# Numerical Methods for Diffusive-Convective Equations with Applications to the Flow of Bacteria in Waterways

by

Jeffrey T. Crabill

A Thesis
Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Science in Mathematics

Northern Arizona University
August 1998

Approved:

[illegible]
John M. Neuberger, Ph.D., Chair

[illegible]
Terence R. Blows, Ph.D.

[illegible]
Catherine A. Roberts, Ph.D.

# ABSTRACT

## NUMERICAL METHODS FOR DIFFUSIVE-CONVECTIVE EQUATIONS WITH APPLICATIONS TO THE FLOW OF BACTERIA IN WATERWAYS

**Jeffrey T. Crabill**

Water quality is one of the more important health issues facing the human race. Water resources must be kept clean and pure so that a clean drinking water supply can be insured for years to come. Pollutants in water create an unpleasant environment. A team of microbiologists at Northern Arizona University sampled Oak Creek Canyon, AZ, for fecal coliform bacteria. This group of organisms serves as an indicator for more pathogenic organisms.

Diffusive-convective partial differential equations can be used to model the flow of bacteria through waterways such as Oak Creek Canyon, AZ. In this thesis, we derive a family of mathematical models using diffusive-convective equations. We present algorithms for approximating solutions to these equations as well as examine the usefulness and effectiveness of the equations in modeling the situation in Oak Creek Canyon. We examine the effect of initial conditions, boundary conditions, source terms, and differing magnitudes of diffusion and convection on the solution of the diffusive-convective equations.

Finally, we present various mathematical models that simulate the major features of the contamination data. We seek to find solutions whose graphs closely resemble the actual data and hope to gain insight about the processes that govern the contaminants in Oak Creek Canyon. In general, we find models that simulate the major features of the Oak Creek data relatively well and although the numbers generated are not exactly those collected in the field, the shape of the solution is similar to the actual data. Further study can be done to improve both the values given by the mathematical models as well as the collection of the actual data in the field.

# ACKNOWLEDGEMENTS

I would most like to thank my Thesis Advisor, Dr. John M. Neuberger, for his never-ending help and support since the undertaking of this project. As a brand new professor in the department, he took me under his wing and never floundered in his support. He has taught me much about partial differential equations in addition to having fostered my mathematical intuition. I sincerely hope that future students of mathematics will take the opportunity to work with him on theses and research projects.

Many thanks as well to Dr. Terence Blows, Dr. Catherine Roberts, Dr. Ann Cleveland, and Christine Crabill for their helpful comments on both the content and the mechanics of the manuscript. Their efforts, as well as those of my advisor, have helped me to greatly improve my mathematical writing skills.

Special thanks to my family who have supported me in every possible way since my first day of kindergarten. Without their support, I would not have been able to achieve this and every other milestone in my life.

Finally, I must thank my lovely wife, Christine Crabill, who not only inspired this work with her research on fecal coliform bacteria in Oak Creek, AZ, but who has unselfishly let me dominate the computer everyday for months, while she did all the housework and gave me hours of uninterrupted work. Thanks Christine, I owe you one!

# LIST OF TABLES

# LIST OF FIGURES

# INTRODUCTION

Water quality is an issue that affects people throughout the world. Clean water is necessary to support life, and therefore it is essential that we maintain a clean and pure water supply for future generations. As such, it is necessary to monitor water quality frequently in order to detect signs of potential problems. Waterways, such as Oak Creek Canyon in Arizona, attract use because of their recreational and aesthetic values. These waters must conform to strict government standards to maintain a level of safety for all who use the water.

There are many potential pathogens, such as giardia and polio virus, that can be found in natural waters. The presence of these pathogens can be indicated by the presence of fecal coliform bacteria. Fecal coliforms serve as "indicator organisms," i.e., organisms that are abundant when the pathogen is abundant and that are absent when pathogens are not present. The goal of a study conducted by microbiologists at Northern Arizona University was to monitor the fecal coliform levels in Oak Creek Canyon from Pine Flats Campground (upstream) to Grasshopper Point (downstream) [C].

It is our goal to simulate the features of the Oak Creek water quality data (Figure 1) with mathematical models; we make no claim, however, that our numbers are realistic, e.g., of appropriate scale. Also, it is difficult to compare our numbers to those in the Oak Creek study because the actual data is discrete and widely and unevenly spaced throughout the study area. Despite these difficulties, our final best model has a solution whose graph portrays the major features of the actual data. We hope that the model will provide insight into the processes that govern the presence of fecal coliforms in the creek. The effects of various amounts of diffusion and convection, types of source terms, boundary values, and initial values on the graphs of the solutions will be shown and explained. Once quantified, scientists may be able to better understand and predict levels of fecal contamination in Oak Creek with more success.

The processes mentioned above have an effect on the distribution of the contamination in waterways, such as Oak Creek. This distribution will be approximated by the solution of a diffusive-convection equation. Consider a material suspended in a fluid medium. Diffusion of the material is the movement down the concentration gradient and convection (referred to as advection in some texts) is the movement of the material due to the flow of the medium. The source terms will indicate when and where contamination is added or removed from the system. Finally, boundary values and initial conditions govern the behavior of the solution at the spatial limits of the study area (upstream and downstream) and at the beginning of a given year.

We will make use of diffusive-convective partial differential equations (PDEs) in our modeling applications. These equations have been shown in the literature to be appropriate for modeling water quality and bacterial movement [J]; [S-I]. Experimental data on the dispersal rate of a specific fecal coliform, *Escherichia coli*, have been collected in a laboratory but may not be suitable to model the natural situation. To the best of our knowledge, data on the dispersal rate of fecal coliforms have not been collected in the field. Despite this, one can experiment with various choices of amounts of diffusion and convection, types of boundary conditions, and source terms to develop suitable models. Each experimental result can then be compared to features present in the Oak Creek data.

In Chapter One of this work, we will describe, mathematically, a family of diffusive-convective PDEs and consider the solution to a simple special case. The simple case is considered to introduce and exhibit some of the properties of the solutions to our family of PDEs. In more complex cases, we will consider the effects of both diffusion and convection, as well as three types of boundary conditions and the addition of forcing terms. The existence, uniqueness, and continuous dependence on parameters of solutions to such PDEs will be considered in Chapter Two. With a well-posed PDE in mind, we present three numerical techniques in Chapter Three which can be used to approximate solutions. Graphing and comparing these approximate solutions to available Oak Creek fecal contamination data will guide us in our selection of forcing terms, boundary conditions, and magnitudes of diffusion and convection, all of which are the focus of Chapter Four. Future research directions permitting refinements to our models will be discussed in Chapter Five as well as other methods for approximating solutions of diffusive-convective PDEs.

# CHAPTER ONE

# THE GENERAL DIFFUSIVE-CONVECTIVE EQUATION

In this chapter, we present and describe a family of diffusive-convective PDEs. We shall use these equations in our modeling applications in Chapter Four. In the modeling application, we shall consider the domain, $\Omega = (0,1) \times (0,1)$, where time is the first coordinate and space is the second coordinate. For our purposes, time $t = 0$ represents any arbitrary point in time with time $t = 1$ representing exactly one year later. Similarly, $x = 0$ represents an arbitrary point in space and $x = 1$ represents an arbitrary point downstream of the initial point.

Let $u$ be a real-valued function of $t$ (time) and $x$ (space). We consider the general one-dimensional diffusive-convective equation

$$u_t(t,x) = \alpha^2 u_{xx}(t,x) - \beta u_x(t,x) + f(t,x) \qquad \alpha > 0,\ \beta \geq 0,\ (t,x) \in \Omega \tag{1.1}$$

with initial condition

$$u(0,x) = g(x) \qquad x \in [0,1] \tag{1.2}$$

and boundary conditions

$$u(t,0) = 0 \qquad t \in [0,1] \tag{1.3}$$
$$au(t,1) + (1-a)\,u_x(t,1) = 0 \qquad a \in [0,1],\ t \in [0,1], \tag{1.4}$$

where the coefficients $\alpha^2$ and $\beta$ give the magnitude of the diffusion and convection, respectively, and the function $f(t,x)$ is a source term. For our purposes, we shall consider Zero Dirichlet conditions at $x = 0$, i.e., equation (1.3). We assume equation (1.3) for the modeling application, but the methods to be described in the thesis can be modified to take into account other boundary conditions at $x = 0$. We say that a Zero Dirichlet condition exists at $x = 1$ if $a = 1$, and we say that a Zero Neumann condition exists if $a = 0$. Otherwise, for $a \in (0,1)$, we say that a Robin condition exists. Throughout the paper, in particular Chapter Four, we will discuss the appropriateness of each of the above mentioned boundary conditions, as well as the "best" choice of boundary conditions for our modeling application.

We first consider the simplest case of equation (1.1) where a closed-form solution is easily obtainable. Several numerical approximations will be compared with the actual solution to demonstrate the capabilities of the numerical algorithms presented in Chapter Three. This simple experiment will also serve to introduce the diffusive nature of equation (1.1). Let $a = 1$, $\alpha = 0.5$, $\beta = 0$, $f(t,x) \equiv 0$ on $\Omega$, and $g(x) = \sin(\pi x)$. (The value of $\alpha$ was chosen for comparison purposes later in the chapter.) This is a simple case of the Heat Equation and models pure diffusion, i.e., the movement of a substance down a concentration gradient. It can also be viewed as a model of the distribution of heat within a rod over a given period of time [Z]. If the ends of the rod are kept at a constant temperature (say

0° C), then a Zero Dirichlet conditions exists at both ends of the rod. Using basic calculus, it follows that the Heat Equation, with Zero Dirichlet boundary conditions has solution

$$u(t,x) = e^{-0.5\pi^2 t}\sin(\pi x). \tag{1.5}$$

The graph of equation (1.5), shown in Figure 2, shows the diffusive nature of $u(t,x)$ on $\Omega$. When $t = 0$, there is an initial "temperature," which decreases as $t \to 1$. As $t$ increases without bound, the solution approaches the zero function. This is just as one would suspect since the temperature of the rod would decrease over time as heat escapes the ends of the rod.

We could also observe the diffusive nature of the Heat Equation in the case of a one-dimensional "lake," in which material does not enter or leave, i.e., a closed system. In this situation, Zero Neumann boundary conditions are appropriate at both ends of the lake, since no inflow or outflow is assumed. If there were some initial pollution distribution in the lake, then we could make use of the Heat Equation in the same manner as above. Instead, we shall consider the solution to the Heat Equation with mixed boundary conditions, i.e., Dirichlet at $x = 0$ and Neumann at $x = 1$, since we are interested in systems such as Oak Creek where these mixed boundary conditions are more appropriate. In this case, the solution, $u(t,x)$ (Figure 3) would represent the amount of pollution at the point $(t,x) \in \Omega$. The level of contamination near the boundary at $x = 1$ is nearly constant and the contaminant is distributed quite differently than in the Dirichlet case. The material has spread out, rather than having been removed at $x = 1$.

Convection is the mass transport of material caused by the flow of the medium in which the material is suspended. Allowing $\beta > 0$ changes equation (1.1) to a diffusive-convective equation. The additional term will have the effect of pushing the material forward in space. The amount of material is reduced at points in $\Omega$ where $u_x(t,x) > 0$ (upstream in the case of Oak Creek) and the material is increased where $u_x(t,x) < 0$ (downstream in the case of Oak Creek). Hence, we subtract the convection term to reflect this situation. Let us consider the following simple diffusive-convective equation:

$$u_t(t,x) = 0.5u_{xx}(t,x) - u_x(t,x), \tag{1.6}$$

with Zero Dirichlet conditions and the initial function $g(x) = \sin(\pi x)$. The effects of the convection term can be seen in the skew of the solution surface shown in Figure 4. The solution without convection was shown in Figure 2. Comparing these two surfaces gives a clear view of the effect of convection on the solution surface. There is a distinct pull toward $x = 1$ in the example with convection.

# CHAPTER TWO

# WELL-POSEDNESS OF THE PROBLEM

In this chapter, we consider the well-posedness of the family of diffusive-convective equations that we will use in our modeling application. This involves both the existence and uniqueness of solutions in addition to continuous dependence on initial data, boundary data, parameters and source terms. First, we establish the existence and uniqueness of the solution, then we will give justification for the continuous dependence on the data.

Such well-posed problems are necessary if we are to have any faith in the results of our numerical algorithms. For the PDEs of interest in our modeling application, it may be difficult or impossible to find a closed-form solution. Also, it is possible that future models, not included in this thesis, will include nonlinear terms in the source term of equation (1.1). In many cases, closed-form solutions do not exist and we shall therefore use numerical approximations of the solutions to the PDEs. (Note that the methods of this thesis work in cases where closed-form solutions do exist.)

We will show that a simple case of equation (1.1) has a unique solution and then generalize that result to other cases of equation (1.1).

In this chapter, we present an adaptation of a proof by Berg and McGregor [B] in which the existence and uniqueness of the solution to the Heat Equation is exhibited. We first state the following three lemmas. Proofs of these facts can be found in the text [B]. The first lemma gives eigenvalues and eigenfunctions for a simple ordinary differential equation.

**Lemma 2.1** *Let $\varphi$ be a function of x. For $x \in (0,1)$, the ordinary differential equation*

$$\varphi_{xx} + \lambda\varphi = 0$$

*with boundary conditions $\lim_{x\to 0^+} \varphi(x) = \lim_{x\to 1^-} \varphi(x) = 0$ has eigenvalues $\lambda_n = (n\pi)^2$ and eigenfunctions $\varphi_n = \sin(n\pi x)$, for each natural number, $n$.*

Next, we change pace a bit and exhibit the familiar Fourier series representation of a function.

**Lemma 2.2** *If $p$ and $p'$ are continuous functions in $x \in [0,1]$ such that $p(0) = p(1) = 0$, then*

$$p(x) = \sum_{n=1}^{\infty} b_n \sin(n\pi x)$$

*where*

$$b_n = 2\int_0^1 p(x)\sin(n\pi x).$$

Finally, we present a technical lemma that will be important in the proof the major result of this chapter.

***Lemma* 2.3** *If $\varphi$, $\varphi'$, $\psi$, and $\psi'$ are continuous functions in $x \in [0,1]$, and $\varphi''$ and $\psi''$ are continuous functions in $x \in (0,1)$, then Green's Formula,*

$$\int_0^1 [\varphi\psi'' - \psi\varphi'']dx = [\varphi\psi' - \psi\varphi']_0^1,$$

*holds, where the integral on the left is improper. In addition, if either $\varphi''$ or $\psi''$ is continuous on $[0,1]$, then*

$$\int_0^1 \varphi\psi'' dx = \int_0^1 \psi\varphi'' dx.$$

The major result of existence and uniqueness of solutions can now be stated.

***Theorem* 2.4** *Suppose that there is a function $u(t,x)$ such that the following hold:*

*(i) $u$, $u_t$, $u_x$, and $u_{xx}$ are defined and continuous for $(t,x) \in \Omega$ and*

$$u_t(t,x) = u_{xx}(t,x) \qquad (t,x) \in \Omega \tag{2.1}$$

*and further, $u_t(t,x)$ is bounded when $x \in (0,1)$ and $t \in (t_1,t_2)$ for each $0 \le t_1 < t_2 \le 1$.*

*(ii) $\lim_{x\to0^+} u(t,x)$, $\lim_{x\to1^-} u(t,x)$, $\lim_{x\to0^+} u_x(t,x)$, and $\lim_{x\to1^-} u_x(t,x)$ exist for all $t \in (0,1)$.*

*(iii) For every piecewise continuous function $\psi$ on $[0,1]$, there exists a*

*piecewise continuous function,* $f(x)$*, such that*

$$\lim_{t \to 0^+} \int_0^1 u(t,x)\, \psi(x)\, dx = \int_0^1 f(x)\, \psi(x)\, dx. \tag{2.2}$$

*Then there is only one such function,* $u(t,x)$*, and it is given by*

$$u(t,x) = \sum_{n=1}^{\infty} b_n\, e^{-(n\pi)^2 t} \sin(n\pi x) \tag{2.3}$$

*where*

$$b_n = 2\int_0^1 f(x) \sin(n\pi x)\, dx, \tag{2.4}$$

$x \in (0,1)$*, and* $t \in (0,1)$*.*

Before giving the proof, we note that it can be seen that the function $f$ in (iii) is the initial condition $u(0,x)$.

*Proof*: Suppose that $u(t,x)$ is a solution to (2.1) and satisfies (i), (ii), and (iii). By assumption, for each $t > 0$, $u(t,x)$ and $u_x(t,x)$ are continuous functions on $\Omega$. Lemma 2.2 gives $u(t,x) = \sum_{n=1}^{\infty} B_n(t) \sin(n\pi x)$ with $x \in (0,1)$, where $B_n(t) = 2\int_0^1 u(t,x) \sin(n\pi x)\, dx$. Now $B_n(t)$ is continuous and differentiable, thus differentiation yields

$$\frac{d}{dt} B_n(t) = 2\int_0^1 u_t(t,x) \sin(n\pi x)\, dx. \tag{2.5}$$

From (2.1), we have

$$\frac{d}{dt}B_n(t) = 2\int_0^1 u_{xx}(t,x)\ \sin(n\pi x)\ dx. \tag{2.6}$$

Consequently, the functions $u(t,x)$ and $\sin(n\pi x)$ satisfy the hypotheses of Lemma 2.3, and hence

$$\begin{aligned} \frac{d}{dt}B_n(t) &= 2\int_0^1 u(t,x)\ \frac{d^2}{dx^2}\sin(n\pi x)\ dx \\ &= -2(n\pi)^2\int_0^1 u(t,x)\ \sin(n\pi x)\ dx. \end{aligned} \tag{2.7}$$

So, $B_n(t)$ satisfies the ordinary differential equation

$$B_n'(t) + (n\pi)^2 B_n(t) = 0 \tag{2.8}$$

which has solution

$$B_n(t) = b_n e^{-(n\pi)^2 t} \qquad t > 0 \tag{2.9}$$

where $b_n$ is a constant. Thus by substitution, we may obtain the following (unique) representation:

$$u(t,x) = \sum_{n=1}^{\infty} b_n e^{-(n\pi)^2 t}\ \sin(n\pi x) \qquad (t,x) \in \Omega. \tag{2.10}$$

Finally, since $\lim_{t\to 0^+} B_n(t) = b_n$, putting $\psi(x) = \sin(n\pi x)$ (see iii) gives

$$\begin{aligned} \lim_{t\to 0^+} B_n(t) &= \lim_{t\to 0^+} 2\int_0^1 u(t,x)\sin(n\pi x)\ dx \\ &= 2\int_0^1 f(x)\sin(n\pi x)\ dx. \end{aligned} \tag{2.11}$$

Thus $u(t,x)$ is given by (2.3) and (2.4) and hence exists for $(t,x) \in \Omega$. The uniqueness of the solution is immediate. Suppose that there are two solutions to

(2.1). Each, then, is equal to the infinite sum in (2.10) and are therefore equal to each other. ■

To complete the well-posedness argument in a more general setting, we note that our family of diffusive-convective PDEs depends continuously on the initial and boundary data [Y].

The above proof can be modified to account for other boundary conditions. The second hypothesis requires only that the various limits at the boundaries exist, therefore both Neumann and Robin conditions also would give unique solutions to the PDE. In addition, these problems are well-posed [Y].

For more complicated problems, such as those including convection and/or source terms, the literature tends to favor another approach, which we introduce here for completeness. This approach, suggested by Lieberman uses the notion of Hölder continuity [L]. Let $L$ be a linear operator defined by

$$Lu \equiv -u_t + a^{ij}D_{ij}u + b^iD_iu + cu,$$

where $a^{ij}$ is a matrix-valued function, $b^i$ is a vector-valued function, $c$ is a scalar-valued function, and $D_i$ and $D_{ij}$ are the first and second differential operators (with respect to space.) It can be shown that there is a unique function, $u$, satisfying

$$\begin{aligned} Lu &= f \\ u &= \phi \qquad on\ P\Omega \end{aligned}$$

where $P\Omega$ is the parabolic boundary of $\Omega$, if the function $f$ is Hölder continuous. In our Oak Creek setting, the parabolic boundary is a subset of $\Omega$ which consists of the lines $t = 0$, $t = 1$, and $x = 0$. For the purposes of this work, we shall only consider functions $f$ such that $f \in C^{\infty}$. It can then be shown that such $f$ are Hölder continuous [L].

Hence, any diffusive-convective PDE with appropriately defined boundary conditions has a unique solution on $\Omega$. [Y]

# CHAPTER THREE

# NUMERICAL METHODS

We shall now consider three different numerical methods used to solve equation (1.1). These methods are able to handle all of the types of initial and boundary conditions that we will be using in our modeling application, as well as more general terms that may involve nonlinear functions. We will develop each method and then apply it to a specific case of equation (1.1). Finally, we will compare the results of the methods, both to each other and against the known solutions (where available.)

In this chapter, we first consider numerical methods to solve the Heat Equation with Zero Dirichlet boundary conditions as given below.

$$\begin{aligned} u_t(t,x) &= u_{xx}(t,x) \qquad (t,x) \in \Omega \\ u(0,x) &= g(x) \qquad x \in [0,1] \\ u(t,0) &= 0 \qquad t \in (0,1) \\ u(t,1) &= 0 \qquad t \in (0,1). \end{aligned} \tag{3.1}$$

One should note that although the pair $(t,x) \in \Omega$, these procedures can also be extended to the domain $(0,T) \times (0,1)$ (See Section 4.4). We also define some useful notation for the forthcoming sections. Let the quantities $t_i$ and $x_j$ be $\frac{i}{M}$ and $\frac{j}{N}$, respectively, and define $u[i,j]= u(t_i,x_j)$.

## 3.1 The Explicit Method of Numerical Approximation

The Explicit Method is a well-known and well-studied method of approximation that uses a simple forward-difference approximation to the PDE to approximate a solution [B-F]. The algorithm is as follows:

Choose $\Delta t = \frac{1}{M}$ and $\Delta x = \frac{1}{N}$ such that $\frac{\Delta t}{(\Delta x)^2} \leq \frac{1}{2}$. The latter, well-known, condition is due to Courant, Friederics, and Levy ("C.F.L." condition). This is necessary for the convergence of the method [B-F]; [I]. Hence the Explicit Method is *conditionally stable*.

The set $\Omega$ is thus divided into a grid of $M \times N$ rectangles. The PDE (3.1) can be approximated by

$$\frac{u[i+1,j]-u[i,j]}{\Delta t} = \frac{u[i,j+1]-2u[i,j]+u[i,j-1]}{(\Delta x)^2}, \qquad (3.1.1)$$
$$\begin{aligned} for \quad & i \in \{0,1,\ldots,M-1\}, \\ & j \in \{1,2,\ldots,N-1\}. \end{aligned}$$

Note that the left hand side of equation (3.1.1) is a two-point forward time difference approximating $u_t$ and that the right hand side of equation (3.1.1) is a three-point central space difference that approximates $u_{xx}$ [B-F]. At any given time step, $i$, all quantities in equation (3.1.1) are known except for $u[i+1,j]$. Solving for $u[i+1,j]$ yields

$$u[i+1,j] = \frac{\Delta t}{(\Delta x)^2}\Big[u[i,j+1] - 2u[i,j] + u[i,j-1]\Big] + u[i,j], \tag{3.1.2}$$
$$\begin{aligned} for \quad i &\in \{0,1,\ldots,M-1\}, \\ j &\in \{1,2,\ldots,N-1\}. \end{aligned}$$

Equation (3.1.2) suggests an iterative method for solving the PDE explicitly. A program to implement this algorithm was written in C and can be found in Appendix 1. Let us first test the program on the problem given in equation (3.1) with the following definition of $g(x)$:

$$g(x) = \sin(\pi x). \tag{3.1.3}$$

The closed-form solution to this problem is given by

$$u(t,x) = e^{-\pi^2 t}\sin(\pi x). \tag{3.1.4}$$

Table 1 contains the results of executions of the above algorithm on this example for the following pairs of $M$ and $N$: (5000, 50), (20000, 100), and (2000000, 1000). The method provides very good approximations to the actual solution that improve as the number of time and space steps increase. For the purposes of this work, however, the pair (5000, 50) will be acceptable. In this simplest case, the error associated with this method with this choice of $M$ and $N$ is at most $3 \times 10^{-7}$; a value which hardly has any effect on the graph. As a consequence of the well-posedness of the problem, we are confident that similar, adequately small errors will be observed in the execution of the algorithm on more complex problems. In addition, other examples were run with finer time and space grids to determine if this would have a significant effect on the shape of the graphs. The graphs of the resultant numerical approximations were identical and the execution required more time. Therefore, we will continue to use the pair (5000, 50) since the graphs are of primary interest.

The iteration described in (3.1.2) can be modified to handle other boundary conditions, as well as the addition of forcing terms, convection terms, and others terms that may be necessary for a model. Let us now consider the more general problem stated in (1.1) through (1.4). This can be approximated by

$$\begin{aligned}\frac{u[i+1,j]-u[i,j]}{\Delta t} &= \alpha^2\frac{u[i,j+1]-2u[i,j]+u[i,j-1]}{(\Delta x)^2}\\ &\quad -\beta\frac{u[i,j+1]-u[i,j]}{\Delta x}+f(t_i,x_j)\\ &\quad \textit{for}\quad i\in\{0,1,\dots,M-1\},\\ &\quad\quad\quad j\in\{1,2,\dots,N-1\},\end{aligned}\tag{3.1.5}$$

which, after solving for $u[i+1,j]$, yields

$$\begin{aligned}u[i+1,j] &= \alpha^2\frac{\Delta t}{(\Delta x)^2}\Big[u[i,j+1]-2u[i,j]+u[i,j-1]\Big]+u[i,j]\\ &\quad -\beta\frac{\Delta t}{\Delta x}\Big[u[i,j+1]-u[i,j]\Big]+f(t_i,x_j)\,\Delta t\\ \textit{for}\quad & i\in\{0,1,\dots,M-1\},\\ & j\in\{1,2,\dots,N-1\}.\end{aligned}\tag{3.1.6}$$

Note that in (3.1.5), $u_x$ is approximated with a two-point forward difference [B-F].

For each $i$, the approximations for $u[i,N-2]$ and $u[i,N-1]$ are used to approximate $u[i,N]$ as per the Robin condition, in the following manner. For each $i\in\{1,2,\dots,M\}$, approximate (1.4) with

$$au[i,N]+(1-a)\frac{u[i,N-2]-4u[i,N-1]+3u[i,N]}{2\Delta x}=0.\tag{3.1.7}$$

Solving for $u[i,N]$ yields

$$u[i,N] = \frac{((a-1)u[i,N-2] + 4(1-a)u[i,N-1])}{2a\Delta x - 3a + 3}, \qquad (3.1.8)$$

for each $i$. Note that in (3.1.7), $u_x(i,N)$ is approximated with a three-point backwards difference [B-F]. To the best of our knowledge, this approach to approximating the solution near the boundary $x = 1$ is new, as we have not seen this method used in the literature that we have reviewed.

The algorithm for approximating the solution to a PDE of the general form (1.1) can then be implemented. For each $i$, apply (3.1.6) for $j \in \{1,2,...N-1\}$. Then, for each $i$, $u[i,N]$ can be approximated with (3.1.8). Refer to Appendix 1 for the computer code that implements this algorithm.

*Since this algorithm is easy to code and the parameters are easy to modify within the code, we shall use the Explicit Method for the modeling applications in Chapter* 4.

## 3.2 The Implicit Method of Numerical Approximation

Equation (3.1.1) is only one of several ways to approximate numerically the Dirichlet problem in equation (3.1). For completeness, we present a second way to approximate equation (3.1) that gives a system of linear equations in $N-1$ unknowns. We modify equation (3.1.1) by approximating the right hand side with future times rather than present times. The new approximation is

$$\frac{u[i+1,j] - u[i,j]}{\Delta t} = \frac{u[i+1,j-1] - 2u[i+1,j] + u[i+1,j+1]}{(\Delta x)^2} \qquad (3.2.1)$$
$$\begin{aligned} \textit{for} \quad & i \in \{0,1,...,M-1\}, \\ & j \in \{1,2,...,N-1\}. \end{aligned}$$

Letting $p = \frac{\Delta t}{(\Delta x)^2}$, equation (3.2.1) simplifies to

$$-pu[i+1,j-1] + (2p+1)u[i+1,j] - pu[i+1,j+1] = u[i,j], \quad (3.2.2)$$
$$for \quad i \in \{0,1,...,M-1\},$$
$$j \in \{1,2,...,N-1\},$$

which gives a system of $N-1$ linear equations. The resulting system can be written as

$$\mathbf{A}w = v \quad (3.2.3)$$

where

$$w = (u[i+1,j])_{j=1}^{N-1} \quad (3.2.4)$$
$$v = (u[i,j])_{j=1}^{N-1} \quad (3.2.5)$$

and

$$\mathbf{A} = \begin{bmatrix} 2p+1 & -p & 0 & .... & .... & 0 \\ -p & 2p+1 & -p & 0 & .... & .... \\ 0 & -p & 2p+1 & -p & .... & .... \\ .... & 0 & .... & .... & .... & 0 \\ .... & .... & .... & .... & .... & -p \\ 0 & .... & .... & 0 & -p & 2p+1 \end{bmatrix}.$$

Note that **A** is row equivalent to the identity matrix and therefore invertible. We now define some useful notation. Let $\mathbf{A}u[i+1,\bullet]$ represent matrix multiplication of **A** and the column vector $u[i+1,\bullet]= (u[i+1,j])_{j=1}^{N-1}$.

The algorithm is as follows: Choose $M$ and $N$. Initialize the vector $u$ with $u[0,j]= g(\frac{j}{N})$, where $u[0,j]$ is the $j^{th}$ component of $u$, for each $j \in \{1,2,...,N\}$.

Then, for each $i \in \{1,2,...,M\}$, solve system (3.2.3) with a tridiagonal solver (the computer code that implements the algorithm to solve a tridiagonal system can be found in Appendix 2). After each implementation of the tridiagonal solver, $u[i,\bullet]$ will be overwritten with $u[i+1,\bullet]$. The discretized approximation to the solution at $t = 1$ will be given in the final vector, $u[M,\bullet]$. The results of the execution of this algorithm on equation (3.1) with initial condition (3.1.3) can be found in Table 2. Refer to Appendix 3 for the computer code that implements the Implicit Method.

The Implicit Method described here can also be modified to involve other boundary conditions and forcing terms. For this paper, we shall use the Explicit Method for more complicated problems (e.g., problems with forcing terms) due to the extreme ease of the computer code for that method.

## 3.3 The Crank-Nicolson Method of Numerical Approximation

Again, for completeness, we present a third numerical approximation algorithm. If we change the approximation of the second space derivative slightly, we can modify the Implicit method to obtain the so-called Crank-Nicolson method. An average of the approximations to $u_{xx}$ found in equations (3.1.1) and (3.2.1) is used to obtain the following approximation to the Dirichlet problem in equation (3.1):

$$\frac{u[i+1,j]-u[i,j]}{\Delta t} = \frac{1}{2}\Big[\frac{u[i,j+1]-2u[i,j]+u[i,j-1]}{(\Delta x)^2} + \frac{u[i+1,j-1]-2u[i+1,j]+u[i+1,j+1]}{(\Delta x)^2}\Big], \tag{3.3.1}$$

$$for \quad i \in \{0,1,\ldots,M-1\},$$
$$j \in \{1,2,\ldots,N-1\}.$$

After solving for $u[i+1,j]$, (3.3.1) can be written in the matrix form

$$\widetilde{\mathbf{A}}w = \mathbf{B}v, \tag{3.3.2}$$

with $w$ and $v$ defined in equations (3.2.4) and (3.2.5), respectively. The matrices $\widetilde{\mathbf{A}}$ and **B** are given by

$$\widetilde{\mathbf{A}} = \begin{bmatrix} 1+p & \frac{-p}{2} & 0 & .... & .... & 0 \\ \frac{-p}{2} & 1+p & \frac{-p}{2} & 0 & .... & .... \\ 0 & \frac{-p}{2} & 1+p & \frac{-p}{2} & .... & .... \\ .... & 0 & .... & .... & .... & 0 \\ .... & .... & .... & .... & .... & \frac{-p}{2} \\ 0 & .... & .... & 0 & \frac{-p}{2} & 1+p \end{bmatrix} \tag{3.3.3}$$

and

$$\mathbf{B} = \begin{bmatrix} 1-p & \frac{p}{2} & 0 & .... & .... & 0 \\ \frac{p}{2} & 1-p & \frac{p}{2} & 0 & .... & .... \\ 0 & \frac{p}{2} & 1-p & \frac{p}{2} & .... & .... \\ .... & 0 & .... & .... & .... & 0 \\ .... & .... & .... & .... & .... & \frac{p}{2} \\ 0 & .... & .... & 0 & \frac{p}{2} & 1-p \end{bmatrix}. \tag{3.3.4}$$

Like the Implicit method, described in the previous section, the matrix **A** has tridiagonal form and we can use the same tridiagonal solver implemented previously to iterate the algorithm in equation (3.3.2). One must perform the matrix multiplication in the right hand side of equation (3.3.2) and then the process is identical to the Implicit method described above. The C code implementing this algorithm can be found in Appendix 4. Again, notice that $\widetilde{\mathbf{A}}$ is invertible.

## 3.4 Stability and Error of the Three Algorithms

To ensure accuracy for modeling purposes, we must be sure that the three iterative methods described above are stable and hence converge. There has been extensive study on the stability of the Explicit, Implicit, and Crank-Nicolson algorithms and we shall briefly outline the requirements for stability of the algorithms here [B-F]; [I].

As already mentioned in section 3.1, the Explicit Method must satisfy the "C.F.L" condition, i.e.,

$$\alpha^2 \frac{\Delta t}{(\Delta x)^2} \le \frac{1}{2}. \qquad (3.4.1)$$

This method is said to be *conditionally stable.* For a detailed explanation and proof, see [B-F] and [I], respectively. In fact, the methods are very similar to the following argument for the stability of the Implicit Method.

The Implicit Method has an important advantage over the Explicit Method. It is *unconditionally stable.* This could therefore require less execution time, since the values of $M$ and $N$ can be smaller than for the Explicit Method. We have found, however, that the Explicit Method with the pair (5000, 50) is generally quicker than the Implicit Method.

To give the reader a feel for a possible treatment of the issue of stability using numerical analysis, we present justification for the stability of the Implicit Method. For stability, the matrix **A** must be such that the spectral radius of its inverse, i.e., $\rho(\mathbf{A}^{-1})$, is bounded above by 1 [B-F]. Recall that for any square matrix **M**, $\rho(\mathbf{M}) = \max_i |\omega_i|$, where $\omega_i$ is an eigenvalue of **M**. The eigenvalues of **A** are

$$\lambda_i = 1 + 4p\left(\sin\left(\frac{i\pi}{2N}\right)\right)$$

for each $i \in \{1,2,\ldots,N-1\}$. Since $p > 0$, we have that each $\lambda_i > 1$. Define $\omega_i = \frac{1}{\lambda_i}$. The eigenvalues of $\mathbf{A}^{-1}$ are the $\omega_i$ and $\omega_i < 1$ for each $i \in \{1,2,\ldots,N-1\}$. Hence $\rho(\mathbf{A}^{-1}) < 1$, and the method is stable, independent of the values of $\Delta t$ and $\Delta x$. Using this method, we are no longer bound to the restrictions imposed on the Explicit Method by the "C.F.L" condition. Refer to [B-F] for a more detailed treatment of the stability of the Implicit Method.

Finally, it can be shown that the Crank-Nicolson Method is also unconditionally stable. The details may be found in Isaacson and Keller [I]. The results in Table 3 show the much improved accuracy of the Crank-Nicolson Method over the Implicit Method in the case of $M = N = 5000$; however, time of execution was significantly greater.

data. More complex observations may be incorporated in later models.

It can be argued that single-celled organisms have random or pseudo-random movement [E-K]. Edelstein-Keshet (1987) states, however, that the organisms could be modeled similarly to diffusion at the population level. One could then use equation (1.1) to model the dispersion of single-celled organisms, such as fecal coliform bacteria.

Diffusive equations have been used to model the dispersal of bacteria. A simple example is the application of the Heat Equation to the dispersion of *Pseudomonis fluorescens* by Segal *et al.* [S-II]. Segal calculated, from experimental observations, a dispersal rate of *P. fluorescens* of $0.2\ \mathrm{cm}^2\,\mathrm{h}^{-1}$. The dispersal rate for *E. coli* is $0.03\,\mathrm{cm}^2\,\mathrm{h}^{-1}$, suggesting that, in a waterway such as Oak Creek, the movement of the bacteria can be attributed to both convection and diffusion [S-II]. These numbers are mentioned solely to demonstrate that diffusion does have an effect on the presence of fecal coliforms in the creek. We will not, however, try to incorporate these values into our models. One must be cautious with the values and with the units attached to them. The solutions to the mathematical models in this work give numbers that need scaling before any units can be attached.

There are several other characteristics of the data that we will consider. These are (a) the peak of the contamination at the start and end of the summer season near Slide Rock State Park, (b) the periodic nature of the contamination in time, and (c) the outflow of contaminants beyond the lower boundary of the study area.

The inclusion of an appropriate *source term* is necessary to consider these factors. Source terms could also take into account natural phenomena such as spillage of waste, reproduction, and predation.

*Note that the solutions to all PDE models to follow were approximated using the Explicit Method with $M = 50$ and $N = 5,000$, given the fast execution time and the simplicity of the C code that implements this method.*

## 4.2 Simple Empirical Models

We begin with a stream that is free of contamination at the upstream and downstream boundaries and that has relatively little flow. In this case, we shall assume Zero Dirichlet boundary conditions both upstream and downstream and no convection. We consider a source term that is periodic in time and has a maximum value at $(0.5, 0.5) \in \Omega$, in order to satisfy factors (a), (b), and (c) above. Consider the function

$$f(t,x) = (1 - \cos(2\pi t))\, e^{(x-0.5)^2}. \tag{4.2.1}$$

as the source term. The data suggests that an initial condition reflecting no contamination at the first of the year is appropriate, so we will define

$$g(x) \equiv 0. \tag{4.2.2}$$

An approximation to the solution of

$$u_t = u_{xx} + f(t,x), \tag{4.2.3}$$

with Zero Dirichlet boundary conditions and initial function $g(x)$ is shown in Figure 5. Scaling of the contamination axis aside, this solution surface more closely resembles a smooth approximation to the data in Figure 1 than any of the graphs we have thus far obtained. If we then consider a running stream, the effects of convection can be seen in Figure 6, which shows an approximation to the solution of

$$u_t(t,x) = u_{xx}(t,x) - u_x(t,x) + f(t,x). \tag{4.2.4}$$

Thus far, our empirical modeling techniques have generated PDE models that are not unreasonable. The models (4.2.3) and (4.2.4) are the most reasonable so far. These models give solution surfaces that vanish at $t = 0$, peak at $(0.5, 0.5)$, and vanish at $t = 1$. Nonetheless, we can take the next step and investigate the effects of other boundary conditions as well as other source terms.

A (nearly) Zero Dirichlet boundary condition seems appropriate at $x = 0$ since the contamination upstream in Oak Creek has been observed to be very low [C]. We may, however, consider a Zero Neumann condition at $x = 1$. This is to assume that there is no change in contamination downstream. Numerical approximations of solutions to (4.2.3) and (4.2.4), with these boundary conditions, are shown in Figures 7 and 8, respectively. The interesting aspect of these surfaces is at $x = 1$. The Zero Neumann condition yields an approximate solution that is nearly constant near the boundary $x = 1$, which resembles the graphs in Figure 1. There were nonzero measurements taken downstream that would tend to favor this boundary condition over the Zero Dirichlet condition.

It is therefore reasonable to consider the effect of the Robin condition at $x = 1$. The Robin condition (1.4) gives an intermediate boundary condition that may be useful in the modeling process. The parameter $a \in [0,1]$ gives a measure of "how Dirichlet" or "how Neumann" the model is. Consider the simple diffusive-convective equations (4.2.3) and (4.2.4) with a Robin condition at $x = 1$ with $a = 0.5$. The approximations to the solutions are shown in Figures 9 and 10, respectively.

## 4.3 More Complex Models

Section 4.2 detailed the effect of each part of the PDE model, i.e., boundary conditions, source terms, etc., individually. In this section, we will take combinations of the above in an attempt to obtain a model whose solution more closely resembles the Oak Creek data.

Thus far, the models have shown that the peak of the contamination has been isolated to one point along the creek in the middle of the year. This may not be satisfactory as Figure 1 shows several peaks throughout the summer near Slide Rock and extending as far as Grasshopper Point in 1996. Let us consider equation (4.2.4) with a multi-peak source term such as

$$f(t,x) = (1 - \cos(2\pi t))\,(e^{(x-0.3)^2} + 3e^{(x-0.7)^2}) \tag{4.3.1}$$

The coefficient of the second exponential increases the effect of the source near Slide Rock (near $x = 0.7$). Figures 11 and 12 show approximations to the solution surfaces to this problem with downstream Zero Dirichlet and Zero Neumann conditions, respectively. The Neumann condition downstream gives a more accurate picture of the contamination downstream as it reflects the nonzero measurements taken in the study of Oak Creek [C]. Figure 13 shows an approximation to the solution of the previous model with the coefficient of convection $\beta = 3$. The increased effect of convection pushes more contaminants downstream. We have now obtained a more realistic model since there is a higher concentration of contaminants downstream for the majority of the year (compare to Figure 1).

## 4.4 Conclusions

It appears that the downstream Zero Neumann boundary condition generally improves the model. This makes sense because the data indicate nearly constant amounts of pollution near the end of the study area. One could also experiment with values of $a$ that are nearly Neumann (such as $a = 0.2$) to decrease slightly the amount of contamination downstream.

The models presented in this chapter are not limited in time to one year. We defined the domain of our PDEs with only one year in mind, however, it is easy to run the simulations for longer periods of time. As we would suspect, the final contaminant distribution along the creek at the end of year $X$ would become the initial contaminant distribution at the beginning of experiment year $X+1$. For example, consider the following problem:

$$\begin{aligned} u_t(t,x) &= u_{xx}(t,x) + f(t,x) \qquad t \in (0,3), \quad x \in (0,1) \\ u(0,x) &\equiv 0 \qquad x \in [0,1] \\ u(t,0) &= 0 \qquad t \in [0,3] \\ u_x(t,1) &= 0 \qquad t \in [0,3], \end{aligned} \tag{4.4.1}$$

where $f(t,x)$ is defined in equation (4.3.1). This extends our previous model (Figure 7) to three years. The approximation to the solution is shown in Figure 14. Note that the solution appears to have become periodic in time and it is reasonable to assume that it would continue in this manner indefinitely. Figure 15 shows the same problem with convection. Again, we have a periodic solution, however, the convection term has pushed the contaminants downstream.

Using our modeling techniques, it would be very difficult to match the exact data shown in Figure 1. There are random elements, such as wading tourists, irregular water flow, and rainfall, that have not been considered. These and other unpredictable events can upset the balance and hence make the models less accurate. We have, however, shown how each element of the PDE model influences the solution surface. In addition, we have attempted to find the parameters that give a reasonable model for the Oak Creek contamination data.

In conclusion, the model whose solution is given in Figure 10 appears to be the best representation of the Oak Creek data. It most accuratly places the peak of the contamination and gives the relative shape of the actual data throughout $\Omega$. It also accounts for all of the factors whose effects we originally set out to exhibit.

# CHAPTER FIVE

# FURTHER RESEARCH

While the mathematics behind the linear parabolic PDEs used in this work is well-developed, the application to Oak Creek is new. The work contained in this thesis is a step toward modeling the flow of contaminants through the Oak Creek Canyon waterway. We have demonstrated both PDE models and techniques for approximating solutions to these PDEs. There are further possibilities that can be explored.

Due to time constraints, comparison to theoretical error bounds was not done. Estimation of higher partial derivatives is required in order to determine an error bound for each of our three numerical methods [B-F]. While we are confident that our approximate solutions are within those error bounds, it would be desirable to verify this. Other numerical methods implemented in future studies would benefit from similar error analysis.

Further analysis of our mathematical models can be performed to determine the actual effect of each parameter on the solution. Since it appears that the source terms in the models have a significant effect on the shape of the solution, it is reasonable to question the usefulness of the other terms in the model. We are confident that the parameter $\beta$ has a significant effect on the model because the introduction of convection did show a difference in the solution surfaces. Further study can be done on other terms to determine what effect they have on the model as a whole.

This project was limited to PDEs with one space dimension. Currently, only one-dimensional space data exists for the water quality of Oak Creek. One could also consider the width and/or depth of the creek. At each sampling site, contaminants could be measured at different depths. A diffusive-convective equation in higher space dimensions would then be required. The Explicit Method could be applied to higher dimensional problems with regular grids.

To consider irregular grids and more complex boundary conditions, other numerical techniques could then be employed to solve the PDEs. One could use the so-called Finite Element methods. These methods are better suited to handle complex boundary conditions that can arise in modeling situations, such as wildly varying shorelines and creekbeds [B-F]. More biological and physical study could be undertaken at Oak Creek to better understand the processes that occur near the boundaries of the study area.

More research can also be undertaken on the nature of the contamination so that the source term in equation (1.1) could be more precise. For example, the bacteria may be reproducing at a detectable rate or may be decreasing in number due to the predation by other organisms. It has also been hypothesized that the sediments can act as reservoirs for *E. coli* and could increase the number of bacteria in the water column when the sediment is disturbed by human or animal activity in the creek [C]. Hence, it may be useful to consider an additional spatial dimension in future models.

In section 4.1, we mentioned the pseudo-random movements of the bacteria from [E-K]. In addition, events such as rainfall and tourism can affect the levels of contamination in the creek. This idea suggests the possibility of incorporating some random noise into the models. Perhaps this would create a more realistic model.

Finally, it would be insightful if one could find a smoothing algorithm, such as a three dimensional spline, for the data collected at Oak Creek. If we could find an interpolation of the data that can be expressed in closed form, we could get a better idea for boundary conditions and forcing terms.

**Table 1** -- Results of execution of the Explicit Algorithm with the indicated values of M and N at t=1. All errors are absolute errors.

A. M = 50 and N = 5,000

| x | u(1, x) | Actual | Error |
|---|---|---|---|
| 0.0 | 0.000000000 | 0.000000000 | 0.000000000 |
| 0.1 | 0.000015880 | 0.000015983 | 0.000000104 |
| 0.2 | 0.000030205 | 0.000030402 | 0.000000197 |
| 0.3 | 0.000041574 | 0.000041845 | 0.000000271 |
| 0.4 | 0.000048873 | 0.000049192 | 0.000000319 |
| 0.5 | 0.000051388 | 0.000051723 | 0.000000335 |
| 0.6 | 0.000048873 | 0.000049192 | 0.000000319 |
| 0.7 | 0.000041574 | 0.000041845 | 0.000000271 |
| 0.8 | 0.000030205 | 0.000030402 | 0.000000197 |
| 0.9 | 0.000015880 | 0.000015983 | 0.000000104 |
| 1.0 | 0.000000000 | 0.000000000 | 0.000000000 |

B. M = 100 and N = 20,000

| x | u(1, x) | Actual | Error |
|---|---|---|---|
| 0.0 | 0.000000000 | 0.000000000 | 0.000000000 |
| 0.1 | 0.000015957 | 0.000015983 | 0.000000026 |
| 0.2 | 0.000030353 | 0.000030402 | 0.000000049 |
| 0.3 | 0.000041777 | 0.000041845 | 0.000000068 |
| 0.4 | 0.000049112 | 0.000049192 | 0.000000080 |
| 0.5 | 0.000051639 | 0.000051723 | 0.000000084 |
| 0.6 | 0.000049112 | 0.000049192 | 0.000000080 |
| 0.7 | 0.000041777 | 0.000041845 | 0.000000068 |
| 0.8 | 0.000030353 | 0.000030402 | 0.000000049 |
| 0.9 | 0.000015957 | 0.000015983 | 0.000000026 |
| 1.0 | 0.000000000 | 0.000000000 | 0.000000000 |

**Table 1** -- Results of execution of the Explicit Algorithm with the indicated values of M and N at t=1. All errors are absolute errors.

C. M = 1,000 and N = 2,000,000

| x | u(1, x) | Actual | Error |
|---|---|---|---|
| 0.0 | 0.000000000 | 0.000000000 | 0.000000000 |
| 0.1 | 0.000015983 | 0.000015983 | 0.000000000 |
| 0.2 | 0.000030402 | 0.000030402 | 0.000000000 |
| 0.3 | 0.000041844 | 0.000041845 | 0.000000001 |
| 0.4 | 0.000049191 | 0.000049192 | 0.000000001 |
| 0.5 | 0.000051722 | 0.000051723 | 0.000000001 |
| 0.6 | 0.000049191 | 0.000049192 | 0.000000001 |
| 0.7 | 0.000041844 | 0.000041845 | 0.000000001 |
| 0.8 | 0.000030402 | 0.000030402 | 0.000000000 |
| 0.9 | 0.000015983 | 0.000015983 | 0.000000000 |
| 1.0 | 0.000000000 | 0.000000000 | 0.000000000 |

**Table 2** -- Results of execution of the Implicit Algorithm (implicit.c) at t=1. All errors are absolute errors.

A. M = N = 5000

| x | u(1, x) | Actual | Error |
|---|---|---|---|
| 0.0 | 0.000000000 | 0.000000000 | 0.000000000 |
| 0.1 | 0.000016144 | 0.000015983 | 0.000000160 |
| 0.2 | 0.000030674 | 0.000030402 | 0.000000272 |
| 0.3 | 0.000042200 | 0.000041845 | 0.000000355 |
| 0.4 | 0.000049594 | 0.000049192 | 0.000000402 |
| 0.5 | 0.000052131 | 0.000051723 | 0.000000408 |
| 0.6 | 0.000049564 | 0.000049192 | 0.000000372 |
| 0.7 | 0.000042142 | 0.000041845 | 0.000000297 |
| 0.8 | 0.000030594 | 0.000030402 | 0.000000192 |
| 0.9 | 0.000016050 | 0.000015983 | 0.000000067 |
| 1.0 | 0.000000000 | 0.000000000 | 0.000000000 |

B. M = 5000 and N = 10,000

| x | u(1, x) | Actual | Error |
|---|---|---|---|
| 0.0 | 0.000000000 | 0.000000000 | 0.000000000 |
| 0.1 | 0.000016050 | 0.000015983 | 0.000000066 |
| 0.2 | 0.000030495 | 0.000030402 | 0.000000093 |
| 0.3 | 0.000041954 | 0.000041845 | 0.000000109 |
| 0.4 | 0.000049305 | 0.000049192 | 0.000000113 |
| 0.5 | 0.000051827 | 0.000051723 | 0.000000104 |
| 0.6 | 0.000049275 | 0.000049192 | 0.000000083 |
| 0.7 | 0.000041897 | 0.000041845 | 0.000000052 |
| 0.8 | 0.000030416 | 0.000030402 | 0.000000014 |
| 0.9 | 0.000015957 | 0.000015983 | 0.000000027 |
| 1.0 | 0.000000000 | 0.000000000 | 0.000000000 |

**Table 3** -- Results of execution of the Crank-Nicholson Algorithm at t=1 with M = N = 5000. All errors are absolute errors.

| x | u(1, x) | Actual | Error |
|---|---|---|---|
| 0.0 | 0.000000000 | 0.000000000 | 0.000000000 |
| 0.1 | 0.000015956 | 0.000015983 | 0.000000028 |
| 0.2 | 0.000030317 | 0.000030402 | 0.000000085 |
| 0.3 | 0.000041709 | 0.000041845 | 0.000000136 |
| 0.4 | 0.000049017 | 0.000049192 | 0.000000175 |
| 0.5 | 0.000051524 | 0.000051723 | 0.000000199 |
| 0.6 | 0.000048987 | 0.000049192 | 0.000000205 |
| 0.7 | 0.000041652 | 0.000041845 | 0.000000193 |
| 0.8 | 0.000030238 | 0.000030402 | 0.000000164 |
| 0.9 | 0.000015863 | 0.000015983 | 0.000000120 |
| 1.0 | 0.000000000 | 0.000000000 | 0.000000000 |

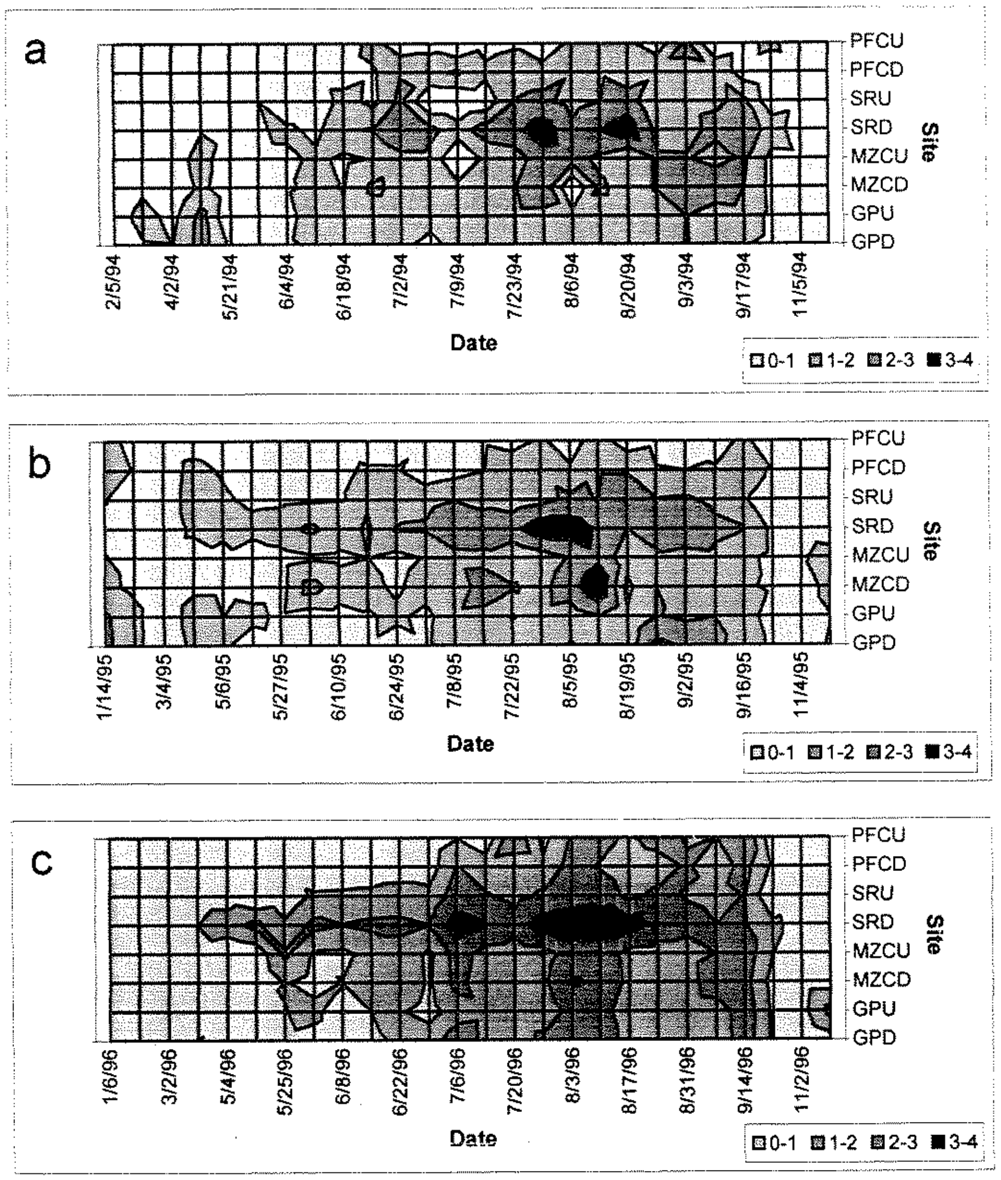
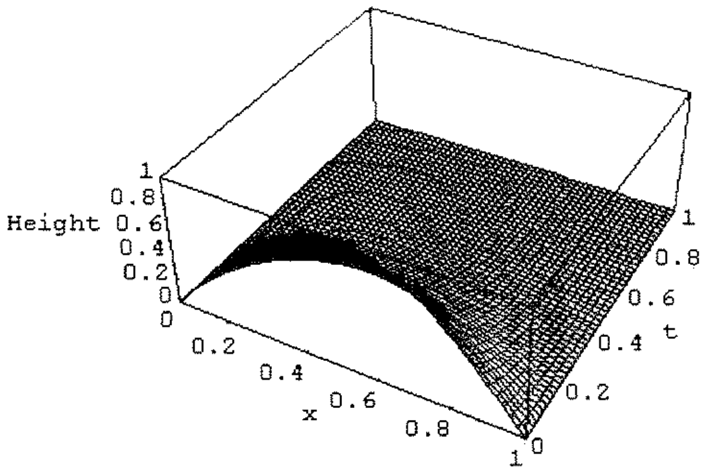


**Figure 1.** Data from Crabill *et al.* [C] showing the log order levels of contamination throughout Oak Creek Canyon during the three-year study (a - c represent the years 1994 -1996). The sampling sites are Pine Flats (PFC), Slide Rock (SR), Manzanita Campground (MZC), and Grasshopper Point (GP). The suffixes U and D represent upstream and downstream collections.



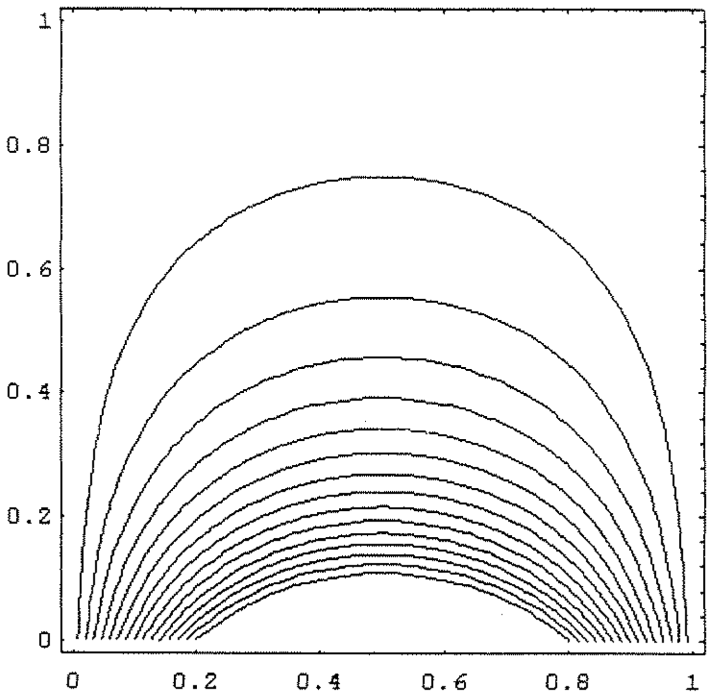
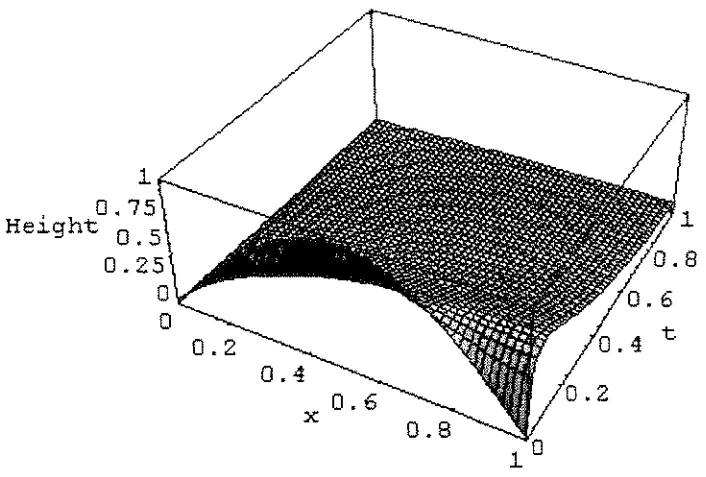


**Figure 2.** Three dimensional plot and contour plot of equation (1.5). The contour plot shows x on the horizontal axis and t on the vertical axis.



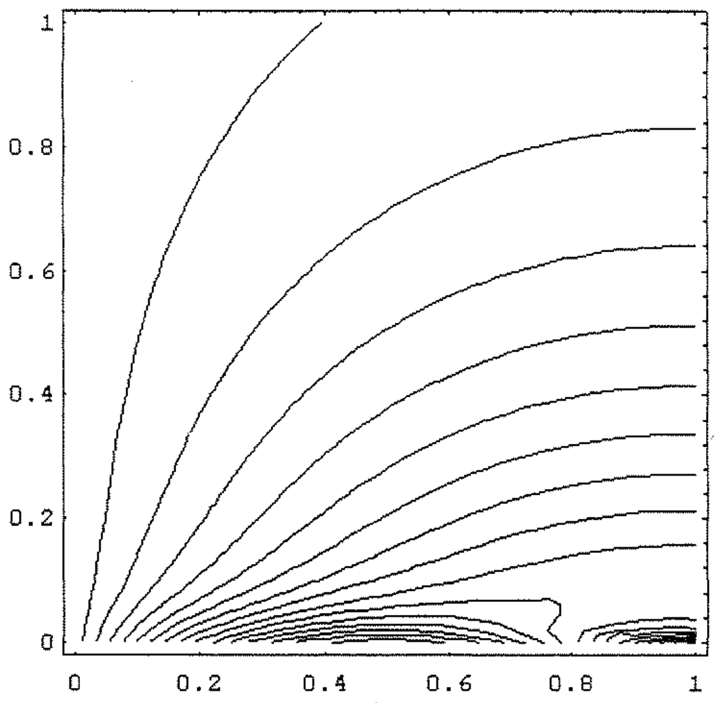
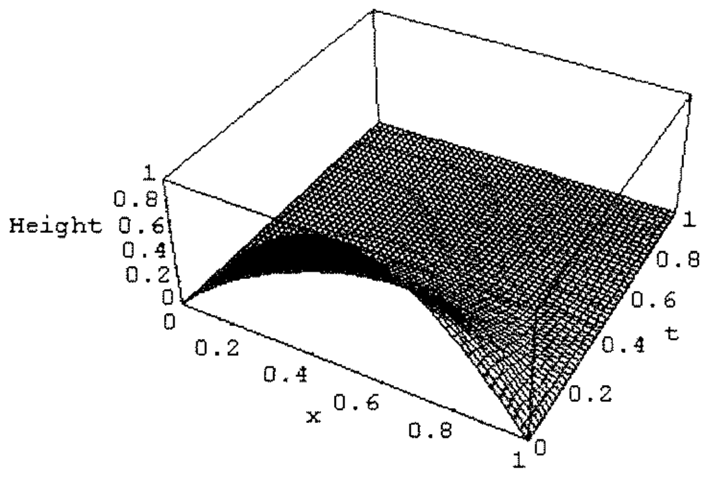


**Figure 3**. Approximation to the solution surface and contour plot of the approximation to the solution of equation (1.5) with Neumann boundary conditions at x = 1. The contour plot shows x on the horizontal axis and t on the vertical axis.



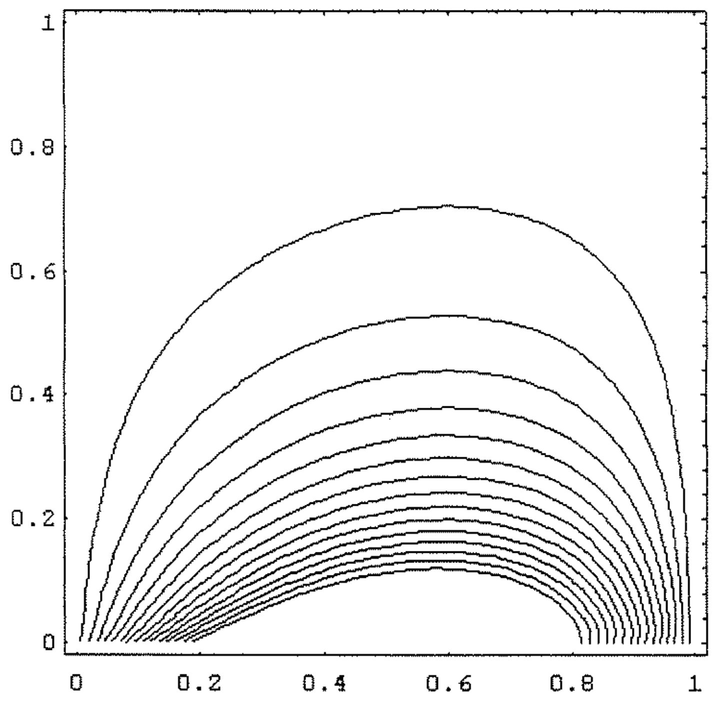
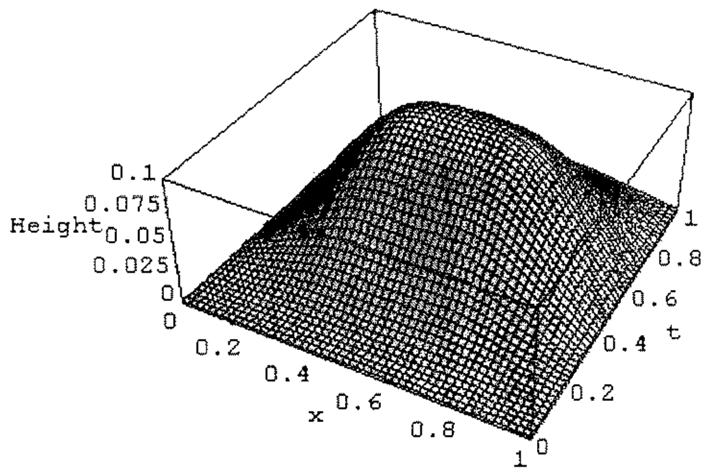


**Figure 4**. Approximation to the solution surface and contour plot of the approximation to the solution surface of equation (1.6) showing the effect of convection. The contour plot shows x on the horizontal axis and t on the vertical axis.



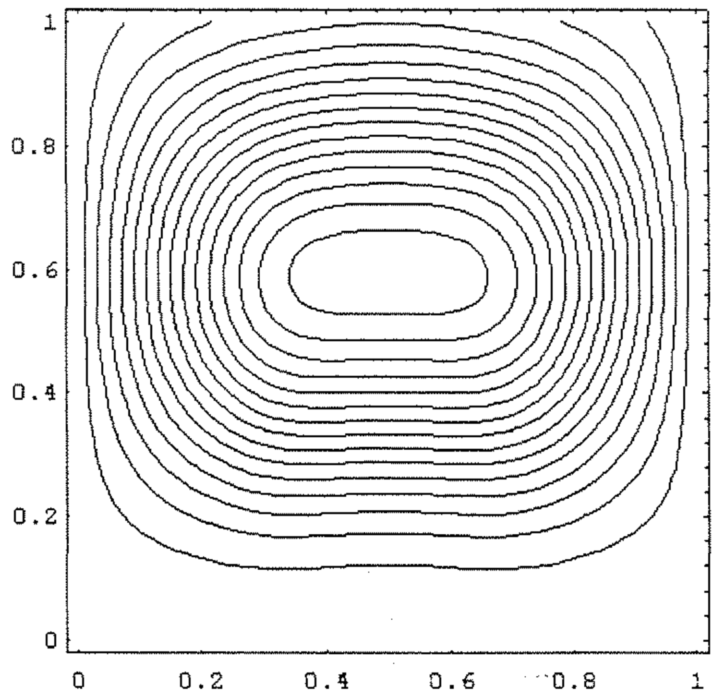


**Figure 5**. Approximation to the solution surface and contour plot of the approximation to the solution surface of equation (4.2.3). The contour plot shows x on the horizontal axis and t on the vertical axis.

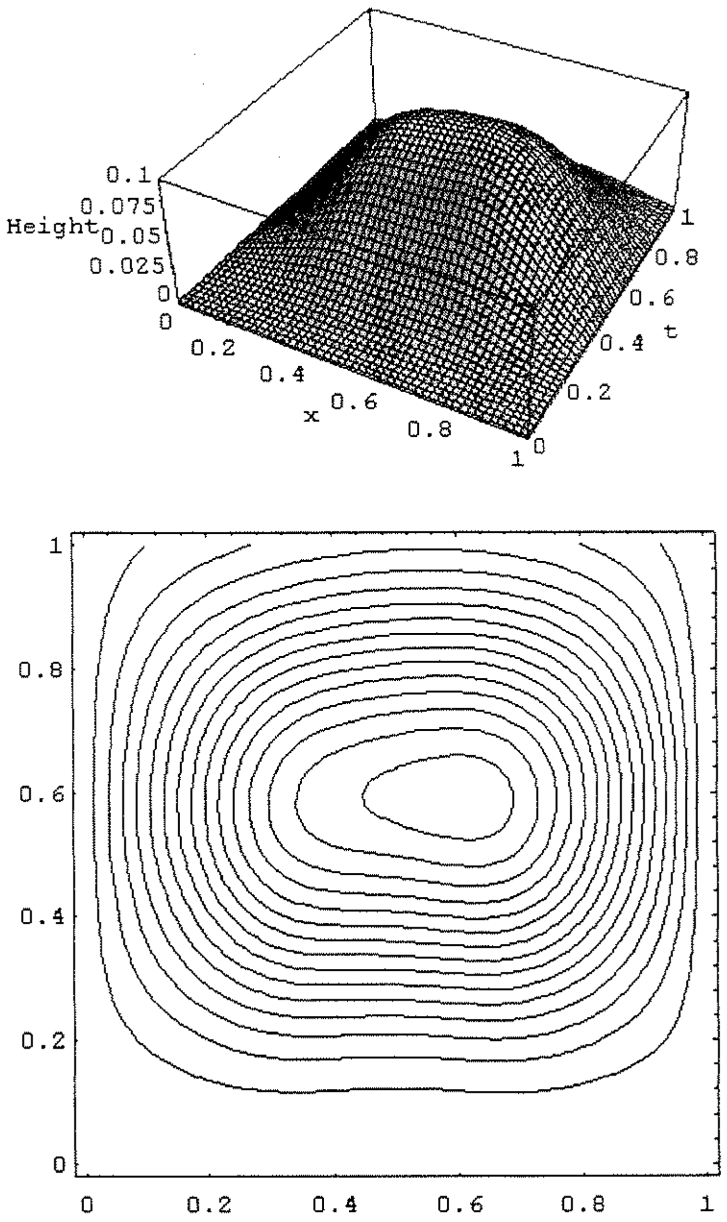
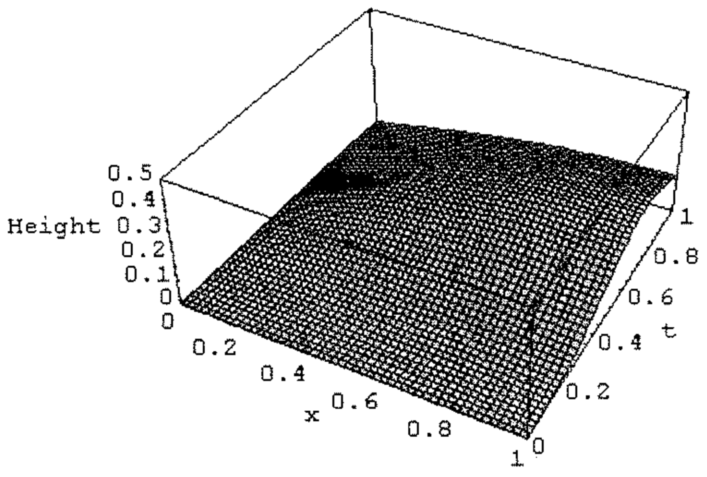


**Figure 6**. Approximation to the solution surface and contour plot of the approximation to the solution surface of equation (4.2.4). The contour plot shows x on the horizontal axis and t on the vertical axis.



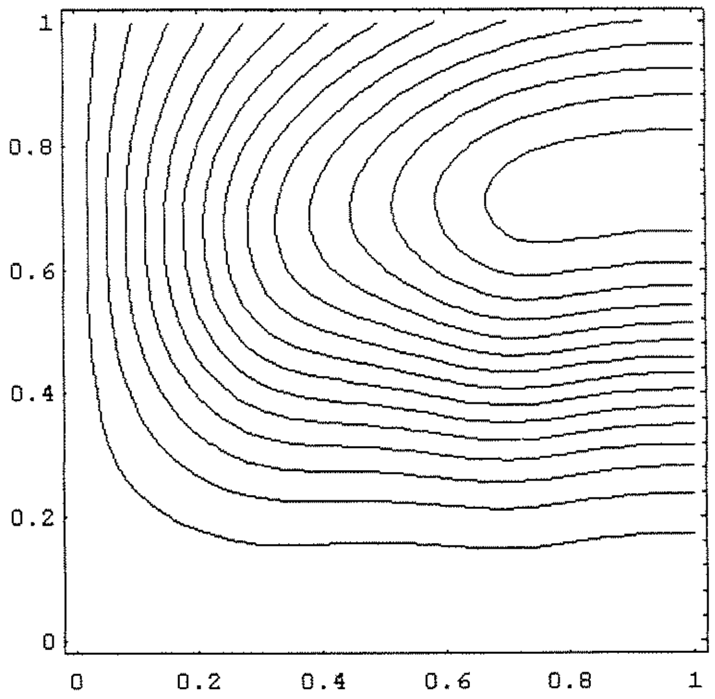
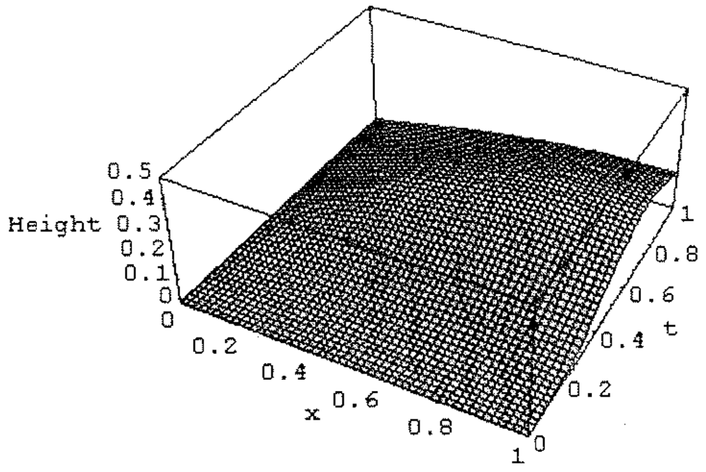


**Figure 7**. Approximation to the solution surface and contour plot of the approximation to the solution surface of equation (4.2.3) with a zero Neumann condition at x=1. The contour plot shows x on the horizontal axis and t on the vertical axis.



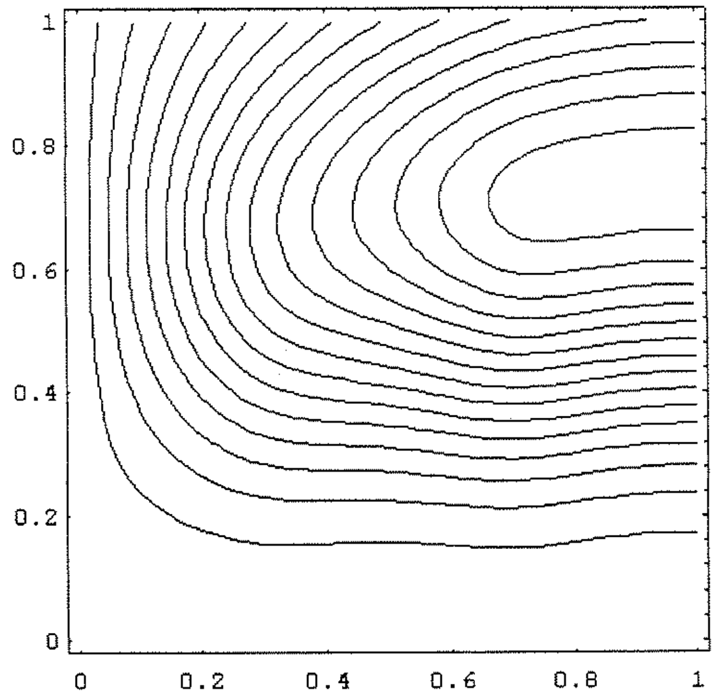


**Figure 8**. Approximation to the solution surface and contour plot of the approximation to the solution surface of equation (4.2.4) with a zero Neumann condition at x=1 and convection. The contour plot shows x on the horizontal axis and t on the vertical axis.

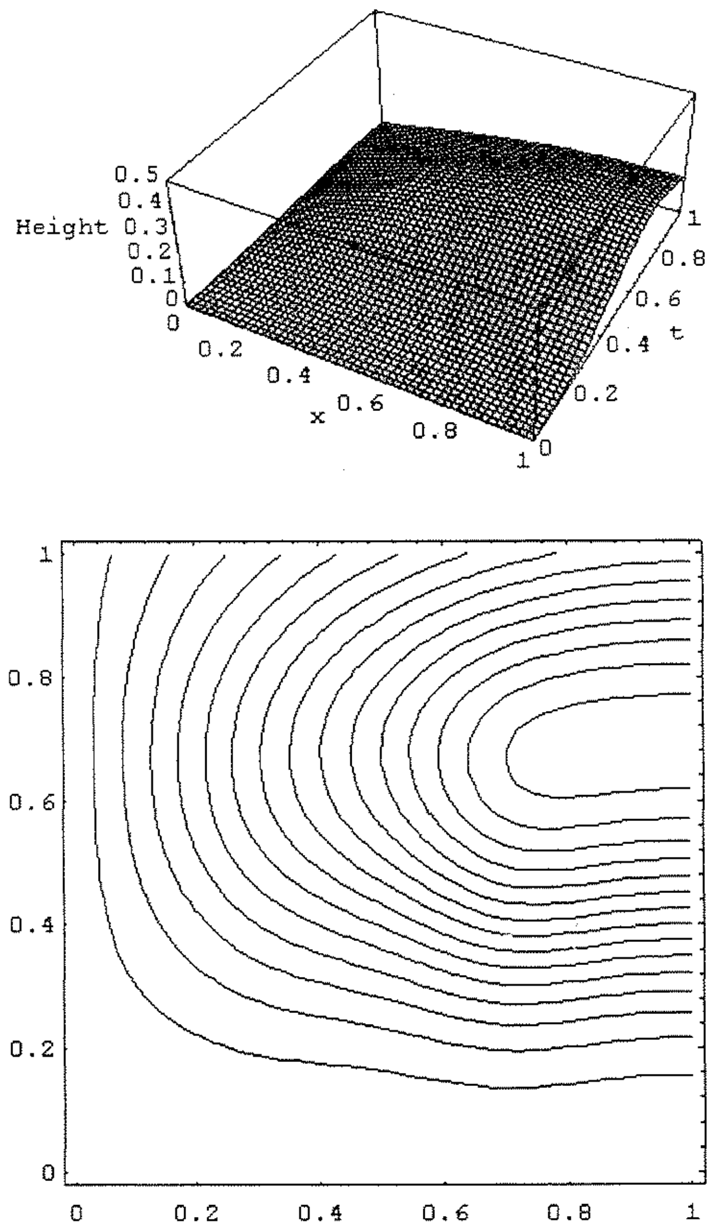
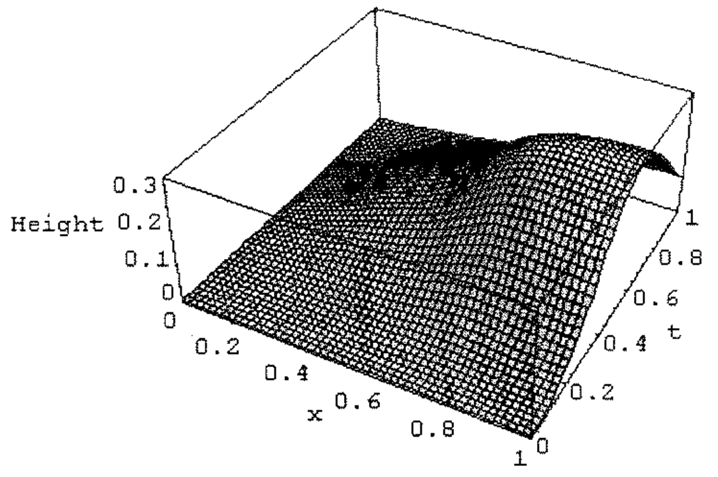


**Figure 12**. Solution surface and contour plot of the solution surface of equation (4.2.4) with source term (4.3.1), a zero Neumann condition at x=1 and convection. The contour plot shows x on the horizontal axis and t on the vertical axis.



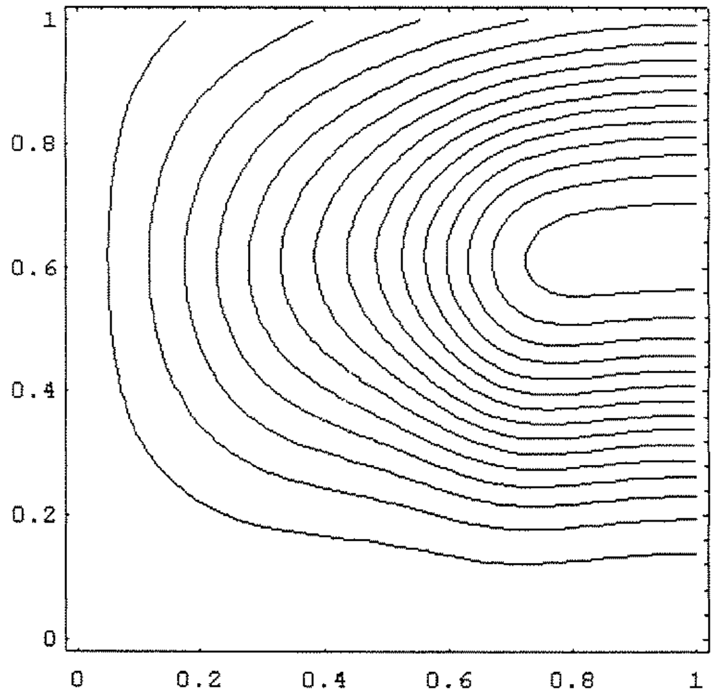
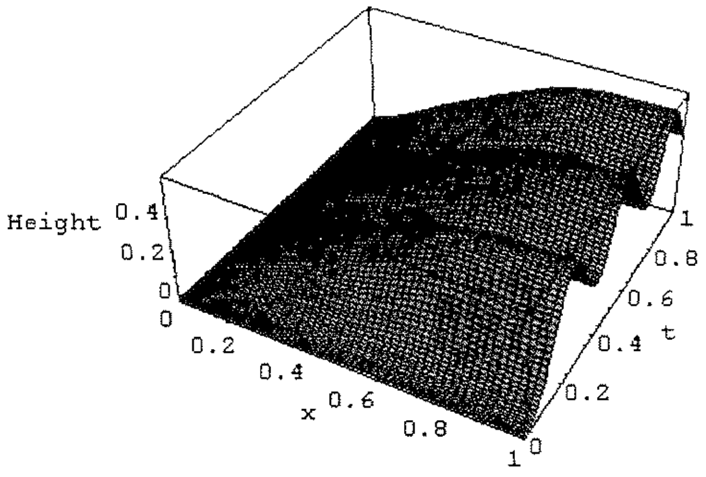


**Figure 13**. Approximation to the solution surface and contour plot of the approximation to the solution surface of equation (4.2.4) with source term (4.3.1), a zero Neumann condition at x=1 and convection with a coefficent of three. The contour plot shows x on the horizontal axis and t on the vertical axis.



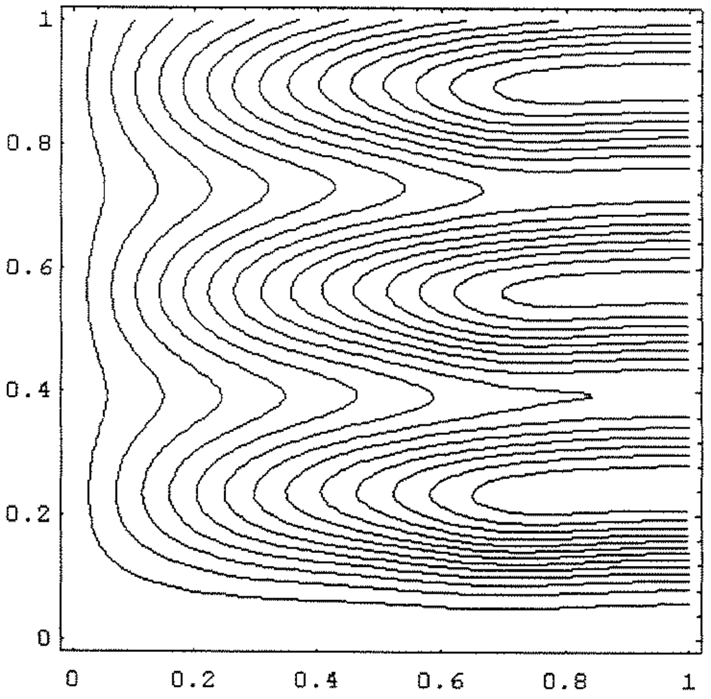
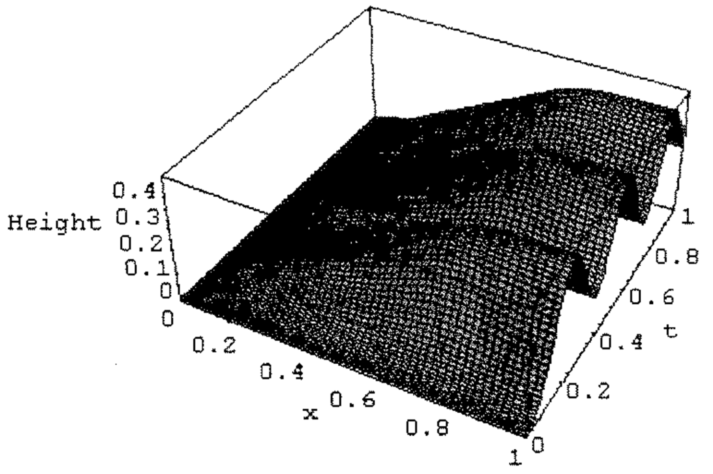


**Figure 14.** Approximation to the solution surface and contour plot of the approximation to the solution surface of equation (4.4.1). The contour plot shows x on the horizontal axis and t on the vertical axis.



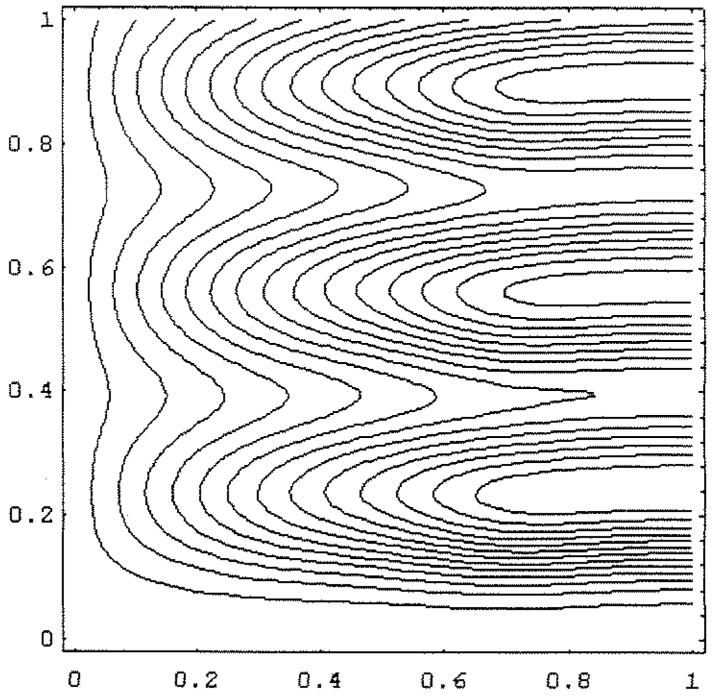


**Figure 15.** Approximation to the solution surface and contour plot of the approximation to the solution surface of equation (4.4.1) with convection. The contour plot shows x on the horizontal axis and t on the vertical axis.

**Appendix 1** Computer code used to implement the Explicit Method with any source term and downstream boundary condition.

```
/ *     This program implements the Explicit Method to approximate          * /
/ *      solutions to diffusive-convective equations with different source  * /
/ *     terms and downstream boundary conditions.                           * /


#include <stdio.h>
#include <math.h>

#define pi 3.14159265359

#define  M 50     / * Initial number of divisions on x-axis * /
#define  N 5000   / * Initial number of divisions on t-axis * /


void main()

{

int         i , j;                        / * Counters * /
int         t;

double      u[M+1];                       / *  Array of values, u(t,x) * /
double      new_u[M+1];                   / *  Array of new values to be passed
to u * /

double      f();                          / *  Forcing Function        * /
double      g();                          / *  Initial value function.   * /

double  m2n = ((double)M * (double)M  /  (double)N);

                                          / * M^2 divided by N * /

double      alpha = 1.0;                  / *  Robin conditions (1 = Dirichlet) * /
double      k = 1.0;                      / *  Diffusion coefficient            * /
double      conv = 1.0;                   / *  Convection coefficient           * /
double      forc = 1.0;                   / *  Flag for forcing term  1=On      * /

FILE *writefile;
writefile=fopen("results","w");
/ *     Initalize the array     * /

for(i=0; i<=M; i++)
        {
        u[ i ] = g( (double)i / M );
        }

for(t=0; t<=(1*N); t++)
        {
        for(j = 1; j < M; j++)
         new_u[ j ] = k * m2n * (u[ j+1 ] - 2*u[ j ] + u[ j-1 ]) + u[ j ]
                        + forc * (1.0 / (double)N) * f( (double)j / M , (double)t / N )
                        - conv * M / (double)N / 2.0 * (u[ j+1 ] - u[ j-1 ]);

 / * Downstream boundary condition * /

        new_u[M] =  (alpha-1) * (-u[M-2] + 4*u[M-1])  /  (3-3*alpha+2*alpha /
                                                (double)M);


 / *************** Uncomment for graphics **************************************** /

        if((t%100)==0)
                {
                for(i=0; i<=M; i++)   / * =(M / 50))  * /
                        fprintf(writefile,"%12.9f\n",u[ i ]);
                }

 / ******************************************************************************** /

        for(j=1; j<=M; j++)
                u[ j ] = new_u[ j ];

 / *************** Uncomment for nice five-colunm output ***********************

   if( (t%100) == 0 )

                {
                for(j=0; j<=M; j+=M / 5)
                        fprintf(writefile,"%9.7f   ",u[ j ]);
                fprintf(writefile,"\n");
                }

******************************************************************************** /


        }
```

```
/ *********************** Uncomment for simple output ********************************

        for(j=0; j<=M; j+=(M / 10))

                fprintf(writefile,"Final u[%3d] = %12.9f  \n",j,u[ j ]);

*************************************************************************************** /

fclose(writefile);

}


 / *     Initial value function declaration    * /

double          g(double X)

        {
        return ( sin(pi*X) );
        }

 / *     Forcing term    * /

double          f(double X, double T)

        {
        return (
                (1-cos(2*pi*T))*(1*exp(-100*(X-0.3)*(X-0.3)) +
                        1*exp(-100*(X-0.7)*(X-0.7)))
                  );
        }
```

**Appendix 2** The "tridiagsolve" subroutine, written in C, that implements Gaussian Reduction to solve a matrix equation of the form **A**u=v for u, where **A** is a square, tridiagonal matrix.

```
/*  Subroutine that implements Gaussian Reduction on a tridiagonal matrix   */
/*  This subroutine will read in the upper diagonal, lower diagonal, diagonal, */
/*     and the "b" vector and then manipulate the b vector.                 */
/*     N is the size of the square matrix.                                   */


void tridiagsolve(int N, double U[ ], double L[ ], double D[ ], double B[ ])

{

int                 i;


for(i=0; i<=N-2; i++)
        D[ i+1 ] = D[ i+1 ] - (L[ i ]/D[ i ]) * U[ i ];

for(i=0;  i<=N-2;  i++)
        B[ i+1 ] = B[ i+1 ] - (L[ i ]/D[ i ]) * B[ i ];

B[ N-1 ] = B[ N-1 ]/D[ N-1 ];

for(i=N-1;  i>=0;  i--)
        B[ i ] =  (B[ i ] - U[ i ] * B[ i+1 ])/D[ i ];

}
```

**Appendix 3** Computer code used to implement the Implicit Method with any source term and downstream boundary condition.

```
/ *     This program implements the Implicit Method to approximate            * /
/ *     solutions to diffusive-convective equations with different source     * /
/ *     terms and downstream boundary conditions.                             * /


#include <stdio.h>
#include <math.h>
#include "..\lib\matrix.h"

#define pi 3.141592654

#define M  5000             /* Number of x - divisions */
#define N  5000             /* Number of t - divisions */

/************************* Initial Function *****************************************/

double initialf(double X)
        {
        return sin(pi * X);
        }

/*********************** Actual Function *****************************************/

double        actualf(double X)
        {
        return (exp(-pi * pi) * sin(pi * X));
        }

/********************** Source Term *******************************************/

double        f(double T, double X)

        {
        return (
                ( 1 - cos(2*pi*T) )*( exp( -100 * (X-0.4)*(X-0.4) ) +
                        exp( -100 * (X-0.6)*(X-0.6) ))
                );
        }

/**********************************************************************************/
```

```
void main()

{

int             i;
int             j;

double          p = (double)(M*M)/(double) N;

                                                /* Delta t divided by Delta x squared */

double          q = (1.0)/N;


double          u[M];                           /* Array of function values */

double          D[M];                           /* Diagonal */
double          U[M-1];                         /* Upper diagonal */
double          L[M-1];                         /* Lower diagonal */

double          initialf();                     /* Initial value function */
void    tridiagsolve();                         /* Tri Diagonal Solver */


FILE *writefile;
writefile = fopen("Iresults","w");

/* Initialize the arrays */

for(i=0; i<=M-2; i++)
        {
        u[i] = initialf( (double) (i+1) / (double) M );
        }

/* Solve the PDE */

for(i=0; i<=(N-1); i++)
        {

        u[0] = 0.0;
        u[M-1] = 0.0;                           /* Initialize the Dirichlet cond'n */
        for(j=0; j<=M-3; j++)
                {
                D[j] = 2*p + 1;
                U[j] = -p;
                L[j] = -p;
                }
        D[M-2]= 2*p + 1;


/* Add the forcing term to the u[i]'s */

        for(j=0; j<=M-2; j++)
                u[j] = u[j] + q * f( (double)i/N, (double)j/M );

        tridiagsolve(M-1,U,L,D,u);

/********* Use these two lines when printing graphics *****************************
        for(j=0; j<=M-2; j++)
                fprintf(writefile,"%12.9g\n",u[j]);
***********************************************************************************/

        }

/********* Use these two lines when only the numbers are wanted ***************/

for(i=0; i<=M-2; i+=(M/10))
        fprintf(writefile,"u[%2d] = %12.9f      Actual = %12.9f     Error =
                %12.9f\n",i,u[i],actualf((double)(i)/M),u[i]-actualf((double)(i)/M));

***********************************************************************************/

fclose(writefile);

}
```

**Appendix 4** Computer code used to implement the Crank-Nicholson Method with any source term and downstream boundary condition.

```
/ *     This program implements the Crank-Nicholson Method to approximate */
/ *     solutions to diffusive-convective equations with different source  * /
/ *     terms and downstream boundary conditions.                          * /


#include <stdio.h>
#include <math.h>
#include "matrix.h"

#define pi 3.141592654

#define M  5000      /* Number of x - divisions */
#define N  5000      /* Number of t - divisions */


/************************** Actual Solution ****************************************/

double  actualf(double X)
        {
        return (exp(-pi * pi) * sin(pi * X));
        }

/************************** Source Term *******************************************/

double        f(double T, double X)

        {
        return (
           ( 1 - cos(2*pi*T) )*( exp( -100 * (X-0.4)*(X-0.4) ) +
                exp( -100 * (X-0.6)*(X-0.6) ))
                );
        }

/**********************************************************************************/
```

```
void main()

{

int             i;
int             j;

double          p = (double)(M*M)/(double) N;

                                        /* Delta t divided by Delta x squared */

double          q = (1.0)/N;

double          u[M];                   /* Array of function values */
double          b[M];                   /* Temp array of fct values */

double          D[M];                   /* Diagonal */
double          U[M-1];                 /* Upper diagonal */
double          L[M-1];                 /* Lower diagonal */

double          initialf();             /* Initial value function */
void            tridiagsolve();         /* Tri Diagonal Solver */

double          alpha = 1.0;            /* Boundary condition */
double          forc  = 0.0;            /* Flag for the forcing term */


FILE *writefile;
writefile = fopen("CNresults","w");

/* Initialize the arrays */

for(i=0; i<M-2; i++)
        {
        u[i+1] = initialf( (double) (i) / (double) M );
        }


/* Solve the PDE */

for(i=0; i<=(N-1); i++)
        {
```

```
        for(j=0; j<=M-3; j++)
                {
                D[j] = p + 1;
                U[j] = -p/2;
                L[j] = -p/2;
                }
        D[M-2]= p + 1;


        b[0] = (1-p)*u[0] + p/2 * u[1];

        for(j=1; j<=(M-2); j++)
                {
                b[j] = p/2 * u[j-1] + (1-p) * u[j] + p/2 * u[j+1];
                }

        b[M-2] = p/2 * u[M-3] + (1-p) * u[M-3];

        for(j=0; j<=(M-2); j++)
                u[j] = b[j];


        tridiagsolve(M-1,U,L,D,u);

/*********** Use these two lines when printing graphics *************************/

        for(j=0; j<=M-2; j++)
                printf("%12.9g\n",u[j]);

/**********************************************************************************/

/* Add the forcing term to the u[i]'s */

        for(j=0; j<=M-2; j++)
                u[j] = u[j] + forc * q * f( (double)i/N, (double)j/M );


        }
```

```
/************** Use these two lines when only the numbers are wanted **********/

for(i=0; i<=M-2; i+=(M/10))
        fprintf(writefile,"u[%2d] = %12.9f     Actual = %12.9f     Error =
                %12.9f\n",i,u[i],actualf((double)(i)/M),u[i]-actualf((double)(i)/M));

/*************************************************************************************/

fclose(writefile);

}

/*********************** Initial value function ****************************************/

double initialf(double X)

{
return sin(pi * X);
}

/*************************************************************************************/
```

# BIBLIOGRAPHY

[A-I] Julius Adler. Chemoreceptors in Bacteria. *Science*. **166**:1588-1596. 26 December 1969.

[A-II] Julius Adler and Margaret Dahl. A Method for Measuring the Motility of Bacteria and for Comparing Random and Non-random Motility. *Journal of General Microbiology*. **46**:161-173. 1967.

[B] Paul W. Berg and James L. McGregor. *Elementary Partial Differential Equations*. Holden-Day, San Francisco, 1966.

[B-F] Richard Burden and J. Douglas Faires. *Numerical Analysis, Fifth Edition*. PWS Publishing Co., Boston, 1993.

[C] C. Crabill, R. Donald, J. Snelling, R. Foust, and G. Southam. The impact of sediment fecal coliform reservoirs on seasonal water quality in Oak Creek, AZ. Submitted to *Water Research*, 1998.

[E-K] Leah Edelstein-Keshet. *Mathematical Models in Biology*. Random House, New York, 1987.

[I] E. Isaacson and H.B. Keller. *Analysis of numerical methods*. John Wiley and Sons, New York, 1966.

[L] Gary Lieberman. *Second Order Parabolic Differential Equations*. World Scientific, Singapore, 1996.

[Lu] D. Ludwig, D.G. Aronson, and H.H. Weinberger. Spatial patterning of the spruce budworm. *Journal of Mathmatical Biology*, **8**:217-258, 1979.

[R] Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, New York, 1976.

[S-I] Lee A. Segal. *Modeling dynamic phenomena in molecular and cellular biology*. Cambridge University Press, Cambridge, 1984.

[S-II] Lee A. Segal, I. Chet, and Y. Henis. A simple quantitative assay for bacterial motility. *Journal of General Microbiology*, **98**:329-337, 1977.

[Y] E. C. Young. *Partial Differential Equations, An Introduction*. Allyn and Bacon, Inc., Boston, 1972.

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