FINITE ELEMENT APPROXIMATIONS OF BURGERS' EQUATION WITH ROBIN'S BOUNDARY CONDITIONS

by

Lyle C. Smith III

Thesis submitted to the faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Mathematics

APPROVED:

John A. Burns, Chairman

Terry L. Herdman | Robert C. Rogers

July, 1997

Blacksburg, Virginia

FINITE ELEMENT APPROXIMATIONS OF BURGERS' EQUATION WITH ROBIN'S BOUNDARY CONDITIONS

by

Lyle C. Smith III

Committee Chairman: John A. Burns

Mathematics

(ABSTRACT)

This work is a numerical study of Burgers' equation with Robin's boundary conditions. The goal is to determine the behavior of the solutions in the limiting cases of Dirichlet and Neumann boundary conditions. We develop and test two separate finite element and Galerkin schemes. The Galerkin/Conservation method is shown to give better results and is then used to compute the response as the Robin's Boundary conditions approach both the Dirichlet and Neumann boundary conditions. Burgers' equation is treated as a perturbation of the linear heat equation with the appropriate realistic constants.

The goal is to determine if the use of the Robin's boundary conditions to approximate Dirichlet and Neumann boundary conditions affords any advantage over schemes that employ only "exact" Dirichlet or Neumann boundary conditions. Our finite element results indicate that solutions using appropriate Robin's boundary conditions approach the same solutions obtained by "exact" Dirichlet or Neumann boundary conditions. This allows us to obtain realistic solutions in some cases where the other schemes had previously failed.

ACKNOWLEDGMENTS

I want to express my gratitude to my friend, advisor, and thesis committee chairman Dr. John Burns. His encouragement, guidance, and insight have been invaluable both with this thesis and also with my undergraduate and graduate studies. I would also like to thank Dr. Terry Herdman and Dr. Robert Rogers for serving on my thesis committee and for their help throughout my years at Virginia Tech. The enthusiasm and insight of these three men has made my pursuit of mathematics both fun and rewarding.

I would like to acknowledge the financial support I received from the Air Force Office of Scientific Research under Grant F49620-96-1-0329 and the Center for Transportation Research.

Many thanks also to my companions Tom Bail and Shana Olds. They have been an encouragement for over five years as we have persevered together first through undergraduate school and now finishing our Masters' degrees. They have served as models of hard work and dedication. Thanks to Ken Massa as well for his patience and determination as we finished our theses together this summer.

Finally, I would like to express my deepest gratitude to Jesus Christ, my Creator, Redeemer, and Friend. I am forever indebted to Him for revealing to me true spiritual wisdom and mathematical insight. I look forward to spending eternity thanking Him face-to-face.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter One

INTRODUCTION

1.1 Introduction

In this thesis, we investigate the Burgers' equation with Robin's boundary conditions. The Burgers' equation is treated as a perturbation of the linear heat equation. We employ several realistic boundary conditions. We develop two finite element schemes to solve this problem numerically. Pugh [1] determined that the Galerkin/Conservation method was more accurate and computed more quickly than the Galerkin method for the Burgers' equation with Neumann boundary conditions. We compare these two schemes for Robin's boundary conditions. The Galerkin/Conservation method is shown to be preferable and is therefore used to solve the subsequent problems in this paper.

It is known that all steady state solutions of Burgers' equation with homogeneous Neumann boundary conditions are constant [2]. In this thesis, we study the behavior of the Burgers' equation with Robin's boundary conditions. We let the Robin's boundary conditions approach homogeneous Neumann boundary conditions and compare the Neumann solution to the asymptotic solution as Robin's boundary conditions approach Neumann boundary conditions. We also investigate asymptotic solutions as the Robin's boundary conditions approach Dirichlet boundary conditions.

1.2 The Heat Equation

Many engineering applications today rely on the dynamics of heat intensity and flow within objects. From bridge I-beams to microprocessors, knowledge of the temperature and its transient distribution is vital in design and implementation. Therefore we would like to be able to predict the temperature of an object given a set of initial conditions and boundary conditions.

Consider a rod composed of a heat-conducting material, as in Figure 1.2.1. Let the rod have length L and uniform cross-sectional area A. We assume that the lateral surface of the rod is perfectly insulated, so that heat energy can only be exchanged at the ends. We also assume that the temperature of the rod is constant throughout any given cross section, but may vary from one cross section to another. Temperature differences in the rod or at the ends will cause heat energy to flow lengthwise along the rod. We are interested in the equations that model this heat flow.

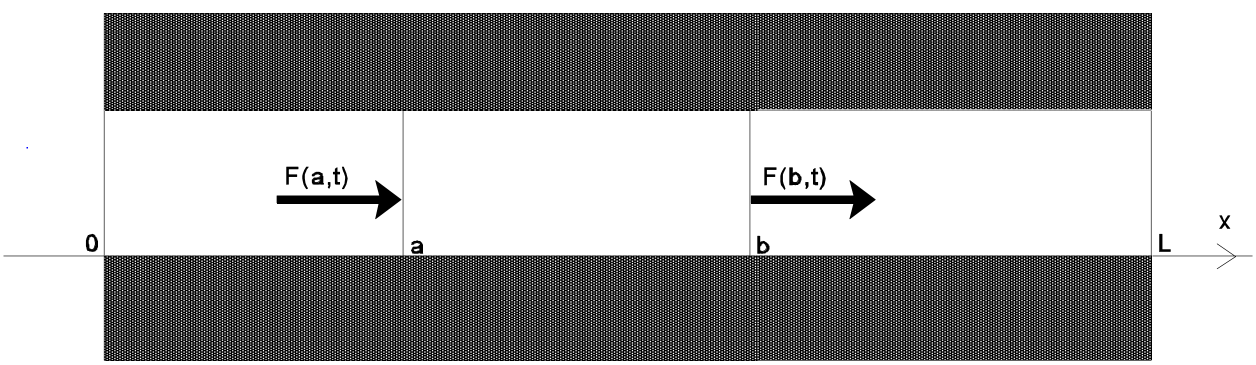

Figure 1.2.1

This problem was first addressed successfully by Joseph Fourier (1768-1830) in his *Théorie Analytique de la Chaleur* [3]. He was able to determine empirically that heat flow was governed (approximately) by a simple relationship,

$$F(t,x) = -\kappa \frac{\partial w}{\partial x}(t,x)$$

in one dimension. F(t,x) represents the flux term, defined as the quantity of heat energy passing unit area of the x cross section per unit time in the positive direction of the *x*-axis [4]. The temperature at position x and time t is represented by w(t,x). The negative sign proceeds from the fact that heat energy flows from warmer areas to cooler areas. The constant κ is called the thermal conductivity and is measured in calories per second centimeter Celsius degree (cal/s·cm·C°). The thermal conductivity is a property of the material used. A higher thermal conductivity indicates that the material is a better thermal conductor. Most metals are good conductors of electricity and also good conductors of heat. The ability of a metal to conduct an electric current is due to the fact that some electrons in the material have become detached from their parent atoms. These "free" electrons also provide an effective mechanism for heat transfer from the hotter to the cooler portions of the metal [5].

Let w(t,x) represent the temperature of the rod at time t and everywhere in the uniform cross-section at position x. Let ρ(x) be the density (mass per unit volume) at x and let c(x) be the specific heat of the rod at x. The specific heat is the amount of heat energy required to raise one unit of mass one degree of temperature and is measured in calories per gram Celsius degree (cal/g·C°). Strictly speaking, the specific heat can vary with temperature, but the variations are small and we assume the specific heat is constant with respect to temperature.

The mass of the rod between x and x+dx is ρ(x)Adx, where A is the cross-sectional area. The heat energy in the rod between x=a and x=b is then

$$Q(t) = \int_a^b w(t,x)\rho(x)c(x)A\,dx.$$

In this definition, we assume that the heat energy is zero when the temperature is zero. The heat energy in [a,b] can change either by heat flowing into or out of the region or by a source of heating or cooling in the region. Such a heat source may come from a heating element, a chemical reaction, or some other mechanism. We assume that the rate of heat production is constant on any given cross section. We let $f_1(t, x)$ represent this rate of heat production per unit time. The source term for the region [a,b] is then

$$\int_a^b f_1(t,x)\ A dx.$$

The rate of heat flow at x=a into [a,b] is A F(t,a) and the rate of heat flow at x=b out of [a,b] is - A F(t,b) (see Figure 1.2.1). Therefore, the flux term becomes

$$\begin{aligned} flux\ term &= -A[F(t,b) - F(t,a)] = A[\kappa(b)\frac{\partial w}{\partial x}(t,b) - \kappa(a)\frac{\partial w}{\partial x}(t,a)] \\ &= \int_a^b A\ \frac{\partial}{\partial x}\ (\kappa(x)\frac{\partial w}{\partial x}(t,x))\ dx. \end{aligned}$$

The law of conservation of energy then gives us

$$\frac{dQ}{dt} = flux\ term + source\ term,\ or$$

$$\int_a^b w_t(t,x)\ \rho(x)c(x)\ Adx = \int_a^b \frac{\partial}{\partial x}\ (\kappa(x)w_x(t,x))\ Adx + \int_a^b f_1(t,x)\ Adx,$$

which becomes

$$A\int_a^b [\rho(x)c(x)w_t(t,x) - \frac{\partial}{\partial x}(\kappa(x)w_x(t,x)) - f(t,x)]\ dx = 0.$$

From this equation, we get the heat equation

$$\rho(x)c(x)\ w_t(t,x) \ = \ \frac{\partial}{\partial x}\ (\kappa(x)\ w_x(t,x)) \ + \ f_1(t,x).$$

In this thesis we assume that the density, the specific heat and the thermal conductivity are independent of position. We then divide by ρc (assuming $\rho c \neq 0$) and define f(t,x) = f_1(t,x)/ρc to obtain the linear equation

$$\frac{\partial w}{\partial t}(t,x) \ = \ \varepsilon\ \frac{\partial^2 w}{\partial x^2}(t,x) \ + \ f(t,x). \tag{1.1.1}$$

Here ε represents the thermal diffusivity, equal to κ/cρ. Table 1.2.1 lists thermal properties for some common materials. The values in Table 1.2.1 are taken from Schenck [6] and converted with the conversions supplied by Kern [7]. The column R_E lists the equivalent "Reynold's numbers," which will be discussed in Section 1.4.

Equation (1.1.1) is a linear second order partial differential equation. In order to solve this equation, we need a set of initial conditions and boundary conditions. Since the heat equation is first order in time and second order spatially, we need one initial condition

$$w(0,x) \ = \ w_0(x)$$

and two boundary conditions. There are three common types of boundary conditions:

1. Dirichlet:

$$w(t,0) = u_1(t) \qquad w(t,L) = u_2(t) \qquad 0 \leq x \leq L$$

2. Neumann:

$$\kappa\frac{\partial w}{\partial x}(t,0)=u_1(t) \qquad \kappa\frac{\partial w}{\partial x}(t,L)=u_2(t) \qquad 0\le x\le L$$

3. Robin's:

$$\frac{\kappa_1}{L_1}w(t,0)-\kappa\frac{\partial w}{\partial x}(t,0)=\frac{\kappa_1}{L_1}u_1(t) \qquad \frac{\kappa_2}{L_2}w(t,L)+\kappa\frac{\partial w}{\partial x}(t,L)=\frac{\kappa_2}{L_2}u_2(t) \qquad 0\le x\le L$$

where κ_1, κ_2, L_1, and L_2 are the thermal conductivities and lengths of the film on the left and the film on the right, respectively. This paper will focus on the Robin's boundary conditions.

Material	κ (cal/s·cm·C°)	ρ (g/cm^3)	c (cal/g·C°)	ε (cm^2/s)	R_E
Aluminum	0.55	2.7	0.217	0.94	1.06
Copper	0.93	8.9	0.092	1.14	0.88
Iron	0.124	7.3	0.099	0.172	5.8
Silver	1.00	10.5	0.056	1.70	0.59
Tin	0.153	7.3	0.054	0.39	2.6
Masonry Brick	0.00149	1.70	0.199	0.0044	228
Corkboard	0.000089	0.112	0.38	0.0021	484
Soft Rubber	0.00041	1.11	0.48	0.00077	1290
Oak Wood	0.00046	0.70	0.41	0.00160	625
Ice	0.0053	0.91	0.49	0.0119	84
Granite	0.0066	2.7	0.189	0.0129	78
Stainless Steel	0.054	7.7	0.111	0.063	16
Glass wool	0.000083	0.0160	0.173	0.030	33

Table 1.2.1

1.3 Robin's Boundary Conditions

The third or Robin's boundary conditions (also called the convection surface condition [8]) describe the realistic situation of heating or cooling the body through contact with a second body, which is in contact with a heat source. Often the second body is a film between the first body and the heating element. This film might be the adhesive and/or casing of the heating element and is usually thin compared to the length of the first body. The temperature of the heating element is then varied to control the temperature at the end of the first body.

Suppose the first body is a uniform rod with thermal conductivity κ that extends from x=0 to x=L, as in Figure 1.3.1. Suppose also that the film has length L_1, thermal conductivity κ_1, and is in contact with the first body at x=0. Let $w_1(t, x)$ represent the temperature of the film. Assuming perfect thermal contact between the end of the rod and the film, the jump conditions at x=0 are (see [4])

$$w(t,0) = w_1(t,0) \qquad \kappa\frac{\partial w}{\partial x}(t,0) = \kappa_1\frac{\partial w_1}{\partial x}(t,0).$$

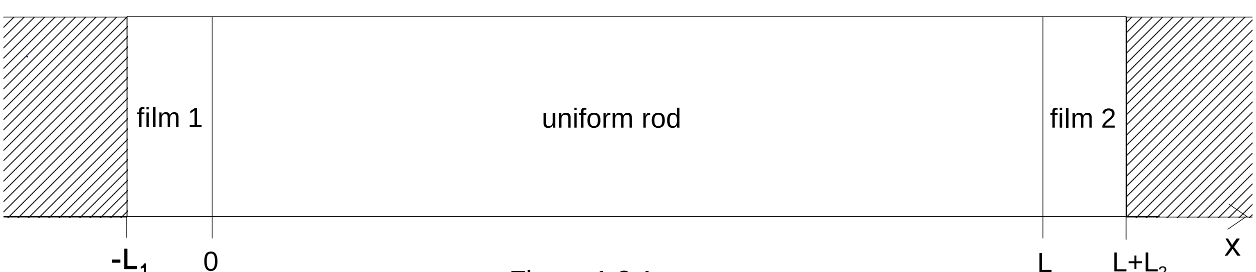

Figure 1.3.1

At the other end of the film, the temperature of the film is equal to the temperature of the heat source, $w_1(t, -L_1) = u_1(t)$.

Newton's law of "radiation" states that the heat flux is proportional to the temperature difference between the two bodies [9]. Equivalently, we approximate the derivative $\partial w_1/\partial x$ by the difference quotient,

$$\kappa_1 \frac{\partial w_1}{\partial x}(t,0) = \kappa_1 \frac{w_1(t,-L_1) - w_1(t,0)}{-L_1} = \frac{\kappa_1}{-L_1}\left(u_1(t) - w(t,0)\right).$$

The boundary condition at x=0 becomes

$$\kappa \frac{\partial w}{\partial x}(t,0) = \frac{\kappa_1}{-L_1}\left(u_1(t) - w(t,0)\right)$$

or equivalently,

$$\frac{\kappa_1}{L_1} w(t,0) - \kappa \frac{\partial w}{\partial x}(t,0) = \frac{\kappa_1}{L_1} u_1(t).$$

Similarly, the boundary condition at x=L becomes

$$\frac{\kappa_2}{L_2} w(t,L) + \kappa \frac{\partial w}{\partial x}(t,L) = \frac{\kappa_2}{L_2} u_2(t).$$

Again, κ refers to the thermal conductivity of the rod and κ_1 and κ_2 refer to the thermal conductivities of films one and two respectively. The constants L_1 and L_2 are the thicknesses of films one and two respectively. A list of thermal conductivities and other thermal properties for some common materials are given in Table 1.2.1.

1.4 The Burger's Equation

The Burgers' equation serves as a useful model for many interesting problems in applied mathematics. It models effectively certain problems of a fluid flow nature, in which either shocks or viscous dissipation is a significant factor.

The first steady-state solutions of Burgers' equation were given by Bateman [10] in 1915. However, the equation gets its name from the extensive research of Burgers [11] beginning in 1939. Burgers focussed on modeling turbulence, but the equation is useful for modeling such diverse physical phenomena as shock flows, traffic flow, acoustic transmission in fog, etc. In fact, it can be used as a model for any nonlinear wave propagation problem subject to dissipation [12]. Depending on the problem being modeled, this dissipation may result from viscosity, heat conduction, mass diffusion, thermal radiation, chemical reaction, or other source. We shall take the point of view that Burgers' equation is a perturbation of the linear heat equation. In particular, we consider the viscous Burgers' equation,

$$\frac{\partial w}{\partial t}(t,x) + w(t,x)\frac{\partial w}{\partial x}(t,x) = \varepsilon\frac{\partial^2 w}{\partial x^2}(t,x) + f(t,x), \tag{1.4.1}$$

where $\varepsilon>0$ is a viscosity constant in most problems. If equation (1.4.1) is given as a non-dimensional equation, ε is the inverse of the Reynolds number, R_E. In this paper, ε is the thermal diffusivity, described in Section 1.2. Burgers' equation is a second-order quasi-linear partial differential equation describing the time evolution of the dependent variable w(t,x). Depending on the magnitude of the various terms in the equation, it behaves as an elliptic, parabolic, or hyperbolic partial differential equation.

Burgers' equation has not been solved in closed form, but for many combinations of initial and boundary conditions, an exact solution can easily be found. Hopf [13] and Cole [14] discovered a transformation that reduces the Burgers' equation to the linear heat equation. However, these solution are restricted, in most cases, to the infinite interval $-\infty<x<\infty$ (see Fletcher [12]). Many practical applications require a finite interval and therefore cannot make use of this transformation.

The difficulty in computing solutions to the Burgers' equation lies in the inability to effectively balance the nonlinear convective term, ww_x, and the diffusive term, εw_{xx}. This can be readily seen by simplifying equation (1.4.1), as shown in Fletcher [12]. Dropping the convective term results in the heat equation, which is discussed in Section 1.2. Removal of the diffusive term gives the inviscous Burgers' equation:

$$w_t(t,x) + w(t,x)w_x(t,x) = 0,$$

which is a hyperbolic partial differential equation. Points on the solution curve to this equation with larger values of w convect faster than points with smaller values of w, resulting in w having more than one value for some interval at some future time t_1. In order to maintain a unique solution, we need to postulate a shock at which w is discontinuous.

When the time derivative term is dropped from equation (1.4.1), we are left with an elliptic partial differential equation representing the steady-state balance between the convective and diffusive terms. In transient solutions of this equation, the points on the solution curve with larger values of w again convect faster than points with smaller values of w. However, the diffusive term grows large as the curve steepens, thus preventing multi-valued solutions from

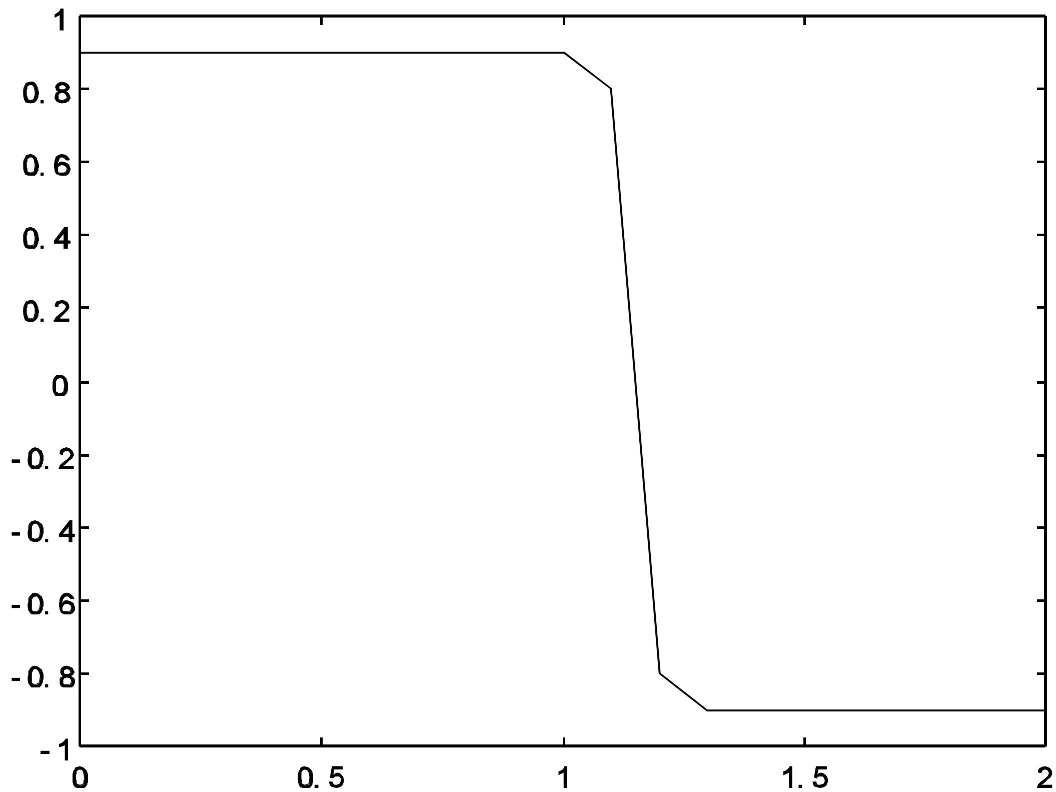

Figure 1.4.1
A typical numerical solution of $ww_x = \varepsilon w_{xx}$.

developing. A typical numerical steady-state solution is shown in Figure 1.4.1. In the limit as ε approaches 0 the solution changes discontinuously to a piece-wise constant function.

Burgers' equation exhibits these and other phenomena, but is simple enough to provide comparisons to more complex problems. For this reason, Burgers' equation is often used to test and compare numerical techniques. We would like to investigate some of the properties of Burgers' equation with Robin's boundary conditions.

1.5 Perspective

As mentioned in Section 1.4, Burgers' equation was first studied by Bateman [10] and then studied extensively by Burgers [11]. Fletcher [12] illustrates some of the dynamics and difficulties of solving Burgers' equation.

Pugh [1] compares the Galerkin finite element method and the Galerkin/Conservation method in solving the homogeneous Burgers' equation

$$\frac{\partial w}{\partial t}(t,x) + w(t,x)\frac{\partial w}{\partial x}(t,x) = \varepsilon\frac{\partial^2 w}{\partial x^2}(t,x)$$

with homogeneous Neumann boundary conditions on the interval [0,1]

$$\frac{\partial w}{\partial x}(t,0) = 0; \qquad \frac{\partial w}{\partial x}(t,1) = 0$$

with initial condition

$$w(0,x) = w_0(x).$$

In the conservation method, the homogeneous Burgers' equation is written as

$$\frac{\partial w}{\partial t}(t,x) + \frac{1}{2}\frac{\partial}{\partial x}(w^2(t,x)) = \varepsilon\frac{\partial^2 w}{\partial x^2}(t,x).$$

Pugh determined that N=16 elements provided adequate accuracy for his computations. In several test cases, these computations gave solutions that converged to the exact solution as N approaches infinity, as expected. Pugh discovered that the Galerkin/Conservation method computed more quickly with less error than the Galerkin finite element method. In some examples, the Galerkin/Conservation method converged to a solution when the Galerkin finite element method diverged.

Byrnes, Gilliam, and Shubov [15] proved that for a sufficiently small initial condition, the solution of the Burgers' equation with Neumann boundary conditions must converge to a constant as time approaches infinity. Burns et.al. [2] later showed that this was true for any initial condition, provided the steady-state limit exists. Pugh, however, obtained non-constant steady-state solutions for larger initial conditions and smaller ε ($R_E = 120, 240$). We now know that this is a purely numerical solution (see [2]).

Burgers' equation has also provided a test case for research on the problem of active control of fluid flows. Burns and Kang considered two linear regulator problems with Burgers' equation. Both problems involved applying boundary conditions to finite spatial intervals. In the first paper [16], they formulate a finite dimensional approximating system for the Burgers' equation with homogeneous Dirichlet boundary conditions on the finite interval [0, L]. In the second paper [17], the model was applied to the inhomogeneous Dirichlet case

$$w(t,0) = 0; \qquad w(t,L) = v(t).$$

In both papers, this nonlinear system of ordinary differential equations was solved using a fourth order Runge-Kutta method [18].

Marrekchi [19] studied another boundary control problem using the Burgers' equation with homogeneous Neumann boundary conditions. The resulting system of N+2 ordinary differential equations was solved with MATLAB using ODE45, which implements a fourth and fifth order Runge-Kutta method. He found that on the interval [0,1] with $R_E = 60$, the solution converged to a constant steady-state given the initial condition $w_0(x) = -\cos(\pi x)$, but diverged for the initial condition $w_0(x) = \cos(\pi x)$. Again, this is now known to be a purely numerical solution (see [2]).

Massa [20] is considering the boundary control problem of the Burgers' equation with Robin's boundary conditions, using the same model and MATLAB code as this paper.

1.6 Goals

In order to investigate the Burgers' equation with Robin's boundary conditions, we need to implement a numerical scheme to approximate the solution. We will be using a finite element scheme employing piece-wise linear polynomial basis functions, which is developed in Chapter two. We will investigate the accuracy of these schemes and determine which method is best (Galerkin or Galerkin/Conservation) for Burgers' equation with Robin's boundary conditions.

We will also investigate the nature of the numerical solutions to the Burgers' equation as the Robin's boundary conditions approach both the Dirichlet and the Neumann boundary conditions. In particular, we want to know if solutions for the Robin's boundary conditions approach solutions resulting from the homogeneous Neumann boundary conditions as L_1 and L_2 approach infinity. A similar issue is considered when the Robin's boundary conditions approach the Dirichlet boundary conditions, i.e. as L_1 and L_2 approach zero. We would like to determine if this asymptotic method yields a realistic answer when the methods using only Dirichlet or Neumann boundary conditions fail to.

Chapter Two

The Finite Element Method

2.1 A Review of the Equations

We now turn our attention to the computations. We wish to numerically solve Burgers' equation with Robin's Boundary conditions on the interval [0,L]. In particular, consider Burgers' equation

$$\frac{\partial w}{\partial t}(t,x) + w(t,x)\frac{\partial w}{\partial x}(t,x) = \varepsilon\frac{\partial^2 w}{\partial x^2}(t,x) + f(t,x) \tag{2.1.1}$$

with boundary conditions

$$\frac{\kappa_1}{L_1}w(t,0) - \kappa\frac{\partial w}{\partial x}(t,0) = \frac{\kappa_1}{L_1}u_1(t) \qquad \frac{\kappa_2}{L_2}w(t,L) + \kappa\frac{\partial w}{\partial x}(t,L) = \frac{\kappa_2}{L_2}u_2(t) \tag{2.1.2}$$

and initial condition

$$w(0,x) = w_0(x). \tag{2.1.3}$$

Again, it is helpful to think of w(t,x) as the temperature of the rod at time t and position x in our model, described in Section 1.2. In this case, κ is the thermal conductivity of the rod and ε is the thermal diffusivity of the rod. The constants κ_1 and κ_2 are the thermal conductivities of the left and right films of lengths L_1 and L_2, respectively.

2.2 The Galerkin Method and the Weak Form

First, we develop a weak formulation, from which we will derive the discretization. Multiplying both sides of the Burgers' equation by a test function, $v(\cdot)\in V$ (where V is a space of functions to be chosen later) and integrating, we get

$$\int_0^L \frac{\partial w}{\partial t}(t,x)v(x)dx \ + \ \int_0^L w(t,x)\frac{\partial w}{\partial x}(t,x)v(x)dx \ = \ \int_0^L \varepsilon \frac{\partial^2 w}{\partial x^2}(t,x)v(x)dx \ + \ \int_0^L f(t,x)v(x)dx.$$

Integrating by parts yields the equation

$$\int_0^L \frac{\partial w}{\partial t}(t,x)v(x)dx = -\int_0^L w(t,x)\frac{\partial w}{\partial x}(t,x)v(x)dx - \int_0^L \varepsilon\frac{\partial w}{\partial x}(t,x)v'(x)dx + \int_0^L f(t,x)v(x)dx + \varepsilon\left[\frac{\partial w}{\partial x}(t,x)v(x)\right]_0^L.$$

Substituting for the last term on the right from the boundary conditions, we get the weak formulation of the problem,

$$\begin{aligned}\int_0^L \frac{\partial w}{\partial t}(t,x)v(x)dx \ &= \ -\int_0^L w(t,x)\frac{\partial w}{\partial x}(t,x)v(x)dx \ - \ \varepsilon\int_0^L \frac{\partial w}{\partial x}(t,x)v'(x)dx \ + \ \int_0^L f(t,x)v(x)dx \\ &\quad + \ \frac{\varepsilon\kappa_2}{\kappa L_2}u_2(t)v(L) \ - \ \frac{\varepsilon\kappa_2}{\kappa L_2}w(t,L)v(L) \ + \ \frac{\varepsilon\kappa_1}{\kappa L_1}u_1(t)v(0) \ - \ \frac{\varepsilon\kappa_1}{\kappa L_1}w(t,0)v(0).\end{aligned} \tag{2.2.1}$$

2.3 The Finite Element Approximation

Now we want to choose V such that the matrices resulting from our discretization will be as easily solvable as possible. We choose linear basis functions, $b_n(x)$, on the interval [0,L]. The resulting matrices are tridiagonal. This sparseness given by these "hat" functions often improves computation time.

The interval of investigation, [0,L], is divided into N+1 subintervals $[x_i, x_{i+1}]$, each of length h=L/(N+1). Let the N+2 linear basis functions be denoted by $b_i(x)$ and defined by

$$b_0(x)=\begin{cases} \dfrac{x_1-x}{h} & \text{for } x_0\le x\le x_1 \\ 0 & \text{elsewhere} \end{cases}$$

$$b_i(x)=\begin{cases} \dfrac{x-x_{i-1}}{h} & \text{for } x_{i-1}\le x\le x_i \\ \dfrac{x_i-x}{h} & \text{for } x_i\le x\le x_{i+1} \\ 0 & \text{elsewhere} \end{cases}$$

$$b_{N+1}(x)=\begin{cases} \dfrac{x-x_N}{h} & \text{for } x_N\le x\le x_{N+1} \\ 0 & \text{elsewhere} \end{cases}$$

for $1\le i \le N$ and $0\le x \le L$. The approximate solution is then written as an expansion of these basis functions. In particular, we assume that

$$w^N(t,x) = \sum_{i=0}^{N+1} d_i(t)b_i(x),$$

where the coefficients $d_i(t)$ remain to be computed. We substitute $w^N(t,x)$ for w(t,x) and replace v(x) by our basis functions $b_j(x)$ in equation (2.2.1). The basis functions are orthogonal when i>j+1 and i<j-1. The result is a first-order non-linear differential equation for the vector function **d**(t):

$$\int_0^L \sum_{i=0}^{N+1} \dot{d}_i(t)b_i(x)b_j(x)dx = -\int_0^L \sum_{i=0}^{N+1} d_i(t)b_i(x)\sum_{k=0}^{N+1} d_k(t)b_k'(x)b_j(x)dx - \varepsilon\int_0^L \sum_{i=0}^{N+1} d_i(t)b_i'(x)b_j'(x)dx$$
$$+ \int_0^L f(x,t)b_j(x)dx - \frac{\varepsilon\kappa_2}{\kappa L_2}\sum_{i=0}^{N+1} d_i(t)b_i(L)b_j(L) + \frac{\varepsilon\kappa_2}{\kappa L_2}u_2(t)b_j(L) \tag{2.3.1}$$
$$+ \frac{\varepsilon\kappa_1}{\kappa L_1}u_1(t)b_j(0) - \frac{\varepsilon\kappa_1}{\kappa L_1}\sum_{i=0}^{N+1} d_i(t)b_i(0)b_j(0).$$

The basis functions satisfy $b_i(x_j) = \delta_{i\,j}$, thus we have $w^N(t,x_i) = d_i(t)$. So the coefficients $d_i(t)$ give the value of the approximate solution at the i^{th} node.

Looking at the term on the left, we have an (N+2) by (N+2) matrix with rows given by j and columns by i. We factor out the vector

$$\dot{\boldsymbol{d}}(t) = \begin{bmatrix} \dot{d}_0(t) \\ \dot{d}_1(t) \\ \vdots \\ \dot{d}_{N+1}(t) \end{bmatrix}_{(N+2)\times 1}$$

and call the remaining matrix the mass matrix, [M], due to its relation to the mass of the object being studied. The entries of the mass matrix are given by

$$m_{i,j} = \int_0^L b_i(x)b_j(x)dx.$$

The computation of this integral yields the mass matrix

$$[M] = \frac{h}{6}\begin{bmatrix} 2 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & 4 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & 0\cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & & & 1 & 4 & 1 \\ 0 & & \cdots & 0 & 1 & 2 \end{bmatrix}_{(N+2)\times(N+2)} .$$

The first term on the right of equation (2.3.1) results in a non-linear vector functional of **d**(t), which we call **J**(**d**(t)):

$$\boldsymbol{J}(\boldsymbol{d}(t)) = \frac{1}{6}\begin{bmatrix} -2(d_0(t))^2 + d_0(t)d_1(t) + (d_1(t))^2 \\ -(d_0(t))^2 + d_0(t)d_1(t) + d_1(t)d_2(t) + (d_2(t))^2 \\ \vdots \\ -(d_{N-1}(t))^2 + d_{N-1}(t)d_N(t) + d_N(t)d_{N+1}(t) + (d_{N+1}(t))^2 \\ -(d_N(t))^2 + d_N(t)d_{N+1}(t) + 2(d_{N+1}(t))^2 \end{bmatrix}_{(N+2)\times 1} .$$

The third term on the right of equation (2.3.1) yields the vector function

$$\hat{\boldsymbol{F}}(t) = \begin{bmatrix} \int_0^L f(t,x)b_0(x)dx \\ \int_0^L f(t,x)b_1(x)dx \\ \vdots \\ \int_0^L f(t,x)b_{N+1}(x)dx \end{bmatrix}_{(N+2)\times 1} .$$

Inspecting the second term on the right of equation (2.3.1), we have an (N+2) by (N+2) matrix with rows given by j and columns by i. The fourth and last terms of equation (2.3.1) also involve elements of **d**(t), so they are added to the (1, 1) and (N+2, N+2) entries of this matrix, respectively. We then factor out the vector

$$\boldsymbol{d}(t) = \begin{bmatrix} d_0(t) \\ d_1(t) \\ \vdots \\ d_{N+1}(t) \end{bmatrix}_{(N+2)\times 1}$$

and call the remaining matrix the stiffness matrix, [K]. The entries of the stiffness matrix are computed by

$$k_{i,j} = \int_0^L b_i'(x) b_j'(x) dx$$

yielding

$$[K] = \frac{1}{h} \begin{bmatrix} -1-\frac{h\kappa_1}{\kappa L_1} & 1 & 0 & \cdots & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0\cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & & & 1 & -2 & 1 \\ 0 & & \cdots & 0 & 1 & -1-\frac{h\kappa_2}{\kappa L_2} \end{bmatrix}_{(N+2)\times(N+2)} .$$

The fifth and sixth terms on the right side of equation (2.3.1) produce an (N+2) by 2 matrix with rows given by j and with 2 columns. Again, we factor out the vector **u**(t) (which we will call the control vector), giving the vector

$$\boldsymbol{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}_{(2\times 1)} .$$

The control vector represents the temperatures of the two heating elements, which can be varied in an effort to control the temperature of the rod. The remaining matrix is

$$[G] = \begin{bmatrix} \frac{\kappa_1}{\kappa L_1} & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & \frac{\kappa_2}{\kappa L_2} \end{bmatrix}_{(N+2)\times 2} .$$

We have now reduced the equation (2.3.1) to an (N+2)-dimensional matrix equation,

$$[M]\frac{d}{dt}\boldsymbol{d}(t) = -\boldsymbol{J}(\boldsymbol{d}(t)) + \varepsilon[G]\boldsymbol{u}(t) + \varepsilon[K]\boldsymbol{d}(t) + \hat{\boldsymbol{F}}(t). \tag{2.3.2}$$

Since [M] is known to be invertible, we can multiply both sides of equation (2.3.2) by $[M]^{-1}$ to obtain

$$\begin{aligned} \frac{d}{dt}\boldsymbol{d}(t) &= -[M]^{-1}\boldsymbol{J}(\boldsymbol{d}(t)) + \varepsilon[M]^{-1}[G]\boldsymbol{u}(t) + \varepsilon[M]^{-1}[K]\boldsymbol{d}(t) + [M]^{-1}\hat{\boldsymbol{F}}(t) \\ &= -[M]^{-1}\boldsymbol{J}(\boldsymbol{d}(t)) + \varepsilon[B]\boldsymbol{u}(t) + \varepsilon[A]\boldsymbol{d}(t) + \boldsymbol{F}(t). \end{aligned}$$

We now turn our attention to the initial condition. Multiplying both sides of the initial condition by v(x) and integrating on the interval [0, L], we get

$$\int_0^L w(0,x)v(x)dx = \int_0^L w_0(x)v(x)dx.$$

The initial condition is expanded in the linear basis functions and yields

$$\int_0^L \sum_{i=0}^{N+1} d_i(0)b_i(x)b_j(x)dx = \int_0^L w_0(x)b_j(x)dx, \qquad (2.3.3)$$

for j = 0, 1, 2, ... N+1.

Inspecting the term on the left side of equation (2.3.3), we again have an (N+2) by (N+2) matrix with the rows given by j and the columns by i. We factor out the vector

$$\boldsymbol{d}(0) = \begin{bmatrix} d_0(0) \\ d_1(0) \\ \vdots \\ d_{N+1}(0) \end{bmatrix}_{(N+2)\times 1}$$

and we see that the remaining matrix is again the mass matrix, [M].

On the right of equation (2.3.3) we have the (N+2)-dimensional vector

$$\boldsymbol{w_0}^{\boldsymbol{N}} = \begin{bmatrix} \int_0^L w_0(x)b_0(x)dx \\ \int_0^L w_0(x)b_1(x)dx \\ \vdots \\ \int_0^L w_0(x)b_{N+1}(x)dx \end{bmatrix}_{(N+2)\times 1} .$$

For the initial condition, we have

$$[M]\mathbf{d}_0 = \mathbf{w}_0^{N}.$$

Again multiply both sides by $[M]^{-1}$ to obtain the final form of the initial condition:

$$\mathbf{d}_0 = [M]^{-1}\, \mathbf{w}_0^N.$$

We have now reached the final form of the problem. We want to solve

$$\dot{\boldsymbol{d}}(t) \;=\; -[M]^{-1}\boldsymbol{J}(\boldsymbol{d}(t)) \;+\; \varepsilon[A]\boldsymbol{d}(t) \;+\; \varepsilon[B]\boldsymbol{u}(t) \;+\; \boldsymbol{F}(t) \tag{2.3.4}$$

given the initial condition

$$\boldsymbol{d}_{\mathbf{0}} \;=\; [M]^{-1}\boldsymbol{w}_{\mathbf{0}}^{\boldsymbol{N}}. \tag{2.3.5}$$

We are now ready to implement the computational code. In Chapter 3 this scheme is implemented in MATLAB and the resulting ODE is solved using the MATLAB function ODE45. The routine ODE45 uses fourth and fifth order Runge-Kutta methods to solve the ODE. The results are compared with known solutions and with Pugh [1] to determine the accuracy of the code.

2.4 The Galerkin/Conservation Method

An alternate form of Burgers' equation was given by Fletcher [21]. Here the ww_x term is replaced by $\frac{1}{2}(w^2)_x$. Burgers' equation is then written as

$$\frac{\partial w}{\partial t}(t,x) \;+\; \frac{1}{2}\frac{\partial}{\partial x}(w^2(t,x)) \;=\; \varepsilon\frac{\partial^2 w}{\partial x^2}(t,x) \;+\; f(t,x).$$

As before, we develop the weak form by multiplying both sides of the equation by a test function, v(x), and integrating to get

$$\int_0^L \frac{\partial w}{\partial t}(t,x)v(x)dx \ + \ \int_0^L \frac{1}{2}\frac{\partial}{\partial x}(w^2(t,x))v(x)dx \ = \ \int_0^L \varepsilon\frac{\partial^2 w}{\partial x^2}(t,x)v(x)dx \ + \ \int_0^L f(t,x)v(x)dx.$$

We integrate the second order term by parts to get a weak form

$$\begin{aligned}\int_0^L \frac{\partial w}{\partial t}(t,x)v(x)dx \ = \ -\int_0^L \frac{1}{2}\frac{\partial}{\partial x}[w^2(t,x)]v(x)dx \ - \ \varepsilon\int_0^L \frac{\partial w}{\partial x}(t,x)v'(x)dx \ + \ \int_0^L f(t,x)v(x)dx \\ + \ \frac{\varepsilon\kappa_2}{\kappa L_2}u_2(t)v(L) \ - \ \frac{\varepsilon\kappa_2}{\kappa L_2}w(t,L)v(L) \ + \ \frac{\varepsilon\kappa_1}{\kappa L_1}u_1(t)v(0) \ - \ \frac{\varepsilon\kappa_1}{\kappa L_1}w(t,0)v(0).\end{aligned} \tag{2.4.1}$$

Again we choose the space of linear polynomials for our test space, using the basis functions given in Section 2.3. In addition to the trial solution $w^N(t,x)$ given in Section 2.3, we now need to postulate a trial solution for $[w(t,x)]^2$,

$$[w^N(t,x)]^2 \ = \ \sum_{i=0}^{N+1} (d_i(t))^2 h_i(x).$$

Letting $v(x)=b_j(x)$ and substituting into equation (2.4.1) as before, we get a non-linear differential equation for the vector function $\mathbf{d}(t)$,

$$\begin{aligned}\int_0^L \sum_{i=0}^{N+1} \dot{d}_i(t)b_i(x)b_j(x)dx \ = \ -\int_0^L \frac{1}{2}\sum_{i=0}^{N+1} (d_i(t))^2 b_0 i'(x)b_j(x)dx \ - \ \varepsilon\int_0^L \sum_{i=0}^{N+1} d_i(t)b_i'(x)b_j'(x)dx \\ + \ \int_0^L f(x,t)b_j(x)dx \ - \ \frac{\varepsilon\kappa_2}{\kappa L_2}\sum_{i=0}^{N+1} d_i(t)b_i(L)b_j(L)+\frac{\varepsilon\kappa_2}{\kappa L_2}u_2(t)b_j(L) \\ + \ \frac{\varepsilon\kappa_1}{\kappa L_1}u_1(t)b_j(0)-\frac{\varepsilon\kappa_1}{\kappa L_1}\sum_{i=0}^{N+1} d_i(t)b_i(0)b_j(0).\end{aligned}$$

This then reduces to

$$[M]\frac{d}{dt}\boldsymbol{d}(t) \;=\; -\boldsymbol{J_C}(\boldsymbol{d}(t)) \;+\; \varepsilon[G]\boldsymbol{u}(t) \;+\; \varepsilon[K]\boldsymbol{d}(t) \;+\; \hat{\boldsymbol{F}}(t) \tag{2.4.2}$$

through the process shown in Section 2.3. This equation is equal to equation (2.3.2), except that **J**(**d**(t)) has been replaced by $\mathbf{J_C}$(**d**(t)):

$$\boldsymbol{J_C}(\boldsymbol{d}(t)) \;=\; \frac{1}{4}\begin{bmatrix} (d_1(t))^2 \;-\; (d_0(t))^2 \\ (d_2(t))^2 \;-\; (d_0(t))^2 \\ (d_3(t))^2 \;-\; (d_1(t))^2 \\ \vdots \\ (d_{N+1}(t))^2 \;-\; (d_{N-1}(t))^2 \\ (d_{N+1}(t))^2 \;-\; (d_N(t))^2 \end{bmatrix}_{(N+2)\times 1} .$$

Finally, we multiply both sides of equation (2.4.2) by [M^{-1}]. The initial condition is transformed in the same manner as in Section 2.3, giving us the final system to be solved. We want to solve

$$\dot{\boldsymbol{d}}(t) \;=\; -[M]^{-1}\boldsymbol{J_C}(\boldsymbol{d}(t)) \;+\; \varepsilon[A]\boldsymbol{d}(t) \;+\; \varepsilon[B]\boldsymbol{u}(t) \;+\; \boldsymbol{F}(t) \tag{2.4.3}$$

given the initial condition

$$\boldsymbol{d_0} \;=\; [M]^{-1}\; \boldsymbol{w_0^N}. \tag{2.4.4}$$

This system is solved in Chapter 3 using the MATLAB function ODE45. The results are compared with the results from the Galerkin finite element method of Section 2.3.

Chapter Three

NUMERICAL RESULTS

3.1 Setting Up the Problem

Consider the Burgers' equation

$$\frac{\partial w}{\partial t}(t,x) + w(t,x)\frac{\partial w}{\partial x}(t,x) = \varepsilon\frac{\partial^2 w}{\partial x^2}(t,x) + f(t,x)$$

with Robin's boundary conditions

$$\frac{\kappa_1}{L_1}w(t,0) + \kappa\frac{\partial w}{\partial x}(t,0) = \frac{\kappa_1}{L_1}u_1(t); \qquad \frac{\kappa_2}{L_2}w(t,L) + \kappa\frac{\partial w}{\partial x}(t,L) = \frac{\kappa_2}{L_2}u_2(t)$$

and initial condition $w_0(x)$

$$w(x,0) = w_0(x).$$

We are interested in computing approximate solutions of this problem.

The Galerkin finite element model was described in Chapter 2 and results in a system of ordinary differential equations:

$$\begin{gathered}\dot{\boldsymbol{d}}(t) + [M]^{-1}\boldsymbol{J}(\boldsymbol{d}(t)) = \varepsilon[A]\boldsymbol{d}(t) + \varepsilon[B]\boldsymbol{u}(t) + \boldsymbol{F}(t) \\ \boldsymbol{d_0} = [M]^{-1}\mathbf{w}_\mathbf{0}^N.\end{gathered} \tag{3.1.1}$$

Here $\mathbf{J}(\mathbf{d}(t))$ changes to $\mathbf{J_C}(\mathbf{d}(t))$ when the Galerkin/Conservation method is used, as shown in Section 2.4.

This problem is set up in MATLAB and solved using the ODE45 ordinary differential equation solver. Most of the examples presented in this paper employ numerical approximation schemes using N=16 elements. This is the most cost-effective number of elements, as shown in Section 3.2. This partitions the x-interval, [0,L], into N+1=17 subintervals of uniform length. The result is N+2=18 nodes, making (3.1.1) a system of 18 ordinary differential equations.

In nearly all of the examples in this paper, we use the Galerkin/Conservation method due to the increased efficiency afforded by it over the Galerkin method. This choice is supported in Sections 3.2 and 3.3, as well as by Pugh [1].

In some examples, we have an a priori knowledge of the exact solution. In this case, we measure the numerical error introduced by the two approximating schemes, by calculating the L^2 error at time t given by

$$Error\ (t) = [\ \int_0^L (w^N(t,x) - w(t,x))^2 dx\ ]^{1/2}.$$

This integral is computed using a three-point Guass quadrature scheme. This scheme evaluates $(w^N(t,x)-w(t,x))^2$ at three points in each subinterval, weighting each point differently. On the interval [-1,1], these points are x=-0.77459667, 0, 0.77459667 with weights 0.55555556, 0.88888889, and 0.55555556, respectively. These points and weights are chosen so as to optimize the accuracy of the numerical integration. The three-point Gauss quadrature scheme can integrate polynomials of order (2n-1)=5 exactly [18]. This is more than enough accuracy for computing the error of our numerical solutions, which are composed of piece-wise polynomials of order one.

The error is also shown graphically by computing $w^N(t,x)$ and w(t,x) at five evenly-spaced points in each subinterval. The exact solution and the difference $w^N(t,x)$ - $w(t,x)$ are then plotted at these points for comparison.

3.2 A Comparison to Known Solutions

In this section, we compare two methods of computation on two problems.The goal is to verify that the numerical schemes are returning the correct answer. We also want to determine which method and how many elements to use. The Galerkin method is compared to the Galerkin/Conservation method for N=4, 8, 16, and 32 elements. We conclude that the Galerkin/ Conservation method employing N=16 elements is the best choice for our computations.

For the examples in this section, we assume the form of a solution $w(t,x)$ and set up the system. We solve for the forcing function, $f(t,x)$, and the controls, $u_1(t)$ and $u_2(t)$. From Burgers' equation we have

$$f(t,x) = w_t(t,x) + w(t,x)w_x(t,x) - \varepsilon w_{xx}(t,x),$$

and the Robin's boundary conditions yield

$$u_1(t) = w(t,0) - \frac{\kappa L_1}{\kappa_1} w_x(t,0) \quad and \quad u_2(t) = w(t,L) + \frac{\kappa L_2}{\kappa_2} w_x(t,L).$$

Example 3.2.1

We first consider a problem with homogeneous Robin's boundary conditions where the function depends on the choices of κ, κ_1, κ_2, L_1, and L_2. Let

$$w(t,x) = e^{-\beta t}[Ax^2 + Bx + C + \sin^2(\frac{\pi}{100}x)]$$

where

$$A = \frac{100\kappa_1\kappa_2 + \kappa\kappa_2 L_1 + \kappa\kappa_1 L_2}{250{,}000}$$
$$B = -\frac{100^2\kappa_1\kappa_2 + 200\kappa\kappa_1 L_2}{250{,}000}$$
$$C = -\frac{100^2\kappa\kappa_2 L_1 + 200\kappa^2 L_1 L_2}{250{,}000}.$$

Then

$$f(t,x) = -(\beta[Ax^2+Bx+C+\sin^2(\frac{\pi}{100}x)]+\varepsilon[2A+2\frac{\pi^2}{100^2}(\cos^2(\frac{\pi}{100}x)-\sin^2(\frac{\pi}{100}x))])e^{-\beta t}$$
$$+ [Ax^2+Bx+C+\sin^2(\frac{\pi}{100}x)][2Ax+B+2\frac{\pi}{100}\sin(\frac{\pi}{100}x)\cos(\frac{\pi}{100}x)]e^{-2\beta t}$$

and

$$w_0(x) = w(0,x) = [Ax^2+Bx+C+\sin^2(\frac{\pi}{100}x)],$$
$$u_1(t) = 0; \qquad u_2(t) = 0.$$

The rod is assumed to be copper with length 1 meter, i.e. L=100, κ=.93, ρ=8.9, c=.092, and ε=1.14, from Table 1.2.1. The films are composed of 10 centimeters of aluminum on both ends, or L_1=L_2=10 and κ_1=κ_2=.55. We set the dimensionless variable β equal to .3 and solve the system from 0 to 15 seconds.

The exact solution is shown in Figure 3.2.1. Figures 3.2.2, 3.2.3, 3.2.4, and 3.2.5 show the numerical solution and the error for the Galerkin method at N = 4, 8, 16, and 32 respectively. Figures 3.2.6, 3.2.7, 3.2.8, and 3.2.9 show the numerical solution and the error for the

Galerkin/Conservation method at N = 4, 8, 16, and 32 respectively.

Table 3.2.1 compares the errors and computational time on a Pentium 133 for the Galerkin and Galerkin/Conservation methods using N= 4, 8, 16, and 32. At the initial time, t_i=0, the errors for the Galerkin and the Galerkin/Conservation methods will always be equal due to the fact that $\mathbf{d}_0 = [\mathbf{M}^{-1}]\, \mathbf{w_0}^N$, which has no dependence on the nonlinear term. The error at the initial time is dependent only on the spatial grid size, i.e. N. The error decreases with time because the solution decreases with time.

Comparing the errors at the final time for N = 4, 8, 16, and 32, we see that our numerical solutions are converging to the exact solution at better than the expected rate ($1/N^{½}$) for both the Galerkin and the Galerkin/Conservation methods. This is consistent with the finite element theory found in [21]. The errors for the Galerkin method are slightly smaller than the errors for the Galerkin/Conservation method. However, the Galerkin/Conservation method is computationally faster than the Galerkin method, especially as N increases to 16 and 32. Since the Galerkin/Conservation method is faster with only a small error difference, we will use it for all of the computations in Sections 3.3 and 3.4.

Comparing the errors at the initial and final time for N = 4, 8, 16, and 32, we see that sufficient accuracy is achieved at N = 16 elements. This is supported graphically in Figures 3.2.4 and 3.2.8.

Example 3.2.2

Next, we look at a problem with oscillating boundary conditions. Let

$$w(t,x) = \frac{(100-x)^2}{100^2}\sin(\alpha t) + \frac{x^2}{100^2}\cos(\beta t).$$

Then

$$f(t,x) = \alpha\frac{(100-x)^2}{100^2}\cos(\alpha t) - \beta\frac{x^2}{100^2}\sin(\beta t) - 2\frac{(100-x)^3}{100^4}\sin^2(\alpha t) + 2\frac{x^3}{100^4}\cos^2(\beta t)$$
$$+ \frac{2}{100^4}(2x^3 - 300x^2 + 100^2 x)\sin(\alpha t)\cos(\beta t) - \frac{2\varepsilon}{100^2}(\sin(\alpha t) + \cos(\beta t))$$

and

$$w_0(x) = w(0,x) = \frac{x^2}{100^2},$$
$$u_1(t) = (1 + \frac{2\kappa L_1}{100\kappa_1})\sin(\alpha t); \qquad u_2(t) = (1 + \frac{2\kappa L_2}{100\kappa_2})\cos(\beta t).$$

One thing we notice about this problem is that at the left end w(t,x)=sin(αt) and at the right end w(t,x)=cos(βt). The oscillations of the boundaries can be made as slow or fast as desired by varying α and β.

Again, we let the rod be composed of one meter of copper and the films be composed of 10 centimeters of aluminum on both ends. We set the dimensionless variables α and β equal to 1.5π and .5π respectively. We then solve the system from zero to ten seconds.

The exact solution is shown in Figure 3.2.10. Figures 3.2.11, 3.2.12, 3.2.13, and 3.2.14 show the numerical solution and the error for the Galerkin method at N = 4, 8, 16, and 32 respectively. Figures 3.2.15, 3.2.16, 3.2.17, and 3.2.18 show the numerical solution and the error for the Galerkin/Conservation method at N = 4, 8, 16, and 32 respectively.

Table 3.2.2 compares the errors and computational time on the same Pentium 133 for the

Galerkin and Galerkin/Conservation methods for N= 4, 8, 16, and 32. Again, at the initial time, t_i=0, the errors for the Galerkin and the Galerkin/Conservation methods will always be equal since the error is dependent only on the spatial grid size, N.

Comparing the errors at the final time for N = 4, 8, 16, and 32, we see that our numerical solutions are again converging to the exact solution at better than the expected rate ($1/N^{1/2}$) for both the Galerkin and the Galerkin/Conservation methods. As before, the errors for the Galerkin method are slightly smaller than the errors for the Galerkin/Conservation method. However, the Galerkin/Conservation method is computationally faster than the Galerkin method, especially as N increases to 16 and 32. Since the Galerkin/Conservation method is faster with only a small error difference, we will use it for all of the computations in Sections 3.4 and 3.5.

Comparing the errors at the initial and final time for N = 4, 8, 16, and 32, we see that sufficient accuracy is achieved at N = 16 elements. This is supported graphically as well in Figures 3.2.13 and 3.2.17. One thing we notice about this problem is that the error at the final time is greater than the initial error. This is due to computational errors in the ODE45 solver as it integrates over the time domain.

		Galerkin Method	Galerkin/Conservation
N = 4	computation time	.33	.33
	L_2 error at 0 seconds	.25374	.25374
	L_2 error at 15 seconds	.023151	.030779
N = 8	computation time	.39	.33
	L_2 error at 0 seconds	.069654	.069654
	L_2 error at 15 seconds	.0053532	.0061617
N = 16	computation time	.55	.44
	L_2 error at 0 seconds	.018768	.018768
	L_2 error at 15 seconds	.0012876	.0015233
N = 32	computation time	.99	.77
	L_2 error at 0 seconds	.0049239	.0049239
	L_2 error at 15 seconds	.00032822	.00039183

Table 3.2.1

Computation times and errors for N = 4, 8, 16, 32;

$w(t,x) = e^{-.3t}[Ax^2 + Bx + C + \sin(.01\pi x)]$.

		Galerkin Method	Galerkin/Conservation
N = 4	computation time	2.42	2.58
	L_2 error at 0 seconds	.029814	.029814
	L_2 error at 10 seconds	.034256	.055672
N = 8	computation time	3.46	3.08
	L_2 error at 0 seconds	.0092019	.0092019
	L_2 error at 10 seconds	.010892	.019133
N = 16	computation time	4.72	3.19
	L_2 error at 0 seconds	.0025791	.0025791
	L_2 error at 10 seconds	.0030195	.0052827
N = 32	computation time	6.15	4.78
	L_2 error at 0 seconds	.00068444	.00068444
	L_2 error at 10 seconds	.00078870	.0013485

Table 3.2.2

Computation times and errors for N = 4, 8, 16, 32;

$w(t,x) = [(100-x)/100]^2\sin(1.5\pi t) + (x/100)^2\cos(.5\pi t)$.

3.3 A Comparison to Previous Work

In this section, we compare our solutions to the results obtained by Pugh [1] for the homogeneous Burgers' equation with homogeneous Neumann boundary conditions. Here f(t,x) is set equal to zero producing the homogeneous Burgers' equation. The homogeneous Neumann boundary conditions are obtained by setting κ_1 and κ_2 equal to zero. This is equivalent to using two films both made of a perfect insulator. The other constants are treated as dimensionless with κ=1 and ε=1/R_E. We want to solve

$$\frac{\partial w}{\partial t}(t,x) \;+\; w(t,x)\frac{\partial w}{\partial x}(t,x) \;=\; \frac{1}{R_E}\frac{\partial^2 w}{\partial x^2}(t,x)$$

subject to the homogeneous Neumann boundary conditions

$$\frac{\partial w}{\partial x}(t,0) \;=\; \frac{\partial w}{\partial x}(t,L) \;=\; 0$$

given the initial condition

$$w(0,x) \;=\; w_0(x).$$

The interval of interest we pick to be x $\in$ [0,1], matching Pugh.

Example 3.3.1

In this example we compare our solution to those obtained by Pugh for $w_0(x)=(1/16)\cos(\pi x)$. Figures 3.3.1, 3.3.2 and 3.3.3 show the results for R_E = 60, 120, and 240 respectively. Comparing with pages 47, 48, and 49 of Pugh [1], we see that our results are in very good agreement with his results. We also see that the Galerkin and Galerkin/Conservation method give similar results in all three cases. This supports our choice of the Galerkin/Conservation method for our computations due to its faster computation.

Example 3.3.2

In this example, we again compare our solution to those obtained by Pugh, this time for $w_0(x)=(½)\cos(\pi x)$. Figures 3.3.4, 3.3.5 and 3.3.6 show the results for R_E = 60, 120, and 240 respectively. Comparing with pages 50, 51, and 52 of Pugh [1], we again see very good agreement of the results. This example gives strong support of our choice of the Galerkin/Conservation method over the Galerkin method. For R_E = 60 and 120, the two methods give similar results, but in Figure 3.3.6 we see that the Galerkin/Conservation method again gives a slowly decreasing solution while the Galerkin solution blows up at approximately 5 seconds for R_E=240. It is much more likely, in view of the other solutions given in Figures 3.3.1 through 3.3.5, that the Galerkin/Conservation method gives the more accurate solution.

We saw that the Galerkin/Conservation method is computationally faster than the Galerkin method for the examples in Section 3.2. We now see that the Galerkin/Conservation method yields more realistic answers in some cases than the Galerkin method. Based on these numerical experiments, we conclude that the Galerkin/Conservation is better than the Galerkin method (for problems considered in this work). The rest of the computations in this paper will use the Galerkin/Conservation method.

3.4 Results Approaching Dirichlet Boundary Conditions

In this section, we examine the behavior of the solutions as the Robin's boundary conditions approach the Dirichlet boundary conditions. As noted in Section 1.3, the Robin's boundary conditions are

$$\frac{\kappa_1}{L_1}w(t,0) - \kappa\frac{\partial w}{\partial x}(t,0) = \frac{\kappa_1}{L_1}u_1(t); \qquad \frac{\kappa_2}{L_2}w(t,L) + \kappa\frac{\partial w}{\partial x}(t,L) = \frac{\kappa_2}{L_2}u_2(t).$$

The Dirichlet boundary conditions are given by

$$w(t,0) = u_1(t); \qquad w(t,L) = u_2(t).$$

It is easily seen that the Robin's boundary conditions approach the Dirichlet boundary conditions when κ approachs zero. This can also be implemented by letting κ_1 and κ_2 approach infinity or by letting L_1 and L_2 approach zero. If we let κ approach zero, we are assuming an increase in the insulating properties of the rod. This implies the existence of a perfect insulator in the limit (a common assumption). If we let κ_1 and κ_2 approach infinity, we require a perfect thermal conductor in the limit. However, letting L_1 and L_2 approach infinity or zero is mathematically equivalent and easily implemented in the code. Thus, we use L_1 and L_2 as parameters to vary. As we will see, for a rod one meter long, L_1 and L_2 need only be as small as one millimeter in some cases to achieve results similar to what is expected for "exact" Dirichlet boundary conditions.

Although the Galerkin and Galerkin/Conservation methods give similar solutions in many cases, we have concluded that the Galerkin/Conservation method is more likely to capture the actual evolution of the solution and that this approximation is generally faster to compute. We therefore use the Galerkin/Conservation method for the rest of the computations in this and the next section.

Example 3.4.1

In this example, we consider a one meter long rod composed of iron with copper films on both ends, i.e. $L=100$, $\kappa=.124$, $\rho=7.3$, $c=.099$, $\varepsilon=.172$, and $\kappa_1=\kappa_2=.93$ from Table 1.2.1. We choose $w_0(x) = (½)\cos(.01\pi x)$ for the initial condition and set the controls to zero, i.e. $u_1(t)=u_2(t)=0$. Figure 3.4.1 shows the long term evolution of the solution for $L_1=L_2= 10$. This solution is representative of the long term evolution for the problems in this example. The solution eventually reduces to near zero, as expected. We are interested in the short term evolution of the solutions, however. True Dirichlet boundary conditions would force the solutions to zero at the ends almost immediately. Figures 3.4.2, 3.4.3, 3.4.4, and 3.4.5 show the results for $L_1=L_2= 10$, 1, 0.1, and 0.01 respectively.

We see that the solutions are approaching what we would expect the solution to look like for true Dirichlet boundary conditions. When $L_1=L_2= 0.01$, the solution is forced to zero almost immediately, as expected. We see that, in this case, film lengths of 0.01cm are small enough to yield qualitatively the Dirichlet results. However, the film lengths are not the only factor. We also need to be concerned with κ, κ_1, and κ_2, since it is the ratios of κ_1/L_1 and κ_2/L_2 to κ that determines how close we are to the Dirichlet boundary conditions. In this case, $(\kappa_1/L_1)/\kappa = (\kappa_2/L_2)/\kappa = 750$ is sufficient to produce qualitatively the Dirichlet results.

Example 3.4.2

In this example, we consider a one meter long rod now composed of aluminum with silver films on both ends, i.e. $L=100$, $\kappa=.55$, $\rho=2.7$, $c=.217$, $\varepsilon=.94$, and $\kappa_1=\kappa_2=1$. Now we choose $w_0(x) = .3\sin(.01\pi x)$ for the initial condition and again set the controls to zero, i.e. $u_1(t)=u_2(t)=0$. Figure 3.4.6 shows the long term evolution of the solution for $L_1=L_2= 10$. This solution is representative of the long term evolution for the problems in this example. The solutions eventually reduce to near zero, as expected. Again we are interested in the short term evolution of the solution, however. True Dirichlet boundary conditions would force the solution to remain at zero at the ends for all time. Figures 3.4.7, 3.4.8, and 3.4.9 show the results for $L_1=L_2= 10$, 1, and 0.1 respectively.

We see that the solutions are approaching what we would expect the solution to look like for true Dirichlet boundary conditions. When $L_1=L_2= 0.1$, the solution remains at zero at both ends, as expected. We see that, in this example, film lengths of 0.1cm are small enough to yield qualitatively the Dirichlet results. In this case, $(\kappa_1/L_1)/\kappa = (\kappa_2/L_2)/\kappa$ is approximately 18.

3.5 Results Approaching Neumann Boundary Conditions

In this section we examine the behavior of the solutions as we let the Robin's boundary conditions approach the homogeneous Neumann boundary conditions. As given in Section 1.3, the Robin's boundary conditions are

$$\frac{\kappa_1}{L_1}w(t,0)-\kappa\frac{\partial w}{\partial x}(t,0)=\frac{\kappa_1}{L_1}u_1(t); \qquad \frac{\kappa_2}{L_2}w(t,L)+\kappa\frac{\partial w}{\partial x}(t,L)=\frac{\kappa_2}{L_2}u_2(t)$$

and the homogeneous Neumann boundary conditions are

$$\kappa\frac{\partial w}{\partial x}(t,0)=0; \qquad \kappa\frac{\partial w}{\partial x}(t,L)=0.$$

It is easily seen that we can let the Robin's boundary conditions approach the homogeneous Neumann boundary conditions by letting κ approach infinity, by letting κ_1 and κ_2 approach zero, or by letting L_1 and L_2 approach infinity. Again, we will vary L_1 and L_2 since this is mathematically equivalent and easily implemented in the code..

Example 3.5.1

In this example, we consider a one meter long rod composed of aluminum with iron films on both ends, i.e. L=100, κ=.55, ρ=2.7, c=.217, ε=.94, and $\kappa_1=\kappa_2$=.124. We choose $w_0(x)$=.4cos(.01πx) for the initial condition and compute the solution for 10,000 seconds. Figure 3.5.1 shows the Neumann solution obtained by setting $\kappa_1=\kappa_2=0$. This corresponds to L_1 and L_2 equal to infinity. The true Neumann boundary conditions force the solution to remain flat (w_x=0) at the ends for all time. Figures 3.5.2, 3.5.3, 3.5.4, and 3.5.6 show the results for $L_1=L_2$= 100, 1000, 10,000, and 100,000 respectively. Figure 3.5.5 shows the long term solution for $L_1=L_2$=

10,000. It appears that the solution reaches a non-zero constant steady state.after approximately 170,000 seconds.

We see that the solutions are approaching the solution for Neumann boundary conditions as L_1 and L_2 approach zero. However, it appears that the solutions using the Robin's boundary conditions will eventually reach zero, while the Neumann solution appears to remain nonzero indefinitely. We see that, in this example, $L_1 = L_2 = 100{,}000$ is large enough to yield approximately the Neumann results. In this case, $(\kappa_1/L_1)/\kappa = (\kappa_2/L_2)/\kappa$ is approximately 0.0000023.

Example 3.5.2

In this example, we consider a one meter long rod composed of granite with iron films on both ends, i.e. L=100, κ=.0066, ρ=2.7, c=.189, ε=.0129, and $\kappa_1=\kappa_2$=.124. We choose $w_0(x)$ = cos(.01πx) for the initial condition and compute the solution for 10,000 seconds. Figure 3.5.7 shows the Neumann solution obtained by setting $\kappa_1=\kappa_2=0$. This corresponds to L_1 and L_2 equal to infinity or to perfectly insulating films. The true Neumann boundary conditions force the solution to remain flat (w_x=0) at the ends for all time. Our solution is similar to what Marrekchi obtained [19, p.76]. He set R_E=60 while our granite rod corresponds to a Reynolds number of 78.

Figures 3.5.8, 3.5.9, and 3.5.10 show the results for $L_1 = L_2$ = 1000, 10,000, and 100,000 respectively. We needed to use N=256 elements to obtain "smooth" results in this example. We see that the solutions are approaching the Neumann solution as L_1 and L_2 approach infinity. We see that, in this example, film lengths of 100,000 are large enough to yield approximately the Neumann results. In this case, $(\kappa_1/L_1)/\kappa = (\kappa_2/L_2)/\kappa$ is approximately 0.00019.

Chapter Four

CONCLUSIONS

4.1 Overview of Results

In this thesis, we used the Galerkin method and the Galerkin/Conservation method (outlined in Chapter 2) to approximate solutions to the initial/boundary value problem

$$\begin{gathered}\frac{\partial w}{\partial t}(t,x) + w(t,x)\frac{\partial w}{\partial x}(t,x) = \varepsilon\frac{\partial^2 w}{\partial x^2}(t,x) + f(t,x) \;;\quad w(0,x) = w_0(x)\\ \frac{\kappa_1}{L_1}w(t,0) - \kappa\frac{\partial w}{\partial x}(t,0) = \frac{\kappa_1}{L_1}u_1(t) \qquad \frac{\kappa_2}{L_2}w(t,L) + \kappa\frac{\partial w}{\partial x}(t,L) = \frac{\kappa_2}{L_2}u_2(t).\end{gathered} \tag{4.1.1}$$

A solution to this equation, starting initially at $w(0,x)=w_0(x)$, describes the time evolution of the function w(t,x) subject to the Robin's boundary conditions on the closed interval $x\in[0, L]$. We treated problem (4.1.1) as a perturbation of the linear heat equation, with w(t,x) representing the temperature of a rod at position x and time t. Using N = 4, 8, 16, and 32 elements, we compared our results obtained from the two approximation methods on the bases of accuracy, speed (computational execution time), and behavior. We investigated two specific problems where we knew the exact solution and found that the two methods both produced solutions that converged to the exact solution at better than the expected $(1/N)^{1/2}$ rate [21]. The Galerkin method produced solutions with slightly less error, but the Galerkin/Conservation method required less CPU time. We also compared our solutions to those for the homogeneous Neumann boundary conditions obtained by Pugh [1] for two initial conditions and Reynolds numbers of 60, 120, and 240.

Again, both methods produced virtually identical results except for one case where the Galerkin solution grew exponentially in time despite the convergent behavior of the Galerkin/Conservation solution. These reults are similar to what was obtained by Pugh [1]. Since the Galerkin/Conservation method is also generally faster with respect to computation time, we conclude that it is "better" in the sense that it is computationally more efficient.

We also investigated the cases when the Robin's boundary conditions were allowed to approach both the Dirichlet and the Neumann boundary conditions. We looked at two examples for each case. For the Dirichlet case, we let $L_1=L_2$ approach zero (or $\kappa_1=\kappa_2\rightarrow\infty$), representing a very thin film (or a very good conductor) on each end of the rod. In both examples, the solutions seemed to uniformly approach a limiting solution as $L_1=L_2$ approached zero. For the Neumann case, we let $L_1=L_2$ approach infinity (or $\kappa_1=\kappa_2\rightarrow 0$), representing a very thick film (or a very good insulator) on each end of the rod. These thick films act as insulators, allowing us to approach the Neumann case, which corresponds to films composed of perfect insulators. We produced the Neumann solution by setting $\kappa_1=\kappa_2=0$, for comparison. In both examples, our solution seemed to uniformly approach the Neumann solution as $L_1=L_2$ approached infinity. However, the solutions using the Robin's appeared to reach zero at some large time, where the Neumann solution seemed to have reached a nonconstant steady state.

4.2 Conclusions

The numerical results contained in this paper are consistent with Burns et.al. [2]. We seemed to obtain nonconstant steady state solutions for certain initial conditions and Reynolds numbers using "exact" Neumann boundary conditions. However, when the more realistic Robin's boundary conditions are employed, our results seem to show that the solutions will reach zero in the steady state. However, if $L_1=L_2$ are large enough, we would expect to see nonconstant steady state solutions due to computational errors, as described in [2].

Our numerical results suggest that when the Robin's boundary conditions are allowed to approach the Dirichlet and the Neumann boundary conditions, the solutions obtained approach the solutions corresponding to the Dirichlet and Neumann problems respectively. This could be very useful in cases where the solutions for the "exact" Dirichlet and Neumann problems are not easily obtainable by direct computation.

4.3 Open Problems

Our numerical results give rise to several interesting questions. It is our hope that we have provided motivation for research into the following areas:

- Can one prove that the solutions for the Robin's boundary conditions approach the Dirichlet and Neumann solutions as the Robin's boundary conditions are allowed to approach the Dirichlet and Neumann boundary condition, respectively?
- Can the methods of this paper be successfully applied to other areas of research on Burgers' equation, e.g. control theory? This question is being investigated in part by Massa [20].
- Can one prove that non-constant steady state solutions exist for the Burgers' equation with Robin's boundary conditions?

Appendix A

MATLAB CODE

FINEBURG.M

```
% This is the main body of the Finite Element Method for solving Burgers' Equation.
clear all
time1=cputime;

% Set up the geometry vectors.
geometry

% Set up the constants.
tfinal=7.5;
L1=10; % length of film 1 in cm
L2=10; % length of film 2 in cm
kap1=.55; % conductivity of film 1 in cal/(s*cm*C)
kap2=.55; % conductivity of film 2 in cal/(s*cm*C)
kappa=.93; % conductivity of the rod in cal/(s*cm*C)
epsilon=kappa/(8.9*.092)  % epsilon=kappa/(rho*c)
%epsilon=1/60;
alpha=1.5*pi;
beta=.3;     % for example 1
%beta=.5*pi;  % for example 2
% set up A,B,C used in example 1
quadA=(100*kap1*kap2 +kappa*kap2*L1 +kappa*kap1*L2)/250000;
quadB=-(10000*kap1*kap2 +200*kappa*kap1*L2)/250000;
quadC=-(10000*kappa*kap2*L1 +200*kappa*kappa*L1*L2)/250000;

% Set up the matrices.
Matrices

% Set up the initial condition.
v0=M*V;

time2=cputime;
% Solve the system
global A B F L1 L2 kap1 kap2 kappa alpha beta M
```

```
global epsilon quadA quadB quadC Acl
% Use 'burgrhs' for the Galerkin method, 'conrhs' for the G/C method.
[T,v] = ODE45('clrhs', 0,tfinal, v0);
%  for Example three
%[Tgc,vgc] = ODE45('conrhs', 0,tfinal, v0);
time3=cputime;
odetime=time3-time2
totaltime=time3-time1

% Begin the plotter routine.
% Use step to set the number of time steps plotted, the denominator.
step = ceil(max(size(T))/40);
count=0;
xxx=zeros(5*EN+6,1);
vplot=zeros(ceil(max(size(T))/step)+1,EN+2);
%error=zeros(ceil(max(size(T))/step)+1,1);
%errplt=zeros(ceil(max(size(T))/step)+1,5*EN+6);
%eplot=zeros(ceil(max(size(T))/step)+1,5*EN+6);
Tplot=zeros(ceil(max(size(T))/step)+1,1);
for i = 1:step:max(size(T))
  count=count+1;
  count2=0;
  vplot(count,:)=v(i,:);
% This subroutine computes the errors and error plots.
%   for j=1:EN+1
%     for jj=1:5
%       fem=0;
%       count2=count2+1;
%       xxx(count2+1,1)=(b-a)*(count2)/(5*EN+5);
%       eplot(count,count2)=solution(xxx(count2,1),T(i));
%       fem=vplot(count,j)*basis(j-1,xxx(count2,1),Xpt,h)+...
%         vplot(count,j+1)*basis(j,xxx(count2,1),Xpt,h);
%       errplt(count,count2)=fem-eplot(count,count2);
%     end
%   end
%       eplot(count,5*EN+6)=solution(100,T(i));
%       fem=vplot(count,EN+2);
%       errplt(count,5*EN+6)=fem-eplot(count,5*EN+6);
  Tplot(count,1)=T(i);
%   error(count,1)=lerr(vplot,XQ3pt,Node,Xpt,h,count,EN,Tplot);
  plot(Xpt,vplot(count,:))  %,xxx,eplot(count,:),'red')
  axis([0,100,-1,1])
  xlabel('x')
```

```
   ylabel('w(t,x)')
   title('Heat Flow')
   pause (.9)
end
%  for Example three
%count=0;
%step2 = ceil(max(size(Tgc))/60);
%v2plot=zeros(ceil(max(size(Tgc))/step2)+1,EN+2);
%T2plot=zeros(ceil(max(size(Tgc))/step2)+1,1);
%for i = 1:step2:max(size(Tgc))
%   count=count+1;
%   v2plot(count,:)=vgc(i,:);
%   T2plot(count,1)=Tgc(i);
%end

% This calculates the vectors at the final time.
%v2plot(max(size(T2plot)),:)=vgc(max(size(Tgc)),:);
%T2plot(max(size(T2plot)),1)=Tgc(max(size(Tgc)),1);
vplot(max(size(Tplot)),:)=v(max(size(T)),:);
Tplot(max(size(Tplot)),1)=T(max(size(T)),1);
%error(max(size(Tplot)),1)=lerr(vplot,XQ3pt,Node,Xpt,h,max(size(Tplot)),EN,Tplot);
%count2=0;
%for j=1:EN+1
%  for jj=1:5
%    fem=0;
%    count2=count2+1;
%    eplot(max(size(Tplot)),count2)=solution(xxx(count2,1),T(max(size(T))));
%    fem=vplot(max(size(Tplot)),j)*basis(j-1,xxx(count2,1),Xpt,h)+...
%       vplot(max(size(Tplot)),j+1)*basis(j,xxx(count2,1),Xpt,h);
%    errplt(max(size(Tplot)),count2)=fem-eplot(max(size(Tplot)),count2);
%  end
%end
%eplot(max(size(Tplot)),5*EN+6)=solution(100,T(i));
%fem=vplot(max(size(Tplot)),EN+2);
%errplt(max(size(Tplot)),5*EN+6)=fem-eplot(max(size(Tplot)),5*EN+6);

figure(1)
%subplot(1,2,1)
mesh(Xpt, Tplot, vplot)
%view(45,35)
view(25,25)
%view(75,35)
%view(25,10)
```

```
%COLORMAP([1 1 1])
%  Example one and six
%axis([0,100,0,tfinal,-.4,.4])
%  Example two and seven
axis([0,100,0,tfinal,-1,1])
%  Example three
%axis([0,1,0,10,-1.5,1.5])
%  Example four
%axis([0,100,0,tfinal,-.5,.5])
%  Example five
%axis([0,100,0,tfinal,0,.4])
grid
   xlabel('x')
   ylabel('t')
   zlabel('w(t,x)')
   title('Numerical Solution')
%figure(2)
%subplot(1,2,2)
%mesh(Xpt, T2plot, v2plot)
%view(75,35)
%view(25,10)
%COLORMAP([1 1 1])
%axis([0,100,0,tfinal,-.4,.4])
%grid
%   xlabel('x')
%   ylabel('t')
%   zlabel('w(t,x)')
%   title('Numerical Solution')
% Plot the exact solution.
%figure(3)
%mesh(xxx, Tplot, eplot)
%view(75,35)
%grid
%COLORMAP([1 1 1])
%axis([0,100,0,tfinal,-1,1])
%   xlabel('x')
%   ylabel('t')
%   zlabel('w(t,x)')
%   title('Exact Solution')
% Plot the error.
%figure (3)
%mesh(xxx, Tplot, errplt)
%view(75,35)
```

```
%grid
%  Example one
%axis([0,100,0,tfinal,-.1,.1])
%  Example two
%axis([0,100,0,tfinal,-.02,.02])
%   xlabel('x')
%   ylabel('t')
%   zlabel('wN(t,x) - w(t,x)')
%   title('Error')
```

GEOMETRY.M

```
%This script sets up the necessary arrays that allow the main program
%to compute results for a specific problem.  Contained herein is the
%correspondence between the actual layout of the problem and the
%layout of the functions used in the main program.
%=================================================================

% a, b are the endpoints.  EN the number of internal nodes.
a=0;
b=100;
EN=16;

% step-size if uniform.
unifh=(1/(EN+1))*(b-a)
numint=(b-a)/unifh;
h=0;
for i=1:numint, h(i)=unifh;, end
% step-size if not uniform.
%h=[.1 .2 .3 .4];

% Total number of nodes.
numnode=max(size(h))+1;

% Set up the x-coordinate of each node.
Xpt=zeros(numnode,1);
Xpt(1) = a;
for i=2:numnode, Xpt(i)=Xpt(i-1)+h(i-1);, end

% Set up Index, which describes the number of unknowns at each global unknown.
Index=zeros(numnode,1);
for i=1:numnode
```

```
  Index(i)=i;
end

% Set up the 1-pt. quadrature points in each element.
XQpt=0;
for i=1:max(size(h)), XQpt(i,1)=Xpt(i)+h(i)/2;, end
%nquad=1;

% Set up the 2-pt. quadrature points.
XQSpt=0;
for i=1:max(size(XQpt)),
    XQSpt(i,1)=XQpt(i) - .5773502692*h(i)/2;
    XQSpt(i,2)=XQpt(i) + .5773502692*h(i)/2;
end

% Set up the 3-pt. quadrature points.
XQ3pt=0;
for i=1:max(size(XQpt)),
    XQ3pt(i,1)=XQpt(i) - .7745966692*h(i)/2;
    XQ3pt(i,2)=XQpt(i);
    XQ3pt(i,3)=XQpt(i) + .7745966692*h(i)/2;
end

% Set up the relation between the local node for the basis
%functions and the global nodes.
Node=zeros(max(size(h)),2);
for i=1:max(size(h)), Node(i,1)=i;,end
for i=1:max(size(h)), Node(i,2)=i+1;, end
nlocal=2;
```

BASIS.M

```
function y = basis(z,x,Xpt,h)

% This function gives the value of the basis function at a given node.
%z is which basis function it is and x is the value at which to evaluate.

temp1=0;
%---------Fix it for z=0-----------
if z==0
   if temp1==0
          if x<Xpt(2)
```

```
        y=(Xpt(2)-x)/(h(1));
        temp1 =1;
        end
  end
  if temp1==0
        y=0;
        temp1=1;
  end
end
%---------Fix it for z=max(size(Xpt))--------
if z==max(size(Xpt))-1
  if temp1==0
        if x<Xpt(z)
        y=0;
        temp1=1;
        end
  end
  if temp1==0
        y=(x-Xpt(z))/h(z);
        temp1 =1;
  end
end
%----------------------------------
if temp1==0
        if x<Xpt(z)
        y=0;
        temp1 =1;
        end
end
if temp1==0
        if x<Xpt(z+1)
        y=(x-Xpt(z))/h(z);
        temp1 =1;
        end
end
if temp1==0
        if x<Xpt(z+2)
        y=(Xpt(z+2)-x)/(h(z+1));
        temp1 =1;
        end
end
if temp1==0
        if x<=1
```

```
            y=0;
            temp1 =1;
            end
end
```

BASISD.M

```
function y = basisd(z,x,Xpt,h)

% This function gives the value of the derivative of the basis function at a given node.
%z is which basis function it is and x is the value at which to evaluate.

temp1=0;
%---------Fix it for z=0-----------
if z==0
   if temp1==0
            if x<Xpt(2)
            y=-1/(h(1));
            temp1 =1;
            end
   end
   if temp1==0
            y=0;
            temp1=1;
   end
end
%---------Fix it for z=max(size(Xpt))--------
if z==max(size(Xpt))-1
   if temp1==0
            if x<Xpt(z)
            y=0;
            temp1=1;
            end
   end
   if temp1==0
            y=1/h(z);
            temp1 =1;
   end
end
%----------------------------------
if temp1==0
            if x<Xpt(z)
```

```
            y=0;
            temp1 =1;
            end
end
if temp1==0
            if x<Xpt(z+1)
            y=1/h(z);
            temp1 =1;
            end
end
if temp1==0
            if x<Xpt(z+2)
            y=-1/(h(z+1));
            temp1 =1;
            end
end
if temp1==0
            if x<=1
            y=0;
            temp1 =1;
            end
end
```

MATRICES.M

```
% This is the subroutine where the mass(MM) and stiffness(KK) matrices are assembled.
% Build F(v), the inhomogeneous term matrix.

numunk=0;
for i=1:max(size(Index))
            if Index(i)>0,numunk=numunk+1;,end
end
MM=zeros(numunk,numunk);
KK=zeros(numunk,numunk);
FF=zeros(numunk,2);
%FF=zeros(numunk,6); % for Example 2
V=zeros(numunk,1);
for nel=1:max(size(h))
  for quad=1:2
%    quad=1;
    for j=1:nlocal
      if Index(Node(nel,j))>0
```

```
      %integrate basis functions
      x=XQSpt(nel,quad);
      nnn=Index(Node(nel,j));
      KK(nnn,nnn)=KK(nnn,nnn)-(basisd(nnn-1,x,Xpt,h)^2)*h(nel)/2;
      MM(nnn,nnn)=MM(nnn,nnn)+(basis(nnn-1,x,Xpt,h)^2)*h(nel)/2;
      if nnn<max(size(MM))
       KK(nnn,nnn+1)=KK(nnn,nnn+1)-basisd(nnn-1,x,Xpt,h)*basisd(nnn,x,Xpt,h)*h(nel)/2;
       KK(nnn+1,nnn)=KK(nnn,nnn+1);
       MM(nnn,nnn+1)=MM(nnn,nnn+1)+basis(nnn-1,x,Xpt,h)*basis(nnn,x,Xpt,h)*h(nel)/2;
       MM(nnn+1,nnn)=MM(nnn,nnn+1);
      end
      %integrate basis functions against f(x)
%  Example one
%     FF(nnn,1)=FF(nnn,1)-(beta*(quadA*x*x+quadB*x+quadC+(sin(.01*pi*x))^2)+ ...
%      epsilon*(2*quadA+.0002*pi*pi*((cos(.01*pi*x))^2 - ...
%      (sin(.01*pi*x))^2)))*basis(nnn-1,x,Xpt,h)*h(nel)/2;
%     FF(nnn,2)=FF(nnn,2)+(quadA*x*x+quadB*x+quadC+(sin(.01*pi*x))^2)* ...
%      (2*quadA*x+quadB+.02*pi*sin(.01*pi*x)*cos(.01*pi*x))* ...
%      basis(nnn-1,x,Xpt,h)*h(nel)/2;
%  Example two
%     FF(nnn,1)=FF(nnn,1)+.0001*alpha*((100-x)^2)*basis(nnn-1,x,Xpt,h)*h(nel)/2;
%     FF(nnn,2)=FF(nnn,2)-.0001*beta*(x^2)*basis(nnn-1,x,Xpt,h)*h(nel)/2;
%     FF(nnn,3)=FF(nnn,3)-(2/(10^8))*((100-x)^3)*basis(nnn-1,x,Xpt,h)*h(nel)/2;
%     FF(nnn,4)=FF(nnn,4)+(2/(10^8))*(x^3)*basis(nnn-1,x,Xpt,h)*h(nel)/2;
%     FF(nnn,5)=FF(nnn,5)+(4/(10^8))*(x^3-150*x*x+5000*x)*basis(nnn-1,x,Xpt,h)*h(nel)/2;
%     FF(nnn,6)=FF(nnn,6)-.0002*epsilon*basis(nnn-1,x,Xpt,h)*h(nel)/2;

      %integrate basis functions against the initial function
%  Example one
%     V(nnn,1)=V(nnn,1)+(quadA*x*x+quadB*x+quadC+(sin(.01*pi*x))^2)* ...
%      basis(nnn-1,x,Xpt,h)*h(nel)/2;
%  Example two
%     V(nnn,1)=V(nnn,1)+.0001*(x^2)*basis(nnn-1,x,Xpt,h)*h(nel)/2;
%  Example three
%     V(nnn,1)=V(nnn,1)+.5cos(pi*x)*basis(nnn-1,x,Xpt,h)*h(nel)/2;
%  Example four
%     V(nnn,1)=V(nnn,1)+.5*cos(.01*pi*x)*basis(nnn-1,x,Xpt,h)*h(nel)/2;
%  Example five, six, seven
      V(nnn,1)=V(nnn,1)+cos(.01*pi*x)*basis(nnn-1,x,Xpt,h)*h(nel)/2;
      end
    end
  end
end
```

```
% Adjust for the Robin's BCs.
KK(1,1)=-1/h(1) - kap1/(kappa*L1);
KK(numunk,numunk)=-1/h(numunk-1) - kap2/(kappa*L2);
G=zeros(numunk,2);
G(1,1)=kap1/(kappa*L1);
G(numunk,2)=kap2/(kappa*L2);

M=inv(MM);
A=M*KK;
B=M*G;
F=M*FF;
```

BURGRHS.M

```
function vdot = burgrhs(t,y)

% This function gives the right hand side for the Galerkin method.
global A B F L1 L2 kap1 kap2 kappa alpha beta M
global epsilon quadA quadB quadC

u=zeros(2,1);
%  Example two
%u(1,1)=(1+.02*kappa*L1/kap1)*sin(alpha*t);
%u(2,1)=(1+.02*kappa*L2/kap2)*cos(beta*t);

%H=zeros(2,1);
%  Example one
%H(1,1)=exp(-beta*t);
%H(2,1)=exp(-2*beta*t);

%  Example two
%H=zeros(6,1);
%H(1,1)=cos(alpha*t);
%H(2,1)=sin(beta*t);
%H(3,1)=(sin(alpha*t))^2;
%H(4,1)=(cos(beta*t))^2;
%H(5,1)=sin(alpha*t)*cos(beta*t);
%H(6,1)=sin(alpha*t) + cos(beta*t);

ll=max(size(y));
J=zeros(ll,1);
J(1,1)=-2*y(1)^2 +y(1)*y(2) + y(2)^2;
```

```
for i=2:ll-1
   J(i,1)=-y(i-1)^2 -y(i-1)*y(i) + y(i)*y(i+1) + y(i+1)^2;
end
J(ll,1)=-y(ll-1)^2 -y(ll-1)*y(ll)+2*y(ll)^2;
J=(1/6)*J;

vdot = epsilon*(A*y +B*u) -M*J ; %+F*H;
```

CONRHS.M

```
function vdot = conrhs(t,y)

% This function gives the right hand side for the G/C method.
global A B F L1 L2 kap1 kap2 kappa alpha beta M
global epsilon quadA quadB quadC

u=zeros(2,1);
%  Example two
%u(1,1)=(1+.02*kappa*L1/kap1)*sin(alpha*t);
%u(2,1)=(1+.02*kappa*L2/kap2)*cos(beta*t);

%H=zeros(2,1);
%  Example one
%H(1,1)=exp(-beta*t);
%H(2,1)=exp(-2*beta*t);

%  Example two
%H=zeros(6,1);
%H(1,1)=cos(alpha*t);
%H(2,1)=sin(beta*t);
%H(3,1)=(sin(alpha*t))^2;
%H(4,1)=(cos(beta*t))^2;
%H(5,1)=sin(alpha*t)*cos(beta*t);
%H(6,1)=sin(alpha*t) + cos(beta*t);

ll=max(size(y));
J=zeros(ll,1);
J(1,1)=.25*(y(2)^2 - y(1)^2);
for i=2:ll-1
   J(i,1)=.25*(y(i+1)^2 - y(i-1)^2);
end
J(ll,1)=.25*(y(ll)^2 - y(ll-1)^2);
```

```
vdot = epsilon*(A*y + B*u) -M*J; %  +F*H;
```

SOLUTION.M

```
function y = solution(x,t)

% Set up the exact solution function.

global epsilon alpha beta quadA quadB quadC

%  Example one
y = exp(-beta*t)*(quadA*x*x + quadB*x + quadC + (sin(.01*pi*x))^2);

%  Example two
%y= .0001*((100-x)^2)*sin(alpha*t) + .0001*(x^2)*cos(beta*t);

%y = .25*cos(t)*cos(pi*x);

%y = .25*exp(-epsilon*t)*cos(pi*x);

%y = .25*exp(-epsilon*t)*(x*x+x-1);

%y = exp(-epsilon*t)*(quadA*x*x + quadB*x + quadC);
```

LERR.M

```
function y = lerr(vplot,XQ3pt,Node,Xpt,h,count,EN,Tplot)

% This function computes the L2 error in our finite element approximation.

err=0;
for nel=1:EN+1
   sol=zeros(3,1);
   fem=zeros(3,1);
   for quad=1:3
     x=XQ3pt(nel,quad);
     for j=1:2  %nlocal
       %integrate the difference w-wh
       nnn=Node(nel,j);
       fem(quad,1)=fem(quad,1)+vplot(count,nnn)*basis(nnn-1,x,Xpt,h);
     end
```

```
    sol(quad,1)=solution(x,Tplot(count));
  end
  err=err+.55555556*((sol(3,1)-fem(3,1))^2+(sol(1,1)-fem(1,1))^2)*h(nel)/2 ...
     + .88888889*((sol(2,1)-fem(2,1))^2)*h(nel)/2;
end
y=(err)^.5;
```

BIBLIOGRAPHY

1. Pugh, S.M., “Finite Element Approximations of Burgers’ Equation”, Virginia Polytechnic Institute and State University, (1995).

2. Burns, J.A., Balogh, A., Gilliam, D.S., and Shubov, V.I., “Numerical Stationary Solutions for a Viscous Burgers’ Equation”, Journal of Mathematical Systems, Estimation, and Control, to appear.

3. Fourier, J. *Théorie Analytique de la Chaleur*. Paris, Didot, 1822.

4. Berg, P.W., and McGregor, J.L., *Elementary Partial Differential Equations*, Holden-Day, Oakland, California, (1966).

5. Sears, F.W., Zemansky, M.W., and Young, H.D., *University Physics*, Addison-Wesley, Reading, Massachusetts, (1988).

6. Schenck, Hilbert, Jr., *Heat Transfer Engineering*, Prentice-Hall, Englewood Cliffs, NJ, (1959).

7. Kern, D.Q., *Process Heat Transfer*, McGraw-Hill, New York, NY, (1950).

8. Incropera, F.P., and DeWitt, D.P., *Fundamentals of Heat Transfer*, John Wiley & Sons, New York, New York, (1981).

9. Widder, D.V., *The Heat Equation*, Academic Press, New York, New York, (1975).

10. Bateman, H., “Some Recent researches on the motion of fluids,” Mon. Weather Rev., 43, (1915).

11. Burgers, J.M., “Mathematical Examples Illustrating Relations Occurring in the Theory of Turbulent Fluid Motion,” Trans. Roy. Neth. Acad. Sci., Amsterdam, 17, (1939), 1-53.

12. Fletcher, C.A.J., “Burgers’ Equation: A Model for All Reasons”, *Numerical Solutions of Partial Differential Equations*, J. Noye (Editor), North-Holland Pub. Co. (1982).

13. Hopf, E., “The Partial Differential Equation $u_t + uu_x = \mu u_{xx}$.”, Comm. Pure & Applied Math, 3, (1950), 201-230.

14. Cole, J.D., “On a Quasi-linear Parabolic Equation Occurring in Aerodynamics,” Quart. Appl. Math., 9, (1951), 225-236.

15. Byrnes, C.L., Gilliam, D.S., and Shubov, V.I., “Boundary Control, Feedback Stabilization, and the Existence of Attractors for a Viscous Burgers’ Equation,” (1990).

16. Burns, J.A., and Kang, S., *A control problem for Burgers' equation with bounded input/output*, Institute for Computer Applications for Science and Engineering, ICASE Report 90-45, (1990), NASA Langley Research Center, Hampton, VA.

17. Burns, J.A., and Kang, S., *A stabilization problem for Burgers' equation with unbounded control and observation*, International Series of Numerical Mathematics, Vol. 100, Birkhauser Verlag Basel, (1991).

18. Burden, R.L., Faires, J.D., *Numerical Analysis*, PWS Pub. Co., Boston, Massachusetts, (1993).

19. Marrekchi, H., "Dynamic Compensators for A Nonlinear Conservation Law," Virginia Tech, (1993).

20. Massa, K.L., "Computational Analysis and Control of Burgers' Equation," M.S. Thesis, Virginia Tech, (1997).

21. Fletcher, C.A.J., *Computational Galerkin Methods*, Springer-Verlag, New York, New York, (1984).