Finite Volume Solution of the Heat Conduction Equation -An Application in 3D

Technical Report · August 2020

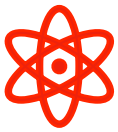
DOI: 10.13140/RG.2.2.29422.84807

1 author:

Douglas Vinson Nance
Air Force Research Laboratory
27 PUBLICATIONS **74** CITATIONS

SEE PROFILE

Computational Physics Notes



Finite Volume Solution of the Heat Conduction Equation An Application in 3D

Douglas V. Nance

August 2020

Finite Volume Solution of the Heat Conduction Equation - An Application in 3D

Douglas V. Nance
Independent Research Scientist

August 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 work is the third in a series of reports concerned with the application of the Finite Volume Method for numerically solving the Heat Conduction Equation, or simply put, the Heat Equation. The first report concentrates on the development of this method for a two-dimensional problem cast in curvilinear coordinates using both explicit and simple implicit time propagation schemes. A certain level of personal dissatisfaction with the implicit scheme led, nearly ten years later, to the creation of more computer programming and a second report on the use of iterative schemes for solving a more heavily populated implicit coefficient matrix. The methodology employed in that report considers both the Gauss-Seidel and Conjugate Gradient Methods. In this report, some of these ideas are combined to solve a problem cast in three dimensions and in curvilinear coordinates. In addition to solving a basic heating problem in Cartesian coordinates, a heating problem is solved on a multi-block spherical geometry.

Contents

1 Introduction

This report, the third in a series, discusses the development of the Finite Volume Method (FVM) for solving the Heat Conduction Equation (or the Heat Equation (HE)), in three dimensions. The first report [1] regards a problem cast in 2D curvilinear coordinates while the second report [2] concentrates on implicit algorithms. To simplify the work, the second report uses the Finite Difference Method (FDM) for its framework. The common ground existing between FVM and FDM makes logical the application of certain classical implicit schemes for these two methods. In the way of review, one may recall that the Heat Equation is a parabolic partial differential equation (PDE) ubiquitous in applied mathematics and physics. Also, it is frequently used for didactic purposes.[3] A reason underlying its utility is that this PDE incorporates time variation with the diffusive nature of the Laplace operator. The FVM exhibits great utility for solving this type of equation since the flow of heat into a volume can be formulated in terms of the heat flux traveling across the boundary of a volume. For those individuals who have read either the first or the second report, the background discussion below will seem familiar.

1.1 Background

In 2010, I completed a technical report on numerical solutions of the heat equation via the finite volume method (FVM).[1] This technique is a departure from the more standard FDM process. The FVM type of discretization is usually applied for a different set of applications, namely fluid dynamics. That is, it is well suited for systems of conservation laws such as the Euler or Navier-Stokes equations.[4] As it happens, this first 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, many months. The level of interest exhibited by the research community in this topic is appreciated and serves as motivation for both the second report and the present manuscript. In truth, my continued 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 caused by high speed flight and to determine the attendant change on the flow field. The CFD solver is discretized via the finite volume technique, so it makes sense to apply the same discretization procedure to the heat equation. It makes for easier interface coding since the grids can be generated so that the volumes in the flow field seamlessly interface with those in the solid structure. All in all, this effort is successful, and I have tested both explicit and implicit discretization methods for the Heat Equation. As is documented in the first report, for the implicit discretization, I chose the Crank-Nicolson method cast in two 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 two directions as a tridiagonal matrix solver.[5] Unfortunately, the accuracy of the ADI numerical solution suffered in comparison with the explicit time integration scheme. My unhappiness with this result led to the research documented in the second report. In the second report, I discarded the ADI factorization and lifted the restriction on the number of diagonals existing in the implicit matrix scheme.[6] To

solve these more complicated matrices, I applied both Gauss-Seidel and Conjugate Gradient algorithms.[7] In my view, the results shown are very impressive. In fact, this happy outcome whetted my appetite to see how well some of these algorithms perform in solving a more complicated 3D problem. Not all together incidentally, further work has led to the writing of this report.

After a hiatus of ten years, it is only during the past year that I have had time to return to work on numerical methods for the Heat Equation. Solutions of this equation exhibit interesting behavior, and in my opinion, heat conduction problems are fun to solve. My interest also extends to the generalized coordinate transformations needed in solving heat conduction problems on complex geometries. When I first encountered these techniques as a young graduate student in 1991, I was stunned by their computational power, and I still am.[8] These techniques truly receive a Gold Star for fineness in applied mathematics. 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 my thoughts on the application of these methods to a problem cast in three dimensions. I also discuss an ancillary item or two.

1.2 The Heat Equation

The heat equation (HE) is classified as a parabolic PDE.[3] The model HE used in this report is written as

$$\rho\, C_P \frac{\partial u}{\partial t} = \nabla \cdot (K \nabla u,) \;\; u = u(\vec{x}, t), \;\; t > 0, \;\; \vec{x} \in D \tag{1}$$

while the Gradient operator in three Cartesian dimensions is given as

$$\nabla = \hat{i}\frac{\partial}{\partial x} + \hat{j}\frac{\partial}{\partial y} + \hat{k}\frac{\partial}{\partial z}$$

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. This problem also incorporates physical properties such density ρ, heat capacity C_P and thermal conductivity K as reflected in (1). 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).[9] 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 Heat Equation that incorporate properties like thermal diffusivity, but these forms are still similar to Equation (1). It is interesting to note that the structure of the Heat Equation 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.[10] Therefore, (1) is not time reversible, so it 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.[10]

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.[11] For example, a set of test problems for the Heat Equation are selected and analytically solved in Section 3 of my second report.[2] Such exact solutions are quite valuable in validating a wide array of numerical procedures for this type of problem. For more advanced problems involved complicated structural components or systems involving many components, analytical methods can become intractable. For this reason, FDM and FVM computer codes take center stage.

The heat conduction problems of interest in this report have non-trivial boundary conditions. For this type of problem, I specify a uniform temperature (or internal energy) distribution as the initial condition. The variation or unsteadiness in the field is provided by non-zero boundary conditions. The test cases used here may be characterized as immersion problems. In this type of scenario, a body, originally at a uniform temperature, is immersed in a large reservoir possessing a different uniform temperature. This scenario is really "cartoon-like" because the reservior does not respond to heat transfer. Still, the thrust of this work is to solve problems of this sort by using the FVM.

The Finite Volume Method is a cousin of the Finite Difference Method. The relationship between the two is rather close, but there is a stark difference revealed especially for problems cast in two or three space dimensions. In a simple determination, the FVM discretizes the volume of the problem. That is to say, for a problem set in 3D, the FDM stencil uses numerical stencils to grid the 3D volume in question. On the other hand, the FVM pulls a calculus trick. Through the use of the divergence theorem, the FVM functions by discretizing the boundary of the volume.[4, 12] At the boundary of the volume, the differential space operator is reduced by one order. That, in itself, can be viewed as an advantage, but as we shall see, FVM also allows quantities such a mass, energy and momentum to be accurately tracked as they flow across the boundary. A brief comparison between one-dimensional FVM and FDM is presented below.

1.3 FVM vs. FDM

The best way to illustrate both the common ground and diverging characteristics of FVM and FDM is to discretize the same problem with both methods. For this purpose, we select the 1D Heat Equation, i.e.,

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

In the FDM context, (2) is easily discretized by using Newton divided differences.[13] That is,

$$\frac{\partial u}{\partial t} \approx \frac{u_i^{n+1} - u_i^n}{\Delta t} \tag{3}$$

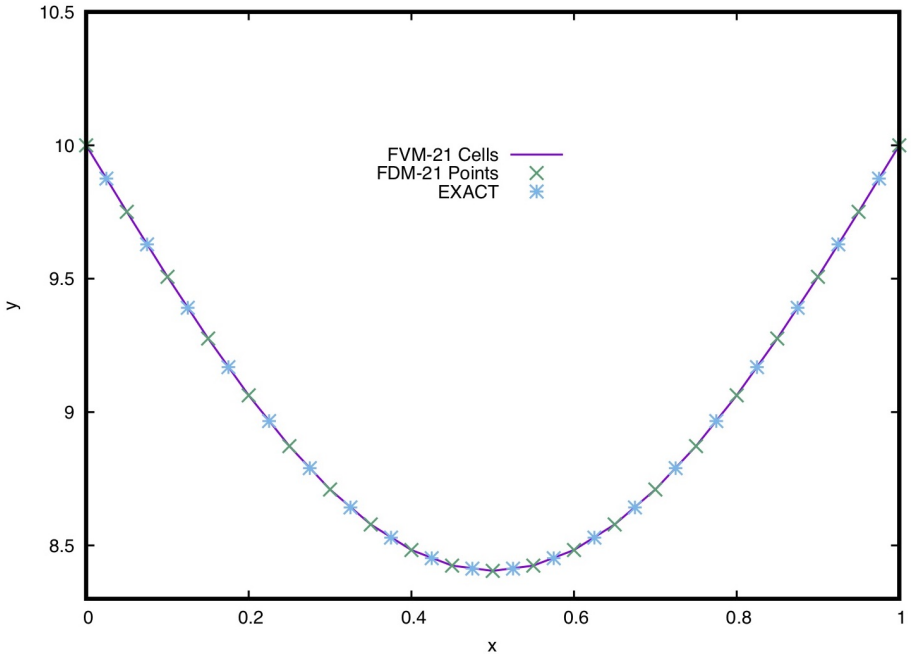


Figure 1: FDM and FVM time 0.2 numerical solutions for the symmetric heating problem compared with the exact solution

and

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \tag{4}$$

where the discretization mapping is as follows.

$$x_i = i\Delta x,\ i = 1, \ldots, \text{imax}; \quad t = n\Delta t,\ n = 1, \ldots, \text{nmax} \tag{5}$$

with i being the space index and n the time level. It follows that the difference equation representing (2) is

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \tag{6}$$

where $i = 1, \ldots, \text{imax}$, and $n = 1, \ldots, \text{nmax}$. For comparison, we develop the finite volume representation of the 1D Heat Equation. Recall (2) and integrate this equation in space.

$$\int_{x_i}^{x_{i+1}} \frac{\partial u}{\partial t} dx = \int_{x_i}^{x_{i+1}} \frac{\partial^2 u}{\partial x^2} dx \tag{7}$$

On the left, we can interchange the order of integration and differentiation, and on the right we have anti-differentiation, so

$$\frac{\partial}{\partial t} \int_{x_i}^{x_{i+1}} u dx = \frac{\partial u}{\partial x}\Big|_{x_i}^{x_{i+1}} \tag{8}$$

The integral on the left effectively creates a 1D "volume" of u while the right expression evaluates u say, its average, at the endpoints of what is the finite volume cell. As it happens, we regard the integral on left side as the *average* value of u for the cell. We denote this average as $\bar{u}$. Thus,

$$\Delta x \, \frac{\partial \bar{u}_i}{\partial t} = \frac{\partial \bar{u}_{i+1}}{\partial x} - \frac{\partial \bar{u}_i}{\partial x} \tag{9}$$

If we apply backward differences to the space terms, we arrive at the form

$$\Delta x \, \frac{\partial \bar{u}_i}{\partial t} = \frac{\bar{u}_{i+1} - \bar{u}_i}{\Delta x} - \frac{\bar{u}_i - \bar{u}_{i-1}}{\Delta x} \tag{10}$$

By evaluating the above expression at time level n and applying a forward difference to the time derivative, we obtain

$$\Delta x \, \frac{\bar{u}_i^{n+1} - u_i^n}{\Delta t} = \frac{\bar{u}_{i+1}^n - \bar{u^n}_i}{\Delta x} - \frac{\bar{u}_i^n - \bar{u}_{i-1}^n}{\Delta x} \tag{11}$$

Simplification provides the finite volume difference equation, i.e.,

$$\frac{\bar{u}_i^{n+1} - \bar{u}_i^n}{\Delta t} = \frac{\bar{u}_{i+1}^n - 2\bar{u}_i^n + \bar{u}_{i-1}^n}{\Delta x^2} \tag{12}$$

In terms of appearance, (12) is the same as (6) with the exception of the averaging bars. So it appears that the 1D FDM and FVM are mostly equivalent. This declaration does require some caution as we shall see.

Consider a basic, symmetric heating problem in 1D, i.e.,

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \quad u = u(x,t), \quad 0 < x < 1, \quad t > 0 \tag{13}$$

$$u(0,t) = 10, \quad u(1,t) = 10 \tag{14}$$

$$u(x,0) = 1 \tag{15}$$

In either of the FDM and FVM formulations, an Euler explicit time integration scheme can be applied, i.e., for $T_1 = 1$ and $T_2 = 10$,

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{\Delta x}\left(\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}\right) \tag{16}$$

For the finite difference model, we create a uniform mesh consisting of 21 points, so $x(i) = (i-1)\Delta x,\ \Delta x = 1/20$. The boundary conditions are applied at $i = 1, 21$. Initial conditions are applied for $i = 2, \ldots, 20$. On the other hand, the finite volume model is cast on the same grid with the exception that 21 grid points are used to create 21 1D volumes. With the exception of volume 21, each "volume" is bounded by two grid points. In this model, the 21^{st} cell exists external to the grid at imax. It follows that the boundary values ($u = 10$) are located at $i = 1$ and $i = 21$. Accordingly, the solvable field exists at indices $i = 2, \ldots, 20$; the spatial step size $\Delta x = 0.4$ remains the same in each case. The major difference is that

the FVM requires an additional cell exterior to the right boundary of the mesh. The same calculation can be done without the external cell at imax. Rather the right boundary cell can be placed at i = 20, but doing so forces Δx to be based upon the value of imax-2. The results of both of these calculations are compared with the exact solution in Figure 1. For those interested, a similar exact solution is described in my second report.[2] Still, for completeness, the exact solution for the problem above is written as

$$u(x,t) = 2(T_1 - T_2)\sum_{n=1}^{100}\left(\frac{1-\cos(n\pi)}{n\pi}\right)\sin(n\pi x)\exp(-n^2\pi^2 t) + T_2 \tag{17}$$

I chose 100 Fourier modes for this exact solution. As one can see, both numerical schemes perform quite well, but shifting over from the FDM to the FVM does require care. Yet, both schemes are equivalent on the uniform mesh.

In the sections of the report that follow, our concern is in implementing the FVM for 3D problems. First, we examine Cartesian FVM then FVM on arbitrary, yet nearly orthogonal 3D grids. This approach permits a gradual exposure to the issue of transformation. Also, it helps to keep the exposition as simple as possible for pedagogical purposes. At least, that is the intent; the author's success or failure in this endeavor must be judged by the reader. This report can be mapped out as follows. Next, the FVM is developed for 3D problems starting with the Cartesian, uniform grid version. After that introduction, the method is further developed for arbitrary, yet closely orthogonal grids. The close othogonality qualification is important because for the heat conduction problem, partial derivatives must be represented on the grid. For non-orthogonal grids, the representation of space derivatives is decidedly more complicated through the use of covariant differentiation. That topic is beyond the scope of this report.

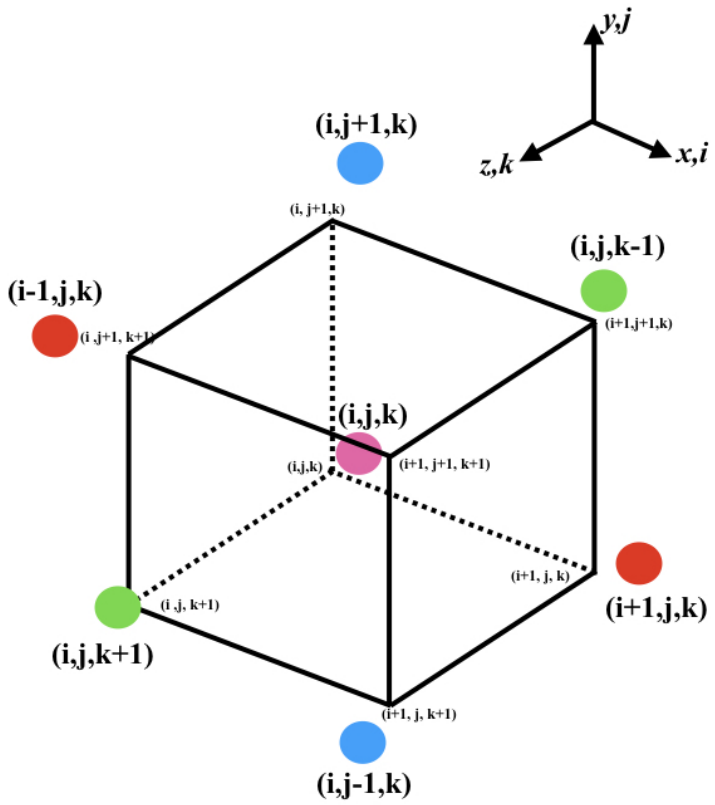


Figure 2: Typical Hexahedral Cartesian Finite Volume Cell

2 Finite Volume Methods

The Finite Volume Methods are built on a foundation created by the Divergence Theorem of Integral Calculus.[12] In vector notation, the Heat Equation may be written as

$$\frac{\partial u}{\partial t} = \nabla^2 u = \nabla \cdot \nabla u \tag{18}$$

For this analysis, space can be divided up into a set of cubic volumes. The side lengths for each volume are all of all equivalent $\Delta x = \Delta y = \Delta z$. As one example, a typical hexahedral Cartesian finite volume cell is shown in Figure 2; the so-called northeast numbering convention is employed. That is the say, the cell indices (i, j, k) are the same as those of the minimal grid point. The center for this cell is represented by the lavendar disc while the adjacent cell centers in the x, i direction are shown as red discs. In the y, j and z, k directions, the cell centers are indicated by blue and green discs, respectively. The grid point indices are given in the smaller black type. Later, the cell structure is important for geometry calculations. The first step in developing the FVM equations is to integrate (18) over the cell volume.

$$\int_{V_{i,j,k}} \frac{\partial u}{\partial t}\, dV = \int_{V_{i,j,k}} \nabla \cdot \nabla u\, dV \tag{19}$$

Since the cell volumes are fixed, they are independent of time, and the partial time derivative can be moved out of the integral on the left side of the equation. Since spatial variation has been integrated out of u on the left side, the time derivative transforms from a partial derivative to a ordinary derivative. On the right side of the equation, we employ the divergence theorem and convert the volume integral to a closed integral over the cell boundary. Hence,

$$\frac{d}{dt}\int_{V_{i,j,k}} u\, dV = \oint_{\partial V_{i,j,k}} \nabla u \cdot d\vec{a} \tag{20}$$

where $d\vec{a}$ is the vector element of surface area. In this equation, $\partial V_{i,j,k}$ denotes the boundary surface for the cell volume. The integral on the left side of the equation is the "average" of u computed over the cell volume. The argument is formal, but we represent the average of u as $\bar{u}$, the volume integral becomes $\bar{u}$ multiplied by the cell volume, i.e.,

$$V_{i,j,k}\frac{d\bar{u}_{i,j,k}}{dt} = \oint_{\partial V_{i,j,k}} \nabla u \cdot d\vec{a} \tag{21}$$

2.1 Cartesian FVM Development

For my Cartesian FVM, hexahedral finite volume cells are aligned with the Cartesian axes. With the volumes structured in this way, the surface integral on the right can be written as follows.

$$V_{i,j,k}\frac{d\bar{u}_{i,j,k}}{dt} = \sum_{l=1}^{6} \nabla u^{l}_{i,j,k} \cdot \vec{a}^{\,l}_{i,j,k} \tag{22}$$

Formally, we now designate all values of u to be cell-averaged values, and for brevity, we discontinue the "bar" notation. The cell side index $l = 1, \ldots, 6$ really represents coordinate directions for $\pm x$, $\pm y$ and $\pm z$ since the area vectors point away from the cell center; l is not an exponent. Our finite volume cells are cube shaped, so all of the cell sides have the same area. With Figure 2 in mind, the sides of the cell may be labeled $i+1$, i, $j+1$, j, $k+1$ and k. Accordingly, let $\hat{n}^{\,l}$ be the unit normal vector for cell side l. Then (22) can be expanded as follows.

$$\begin{aligned} V_{i,j,k}\frac{du_{i,j,k}}{dt} = {} & \nabla u^{i+1}_{i,j,k} \cdot \hat{n}^{\,i+1}_{i,j,k} a + \nabla u^{i}_{i,j,k} \cdot \hat{n}^{\,i}_{i,j,k} a \\ & + \nabla u^{j+1}_{i,j,k} \cdot \hat{n}^{\,j+1}_{i,j,k} a + \nabla u^{j}_{i,j,k} \cdot \hat{n}^{\,j}_{i,j,k} a \\ & + \nabla u^{k+1}_{i,j,k} \cdot \hat{n}^{\,k+1}_{i,j,k} a + \nabla u^{k}_{i,j,k} \cdot \hat{n}^{\,k}_{i,j,k} a \end{aligned} \tag{23}$$

By evaluating the unit normal vectors, we have that

$$V_{i,j,k}\frac{du_{i,j,k}}{dt} = a\Big[\nabla u^{i+1}_{i,j,k}\cdot(+\hat{i}) + \nabla u^{i}_{i,j,k}\cdot(-\hat{i}) + \nabla u^{j+1}_{i,j,k}\cdot(+\hat{j}) + \nabla u^{j}_{i,j,k}\cdot(-\hat{j}) + \nabla u^{k+1}_{i,j,k}\cdot(+\hat{k}) + \nabla u^{k}_{i,j,k}\cdot(-\hat{k})\Big] \tag{24}$$

Moreover,

$$V_{i,j,k}\frac{du_{i,j,k}}{dt} = a\Big[(\nabla u^{i+1}_{i,j,k} - \nabla u^{i}_{i,j,k})\cdot\hat{i} + (\nabla u^{j+1}_{i,j,k} - \nabla u^{j}_{i,j,k})\cdot\hat{j} + (\nabla u^{k+1}_{i,j,k} - \nabla u^{k}_{i,j,k})\cdot\hat{k}\Big] \tag{25}$$

Recall that the gradient of u is given by

$$\nabla u = \hat{i}\frac{\partial u}{\partial x} + \hat{j}\frac{\partial u}{\partial y} + \hat{k}\frac{\partial u}{\partial z} \tag{26}$$

By using (26) in (25), we obtain

$$V_{i,j,k}\frac{du_{i,j,k}}{dt} = a\left[\left(\frac{\partial u^{i+1}_{i,j,k}}{\partial x} - \frac{\partial u^{i}_{i,j,k}}{\partial x}\right) + \left(\frac{\partial u^{j+1}_{i,j,k}}{\partial y} - \frac{\partial u^{j}_{i,j,k}}{\partial y}\right) + \left(\frac{\partial u^{k+1}_{i,j,k}}{\partial z} - \frac{\partial u^{k}_{i,j,k}}{\partial z}\right)\right] \tag{27}$$

The right side of (27) can now be discretized by using Newton divided differences. Note that

$$\begin{aligned} \frac{\partial u^{i+1}_{i,j,k}}{\partial x} &= \frac{u^{i+1}_{i,j,k} - u^{i}_{i,j,k}}{\Delta x}; \quad \frac{\partial u^{i}_{i,j,k}}{\partial x} = \frac{u^{i}_{i,j,k} - u^{i-1}_{i,j,k}}{\Delta x} \\ \frac{\partial u^{j+1}_{i,j,k}}{\partial y} &= \frac{u^{j+1}_{i,j,k} - u^{j}_{i,j,k}}{\Delta y}; \quad \frac{\partial u^{j}_{i,j,k}}{\partial y} = \frac{u^{j}_{i,j,k} - u^{j-1}_{i,j,k}}{\Delta y} \\ \frac{\partial u^{k+1}_{i,j,k}}{\partial z} &= \frac{u^{k+1}_{i,j,k} - u^{k}_{i,j,k}}{\Delta z}; \quad \frac{\partial u^{k}_{i,j,k}}{\partial z} = \frac{u^{k}_{i,j,k} - u^{k-1}_{i,j,k}}{\Delta z} \end{aligned} \tag{28}$$

By substituting (28) into (27), the semi-discrete from of

$$V_{i,j,k}\frac{du_{i,j,k}}{dt} = a\left[\left(\frac{u^{i+1}_{i,j,k}-u^{i}_{i,j,k}}{\Delta x}\right)-\left(\frac{u^{i}_{i,j,k}-u^{i-1}_{i,j,k}}{\Delta x}\right)\right.$$
$$+\left(\frac{u^{j+1}_{i,j,k}-u^{j}_{i,j,k}}{\Delta y}\right)-\left(\frac{u^{j}_{i,j,k}-u^{j-1}_{i,j,k}}{\Delta y}\right) \tag{29}$$
$$\left.+\left(\frac{u^{k+1}_{i,j,k}-u^{k}_{i,j,k}}{\Delta z}\right)-\left(\frac{u^{k}_{i,j,k}-u^{k-1}_{i,j,k}}{\Delta z}\right)\right]$$

is obtained. Recall that the proposed Cartesian grid contains cube shaped cells, so the mesh increments are equal, i.e.,

$$\Delta = \Delta x = \Delta y = \Delta z \tag{30}$$

Being mindful of this assertion and with the use of some algebra, the semi-discrete form becomes

$$\frac{du_{i,j,k}}{dt} = \frac{a}{V_{i,j,k}\,\Delta}\left[u_{i+1,j,k}+u_{i-1,j,k}+u_{i,j+1,k}+u_{i,j-1,k}+u_{i,j,k+1}+u_{i,j,k-1}-6u_{i,j,k}\right] \tag{31}$$

Let the right side of this result at time level n be written as

$$R(u^n_{i,j,k}) = \frac{a}{V_{i,j,k}\,\Delta}\left[u^n_{i+1,j,k}+u^n_{i-1,j,k}+u^n_{i,j+1,k}+u^n_{i,j-1,k}+u^n_{i,j,k+1}+u^n_{i,j,k-1}-6u^n_{i,j,k}\right] \tag{32}$$

Note that the superscript on u now indicates the time level; before, it marked a particular side of the finite volume cell. It is still not an exponent. Such mathematical typography is clearly an abuse of notation, but that is the nature of busy equations. Explanation is my remedy of choice.

The symbol $R(u)$, as in (32) is a convenient form of notation for elucidating a time propagation scheme. For simplicity, consider the Modified Euler method applied to (31).[13]

$$u^*_{i,j,k} = u^n_{i,j,k} + R(u^n_{i,j,k})$$
$$u^{n+1}_{i,j,k} = \frac{1}{2}\left[u^*_{i,j,k}+u^n_{i,j,k}+R(u^*_{i,j,k})\right] \tag{33}$$

The Modified Euler scheme is not known for its stability, but it is an excellent choice for debugging and initial numerical experiments.[13] Equations (31) through (33) are effective in solving 3D Cartesian heating problems.

2.2 FVM For Near-Orthogonal Curvilinear Grids

The content presented below forms the crux of this report allowing a wider class of problems to be solved than is afforded by Cartesian FVM. Also, a slightly more general form of the Heat Equation is addressed, that is,

$$\rho\, C_P \frac{\partial u}{\partial t} = \nabla \cdot (K\, \nabla u), \quad u = u(\vec{x}, t) \tag{34}$$

where ρ is the density of the material in question, and C_P is its heat capacity. In this case, u is temperature, and K is the material's thermal conductivity. These properties may also be space functions. As in the Cartesian case, we integrate over the volume of a curvilinear grid cell. Again, we refer to Figure 2 except the index i corresponds to the curvilinear coordinate ξ; index j corresponds to coordinate η while index k is associated to coordinate ζ. Integrate as follows.

$$\int_{V_{i,j,k}} \rho\, C_P \frac{\partial u}{\partial t}\, dV = \int_{V_{i,j,k}} \nabla \cdot (K\, \nabla u)\, dV, \quad u = u(\vec{x}, t) \tag{35}$$

On the left side of (35),the integral removes the spatial variation from u, ρ and C_P within cell (i, j, k). In so doing, the partial time derivative is transformed to an ordinary derivative. With bar notation used for certain averages, we have that

$$\bar{\rho}\, \bar{C_P} V_{i,j,k} \frac{d\bar{u}}{dt} = \int_{V_{i,j,k}} \nabla \cdot (K\, \nabla u)\, dV, \quad u = u(\vec{x}, t) \tag{36}$$

Now on the right side of (36), apply the divergence theorem to obtain

$$\bar{\rho}\, \bar{C_P} V_{i,j,k} \frac{d\bar{u}}{dt} = \oint_{\partial V_{i,j,k}} K\, \nabla u \cdot\, d\vec{a} \tag{37}$$

The integral on the right side is now a surface integral over $\partial V_{i,j,k}$, the boundary of cell (i, j, k).

In deference to the FDM, FVM operates on cell average values of $u_{i,j,k}$. In theory, these quantities are defined at the cell centroid. That assertion is not clear from examining (37), and it is a limitation of the notation. Given that we know that we are discussing $\bar{u}$ at the cell centers, we now drop the bar notation. It just gets in the way. It follows that the right side of (37) also operates on u at the cell centers. Complication does ensue because the integral is defined over a closed surface, not at points in the cell volume. As a result, ∇u must be mapped to the cell sides, and again, we must think of $u_{i,j,k}$ in terms of an average value for cell (i,j,k). It is in this mapping that the power of the curvilinear transformation can become clear, but first, we expand the right side of (37) based upon the hexahedral cell structure shown in Figure 2. The expanded equation can be written as follows.

$$\begin{aligned}\rho\, C_P V_{i,j,k}\frac{du}{dt} = {} & K\nabla u\cdot\ \vec{\Delta a}\Big|_{i,j,k}^{i+1} + K\nabla u\cdot\ \vec{\Delta a}\Big|_{i,j,k}^{i^-} \\ & + K\nabla u\cdot\ \vec{\Delta a}\Big|_{i,j,k}^{j+1} + K\nabla u\cdot\ \vec{\Delta a}\Big|_{i,j,k}^{j^-} \\ & + K\nabla u\cdot\ \vec{\Delta a}\Big|_{i,j,k}^{k+1} + K\nabla u\cdot\ \vec{\Delta a}\Big|_{i,j,k}^{k^-}\end{aligned} \tag{38}$$

The upper index on a vertical evaluation bar | simply identifies the cell side or interface in question. The minus sign on sides i, j, and k indicates that the associated area vector is to be reversed in sense to follow tangent to the coordinate index. We also make an approximation at this point. The integral value for the cell side is represented by a single value theoretically located at the center of the cell side. To this end, the side area vector is denoted $\vec{\Delta a}$ while ∇u is the gradient vector defined at the center point of the cell side. The cell side area vector can be written as follows.

$$\vec{\Delta a} = \hat{n}\Delta a \tag{39}$$

where $\hat{n}$ is the outward oriented unit normal vector for the cell side where Δa is the scalar area of the cell side. To continue with the development of this method, my notation again becomes abusive, this time regarding i, j and k. When i is used as an index and not as a vector, it indexes cells (and grid points for flux calculations) in the curvilinear ξ coordinate direction. On the other hand, $\hat{i}$ is a unit vector that points in the direction of the Cartesian coordinate x. Similarly, when j and k are used as indices, not as vectors, they index cells (and grid points) in the curvilinear η and ζ directions, respectively. In the same manner, $\hat{j}$ and $\hat{k}$ are unit vectors that point in the Cartesian y and z directions, respectively.

In the true sense of FVM, we can now speak of *flux*, the quantity of energy crossing the cell sides per unit time, per unit area. We can rewrite (38) in terms of flux formed along the three curvilinear coordinate directions, i.e.,

$$\rho\, C_P V_{i,j,k}\frac{du}{dt} = F_i + F_j + F_k \tag{40}$$

Poor notation strikes again in the form of F_i, F_j and F_k since I did not annotate these fluxes as being associated with cell (i,j,k). My apologies, I will attempt to supplement and clarify the notation in the specifics below. Using the information given above, let us evaluate F_i, the flux in the ξ or i direction for cell (i,j,k).

$$F_i = K\nabla u\cdot\hat{n}\Delta a\big|_{j,k}^{i+1} + K\nabla u\cdot\hat{n}\Delta a\big|_{j,k}^{i^-} \tag{41}$$

Recall that the superscript - on the upper index in the second term in (41) indicates that the associated unit normal vector should be reversed in sense to render a convenient form for computing the i flux difference across cell (i,j,k). The lower indices simply complete the identification of the cell. Hence,

$$F_i = K\nabla u\cdot\hat{n}\Delta a\big|_{j,k}^{i+1} - K\nabla u\cdot\hat{n}\Delta a\big|_{j,k}^{i} \tag{42}$$

Taking the above step makes it easier to compute the i flux since the same formula can be used in a loop to calculate the area vector. The j and k fluxes can be easily computed by using similar formulas, e.g.,

$$F_j = K\nabla u \cdot \hat{n}\Delta a|_{i,k}^{j+1} - K\nabla u \cdot \hat{n}\Delta a|_{i,k}^{j} \tag{43}$$

and

$$F_k = K\nabla u \cdot \hat{n}\Delta a|_{i,j}^{k+1} - K\nabla u \cdot \hat{n}\Delta a|_{i,j}^{k} \tag{44}$$

Note that the indices on an evaluation sign apply to all terms to the left of that sign up to and including K. By recalling (26) and noting that the unit normal vector is given as

$$\hat{n} = \hat{i}\, n_x + \hat{j}\, n_y + \hat{k}\, n_z, \quad \sqrt{n_x^2 + n_y^2 + n_z^2} = 1 \tag{45}$$

Therefore,

$$\nabla u \cdot \hat{n}\Delta a = \left(n_x \frac{\partial u}{\partial x} + n_y \frac{\partial u}{\partial y} + n_z \frac{\partial u}{\partial z} \right) \Delta a \tag{46}$$

The unit normal may be calculated from the curvilinear grid without difficulty, but the gradient term requires transformation. Fletcher [8] is a good reference for this technique, but it is important to realize that the focus of this reference is the FDM, so the process shown here is a bit different. The transformation between the Cartesian and curvilinear systems can be initially written as

$$u(x, y, z) \rightarrow u(\xi, \eta, \zeta) \tag{47}$$

By applying the chain rule, we have that

$$\begin{aligned}
\frac{\partial u}{\partial x} &= \frac{\partial u}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial u}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial u}{\partial \zeta}\frac{\partial \zeta}{\partial x} \\
\frac{\partial u}{\partial y} &= \frac{\partial u}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial u}{\partial \eta}\frac{\partial \eta}{\partial y} + \frac{\partial u}{\partial \zeta}\frac{\partial \zeta}{\partial y} \\
\frac{\partial u}{\partial z} &= \frac{\partial u}{\partial \xi}\frac{\partial \xi}{\partial z} + \frac{\partial u}{\partial \eta}\frac{\partial \eta}{\partial z} + \frac{\partial u}{\partial \zeta}\frac{\partial \zeta}{\partial z}
\end{aligned} \tag{48}$$

In matrix form, we have

$$\begin{bmatrix} \frac{\partial u}{\partial x} \\ \\ \frac{\partial u}{\partial y} \\ \\ \frac{\partial u}{\partial z} \end{bmatrix} = \begin{bmatrix} \xi_x & \eta_x & \zeta_x \\ \\ \xi_y & \eta_y & \zeta_y \\ \\ \xi_z & \eta_z & \zeta_z \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial \xi} \\ \\ \frac{\partial u}{\partial \eta} \\ \\ \frac{\partial u}{\partial \zeta} \end{bmatrix} \tag{49}$$

Transforming in the opposite direction, we have the equations

$$\begin{bmatrix} \frac{\partial u}{\partial \xi} \\ \\ \frac{\partial u}{\partial \eta} \\ \\ \frac{\partial u}{\partial \zeta} \end{bmatrix} = \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ \\ x_\eta & y_\eta & z_\eta \\ \\ x_\zeta & y_\zeta & z_\zeta \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial x} \\ \\ \frac{\partial u}{\partial y} \\ \\ \frac{\partial u}{\partial z} \end{bmatrix} \tag{50}$$

By substituting (50) into (49), we have the identity

$$\begin{bmatrix} \frac{\partial u}{\partial x} \\ \\ \frac{\partial u}{\partial y} \\ \\ \frac{\partial u}{\partial z} \end{bmatrix} = \begin{bmatrix} \xi_x & \eta_x & \zeta_x \\ \\ \xi_y & \eta_y & \zeta_y \\ \\ \xi_z & \eta_z & \zeta_z \end{bmatrix} \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ \\ x_\eta & y_\eta & z_\eta \\ \\ x_\zeta & y_\zeta & z_\zeta \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial x} \\ \\ \frac{\partial u}{\partial y} \\ \\ \frac{\partial u}{\partial z} \end{bmatrix} \tag{51}$$

From looking at (51), the product of the two Jacobian matrices has to be the identity matrix. Locally, on the grid, we require these Jacobian matrices to be non-singular. Therefore, for a non-singular coordinate transformation, we have that

$$\begin{bmatrix} \xi_x & \eta_x & \zeta_x \\ \\ \xi_y & \eta_y & \zeta_y \\ \\ \xi_z & \eta_z & \zeta_z \end{bmatrix} = \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ \\ x_\eta & y_\eta & z_\eta \\ \\ x_\zeta & y_\zeta & z_\zeta \end{bmatrix}^{-1} \tag{52}$$

Quantities in the right hand matrix may be computed directly from the grid point coordinates for the finite volume cells. The inversion of this Jacobian matrix renders the transformation derivatives needed for calculating Cartesian derivatives on the curvilinear grid. It is these derivatives that appear in (48) and hence (46). For brevity, we have applied a shorthand notation in that

$$\frac{\partial u}{\partial x} = u_x; \quad \frac{\partial \xi}{\partial x} = \xi_x \tag{53}$$

and so forth. With this notation, the derivatives in the Cartesian gradient form (26) may be written

$$\begin{aligned} u_x &= u_\xi \xi_x + u_\eta \eta_x + u_\zeta \zeta_x \\ u_y &= u_\xi \xi_y + u_\eta \eta_y + u_\zeta \zeta_y \\ u_z &= u_\xi \xi_z + u_\eta \eta_z + u_\zeta \zeta_z \end{aligned} \tag{54}$$

Now substitute the derivatives in (54) to the common term $\Delta u \cdot \hat{n}$ found in the flux formulas (42), (43) and (44). We obtain

$$\begin{aligned} \nabla u \cdot \hat{n} = \Big[&\hat{i}(u_\xi \xi_x + u_\eta \eta_x + u_\zeta \zeta_x) + \hat{j}(u_\xi \xi_y + u_\eta \eta_y + u_\zeta \zeta_y) \\ &+ \hat{k}(u_\xi \xi_z + u_\eta \eta_z + u_\zeta \zeta_z) \Big] \cdot (\hat{i} n_x + \hat{j} n_y + \hat{k} n_z) \end{aligned} \tag{55}$$

By simplifying and collecting terms, we have that

$$\begin{aligned}\nabla u \cdot \hat{n} = u_\xi(\xi_x n_x + \xi_y n_y + \xi_z n_z) \\ + u_\eta(\eta_x n_x + \eta_y n_y + \eta_z n_z) \\ + u_\zeta(\zeta_x n_x + \zeta_y n_y + \zeta_z n_z)\end{aligned} \tag{56}$$

The terms in parentheses each have the form of an inner product; let us exploit this form to develop another shorthand notation. Let

$$\begin{aligned}\xi_{\vec{x}} \cdot \hat{n} &= \xi_x n_x + \xi_y n_y + \xi_z n_z \\ \eta_{\vec{x}} \cdot \hat{n} &= \eta_x n_x + \eta_y n_y + \eta_z n_z \\ \zeta_{\vec{x}} \cdot \hat{n} &= \zeta_x n_x + \zeta_y n_y + \zeta_z n_z\end{aligned} \tag{57}$$

If we substitute these relations into (56), we obtain

$$\nabla u \cdot \hat{n} = u_\xi(\xi_{\vec{x}} \cdot \hat{n}) + u_\eta(\eta_{\vec{x}} \cdot \hat{n}) + u_\zeta(\zeta_{\vec{x}} \cdot \hat{n}) \tag{58}$$

Now the above relation is substituted into the i, j and k flux formulas (42), (43) and (44). The i flux becomes

$$\begin{aligned}F_i = K[u_\xi(\xi_{\vec{x}} \cdot \hat{n}) + u_\eta(\eta_{\vec{x}} \cdot \hat{n}) + u_\zeta(\zeta_{\vec{x}} \cdot \hat{n})]\Delta a|_{j,k}^{i+1} \\ - \ K[u_\xi(\xi_{\vec{x}} \cdot \hat{n}) + u_\eta(\eta_{\vec{x}} \cdot \hat{n}) + u_\zeta(\zeta_{\vec{x}} \cdot \hat{n})]\Delta a|_{j,k}^{i}\end{aligned} \tag{59}$$

Rearranging these terms produces an interesting result.

$$\begin{aligned}F_i = (Ku_\xi(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i+1} - (Ku_\xi(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i} \\ + (Ku_\eta(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i+1} - (Ku_\eta(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i} \\ + (Ku_\zeta(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i+1} - (Ku_\zeta(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i}\end{aligned} \tag{60}$$

The first two terms in (60) represent the changes in u exclusively in the ξ direction. These terms may be identified by the presence of u_ξ and $\xi_{\vec{x}}$ in conjunction with the upper index $i+1$ or i representing the cell side. I call this type of change *longitudinal*, and it is of primary importance because it involves changes in u along the ξ coordinate. The remaining terms are changes in u via terms η and ζ effecting some change in the ξ direction again as given by the upper evaluation limits $i+1$ and i. These terms I designate as *transverse*. I believe these terms to be of secondary importance. Now consider the j flux as in (43). By substituting (58), we have that

$$\begin{aligned}F_j = K[u_\xi(\xi_{\vec{x}} \cdot \hat{n}) + u_\eta(\eta_{\vec{x}} \cdot \hat{n}) + u_\zeta(\zeta_{\vec{x}} \cdot \hat{n})]\Delta a|_{i,k}^{j+1} \\ - K[u_\xi(\xi_{\vec{x}} \cdot \hat{n}) + u_\eta(\eta_{\vec{x}} \cdot \hat{n}) + u_\zeta(\zeta_{\vec{x}} \cdot \hat{n})]\Delta a|_{i,k}^{j}\end{aligned} \tag{61}$$

Again, we reorganize this expression to achieve

$$\begin{aligned} F_j = &(Ku_\xi(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j+1} - (Ku_\xi(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j} \\ &+ (Ku_\eta(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j+1} - (Ku_\eta(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j} \\ &+ (Ku_\zeta(\zeta_{\vec{x}} \cdot \hat{n})]\Delta a|_{i,k}^{j+1} - (Ku_\zeta(\zeta_{\vec{x}} \cdot \hat{n})]\Delta a|_{i,k}^{j} \end{aligned} \tag{62}$$

If we examine (62), we see that the first two terms convey variation along coordinate ξ but are computed at the upper and lower η interfaces denoted by the upper indices $j+1$ or j. Since the coordinate direction is different from the direction of the cell side, these changes are transverse, not longitudinal. The second set of terms describes change in the η direction computed at the η cell interfaces. As a result, this change is longitudinal. Similarly, the last two terms in (62) detail changes in u along the ζ coordinate but measured at the η interfaces identified by the η cell interfaces. As a result, these terms describe transverse flux contributions.

Finally, we examine F_k, the flux in the ζ coordinate direction. Recall (44) with the substitution (58).

$$\begin{aligned} F_k = &\ K[u_\xi(\xi_{\vec{x}} \cdot \hat{n}) + u_\eta(\eta_{\vec{x}} \cdot \hat{n}) + u_\zeta(\zeta_{\vec{x}} \cdot \hat{n})]\Delta a|_{i,j}^{k+1} \\ &- K[u_\xi(\xi_{\vec{x}} \cdot \hat{n}) + u_\eta(\eta_{\vec{x}} \cdot \hat{n}) + u_\zeta(\zeta_{\vec{x}} \cdot \hat{n})]\Delta a|_{i,j}^{k} \end{aligned} \tag{63}$$

As with the preceding flux terms, we split (63) into contributions to the ζ flux respective of the differing curvilinear coordinate directions.

$$\begin{aligned} F_k = &(Ku_\xi(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k+1} - (Ku_\xi(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k} \\ &+ (Ku_\eta(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k+1} - (Ku_\eta(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k} \\ &+ (Ku_\zeta(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k+1} - (Ku_\zeta(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k} \end{aligned} \tag{64}$$

In manner analogous to the (ξ, i) and (η, j) flux equations, we note that for the ζ flux, the cell sides or interfaces are normal to the ζ or k direction. This fact is indicated by the upper evaluation limit ($k+1$ or k) shown in (64). It follows that the first two lines in (64) represent contributions from the (ξ, i) and (η, j) directions. As a result, these flux contributions are transverse, not longitudinal. The remaining flux contribution is from the (ζ, k) direction, so it is longitudinal and regarded as a primary ζ flux component. Note that the six terms in each of (60), (62) and (64) may be computed from grid and solution information. Specific instructions for these computations follow in Section 3. Now a semi-discrete formula for (40) is easily written as

$$\frac{du_{i,j,k}}{dt} = \frac{1}{\rho\, C_P V_{i,j,k}}(F_i + F_j + F_k) \tag{65}$$

It is important to observe that the volume integration has removed the spatial dependency of $u_{i,j,k}$ from the cell itself, so as indicated earlier, this value of u is an average value for cell

(i,j,k). More importantly, $u_{i,j,k}$ is purely a time-dependent quantity making (65) an ordinary differential equation. Later, we examine techniques available for integrating this type of equation.

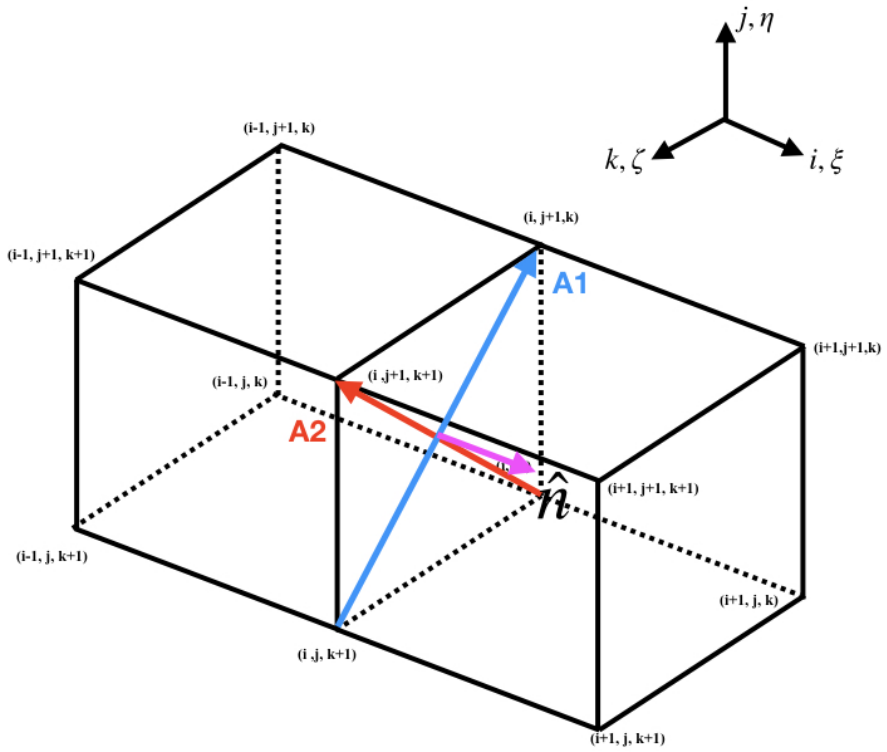


Figure 3: Geometry for ξ Surface Cartesian Coordinate Derivatives

3 Computational Details

Three dimensional computational physics codes tend to require a great many calculations, and the heat conduction code described here is no exception. Also, if we consider the logic involved in performing these calculations, even more computational effort is needed. Here, we concentrate on required geometric and differential calculations, and these are grouped by type.

3.1 Curvilinear Transformation Details

The crux of transitioning from the physical space to the computational space lies in the curvilinear coordinate transformation. In particular, the Cartesian coordinate derivatives are needed for the sides (or interfaces) of each finite volume cell. These derivatives are x_ξ, x_η, x_ζ, y_ξ, y_η, y_ζ, z_ξ, z_η and z_ζ as in (50). Let us consider the constant ξ, i cell side or interface. The geometry for this case is shown in Figure 3. These derivatives are formally computed at the center of the constant i interface (where the red and blue vectors cross). The interface normal vector $\hat{n}$ is denoted by a magenta arrow and points in the $+i$ direction. To calculate these derivatives, we use the notation $\vec{x} = (x, y, z)$ and construct Newton divided differences as shown below.

$$\vec{x}_\xi = \left(\frac{1}{8} \left[\vec{x}_{i+1,j,k} + \vec{x}_{i+1,j+1,k} + \vec{x}_{i+1,j,k+1} + \vec{x}_{i+1,j+1,k+1} \right] \right.$$
$$\left. -\frac{1}{8} \left[\vec{x}_{i-1,j,k} + \vec{x}_{i-1,j+1,k} + \vec{x}_{i-1,j,k+1} + \vec{x}_{i-1,j+1,k+1} \right] \right) / (\Delta\eta) \tag{66}$$

$$\vec{x}_\eta = \frac{1}{2} \left[\vec{x}_{i,j+1,k+1} + \vec{x}_{i,j+1,k} - (\vec{x}_{i,j,k+1} + \vec{x}_{i,j,k}) \right] / (\Delta\eta) \tag{67}$$

$$\vec{x}_\zeta = \frac{1}{2} \left[\vec{x}_{i,j,k+1} + \vec{x}_{i,j+1,k+1} - (\vec{x}_{i,j,k} + \vec{x}_{i,j+1,k}) \right] / (\Delta\zeta) \tag{68}$$

Computational mesh increments, $\Delta\xi$, etc. are included for completeness. Normally, they are assigned the value of unity. The area vector for the cell side is given by

$$A\,\hat{n} = \frac{1}{2} \vec{a}_1 \times \vec{a}_2 \tag{69}$$

where $\hat{n}$ is the unit normal for the cell interface, and

$$\vec{a}_1 = \vec{x}(i, j+1, k) - \vec{x}(i, j, k+1) \tag{70}$$
$$\vec{a}_2 = \vec{x}(i, j+1, k+1) - \vec{x}(i, j, k) \tag{71}$$

The derivatives above along with the cell side normal vector components (and the attendant area) are derived for use in (60). Accompanying these grid based derivatives are the associated curvilinear derivatives of u. For the i or ξ flux, these derivatives are computed by using the following formulas. Derivatives at the $i+1$ interface are

$$u_\xi|_{jk}^{i+1} = u_{i+1,j,k} - u_{i,j,k} \tag{72}$$
$$u_\eta|_{jk}^{i+1} = \frac{1}{4}(u_{i,j+1,k} - u_{i+1,j+1,k} - u_{i,j-1,k} - u_{i+1,j-1,k}) \tag{73}$$
$$u_\zeta|_{jk}^{i+1} = \frac{1}{4}(u_{i,j,k+1} + u_{i+1,j,k+1} - u_{i,j,k-1} - u_{i+1,j,k-1}) \tag{74}$$

Computational mesh increments are omitted from these formulas in order to remove notational clutter. Still, their presence is implied. Curvilinear derivatives of u at the i interface are given as

$$u_\xi|_{jk}^{i} = u_{i,j,k} - u_{i-1,j,k} \tag{75}$$
$$u_\eta|_{jk}^{i} = \frac{1}{4}(u_{i-1,j+1,k} + u_{i,j+1,k} - u_{i-1,j-1,k} - u_{i,j-1,k}) \tag{76}$$
$$u_\zeta|_{jk}^{i} = \frac{1}{4}(u_{i-1,j,k+1} + u_{i,j,k+1} - u_{i-1,j,k-1} - u_{i,j,k-1}) \tag{77}$$

Note that (72) and (75) are primary derivatives of u along the ξ curvilinear coordinate.

For the η or j cell interfaces similar formulas exist for curvilinear derivatives of Cartesian coordinates. A figure similar to Figure 3 can be constructed for these derivatives, but it would look the same as Figure 3 except it would be oriented along the η coordinate with the appropriate grid points centered around the j interface. The derivatives of the Cartesian coordinates are shown below.

$$\vec{x}_\xi = \frac{1}{2}\left[\vec{x}_{i+1,j,k+1} + \vec{x}_{i+1,j,k} - \vec{x}_{i,j,k+1} - \vec{x}_{i,j,k}\right]/(\Delta\xi) \tag{78}$$

$$\begin{aligned}\vec{x}_\eta = \Bigg(&\frac{1}{8}\left[\vec{x}_{i,j+1,k} + \vec{x}_{i,j+1,k+1} + \vec{x}_{i+1,j+1,k} + \vec{x}_{i+1,j+1,k+1}\right] \\ &-\frac{1}{8}\left[\vec{x}_{i,j-1,k} + \vec{x}_{i,j-1,k+1} + \vec{x}_{i+1,j-1,k} + \vec{x}_{i+1,j-1,k+1}\right]\Bigg)/(\Delta\eta)\end{aligned} \tag{79}$$

$$\vec{x}_\zeta = \frac{1}{2}\left[\vec{x}_{i,j,k+1} + \vec{x}_{i+1,j,k+1} - \vec{x}_{i,j,k} - \vec{x}_{i+1,j,k}\right]/(\Delta\zeta) \tag{80}$$

The crossed vectors used to construct the cell side normal vector and surface area are

$$\vec{a}_1 = \vec{x}(i+1,j,k+1) - \vec{x}(i,j,k) \tag{81}$$

$$\vec{a}_2 = \vec{x}(i+1,j,k) - \vec{x}(i,j,k+1) \tag{82}$$

Of course, the formula (69) for calculating the area vector remains the same. It is also necessary to compute the curvilinear derivatives of the solution u. Those equations are provided below; note that as before, the upper limit on the evaluation bar indicates the associated cell interface (side). Again, computational mesh increment notation is excluded. So at the $j+1$ interface, we have

$$u_\xi|_{ik}^{j+1} = \frac{1}{4}(u_{i+1,j,k} - u_{i+1,j+1,k} - u_{i-1,j,k} - u_{i-1,j+1,k}) \tag{83}$$

$$u_\eta|_{ik}^{j+1} = u_{i,j+1,k} - u_{i,j,k} \tag{84}$$

$$u_\zeta|_{ik}^{j+1} = \frac{1}{4}(u_{i,j,k+1} + u_{i,j+1,k+1} - u_{i,j,k-1} - u_{i,j+1,k-1}) \tag{85}$$

At the lower j cell side, the equations are

$$u_\xi|_{ik}^{j} = \frac{1}{4}(u_{i+1,j-1,k} + u_{i+1,j,k} - u_{i-1,j-1,k} - u_{i-1,j,k}) \tag{86}$$

$$u_\eta|_{ik}^{j} = u_{i,j,k} - u_{i1,j-1,k} \tag{87}$$

$$u_\zeta|_{ik}^{j} = \frac{1}{4}(u_{i,j-1,k+1} + u_{i,j,k+1} - u_{i,j-1,k-1} - u_{i,j,k-1}) \tag{88}$$

Equations (84) and (87) are the primary or longitudinal curvilinear derivatives of u in the η direction.

Finally, we consider derivatives of the Cartesian grid coordinates at the ζ cell sides. These values are needed by flux formula (64).

$$\vec{x}_\xi = \frac{1}{2}\left[\vec{x}_{i+1,j,k} + \vec{x}_{i+1,j+1,k} - \vec{x}_{i,j+1,k} - \vec{x}_{i,j,k}\right]/(\Delta\xi) \tag{89}$$

$$\vec{x}_\eta = \frac{1}{2}\left[\vec{x}_{i,j+1,k} + \vec{x}_{i+1,j+1,k} - \vec{x}_{i,j,k} - \vec{x}_{i+1,j,k}\right]/(\Delta\eta) \tag{90}$$

$$\begin{aligned}\vec{x}_\zeta = \Bigg(&\frac{1}{8}\left[\vec{x}_{i,j,k+1} + \vec{x}_{i+1,j,k+1} + \vec{x}_{i,j+1,k+1} + \vec{x}_{i+1,j+1,k+1}\right] \\ &-\frac{1}{8}\left[\vec{x}_{i,j,k-1} + \vec{x}_{i+1,j,k-1} + \vec{x}_{i,j+1,k-1} + \vec{x}_{i+1,j+1,k-1}\right]\Bigg)/(\Delta\zeta)\end{aligned} \tag{91}$$

The crossed vectors used to construct the ζ cell side normal vector and surface area are

$$\vec{a}_1 = \vec{x}(i+1,j,k) - \vec{x}(i,j+1,k) \tag{92}$$
$$\vec{a}_2 = \vec{x}(i+1,j+1,k) - \vec{x}(i,j,k) \tag{93}$$

The curvilinear derivatives of u at the $k+1$ interface may be computed from the following formulas.

$$u_\xi|_{ij}^{k+1} = \frac{1}{4}(u_{i+1,j,k} - u_{i+1,j,k+1} - u_{i-1,j,k} - u_{i-1,j,k+1}) \tag{94}$$
$$u_\eta|_{ij}^{k+1} = \frac{1}{4}(u_{i,j+1,k} + u_{i,j+1,k+1} - u_{i,j-1,k} - u_{i,j-1,k+1}) \tag{95}$$
$$u_\zeta|_{ij}^{k+1} = u_{i,j,k+1} - u_{i,j,k} \tag{96}$$

At the k or lower ζ interface, we have the formulas

$$u_\xi|_{ij}^{k} = \frac{1}{4}(u_{i+1,j,k-1} + u_{i+1,j,k} - u_{i-1,j,k-1} - u_{i-1,j,k}) \tag{97}$$
$$u_\eta|_{ij}^{k} = \frac{1}{4}(u_{i,j+1,k-1} + u_{i,j+1,k} - u_{i,j-1,k-1} - u_{i,j-1,k}) \tag{98}$$
$$u_\zeta|_{ij}^{k} = u_{i,j,k} - u_{i,j,k-1} \tag{99}$$

Of course, (96) and (99) represent longitudinal derivatives of u in the ζ direction.

3.2 Explicit Time Propagation

The Heat Equation is time dependent, so methods are needed for accurately propagating the spatially discretized equations in time. We begin by recalling the semi-discrete form (65), i.e.,

$$\frac{du_{i,j,k}}{dt} = \frac{1}{\rho\, C_P V_{i,j,k}}(F_i + F_j + F_k) \tag{100}$$

Consider the present time solution indexed by time n. The time derivative of u can be represented by the divided difference

$$\frac{du_{i,j,k}}{dt} = \frac{u^{n+1}_{i,j,k} - u^{n}_{i,j,k}}{\Delta t} \tag{101}$$

So (100) becomes

$$\frac{u^{n+1}_{i,j,k} - u^{n}_{i,j,k}}{\Delta t} = \frac{1}{\rho\, C_P V_{i,j,k}}(F_i + F_j + F_k) \tag{102}$$

The simplest explicit time propagation method is the Euler explicit scheme. It can be written as

$$u^{n+1}_{i,j,k} = u^{n}_{i,j,k} + \frac{\Delta t}{\rho\, C_P V_{i,j,k}}\left[F_i(u^{n}_{i,j,k}) + F_j(u^{n}_{i,j,k}) + F_k(u^{n}_{i,j,k})\right] \tag{103}$$

This scheme requires a very small step size to remain stable, and it suffers from low accuracy (first order in time). It is still used, but I tend to employ it for early debugging work. A more accurate explicit scheme is the Modified Euler Method that I use extensively; it is especially effective in debugging. It is a scheme consisting of two steps that may be written as

$$u^{*}_{i,j,k} = u^{n}_{i,j,k} + \frac{\Delta t}{\rho\, C_P V_{i,j,k}}\left[F_i(u^{n}_{i,j,k}) + F_j(u^{n}_{i,j,k}) + F_k(u^{n}_{i,j,k})\right] \tag{104}$$

$$u^{n+1}_{i,j,k} = \frac{1}{2}\left[u^{*}_{i,j,k} + u^{n}_{i,j,k} + \frac{\Delta t}{\rho\, C_P V_{i,j,k}}\left(F_i(u^{*}_{i,j,k}) + F_j(u^{*}_{i,j,k}) + F_k(u^{*}_{i,j,k})\right)\right] \tag{105}$$

In (104),(105), the quantity u^* is a form of the numerical solution that is cast at a so-called *intermediate* time level.[13] Sometimes it is referred to as a predictor-corrector method. This method is rather easy to program and is quite accurate for small time steps. Of course, it does have stability restrictions. For 3D problems, the numerical time step must be kept small.

3.3 Implicit Time Propagation

One advantage of implicit time propagation schemes is to partially lift the restriction on the size of the time step. That is, in some cases it is desirable to advance a time dependent numerical solution by larger time steps. Perhaps one wishes to reach a desired end time more quickly while there is less interest in the details of the solution at intervening time levels. To construct an implicit scheme, the solution at the new time level is not only a function of the old time level, but it is also a function of itself at a select set of points in the integration stencil. Therefore, the numerical solution at the new time level is an *implicit* function of itself. A classic example of an implicit scheme is the Crank-Nicolson method frequently used for solving the Heat Equation.[5] Here we derive an implicit form for integrating the

Heat Equation that involves solving a heptadiagonal matrix system. We begin by recalling equations (65), (60), (62) and (64) from the preceding section. By substitution in (65), we obtain

$$\frac{du_{i,j,k}}{dt} = \frac{1}{\rho C_P V_{i,j,k}} \left(F_L + F_T \right) \tag{106}$$

where

$$\begin{aligned} F_L = &(Ku_\xi(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i+1} - (Ku_\xi(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i} \\ &+(Ku_\eta(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j+1} - (Ku_\eta(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j} \\ &+(Ku_\zeta(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k+1} - (Ku_\zeta(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k} \end{aligned} \tag{107}$$

and

$$\begin{aligned} F_T = &(Ku_\eta(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i+1} - (Ku_\eta(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i} \\ &+(Ku_\zeta(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i+1} - (Ku_\zeta(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i} \\ &+(Ku_\xi(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j+1} - (Ku_\xi(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j} \\ &+(Ku_\zeta(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j+1} - (Ku_\zeta(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j} \\ &+(Ku_\xi(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k+1} - (Ku_\xi(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k} \\ &+(Ku_\eta(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k+1} - (Ku_\eta(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k} \end{aligned} \tag{108}$$

F_L in (107) contains the longitudinal flux terms. These terms contain derivatives cast along the respective curvilinear coordinate directions. The transverse flux terms in F_T (108) contain derivatives cast transverse to the curvilinear coordinate directions. Intuitively, I regard the longitudinal terms as primary effectual fluxes. The transverse terms I regard to be of secondary importance. Of course, at present, these assertions are unproven. To derive the implicit difference equation, we apply temporal splitting to the longitudinal terms. Note that the individual flux terms are linear in the curvilinear coordinate derivatives u_ξ, u_η and u_ζ, so we can write a linear combination of a particular derivative at two different time levels say, n and $n+1$, e.g.,

$$u_\xi^{\text{split}} = \theta u_\xi^{n+1} + (1-\theta) u_\xi^{n} \tag{109}$$

where θ is a constant parameter chosen with $0 \leq \theta \leq 1$. Apply (109) to (107); we find that

$$\begin{aligned} F_L(u_{i,j,k}^{\text{split}}) = &(K\theta u_\xi^{n+1}(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i+1} + (K(1-\theta) u_\xi^{n}(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i+1} \\ &- (K\theta u_\xi^{n+1}(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i} - (K(1-\theta) u_\xi^{n}(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i} \\ &+ (K\theta u_\eta^{n+1}(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j+1} + (K(1-\theta) u_\eta^{n}(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j+1} \\ &- (K\theta u_\eta^{n+1}(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j} - (K(1-\theta) u_\eta^{n}(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j} \\ &+ (K\theta u_\zeta^{n+1}(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k+1} + (K(1-\theta) u_\zeta^{n}(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k+1} \\ &- (K\theta u_\zeta^{n+1}(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k} - (K(1-\theta) u_\zeta^{n}(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k} \end{aligned} \tag{110}$$

Equation (106) can be written as

$$\frac{du^n_{i,j,k}}{dt} = Q_{i,j,k}\left(F_L(u^{\text{split}}_{\vec{\xi}}) + F_T(u^n_{\vec{\xi}})\right) \tag{111}$$

where

$$Q_{i,j,k} = \frac{1}{\rho C_P V_{i,j,k}} \tag{112}$$

The ordinary time derivative of u may be discretized as

$$\frac{du^n_{i,j,k}}{dt} = \frac{u^{n+1}_{i,j,k} - u^n_{i,j,k}}{\Delta t} \tag{113}$$

By using (112) and (113) in (111), we can show that

$$u^{n+1}_{i,j,k} - u^n_{i,j,k} = q_{i,j,k}\left(F_L(u^{\text{split}}_{\vec{\xi}}) + F_T(u^n_{\vec{\xi}})\right) \tag{114}$$

where

$$q_{i,j,k} = \frac{\Delta t}{\rho C_P V_{i,j,k}} \tag{115}$$

Now we apply (110) in (114). The result is

$$\begin{aligned} u^{n+1}_{i,j,k} - q_{i,j,k}\,\theta\,\big[&(Ku^{n+1}_{\xi}(\xi_{\vec{x}}\cdot\hat{n})\Delta a|^{i+1}_{j,k} - (Ku^{n+1}_{\xi}(\xi_{\vec{x}}\cdot\hat{n})\Delta a|^{i}_{j,k} \\ &+(Ku^{n+1}_{\eta}(\eta_{\vec{x}}\cdot\hat{n})\Delta a|^{j+1}_{i,k} - (Ku^{n+1}_{\eta}(\eta_{\vec{x}}\cdot\hat{n})\Delta a|^{j}_{i,k} \\ &+(Ku^{n+1}_{\zeta}(\zeta_{\vec{x}}\cdot\hat{n})\Delta a|^{k+1}_{i,j} -(Ku^{n+1}_{\zeta}(\zeta_{\vec{x}}\cdot\hat{n})\Delta a|^{k}_{i,j}\big] \\ = u^n_{i,j,k} + q_{i,j,k}(1-\theta)\,\big[&(Ku^{n}_{\xi}(\xi_{\vec{x}}\cdot\hat{n})\Delta a|^{i+1}_{j,k} - (Ku^{n}_{\xi}(\xi_{\vec{x}}\cdot\hat{n})\Delta a|^{i}_{j,k} \\ &+(Ku^{n}_{\eta}(\eta_{\vec{x}}\cdot\hat{n})\Delta a|^{j+1}_{i,k} - (Ku^{n}_{\eta}(\eta_{\vec{x}}\cdot\hat{n})\Delta a|^{j}_{i,k} \\ &+(Ku^{n}_{\zeta}(\zeta_{\vec{x}}\cdot\hat{n})\Delta a|^{k+1}_{i,j} -(Ku^{n}_{\zeta}(\zeta_{\vec{x}}\cdot\hat{n})\Delta a|^{k}_{i,j}\big] \\ &+q_{i,j,k}\,F_T(u^n_{\vec{\xi}}) \end{aligned} \tag{116}$$

The above equation is split in the temporal sense; time level $n+1$ terms are all on the left side of the equation while time level n terms are located on the right side. The relative scaling between the longitudinal flux terms is dictated by the parameter θ. The transverse flux terms F_T are effectively known quantities at time level n. If θ is set at $1/2$, we retrieve a Crank-Nicolson type implicit difference equation.[5] Equation (116) is also nearly in the form of a linear system. The right side of the equation involves only time n terms. These terms are know quantities, so this part of the equation becomes the known right hand side of a linear system. That is to say, if a linear system is written as $A\vec{u}^{\,n+1} = \vec{b}$, then the right hand side of (116) is $\vec{b}$. To achieve the true linear system form, it is necessary to express the partial derivatives of u with algebraic combinations of the indexed values of $u_{i,j,k}$. The formulas for

these derivatives are provided above in Section 3.1. For convenience, the formulas occurring in the longitudinal flux terms are recapped here.

$$\begin{aligned} u_\xi|_{j,k}^{i+1} &= u_{i+1,j,k} - u_{i,j,k} \quad & u_\xi|_{j,k}^{i} &= u_{i,j,k} - u_{i-1,j,k} \\ u_\eta|_{i,k}^{j+1} &= u_{i,j+1,k} - u_{i,j,k} \quad & u_\eta|_{i,k}^{j} &= u_{i,j,k} - u_{i,j-1,k} \\ u_\zeta|_{i,j}^{k+1} &= u_{i,j,k+1} - u_{i,j,k} \quad & u_\zeta|_{i,j}^{k} &= u_{i,j,k} - u_{i,j,k-1} \end{aligned} \tag{117}$$

For brevity, denote the right hand side of (116) as $R_{i,j,k}^n$ and substitute the differences shown in (117). We then obtain

$$\begin{aligned} u_{i,j,k}^{n+1} - q_{i,j,k} \big[&(u_{i+1,j,k}^{n+1} - u_{i,j,k}^{n+1})(K(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i+1} - (u_{i,j,k}^{n+1} - u_{i-1,j,k}^{n+1})(K(\xi_{\vec{x}} \cdot \hat{n})\Delta a|_{j,k}^{i} \\ &(u_{i,j+1,k}^{n+1} - u_{i,j,k}^{n+1})(K(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j+1} - (u_{i,j,k}^{n+1} - u_{i,j-1,k}^{n+1})(K(\eta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,k}^{j} \\ &(u_{i,j,k+1}^{n+1} - u_{i,j,k}^{n+1})(K(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k+1} - (u_{i,j,k}^{n+1} - u_{i,j,k-1}^{n+1})(K(\zeta_{\vec{x}} \cdot \hat{n})\Delta a|_{i,j}^{k} \big] \\ &= R_{i,j,k}^n \end{aligned} \tag{118}$$

Now it is of practical importance to collect terms with common mesh values of u^{n+1}. By doing so, we obtain

$$\begin{aligned} &-q_{i,j,k}\,\theta\,(K\xi_{\vec{x}} \cdot \hat{n}\Delta a|_{j,k}^{i+1} \times u_{i+1,j,k}^{n+1} \\ &-q_{i,j,k}\,\theta\,(K\eta_{\vec{x}} \cdot \hat{n}\Delta a|_{i,k}^{j+1} \times u_{i,j+1,k}^{n+1} \\ &-q_{i,j,k}\,\theta\,(K\zeta_{\vec{x}} \cdot \hat{n}\Delta a|_{i,j}^{k+1} \times u_{i,j,k+1}^{n+1} \\ &+\big(\,1 + q_{i,j,k}\,\theta\,\big[\,(K\xi_{\vec{x}} \cdot \hat{n}\Delta a|_{j,k}^{i+1} + (K\eta_{\vec{x}} \cdot \hat{n}\Delta a|_{i,k}^{j+1} \\ &\quad +(K\zeta_{\vec{x}} \cdot \hat{n}\Delta a|_{i,j}^{k+1} + (K\zeta_{\vec{x}} \cdot \hat{n}\Delta a|_{i,j}^{k} \\ &\quad +(K\eta_{\vec{x}} \cdot \hat{n}\Delta a|_{i,k}^{j} + (K\xi_{\vec{x}} \cdot \hat{n}\Delta a|_{j,k}^{i}\,\big]\,\big) \times u_{i,j,k}^{n} \\ &-q_{i,j,k}\,\theta\,(K\zeta_{\vec{x}} \cdot \hat{n}\Delta a|_{i,j}^{k} \times u_{i,j,k-1}^{n+1} \\ &-q_{i,j,k}\,\theta\,(K\eta_{\vec{x}} \cdot \hat{n}\Delta a|_{i,k}^{j} \times u_{i,j-1,k}^{n+1} \\ &-q_{i,j,k}\,\theta\,(K\xi_{\vec{x}} \cdot \hat{n}\Delta a|_{j,k}^{i} \times u_{i-1,j,k}^{n+1} \\ &= R_{i,j,k}^n \end{aligned} \tag{119}$$

Equation (119) is in a form suitable for solution by either the Gauss-Seidel and Conjugate Gradient methods. It is even easier to program if we use coefficient notation. i.e.,

$$\begin{aligned} &-c_{i,j,k}u_{i+1,j,k}^{n+1} - e_{i,j,k}u_{i,j+1,k}^{n+1} - g_{i,j,k}u_{i,j,k+1}^{n+1} \\ &\quad +b_{i,j,k}u_{i,j,k}^{n+1} - f_{i,j,k}u_{i,j,k-1}^{n+1} - d_{i,j,k}u_{i,j-1,k}^{n+1} - a_{i,j,k}u_{i-1,j,k}^{n+1} = R_{i,j,k}^n \end{aligned} \tag{120}$$

for $1 \le i, j, k \le$ imax-1, jmax-1, kmax-1 where

$$c_{i,j,k} = q_{i,j,k}\,\theta\,(K\xi_{\vec{x}}\cdot\hat{n}\Delta a|_{j,k}^{i+1} \tag{121}$$

$$e_{i,j,k} = q_{i,j,k}\,\theta\,(K\eta_{\vec{x}}\cdot\hat{n}\Delta a|_{i,k}^{j+1} \tag{122}$$

$$g_{i,j,k} = q_{i,j,k}\,\theta\,(K\zeta_{\vec{x}}\cdot\hat{n}\Delta a|_{i,j}^{k+1} \tag{123}$$

$$\begin{aligned} b_{i,j,k} = \big(\, 1 + q_{i,j,k}\,\theta\, \big[\, & (K\xi_{\vec{x}}\cdot\hat{n}\Delta a|_{j,k}^{i+1} + (K\eta_{\vec{x}}\cdot\hat{n}\Delta a|_{i,k}^{j+1} \\ & +(K\zeta_{\vec{x}}\cdot\hat{n}\Delta a|_{i,j}^{k+1} + (K\zeta_{\vec{x}}\cdot\hat{n}\Delta a|_{i,j}^{k} \\ & +(K\eta_{\vec{x}}\cdot\hat{n}\Delta a|_{i,k}^{j} + (K\xi_{\vec{x}}\cdot\hat{n}\Delta a|_{j,k}^{i}\,\big]\,\big) \end{aligned} \tag{124}$$

$$f_{i,j,k} = q_{i,j,k}\,\theta\,(K\zeta_{\vec{x}}\cdot\hat{n}\Delta a|_{i,j}^{k} \tag{125}$$

$$d_{i,j,k} = q_{i,j,k}\,\theta\,(K\eta_{\vec{x}}\cdot\hat{n}\Delta a|_{i,k}^{j} \tag{126}$$

$$a_{i,j,k} = q_{i,j,k}\,\theta\,(K\xi_{\vec{x}}\cdot\hat{n}\Delta a|_{j,k}^{i} \tag{127}$$

As one may glean from its appearance, the above equations represent a heptadiagonal linear system. At index locations, $i, j, k = 0$ and $i, j, k =$, imax, jmax, kmax, we enforce Dirichlet boundary conditions on u.[3] The calculations involved in computing the above coefficients and other terms in (119) are not difficult, per se, but they are tedious requiring much book-keeping. That aspect of this work is especially true if the transverse flux terms in F_T are included. Theory behind the Gauss-Seidel and standard Conjugate Gradient linear system solvers applied to generate results for the main test problem is not discussed in this report. Interested readers are referred to Reference [2] for a discussion of these algorithms. Additional references for these methods are found in its bibliography. In the next section, we discuss the set-up of the test problems solved by the algorithms presented thus far.

4 Test Problem Set-up

Given the amount of time required to accomplish the design and coding of the FVM algorithms used in this work, we consider rather simple test problems. Although exact solutions do exist for more complicated test problems, those solutions can be very difficult and time consuming to obtain. A primary reason for the difficulty is the need for more complicated orthogonal families of polynomials in a Fourier Series solution cast in the proper coordinate system. As usual, my grid resolution is kept fairly low in order to examine the strength of our solution schemes.

4.1 Single Domain Test Problem

Bitter experience in over forty years has taught me to proceed slowly when developing computational physics programming. Simpler problems should be solved thoroughly before advanced test cases. For this reason, we perform a single domain test first. The boundary conditions are simpler, and cross domain boundary mapping is not needed. Our FVM solver is designed as a multi-block computer program that loops over domains and solves them individually, so is a simple matter to set the maximum number of domains at ONE for this test.

The first test problem is the same 3D test case used in Reference [2]. A solid cube of material is immersed in a high temperature reservoir forming a time dependent heating problem. Actually, this scenario is more like a cartoon of the real problem since the heat source cannot respond to the out flow of heat. The discretized cube is initialized at a dimensionless temperature of one, and Dirichlet boundary conditions are enforced at the cube's surface to maintain it at a temperature of ten. The cube has unit dimensions, and it is represented by a uniform grid with 21 points along each of the three Cartesian directions. The set-up for this case is, as you can see, quite simple. The exact solution for this problem is presented in Reference [2]

4.2 Multiple Domain Test Problem

As a test of the basic FVM code, a single domain test is required at the most basic level. To go beyond that level, the multiple domain test is of fundamental importance in showing the strength of FVM in solving the heat conduction equation for complicated geometric objects, even those with more than one component. In this case, we have chosen to solve a simple heat conduction problem within a sphere of unit radius. This problem is very similar to the single domain test case because we apply a uniform high temperature to the exterior of the sphere originally set a uniform lower temperature. Again, the high temperature is chosen as ten while the lower temperature is set at one. This test problems is completely dimensionless.

For simplicity, It is my desire that the finite volume equations be adapted for a single type of cell volume. Also, individual component grids are structured to support counting index systems. The choice for the present work is a system of hexahedral (six-sided) grid cells. If the grid systems are set up properly for the solid sphere, we can avoid the singularity that

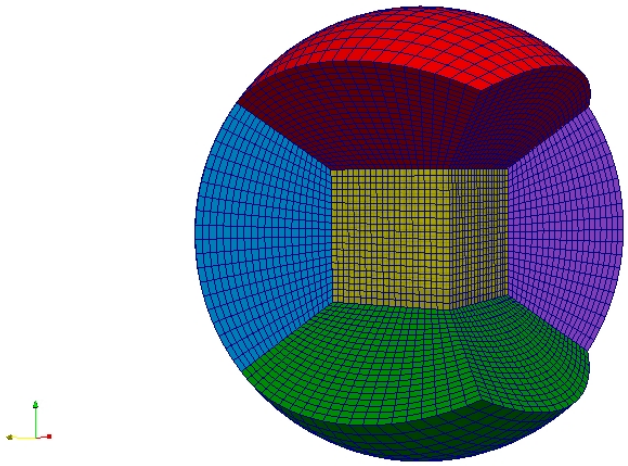


Figure 4: Grid domains for the sphere; domains 4 and 6 are not shown in order to clarify domain connectivity

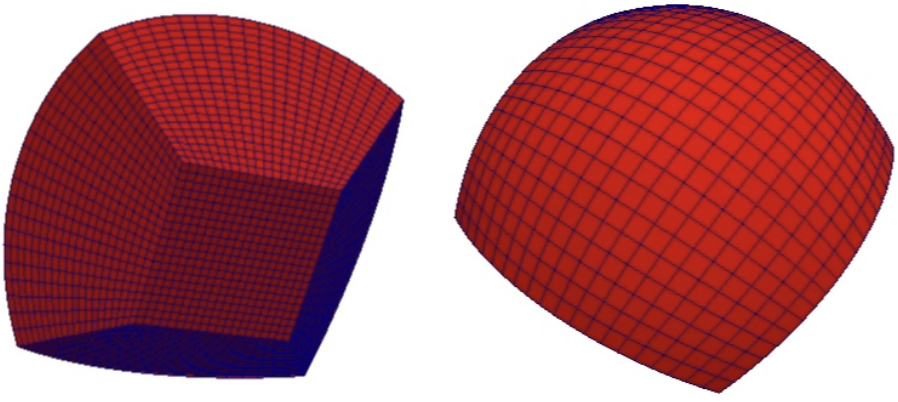


Figure 5: Shape of the curvilinear domain used to construct the outer sections of the sphere

exists at the origin of a spherical coordinate system. Here, we choose to represent the sphere by seven domains with each domain possessing six sides exactly like the individual finite volume cells. Figure 4 illustrates this idea for the sphere. The central domain (numbered as seven) is a cube, and each domain is represented by 21 points in each of three curvilinear directions and 8000 cells (volumes). The cube is centered at the origin and has a side length of 0.58428. Its volume is 0.1994 compared to the full sphere volume of 4.188. Actually, this volume is equal to the sphere's volume divided by 21. This an odd choice for the cube's volume, but it is a lukewarm attempt to smooth the radial grid spacing in transition to the grid volumes outside of cube. These six domains around the central cube all have the same geometric shape and size. The shape is illustrated by Figure 5, and its volume is 0.664. A FORTRAN code developed by the author is used to generate the seven grid domains.

The finite volume computer code solves for temperature in these domains one at a time, so the boundaries of these domains must communicate with their neighbors. Thus, numerical boundary condition coding is required to update a layer of boundary cells on each side of a given domain. This activity contains some of the tedious logic that I mentioned earlier in the report. A significant amount of debugging was required to correctly update these phantom cells. These updates are performed at the beginning of each time step. The exact solution for this test problem is derived in Appendix A.

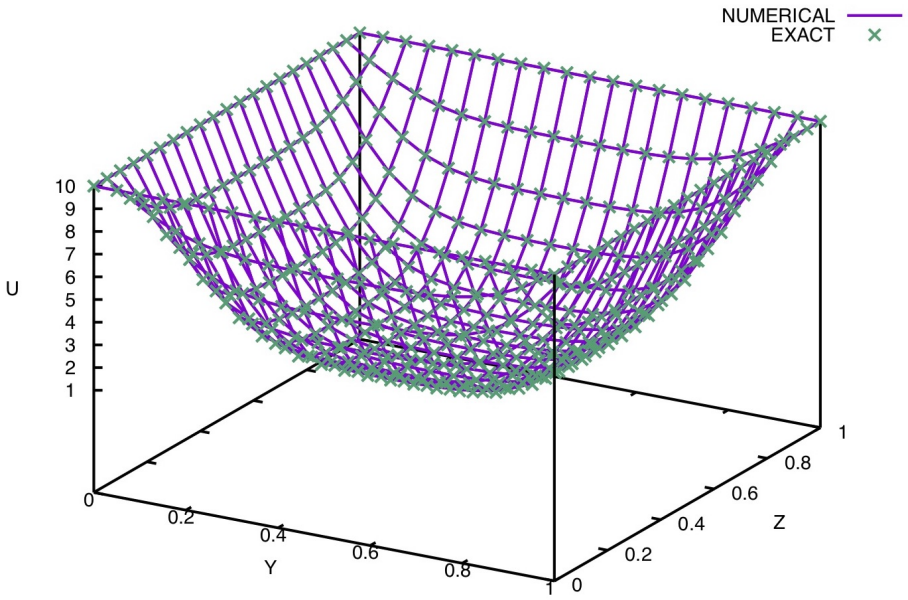


Figure 6: Numerical results for the first test problem at time 0.01 with explicit time propagation at i station 10; symbols X represent the exact solution

5 Results

In this section, the results for the two chosen test problems are presented. In each case, the temperature-like parameter is compared against the exact solution computed by Fourier series expansion. The computing resources utilized for this work are quite modest. An Apple MacBook Pro (Mid 2014) with a 2.6 GHz Intel Core i5 processor running MacOS Mojave Version 10.14.6 is employed for the calculations alonog with the GNU FORTRAN Compiler Version GCC 5.0.0 20141005.

5.1 Single Domain Test Problem

Computational results are shown at dimensionless time 0.01 for this test problem in Figure 6. The Modified Euler Explicit time propagation algorithm is employed. Since this is a 3D problem, the numerical solution is shown at $i = 10$ (or x index location 10). The exact solution is shown on the same plot. It is evident that the agreement between the numerical and exact solutions is quite good. Figure 7 contains the solution plots at dimensionless time 0.02. Again, the agreement between the numerical and exact solutions is very good. Since the explicit code adequately checks the functioning of the FVM and since the Cartesian coefficient matrix is symmetric, results for implicit time propagation are not shown here. If the reader is interested, for the finite difference method, implicit results are shown for this test problem in Reference [2].

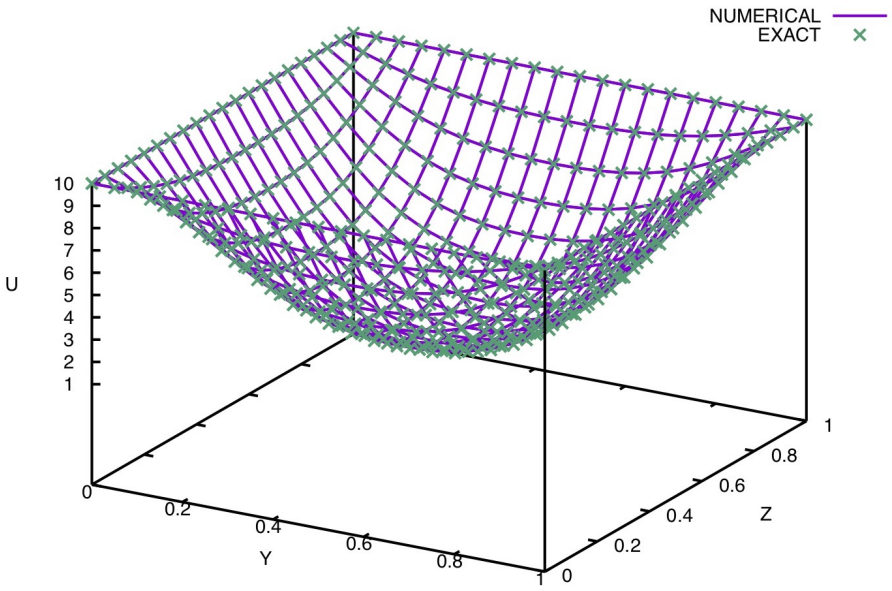


Figure 7: Numerical results for the first test problem at time 0.02 with explicit time propagation at i station 10; symbols X represent the exact solution

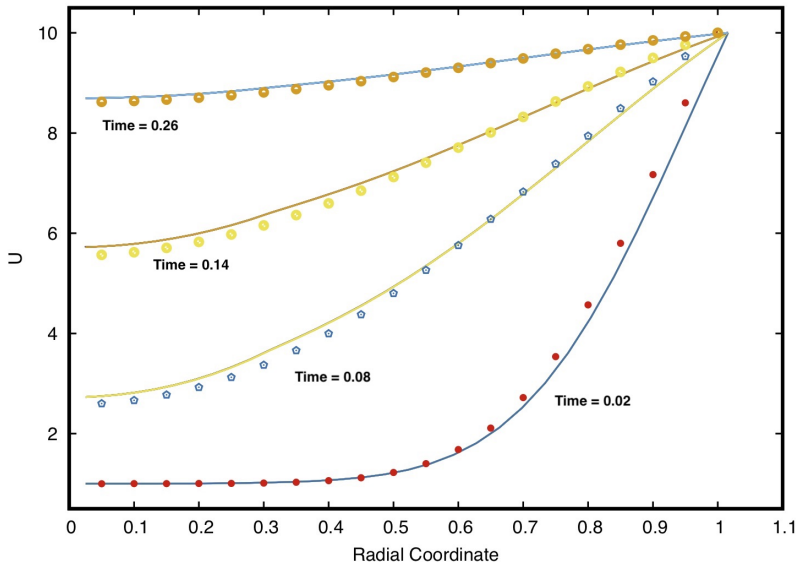


Figure 8: Numerical results for the spherical test problem at times 0.02, 0.08, 0.14 and 0.26 with explicit time propagation for the longitudinal flux model; solid lines represent the numerical solution; discrete symbols represent the exact solution

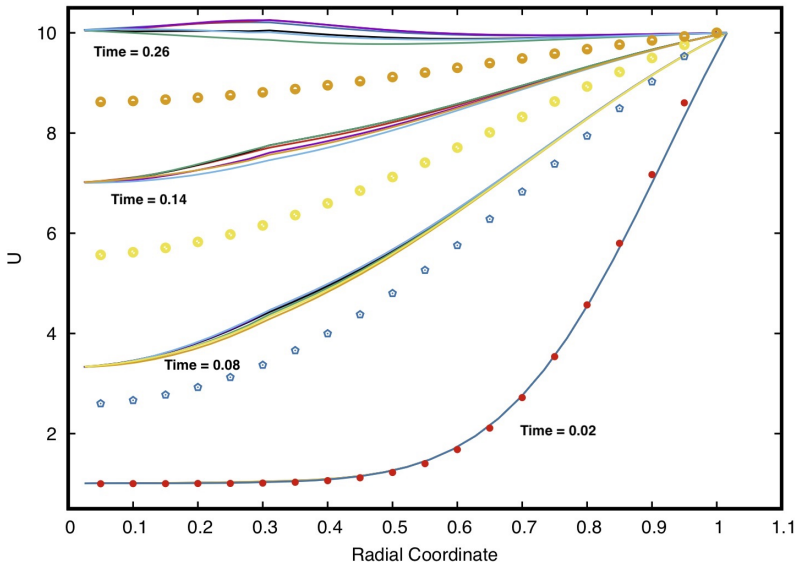


Figure 9: Numerical results for the spherical test problem at times 0.02, 0.08, 0.14 and 0.26 with explicit time propagation for the transverse flux model; solid lines represent the numerical solution; discrete symbols represent the exact solution

5.2 Multiple Domain Test Problem

As we discussed in the preceding section, a unit sphere has been divided into seven domains. The central cube is domain 7 while the surrounding 6 geometrically identical domains are oriented around the cube. With the arrangment of these domains, this test problem is spherically symmetric, so it is sufficient to present radial plots of temperature within the sphere's mass for different time levels. As it happens, we can obtain radial plots for each of the six sphere sections around the central cube. Calculations are performed for the longitudinal flux model alone and then with the transverse flux terms included.

It is reasonable to begin by presenting results for the explicit time propagation scheme. The time step utilized for this case is $\Delta t = 10^{-5}$. The numerical solution is propagated for 26,000 time steps or iterations. A sampling of the radial temperatures at dimensionless times 0.02, 0.08, 0.14 and 0.26 are shown in Figure 8 for the longitudinal flux model and compared against the exact solution. The agreement between these solutions is quite good. What is not directly evident from this plot is that the radial solutions are plotted for all six domains around the sphere. As you can see, these contours plot over one another in a nearly exact manner. Alone, this is a good result. The next result displayed in Figure 9 incorporates the transverse flux terms. In all candor, this result is a grave disappointment characterized by significant errors. The intent is that the transverse terms enhance the solution by including variation transverse to the main coordinate curves. Such poor numerical performance is not completely unexpected because the transverse geometric terms are oddly of a similar mag-

nitude to those for the longitudinal coordinates. The reason for this deficiency is unclear. Intuitively, I expect that these terms would be of lesser magnitude. It is obvious that the transverse terms, in this case, are not only of no use, but in fact, they are destructive. Still, the longitudinal flux model performs quite well, even at this lower level of grid resolution.

Now it is instructive to consider implicit time propagation of our FVM algorithm. A reason for employing an implicit algorithm is to allow the use of larger time steps while maintaining stability of the time marching scheme. A variant of the Crank-Nicolson scheme is applied in this case, and of course, it may suffer some of the disadvantages of its namesake.[2] Figure 10 contains the numerical solution for the sphere problem computed by using Gauss-Seidel iteration for the longitudinal flux model.[2] A time step of 10^{-4} is used, ten times that used for the explicit scheme. As you can see, agreement between the numerical and exact solutions is very good. A consideration in using an iterative linear solver for time propagation is that a number of iterations may be required for each time step. For this solution, the iteration count information is provided in Figure 11. As you can see, the work involved in the Gauss-Seidel algorithm is rather uniform requiring 9 to 12 iterations for convergence to the level of 10^{-10}. It is pleasant surprise to see just how well this scheme performs; that is especially true since convergence of the Gauss-Seidel algorithm is only proven for symmetric, positive definite matrices. In addtion, it is important to mention that symmetric Gauss-Seidel is implemented here. The direction of index progression reverses with each iteration. For brevity, the iteration count is only reported for domain 6. This test case is spherically symmetric, so the count should be basically the same for domains 1 through 6. The continued success with the longitudinal flux model is damped only by the failure of the transverse flux terms. For the Gauss-Seidel solver, the transverse model's poor performance is demonstrated in Figure 12. The problem with the transverse flux terms is not clear, but what is likely is that the derivatives of u are inaccurately estimated. The error also grows with solution time. Another possibility is a Crank-Nicolson instability (see Reference [2]) in conjunction with a perturbation in the grid existing at the interfaces between the central cube-shaped domain and the spherical sector domains. Smoothing the grid may remedy, in part, this trouble, but I have not investigated this idea.

We have also tested the implicit algorithm in our computer code by using the standard Conjugate Gradient linear solver.[2] Results for the longitudinal flux model are shown in Figure 13. As with the previous cases involving the longitudinal flux model, the agreement between the numerical and exact solutions is very good. The performance of the Conjugate Gradient solver may be examined by considering the number of iterations required for each time step. Figure 14 contains this information. It is evident that between 7 and 10 Conjugate Gradient iterations are required for this test problem. In this case, we can conclude that the Conjugate Gradient solver is a bit more efficient than the Gauss-Seidel solver. Again, the longitudinal flux model performs quite well while the transverse flux model performs poorly. The transverse model results are shown in Figure 15. It is unfortunate, but the results are equally as inaccurate (or just plain wrong) as those for the previous transverse cases. The upshot of these numerical experiments is that the longitudinal flux model works well for our 3D finite volume coding. The longitudinal flux equations are quite easy to program, much moreso than the transverse terms. The transverse flux model (really intended to be used in

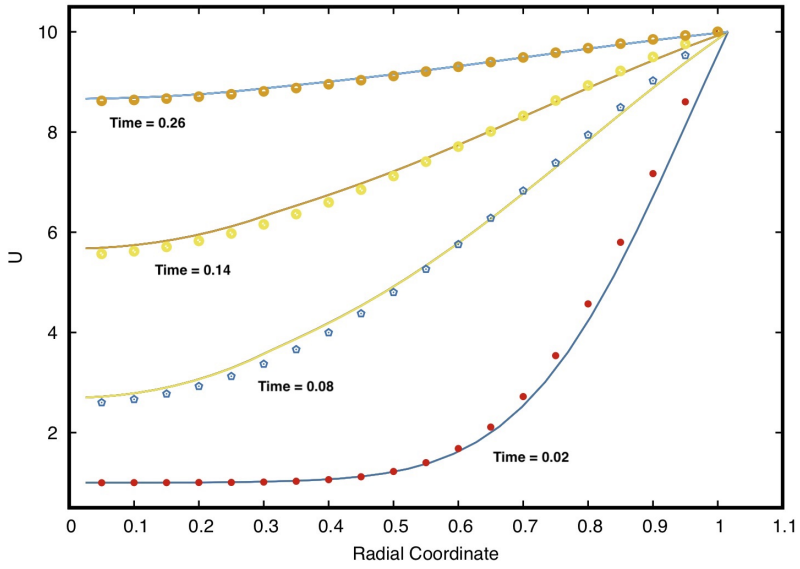


Figure 10: Numerical results for the spherical test problem at times 0.02, 0.08, 0.14 and 0.26 with implicit Gauss-Seidel time propagation for the longitudinal flux model; solid lines represent the numerical solution; discrete symbols represent the exact solution

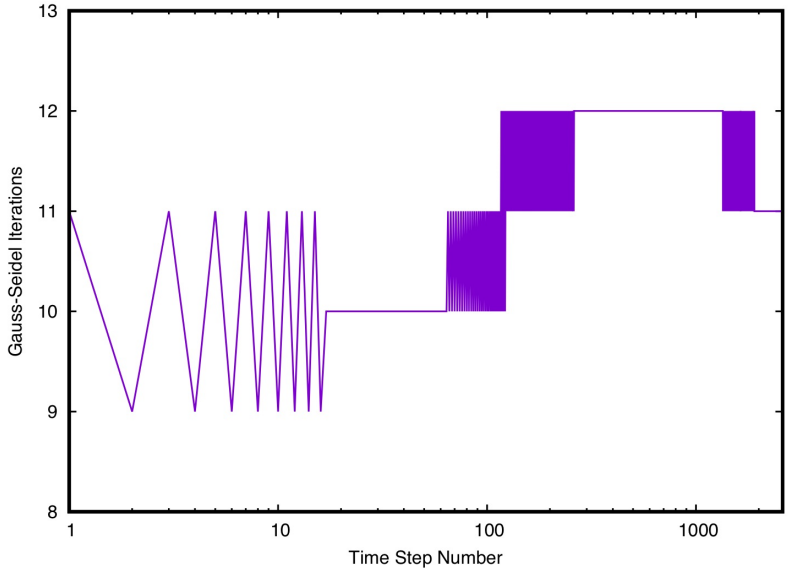


Figure 11: Count of Gauss-Seidel iterations required to advance the numerical solution at each time step

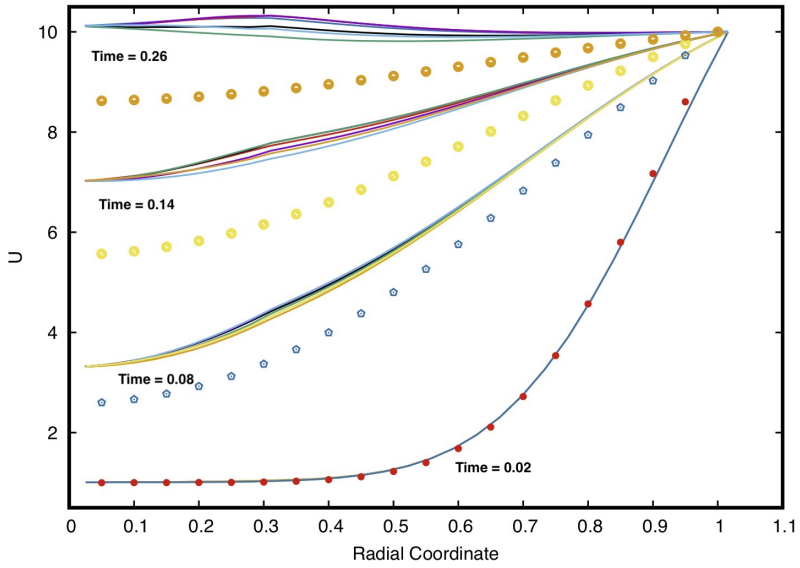


Figure 12: Numerical results for the spherical test problem at times 0.02, 0.08, 0.14 and 0.26 with implicit Gauss-Seidel time propagation for the transverse flux model; solid lines represent the numerical solution; discrete symbols represent the exact solution

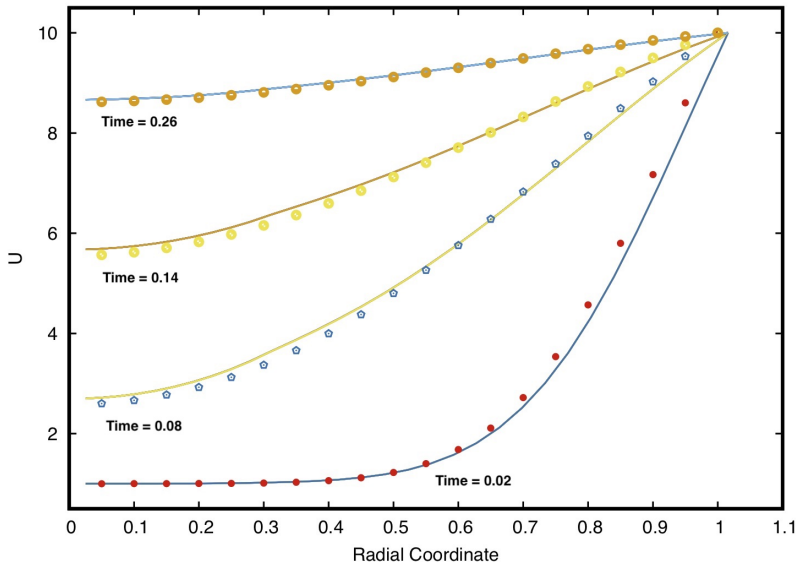


Figure 13: Numerical results for the spherical test problem at times 0.02, 0.08, 0.14 and 0.26 with implicit Conjugate Gradient time propagation for the longitudinal flux model; solid lines represent the numerical solution; discrete symbols represent the exact solution

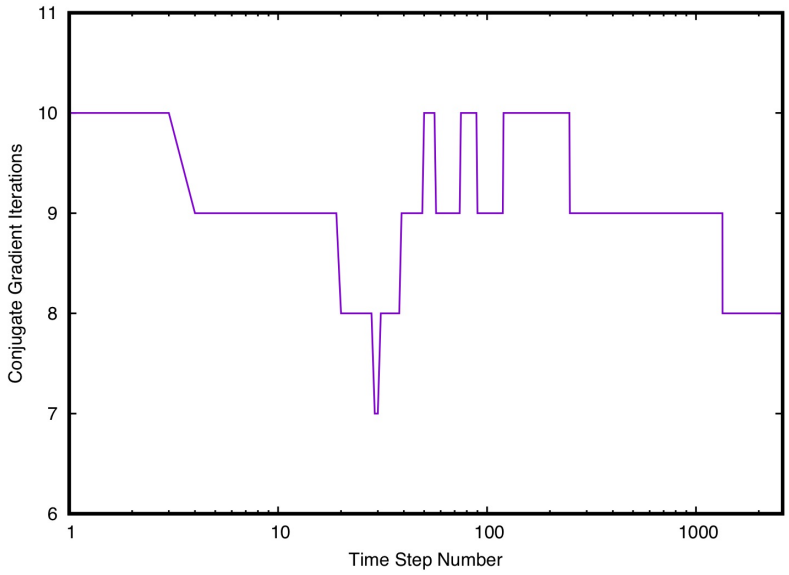


Figure 14: Count of Conjugate Gradient iterations required to advance the numerical solution at each time step

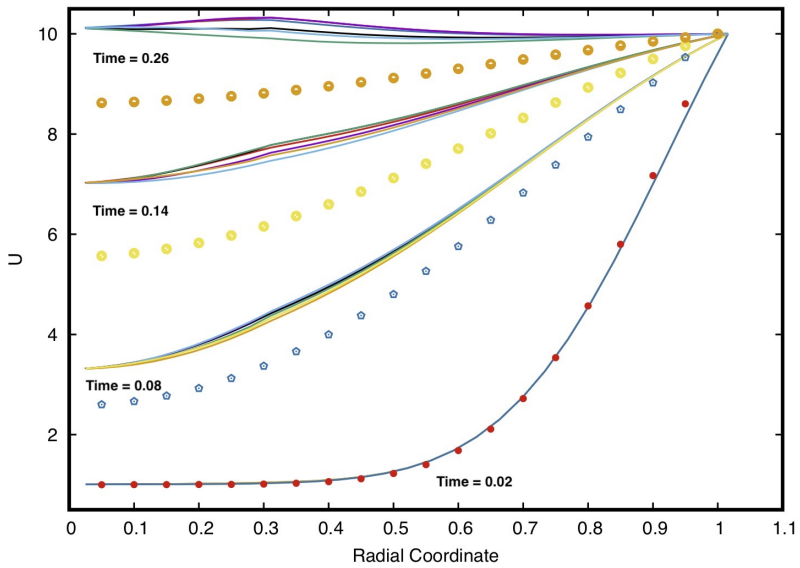


Figure 15: Numerical results for the spherical test problem at times 0.02, 0.08, 0.14 and 0.26 with implicit Conjugate Gradient time propagation for the transverse flux model; solid lines represent the numerical solution; discrete symbols represent the exact solution

conjunction with and improve the longitudinal flux model) is a complete disappointment.

6 Conclusions

This work is the third in a series concerned with the numerical solution of the heat conduction equation. You can think of these three reports as a Trilogy with a noticeable slant toward the Finite Volume Method. This work is not restricted to Cartesian geometries, nor is it confined to single block problems. The core example here is a representation of the unit sphere consisting of seven blocks. The finite volume equations are developed in detail for near-orthogonal curvilinear geometries. Many problems can be solved by using these equations. The restriction is that the equations are not developed for non-orthogonal geometries. Why? The reason is that I do not develop the formulas for covariant differentiation. That topic quickly becomes complicated and lies beyond the scope of this report. However, I do develop flux forms for near-orthogonal systems in three curvilinear directions. Also, for any curvilinear coordinate direction, the flux contributions are sorted into longitudinal (along the coordinate) and transverse (across the coordinate) models. The longitudinal model has proven to function quite well while the supplemental transverse model performs poorly. The cause of the failure of the transverse flux terms remains unclear. It is possible that the form of these terms provides excessive asymmetry in the overall scheme stencil. Perhaps greater grid refinement is needed. At present, the reason for this trouble is unknown.

On the other hand, the longitudinal flux solutions compare very well with the exact solution. The flux formulas are developed for calculation at the grid indices. Between each two flux positions (in a coordinate direction), we have a finite volume cell, so the scheme presented here is a cell-centered finite volume method. The cell-centered method is a bit easier to implement for multi-block problems. The block solutions are computed separately, but they are tied together at the boundary regions by boundary condition coding. The boundary conditions routine executes before each time step.

Also in the scope of this work, we have developed a fully explicit FVM heat equation solver based upon the Modified Euler method for time propagation. Although it is restricted to small time steps, this scheme is accurate and reliable. It has the added advantage of being very useful for testing new partial differential equation solvers. With the longitudinal flux scheme, this method performs quite well. In addition, an implicit solver has been developed based upon the Crank-Nicolson scheme that splits the main difference equation into explicit and implicit parts. We have chosen to create a heptadiagonal matrix system for 3D problems. This scheme has been successfully tested using a temporal step size ten times larger than is used by the explicit scheme. Linear system require matrix solutions at each time step, so we have tested two iterative matrix solvers for this task, a symmetric Gauss-Seidel scheme and the standard Conjugate Gradient method. Both of the these solvers perform well, but the Conjugate Gradient method has a slight edge in requiring fewer iterations to converge to the requested error tolerance. That these iterative methods converge well is a pleasant surprise because both methods are intended for symmetric, positive definite systems. I have not taken the time to characterize the matrices for the sphere problem, so I did not know *a priori* whether or not the systems would converge. I am happy to say that they did. Moreover, I feel vindicated since I have coded an implicit scheme that performs better than the ADI scheme I applied in my first report. There are few greater feelings of satisfaction

than that that is rendered by the realization of a long desired goal.

A common lingering question is "where do we go from here?" Perhaps I should work some more challenging problems with the code and incorporate variation within the coefficient of thermal conductivity. Maybe it would be nice to work a real heating problem and compare with data. Time could also be invested in determining the cause of the trouble with the transverse flux terms. An idea that I have not tested is that perhaps the transverse flux terms may function better on their own, without the presence of the longitudinal terms. At best, this notion is purely speculative, but it is an interesting thought. I must also come to terms with the possibility that I may have simply made errors in deriving these terms. After making a few separate derivations, I could find no errors, but human mistakes still occur. That makes me the usual and only suspect.

Of course, it is always interesting to rewrite the code for more general boundary conditions or for heat generation. There are many good projects to tackle, but I have been wanting to segue into numerically solving the Schrödinger equation because of its similarity to the heat equation. That project is very interesting and of significant importance in applications, e.g., the design of semiconductor and micro-miniaturized electronic devices. We can construct electronic devices that are so small that quantum effects becomes significant.[15] It is very important to have the capability for modeling these devices. Still, at some point, a fourth report may appear. Time will tell since I waited ten years to create this report. But as the great science fiction writer Sir Arthur C. Clarke once commented, "No trilogy should have more than four volumes". In context, Sir Arthur was referring to his famous novel, 2001: A Space Odyssey, published in 1968, and its sequels. But for now, I do not know if there will be a fourth report on solving the Heat Equation, but researching and writing these reports has been very instructive. It has also been a lot of fun. In any case, Sir Arthur was almost certainly right.

A An Exact Solution for the Sphere

In this appendix, we derive an exact solution for a sphere heated symmetrically. The statement of the Initial Boundary Value Problem is as follows. In part, we follow Özisik.[14]

$$\frac{1}{r}\frac{\partial^2(ru)}{\partial r^2} = \frac{1}{\alpha}\frac{\partial u}{\partial t}, \quad u = u(r,t), \;\; 0 < r < r_{\max}, \;\; t > 0 \tag{128}$$

$$u(r_{\max}, t) = T_2, \;\; t > 0 \tag{129}$$

$$u(y, 0) = T_1, \;\; 0 \leq r \leq r_{\max} \tag{130}$$

This problem is solvable by first applying a substitution. Let

$$\theta = ru \quad \theta = \theta(r,t) \tag{131}$$

If we multiply (128) by r and use the substitution, we obtain

$$\frac{\partial^2\theta}{\partial r^2} = \frac{1}{\alpha}\frac{\partial\theta}{\partial t} \tag{132}$$

Likewise, we may transform the boundary and initial conditions, i.e.,

$$\theta(r_{\max}, t) = r_{\max}\, T_2 \tag{133}$$

$$\theta(t, 0) = r\, T_1 \tag{134}$$

It is important to realize that the single boundary condition given as (133) is a bit of an oddity since two such conditiions are generally applied to solve a second order problem. Tacitly, we may observe that a second boundary condition does exist. It is the finiteness condition enforced at $r = 0$. That is,

$$\theta(0, t) < \infty \tag{135}$$

Indeed, the solution $\theta(r,t)$ must remain bounded at $r = 0$, and so must $u(0,t)$.

We note that the boundary conditions are non-trivial, so we use linearity to construct a solution consistent with these boundary conditions. It is prudent to apply a solution of the form[2]

$$\theta(r,t) = \theta_{\text{ss}}(r) + \theta_{\text{us}}(r,t) \tag{136}$$

where θ_{ss} is a steady state, time independent solution satisfying the boundary conditions (133,135). The function θ_{us} is concurrent and unsteady. We know that physics dictates that, in time, the sphere attains the steady state temperature T_2, so we can set

$$\theta_{\text{ss}}(r) = r\, T_2 \tag{137}$$

With this assertion, boundary conditions can easily be inferred for θ_{us}. From (136), we have that

$$\theta_{\text{us}}(r,t) = \theta(r,t) - \theta_{\text{ss}}(r) \tag{138}$$

By evaluating this equation at the outer boundary, we obtain

$$\theta_{\text{us}}(r_{\max},t) = \theta(r_{\max},t) - \theta_{\text{ss}}(r_{\max}) = r_{\max}T_2 - r_{\max}T_2 = 0 \tag{139}$$

Similarly, we can evaluate θ_{us} at $r = 0$. Observe that

$$\theta_{\text{us}}(0,t) = \theta(0,t) - \theta_{\text{ss}}(0) = 0 \cdot u - 0 \cdot T_2 = 0 \tag{140}$$

where u in the above equation is finite via (135). With these two results, it is evident that θ_{us} has trivial boundary conditions. Now, if we substitute (136) into the PDE, we have that

$$\frac{\partial^2\theta_{\text{ss}}(r)}{\partial r^2} + \frac{\partial^2\theta_{\text{us}}(r,t)}{\partial r^2} = \frac{1}{\alpha}\left(\frac{\partial\theta_{\text{ss}}(r)}{\partial t} + \frac{\partial\theta_{\text{us}}(r,t)}{\partial t}\right) \tag{141}$$

Note that the two partial derivative terms for θ_{ss} are zero, so we obtain the following PDE for θ_{us}.

$$\frac{\partial^2\theta_{\text{us}}(r,t)}{\partial r^2} = \frac{1}{\alpha}\frac{\partial\theta_{\text{us}}(r,t)}{\partial t} \tag{142}$$

Of course, this equation is effectively a clone for the PDE of $\theta(r,t)$ except it has the trivial boundary conditions

$$\theta_{\text{us}}(0,t) = 0 \tag{143}$$
$$\theta_{\text{us}}(r_{\max},t) = 0 \tag{144}$$

This problem is amenable to solution via separation of variables.[3] Let $\theta_{\text{us}} = R(r)\tau(t)$. By substituting this form into (142) and dividing by $R\,\tau$, we can obtain

$$\frac{R_{rr}}{R} = \frac{1}{\alpha}\frac{\tau_t}{\tau} = -\lambda^2 \tag{145}$$

Here, we have chosen the separation constant as $-\lambda^2$ as a matter of convenience. In so doing, we obtain two ordinary differential equations, i.e.,

$$\tau_t(t) + \alpha\lambda^2\tau(t) = 0 \tag{146}$$
$$R_{rr}(r) + \lambda^2 R(r) = 0 \tag{147}$$

If we assume that τ has exponential variation, we can deduce that

$$\tau(t) = A\exp(-\alpha\lambda^2 t) \tag{148}$$

where A is an arbitrary constant. If we also assume exponential variation of R, then we can deduce that

$$R(r) = B_1\sin(\lambda r) + B_2\cos(\lambda r) \tag{149}$$

By enforcing the boundary condition deduced from (143) to R, we see that $B_2 = 0$, so

$$R(r) = B_1 \sin(\lambda r) \tag{150}$$

By applying the condition on R deduced from (144), we have that

$$\sin(\lambda r_{\max}) = 0 \tag{151}$$

so we conclude that

$$\lambda r_{\max} = n\pi, \quad n = 1, 2, \ldots \tag{152}$$

or equivalently,

$$\lambda = \frac{n\pi}{r_{\max}}, \quad n = 1, 2, \ldots \tag{153}$$

By using (148) and (150) superposed as a product along with the eigenvalue (153), we can construct a solution for θ_{us}, i.e.,

$$\theta_{\text{us}}(r, t) = \sum_{n=1}^{\infty} G_n \exp\left[-\alpha\left(\frac{n^2\pi^2}{r_{\max}^2}\right)t\right] \sin\left(\frac{n\pi}{r_{\max}} r\right) \tag{154}$$

where the G_n are constants that must be chosen consistent with the initial conditions. Recalling (136), we can calculate these coefficients. Observe that

$$\theta_{\text{us}}(r, t) = \theta(r, t) - \theta_{\text{ss}}(r) \tag{155}$$

By evaluating this expression at $t = 0$, we can apply the initial condition, i.e.,

$$\theta_{\text{us}}(r, 0) = \theta(r, 0) - \theta_{\text{ss}}(r) \tag{156}$$

$$\sum_{n=1}^{\infty} G_n \sin\left(\frac{n\pi}{r_{\max}} r\right) = T_1 r - T_2 r = (T_1 - T_2) r \tag{157}$$

Coefficients G_n may be determined by applying orthogonality of the sine function. This calculation is not difficult but tedious, so the details are omitted. The result is stated as

$$G_n = -2(T_1 - T_2) r_{\max} \frac{\cos(n\pi)}{n\pi}, \quad n = 1, 2, \ldots \tag{158}$$

Again, by using (136), we can roll up the previous results into the general solution.

$$\theta(r, t) = rT_2 - 2(T_1 - T_2) r_{\max} \sum_{n=1}^{\infty} \frac{\cos(n\pi)}{n\pi} \exp\left[-\alpha\left(\frac{n^2\pi^2}{r_{\max}^2}\right)t\right] \sin\left(\frac{n\pi}{r_{\max}} r\right) \tag{159}$$

The goal of this analysis is to determine u, not θ. To do so, the above result must be divided by r. For this reason, it important to show that θ/r remains finite as $r \to 0$. To accomplish this, note that (159) can be used to write u as

$$u(r,t) = T_2 - 2(T_1 - T_2)\sum_{n=1}^{\infty}\cos(n\pi)\exp\left[-\alpha\left(\frac{n^2\pi^2}{r_{\max}^2}\right)t\right]\left[\frac{\sin\left(\frac{n\pi r}{r_{\max}}\right)}{\frac{n\pi r}{r_{\max}}}\right] \tag{160}$$

If we take the limit as r approaches zero for the right most bracketted term, we have the situation

$$\lim_{a\to 0}\frac{\sin(a)}{a} = 1 \tag{161}$$

As a result, this bracketted term remains finite as r tends to zero, so the series also remains finite. Remaining terms in the series are damped by the exponential for $t > 0$. For $t = 0$, the cosine series is self canceling and cannot grow, so the overall expression stay finite.

References

[1] 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.

[2] Nance, D.V., *Iterative Solvers for the Heat Conduction Equation*, Independent Research Technical Report, DOI: 10.13140/RG.2.2.21705.49765, February 2020.

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

[4] Hirsch, C., *Numerical Computation of Internal and External flows, Vol. 1, Fundamentals of Numerical Discertization*, John Wiley & Sons, New York, New York, 1988.

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

[6] 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.

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

[8] Fletcher, C.A.J., *Computational Techniques for Fluid Dynamics*, Vol. II, 2^{nd} Ed., Springer-Verlag, New York, New York, 1991.

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

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

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

[12] Thomas, G.B., Jr. and Finney, R.L., *Calculus and Analytic Geometry*, 5^{th} Ed., Addison-Wesley, Reading, Massachusetts, 1979.

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

[14] Özisik, M.N., *Boundary Value Problems of Heat Conduction*, Dover Publications, New York, N.Y., 1989.

[15] 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 devices", *Journal of Applied Physics*, Vol. 95, No. 12, 2004.