

The Pennsylvania State University

The Graduate School

College of Engineering

**A HIGHER-ORDER NUMERICAL FINITE VOLUME METHOD
FOR SOLID CONDUCTIVE HEAT TRANSFER**

A Thesis in

Mechanical Engineering

by

Cameron S. Nelson

© 2016 Cameron S. Nelson

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2016

The thesis of Cameron S. Nelson was reviewed and approved* by the following:

John M. Cimbala
Professor of Mechanical Engineering
Thesis Adviser

Daniel Haworth
Professor of Mechanical Engineering
Thesis Co-Adviser

Mary Frecker
Professor of Mechanical Engineering
Associate Head for Graduate Programs

*Signatures are on file in the Graduate School.

ABSTRACT

This thesis details the development of a higher-order numerical finite volume method for solid conductive heat transfer. Important thermodynamic and mathematical principles applied are conservation of energy, the heat diffusion equation, Fourier's Law, the thermal contact resistance concept, Taylor series expansion, and the first-order backward Euler time-differencing method. The higher-order method turns out to be second-order accurate in space and third-order accurate in time with respect to temperature.

TABLE OF CONTENTS

List of Figures	v
List of Tables	vi
Nomenclature	vii
Acknowledgments.....	ix
 Chapter 1 Introduction	 1
Chapter 2 Development of the Finite Volume Method	4
Chapter 2.1 First-Order Discretization Scheme for 1-D Conduction through a Solid	6
Chapter 2.2 Higher-Order Discretization Scheme for n-D Conduction through a Solid	8
 Chapter 3 Proof of Concept	 14
Chapter 3.1 Steady State 1-D Conduction through a Solid Rod	14
Chapter 3.2 Transient 1-D Conduction through a Solid Rod	17
Chapter 3.3 Convergence Test in both the Spatial and Temporal Domains	22
 Chapter 4 Summary and Conclusion	 25
 Appendix A Multiple Forms of Equation (2.23)	 27
Appendix B Derivation of Equation (2.28).....	28
Appendix C Specific Portions of the C# Code	29
Appendix C.1 Solid Volume Object	29
Appendix C.2 Conduction Face Object.....	34
Appendix C.3 Conduction Sector Object.....	42
 Bibliography	 48

LIST OF FIGURES

Figure 1.1: A typical control volume and the notation used for a Cartesian 2-D grid [1]	1
Figure 2.1: Pictorial representation of geometric entities and their associated scalar and vector unknown variables in 1-D	4
Figure 2.2: Assignment of unknown variables for first-order accurate discretization scheme for 1-D conduction through a solid	6
Figure 2.3: Assignment of unknown variables for a higher-order accurate discretization scheme for 1-D conduction through a solid	8
Figure 3.1: 1-D solid heat generation at steady state comparison between first-order and higher-order solutions.....	15
Figure 3.2: 1-D solid conduction with contact resistance at steady state.....	17
Figure 3.3: 1-D transient semi-infinite solid conduction with fixed temperature boundary condition.....	18
Figure 3.4: 1-D transient solid conductive diffusion of sinusoidal temperature profile	20

LIST OF TABLES

Table 3.1: Convergence study with refinement of grid for 1-D solid heat generation	16
Table 3.2: Convergence study in time domain for 1-D transient semi-infinite solid conduction	19
Table 3.3: Convergence study in time domain for 1-D transient solid conductive diffusion of sinusoidal temperature profile at $x = L/2$	21
Table 3.4: Spatial convergence study for 1-D heat generation with fixed end temperatures with coarse temporal step.....	22
Table 3.5: Spatial convergence study for 1-D heat generation with fixed end temperatures...	23
Table 3.6: Temporal convergence study for 1-D heat generation with fixed end temperatures	23
Table 3.7: Spatial/temporal convergence study for 1-D heat generation with fixed end temperatures	24
Table A.1: Coefficients of the terms in equation (2.23) for different values of A , B , C , and D	27

NOMENCLATURE

Symbol	Description	English Units
A	Coefficient A , multiplied to equations (2.18) and (B.1)	—
A_{F01}	Area of face 01	ft^2
B	Coefficient B , multiplied to equations (2.17) and (B.2)	—
$c_{p,v0}^t$	Specific heat capacity at constant pressure of volume 0 at time t	$\frac{\text{Btu}}{\text{lb}_m \text{ } ^\circ\text{F}}$
C	Coefficient C , multiplied to equation (2.20)	—
D	Coefficient D , multiplied to equation (2.19)	—
$F(\eta)$	Similarity function with nondimensional variable η	$^\circ\text{F}$
h	Characteristic distance of grid	ft
$HOTs$	Higher-order terms, approximated as negligible	$^\circ\text{F}$
k	Thermal conductivity	$\frac{\text{Btu}}{\text{hr ft } ^\circ\text{F}}$
L	Length of rod	ft
L_c	Heat transfer or conduction length	ft
p	Order of convergence	—
q_{in}	Heat rate in	$\frac{\text{Btu}}{\text{hr}}$
$q''_{j,i+1}$	Outward normal heat flux at x_j -position $i + 1$	$\frac{\text{Btu}}{\text{hr ft}^2}$
q'''_{gen}	Heat generation per volume	$\frac{\text{Btu}}{\text{hr ft}^3}$
r_{cont}	Contact resistance between faces	$\frac{\text{hr ft}^2 \text{ } ^\circ\text{F}}{\text{Btu}}$

Symbol	Description	English Units
Δt	Time step, difference in time between time $t + 1$ and time t	hr
t^*	Nondimensional time	—
T_i	Initial temperature of rod	$^{\circ}\text{F}$
T_o	Baseline temperature of rod	$^{\circ}\text{F}$
$(T_s - T_i)$	Step change in temperature for equations (3.4) and (3.6)	$^{\circ}\text{F}$
$(T_s - T_o)$	Temperature amplitude for equations (3.8) and (3.9)	$^{\circ}\text{F}$
ΔT_{i-1}	Change in temperature at x_j -position $i - 1$	$^{\circ}\text{F}$
$\left. \frac{dT}{dt} \right _i^{t+1}$	Differential of temperature w.r.t. time at x_j -position i and time $t + 1$	$\frac{^{\circ}\text{F}}{\text{hr}}$
$\left. \frac{d^2T}{dx_j^2} \right _i^t$	Second derivative of temperature w.r.t. x_j at x_j -position i and time t	$\frac{^{\circ}\text{F}}{\text{ft}^2}$
V	Volume	ft^3
x_j	x_j -position, in tensor notation	ft
α	Thermal diffusivity	$\frac{\text{ft}^2}{\text{hr}}$
ϵ_h^d	Leading-order truncation error	$^{\circ}\text{F}$
η	Nondimensional variable for similarity solution	—
$\vartheta(L_c^6)$	Sixth-order spatial term, approximated as negligible	$^{\circ}\text{F}$
$\vartheta(\Delta t^3)$	Third-order temporal term, approximated as negligible	$^{\circ}\text{F}$
ρ	Density	$\frac{\text{lb}_m}{\text{ft}^3}$
Φ_{2h}	Solution at a point in space and time with h doubled	$^{\circ}\text{F}$

ACKNOWLEDGMENTS

I am thankful for the opportunity to pursue higher education at Penn State, which was made possible by first and foremost the Lord Jesus Christ, and second by my family, specifically, my dad and mom, Andrew and Paula Nelson.

Throughout the completion of this thesis, several people have given valuable lessons and guidance on this subject. Those include, but are not limited to, Dr. John Cimbala, Mr. Douglas Kester, and Dr. Daniel Haworth. Thank you.

Chapter 1

Introduction

The finite volume method is useful for numerically representing partial differential equations in space, and performs particularly well at applying conservation laws. As the name suggests, the method divides a spatial body into a finite number of adjacent control volumes [1]. In the case of pure conductive heat transfer, the First Law of Thermodynamics (conservation of energy) requires that for each solid volume, the change in the energy stored in a volume must be equal to the net energy entering it [2]. Likewise, the energy flow, or heat flux exiting a face of the control volume must be equal to the energy entering the adjacent/coincident face of the neighboring control volume. Unknown variables and solid properties are assigned at nodes in the geometry, which are more commonly located at the volumetric/areal centroids of the volumes and faces.

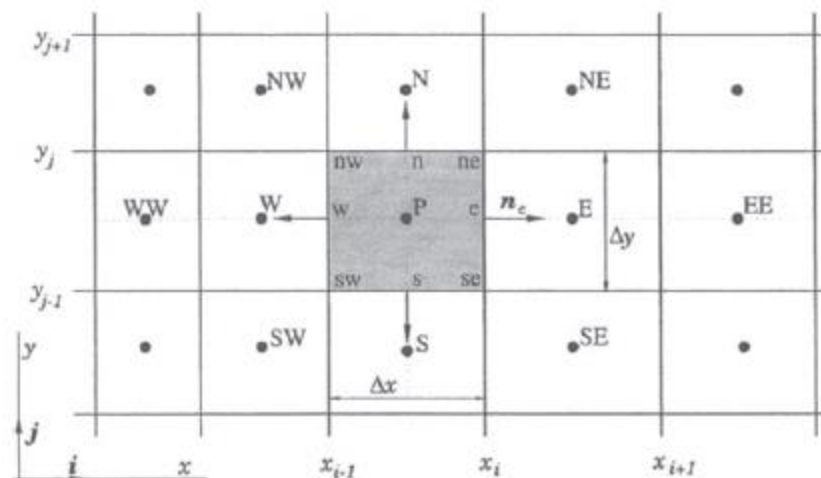


Figure 1.1: A typical control volume and the notation used for a Cartesian 2-D grid [1]

Approximation of physical problems by numerical methods was first introduced by Richardson in 1910 [3]. In his derivation, he applied truncated Taylor Series expansions relating a central point to points around it. This application of Taylor Series expansion to finitely spaced nodes in space was later called Taylor Series formulation. In the 1980's, the representation of physical systems by use of control volumes became popular for modelling heat transfer systems [4,5]. This approach required the application of conservation of energy to every individual control volume. When face heat fluxes of these control volumes were included, Taylor Series approximations were used [5]. This approach was therefore conservative since “the discrete equations maintained an accurate accounting of the energy flows through the domain by ensuring that the approximation for these surface fluxes was unique and independent of the side from which the surface was viewed. [5]” In other words, the control volume or any of its nodal terms did not need their directions defined since the face heat fluxes were always outward normal as shown in Figure 1.1.

Objectives of this thesis are to derive a higher-order finite volume method which better accounts for the higher-order terms to as high of an order of accuracy while maintaining stability. Following this, a series of tests of the numerical method are to be presented to show that the solution converges to the analytical form and that errors decay. In addition, convergence tests are performed to show the order of the method's convergence. Last, future work on the method is proposed to improve its function and applicability to other physical problems, such as for thermal heat transfer and fluid flow.

For the development of the finite volume method in the subsequent chapter, several thermodynamic principles are utilized. The first of these is conservation of energy in the form of the heat diffusion equation, which describes the temperature distribution through a medium.

Alongside this, the relation of the heat rate to the temperature gradient, Fourier's Law, is applied. Furthermore, the thermal contact resistance concept for an infinitesimally thick surface is theoretically defined by a discontinuity in temperature from one side of a surface to the other, while keeping the heat flux equivalent through the surface (conservation of energy) [2].

It should be noted that the methods used in this thesis have been used previously by others in deriving numerical methods, however to the author's knowledge they have not been arranged in quite this combination in previous research.

Chapter 2

Development of the Finite Volume Method

Beginning with 1-D conduction through a solid in Cartesian coordinates, a volume is represented by a rectangular prism (or rectangle in 2-D). The faces associated with the given volume are coincident with the sides of the volume and also coincident with the adjacent volume's respective face, also called the given face's opposing or adjacent face. These geometric pieces are distinct to one another and are thus drawn distinctly, as shown in Figure 2.1. A third kind of geometry is here called a sector, which is distinguished from a face due to the fact that it is not associated with a volume, but rather it defines a boundary condition. A collection of volumes and their associated faces is called a body, and a collection of sectors is called a surface, which envelopes a portion or the whole of the outside of the body.

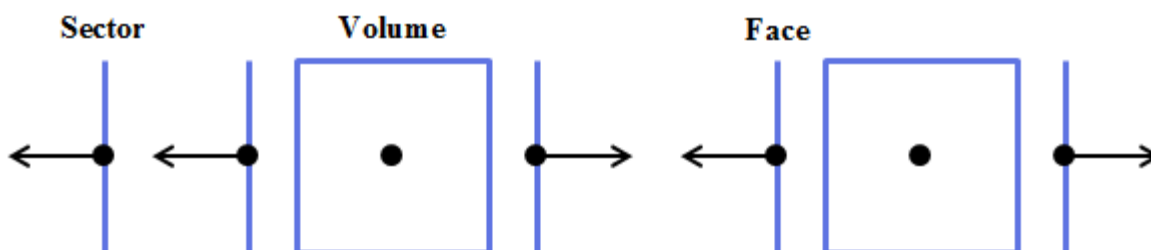


Figure 2.1: Pictorial representation of geometric entities and their associated scalar and vector unknown variables in 1-D

An advantage to “faceting” the faces, that is to split the faces between one volume and the volume next to it, is that the vectors defined at the faces can always be treated as outward normal. Faces can also be paired with sectors; however, the sector vectors do not oppose the face vectors, but are in the same direction. Another advantage of faceting the faces is that two

coincident faces can exist at different state points, such as temperature. This allows for accounting of thermal resistance, which in reality occurs through a region of irregular structure, but in theory is represented here by an area of no thickness, mostly because the volumes contain a substance assumed in continua form. This definition of the faces is also advantageous in placing, for example, a copper volume next to an aluminum volume, or placing a solid volume next to a fluid volume, where the solid face is conductive in nature, whereas the fluid face is convective in nature. This can also be extended to radiative surfaces. Across the coincident faces the fundamental law of conservation of energy is easily applied as the heat flux through one face is equal and opposite to the heat flux through the adjacent face. In addition, allowing a volume to have its own faces associated with it, allows volume equations to interact only with its given faces, making the defining of its behavior easier. In essence, the volume-face interaction is compact and could be defined as a unit. This additionally allows for easy extension to 2-D and 3-D. In other words, a volume “sees” only its respective faces, and a face “sees” only its adjacent face/sector and its own volume. The face in some cases, specific to volumes that are symmetric, may be set up to “see” the opposite face of the volume. Sectors “see” only faces.

Scalar unknown variables are defined at a point, for pictorial representation, but for purposes of numerical discretization actually cover the entire area or volume of the given geometry, as shown in Figure 2.1. The outward vectors of the faces or sectors are in reality fluxes over the area.

Chapter 2.1

First-Order Discretization Scheme for 1-D Conduction through a Solid

The numerical discretization scheme for 1-D conduction through an incompressible solid is developed in the time domain such that it is first-order backward Euler and the system of equations is linear, thus being able to be solved by a matrix inversion. The advantage of this definition is that the scheme is unconditionally stable. Despite unknown variables initially defined at time $t + 1$, no unknown variable within the system matrix is of this form. Rather, all unknown variables are treated as the change in the variable from time t to time $t + 1$ and written with a delta in front, as shown.

$$\Delta T = T^{t+1} - T^t \quad (2.1)$$

An advantage of defining unknown variables in this way is that it makes the central matrix more diagonally dominant resulting in faster computation [6]. This practice will be repeated throughout all successive development of discretization schemes.

When it comes to 1-D solid conduction, unknown variables are defined on the geometry as shown in Figure 2.2. Unlike common CFD software that tend to define variables and properties at the center of volumes, this method distinguishes faces and allows for definement of variables and properties on the faces. This is a benefit, as mentioned above, for representing

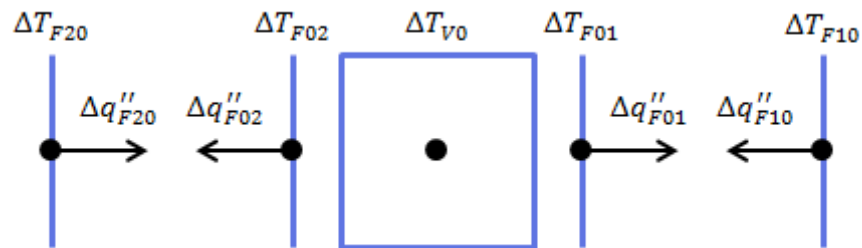


Figure 2.2: Assignment of unknown variables for first-order accurate discretization scheme for 1-D conduction through a solid

surface conditions and behavior more accurately. The volume-faces unit is therefore essentially a 7 point stencil in 3-D.

Development of the system of equations to describe the body includes usage of the heat equation, Fourier's Law, thermal resistance concept, and conservation of energy, where,

$$\sum q_{in} - \sum q_{out} + Vq'''_{gen} = \rho Vc_p \frac{dT}{dt} \quad (2.2)$$

$$q'' = -k \frac{dT}{dx} \quad (2.3)$$

$$r_{cont} q'' = \Delta T \quad (2.4)$$

$$q''_{cond} = q''_{cond} \quad (2.5)$$

Taking equation (2.2) and writing it for first-order backward differencing, we get,

$$-A_{F01}q''^{t+1}_{F01} - A_{F02}q''^{t+1}_{F02} + V_{V0}q'''^{t+1}_{V0} = \rho_{V0}V_{V0}c_{p,V0}^{t+1}(\rho_{V0}, T_{V0}^{t+1}) \frac{T_{V0}^{t+1} - T_{V0}^t}{\Delta t} \quad (2.6)$$

where,

$$q''^{t+1} = q''^t + \Delta q'' \quad (2.7)$$

$$T^{t+1} = T^t + \Delta T \quad (2.8)$$

$$c_{p,V0}^{t+1}(\rho_{V0}, T_{V0}^{t+1}) = c_{p,V0}^t(\rho_{V0}, T_{V0}^t) + \left. \frac{dc_p}{dT} \right|_{V0}^t \Delta T_{V0} \quad (2.9)$$

Plugging these into equation (2.6) and simplifying while neglecting delta-squared terms, we get,

$$\begin{aligned} & [\rho_{V0}V_{V0}c_{p,V0}^t(\rho_{V0}, T_{V0}^t)]\Delta T_{V0} + [A_{F01}\Delta t]\Delta q''_{F01} + [A_{F02}\Delta t]\Delta q''_{F02} = \\ & = -A_{F01}\Delta t q''^t_{F01} - A_{F02}\Delta t q''^t_{F02} + V_{V0}\Delta t q'''^{t+1}_{V0} \end{aligned} \quad (2.10)$$

A similar derivation can be applied to Fourier's Law such that (2.3) becomes,

$$\begin{aligned} & \left[\frac{k_{V0}^t(\rho_{V0}, T_{V0}^t)}{L_c} \right] \Delta T_{F01} + \left[-\frac{k_{V0}^t(\rho_{V0}, T_{V0}^t)}{L_c} \right] \Delta T_{V0} + \Delta q''_{F01} = \\ & = \left[\frac{k_{V0}^t(\rho_{V0}, T_{V0}^t)}{L_c} \right] (T_{V0}^t - T_{F01}^t) - q''^t_{F01} \end{aligned} \quad (2.11)$$

where L_c is a heat transfer or conduction length, which is the normal distance from the face to the center of the volume. Equations (2.4) and (2.5) become,

$$\Delta T_{F01} - \Delta T_{F10} - r_{cont,F01} \Delta q''_{F01} = -T_{F01}^t + T_{F10}^t + r_{cont,F01} q''_{F01}^t \quad (2.12)$$

$$\Delta q''_{F01} + \Delta q''_{F10} = -q''_{F01}^t - q''_{F10}^t \quad (2.13)$$

Equations (2.12) and (2.13) are used in “linking” one face to its opposing face. This process requires that a face-face interaction have an even number of unknown variables, such that half of the variables are referenced by one face from the other and vice-versa. Equation (2.10) would be applied to every volume, and equation (2.11) would be applied to every face. In equation (2.11) the thermal conductivity could be defined as the average of the volume-centered value and the face value, making the solution more accurate.

Chapter 2.2

Higher-Order Discretization Scheme for n-D Conduction through a Solid

An idea applied here is the treatment of differential terms of the fundamental differential equations as unknown variables, which in turn makes the differential equations purely algebraic.

These differential terms are then specifically defined at certain points on the geometry and appropriately related using several Taylor series expansions. Since the face relations are already defined in equations (2.12) and (2.13), the focus here is the interaction between the volume and its associated faces, as shown in Figure 2.3. Since the

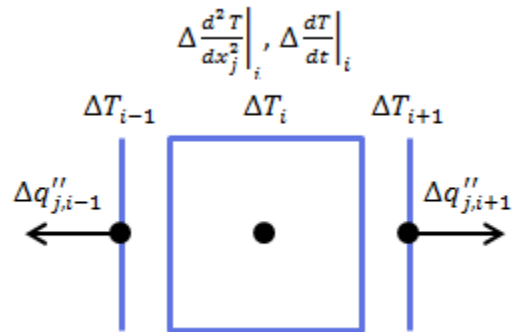


Figure 2.3: Assignment of unknown variables for a higher-order accurate discretization scheme for 1-D conduction through a solid

heat fluxes on the faces are both outward normal, it turns out that the direction of the second derivative of temperature with respect to x_j does not matter. It can be defined either in the positive or negative x_j -direction. The j -subscript is tensor notation, meaning that this 1-D relation can be re-written for up to a 3-D symmetric rectangular prism volume.

The fundamental equations are as follows,

$$\frac{1}{\alpha} \frac{\partial T}{\partial t} = \nabla^2 T + \frac{q'''_{gen}}{k} \quad (2.14)$$

$$q''_j = -k \frac{dT}{dx_j} \quad (2.15)$$

$$T(x_j + L_c) = T(x_j) + L_c \left. \frac{dT}{dx_j} \right|_{x_j} + \frac{L_c^2}{2!} \left. \frac{d^2T}{dx_j^2} \right|_{x_j} + \frac{L_c^3}{3!} \left. \frac{d^3T}{dx_j^3} \right|_{x_j} + \dots + \frac{L_c^n}{n!} \left. \frac{d^n T}{dx_j^n} \right|_{x_j} \quad (2.16)$$

Equation (2.16) is modified so that position $i + 1$ is related to the differentials of position i and position $i - 1$ is related to the differentials of x_j -position i . Therefore, at some time $t + 1$,

$$\begin{aligned} T_{i+1}^{t+1} = & T_i^{t+1} + L_c \left. \frac{dT}{dx_j} \right|_i^{t+1} + \frac{L_c^2}{2} \left. \frac{d^2T}{dx_j^2} \right|_i^{t+1} + \frac{L_c^3}{6} \left. \frac{d^3T}{dx_j^3} \right|_i^{t+1} + \frac{L_c^4}{24} \left. \frac{d^4T}{dx_j^4} \right|_i^{t+1} + \\ & + \frac{L_c^5}{120} \left. \frac{d^5T}{dx_j^5} \right|_i^{t+1} + \frac{L_c^6}{720} \left. \frac{d^6T}{dx_j^6} \right|_i^{t+1} + HOTS \end{aligned} \quad (2.17)$$

$$\begin{aligned} T_{i-1}^{t+1} = & T_i^{t+1} - L_c \left. \frac{dT}{dx_j} \right|_i^{t+1} + \frac{L_c^2}{2} \left. \frac{d^2T}{dx_j^2} \right|_i^{t+1} - \frac{L_c^3}{6} \left. \frac{d^3T}{dx_j^3} \right|_i^{t+1} + \frac{L_c^4}{24} \left. \frac{d^4T}{dx_j^4} \right|_i^{t+1} + \\ & - \frac{L_c^5}{120} \left. \frac{d^5T}{dx_j^5} \right|_i^{t+1} + \frac{L_c^6}{720} \left. \frac{d^6T}{dx_j^6} \right|_i^{t+1} + HOTS \end{aligned} \quad (2.18)$$

where *HOTS* means “higher-order terms”. Taking the derivative with respect to x_j of equations

(2.17) and (2.18) and multiplying by L_c we get,

$$\begin{aligned}
L_c \frac{dT}{dx_j} \Big|_{i+1}^{t+1} &= L_c \frac{dT}{dx_j} \Big|_i^{t+1} + L_c^2 \frac{d^2T}{dx_j^2} \Big|_i^{t+1} + \frac{L_c^3}{2} \frac{d^3T}{dx_j^3} \Big|_i^{t+1} + \frac{L_c^4}{6} \frac{d^4T}{dx_j^4} \Big|_i^{t+1} + \\
&+ \frac{L_c^5}{24} \frac{d^5T}{dx_j^5} \Big|_i^{t+1} + \frac{L_c^6}{120} \frac{d^6T}{dx_j^6} \Big|_i^{t+1} + HOTS
\end{aligned} \tag{2.19}$$

$$\begin{aligned}
L_c \frac{dT}{dx_j} \Big|_{i-1}^{t+1} &= L_c \frac{dT}{dx_j} \Big|_i^{t+1} - L_c^2 \frac{d^2T}{dx_j^2} \Big|_i^{t+1} + \frac{L_c^3}{2} \frac{d^3T}{dx_j^3} \Big|_i^{t+1} - \frac{L_c^4}{6} \frac{d^4T}{dx_j^4} \Big|_i^{t+1} + \\
&+ \frac{L_c^5}{24} \frac{d^5T}{dx_j^5} \Big|_i^{t+1} - \frac{L_c^6}{120} \frac{d^6T}{dx_j^6} \Big|_i^{t+1} + HOTS
\end{aligned} \tag{2.20}$$

Equation (2.15) can be re-written as,

$$\frac{dT}{dx_j} \Big|_{i+1}^{t+1} = -\frac{1}{k} q''_{j,i+1}{}^{t+1} \tag{2.21}$$

$$\frac{dT}{dx_j} \Big|_{i-1}^{t+1} = \frac{1}{k} q''_{j,i-1}{}^{t+1} \tag{2.22}$$

Taking equations (2.18), (2.17), (2.20), (2.19), and respectively multiplying them by A, B, C, and D, while substituting in equations (2.21) and (2.22), and adding them all up, we get,

$$\begin{aligned}
&(-A - B)T_i^{t+1} + \left(-\frac{A}{2} - \frac{B}{2} + C + D\right) L_c^2 \frac{d^2T}{dx_j^2} \Big|_i^{t+1} + \\
&+ (A)T_{i-1}^{t+1} + (B)T_{i+1}^{t+1} + (C) \frac{L_c}{k} q''_{j,i-1}{}^{t+1} + (D) \frac{L_c}{k} q''_{j,i+1}{}^{t+1} = \\
&= (-A + B + C - D) L_c \frac{dT}{dx_j} \Big|_i^{t+1} + \left(-\frac{A}{6} + \frac{B}{6} + \frac{C}{2} - \frac{D}{2}\right) L_c^3 \frac{d^3T}{dx_j^3} \Big|_i^{t+1} + \\
&+ \left(\frac{A}{24} + \frac{B}{24} - \frac{C}{6} - \frac{D}{6}\right) L_c^4 \frac{d^4T}{dx_j^4} \Big|_i^{t+1} + \left(-\frac{A}{120} + \frac{B}{120} + \frac{C}{24} - \frac{D}{24}\right) L_c^5 \frac{d^5T}{dx_j^5} \Big|_i^{t+1} + \\
&+ \left(\frac{A}{720} + \frac{B}{720} - \frac{C}{120} - \frac{D}{120}\right) L_c^6 \frac{d^6T}{dx_j^6} \Big|_i^{t+1} + HOTS
\end{aligned} \tag{2.23}$$

Equation (2.23) is used to get two equations and may be written in several forms of varying orders of accuracy. This is shown in Appendix A. The equations are written so that C and/or D are/is equal to zero as a means of comparison due to the fact that the heat flux terms on the faces are the highest-order term that “communicate” with the adjacent face or unit. Equations are selected from Appendix A starting with the highest-order accurate equation and continuing down in accuracy while noting that the equations selected must be independent of one another. Equations (A.2) and (A.4) are not independent in reference to equation (A.1). Therefore, the equations selected are (A.1) and (A.3) which are sixth-order and fourth-order accurate in space. These are re-written below while neglecting higher-order terms.

$$\begin{aligned}
& (-8)\Delta T_i + (-2L_c^2)\Delta \left. \frac{d^2 T}{dx_j^2} \right|_i + (4)\Delta T_{i-1} + (4)\Delta T_{i+1} + \left(\frac{L_c}{k}\right)\Delta q''_{i-1} + \left(\frac{L_c}{k}\right)\Delta q''_{i+1} = \\
& = (8)T_i^t + (2L_c^2)\left. \frac{d^2 T}{dx_j^2} \right|_i^t + (-4)T_{i-1}^t + (-4)T_{i+1}^t + \left(-\frac{L_c}{k}\right)q''_{i-1}^t + \left(-\frac{L_c}{k}\right)q''_{i+1}^t + \\
& + \vartheta(L_c^6) \tag{2.24}
\end{aligned}$$

$$\begin{aligned}
& (-4)\Delta T_i + (2)\Delta T_{i-1} + (2)\Delta T_{i+1} + \left(\frac{L_c}{k}\right)\Delta q''_{i-1} + \left(\frac{L_c}{k}\right)\Delta q''_{i+1} = \\
& = (4)T_i^t + (-2)T_{i-1}^t + (-2)T_{i+1}^t + \left(-\frac{L_c}{k}\right)q''_{i-1}^t + \left(-\frac{L_c}{k}\right)q''_{i+1}^t + \vartheta(L_c^4) \tag{2.25}
\end{aligned}$$

Equation (2.24) is used to define the second derivative of temperature with respect to x_j while equation (2.25) may be used to define either a face temperature or a face heat flux. One more equation remains to be derived for the other opposite face temperature or face heat flux since equations (2.12) and (2.13) “link” unknown variables from an opposing face in the positive and negative x_j -directions, each respectively. One could choose one of the remaining third-order accurate equations in Appendix A, equations (A.5) to (A.10), but these would result in a system

of equations that is somewhat unstable or non-physical. Instead, a different equation is derived next that does not yield this issue.

Taking equation (2.18) and re-writing it so that i is now $i + 1$, and $i - 1$ is now i , we get,

$$T_i^{t+1} = T_{i+1}^{t+1} - L_c \left. \frac{dT}{dx_j} \right|_{i+1}^{t+1} + \frac{L_c^2}{2} \left. \frac{d^2T}{dx_j^2} \right|_{i+1}^{t+1} - \frac{L_c^3}{6} \left. \frac{d^3T}{dx_j^3} \right|_{i+1}^{t+1} + HOTS \quad (2.26)$$

Then, substituting in equation (2.21) and re-arranging, we get,

$$(-1)\Delta T_i + \Delta T_{i+1} + \left(\frac{L_c}{k}\right) \Delta q''_{j,i+1} = T_i^t + (-1)T_{i+1}^t + \left(-\frac{L_c}{k}\right) q''_{j,i+1} + \vartheta(L_c^2) \quad (2.27)$$

Equation (2.27) is second-order accurate in space and is used as the other equation that may describe either the temperature unknown variable of a face or the opposite face's heat flux, as mentioned above.

A Taylor series expansion for time differencing within the volume is also applied as shown in Appendix B. The result is,

$$(2)\Delta T_i + (-\Delta t)\Delta \left. \frac{dT}{dt} \right|_i = (2\Delta t) \left. \frac{dT}{dt} \right|_i^t + \vartheta(\Delta t^3) \quad (2.28)$$

Using the heat equation, equation (2.14), and substituting equation (2.1), the following is used for the first derivative of temperature with respect to time in a volume.

$$(-1)\Delta \left. \frac{\partial^2 T}{(\partial x_j)^2} \right|_i + \left(\frac{1}{\alpha}\right) \Delta \left. \frac{\partial T}{\partial t} \right|_i = \left. \frac{\partial^2 T}{(\partial x_j)^2} \right|_i^t + \left(-\frac{1}{\alpha}\right) \left. \frac{\partial T}{\partial t} \right|_i^t + \left(\frac{q'''_{gen}}{k}\right) \quad (2.29)$$

Note that the second-order spatial differential is a summation in accordance with tensor notation rules. For the higher-order method, when coding, equations (2.28), (2.24), (2.29), (2.12), (2.25),

(2.27), and (2.13) are applied to unknown variables, ΔT_i , $\Delta \left. \frac{d^2 T}{dx_j^2} \right|_i$, $\Delta \left. \frac{dT}{dt} \right|_i$, ΔT_{i+1} , $\Delta q''_{j,i+1}$, ΔT_{i-1} ,

and $\Delta q''_{j,i-1}$ respectively. Alternatively, the face equations may be switched side to side, or

equation (2.27) may be switched with equation (2.28). The system of linear equations describing

the cell behavior is ultimately second-order accurate in space and third-order accurate in time with reference to temperature. The interaction of these equations with the second derivative spatial term and the first-order differential temporal term of the heat equation is more accurate than the first-order method described in Chapter 2.1. These equations also describe the 2-D or 3-D system as long as the rectangular prism volume remains symmetric in each x_j -direction.

Chapter 3

Proof of Concept

This chapter details a series of examples meant to test the higher-order numerical method with several closed form solutions of both steady state and transient conduction through solid rods.

Chapter 3.1

Steady State 1-D Conduction through a Solid Rod

The first case of comparison is 1-D solid heat generation at steady state with fixed temperatures of $T_o = 0 \text{ }^\circ\text{F}$ ($-17.8 \text{ }^\circ\text{C}$) at either end of a 16 ft (4.877 m) long copper rod with 1 ft (0.305 m) by 1 ft (0.305 m) cross-section. The rod is divided into four volumes down the rod's length thus making each volume's volume 4 ft^3 (0.113 m^3) and each face's area 1 ft^2 (0.093 m^2). Taking the heat equation (2.14) and setting it equal to zero, a closed form solution is derived.

$$T(x) = \frac{q'''_{gen}}{2k}(-x^2 + Lx) + T_o \quad (3.1)$$

where L is the length of the rod. Taking the thermal conductivity for copper at $50 \text{ }^\circ\text{F}$ ($10.0 \text{ }^\circ\text{C}$), $k = 233.8 \frac{\text{Btu}}{\text{hr ft }^\circ\text{F}}$ ($404.6 \frac{\text{W}}{\text{m K}}$), and for visual purposes requesting the maximum temperature be $100 \text{ }^\circ\text{F}$ ($37.8 \text{ }^\circ\text{C}$), therefore, the heat generation of the rod is $730.6 \frac{\text{Btu}}{\text{hr ft}^3}$ ($7,562 \frac{\text{W}}{\text{m}^3}$). Starting the rod at $0 \text{ }^\circ\text{F}$ ($-17.8 \text{ }^\circ\text{C}$) and stepping in time by 5 min for 5 days yields the following figure.

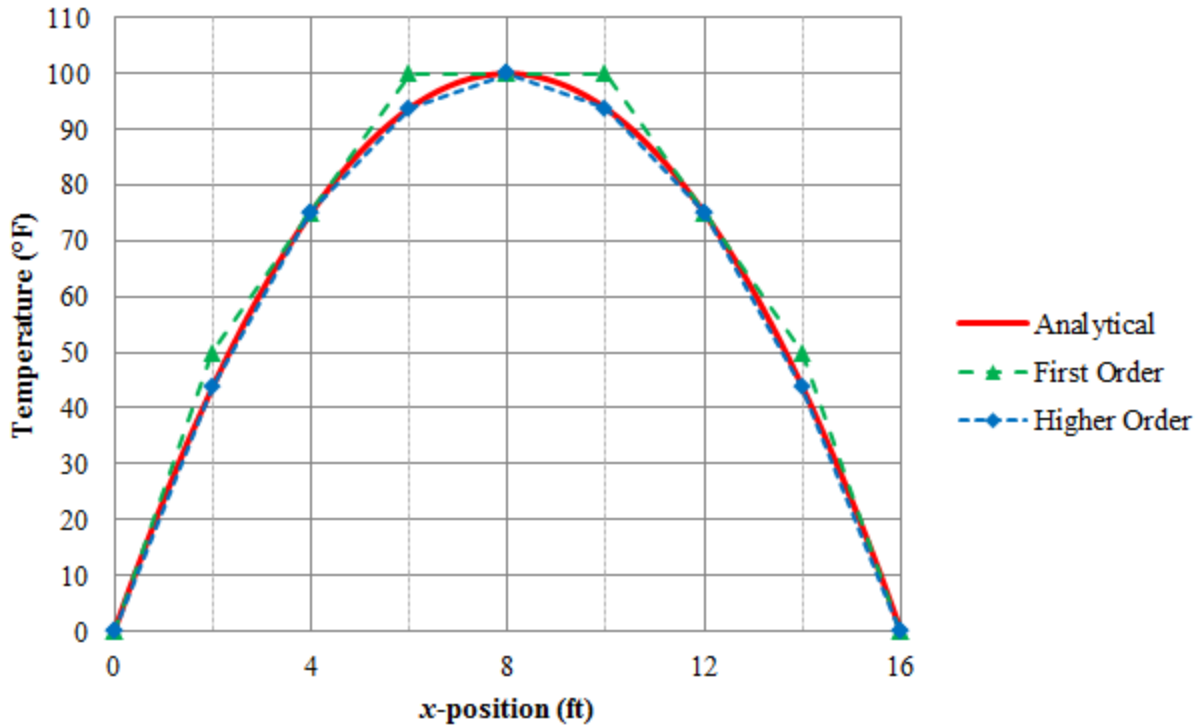


Figure 3.1: 1-D solid heat generation at steady state comparison between first-order and higher-order solutions

As shown in Figure 3.1, a limitation of the first-order numerical method, is that the gradient of temperature in the x_j -direction must be identical between a face and its opposing face, thus making the slopes the same. On the other hand, the higher order nature of the higher-order method allows the interaction between temperature and its first and second derivatives in space to be more correct even though the first derivative, or heat flux, on two opposing faces is the same. This in turn, allows the higher-order method to be more accurate at predicting the volume-centered temperatures.

Taking the same problem, but changing the number of volumes across the length of the rod to 2, 4, 8, 16, 32, and 64 we can perform a convergence study where the order of convergence, p , and the leading-order truncation error, ϵ_h^d , are approximated by,

$$p \approx \frac{1}{\log|2|} \log \left| \frac{\Phi_{2h} - \Phi_{4h}}{\Phi_h - \Phi_{2h}} \right| \quad (3.2)$$

$$\epsilon_h^d \approx \frac{\Phi_h - \Phi_{2h}}{2^{p-1}} \quad (3.3)$$

where Φ would be equivalent to T , h equivalent to L_c , and p related to the order of accuracy for a converging solution. Table 3.1 shows that as the grid is refined the solution approaches second-order convergence, which is identical to the spatial order of accuracy.

Table 3.1: Convergence study with refinement of grid for 1-D solid heat generation

Number of Volumes	L_c (ft)	T (°F)	p	ϵ_h^d (°F)
2	4.000	99.9998620596396	N/A	N/A
4	2.000	99.9998620480975	N/A	N/A
8	1.000	99.9998620367619	0.026	$6.22E-07$
16	0.500	99.9998620325121	1.415	$2.55E-09$
32	0.250	99.9998620313297	1.846	$4.56E-10$
64	0.125	99.9998620310261	1.961	$1.05E-10$

The higher-order method, as well as the first-order method, are both able to handle discontinuities with ease. These may occur either as an instant temperature jump across a face due to contact resistance as shown in Figure 3.2 or may occur in the case of a boundary condition instantaneously applied as shown later in Figure 3.3.

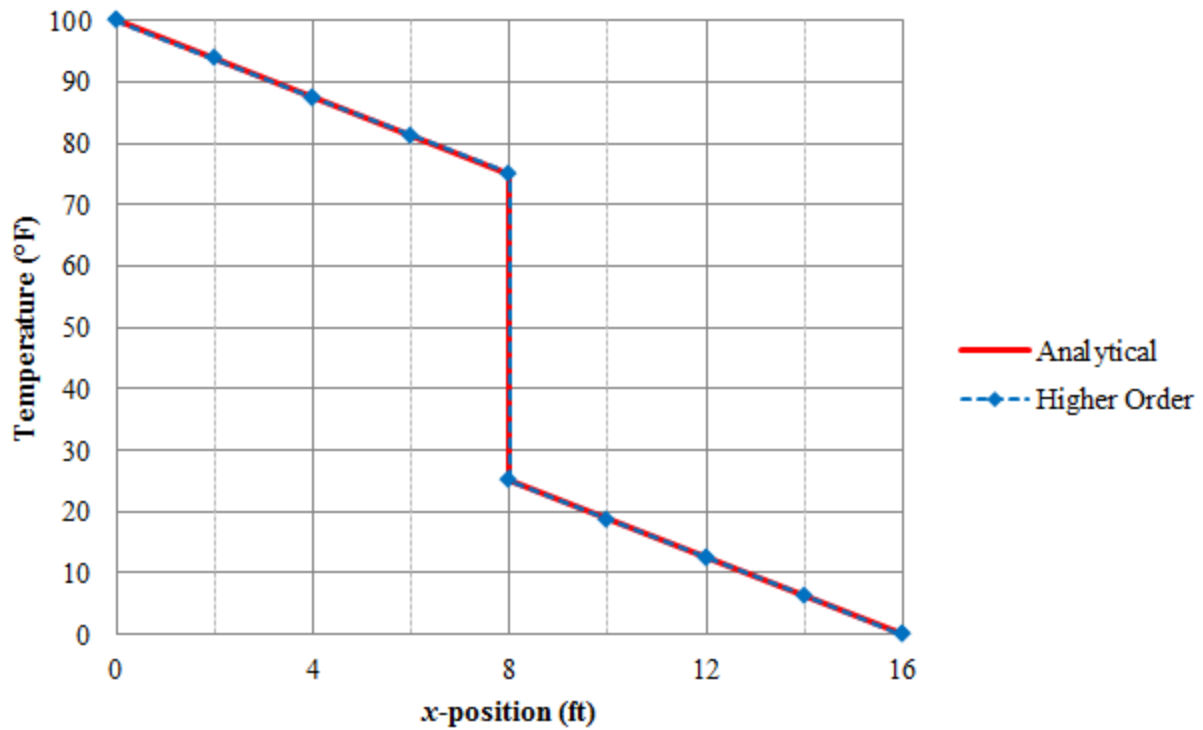


Figure 3.2: 1-D solid conduction with contact resistance at steady state

Chapter 3.2

Transient 1-D Conduction through a Solid Rod

To test the higher-order method in time, we will use the similarity solution for a semi-infinite solid,

$$T(x, t) = (T_i - T_s) \operatorname{erf}\left(\frac{x}{2\sqrt{at}}\right) + T_s \quad (3.4)$$

where T_i is the initial temperature everywhere on the solid, and T_s is the temperature after the sudden step change in temperature at one end of the solid. Variables x and t are nondimensionalized, where,

$$\eta = \frac{x}{2\sqrt{\alpha t}} \quad (3.5)$$

$$F(\eta) = T(x, t) - T_i = (T_s - T_i)[1 - \text{erf}(\eta)] \quad (3.6)$$

$$t^* = \frac{t}{\Delta t} \quad (3.7)$$

Taking the 16 ft (4.877 m) rod from earlier with 16 volumes, and specifying $\Delta t = 1$ min, $T_i = 0$ °F (-17.8 °C), and $T_s = 100$ °F (37.8 °C), the following solution is determined using equations (3.5) to (3.7).

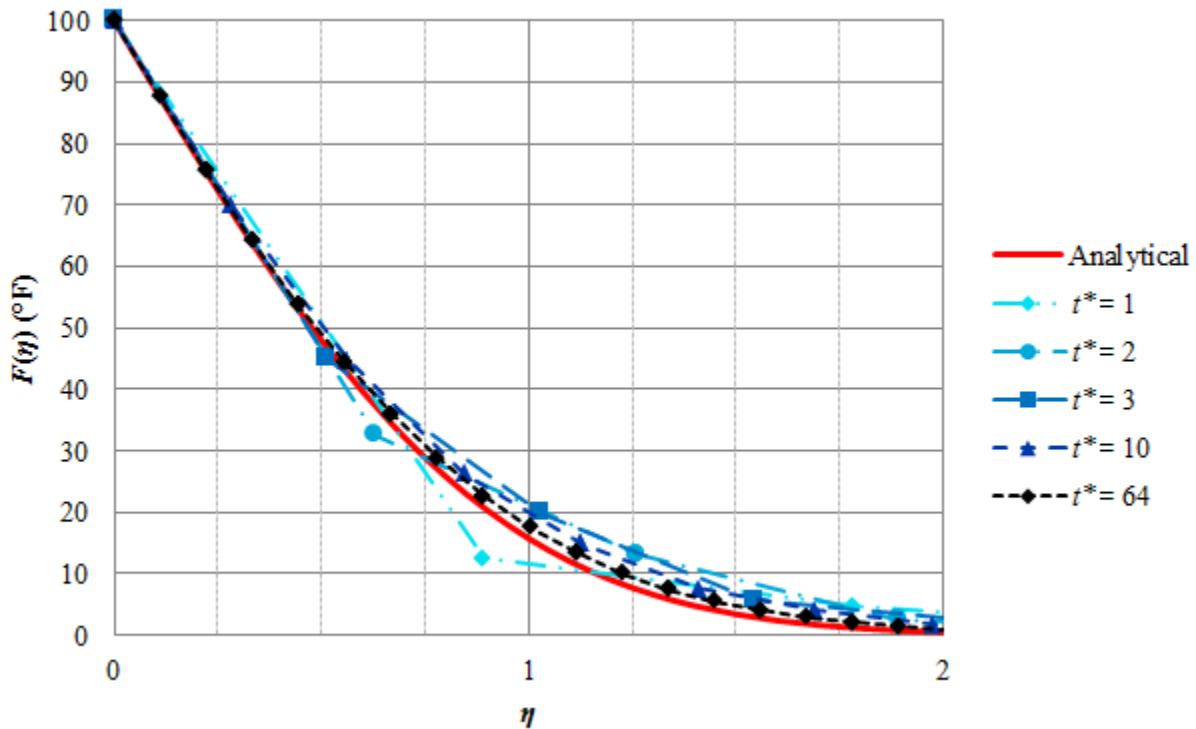


Figure 3.3: 1-D transient semi-infinite solid conduction with fixed temperature boundary condition

Looking at Figure 3.3, it is observed that in the first time step, $t^* = 1$, the solution has no problem in instantaneously jumping from T_i to T_s . The solution continues to converge towards the closed form semi-infinite solid solution as seen by the curves that darken as time increases. This will continue until the rod is no longer “semi-infinite” since at the far end of the rod the

temperature is fixed. At $t^* = 64$, the rod is still sufficient in length that it appears as “semi-infinite”.

The order of convergence can also be found by holding the total time constant at 64 min, and setting the number of time steps to 1, 2, 4, 8, 16, 32, and 64 each respectively while recording the temperature value at $x = 4.5$ ft (1.372 m) or $\eta = 1.003$. The curve where $\Delta t = 1$ min is plotted in Figure 3.3 and labeled as $t^* = 64$.

Table 3.2: Convergence study in time domain for 1-D transient semi-infinite solid conduction

Number of Time Steps	Δt (hr)	T (°F)	p	ϵ_h^d (°F)
1	1.067	7.10421814500833	N/A	N/A
2	0.533	11.3144218783650	N/A	N/A
4	0.267	15.0503968537474	0.172	29.432
8	0.133	16.4178307375104	1.450	0.789
16	0.067	17.0702760586613	1.068	0.595
32	0.033	17.3875978413453	1.040	0.300
64	0.017	17.5441911173689	1.019	0.153

According to Table 3.2, the solution converges at a first-order rate, which is different from equation (2.28) which predicts third-order accuracy. This is unexpected since the system of equations includes terms for the first derivative of temperature with respect to time, so it would be thought that the order of accuracy and order of convergence must be higher than this. It is also observed in Table 3.2 that the temperature overshoots the semi-infinite similarity solution at $\eta = 1.003$ which is 15.586 °F (−9.119 °C). However, this can be resolved as shown in Figure 3.3 by improving the spatial grid resolution. In other words, a special relationship between temporal and spatial discretization probably exists that best predicts the solution following a time step. This will be briefly explored in Chapter 3.3.

To double check the time-convergence of the solution, we develop another similarity problem. Like the previous example, this one is also pure conductive diffusion of heat. Neglecting heat generation, an easy solution to the heat equation (2.14) would be one that is sinusoidal in space and exponential in time, where,

$$T(x, t) = (T_s - T_o)e^{-\frac{\pi^2 \alpha}{L^2} t} \sin\left(\frac{\pi x}{L}\right) + T_o \quad (3.8)$$

Nondimensionalizing in time with equation (3.7), we get,

$$T(x, t^*) = (T_s - T_o)e^{-\frac{\pi^2 \alpha \Delta t}{L^2} t^*} \sin\left(\frac{\pi x}{L}\right) + T_o \quad (3.9)$$

Setting up the problem with $T_s = 100$ °F (37.8 °C), $T_o = 0$ °F (-17.8 °C), $L = 16$ ft (4.877 m), and $\Delta t = 1$ hr with 4 volumes, therefore,

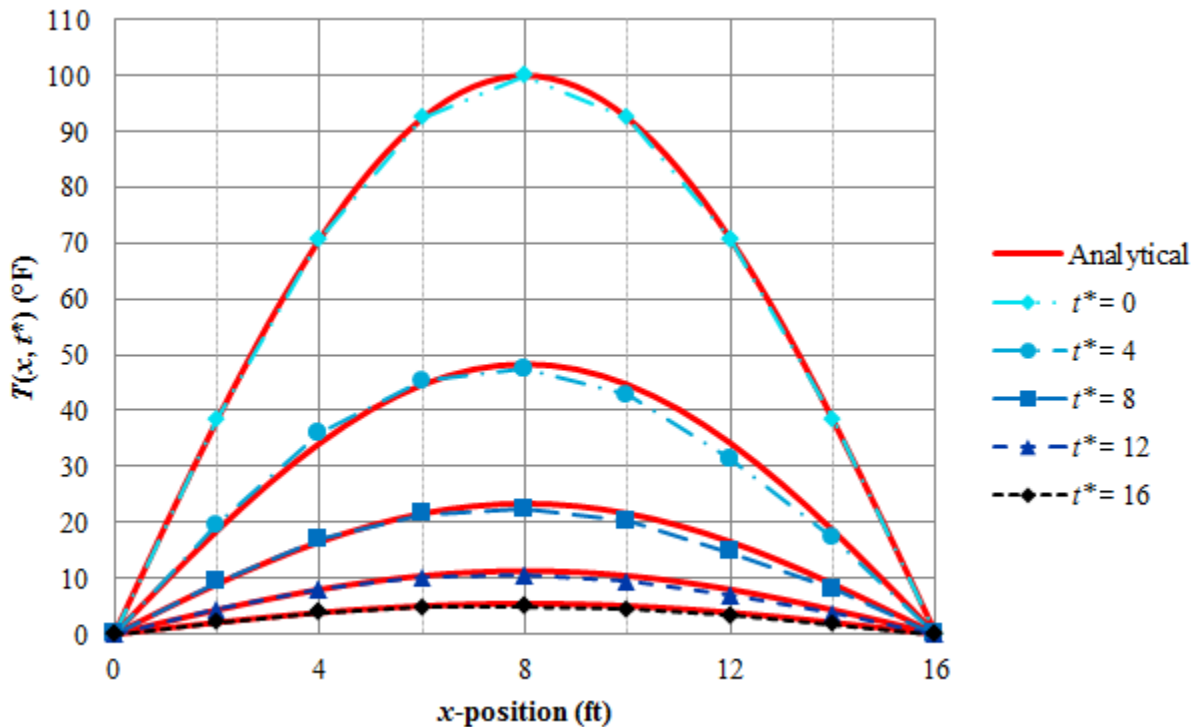


Figure 3.4: 1-D transient solid conductive diffusion of sinusoidal temperature profile

At first, the solution shifts away from the analytical, but then as time continues it converges towards the solution as shown in Figure 3.4. At the centerline, the error is greatest at approximately $t^* = 8$ at about 1.0 °F (0.56 °C). By $t^* = 16$, the error has decayed to approximately 0.5 °F (0.28 °C). It is also observed in Figure 3.4 that the data points bend slightly to one side. This is probably because the equations that define both face temperatures and fluxes are different side to side and repeat consistently in space. The error in the temporal and spatial differencing can be improved by refining the spatial grid and the time step.

Once again, a convergence study in time is performed, taking the duration of time as 16 hr, and the position of interest as $x = L/2 = 8$ ft (2.438 m), where the temperature is theoretically 5.462 °F (−14.743 °C). Note that the numerical solution to this scenario with 16 temporal divisions is plotted as $t^* = 16$ in Figure 3.4.

Table 3.3: Convergence study in time domain for 1-D transient solid conductive diffusion of sinusoidal temperature profile at $x = L/2$.

Number of Time Steps	Δt (hr)	T (°F)	p	ϵ_h^d (°F)
1	16.000	−18.2121333630638	N/A	N/A
2	8.000	2.24688391278668	N/A	N/A
4	4.000	4.37177187468143	3.267	0.246
8	2.000	4.85699022110784	2.131	0.144
16	1.000	4.97077933295543	2.092	0.035
32	0.500	4.99548523694714	2.203	0.007
64	0.250	4.99983833722764	2.505	0.001

As shown in Table 3.3, the order of convergence for this example is somewhere between two and three. This is likely because there is a mismatch in the resolution of the spatial and temporal differencing, or one of these being too coarse. It is interesting here that as the solution converges, a slight oscillation develops, meaning that the solution overshoots its target

repeatedly. This could be fixed by adding an under-relaxation factor, which would potentially lessen the order of convergence, or by better matching the time discretization with the spatial grid. This would require experimentation with the numerical method to determine for a given problem. This will be demonstrated with a different example in the following sub-Chapter.

Chapter 3.3

Convergence Test in both the Spatial and Temporal Domains for 1-D Conduction through a Solid Rod with Heat Generation

Taking the first example for 1-D solid heat generation at steady state with fixed end temperatures, we run the solution for 64 hr instead while varying the number of spatial steps, then the number of temporal steps, then both spatial and temporal at the same time.

Table 3.4: Spatial convergence study for 1-D heat generation with fixed end temperatures with coarse temporal step

Number of Volumes/ Time Steps	L_c (ft)	Δt (hr)	T (°F)	p	ϵ_h^d (°F)
2/64	4.000	1.000	99.9995279468806	N/A	N/A
4/64	2.000	1.000	99.9992464287652	N/A	N/A
8/64	1.000	1.000	99.9990560450706	0.564	3.98E-04
16/64	0.500	1.000	99.9990127336163	2.136	1.28E-05
32/64	0.250	1.000	99.9989775691379	0.301	1.52E-04
64/64	0.125	1.000	99.9989739126895	3.266	4.24E-07

As seen in Table 3.4, an oscillation develops because the temporal step is too coarse. The solution converges with more stability by refining the temporal step as seen in Table 3.5. As Ferziger and Perić suggest, “when solving complicated, non-linear and coupled equations with complicated boundary conditions, there are few stability results so we may have to rely on

experience and intuition. Many solution schemes require that the time step be smaller than a certain limit or that under-relaxation be used. [1]” It is shown here, in Tables 3.5 to 3.7, that when the spatial and temporal discretizations are sized appropriately that the solution converges at second-order rate in space and time.

Table 3.5: Spatial convergence study for 1-D heat generation with fixed end temperatures

Number of Volumes/ Time Steps	L_c (ft)	Δt (hr)	T (°F)	p	ϵ_h^d (°F)
2/512	4.000	0.125	99.9995143119774	N/A	N/A
4/512	2.000	0.125	99.9992246274425	N/A	N/A
8/512	1.000	0.125	99.9990294295439	0.570	4.03E-04
16/512	0.500	0.125	99.9989664334672	1.632	3.00E-05
32/512	0.250	0.125	99.9989496255079	1.906	6.12E-06
64/512	0.125	0.125	99.9989453542530	1.976	1.46E-06

Table 3.6: Temporal convergence study for 1-D heat generation with fixed end temperatures

Number of Volumes/ Time Steps	L_c (ft)	Δt (hr)	T (°F)	p	ϵ_h^d (°F)
64/8	0.125	8.000	100.106036890825	N/A	N/A
64/16	0.125	4.000	99.9991109805812	N/A	N/A
64/32	0.125	2.000	99.9990194433976	10.190	7.84E-08
64/64	0.125	1.000	99.9989739126895	1.008	4.51E-05
64/128	0.125	0.500	99.9989522153830	1.069	1.98E-05
64/256	0.125	0.250	99.9989467293190	1.984	1.86E-06
64/512	0.125	0.125	99.9989453542530	1.996	4.60E-07

Table 3.7: Spatial/temporal convergence study for 1-D heat generation with fixed end temperatures

Number of Volumes/ Time Steps	L_c (ft)	Δt (hr)	T (°F)	p	ϵ_h^d (°F)
2/16	4.000	4.000	99.9996916046458	N/A	N/A
4/32	2.000	2.000	99.9993096440695	N/A	N/A
8/64	1.000	1.000	99.9990560450706	0.591	5.01E-04
16/128	0.500	0.500	99.9989731781393	1.614	4.02E-05
32/256	0.250	0.250	99.9989509958643	1.901	8.11E-06
64/512	0.125	0.125	99.9989453542530	1.975	1.92E-06

Chapter 4

Summary and Conclusion

The higher-order numerical finite volume method is second-order accurate in space and third-order accurate in time. It is shown to be accurate in predicting temperature distributions for both steady state and transient solid conductive heat transfer problems. The steady state problems include heat generation for a solid rod, and conduction through a rod with a contact resistance at its center with two different fixed temperatures at either end. The transient problems include diffusion of heat through a “semi-infinite” solid rod with fixed temperature at one end, and heat diffusion of a sinusoidal temperature distribution with identical fixed temperatures at either end of the rod. In space and time the solution converges at second-order when the both domain’s discretizations are fine enough. As expected, the accuracy of the solution can be improved by refining the spatial grid and/or temporal discretization.

Further development of this method would also involve adapting it to the fourth-order accurate Runge-Kutta time marching method. This may be difficult due to the need for the differentials in time of each unknown variable and a method for predicting the next time step’s differentials. One way to do this could be to take the derivative with respect to time of the fundamental equations governing the process. Special care would need to be taken when handling discontinuities in space and time as the derivatives would be infinite. This would improve accuracy and the order of convergence.

Additional improvement on the accuracy in space could be achieved by inserting more nodes in the volume, perhaps halfway between the volume-centered nodes and the face nodes. This would yield a 13 point stencil in 3-D for the volume-faces unit. This would allow for

higher-order terms of the Taylor series expansion to be accounted for. The concept of the compact unit consisting of the volume and its faces would then be easy to duplicate as it “communicates” only with other units via its face’s two unknown variables, temperature and heat flux.

Furthermore, this method’s derivation process could be adapted to other forms of heat transfer such as convection or radiation, as well as fluid flows, such as the incompressible or compressible Navier-Stokes equations for Newtonian fluids for example. Additionally, geometries may be developed that model systems where some finite volumes behave as solid, others as fluid, and so forth.

Appendix A

Multiple Forms of Equation (2.23)

Table A.1: Coefficients of the terms in equation (2.23) for different values of A , B , C , and D .

T_i^{t+1}	$\frac{d^2 T}{L_c dx_j^2} \Big _i^{t+1}$	T_{i-1}^{t+1}	T_{i+1}^{t+1}	$\frac{L_c}{k} q_{j,i-1}^{t+1}$	$\frac{L_c}{k} q_{j,i+1}^{t+1}$	$= \frac{dT}{L_c dx_j} \Big _i^{t+1}$	$\frac{d^3 T}{L_c dx_j^3} \Big _i^{t+1}$	$\frac{d^4 T}{L_c dx_j^4} \Big _i^{t+1}$	$\frac{d^5 T}{L_c dx_j^5} \Big _i^{t+1}$	$\frac{d^6 T}{L_c dx_j^6} \Big _i^{t+1}$	Eqn.
-8	-2	4	4	1	1	0	0	0	0	$-\frac{1}{180}$	(A.1)
-2	-1	1	1	0	0	0	0	$\frac{1}{12}$	0	$\frac{1}{360}$	(A.2)
-4	0	2	2	1	1	0	0	$-\frac{1}{6}$	0	$-\frac{1}{90}$	(A.3)
0	2	0	0	1	1	0	0	$-\frac{1}{3}$	0	$-\frac{1}{60}$	(A.4)
-2	0	$\frac{1}{2}$	$\frac{3}{2}$	0	1	0	$-\frac{1}{3}$	$-\frac{1}{12}$	$-\frac{1}{30}$	$-\frac{1}{180}$	(A.5)
-1	$\frac{1}{2}$	0	1	0	1	0	$-\frac{1}{3}$	$-\frac{1}{8}$	$-\frac{1}{30}$	$-\frac{1}{144}$	(A.6)
0	1	$-\frac{1}{2}$	$\frac{1}{2}$	0	1	0	$-\frac{1}{3}$	$-\frac{1}{6}$	$-\frac{1}{30}$	$-\frac{1}{120}$	(A.7)
1	$\frac{3}{2}$	-1	0	0	1	0	$-\frac{1}{3}$	$-\frac{5}{24}$	$-\frac{1}{30}$	$-\frac{7}{720}$	(A.8)
$\frac{4}{-3}$	0	0	$\frac{4}{3}$	$-\frac{1}{3}$	1	0	$-\frac{4}{9}$	$-\frac{1}{18}$	$-\frac{2}{45}$	$-\frac{1}{270}$	(A.9)
0	0	-1	1	-1	1	0	$-\frac{2}{3}$	0	$-\frac{1}{15}$	0	(A.10)

Appendix B

Derivation of Equation (2.28)

Equation (2.28) is derived by using equation (2.16), but modifying it for temporal discretization rather than spatial, where,

$$T_i^{t+1} = T_i^t + \Delta t \left. \frac{dT}{dt} \right|_i^t + \frac{(\Delta t)^2}{2} \left. \frac{d^2T}{dt^2} \right|_i^t + \frac{(\Delta t)^3}{6} \left. \frac{d^3T}{dt^3} \right|_i^t + HOTS \quad (\text{B.1})$$

Taking the derivative of equation (B.1) with respect to time and multiplying by Δt ,

$$\Delta t \left. \frac{dT}{dt} \right|_i^{t+1} = \Delta t \left. \frac{dT}{dt} \right|_i^t + (\Delta t)^2 \left. \frac{d^2T}{dt^2} \right|_i^t + \frac{(\Delta t)^3}{2} \left. \frac{d^3T}{dt^3} \right|_i^t + HOTS \quad (\text{B.2})$$

Then multiplying equation (B.1) by A and equation (B.2) by B and adding these together, we get,

$$\begin{aligned} (A)T_i^{t+1} + (B)\Delta t \left. \frac{dT}{dt} \right|_i^{t+1} &= (A)T_i^t + (A+B)\Delta t \left. \frac{dT}{dt} \right|_i^t + \left(\frac{A}{2} + B\right) \frac{(\Delta t)^2}{2} \left. \frac{d^2T}{dt^2} \right|_i^t + \\ &+ \left(\frac{A}{6} + \frac{B}{2}\right) \frac{(\Delta t)^3}{6} \left. \frac{d^3T}{dt^3} \right|_i^t + HOTS \end{aligned} \quad (\text{B.3})$$

Setting A to 2 and B to -1, while substituting in equation (2.1) appropriately, the result is the same as equation (2.28).

$$(2)\Delta T_i + (-\Delta t)\Delta \left. \frac{dT}{dt} \right|_i = (2\Delta t) \left. \frac{dT}{dt} \right|_i^t + \vartheta(\Delta t^3) \quad (\text{B.4})$$

Appendix C

Specific Portions of the C# Code

This appendix is only a portion of the original program that numerically solved the problems mentioned in this thesis by LU decomposition of a linear system of equations. There are three objects included here which were interacted with from a higher routine that initiated these and interacted with them appropriately. A part of their initiation involves defining of specific variables and parameters including assigning certain unknown variables to specific rows of the central matrix. In addition, the objects also control the coefficients of the central matrix which are defined based on the fundamental equations derived earlier, and the objects update their variables and parameters at the end of each given time step.

Appendix C.1

Solid Volume Object

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace FROG_program
{
    class oVolume
    {
        // Variables Initialized with Input
        public string VolumeID;
        private string Shape;
        public double Volume;

        public double PositionX, PositionY, PositionZ;

        public string Phase, Material;
        private string StateDefinitionMethod;
        private double StateProperty1, StateProperty2;
    }
}
```

```

private double HeatGenRate;

public int scalarUV1, scalarUV2, scalarUV3;

// Input
public oVolume(string[] InputLine, int[] UVrows)
{
    this.VolumeID = InputLine[1];
    this.Shape = InputLine[2];
    this.Volume = Convert.ToDouble(InputLine[3]);

    this.PositionX = Convert.ToDouble(InputLine[4]);
    this.PositionY = Convert.ToDouble(InputLine[5]);
    this.PositionZ = Convert.ToDouble(InputLine[6]);

    this.Phase = InputLine[22];
    this.Material = InputLine[23];

    this.StateDefinitionMethod = InputLine[24];
    this.StateProperty1 = Convert.ToDouble(InputLine[25]);
    this.StateProperty2 = 0;
    if (InputLine[26] != "N/A")
    {
        this.StateProperty2 = Convert.ToDouble(InputLine[26]);
    }

    this.HeatGenRate = Convert.ToDouble(InputLine[27]);

    this.scalarUV1 = UVrows[1]; // T_i
    this.scalarUV2 = UVrows[2]; // d2T/dx2_i
    this.scalarUV3 = UVrows[3]; // dT/dt_i
}

/*-----*/

// IDs Initialized with Linking
private oSubVolume[] oSubVolumeRefs = new oSubVolume[16];
public oFace[] oFaceRefs = new oFace[16];

// Receive IDs (this happens once at the beginning)
public int count = 0;
public void ReceiveRefs(oSubVolume oSubVolumeRef, oFace oFaceRef)
{
    this.count = count + 1;
    this.oSubVolumeRefs[count] = oSubVolumeRef;
    this.oFaceRefs[count] = oFaceRef;
}

/*-----*/

// Other Properties of Object
public double Temperature = 0.0;
public double d2T_dx2 = 0.0;
public double dT_dt = 0.0;

```

```

public double Pressure = 0.0;
public double Density = 0.0;
public double IntEnergy = 0.0;
public double Enthalpy = 0.0;
public double Entropy = 0.0;
public double c_v = 0.0;
public double c_p = 0.0;

public double ThermCond = 0.0;
public double Viscosity = 0.0;
public double ThermDiff = 0.0;
public double Prandtl = 0.0;

private SolidProperties Solid = new SolidProperties();

// Initialize Properties (this happens once at the beginning)
public void InitializeProperties()
{
    if (StateDefinitionMethod.Equals("Temperature"))
    {
        Temperature = StateProperty1;
    }
    if (Phase.Equals("Solid"))
    {
        this.Density = Solid.Density(Material, Temperature);
        this.c_p = Solid.c_p(Material, Temperature);
        this.ThermCond = Solid.ThermCond(Material, Temperature);
        this.ThermDiff = Solid.ThermDiff(Material, Temperature);
    }
    d2T_dx2 = 1.0/(2.0*oFaceRefs[1].HTLength*oFaceRefs[2].HTLength)
        *(4*oFaceRefs[1].Temperature
        + 4*oFaceRefs[2].Temperature
        + oFaceRefs[1].HTLength/ThermCond*oFaceRefs[1].HeatFlux
        + oFaceRefs[2].HTLength/ThermCond*oFaceRefs[2].HeatFlux
        - 8*Temperature);
    dT_dt = ThermDiff*(d2T_dx2 + 1.0/ThermCond*HeatGenRate);
}

/*-----*/

// Coefficients of Matrix Rows defined by System of Equations

public int mRow;
public int[] mCol;
public double[] Aval;
public int nCoeffs;

public void LHS_Coeffs(double deltaTime, int mRow_, int[] mCol_,
    double[] Aval_, int nCoeffs_, int irow)
{
    this.mRow = mRow_;
    this.mCol = mCol_;
    this.Aval = Aval_;
    this.nCoeffs = nCoeffs_;
}

```



```

// System of Equations
if (Phase.Equals("Solid"))
{
    if (irow == 1)
    {
        this.mRow = scalarUV1;
        this.nCoeffs = 2;

        // delta T_i
        this.mCol[1] = scalarUV1;
        this.Aval[1] = 2.0;

        // delta dT/dt_i
        this.mCol[2] = scalarUV3;
        this.Aval[2] = -deltaTime;
    }
    else if (irow == 2)
    {
        this.mRow = scalarUV2;
        this.nCoeffs = 6;

        // delta T_{i-1}
        this.mCol[1] = oFaceRefs[1].scalarUV1;
        this.Aval[1] = 4.0;

        // delta T_{i+1}
        this.mCol[2] = oFaceRefs[2].scalarUV1;
        this.Aval[2] = 4.0;

        // delta q"_{i-1}
        this.mCol[3] = oFaceRefs[1].fluxUV1;
        this.Aval[3] = 1.0 * oFaceRefs[1].HTLength / ThermCond;

        // delta q"_{i+1}
        this.mCol[4] = oFaceRefs[2].fluxUV1;
        this.Aval[4] = 1.0 * oFaceRefs[2].HTLength / ThermCond;

        // delta T_i
        this.mCol[5] = scalarUV1;
        this.Aval[5] = -8.0;

        // delta d2T/dx2_i
        this.mCol[6] = scalarUV2;
        this.Aval[6] = -2.0 * oFaceRefs[1].HTLength
            * oFaceRefs[2].HTLength;
    }
    else if (irow == 3)
    {
        this.mRow = scalarUV3;
        this.nCoeffs = 2;

        // delta d2T/dx2_i
        this.mCol[1] = scalarUV2;
    }
}

```

```

        this.Aval[1] = -1.0;

        // delta dT/dt_i
        this.mCol[2] = scalarUV3;
        this.Aval[2] = 1.0 / ThermDiff;
    }
}

public double Bval;

public void RHS_Coeffs(double deltaTime, int mRow_, double Bval_, int irow)
{
    this.mRow = mRow_;
    this.Bval = Bval_;

    // System of Equations
    if (Phase.Equals("Solid"))
    {
        if (irow == 1)
        {
            this.mRow = scalarUV1;
            this.nCoeffs = 1;

            // 3rd Order in Time
            this.Bval = 2.0 * deltaTime * dT_dt;
        }
        else if (irow == 2)
        {
            this.mRow = scalarUV2;
            this.nCoeffs = 1;

            // 6th Order
            this.Bval = -4.0 * oFaceRefs[1].Temperature
                - 4.0 * oFaceRefs[2].Temperature
                - 1.0 * oFaceRefs[1].HTLength / ThermCond
                    * oFaceRefs[1].HeatFlux
                - 1.0 * oFaceRefs[2].HTLength / ThermCond
                    * oFaceRefs[2].HeatFlux
                + 8.0 * Temperature
                + 2.0 * oFaceRefs[1].HTLength * oFaceRefs[2].HTLength
                    * d2T_dx2;
        }
        else if (irow == 3)
        {
            this.mRow = scalarUV3;
            this.nCoeffs = 1;

            // Cons. of Energy
            this.Bval = d2T_dx2 - 1.0 / ThermDiff * dT_dt
                + HeatGenRate / ThermCond;
        }
    }
}

```

```

/*-----*/

// Update Unknown Variables and Properties
public void UpdateUnknownVariables(double[] UVs)
{
    this.Temperature = Temperature + UVs[1];
    this.d2T_dx2 = d2T_dx2 + UVs[2];
    this.dT_dt = dT_dt + UVs[3];
}
public void UpdateProperties()
{
    if (Phase.Equals("Solid"))
    {
        this.Density = Solid.Density(Material, Temperature);
        this.c_p = Solid.c_p(Material, Temperature);
        this.ThermCond = Solid.ThermCond(Material, Temperature);
        this.ThermDiff = Solid.ThermDiff(Material, Temperature);
    }
}
}
}
}

```

Appendix C.2

Conduction Face Object

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace FROG_program
{
    class oFace
    {
        // Variables Initialized with Input
        public string FaceID, SubVolumeID, VolumeID, AdjFaceID, SectorID;
        // Note: There will either be an AdjFaceID or a SectorID not both.
        private string Shape;
        public double Area, HTLength, ContResist;

        public double PositionX, PositionY, PositionZ;

        private string StateDefinitionMethod;
        private double StateProperty1, StateProperty2;

        public double HeatFlux;
    }
}

```

```

public int scalarUV1, fluxUV1;

// Input
public oFace(string[] InputLine, int[] UVrows)
{
    this.FaceID = InputLine[1];
    this.SubVolumeID = InputLine[2];
    this.VolumeID = InputLine[3];
    string InputLine4 = InputLine[4];
    string Letter = InputLine4.Substring(0, 1);
    if (Letter.Equals("F"))
    {
        this.AdjFaceID = InputLine[4];
        this.SectorID = "N/A";
    }
    else if (Letter.Equals("S"))
    {
        this.AdjFaceID = "N/A";
        this.SectorID = InputLine[4];
    }
    this.Shape = InputLine[5];
    this.Area = Convert.ToDouble(InputLine[6]);
    this.HTLength = Convert.ToDouble(InputLine[7]);
    this.ContResist = Convert.ToDouble(InputLine[8]);

    this.PositionX = Convert.ToDouble(InputLine[9]);
    this.PositionY = Convert.ToDouble(InputLine[10]);
    this.PositionZ = Convert.ToDouble(InputLine[11]);

    this.StateDefinitionMethod = InputLine[27];
    this.StateProperty1 = Convert.ToDouble(InputLine[28]);
    this.StateProperty2 = 0;
    if (InputLine[29] != "N/A")
    {
        this.StateProperty2 = Convert.ToDouble(InputLine[29]);
    }

    this.HeatFlux = Convert.ToDouble(InputLine[30]);

    this.scalarUV1 = UVrows[1]; // T_i+1
    this.fluxUV1 = UVrows[2]; // q"_i+1
}

/*-----*/

// Receive and Give Refs (this happens once at the beginning)
public oVolume oVolumeRef;
public oSubVolume oSubVolumeRef;
public oFace oFaceRef, oSymFaceRef, oAdjFaceRef;
public oSector oSectorRef;
public void ReceiveRefs(oVolume oVolumeRef_, oSubVolume oSubVolumeRef_,
    oFace oFaceRef_, oFace oAdjFaceRef_, oSector oSectorRef_)
{
    this.oVolumeRef = oVolumeRef_;
}

```

```

    this.oSubVolumeRef = oSubVolumeRef_;
    this.oFaceRef = oFaceRef_;
    this.oAdjFaceRef = oAdjFaceRef_;
    this.oSectorRef = oSectorRef_;

    oVolumeRef.ReceiveRefs(oSubVolumeRef, oFaceRef);
    oSubVolumeRef.ReceiveRefs(oVolumeRef, oFaceRef);
    string Letter = SectorID.Substring(0, 1);
    if (Letter.Equals("S")) oSectorRef.ReceiveRefs(oFaceRef, oVolumeRef);
}

// Get Symmetric Face Reference (this happens once at the beginning)
// Must know normal vector of Faces
public void GetSymFaceRef()
{
    int count = oVolumeRef.count;
    oFace[] oFaceRefs = new oFace[count + 1];
    int Index;
    for (Index = 1; Index <= count; Index++)
    {
        oFaceRefs[Index] = oVolumeRef.oFaceRefs[Index];
    }
    for (Index = 1; Index <= count; Index++)
    {
        if (nUnitVectX == -oFaceRefs[Index].nUnitVectX)
            if (nUnitVectY == -oFaceRefs[Index].nUnitVectY)
                if (nUnitVectZ == -oFaceRefs[Index].nUnitVectZ)
                {
                    oSymFaceRef = oFaceRefs[Index];
                }
    }
}

/*-----*/

public string FaceColor = "red";

// Return Reference of Adjacent Face and Define Face as Red or Blue
public oFace ColorizeFace()
{
    // This only works if faces are called in same order around a volume for
    // each volume
    if (oSymFaceRef.FaceColor.Equals("red")) this.FaceColor = "blue";
    else if (oSymFaceRef.FaceColor.Equals("blue")) this.FaceColor = "red";

    return oAdjFaceRef;
}

/*-----*/

// Other Properties of Object
public double Temperature = 0.0;
public double Pressure = 0.0;
public double Density = 0.0;

```

```

public double IntEnergy = 0.0;
public double Enthalpy = 0.0;
public double Entropy = 0.0;
public double c_v = 0.0;
public double c_p = 0.0;

public double ThermCond = 0.0;
public double Viscosity = 0.0;
public double ThermDiff = 0.0;
public double Prandtl = 0.0;

public double dk_dT = 0.0;

private SolidProperties Solid = new SolidProperties();

// Initialize Properties (this happens once at the beginning)
public void InitializeProperties()
{
    if (StateDefinitionMethod.Equals("Temperature"))
        Temperature = StateProperty1;
    if (oVolumeRef.Phase.Equals("Solid"))
    {
        this.Density = Solid.Density(oVolumeRef.Material, Temperature);
        this.c_p = Solid.c_p(oVolumeRef.Material, Temperature);
        this.ThermCond = Solid.ThermCond(oVolumeRef.Material, Temperature);
        this.ThermDiff = Solid.ThermDiff(oVolumeRef.Material, Temperature);
        this.dk_dT = Solid.ThermalGradient("ThermCond", oVolumeRef.Material,
            oVolumeRef.Temperature);
    }
}

/*-----*/

// Coefficients of Matrix Rows defined by System of Equations

public int mRow;
public int[] mCol;
public double[] Aval;
public int nCoeffs;

public void LHS_Coeffs(double deltaTime, int mRow_, int[] mCol_,
    double[] Aval_, int nCoeffs_, int irow)
{
    this.mRow = mRow_;
    this.mCol = mCol_;
    this.Aval = Aval_;
    this.nCoeffs = nCoeffs_;

    // System of Equations
    if (oVolumeRef.Phase.Equals("Solid"))
    {
        if (irow == 1)
        {
            this.mRow = scalarUV1;
        }
    }
}

```

```

this.nCoeffs = 3;

string LetterF = AdjFaceID.Substring(0, 1);
string LetterS = SectorID.Substring(0, 1);
if (LetterS.Equals("S"))
{
    if (oSectorRef.BoundaryCondition.Equals("Temperature"))
    {
        // delta T_F01
        this.mCol[1] = scalarUV1;
        this.Aval[1] = 1.0;

        // delta T_F10
        this.mCol[2] = oSectorRef.scalarUV1;
        this.Aval[2] = -1.0;

        // delta qflux_F01
        this.mCol[3] = fluxUV1;
        this.Aval[3] = -ContResist;
    }
    else if (oSectorRef.BoundaryCondition.Equals("HeatFlux"))
    {
        // delta qflux_F01
        this.mCol[1] = fluxUV1;
        this.Aval[1] = 1.0;

        // delta qflux_F10
        this.mCol[2] = oSectorRef.fluxUV1;
        this.Aval[2] = -1.0;          // <-- Sector flux reversed

        // delta (blank)
        this.mCol[3] = scalarUV1;
        this.Aval[3] = 0.0;
    }
}
else if (FaceColor.Equals("red"))
{
    // delta qflux_F01
    this.mCol[1] = fluxUV1;
    this.Aval[1] = 1.0;

    // delta qflux_F10
    this.mCol[2] = oAdjFaceRef.fluxUV1;
    this.Aval[2] = 1.0;

    // delta (blank)
    this.mCol[3] = scalarUV1;
    this.Aval[3] = 0.0;
}
else if (FaceColor.Equals("blue"))
{
    // delta T_F01
    this.mCol[1] = scalarUV1;
    this.Aval[1] = 1.0;
}

```

```

        // delta T_F10
        this.mCol[2] = oAdjFaceRef.scalarUV1;
        this.Aval[2] = -1.0;

        // delta qflux_F01
        this.mCol[3] = fluxUV1;
        this.Aval[3] = -ContResist;
    }
}
else if (irow == 2)
{
    this.mRow = fluxUV1;
    this.nCoeffs = 7;

    string LetterF = AdjFaceID.Substring(0, 1);
    string LetterS = SectorID.Substring(0, 1);
    if (FaceColor.Equals("red"))
    {
        // delta T_i-1
        this.mCol[1] = oSymFaceRef.scalarUV1;
        this.Aval[1] = 0.0;

        // delta T_i+1
        this.mCol[2] = scalarUV1;
        this.Aval[2] = 1.0;

        // delta q"_{i-1}
        this.mCol[3] = oSymFaceRef.fluxUV1;
        this.Aval[3] = 0.0 / oVolumeRef.ThermCond * HTLength;

        // delta q"_{i+1}
        this.mCol[4] = fluxUV1;
        this.Aval[4] = 1.0 / oVolumeRef.ThermCond * HTLength;

        // delta T_i
        this.mCol[5] = oVolumeRef.scalarUV1;
        this.Aval[5] = -1.0;

        // delta d2T/dx2_i
        this.mCol[6] = oVolumeRef.scalarUV2;
        this.Aval[6] = 0.0 * HTLength * HTLength;
    }
    else if (FaceColor.Equals("blue"))
    {
        // delta T_i-1
        this.mCol[1] = oSymFaceRef.scalarUV1;
        this.Aval[1] = 2.0;

        // delta T_i+1
        this.mCol[2] = scalarUV1;
        this.Aval[2] = 2.0;

        // delta q"_{i-1}

```



```

        this.mCol[3] = oSymFaceRef.fluxUV1;
        this.Aval[3] = 1.0 / oVolumeRef.ThermCond * HTLength;

        // delta q"_{i+1}
        this.mCol[4] = fluxUV1;
        this.Aval[4] = 1.0 / oVolumeRef.ThermCond * HTLength;

        // delta T_i
        this.mCol[5] = oVolumeRef.scalarUV1;
        this.Aval[5] = -4.0;

        // delta d2T/dx2_i
        this.mCol[6] = oVolumeRef.scalarUV2;
        this.Aval[6] = 0.0 * HTLength * HTLength;
    }
}

public double Bval;

public void RHS_Coeffs(double deltaTime, int mRow_, double Bval_, int irow)
{
    this.mRow = mRow_;
    this.Bval = Bval_;

    // System of Equations
    if (oVolumeRef.Phase.Equals("Solid"))
    {
        if (irow == 1)
        {
            this.mRow = scalarUV1;
            this.nCoeffs = 1;

            string LetterF = AdjFaceID.Substring(0, 1);
            string LetterS = SectorID.Substring(0, 1);
            if (LetterS.Equals("S"))
            {
                if (oSectorRef.BoundaryCondition.Equals("Temperature"))
                {
                    // Eq 3
                    this.Bval = -Temperature + oSectorRef.Temperature
                        + ContResist*HeatFlux;
                }
                else if (oSectorRef.BoundaryCondition.Equals("HeatFlux"))
                {
                    // Eq 4
                    this.Bval = -HeatFlux + oSectorRef.HeatFlux;
                    // <-- Sector flux reversed
                }
            }
            else if (FaceColor.Equals("red"))
            {
                // Eq 4
            }
        }
    }
}

```

```

        this.Bval = -HeatFlux - oAdjFaceRef.HeatFlux;
    }
    else if (FaceColor.Equals("blue"))
    {
        // Eq 3
        this.Bval = -Temperature + oAdjFaceRef.Temperature
            + ContResist*HeatFlux;
    }
}
else if (irow == 2)
{
    this.mRow = fluxUV1;
    this.nCoeffs = 1;

    string LetterF = AdjFaceID.Substring(0, 1);
    string LetterS = SectorID.Substring(0, 1);
    if (FaceColor.Equals("red"))
    {
        // 2nd Order
        this.Bval = 0.0 * oSymFaceRef.Temperature
            - 1.0 * Temperature
            + 0.0 / oVolumeRef.ThermCond * HTLength
                * oSymFaceRef.HeatFlux
            - 1.0 / oVolumeRef.ThermCond * HTLength * HeatFlux
            + 1.0 * oVolumeRef.Temperature
            + 0.0 * HTLength * HTLength * oVolumeRef.d2T_dx2;
    }
    else if (FaceColor.Equals("blue"))
    {
        // 4th Order
        this.Bval = -2.0 * oSymFaceRef.Temperature
            - 2.0 * Temperature
            - 1.0 / oVolumeRef.ThermCond * HTLength
                * oSymFaceRef.HeatFlux
            - 1.0 / oVolumeRef.ThermCond * HTLength * HeatFlux
            + 4.0 * oVolumeRef.Temperature
            + 0 * HTLength * HTLength * oVolumeRef.d2T_dx2;
    }
}
}
}

/*-----*/

// Update Unknown Variables and Properties
public void UpdateUnknownVariables(double[] UVs)
{
    this.Temperature = Temperature + UVs[1];
    this.HeatFlux = HeatFlux + UVs[2];
}
public void UpdateProperties()
{
    if (oVolumeRef.Phase.Equals("Solid"))
    {

```

```

        this.Density = Solid.Density(oVolumeRef.Material, Temperature);
        this.c_p = Solid.c_p(oVolumeRef.Material, Temperature);
        this.ThermCond = Solid.ThermCond(oVolumeRef.Material, Temperature);
        this.ThermDiff = Solid.ThermDiff(oVolumeRef.Material, Temperature);
        this.dk_dT = Solid.ThermalGradient("ThermCond", oVolumeRef.Material,
            oVolumeRef.Temperature);
    }
}
}
}
}

```

Appendix C.3

Conduction Sector Object

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace FROG_program
{
    class oSector
    {
        // Variables Initialized with Input
        public string SectorID;
        private string Shape;
        public double Area;

        public double PositionX, PositionY, PositionZ;

        public string BoundaryCondition;
        private string StateDefinitionMethod;
        private double StateProperty1, StateProperty2;

        public double HeatFlux;

        public int scalarUV1, fluxUV1;

        // Input
        public oSector(string[] InputLine, int[] UVrows)
        {
            this.SectorID = InputLine[1];
            this.Shape = InputLine[2];
            this.Area = Convert.ToDouble(InputLine[3]);

            this.PositionX = Convert.ToDouble(InputLine[4]);
            this.PositionY = Convert.ToDouble(InputLine[5]);
            this.PositionZ = Convert.ToDouble(InputLine[6]);
        }
    }
}

```

```

this.BoundaryCondition = InputLine[16];
this.StateDefinitionMethod = InputLine[17];
this.StateProperty1 = Convert.ToDouble(InputLine[18]);
this.StateProperty2 = 0;
if (InputLine[19] != "N/A")
{
    this.StateProperty2 = Convert.ToDouble(InputLine[19]);
}

this.HeatFlux = Convert.ToDouble(InputLine[20]);

this.scalarUV1 = UVrows[1]; // T_i+1
this.fluxUV1 = UVrows[2]; // q_i+1
}

/*-----*/

// Receive Refs (this happens once at the beginning)
public oFace oFaceRef;
private oVolume oVolumeRef;
public void ReceiveRefs(oFace oFaceRef_, oVolume oVolumeRef_)
{
    this.oFaceRef = oFaceRef_;
    this.oVolumeRef = oVolumeRef_;
}

/*-----*/

// Other Properties of Object          <-- This may be able to be simplified
public double Temperature = 0.0;
public double Pressure = 0.0;
public double Density = 0.0;
public double IntEnergy = 0.0;
public double Enthalpy = 0.0;
public double Entropy = 0.0;
public double c_v = 0.0;
public double c_p = 0.0;

public double ThermCond = 0.0;
public double Viscosity = 0.0;
public double ThermDiff = 0.0;
public double Prandtl = 0.0;

private SolidProperties Solid = new SolidProperties();

// Initialize Properties (this happens once at the beginning)
<-- This may be able to be simplified
public void InitializeProperties()
{
    if (StateDefinitionMethod.Equals("Temperature"))
        Temperature = StateProperty1;
    if (oVolumeRef.Phase.Equals("Solid"))
    {
        this.Density = Solid.Density(oVolumeRef.Material, Temperature);
    }
}

```

```

        this.c_p = Solid.c_p(oVolumeRef.Material, Temperature);
        this.ThermCond = Solid.ThermCond(oVolumeRef.Material, Temperature);
        this.ThermDiff = Solid.ThermDiff(oVolumeRef.Material, Temperature);
    }
}

/*-----*/

// Coefficients of Matrix Rows defined by System of Equations

public int mRow;
public int[] mCol;
public double[] Aval;
public int nCoeffs;

public void LHS_Coeffs(double deltaTime, int mRow_, int[] mCol_,
    double[] Aval_, int nCoeffs_, int irow)
{
    this.mRow = mRow_;
    this.mCol = mCol_;
    this.Aval = Aval_;
    this.nCoeffs = nCoeffs_;

    // System of Equations
    if (oVolumeRef.Phase.Equals("Solid"))
    {
        if (irow == 1)
        {
            this.mRow = scalarUV1;
            this.nCoeffs = 3;

            if (BoundaryCondition.Equals("Temperature"))
            {
                // delta T_S
                this.mCol[1] = scalarUV1;
                this.Aval[1] = 1.0;

                // delta qflux_S
                this.mCol[2] = fluxUV1;
                this.Aval[2] = 0.0;

                // delta qflux_F10
                this.mCol[3] = oFaceRef.fluxUV1;
                this.Aval[3] = 0.0;
            }
            else if (BoundaryCondition.Equals("HeatFlux"))
            {
                // delta T_S
                this.mCol[1] = scalarUV1;
                this.Aval[1] = 1.0;

                // delta T_F10
                this.mCol[2] = oFaceRef.scalarUV1;
                this.Aval[2] = -1.0;
            }
        }
    }
}

```

```

        // delta qflux_S
        this.mCol[3] = fluxUV1;
        this.Aval[3] = oFaceRef.ContResist;
    }
}
else if (irow == 2)
{
    this.mRow = fluxUV1;
    this.nCoeffs = 3;

    if (BoundaryCondition.Equals("Temperature"))
    {
        // delta T_S
        this.mCol[1] = scalarUV1;
        this.Aval[1] = 0.0;

        // delta qflux_S
        this.mCol[2] = fluxUV1;
        this.Aval[2] = -1.0;

        // delta qflux_F10
        this.mCol[3] = oFaceRef.fluxUV1;
        this.Aval[3] = 1.0;
    }
    else if (BoundaryCondition.Equals("HeatFlux"))
    {
        // delta T_S
        this.mCol[1] = scalarUV1;
        this.Aval[1] = 0.0;

        // delta T_F10
        this.mCol[2] = oFaceRef.scalarUV1;
        this.Aval[2] = 0.0;

        // delta qflux_S
        this.mCol[3] = fluxUV1;
        this.Aval[3] = 1.0;
    }
}
}
}

public double Bval;

public void RHS_Coeffs(double deltaTime, int mRow_, double Bval_, int irow)
{
    this.mRow = mRow_;
    this.Bval = Bval_;

    // System of Equations
    if (oVolumeRef.Phase.Equals("Solid"))
    {
        if (irow == 1)

```

```

{
    this.mRow = scalarUV1;
    this.nCoeffs = 1;

    if (BoundaryCondition.Equals("Temperature"))
    {
        // Eq
        this.Bval = 0.0;    // Make sure this Equation is applied
                           // correctly such that UV will be zero
    }
    else if (BoundaryCondition.Equals("HeatFlux"))
    {
        // Eq 3s
        this.Bval = -Temperature + oFaceRef.Temperature
                    - oFaceRef.ContResist * HeatFlux;
    }
}
else if (irow == 6)
{
    this.mRow = fluxUV1;
    this.nCoeffs = 1;

    if (BoundaryCondition.Equals("Temperature"))
    {
        // Eq 4s
        this.Bval = HeatFlux - oFaceRef.HeatFlux;
    }
    else if (BoundaryCondition.Equals("HeatFlux"))
    {
        // Eq
        this.Bval = 0.0;    // Make sure this Equation is applied
                           // correctly such that UV will be zero
    }
}
}
}

/*-----*/

// Update Unknown Variables and Properties
public void UpdateUnknownVariables(double[] UVs)
{
    this.Temperature = Temperature + UVs[1];
    this.HeatFlux = HeatFlux + UVs[2];
}
public void UpdateProperties()
{
    if (oVolumeRef.Phase.Equals("Solid"))
    {
        this.Density = Solid.Density(oVolumeRef.Material, Temperature);
        this.c_p = Solid.c_p(oVolumeRef.Material, Temperature);
        this.ThermCond = Solid.ThermCond(oVolumeRef.Material, Temperature);
        this.ThermDiff = Solid.ThermDiff(oVolumeRef.Material, Temperature);
    }
}

```

}
 }
 }

Bibliography

- [1] Ferziger, Joel H., Milovan Perić. *Computational Methods for Fluid Dynamics*. 3rd ed. New York: Springer, 2002.
- [2] Incropera, Frank P., David P. Dewitt, Theodore L. Bergman, Adrienne S. Lavine. *Fundamentals of Heat and Mass Transfer*. 7th ed. USA: John Wiley & Sons, 2011.
- [3] Richardson, L.F., "The Approximate Arithmetic Solution by Finite Differences of Physical Problems Involving Differential Equations, with an Application to the Stresses in a Masonry Dam," *Transactions of the Royal Society of London, Series A*. Vol. 120, 1910, pp. 307-357.
- [4] Eckert, E.R.G., R.J. Goldstein, E. Pfender, W.E. Ibele, J.W. Ramsey, T.W. Simon, N.A. Decker, T.H. Kuehn, H.O. Lee, S.L. Girshick. 1986. Heat transfer—a review of 1985 literature. *International Journal of Heat and Mass Transfer* 29, 1767-1842.
- [5] Schneider, G.E., M.J. Raw, V. Hassani. "A Nine-Noded Quadratic Control-Volume-Based Finite Element for Heat Conduction", *Journal of Spacecraft and Rockets*, Vol. 22, No. 5 (1985), pp. 523-529.
- [6] Kester, Doug. Private Discussions. Summer 2015.

ACADEMIC VITA

Cameron S. Nelson
3550 E Prospect RD
York, PA 17402
csn5043@psu.edu

EDUCATION

The Pennsylvania State University

M. S. Degree in Mechanical Engineering, graduating May '16
Current GPA 3.68/4.00

Curriculum Highlights:

- M E 512: Heat Transfer—Conduction
- M E 513: Heat Transfer—Convection
- M E 515: Two-Phase Heat Transfer
- M E 521: Foundations of Fluid Mechanics I
- M E 522: Foundations of Fluid Mechanics II
- M E 523: Numerical Solutions Applied to Heat Transfer & Fluid Mechanics Problems
- M E 422: Principles of Turbomachinery

The Pennsylvania State University

Schreyer Honors College
B. S. Degree in Mechanical Engineering, May '14
GPA: 3.78/4.00

Curriculum Highlights:

- Honors Thesis on Determination of Natural Frequencies of Monolayer Graphene
- 34 total honors credits in ME, physics, calculus, and western music history courses
- Passed Fundamental of Engineering exam (EIT) (Oct. '13)

INTERNSHIPS

Johnson Controls, York, PA (May '14 to Jan '16 off and on)

Post Graduate Intern

- Analyzed and quantified thermal systems, chillers and chiller systems. Designed and wrote object-oriented heat transfer program in C#. Improved center of gravity and seismic calculations for YK chillers, improving accuracy, time, and conservatism

Exelon, Kennett Square, PA (June to Aug. '13), **Engineering Intern**

Weir American Hydro, York, PA (June to Aug. '12), **Engineering Intern**

EMPLOYMENT

Penn State University, University Park, PA (Feb. '16 to present), **Grader**

Penn State University, University Park, PA (Aug. '14 to May '15), **Teaching Assistant**

Penn State University, York, PA (Feb. to Apr. '12), **Tutor**

SOCIETIES & COMMUNITY ACTIVITIES

- ASME, ASHRAE
- Penn State University Park Orchestra ('13 to '16)