)Y(y)T(t) \tag{228}$$

If we substitute this trial solution into the PDE, we obtain

$$\frac{T_t}{T} = \frac{X_{xx}}{X} + \frac{Y_{yy}}{Y} = -\lambda^2 \tag{229}$$

Subscript t denotes the derivative with respect to t. The separation assumption is applied above with separation constant λ^2. Two differential equations result; the first is the time related equation; that is

$$T_t + \lambda^2 T = 0 \tag{230}$$

with solution

$$T(t) = E\exp(-\lambda^2 t) \tag{231}$$

The second is a coupled equation between $X(x)$ and $Y(y)$.

$$\frac{X_{xx}}{X} + \frac{Y_{yy}}{Y} = -\lambda^2 \tag{232}$$

$$\frac{Y_{yy}}{Y} + \lambda^2 = -\frac{X_{xx}}{X} = \mu^2 \tag{233}$$

In the above equation, the separation assumption has been reapplied with constant μ^2. It follows that two ordinary differential equations in space result, i.e.,

$$X_{xx} + \mu^2 X = 0 \tag{234}$$

$$Y_{yy} + (\lambda^2 - \mu^2)Y = 0 \tag{235}$$

Both of these equations entail the form of the simple harmonic oscillator with the solution forms

$$X(x) = A\cos(\mu x) + B\sin(\mu x) \tag{236}$$

$$Y(y) = C\cos(\sqrt{\lambda^2 - \mu^2}\, y) + D\sin(\sqrt{\lambda^2 - \mu^2}\, y) \tag{237}$$

Beginning with (237), we note that the boundary conditions given in (227) imply that

$$Y(0) = 0; \quad Y(1) = 0 \tag{238}$$

so

$$Y(0) = C + D(0) = 0 \rightarrow C = 0 \tag{239}$$

and for $Y(y)$ to be a non-trivial solution, D cannot be zero. Hence,

$$Y(1) = D\sin(\sqrt{\lambda^2 - \mu^2}) = 0 \quad \rightarrow \sin(\sqrt{\lambda^2 - \mu^2}) = 0 \tag{240}$$

This eigenvalue equation is satisfied when

$$\sqrt{(\lambda^2 - \mu^2)} = m\pi, \quad m = 1, 2, \ldots \tag{241}$$

Therefore,

$$\lambda^2 - \mu^2 = m^2\pi^2, \quad m = 1, 2, \ldots \tag{242}$$

Now we may consider a particular solution for (236). As with the equation for $Y(y)$, the boundary condition equations (227) dictate that

$$X(0) = 0; \quad X(1) = 0 \tag{243}$$

In this case,

$$X(0) = A + B(0) = 0 \rightarrow A = 0 \tag{244}$$

We require that $X(x)$ be non-trivial, so B must be non-zero. The logical conclusion is that

$$X(1) = B\sin(\mu) = 0 \rightarrow \sin(\mu) = 0 \tag{245}$$

Thus,

$$\mu = n\pi, \quad n = 1, 2, \ldots \tag{246}$$

Combining (246) and (242) provides the result

$$\lambda = \sqrt{n^2 + m^2}\,\pi, \quad n, m = 1, 2, \ldots \tag{247}$$

Rolling up the preceding results for the ordinary differential equations, we have that

$$X(x) = B\sin(n\pi x), \quad n = 1, 2, \ldots \tag{248}$$

$$Y(y) = D\sin(m\pi y), \quad m = 1, 2, \ldots \tag{249}$$

$$T(t) = E\exp(-(m^2 + n^2)\pi^2 t), \quad m, n = 1, 2, \ldots \tag{250}$$

By returning these ordinary differential equations to (228) and forming a Fourier sum over two indices, we obtain

$$u_{us}(x, y, t) = \sum_{n,m=1}^{\infty} E_{nm}\exp(-(m^2 + n^2)\pi^2\, t)\sin(n\pi x)\sin(m\pi y) \tag{251}$$

Recall that $u(x, y, t) = u_{ss}(x, y) + u_{us}(x, y, t)$, so

$$u(x, y, t) = u_{ss}(x, y) + \sum_{n,m=1}^{\infty} E_{nm}\exp(-(m^2 + n^2)\pi^2\, t)\sin(n\pi x)\sin(m\pi y) \tag{252}$$

Substituting for (225), we have

$$\begin{aligned} u(x, y, t) = {} & u_1 + (u_2 - u_1)x + (u_3 - u_1)y \\ & + \sum_{n,m=1}^{\infty} E_{nm}\exp(-(m^2 + n^2)\pi^2\, t)\sin(n\pi x)\sin(m\pi y) \end{aligned} \tag{253}$$

To evaluate the Fourier coefficients E_{nm}, we evaluate $u(x, y, t)$ at $t = 0$.

$$u(x, y, 0) = u_1 + (u_2 - u_1)x + (u_3 - u_1)y + \sum_{n,m=1}^{\infty} E_{nm}\sin(n\pi x)\sin(m\pi y) \tag{254}$$

Recall that $u(x, y, 0)$ is given by the initial conditions, i.e., $u(x, y, 0) = u_0(x, y)$. We rewrite the Fourier expansion as follows.

$$\sum_{n,m=1}^{\infty} E_{nm}\sin(n\pi x)\sin(m\pi y) = u_0(x, y) - u_1 - (u_2 - u_1)x - (u_3 - u_1)y \tag{255}$$

Orthogonality of the sine function can be applied to this series to determine E_{nm}. When multiplying by complementary sine functions can integrating over x and y, we obtain

$$\begin{aligned} & \sum_{n,m=1}^{\infty} E_{nm}\int_0^1\int_0^1 \sin(n\pi x)\sin(p\pi x)\sin(m\pi y)\sin(q\pi y)dxdy \\ & = \int_0^1\int_0^1 [u_0(x, y) - u_1 - (u_2 - u_1)x - (u_3 - u_1)y]\sin(p\pi x)\sin(q\pi y)dxdy \end{aligned} \tag{256}$$

The above expression requires simplification; observe

$$\sum_{n,m=1}^{\infty} E_{nm} \int_0^1 \sin(n\pi x)\sin(p\pi x)\,dx \int_0^1 \sin(m\pi y)\sin(q\pi y)\,dy$$
$$= \int_0^1\int_0^1 u_0(x,y)\sin(p\pi x)\sin(q\pi y)dxdy \tag{257}$$
$$- \int_0^1\int_0^1 [u_1 + (u_2-u_1)x + (u_3-u_1)y]\sin(p\pi x)\sin(q\pi y)\,dxdy$$

The left side of this expression can be simplified by using a standard Fourier integral result. It is not difficult to show that

$$\int_0^1 \sin(n\pi x)\sin(p\pi x)dx = \frac{\delta_{np}}{2} \tag{258}$$

With this result, (257) becomes

$$\sum_{n,m=1}^{\infty} E_{nm}\left(\frac{\delta_{np}}{2}\right)\left(\frac{\delta_{mq}}{2}\right) = \int_0^1\int_0^1 u_0(x,y)\sin(p\pi x)\sin(q\pi y)dxdy$$
$$- \int_0^1\int_0^1 [u_1 + (u_2-u_1)x + (u_3-u_1)y]\sin(p\pi x)\sin(q\pi y)\,dxdy \tag{259}$$

The delta functions are instrumental in selecting a single coefficient, i.e.,

$$E_{pq} = 4\int_0^1\int_0^1 [u_0(x,y) - u_1 - (u_2-u_1)x - (u_3-u_1)y]\sin(p\pi x)\sin(q\pi y)\,dxdy \tag{260}$$

for $p, q = 1, 2, \ldots$ and where $u_0(x,y)$ has been included in the integral along with the steady state solution. Finally, we can conclude that (253) and (260) constitute the general solution for this two dimensional heat conduction problem given u_1, u_2, u_3 and u_4 along with the initial conditions function $u_0(x,y)$.

Suppose that the initial conditions are given as a constant function say, $u_0(x,y) = u_0$. Then, with some work, each of the integrals in (260) are easily integrable. This is the scenario addressed in our numerical test problem. Omitting the details, we can simply state the formula for the coefficients here. For $p, q = 1, 2, \ldots$,

$$E_{pq} = 4\Bigg[(u_0-u_1)\left(\frac{1-\cos(p\pi)}{p\pi}\right)\left(\frac{1-\cos(q\pi)}{q\pi}\right)$$
$$+(u_2-u_1)\left(\frac{\cos(p\pi)}{p\pi}\right)\left(\frac{1-\cos(q\pi)}{q\pi}\right) \tag{261}$$
$$+(u_3-u_1)\left(\frac{1-\cos(p\pi)}{p\pi}\right)\left(\frac{\cos(q\pi)}{q\pi}\right)\Bigg]$$

These coefficients when used in (253) provide the general solution for this 2D heat conduction test problem.

3.3 Solution of the Heat Equation in Three Dimensions

In the technical sense, this three dimensional test problem is not as complicated as the preceding two dimensional test problem. It is still important to test the addition of a third degree of freedom to our numerical algorithms. This test case is similar to Problem 8 on page 148 of Weinberger.[1] In this scenario, a unit cube at uniform temperature T_1 is "immersed" in a reservoir at a different temperature T_2. In truth, this problem is cartoon-like because the temperature of the external reservoir is enforced as a Dirichlet boundary condition at the cube boundaries. In reality, not quite addressed here, the material comprising the reservoir would also react to the heat flow between itself and the cube. Still, our cartoon problem, as defined, is sufficient to test the numerical algorithms described in Section 2 of this report.

The IBVP is described as follows where $u(x, y, z, t)$ is the temperature distribution within the cube.

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}, \quad 0 < x, y, z < 1, \ t > 0 \tag{262}$$

$$u(x, y, 0, t) = u(x, y, 1, t) = T_2, \quad 0 \leq x, y \leq 1 \tag{263}$$

$$u(0, y, z, t) = u(1, y, z, t) = T_2, \quad 0 \leq y, z \leq 1 \tag{264}$$

$$u(x, 0, z, t) = u(x, 1, z, t) = T_2, \quad 0 \leq x, z \leq 1 \tag{265}$$

$$u(x, y, z, 0) = T_1, \quad 0 < x, y, x < 1, \tag{266}$$

(262) is the governing partial differential equation while (263,264,265) are boundary conditions. Equation (266) is the initial condition.

As described, the boundary conditions for the cube are non-homogeneous. Therefore, it is necessary to express the general solution for the cube as the sum of a steady state solution u_{ss} and a unsteady solution u_{us} with homogeneous boundary conditions. This idea is expressed as

$$u(x, y, z, t) = u_{ss}(x, y, z) + u_{us}(x, y, z, t) \tag{267}$$

Since the external temperature T_2 is fixed, u_{ss} is a constant function. It follows that u_{us} has the form of the heat equation with homogeneous boundary conditions. That is, $u_{us}(x, y, z, t) = 0$ for all (x, y, z) on the boundary of the cube; this locus is described in equations (263,264,265).

$$T_2 = T_1 + u_{us}(x, y, z, t) \tag{268}$$

Separation of Variables is employed to solve u_{us} system. Let

$$u_{us}(x,y,z,t) = X(x)Y(y)Z(z)T(t) \tag{269}$$

After substituting (269) into (262), we see that

$$XYZT_t = X_{xx}YZT + XY_{yy}ZT + XYZ_{zz}T \tag{270}$$

Division by a non-zero $XYZT$ produces the equation

$$\frac{T_t}{T} = \frac{X_{xx}}{X} + \frac{Y_{yy}}{Y} + \frac{Z_{zz}}{Z} \tag{271}$$

Since the sides of the above equation vary with completely different independent variables, then the only way that this equation can be true is if both sides equal to a constant.[1] Let us choose this separation constant as $-\lambda^2$. Hence,

$$\frac{T_t}{T} = \frac{X_{xx}}{X} + \frac{Y_{yy}}{Y} + \frac{Z_{zz}}{Z} = -\lambda^2 \tag{272}$$

The two differential equations result, i.e.,

$$T_t + \lambda^2 T = 0 \tag{273}$$

$$-\frac{X_{xx}}{X} = \frac{Y_{yy}}{Y} + \frac{Z_{zz}}{Z} + \lambda^2 \tag{274}$$

Note that (273) is an ordinary differential equation for the time-like function T. This equation is easily solved for the result

$$T(t) = A\exp(-\lambda^2 t) \tag{275}$$

The remaining relation (274) is a separable equation. Assume a separation constant of μ^2. It follows that two additional differential equations arise.

$$X_{xx} + \mu^2 X = 0 \tag{276}$$

$$\frac{Y_{yy}}{Y} + \frac{Z_{zz}}{Z} + \lambda^2 = \mu^2 \tag{277}$$

We can identify (276) as an ordinary differential equation for the X space function. The remaining equation (277) is separable, so assume a separation constant of ν^2. As before, two new differential equations result.

$$Y_{yy} + \nu^2 Y = 0 \tag{278}$$

$$Z_{zz} + (\lambda^2 - \mu^2 - \nu^2)Z = 0 \tag{279}$$

To complete the derivation of the exact solution for this problem, boundary conditions must be applied. Since we have performed this type of analysis for the 1D and 2D test

problems, this discussion is kept brief. Recall (277); the particular solution and its associated boundary conditions are given as

$$X(x) = A\cos(\mu x) + B\sin(\mu x), \quad X(0) = X(1) = 0 \tag{280}$$

Application of the boundary conditions shows that $A = 0$; thus, $B \neq 0$, and at $x = 1$, we arrive at the eigenvalue equation

$$\sin(\mu) = 0 \ \rightarrow \ \mu = l\pi, \ \ l = 1, 2, \ldots \tag{281}$$

For the second component of u_{us}, recall (278); the particular solution for this equation and its specific boundary conditions are

$$Y(y) = C\cos(\nu y) + D\sin(\nu y), \quad Y(0) = Y(1) = 0 \tag{282}$$

Again, the application of boundary conditions demonstrates that $C = 0$, so $D \neq 0$. $Y(1) = 0$ gives rise to the eigenvalue equation

$$\sin(\nu) = 0 \ \rightarrow \ \nu = m\pi, \ \ m = 1, 2, \ldots \tag{283}$$

The final component of the solution u_{us} is given by the particular solution for (279), i.e.,

$$Z(z) = E\cos(\sqrt{\lambda^2 - \mu^2 - \nu^2}\, z) + F\sin(\sqrt{\lambda^2 - \mu^2 - \nu^2}\, z), \quad Z(0) = Z(1) = 0 \tag{284}$$

By virtue of the boundary conditions, we have that $E = 0$, so $F \neq 0$. The condition $Z(1) = 0$ yields the eigenvalue equation

$$\sin(\sqrt{\lambda^2 - \mu^2 - \nu^2}) = 0 \ \rightarrow \ \sqrt{\lambda^2 - \mu^2 - \nu^2} = n\pi, n = 1, 2, \ldots$$

or

$$\lambda^2 - \mu^2 - \nu^2 = n^2\pi^2, \quad n = 1, 2, \ldots \tag{285}$$

Therefore,

$$\lambda^2 = (l^2 + m^2 + n^2)\pi^2, \quad l, m, n = 1, 2, \ldots \tag{286}$$

The preceding solutions can be superposed to form a Fourier series solution for the cube. This result is written as follows.

$$u(x, y, z, t) = T_2 + \sum_{l,m,n}^{\infty} A_{lmn} \exp[-(l^2 + m^2 + n^2)t] \sin(l\pi x) \sin(m\pi y) \sin(n\pi z) \tag{287}$$

The Fourier coefficients A_{lmn} may be determined by evaluating this solution at $t = 0$.

$$u(x, y, z, 0) = T_1 = T_2 + \sum_{l,m,n}^{\infty} A_{lmn} \sin(l\pi x) \sin(m\pi y) \sin(n\pi z) \tag{288}$$

After rearrangement, we have

$$\sum_{l,m,n}^{\infty} A_{lmn} \sin(l\pi x) \sin(m\pi y) \sin(n\pi z) = T_1 - T_2 \tag{289}$$

As with the previous two test problems, the Fourier coefficients A_{lmn} can be determined through the use of orthogonality. The associated integral is written as

$$\begin{aligned} \sum_{l,m,n}^{\infty} A_{lmn} \int_0^1 \sin(l\pi x) \sin(p\pi x)\, dx \int_0^1 \sin(m\pi y) \sin(q\pi y)\, dy \int_0^1 \sin(n\pi z) \sin(r\pi z)\, dz \\ = (T_1 - T_2) \int_0^1 \sin(p\pi x)\, dx \int_0^1 \sin(q\pi y)\, dy \int_0^1 \sin(r\pi z)\, dz \end{aligned} \tag{290}$$

After evaluation of these integrals, some simplification and a trivial change of indices,

$$A_{lmn} = \frac{8(T_1 - T_2)(1 - \cos(l\pi))(1 - \cos(m\pi)(1 - \cos(n\pi))}{lmn\, \pi^3} \tag{291}$$

These coefficients, in conjunction with (287) form the general solution for the cube.

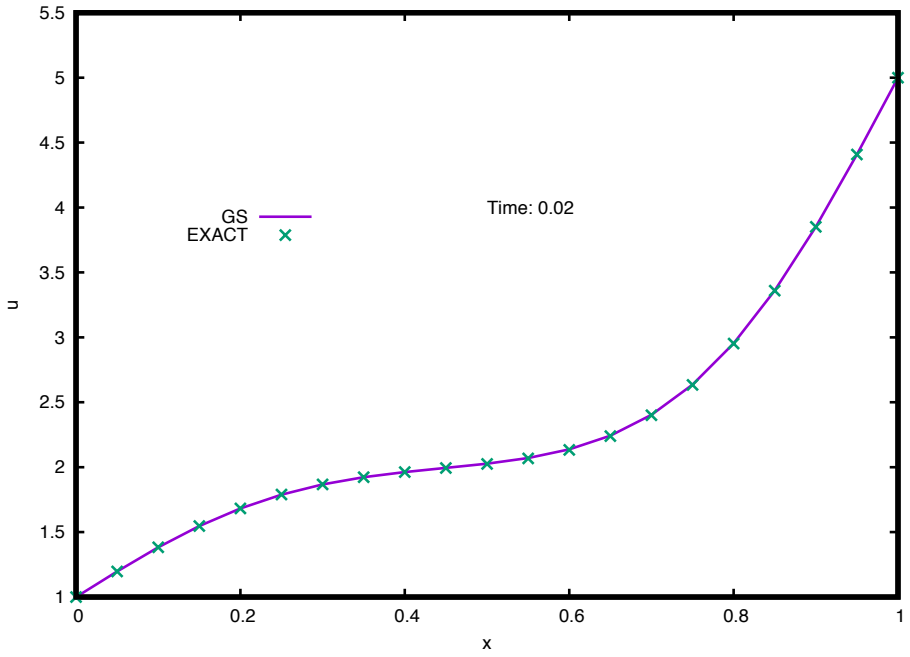

Figure 2: Time 0.02 GS numerical solution for the 1D test problem compared with the exact solution

4 Numerical Results for the 1D Test Problem

In generating the results shown in these three sections, the matrix solvers are converged to a tolerance of 10^{-10}.

This section of the report contains numerical results for the first test problem. The IBVP is stated as follows.

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2}, \quad 0 \leq x \leq 1, \quad t > 0 \\ u(0,t) &= 1; \quad u(1,t) = 5 \\ u(x,0) &= 2, \quad 0 < x < 1 \end{aligned} \tag{292}$$

To start, we solve this problem on the uniform grid via the Gauss-Seidel (GS), standard Conjugate Gradient (CG) and three term recurrence Conjugate Gradient (T3) linear system solvers. The calculations were performed on an Apple Macbook Pro (mid-2014) equipped with a 2.6 GHz Intel Core i5 processor running MacOS Mojave, Version 10.14.6. All programming is written using the GFORTRAN compiler, GCC Version 5.0.0.

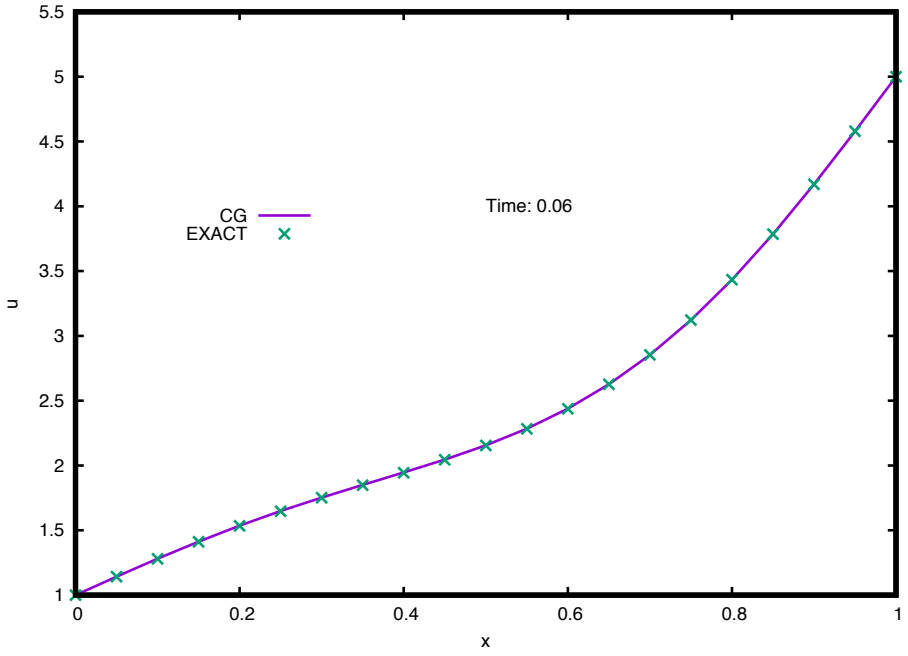

Figure 3: Time 0.06 CG numerical solution for the 1D test problem compared with the exact solution

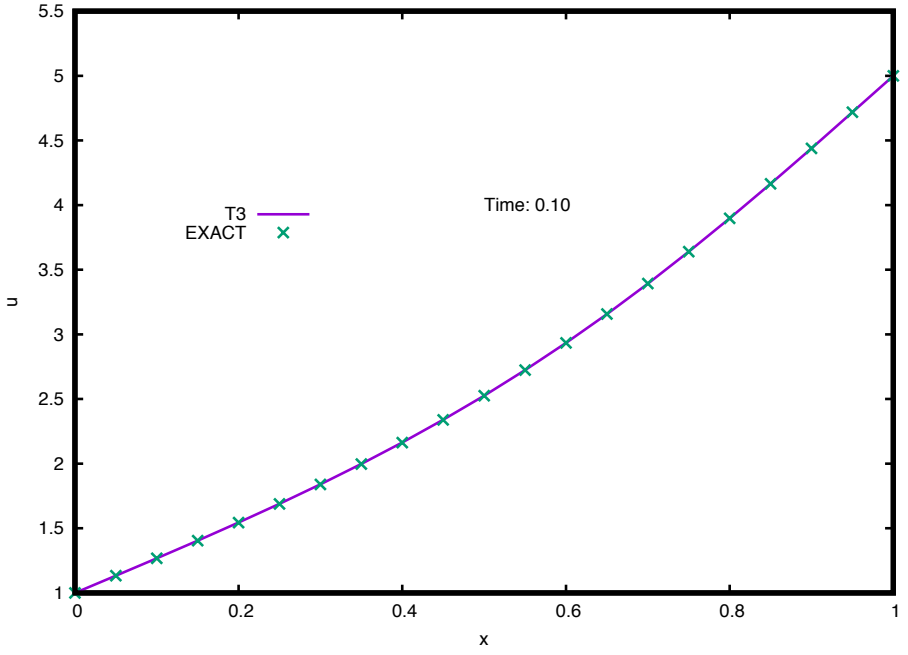

Figure 4: Time 0.10 T3 numerical solution for the 1D test problem compared with the exact solution

4.1 Results Computed on the Uniform Grid

The mesh used for this problem is intentionally under resolved because it is desirable to obtain a clear picture of how numerical error evolves in the course of the time dependent solution. It is also important to compare how the different linear solvers converge at each time step in the unsteady solution. The mesh consists of 21 points bounding 20 cells for the finite difference solution. The time step is fixed at 25×10^{-4}. The associated mesh ratio is one. The time frame for the simulation runs from time zero to time 0.24, when the steady state solution is achieved. The numerical solutions produced by the GS, CG and T3 methods are virtually identical to one another, so plots are shown at three distinct time levels, one time level per solver. To that end, Figure 2 shows the time 0.02 numerical solution computed by using the GS method. Figures 3 and 4 contain similar plots of u vs. x for time levels 0.06 and 0.10, respectively. A glance at these plots suggests that the agreement between numerical and exact solutions is very good. Direct calculations of error for the numerical solutions are presented in Table 1. The maximum error exhibited by the numerical solution is less than one percent for the entirety of the simulation time. For this reason, these results are excellent. Otherwise, the minimum and maximum error magnitudes are relatively steady in time. Statistical formulas are used to estimate the mean and standard deviation for the error as well.[20] As it happens, the mean error values decrease slightly in time as does the spread of the error (σ). Although standard estimators have been used for μ and σ, their use tacitly hints that the distribution of error may be normal. At each time level, the individual errors can be sorted into bins surrounding the mean. The error sorting domain is defined by the interval

$$(\mu - 3\sigma, \quad \mu + 3\sigma) \tag{293}$$

which is then divided into ten subintervals or bins. By counting the errors that fall into these bins, we can plot error as a distribution of sorts. At the time levels for this 1D simulation, the error counting bins have the membership graphed in Figure 5. In this Figure, the mean error is situated between bins 5 and 6. From what we see, errors are not truly centered around the mean. There are signs of skewness in these distributions since the peak error count is shifted to the left into the region where the magnitude of the error is less than the mean value. For another time step, a peak is shifted to the right to slightly larger error. Still, the numerical results are excellent.

Now we consider the convergence of the linear solvers for this test problem. The coding used for this problem is designed to count the number of iterations used to converge the linear solver. The number of iterations used to converge the Gauss-Seidel (GS), standard Conjugate Gradient (CG) and three terms recurrence Conjugate Gradient (T3) algorithms are displayed in Figure 6. Note that in the tables, only a select number of time levels is shown to promote brevity. Note that the Gauss-Seidel method requires between 9 and 11 iterations for convergence throughout the entire range of time levels. On the other hand, the Conjugate Gradient method and the mathematically alike three terms recurrence form require up to 19 iterations at solution start, but quickly decrease to 5 iterations by the 23^{rd} time level. By the problem end, no more that 3 iterations are required per time step. In this sense, the standard conjugate Gradient method offers improved convergence for the equiva-

Table 1: Relative error data (%) calculated for numerical solutions of the 1D test problem on the uniform grid

Time	Min |Error| (%)	Max |Error| (%)	μ (%)	σ (%)
0.02	1.0705×10^{-3}	1.6726×10^{-1}	4.5112×10^{-2}	6.5559×10^{-2}
0.04	1.0187×10^{-3}	8.9335×10^{-2}	3.9434×10^{-2}	4.0650×10^{-2}
0.06	5.1615×10^{-4}	8.9622×10^{-2}	2.9575×10^{-2}	4.1906×10^{-2}
0.08	1.7320×10^{-3}	6.0669×10^{-2}	1.6911×10^{-2}	2.9399×10^{-2}
0.10	1.8792×10^{-3}	3.3086×10^{-2}	6.7445×10^{-3}	1.7217×10^{-2}
0.12	8.9569×10^{-4}	1.3671×10^{-2}	-1.4789×10^{-4}	8.9530×10^{-3}
0.14	6.5893×10^{-4}	1.0363×10^{-2}	-4.4196×10^{-3}	4.4272×10^{-3}
0.16	1.6387×10^{-3}	1.0852×10^{-2}	-6.8684×10^{-3}	3.0787×10^{-3}
0.18	1.2887×10^{-3}	1.1731×10^{-2}	-8.0017×10^{-3}	3.2772×10^{-3}
0.20	1.1604×10^{-3}	1.2664×10^{-2}	-8.5143×10^{-3}	3.5661×10^{-3}
0.22	1.0799×10^{-3}	1.2496×10^{-2}	-8.4160×10^{-3}	3.6706×10^{-3}
0.24	1.1778×10^{-3}	1.2252×10^{-2}	-8.1496×10^{-3}	3.5013×10^{-3}

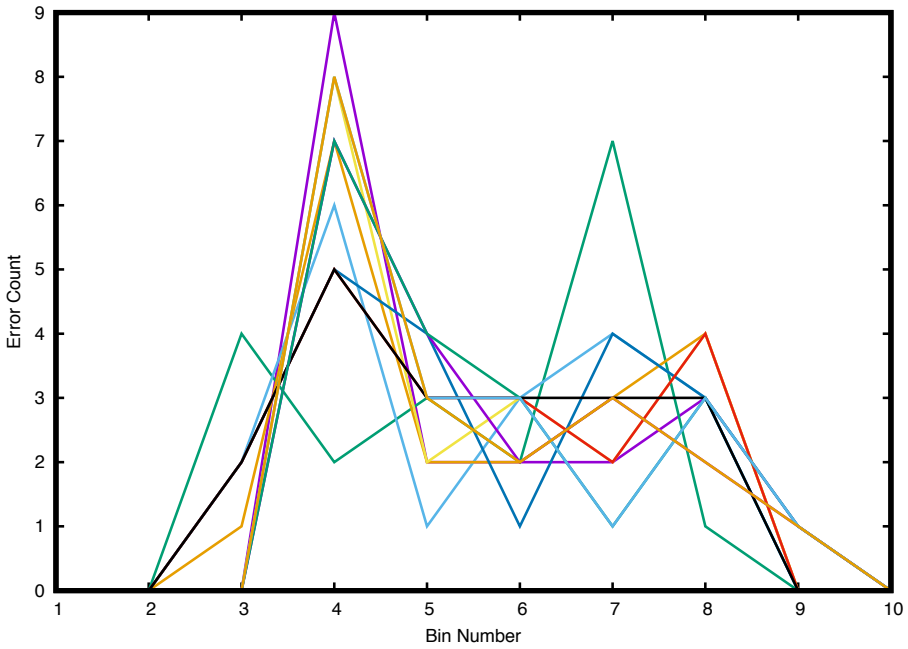

Figure 5: Distribution of error for the 1D test problem on the uniform grid

Table 2: Execution Times in Seconds for the 1D Test Problem on the Uniform Grid

Method	Execution Time (s)
Gauss-Seidel	0.1339
Conjugate Gradient	0.0614
3 Terms Recurrence CG	0.0619

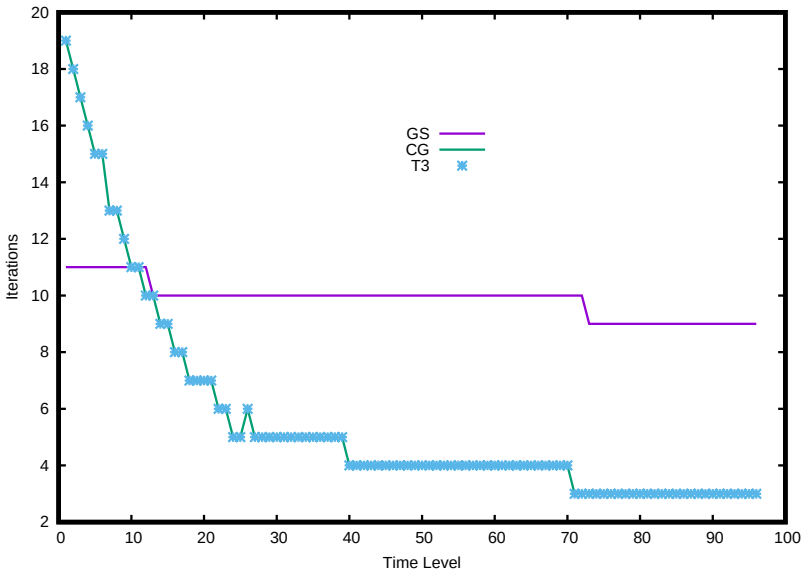

Figure 6: Iterations required to converge the GS, CG and T3 linear solvers displayed versus time levels for the 1D test problem on the uniform grid

lent level of accuracy.

We can spend a moment discussing code execution time. This 1D problem does not require much CPU time because it is a very small system consisting of 21 grid points with associated array storage. Still, it is worth mentioning since different linear solvers are employed for comparison. The times required for computer execution are listed in Table 2. As it happens, the two Conjugate Gradient methods CG and T3 have a clear edge in faster execution.

4.2 Results Computed on the Non-Uniform Grid

The same 1D problem is solved on a non-uniform grid in order to test the Biconjugate Gradient (BC) algorithm. When the grid is made so that it is no longer uniform, the resulting coefficient matrix for the finite difference equation is non-symmetric. This is an important test because, in general, coefficient matrices for PDE difference equations are not symmetric voiding the assumptions behind both the Gauss-Seidel and standard Conjugate Gradient methods. The grid employed in this case is generated by finite geometric series, a method described in the Appendix. The statement of the IBVP and the grid point count remain the same. Figures 7, 8 and 9 show numerical solutions on the non-uniform grid at times 0.02, 0.04 and 0.08, respectively. Visually, there is good agreement between the numerical and exact solutions at each time level. To provide greater detail on accuracy, the relative error (%) is computed and presented in Table 3. The error calculations show that the agreement between numerical and exact solutions is excellent with the largest error being on the order of one-third percent. On the whole, these errors occurring on the non-uniform grid exceed those for the uniform grid. This occurrence is to be expected because we know that accuracy is lost in the derivation of the difference equations on the non-uniform grid. Again, on the average, errors still decrease as time advances and the solution closes on the steady state.

As in the preceding test, the mean and standard deviation of error have been estimated for the non-uniform grid problem. The bin structure for error storage is cast over the domain with three standard deviations to the left and right of the mean error. The error bins are plotted for each time level in Figure 10. The bin structure demonstates that the mean error is shifted to the left and right into the regions of decreased and increased error, respectively. Given the loss in accuracy endemic to the non-uniform grid, this result is expected.

The number of iterations required for convergence of the linear solvers is shown in Figure 11. The performance of the Biconjugate Gradient method is denoted by the abbreviation BC. Numerical solutions have also been computed by using the GS, CG and T3 methods. Although these three methods are designed for symmetric matrices, these algorithms also converge for this test case, so the accuracy data is a basis. As is shown in Figure 11, the BC algorithm requires more iterations to converge to the required level of accuracy. This algorithm requires roughly twice the number of iterations required for the standard conjugate gradient method. The number of iterations motivates an examination of computer execution time.

Computer execution times, measured for the CPU, are rendered in Table 4. We see that the CG and T3 algorithms are, on the whole, faster. The GS algorithm requires about twice as much CPU time to converge. The BC algorithm requires about one and one-half times the amount of time to converge as does the standard CG algorithm. Even with the reduced accuracy afforded by the non-uniform grid, the results generated by the BC algorithm are excellent.

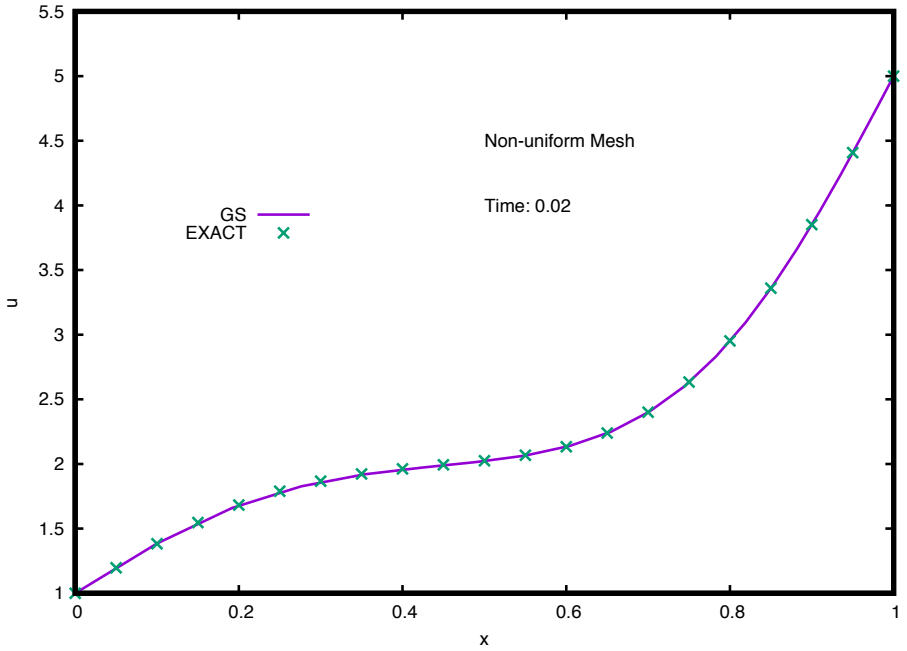

Figure 7: Time 0.02 Solution for the 1D Non-uniform Grid Test Problem using the Biconjugate Gradient Solver

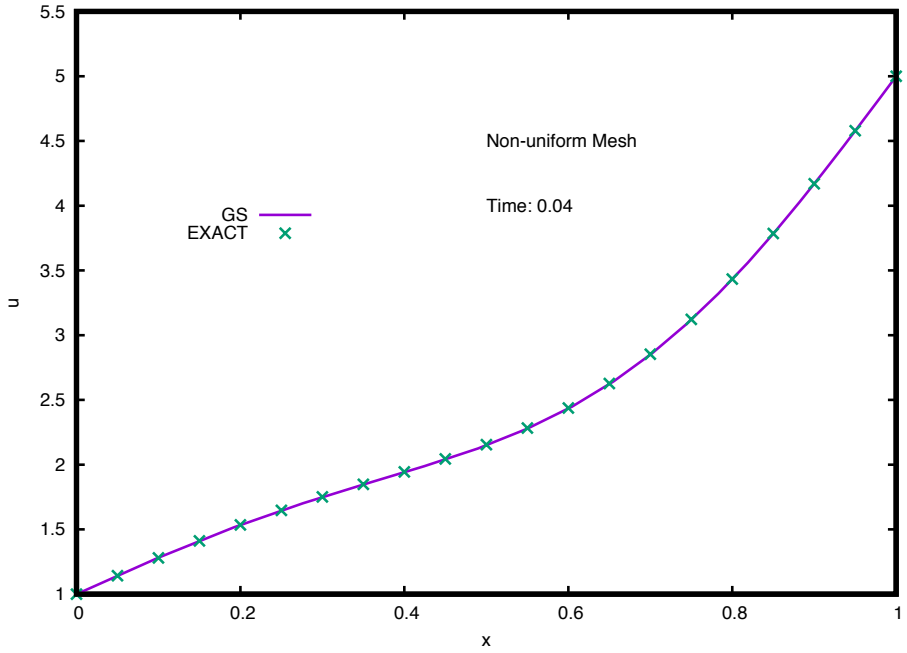

Figure 8: Time 0.04 Solution for the 1D Non-uniform Grid Test Problem using the Biconjugate Gradient Solver

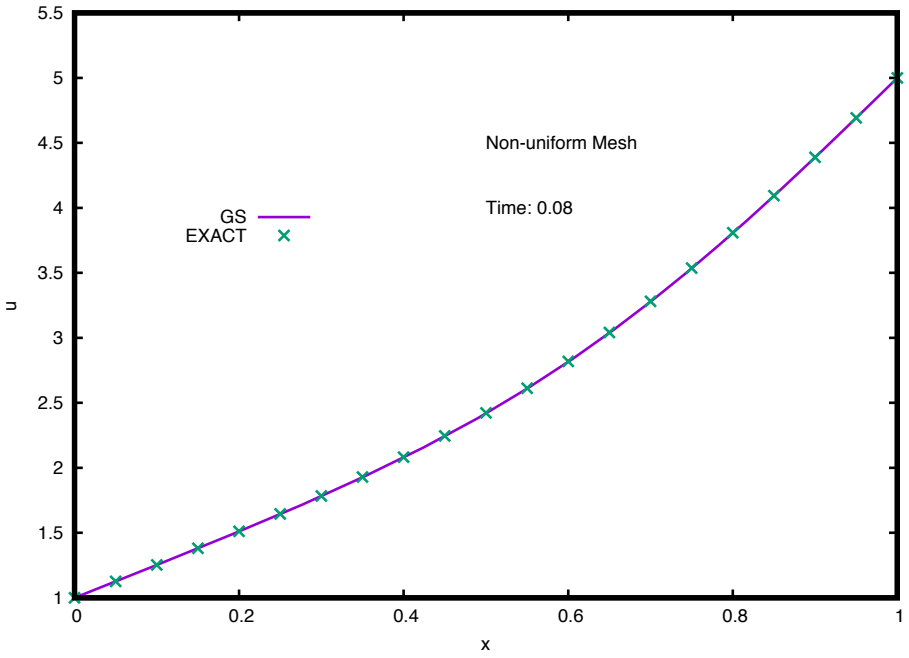

Figure 9: Time 0.08 Solution for the 1D Non-uniform Grid Test Problem using the Biconjugate Gradient Solver

Table 3: Data for relative error (%) calculated for numerical solutions of the 1D test problem computed on the non-uniform grid

Time	Min |Error| (%)	Max |Error| (%)	μ (%)	σ (%)
0.02	4.7208×10^{-3}	3.5556×10^{-1}	-1.0461×10^{-1}	1.1233×10^{-1}
0.04	3.9556×10^{-3}	2.2283×10^{-1}	-8.6648×10^{-2}	9.2346×10^{-2}
0.06	4.5664×10^{-3}	1.8991×10^{-2}	-7.8705×10^{-2}	7.2258×10^{-2}
0.08	4.3884×10^{-3}	1.5496×10^{-1}	-7.4565×10^{-2}	5.1476×10^{-2}
0.10	3.5258×10^{-3}	1.3537×10^{-1}	-6.9916×10^{-2}	4.4545×10^{-2}
0.12	2.9284×10^{-3}	1.2819×10^{-1}	-6.4275×10^{-2}	4.3187×10^{-2}
0.14	2.4217×10^{-3}	1.2285×10^{-1}	-5.8176×10^{-2}	4.1376×10^{-2}
0.16	2.1957×10^{-3}	1.1427×10^{-1}	-5.2164×10^{-2}	3.8311×10^{-2}
0.18	1.9257×10^{-3}	1.0298×10^{-1}	-4.6341×10^{-2}	3.4449×10^{-2}
0.20	1.7311×10^{-3}	9.0971×10^{-2}	-4.0927×10^{-2}	3.0423×10^{-2}
0.22	1.4024×10^{-3}	7.9715×10^{-2}	-3.6007×10^{-2}	2.6618×10^{-2}
0.24	1.2998×10^{-3}	6.9421×10^{-2}	-3.1572×10^{-2}	2.3154×10^{-2}

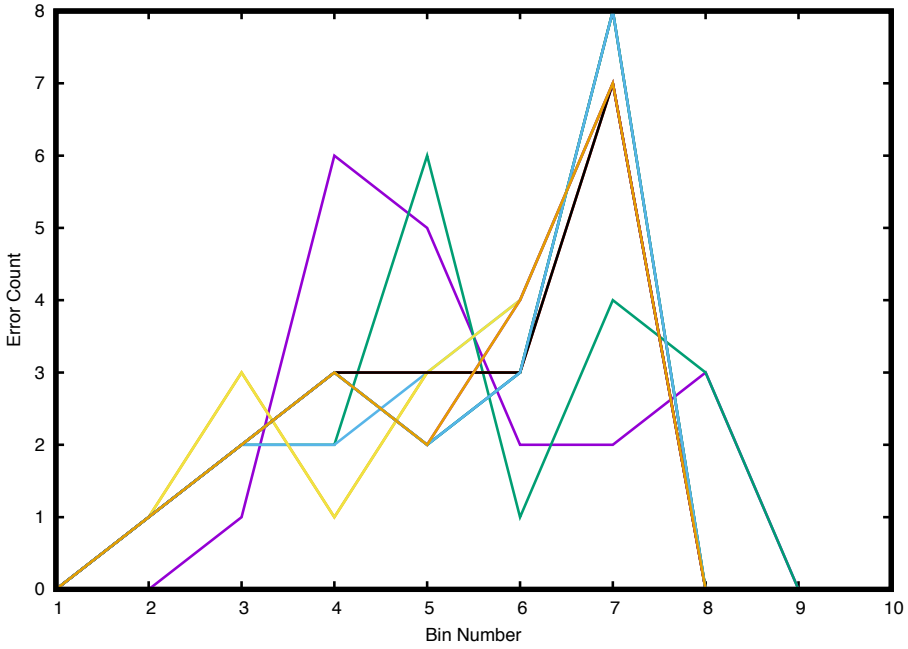

Figure 10: Distribution of error for the 1D test problem on the non-uniform grid

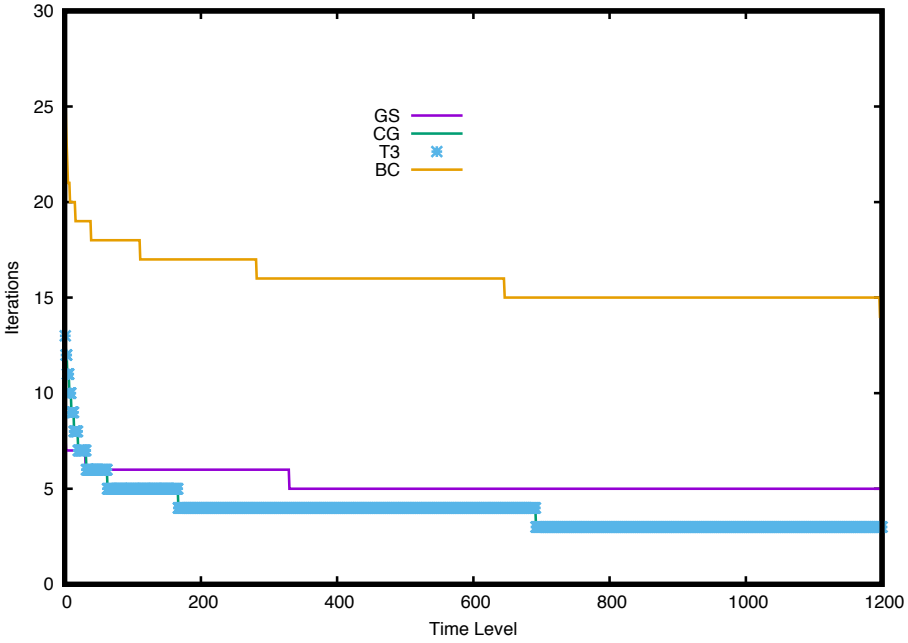

Figure 11: Iterations required to converge the GS, CG and T3 linear solvers displayed versus time levels on the non-uniform grid

Table 4: Execution times in seconds for the 1D test problem on the non-uniform grid

Method	Execution Time (s)
Biconjugate Gradient	0.7321
Gauss-Seidel	1.2286
Conjugate Gradient	0.5943
3 Terms Recurrence CG	0.5840

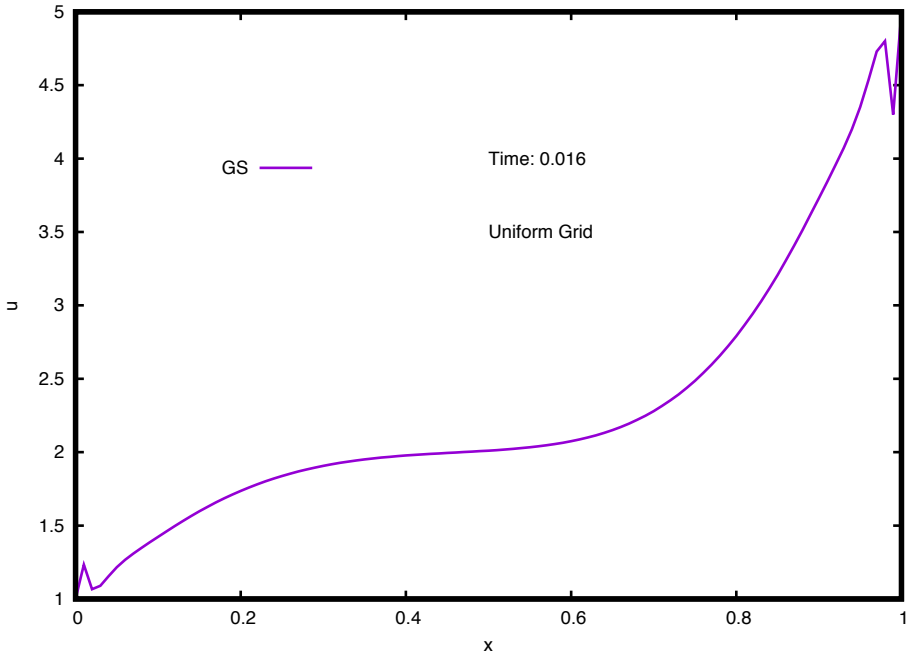

Figure 12: Time 0.016 GS solution demonstrating the Crank-Nicolson oscillatory pathology on a uniform grid

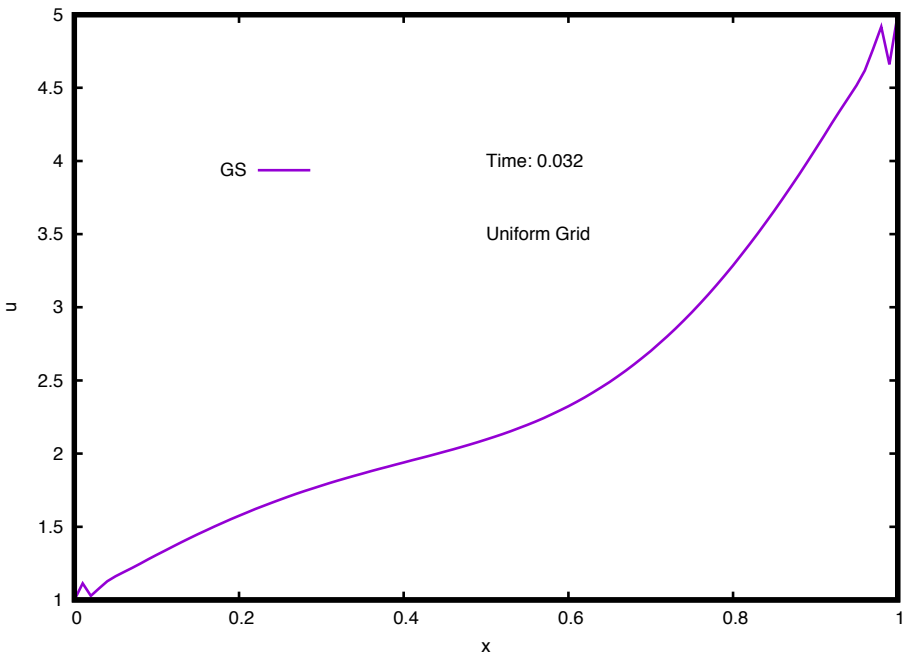

Figure 13: Time 0.032 GS solution demonstrating the Crank-Nicolson oscillatory pathology on a uniform grid

4.3 A Pathology Associated with the Crank-Nicolson Discretization

A placard result associated with the Crank-Nicolson implicit method is that this algorithm is unconditionally stable for all positive mesh ratios.[2] It is worthwhile to revisit the concept of stability; that in itself is a primary concept associated with the convergence of a given numerical scheme. Stability is the situation where the difference between the numerical and exact solutions remains bounded as time advances.[11] It is a fine point, but stability does not mean that a erroneous oscillation cannot exist for a certain amount of time in the course of the solution. Normally, one would expect that a error should decay with time. As it happens, Crank-Nicolson can support a substantial error in the numerical solution over a large number of time steps for stable values of the mesh ratio r. This fact is kind of disappointing, and I was unaware of its existence until I began work on this research project. In fact, I noticed seemingly unstable behavior at values of $r > 1$. I originally thought that there was an error in my computer code until I reduced r below one and saw the error disappear. A search through the literature reveals comments concerning this erroneous solution behavior.[11] Rather than merely mention this problem, it is better to show its behavior. Consider the 1D heat conduction problem on the uniform grid with a spatial step size of 0.01. The associated mesh ratio r is 20. Numerical solutions using the Gauss-Seidel linear solver are shown in Figures 12, 13 and 14 to illustrate this oscillatory numerical pathology. Oscillations are visible at the domain endpoints because the boundary conditions introduce "impulsive" changes to the field. This prompt change spurs the numerical oscillations, the largest excursions appear in Figure 12. The results in the subsequent two figures tend to confirm stability of Crank-Nicolson since the magnitude of the oscillations is seen to decrease

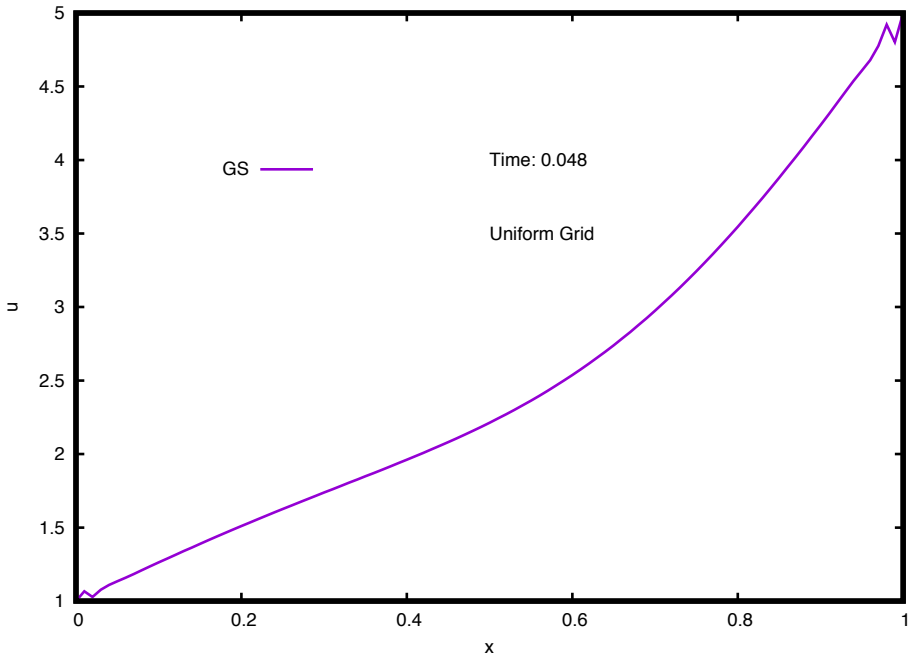

Figure 14: Time 0.048 GS solution demonstrating the Crank-Nicolson oscillatory pathology

in time. I found this behavior intuitively difficult to accept, but the stability of the scheme is preserved since erroneous excursions in the solution do not grow in time.

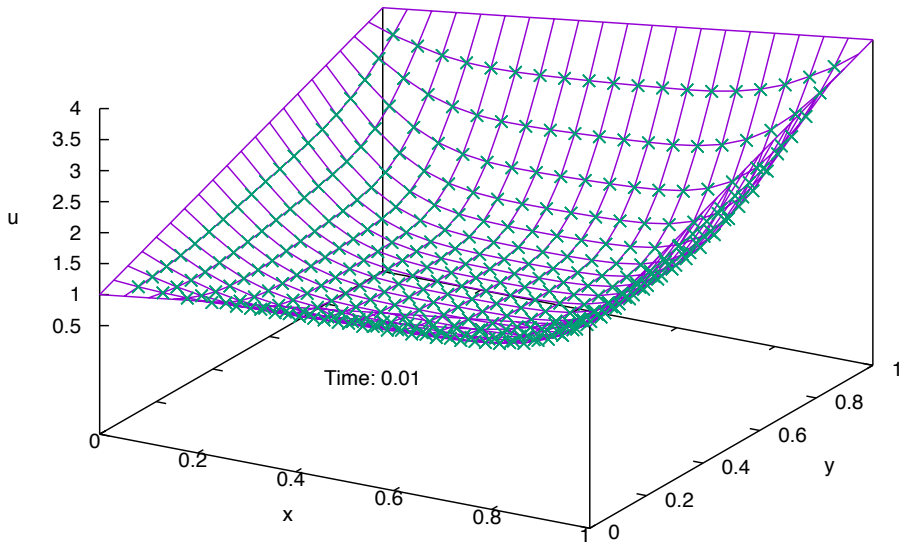

Figure 15: Time 0.01 CG solution for the 2D test problem on the uniform grid; Magenta Lines - Numerical, X - Exact

5 Numerical Results for the 2D Test Problem

Numerical solutions are presented in this section for the two dimensional test problem. The IBVP in this case is described as follows. For the exact solution, 100 Fourier modes are utilized.

$$\begin{aligned} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \quad 0 < x < 1,\ 0 < y < 1,\ t > 0 \\ u(0, y, t) = 1 + 2y, \quad 0 < y < 1 \\ u(1, y, t) = 2 + 2y, \quad 0 < y < 1 \\ u(x, 0, t) = 1 + x, \quad 0 < x < 1 \\ u(x, 1, t) = 3 + x, \quad 0 < x < 1 \\ u(x, y, 0) = 0.5 \end{aligned}$$

The Gauss-Seidel (GS), Standard Conjugate Gradient (CG) and Three Terms Recurrence Conjugate Gradient (T3) linear solvers are employed to solve the linear system constructed at each time step. Of course, the problem is set up on the unit domain.

5.1 Results for the 2D Test Problem Computed on the Uniform Grid

The grid resolution used for the problem is kept the same as used in the 1D test problem. Twenty-one (21) grid points are used in the x and y directions. As a result, the x and y mesh ratios r_x and r_y, respectively, are equal. Plots of the numerical solution versus the

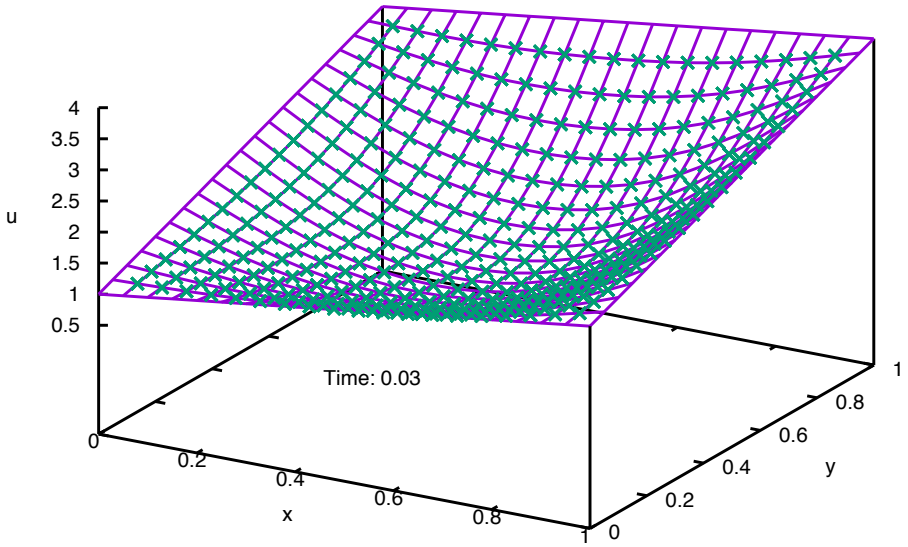

Figure 16: Time 0.03 CG solution for the 2D test problem on the uniform grid; Magenta Lines - Numerical, X - Exact

exact solution are shown in Figures 15, 16 and 17 at times 0.01, 0.03 and 0.09, respectively. Visually, the agreement between numerical and exact solutions is very good. To provide more detailed information on error, the minimum, maximum, mean and standard deviation of relative error in percent is calculated and displayed in Table 5. Absolute error maximizes at about two percent, an acceptable amount considering this course grid. As in the previous test case, it is desirable to investigate the distribution of error. Since we have a substantially higher number of field points than in the 1D test problem, we sort the error values into 20 bins covering the region $(\mu - 3\sigma, \mu + 3\sigma)$ at each time level. The counting of error values are recorded versus bin number in Figure 18 for all time levels. In this figure, the behavior of error is clear. The peak number of errors occur to the left and right of the mean value, so the distribution of error is not truly normal. Overall, the data above demonstrates that the agreement between numerical and exact solutions is excellent. Although the bin sorting data is shown for the CG algorithm, the GS and T3 methods converge and produce identical data. Hence, we have a reasonably good picture of numerical error for all three algorithms tested on this test problem.

Another topic of importance is whether or not our linear solvers are converging in an expeditious manner. In the course of the solution process, the number of iterations are counted at each time step for each of the solvers GS, CG and T3. These results are displayed in Figure 19. In this plot, the mathematical near equivalence of the standard Conjugate Gradient method (CG) with the Three Terms Recurrence (T3) form becomes clear. Indeed, fewer iterations are required to converge these two methods than are needed for the Gauss-Seidel (GS) algorithm. The Gauss-Seidel method requires on the order of ten to twenty-five iterations more than the conjugate gradient procedures. The difference in performance is fairly clear rendering an advantage for the use of the conjugate gradient approach and perhaps

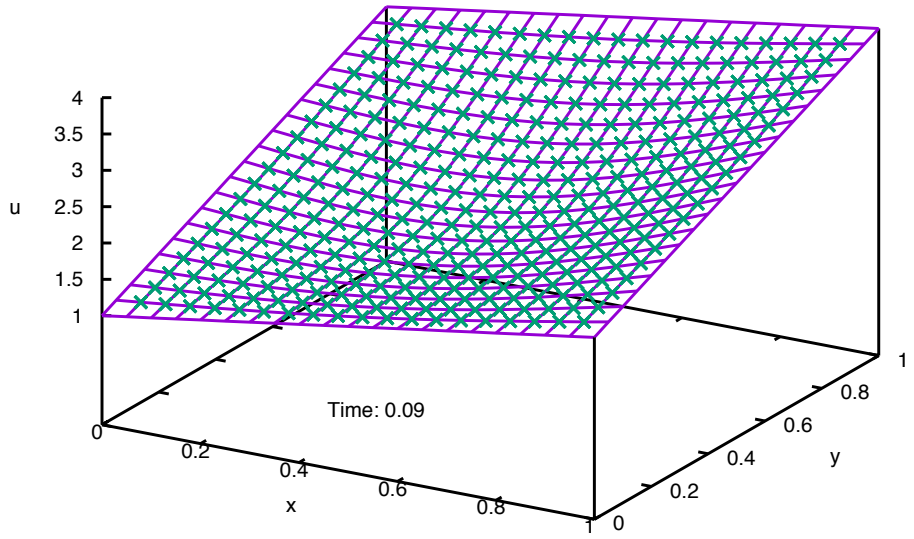

Figure 17: Time 0.09 CG solution for the 2D test problem on the uniform grid; Magenta Lines - Numerical, X - Exact

Table 5: Relative error data (%) calculated for numerical solutions of the 2D test problem on the uniform grid

Time	Min |Error| (%)	Max |Error| (%)	μ (%)	σ (%)
0.01	0.83332×10^{-1}	0.19279×10^{1}	0.63799	0.42180
0.02	0.21800×10^{-2}	0.13671×10^{1}	0.32659	0.35155
0.03	0.11407×10^{-3}	0.85988	0.18791	0.21565
0.04	0.12097×10^{-3}	0.48062	0.11620	0.12380
0.05	0.19401×10^{-3}	0.27413	0.74805×10^{-1}	0.72639×10^{-1}
0.06	0.24331×10^{-3}	0.16230	0.48508×10^{-1}	0.43845×10^{-1}
0.07	0.11060×10^{-3}	0.97244×10^{-1}	0.30903×10^{-1}	0.26785×10^{-1}
0.08	0.13046×10^{-4}	0.57457×10^{-1}	0.18868×10^{-1}	0.16145×10^{-1}
0.09	0.56698×10^{-5}	0.33270×10^{-1}	0.10577×10^{-1}	0.93055×10^{-2}
0.10	0.30382×10^{-5}	0.16842×10^{-1}	0.49291×10^{-2}	0.49405×10^{-2}
0.11	0.43506×10^{-5}	0.69452×10^{-2}	0.11219×10^{-2}	0.23257×10^{-2}
0.12	0.00000	0.46807×10^{-2}	-0.13930×10^{-2}	0.14854×10^{-2}
0.13	0.37365×10^{-4}	0.72432×10^{-2}	-0.29722×10^{-2}	0.19588×10^{-2}
0.14	0.17685×10^{-3}	0.90350×10^{-2}	-0.39112×10^{-2}	0.24820×10^{-2}

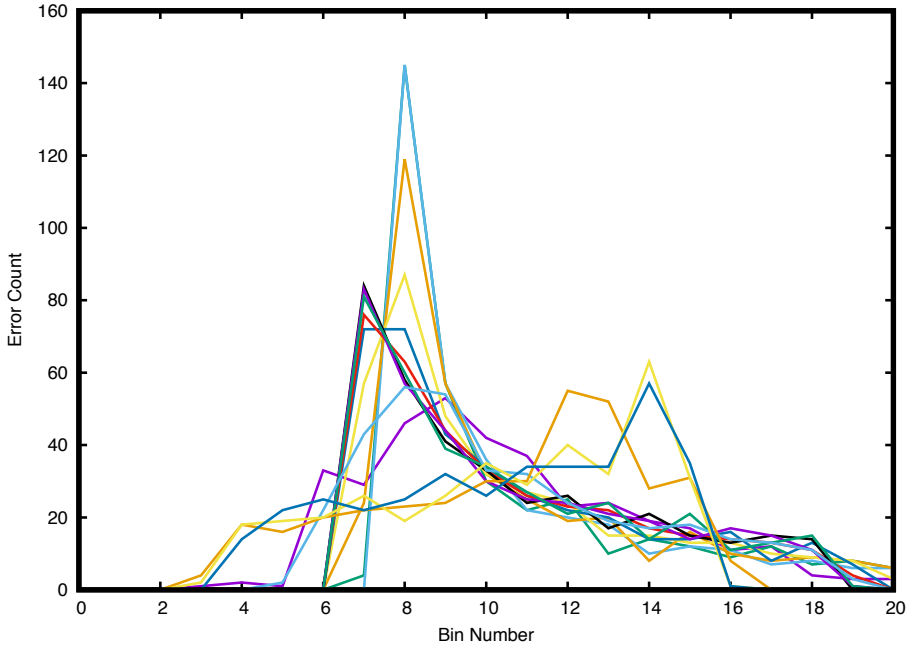

Figure 18: Distribution of error for the 2D test problem on the uniform grid

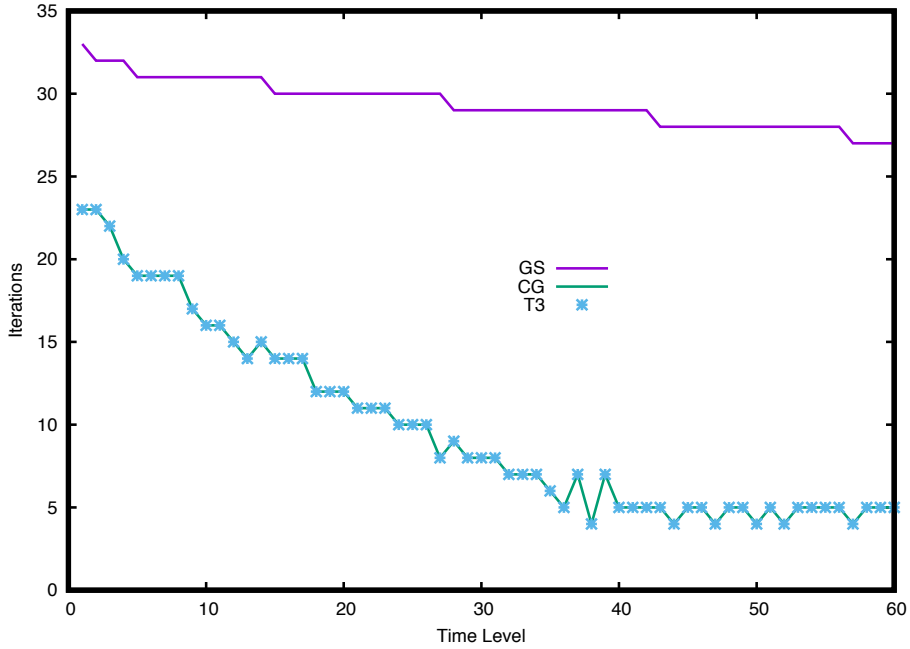

Figure 19: Iterations required to converge the GS, CG and T3 linear solvers displayed versus the time level for the 2D Test Problem on the uniform grid

Table 6: Execution times in seconds for the 2D test problem on the uniform grid

Method	Execution Time (s)
Gauss-Seidel	3.3063
Conjugate Gradient	0.1105
3 Terms Recurrence CG	0.1187

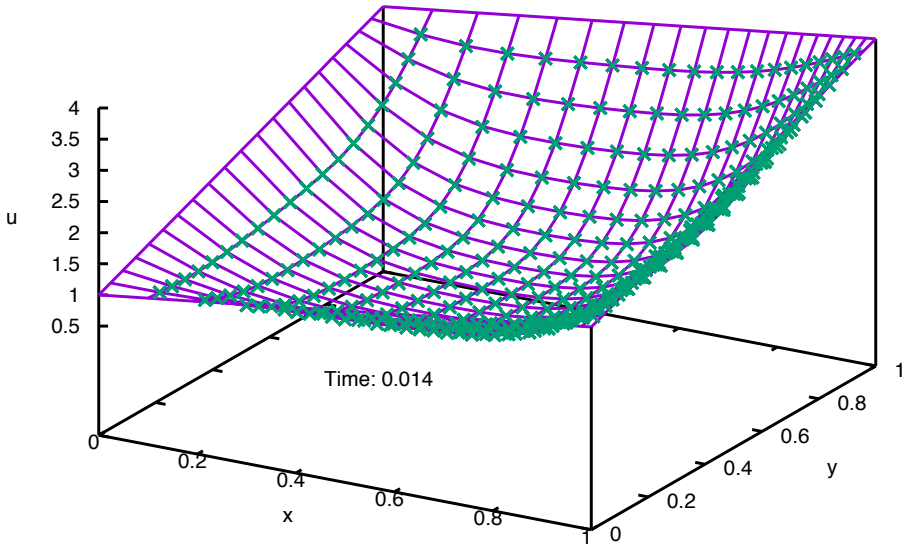

Figure 20: Time 0.014 BC solution for the 2D test problem on the non-uniform grid; Magenta Lines - Numerical, X - Exact

to Krylov space methods in general. As a final issue, the timing information for the 2D problem on the uniform grid is provided in Table 6. It is a stark result, but the Gauss-Seidel execution time exceeds conjugate gradient execution times by a factor of thirty.

5.2 Results for the 2D Test Problem Computed on the Non-uniform Grid

Pursuant to an interest in solving non-symmetric matrices, it is fruitful to solve the 2D test problem on a non-uniform grid. Recall that the difference relation for the non-uniform grid produces coefficients that assemble to form a non-symmetric matrix. This type of matrix motivates testing the Biconjugate Gradient (BC) method, a conjugate gradient method requiring about twice the work of the regular CG algorithm. It is capable of solving a non-symmetric linear system by constructing two sets of orthogonal conjugate vectors.[19] One set of vectors corresponds to the original matrix while the other corresponds to the matrix transpose (for matrices with real components). The non-uniform grid is generated by using a finite geometric series along the x coordinate. In the y direction, a parabolic stretching function is employed. Both of these grid methods are described in the Appendix.

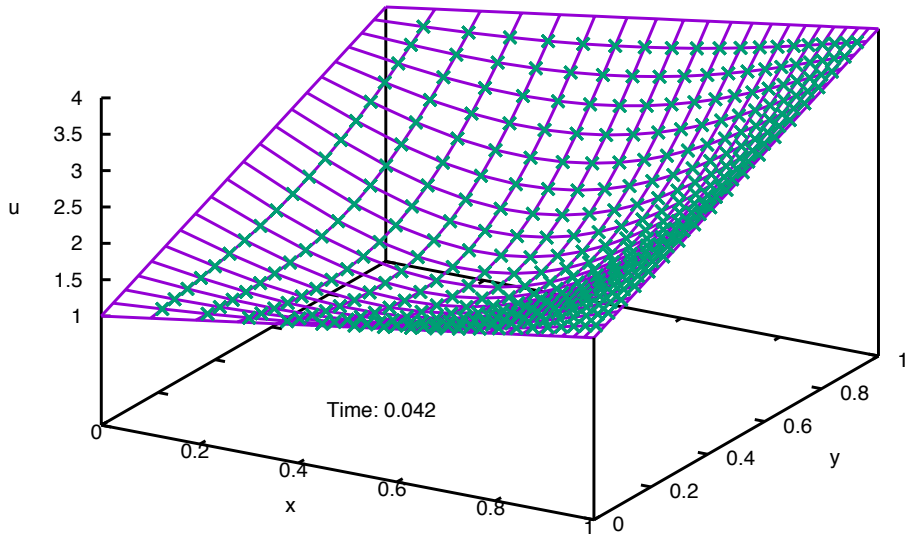

Figure 21: Time 0.042 BC solution for the 2D test problem on the non-uniform grid; Magenta Lines - Numerical, X - Exact

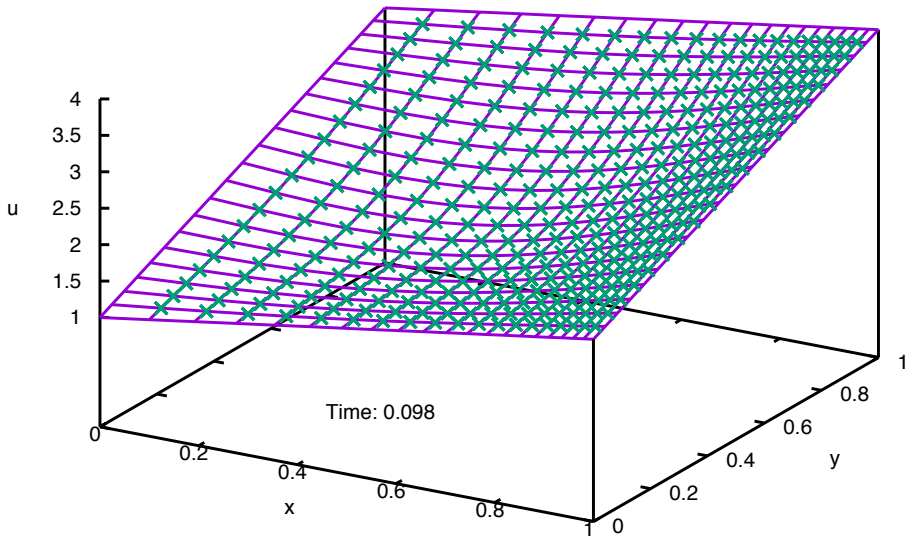

Figure 22: Time 0.098 BC solution for the 2D test problem on the non-uniform grid; Magenta Lines - Numerical, X - Exact

Table 7: Relative error data (%) calculated for numerical solutions of the 2D test problem on the non-uniform grid

Time	Min |Error| (%)	Max |Error| (%)	μ (%)	σ (%)
0.028	0.32398×10^{-3}	0.14674×10^{1}	0.14975	0.31446205
0.042	0.59237×10^{-4}	0.54039	0.75740×10^{-1}	0.12122061
0.056	0.25209×10^{-3}	0.20067	0.40038×10^{-1}	$0.49441833 \times 10^{-1}$
0.070	0.15147×10^{-4}	0.83576×10^{-1}	0.19096×10^{-1}	$0.24704270 \times 10^{-1}$
0.084	0.00000	0.44809×10^{-1}	0.63419×10^{-2}	$0.17664217 \times 10^{-1}$
0.098	0.13164×10^{-3}	0.49143×10^{-1}	-0.11606×10^{-2}	$0.15651004 \times 10^{-1}$
0.112	0.13094×10^{-4}	0.49869×10^{-1}	-0.53150×10^{-2}	$0.14319981 \times 10^{-1}$
0.126	0.48777×10^{-4}	0.46926×10^{-1}	-0.73306×10^{-2}	$0.12876188 \times 10^{-1}$
0.140	0.54298×10^{-5}	0.41986×10^{-1}	-0.80312×10^{-2}	$0.11308550 \times 10^{-1}$

Table 8: Execution times in seconds for the 2D test problem on the non-uniform grid

Method	Execution Time (s)
Biconjugate Gradient	0.6626
Gauss-Seidel	8.9850
Conjugate Gradient	0.5644
3 Terms Recurrence CG	0.5650

The 2D test problem is solved on the non-uniform grid by using the BC method. The results are shown in Figures 20, 21 and 22 for solution times 0.014, 0.042 and 0.098, respectively. At a glance, the results agree with the exact solution quite well. A more thorough examination of error is provided in Table 7 where the maximum, minimum, mean and standard deviation for relative error (percent) are listed. The maximum error encountered during the numerical solution is less than two percent while the mean error is no more than one fifth of a percent. For this reason, we can conclude that the agreement between the numerical and exact solutions is quite good. Of course, it is worthwhile to examine the distribution of error. As in the 2D uniform mesh test case, we divide the error range $(\mu - \sigma, \mu + \sigma)$ into 20 bins and sort the errors exhibited at each time step into those bins. With the number of field points used in this problem, it is best to examine these distributions as graphs of bin count versus bin number. This information is plotted in Figure 23. Plots like this are interesting because the mean error exists between bins ten and eleven. As in the preceding cases, the maximum error count is not located at the mean. Depending upon the time level, the maximum error counts exist largely to the left and right of the center bin. For this reason, we can conclude that the error is not normally distributed.

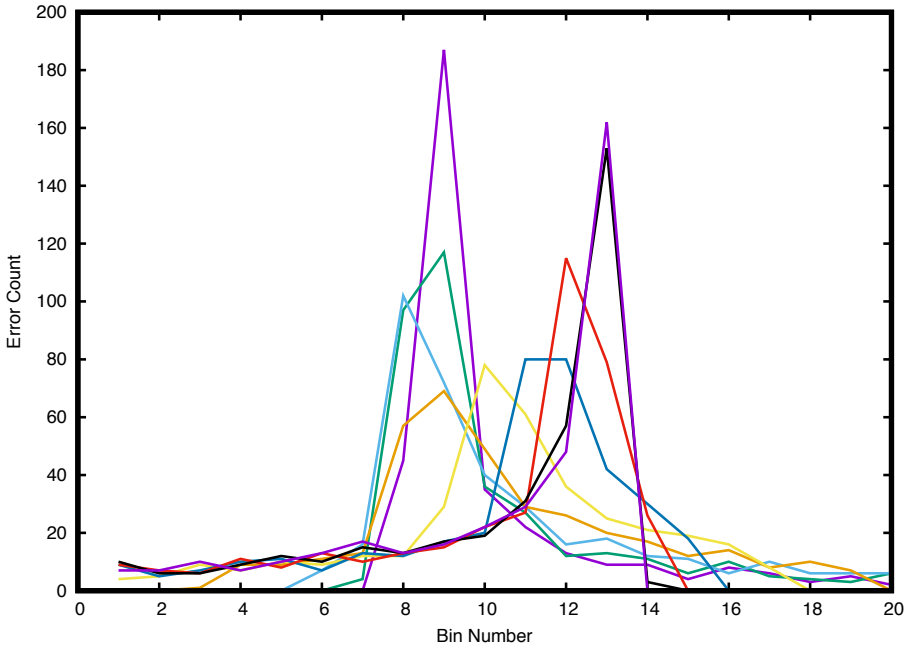

Figure 23: Distribution of error for the 2D test problem on the non-uniform grid

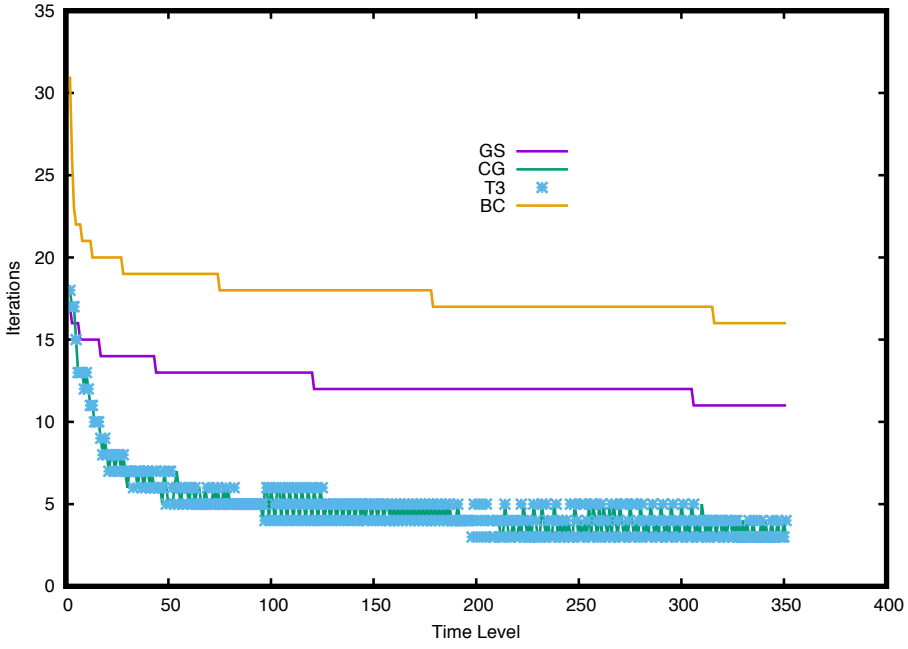

Figure 24: Iterations required to converge the GS, CG and T3 linear solvers displayed versus the time level for the 2D test problem on the non-uniform grid

It is also important to examine the number of iterations required at each time step for the linear system solvers tested here. This information is conveyed in Figure 24. In this instance, the Gauss-Seidel solver exceeds the performance of Biconjugate Gradient solver. It is interesting result, but there is substantially more work required by the BC solver at each iteration. A final area of problem analysis is the consideration of computer execution time. As it happens, the other algorithms (GS, CG and T3) for symmetric positive definite matrices have been successfully applied to this 2D non-uniform grid problem. The associated CPU times are included in Table 8. From the data shown, the Gauss-Seidel algorithm requires the most execution time while the standard Conjugate Gradient and Three Terms Recurrence CG method require about 1/16 of the GS CPU time. Surprisingly enough, the BC method requires only an additional tenth of second more CPU time that the basic CG and T3 methods.

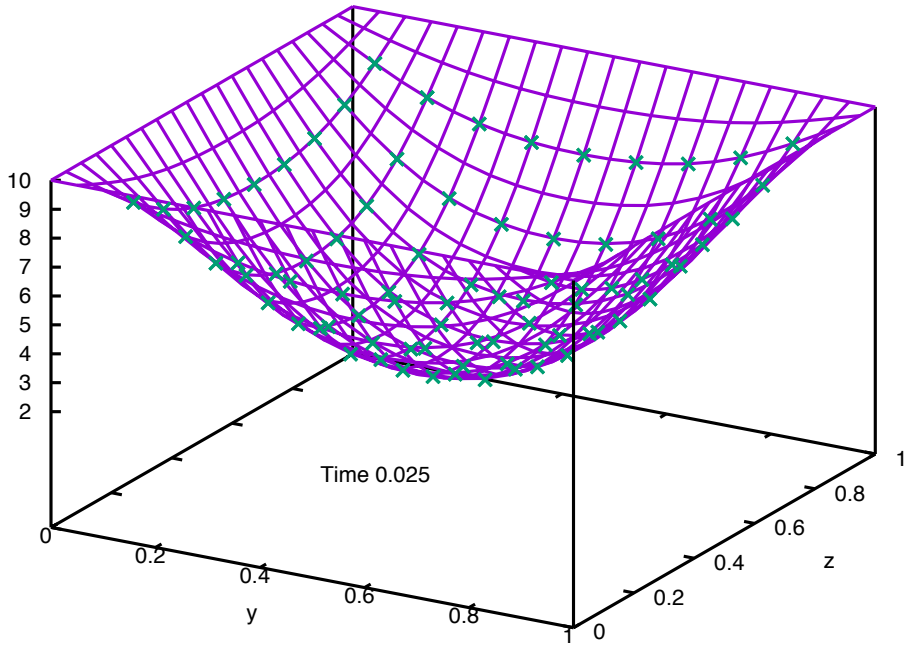

Figure 25: Time 0.025 GS solution for the third test problem on the uniform grid; Magenta Lines - Numerical, X - Exact

6 Numerical Results for the 3D Test Problem

Recall that the 3D test problem is defined as

$$\begin{aligned}
\frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}, \quad 0 < x, y, z < 1, \ t > 0 \\
u(x, y, 0, t) &= u(x, y, 1, t) = T_2, \quad 0 \leq x, y \leq 1 \\
u(0, y, z, t) &= u(1, y, z, t) = T_2, \quad 0 \leq y, z \leq 1 \\
u(x, 0, z, t) &= u(x, 1, z, t) = T_2, \quad 0 \leq x, z \leq 1 \\
u(x, y, z, 0) &= T_1, \quad 0 < x, y, x < 1,
\end{aligned}$$

for solution $u(x, y, z, t)$.

This, the third and final test problem, is cast on the unit cube. The uniform grid resolution is kept the same by using a 21 x 21 x 21 grid. Given the satisfactory performance of the algorithms on the previous test cases, no simulations are conducted on a non-uniform grid. As in earlier cases, 100 Fourier modes are used to generate the exact solution.

Three dimensional configurations are more difficult to visualize, so here the results are visualized on planes. The problem is quite symmetric, so we can choose a plane normal to any of the three Cartesian axes. Also, planes at the same distance along the normal axis show the same solution. Here, we show solutions at the middle station (i = 11) along the x axis. Solutions at times 0.025, 0.050 and 0.075 are presented in Figures 25, 26 and 27,

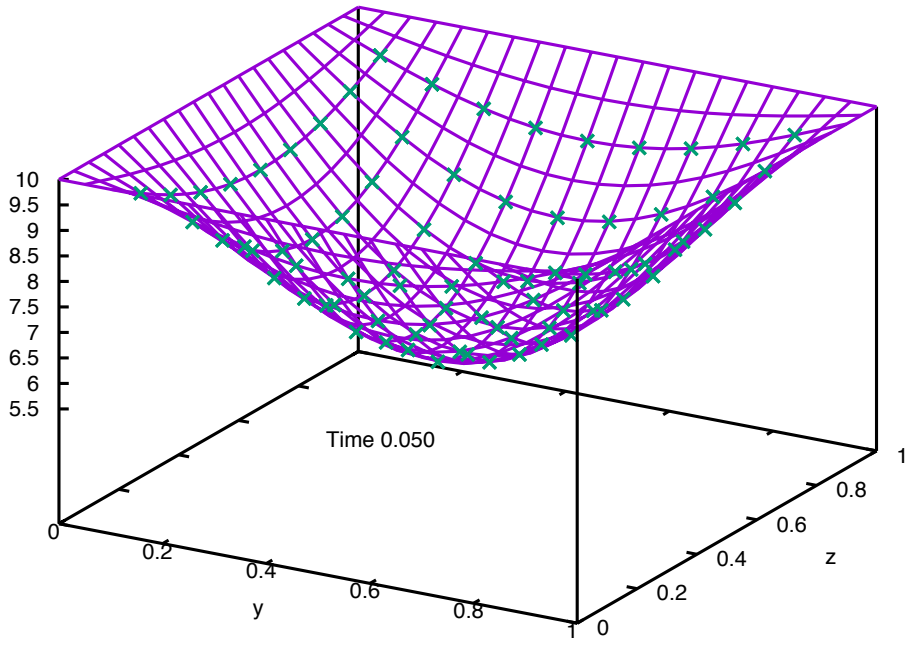

Figure 26: Time 0.050 GS solution for the third test problem on the uniform grid; Magenta Lines - Numerical, X - Exact

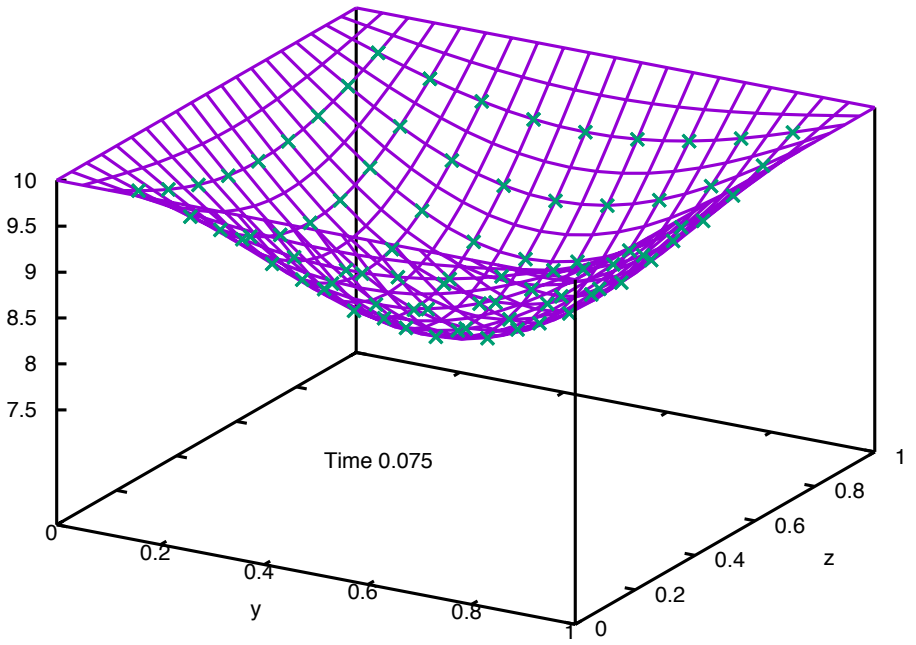

Figure 27: Time 0.075 GS solution for the third test problem on the uniform grid; Magenta Lines - Numerical, X - Exact

Table 9: Relative error data (%) calculated for numerical solutions of the 3D test problem on the uniform grid

Time	Min |Error| (%)	Max |Error| (%)	μ (%)	σ (%)
0.025	0.93439×10^{-2}	$0.21373 \times 10^{+1}$	0.30537	0.35630
0.050	0.30418×10^{-2}	0.31521	0.68972×10^{-1}	0.63270×10^{-1}
0.075	0.13692×10^{-2}	0.69918×10^{-1}	0.20005×10^{-1}	0.15210×10^{-1}
0.100	0.38609×10^{-3}	0.15687×10^{-1}	0.49772×10^{-2}	0.35070×10^{-2}
0.125	0.20027×10^{-5}	0.12231×10^{-2}	0.40941×10^{-3}	0.27600×10^{-3}
0.150	0.22014×10^{-4}	0.20375×10^{-2}	-0.71186×10^{-3}	0.47710×10^{-3}

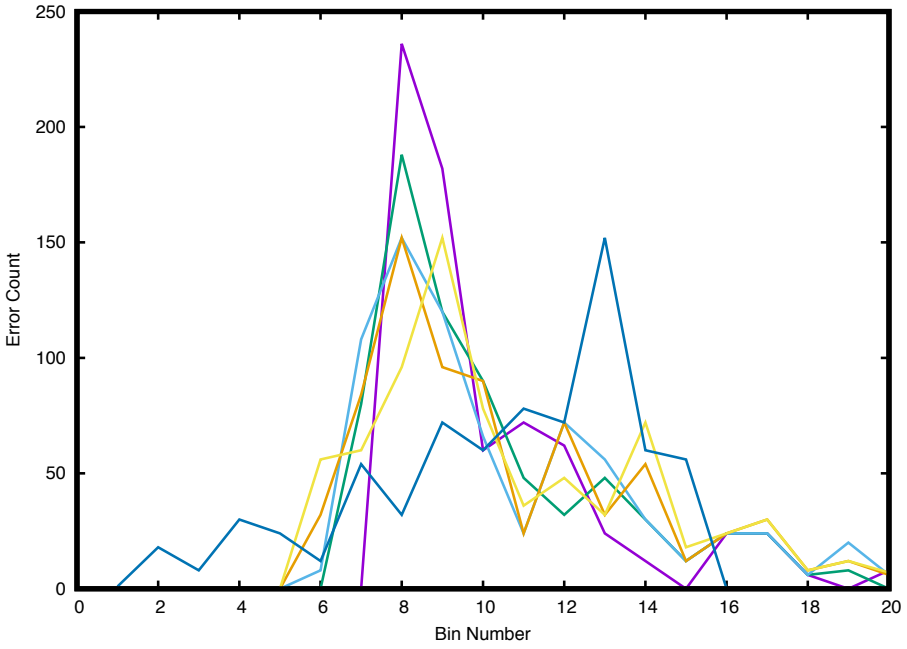

Figure 28: Distribution of error for the 3D test problem on the uniform grid

Table 10: Execution times in seconds for the 3D test problem on the uniform grid

Method	Execution Time (s)
Gauss-Seidel	124.17
Conjugate Gradient	2.0870
3 Terms Recurrence CG	1.9872

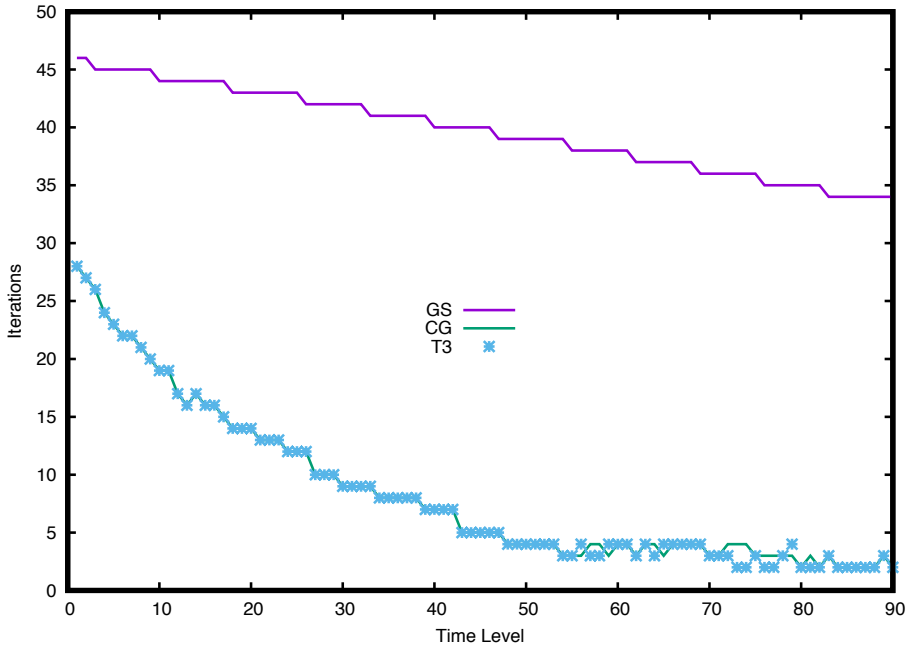

Figure 29: Iterations required to converge the GS, CG and T3 linear solvers displayed versus the time level for the 3D Test Problem on the uniform grid

respectively. The discrete points indicated by X are field points from the exact solution. The lines represent the numerical solution. A more detailed representation of relative error is rendered by direct calculation. The maximum and minimum of relative error (absolute value) along with mean and standard deviation of raw error is listed in Table 11. The maximum absolute error is just over two percent occurring at time 0.025. Remaining error figures are decidedly lower, so the numerical results show excellent agreement with the exact solution. The distribution of error is also of interest. In this case, relative error is sorted into 20 bins, and the error count in each bin is displayed versus bin number in Figure 28. This distribution has an appearance that more closely mimicks the normal distribution, but it still possesses some skewness and secondary peaks. The cause for its more normal appearance is mostly due to the symmetry that is used in the application of boundary conditions. The same, low level of error is exhibited for each of the solvers tested.

The number of iterations required for each solver to converge at a given time step is plotted in Figure 29. The performance of the conjugate gradient solvers is substantially better than the Gauss-Seidel solver. In the final time steps of the solution, the CG methods require less than five iterations in order to converge. In comparison, the GS solver requires over thirty iterations per time step. This fact has a pronounced impact on computer execution time. Code execution times for each solver are shown in Table 12. It is evident that the Gauss-Seidel solver requires far more CPU time than is needed by the conjugate gradient routines. In each case, each of these linear system solvers renders the same level of accuracy.

7 Conclusions

This report addresses an implicit scheme for the Heat Conduction equation and the linear system solver routines required to compute the numerical solution for this equation at each time step. The Crank-Nicolson implicit difference scheme is the canonical scheme that serves as the test bed for the Gauss-Seidel (GS), standard Conjugate Gradient (CG), Three Terms Recurrence Conjugate Gradient (T3) and the Biconjugate Gradient (BC) iterative methods. Exact solutions are constructed for three test problems and used to validate the numerical solutions. The test problems are constructed in 1D, 2D and 3D with uniform grids. Non-uniform grids are also constructed for the 1D and 2D problems to generate non-symmetric matrices for testing with the solvers. Numerical solutions are shown graphically in comparison with exact solutions, and direct calculations of error are performed and tabulated.

A principal thrust of this report is an examination of the Conjugate Gradient Method. Both the Gauss-Seidel and regular Conjugate Gradient method are intended for use with symmetric, positive definite matrices. Gauss-Seidel iteration is conceptually easy to understand based upon the basic structure of the matrix. On the other hand, the Conjugate Gradient and related Krylov Space methods are less intuitive. As is evidenced by this report, the proof of the minimal nature of the linear system solution is rather lengthy. The orthogonal structure of the sequence of residual vectors and search directions is even more arcane. Still, these methods are well worth the added study since they are efficient and equally applicable to large matrix systems. The finite termination property is also of great benefit. An added benefit is that non-symmetric system can be treated by the Biconjugate Gradient method. The method is effective, but it is a bit more temperamental in the selection of initial search directions. More advanced methods such as Conjugate Gradient Squared are expected to be effective for an even wider class of matrix systems.

The 1D test results are altogether satisfactory; the mean error is a fraction of one percent while the maxmimum error encountered in less than two percent. Numerical solutions are computed for a mesh ratio approximating unity. An oddly interesting finding is that the Crank- Nicolson scheme is subject to an odd oscillatory instability at higher mesh ratios exceeding unity. The strange aspect of this phenomenon is that the oscillations do not constitute a true instability; they diminish in time. Until working on this project, I was unfamiliar with this numerical pathology. A few plots illustrating these oscillations are included. The oscillations (as encountered in this work) occur near the boundaries where a discontinuity is induced by the boundary conditions. Another noteworthy observation is that if the mesh ratio is maintained less than unity, these oscillations do not occur. In terms of computer execution time and the number iterations required to converge the solvers at each time step, the Conjugate Gradient methods tend to require less time and fewer iterations than the Gauss-Seidel iterative scheme.

Results for the 2D and 3D problems also offer excellent results as evidenced by time dependent solution plots and direct error calculations. For the 2D problem on the uniform grid, the maximum error is less than two percent while the mean error is less than one percent. In most cases, the Conjugate Gradient methods require between ten and twenty fewer iterations

than the Gauss-Seidel method. Roughly one thirtieth of the CPU time is required for the Conjugate Gradient algorithm in comparison to Gauss-Seidel. For the 3D test problem on the uniform grid, the maximum error is less than three percent while the mean error is less than four tenths of a percent in the course of the solution. Conjugate Gradient methods require about one sixtieth of the CPU time as does the Gauss-Seidel scheme. Also, convergence of the Conjugate Gradient solvers require between one fifteenth and two-thirds of the iterations required by the Gauss-Seidel algorithm. Each of these iterative solvers provide excellent results for the test problems on the uniform grid.

Non-uniform grids are generated for both the 1D and 2D test problems. These grids result in the formation of non-symmetric difference coefficient matrices, a situation closer to real world problems. The Biconjugate Gradient solver is successfully tested for these problems. The Crank-Nicolson implicit scheme is also used in these cases. For the 1D problem, the maximum error is less than four tenths of a percent. The mean error is usually less than one tenth of a percent. For the 2D test problem, the maximum error is less than two percent while the mean error is less than two percent. The Biconjugate Gradient solver does require more iteration per time step than Gauss-Seidel, but these iterations execute at high speed making the overall execution time shorter than the Gauss-Seidel method. Overall, the Biconjugate Gradient method requires less execution time by a factor of fifteen.

Another area of interest addresses the nature of the error. Parameters such as mean error and the standard deviation of the error are estimated with standard formulas. Unfortunately, the use of these parameters tacitly implies the presence of the normal distribution. This issue begs the question; how is the error distributed? I do not present a rigorous examination of this idea, but the error for each problem is sorted into bins across the range of three standard deviations on each side of the mean. On the whole, the distribution of error does not appear to be uniform. The departure from normality may be due, in part, to the asymmetry of the boundary conditions enforced on the system. Even with the distribution of error as it is, the accuracy of the schemes remains excellent. I expect that both the Gauss-Seidel and Conjugate Gradient methods will be quite effective for solving the coupled implicit difference equations associated with the Time Dependent Schrödinger Equation. In later efforts, other implicit difference schemes such as Douglas-Rachford may be worth exploring.

A Appendix: Generating Non-Uniform Grids

Non-uniform grids are used for the 1D and 2D test problems employed in this report. The purpose of grid non-uniformity is to generate non-symmetric finite difference coefficient matrices. This type of matrix is required to test the Biconjugate Gradient linear system solver. The first grid generator is based upon the finite geometric series.

A.1 Finite Geometric Series Grid Generation

A finite geometric series may be used for generating simple 1D stretching functions useful in grid generation. With this function, we can divide a line segment into N non-equal segments. Consider the line segment between $x = 0$ and $x = 1$. The i^{th} sub-segment is assigned length Δx_i for $i = 1, \ldots, N$. If the overall line segment has length one, then

$$1 = \Delta x_1 + \Delta x_2 + \cdots + \Delta x_N \tag{294}$$

To provide this sequence of sub-segment lengths with characteristics of the geometric series, we assume that lengths of two adjacent sub-segments differ by the ratio r, i.e.,

$$\Delta x_i = r^{i-1}\Delta x, \quad i = 1, \ldots, N \tag{295}$$

thus, the segment length formula may be rewritten as

$$1 = \Delta x_1 + r\Delta x + \cdots + r^{N-1}\Delta x \tag{296}$$

or

$$1 = \Delta x(1 + r + \cdots + r^{N-1}) \tag{297}$$

where Δx is a fixed length for the first sub-segment. The term in parentheses is a finite geometric series with ratio r. With some work, we can develop a simple expression for the series sum. Let

$$S = 1 + r + \cdots + r^{N-1} = 1 + r(1 + r + \cdots + r^{N-2}) = 1 + r\tilde{S} \tag{298}$$

From the above equation, we can also see that

$$S = \tilde{S} + r^{N-1} \tag{299}$$

Equating the expressions for S, we obtain

$$1 + r\tilde{S} = \tilde{S} + r^{N-1} \tag{300}$$

By solving for $\tilde{S}$, we have that

$$\tilde{S} = \frac{1 - r^{N-1}}{1 - r} \tag{301}$$

so,

$$S = 1 + r\tilde{S} = \frac{1 - r^N}{1 - r} \tag{302}$$

By substituting for S, the segment length formula becomes

$$1 = \Delta x \left(\frac{1 - r^N}{1 - r} \right) \tag{303}$$

This expression can be rewritten as a polynomial equation, i.e.,

$$r^N - \frac{r}{\Delta x} + \frac{1 - \Delta x}{\Delta x} = 0 \tag{304}$$

Given values of N and Δx, the above expression can be solved for the ratio r by using numerical methods. When r has been determined, the 1D mesh can be generated by using the equation below.

$$x_i = \begin{cases} x_1 = 0 \\ x_i = \frac{1-r^i}{1-r}, \quad i = 2, \ldots, N \end{cases}$$

A.2 Grid Generation by Parabolic Stretching Function

Stretching functions are frequently used in grid generation. This author has an interest in creating his own. In this case, we can base a stretching function on a parabola of the form:

$$y = y_0 - k(x - x_0)^2 \tag{305}$$

This parabola is concave down for $k > 0$, and it intersects the x axis at $x = 0$ and $x = 2x_0$ where x_0 is the x coordinate of the parabola's vertex. The idea behind this stretching function can be articulated as follows. If points are equally spaced along the parabola's arc, then the x coordinates for these points are clustered near $x = 0$ and $x = 2x_0$. In order to create the stretching function, it is necessary to determine the parabola's arc length as a function of x. Recall that the element of arc length ds is given by

$$ds^2 = dx^2 + dy^2 \quad \Rightarrow \quad ds = \sqrt{dx^2 + dy^2} = \sqrt{1 + \left(\frac{dy}{dx} \right)^2}\, dx \tag{306}$$

From the form of the parabola,

$$\frac{dy}{dx} = -2k(x - x_0) \tag{307}$$

The arc length function $S(\tilde{x})$ may be written as

$$S(\tilde{x}) = \int_0^{\tilde{x}} \sqrt{(1 + (-2k(x - x_0))^2}\, dx \tag{308}$$

Note that $\tilde{x}$ coincides with x; x is simply used a dummy variable of integration. Normally, $\tilde{x}$ is used as the dummy variable, but the equations remain cleaner this way. To perform the integral, we use substitution; let

$$\tan\theta = 2k(x - x_0) \quad \Rightarrow \quad x - x_0 = \frac{\tan\theta}{2k} \tag{309}$$

Also,

$$x = x_0 + \frac{\tan\theta}{2k} \quad \Rightarrow \quad dx = \frac{1}{2k} d(\tan\theta) \quad \Rightarrow \quad dx = \frac{\sec^2\theta}{2k} d\theta \tag{310}$$

By subsequent change of the independent variable to $\tilde{\theta}$,

$$S(\tilde{\theta}) = \int_0^{\tilde{\theta}} \sqrt{1 + \tan^2\theta} \cdot \frac{\sec^2\theta}{2k}\, d\theta = \int_0^{\tilde{\theta}} \frac{\sec^3\theta}{2k}\, d\theta \tag{311}$$

In order to evaluate this integral, an antiderivative is needed for $\sec^3\theta$. Observe that

$$\int \sec^3\theta = \int \sec\theta\ \sec^2\theta\ d\theta \tag{312}$$

Integration by parts is useful at this juncture with the substitutions

$$u = \sec\theta;\ \ du = \sec\theta\tan\theta\ d\theta \tag{313}$$

$$dv = \sec^2\theta\, d\theta;\ \ v = \tan\theta \tag{314}$$

Then we have that

$$\int \sec^3\theta\ d\theta = \sec\theta\tan\theta - \int \sec\theta\tan^2\theta\ d\theta \tag{315}$$

A basic trigonometric identity states that

$$\tan^2\theta = \sec^2\theta - 1 \tag{316}$$

so

$$\int \sec^3\theta\ d\theta = \sec\theta\tan\theta - \int \sec^3\theta\ d\theta + \int \sec\theta\ d\theta \tag{317}$$

By rearranging this expression,

$$2\int \sec^3\theta\ d\theta = \sec\theta\tan\theta + \ln|\sec\theta + \tan\theta| \tag{318}$$

Hence, the definite integral becomes

$$S(\tilde{\theta}) = \frac{1}{2k}\left[\sec\theta\tan\theta + \ln|\sec\theta + \tan\theta|\right|_0^{\tilde{\theta}} \tag{319}$$

Now it is desirable to return this integral to the argument of x. To do so, we note recall that $\tan\theta = 2k(x - x_0)$; then in combination with the identity $\sec^2\theta = 1 + \tan^2\theta$, we have that $\sec^2\theta = 1 + 4k^2(x - x_0)^2$, and we can conclude that $\sec\theta = \sqrt{1 + 4k^2(x - x_0)^2}$. If these assertions are substituted into $S(\tilde{\theta})$, we obtain

$$S(\tilde{x}) = \frac{1}{4k}\left[2k(x-x_0)\sqrt{1+4k^2(x-x_0)^2} + \ln\left|\sqrt{1+4k^2(x-x_0)^2} + 2k(x-x_0)\right|\right|_0^{\tilde{x}} \quad (320)$$

We can may select a couple of constants at this point. Let

$$x_0 = \frac{1}{2}; \quad y_0 = \frac{k}{4} \quad (321)$$

With these constants, the parabolic equation becomes

$$y(x) = -kx(x-1) \quad (322)$$

where the domain of x is now $[0,1]$ in keeping with the domain used in the test problems. The full arc length $S(1)$ can be subdivided into segments. The x coordinates at the end-points can be determined by numerically solving $S(x_i)$ for each incremental arc length. The inverse function of $S(\tilde{x})$ applied to the incremental arc length provides the grid value.

References

[1] Weinberger, H.F., *A First Course in Partial Differential Equation*, John Wiley & Sons, New York, New York, 1965.

[2] Mitchell, A.R. and Griffiths, D.F., *The Finite Difference Method in Partial Differential Equations*, John Wiley & Sons, New York, New York, 1980.

[3] Nance, D.V., *Finite Volume Algorithms for Heat Conduction*, Technical Report AFRL-RW-EG-TR-2010-049, Munitions Directorate, Air Force Research Laboratory, May 2010.

[4] Soriano, A., Navarro, E.A., Porti, J.A. and Such, V., "Analysis of the finite difference time domain technique to solve the Schrödinger equation for quantum devicies", *Journal of Applied Physics*, Vol. 95, No. 12, 2004.

[5] Zachmanoglou, E.C. and Thoe, D.W., *Introduction to Partial Differential Equations with Applications*, Dover Publications, New York, New York, 1986.

[6] John, F., *Partial Differential Equations*, 4^{th} Ed., Applied Mathematical Sciences, Vol. 1, Springer-Verlag, New York, New York, 1982.

[7] Gustafson, K.E., *Introduction to Partial Differential Equations and Hilbert Space Methods*, 2^{nd} Ed., John Wiley & Sons, New York, New York, 1987.

[8] Golub, G.H. and Van Loan, C.F., *Matrix Computations*, 2^{nd} Ed., Johns Hopkins University Press, Baltimore, Maryland, 1989.

[9] Dahlquist, G. and Björck, *Numerical Methods*, Prentice-Hall, Englewood Cliffs, New Jersey, 1974.

[10] Burden, R.L., Faires, J.D. and Reynolds, A.C., *Numerical Analysis*, 2^{nd} Ed., Prindle, Weber & Schmidt, Boston, Massachusetts, 1981.

[11] Smith, G.D., *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, 3^{rd} Ed., Oxford Applied Mathematics and Computing Science Series, Oxford University Press, New York, New York, 1985.

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

[13] Axelsson, O., *Iterative Solution Methods*, Cambridge University Press, New York, New York, 1994.

[14] Marsden, J.E., *Elementary Classical Analysis*, W.H. Freeman and Company, San Francisco, California, 1974.

[15] Dennis, J.E., Jr. and Schnabel, R.B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Classics in Applied Mathematics, Vol. 6, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1996.

[16] Gutknecht, M.H., "A brief introduction to Krylov space methods for solving linear systems", Seminar for Applied Mathematics, ETH Zurich, Zurich, Switzerland.

[17] Fan, S., "An Introduction to Krylov Subspace Methods - A Less Mathematical Way To Understand", Zhejiang University, arXiv: 1811.09025v [math.OC], November 2018.

[18] Lanczos, C., "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators", *J. Res. Nat. Bur. Standards*, Vol. 45, pp. 255-282, 1950.

[19] Freund, R., *Conjugate Gradient Type Methods for Linear Systems with Complex Symmetric Coefficient Matrices*, RIACS Technical Report 89.54, Research Institute for Advanced Computer Science, NASA Ames Research Center, 1989.

[20] Hogg, R.V. and Craig, A.T., *Introduction to Mathematical Statistics*, 4^{th} Ed., Macmillan Publishing Company, New York, New York, 1978.