AN ABSTRACT OF THE THESIS OF

HARRY MARKWOOD HULL (Name) for the MASTER OF SCIENCE (Degree)

in MATHEMATICS (Major) presented on January 3, 1969 (Date)

Title: NUMERICAL SOLUTION OF THE HEAT EQUATION BY NET METHODS

Abstract approved: Redacted for Privacy

Dr. Ronald B. Guenther

This thesis examines various net or finite difference methods for solving parabolic partial differential equations in one space variable with constant coefficients. Included in this investigation are explicit, implicit and multi-step methods of varying orders of accuracy. These methods are compared with respect to accuracy, speed, efficiency, stability, simplicity of programming and other criteria. A method for the construction of net methods and analyzing the stability and convergence of the methods is briefly discussed. Sample programs for several of the better methods are given in Appendix C.

Numerical Solution of the Heat Equation by Net Methods

by

Harry Markwood Hull

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

June 1969

APPROVED:

Redacted for Privacy

Assistant Professor of Mathematics

in charge of major

Redacted for Privacy

Acting Chairman of Department of Mathematics

Redacted for Privacy

Dean of Graduate School

Date thesis is presented January 2, 1969

Typed by Clover Redfern for Harry Markwood Hull

ACKNOWLEDGMENT

This is to express my appreciation to Dr. Ronald B. Guenther who suggested the topic of the thesis. Dr. Guenther has given valuable suggestions concerning both the research and the writeup.

TABLE OF CONTENTS

Page

APPENDIX

NUMERICAL SOLUTION OF THE HEAT EQUATION BY NET METHODS

I. INTRODUCTION

In this thesis, we shall be interested in approximating solutions, $u(x, t)$, to the partial differential equation's problem

$$
\begin{aligned}
(1) \qquad & u_{xx} = u_t + f(x, t), && 0 < x < 1, \quad 0 < t \leq 1, \\
& u(x, 0) = \phi(x), && 0 \leq x \leq 1, \\
& u(0, t) = \mu(t), && 0 \leq t \leq 1, \\
& u(1, t) = \nu(t), && 0 \leq t \leq 1,
\end{aligned}
$$

where we have set $u_t = \partial u/\partial t$, $u_{xx} = \partial^2 u/\partial x^2$, and where f, ϕ, μ, ν are given functions. We remark that the seemingly more general parabolic equation $au_{xx} + bu_x + cu - u_t = f(x, t)$, $a > 0$, with constant coefficients can be reduced to an equation of the form (I) by introducing the new unknown function $u = \exp(\alpha x + \beta t)v$ and then changing the scale on the t axis.

The partial differential equation $a^2 u_{xx} = u_t + f(x, t)$ has numerous applications to problems in heat conduction, diffusion processes, flow of fluids in porous media, etc. In particular, if we wish to compute the temperature u of an insulated, homogeneous rod of length ℓ in the position x at the time t, where internal sources or sinks are present, then $u = u(x, t)$ satisfies the equation

$a^2 u_{xx} = f(x, t)$. The constant a^2 depends upon the conductivity, the specific heat and the density of the rod. To determine the temperature uniquely, it is necessary to know the initial temperature distribution, $u(x, 0)$, and to have some information concerning the temperature of the endpoints of the rod, say on $u(0, t)$ and $u(1, t)$.

The problem (1) can be solved by standard separation of variables techniques. In fact the solution to problem (1) may be written in the form $u(x, t) = u^{I}(x, t) + u^{II}(x, t) + u^{III}(x, t)$, where u^{I}, u^{II} and u^{III} satisfy the problems.

(2) $$u^{I}_{xx} = u^{I}_{t}, \qquad 0 < x < 1, \qquad 0 \leq t \leq 1,$$

$$u^{I}(x, 0) = \phi(x), \qquad 0 \leq x \leq 1,$$

$$u^{I}(0, t) = u^{I}(1, t) = 0, \qquad 0 \leq t \leq 1,$$

(3) $$u^{II}_{xx} = u^{II}_{t} + f(x, t), \qquad 0 < x < 1, \qquad 0 < t \leq 1.$$

$$u^{II}(x, 0) = 0, \qquad 0 \leq x \leq 1,$$

$$u^{II}(0, t) = u^{II}(1, t) = 0, \qquad 0 \leq t \leq 1,$$

(4) $$u^{III}_{xx} = u^{III}_{t}, \qquad 0 < x < 1, \qquad 0 < t \leq 1,$$

$$u^{III}(x, 0) = 0, \qquad 0 \leq x \leq 1,$$

$$u^{III}(0, t) = \mu(t), \qquad u^{III}(1, t) = \nu(t), \qquad 0 \leq t \leq 1.$$

The solution to (2), obtained by separation of variables, is given by

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

where

$$a_n = 2\int_0^1 \phi(x) \sin (n\pi x)dx.$$

By a variation of parameters device, one can show that $u^{II}(x, t)$ is given by

$$u^{II}(x, t) = \sum_{n=1}^{\infty} \beta_n(t) \sin (n\pi x),$$

where

$$\beta_n(t) = -\int_0^t e^{-(n\pi)^2 (t-\tau)} f_n(\tau)d\tau$$

and

$$f_n(\tau) = 2\int_0^1 f(x, \tau) \sin (n\pi x)dx$$

Finally, assuming that $\mu(t)$ and $\nu(t)$ are differentiable, we write

$$u^{III}(x, t) = v(x, t) + (1-x)\mu(t) + x\nu(t),$$

where $v(x, t)$ is the solution to the problem

$$v_{xx} = v_t + (1-x)\mu'(t) + x\nu'(t), \quad 0 < x < 1, \quad 0 < t \leq 1,$$

$$v(x, 0) = -(1-x)\mu(0) - x\nu(0), \quad 0 \leq x \leq 1,$$

$$v(0, t) = v(1, t) = 0, \quad 0 \leq t \leq t,$$

and observe that $v(x, t)$ may be determined by solving problems similar to (2) and (3) above.

There are several disadvantages to the above method, however. The most important disadvantage is that it is essentially restricted to partial differential equations with constant coefficients, and only in special cases can it be extended to partial differential equations with variable coefficients. Consequently, from a physical standpoint, one is restricted to treating very ideal and unreal problems. There are other minor annoyances as well. For example, the functions μ, ν and ϕ are obtained by measurements obtained at discrete points and then one defines the functions μ, ν and ϕ either by interpolation, least squares approximation, or some other techniques. However, one has no information how $\mu'(t)$ and $\nu'(t)$ are behaving and this information is needed. This problem can be circumvented by defining a Green's function, but it is difficult to work with numerically. Further, the coefficients $a_n = 2\int_0^1 \phi(x) \sin(n\pi x)dx$, etc., must be evaluated, usually numerically, for many values of n, and one must obtain estimates on the speed of convergence of the series.

It is apparent from this discussion that it would be advantageous

to have a general method for approximating solutions to partial differential equations which can be used for numerical purposes. Such a method is the so called "method of nets" or the "method of finite differences" which we shall describe in detail below. In recent years, however, so many different net methods have been proposed that it is not obvious which methods one should use. Extensive information about the relative merits of the methods is simply not available at this time and one is forced to rely on the current "folklore" or personal prejudices in choosing methods for use. It is the purpose of this thesis to investigate the relative merits of the more popular net methods which have been proposed for approximating solutions to parabolic partial differential equations from a "user's" standpoint. This is done by solving several sample problems by different methods, comparing the accuracy of the methods, the efficiency of the methods, and so on, and to try to conclude which methods are superior.

The basic idea underlying the net methods is the following: Construct a system of mesh or grid points by letting $I \geq 1$, $N \geq 1$ be integers and setting $h = 1/I$ and $k = 1/N$. Here h is the x step size and k is the t step size. Define the mesh or grid points $(x_i, t_n) = (ih, nk)$, $i = 0, 1, \ldots, I$, $n = 0, 1, \ldots, N$. Schematically, these points are the intersections of the sets of lines indicated in Figure 1.

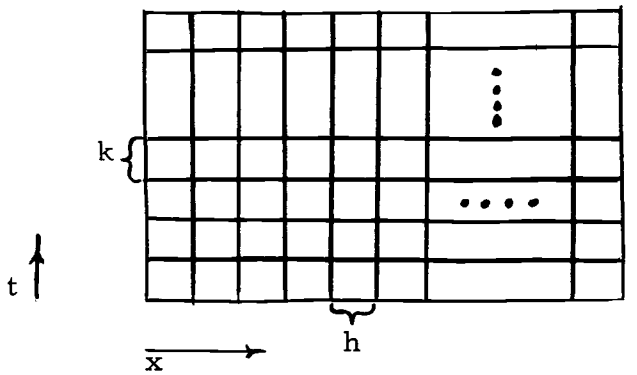


Figure 1.

The method of nets simply consists of algorithms for determining a function U defined on the grid points which in some sense approximates the values of u, the solution to the partial differential equation at the grid points (x_i, t_n).

The values of $U(x, t)$, which will approximate the solution $u(x, t)$ of the differential equation at the mesh points (x_i, t_n), are given at the points $(x_i, 0)$, $i = 0, 1, \ldots, I$; $(0, t_n)$, $(1, t_n)$, $n = 0, 1, \ldots, N$. To begin the computation, one uses the values of U at $(x_i, 0)$, $i = 0, 1, \ldots, I$ and perhaps at $(0, t_1)$ and $(1, t_1)$ to compute the values of U at (x_i, t_1), $i = 1, 2, \ldots, I-1$. Next the values $U(x_i, t_2)$ are computed and so on until values for U have been determined for all mesh points. Most methods use the values of U computed at (x_i, t_{n-1}), $i = 1, 2, \ldots, I-1$ to compute values of U at (x_i, t_n), $i = 1, 2, \ldots, I-1$. However, a few methods, known as multi-step methods, use values of U from two or more previous time levels, or time steps to compute values for U at (x_i, t_n), $i = 1, 2, \ldots, I-1$

for some $n > 1$.

Finally, it should be pointed out that although the considerations below are carried out for $0 \leq x \leq 1$, $0 \leq t \leq 1$, this involves no real loss of generality, since precisely the same techniques apply when $a \leq x \leq b$, $0 \leq t \leq T$, for $a < b$, $T > 0$, arbitrary real numbers.

II. NET METHODS: THEIR CONSTRUCTION AND GENERAL PROPERTIES

Construction of Net Methods

In this chapter a few of the best known methods will be constructed and their accuracy analyzed. First, construct the grid as suggested in Chapter I using h for the x step size and k for the t step size. The algorithms to be examined will determine the values of a function $U(x, t)$ defined at the mesh points (x_i, t_n), $i = 0, 1, 2, \ldots, I$; $n = 0, 1, 2, \ldots, N$. $U(x, t)$ will approximate the solution of the partial differential equation $u_{xx} = u_t + f(x, t)$ with given boundary and initial conditions. Henceforth, let $U_{i,n} = U(x_i, t_n) = U(ih, nk)$, and let $u_{i,n} = u(x_i, t_n)$. The error terms, for the various algorithms, are given in terms of $O(p(h, k))$, where the O is the Landau symbol having the usual meaning.

Using Taylor's formula, we find:

$$\frac{u_{i,n+1} - u_{i,n}}{k} = u_t(x_i, t_n) + O(k) \tag{2.1}$$

$$\frac{u_{i,n+1} - u_{i,n}}{k} = u_t(x_i, t_n) + \frac{k}{2} u_{tt}(x_i, t_n) + \frac{k^2}{6} u_{ttt}(x_i, t_n) + O(k^3) \tag{2.1a}$$

$$\frac{u_{i,n+1} - u_{i,n}}{k} = u_t(x_i, t_{n+1}) + O(k) \tag{2.2}$$

$$\frac{u_{i,n+1} - u_{i,n}}{k} = u_t(x_i, t_{n+\frac{1}{2}}) + O(k^2) \tag{2.3}$$

$$(2.4)\qquad \frac{u(x_i, t_{n+1})-u(x_i, t_n)}{2} = u(x_i, t_{n+\frac{1}{2}}) + O(k^2)$$

If u_{xx} is substituted for u in Formula 2.4, the result is

$$(2.5)\qquad \frac{u_{xx}(x_i, t_{n+1})-u_{xx}(x_i, t_n)}{2} = u_{xx}(x_i, t_{n+\frac{1}{2}}) + O(k^2)$$

$$(2.6)\qquad \frac{u_{i-1,n}-2u_{i,n}+u_{i+1,n}}{h^2} = u_{xx}(x_i, t_n) + O(h^2)$$

$$(2.6a)\qquad \frac{u_{i-1,n}-2u_{i,n}+u_{i+1,n}}{h^2} = u_{xx}(x_i, t_n) + \frac{h^2}{12} u_{xxxx}(x_i, t_n) + \frac{h^4}{360} u_{xxxxxx}(x_i, t_n) + O(h^6)$$

$$(2.7)\qquad \frac{u_{i-1,n+1}-2u_{i,n+1}+u_{i+1,n+1}}{h^2} = u_{xx}(x_i, t_{n+1}) + O(h^2)$$

From 2.5 and 2.6 we get

$$(2.8)\qquad u_{xx}(x_i, t_{n+\frac{1}{2}}) = \frac{1}{2}\left(\frac{u_{i-1,n}-2u_{i,n}+u_{i+1,n}}{h^2} + \frac{u_{i-1,n+1}-2u_{i,n+1}+u_{i+1,n+1}}{h^2}\right) + O(h^2+k^2)$$

We first construct the Classical Explicit Method using relations (2.1) and (2.6). Set

$$u_t(x_i, t_n) + f(x_i, t_n) = u_{xx}(x_i, t_n)$$

or

$$\frac{u_{i,n+1}-u_{i,n}}{k} + f(x_i, t_n) = \frac{u_{i-1,n}-2u_{i,n}+u_{i+1,n}}{h^2} + O(h^2+k)$$

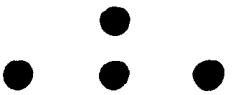
Then

$$u_{i,n+1} = \frac{k}{h^2}(u_{i-1,n}+u_{i+1,n}) + (1-2\frac{k}{h^2})u_{i,n} - kf(x_i, t_n) + O(h^2k+k^2)$$

Thus the algorithm to be used is:

$$(2.10) \qquad U_{i,n+1} = \lambda(U_{i-1,n}+U_{i+1,n}) + (1-2\lambda)U_{i,n} - kf(x_i, t_n)$$

where $\lambda = \frac{k}{h^2}$ is the mesh ratio and the local error $(u-U)$ is $O(h^2k+k^2)$. Formula (2.10) shows that the values of U at the three points (x_{i-1}, t_n), (x_i, t_n) and (x_{i+1}, t_n) are used to calculate the value of U at the point (x_i, t_{n+1}). This is represented schematically by the "stencil"



As we have just seen, the local error for the Classical Explicit Method is $O(h^2k+k^2)$. We now obtain a bound on the total error for any fixed time t. Following Saul'yev [6], we let $f(x, t) = 0$ and $u(0, t) = u(1, t) = 0$ and observe that the grid, as previously described, has $I-1$ interior mesh points at each time step. The $I-1$ values of U at time $t = nk$ can be regarded as a vector V_n. The calculation of V_{n+1} can then be regarded as premultiplication of V_n by the matrix A, that is an AV_n, where A is the

(I-1) x (I-1) matrix:

$$\begin{bmatrix} 1-2\lambda & \lambda & 0 \cdots & & 0 \\ \lambda & 1-2\lambda & \lambda & 0 \ \ldots & 0 \\ 0 & \lambda & 1-2\lambda & \lambda \ \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \\ 0 \ \cdots \ 0 & & \cdots & \lambda & 1-2\lambda \end{bmatrix}$$

Similarly, the error at time $t = nk$ can be regarded as the vector ϵ_n. Then $\epsilon_{n+1} = A\epsilon_n + r$, where the vector r satisfies $|r| = O(h^2k+k^2)$.

Let

$$\|\epsilon_n\| = \max_{1 \le i \le I-1} |\epsilon_{i,n}|,$$

and let

$$\|A\| = \max_{1 \le i \le I-1} \sum_{j=1}^{I-1} |a_{i,j}|.$$

Then $\|A\| = \lambda + |1-2\lambda| + \lambda$. If $2\lambda \le 1$, then $\|A\| = \lambda + (1-2\lambda) + \lambda = 1$. Thus, if $\lambda \le \frac{1}{2}$, then $\|A\| \le 1$. Next observe that $\|A\epsilon_n\| \le \|A\| \cdot \|\epsilon_n\|$.

This gives

$$\|\epsilon_{n+1}\| = \|A\epsilon_n + O(h^2k+k^2)\|$$

$$\le \|A\| \cdot \|\epsilon_n\| + O(h^2+k^2)$$

$$\le O(h^2k+k^2) + \|A\|(\|A\| \cdot \|\epsilon_{n-1}\| + O(h^2k+k^2))$$

Continuing the expansion gives:

$$\|\epsilon_{n+1}\| \leq O(h^2k+k^2) + O(h^2k+k^2)\|A\| + O(h^2k+k^2)\|A\|^2 + \ldots$$

$$= O(h^2k+k^2) \sum_{i=0}^{n} \|A\|^i \leq O(h^2k+k^2) \cdot (n+1)$$

$$= O(h^2+k),$$

Since

$$\|A\| \leq 1 \quad \text{and} \quad nk = t_n \leq T.$$

Thus, $\|\epsilon_{n+1}\| \leq O(h^2+k)$ where ϵ_{n+1} is the total error vector at time $t_{n+1} = (n+1)k$.

Observe that the total error is of the same order as the error in the approximating algorithm before the algorithm was multiplied by k to give $U_{i,n+1}$ explicitly. This is also typical of other net methods.

If in the construction of the Classical Explicit Method we had kept more terms from Taylor's formula, we would have obtained

$$(2.11) \qquad \frac{u_{i,n+1}-u_{i,n}}{k} + f(x_i,t_n) - \frac{u_{i-1,n}-2u_{i,n}+u_{i+1,n}}{h^2}$$

$$= u_t(x_i,t_n) + f(x_i,t_n) - u_{xx}(x_i,t_n) + [\frac{k}{2}u_{tt}(x_i,t_n) - \frac{h^2}{12}u_{xxxx}(x_i,t_n)]$$

$$+ [\frac{k^2}{6}u_{ttt}(x_i,t_n) - \frac{h^4}{360}u_{xxxxxx}(x_i,t_n)] + O(h^6+k^3)$$

Now assume $f(x, t) = 0$ so that $u_{xx} = u_t$. Then $u_{xxt} = u_{tt} = u_{txx} = u_{xxxx}$ so that if $\frac{k}{2} = \frac{h^2}{12}$ i.e., $\lambda = \frac{1}{6}$, then the first pair of terms in parenthesis vanish. The order of the error becomes $O(h^4+k^2)$. This is important because of all the methods tested for the heat equation, the Classical Explicit Method with $\lambda = \frac{1}{6}$ proved to be the most efficient. Of course, it may not always be convenient to set $\lambda = \frac{1}{6}$.

Observe that the increase in accuracy has been achieved by means of a cancellation effect between errors in terms of powers of $k = \Delta t$ and errors in terms of powers of $h = \Delta x$. Also observe that this effect depends on the fact that $u_{xx} = u_t$. This cancellation of errors occurs in certain other higher order methods, as well. On problems of type $u_{xx} = u_t + f(x, t)$ this cancellation does not take place, but as will be shown later, the algorithm can be modified to make it work to maintain the $O(h^4+k^2)$ accuracy. Finally, if $u_{xx} = u_t$, and if the first pair of terms in parentheses in Formula (2.11) can be removed by some means, such as approximation of $u_{tt}(x_i, t_n)$ and $u_{xxxx}(x_i, t_n)$ by finite differences, then setting $\frac{k^2}{6} = \frac{h^4}{360}$, i.e., $\lambda = \frac{1}{2\sqrt{15}}$ will cause the second pair of terms in parentheses to vanish. This is true because $u_{ttt} = u_{xxxxxx}$. Then the order of the error becomes $O(h^6+k^3)$. Testing proved that this order of accuracy is more difficult to achieve in practice than in theory. However, the greatest accuracy in practice, as well as in

theory, does result using this value of λ.

Next, let us construct the Classical Implicit Method using Formulas (2. 2) and (2. 7).

Set

$$u_t(x_i, t_{n+1}) + f(x_i, t_{n+1}) = u_{xx}(x_i, t_{n+1})$$

or

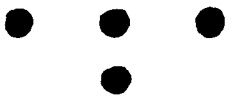
$$\frac{u_{i,n+1} - u_{i,n}}{k} + f(x_i, t_{n+1}) = \frac{u_{i-1,n+1} - 2u_{i,n+1} + u_{i+1,n+1}}{h^2} + O(h^2+k)$$

or

$$-\frac{k}{h^2}(u_{i-1,n+1} + u_{i+1,n+1}) + (1+2\frac{k}{h^2})u_{i,n+1} = u_{i,n} - kf(x_i, t_{n+1}) + O(h^2+k).$$

Set $\lambda = \frac{k}{h^2}$ to get

$$-\lambda(U_{i-1,n+1} + U_{i+1,n+1}) + (1+2\lambda)U_{i,n+1} = U_{i,n} - kf(x_i, t_{n+1}) + O(h^2+k) \tag{2.12}$$

which is of the form 

The values $U_{i,n+1}$; $i = 1, 2, \ldots, I-1$ cannot be computed explicitly from the values $U_{i,n}$; $i = 1, 2, \ldots, I-1$. They must be determined implicitly by solving a set of simultaneous linear equations. This is discussed later in detail.

Finally, let us construct the Crank-Nicholson Method using Formulas (2. 3) and (2. 8).

Set

$$u_t(x_i, t_{n+\frac{1}{2}}) + f(x_i, t_{n+\frac{1}{2}}) = u_{xx}(x_i, t_{n+\frac{1}{2}})$$

or

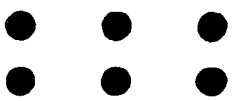
$$\frac{u_{i,n+1} - u_{i,n}}{k} + O(k^2) + f(x_i, t_{n+\frac{1}{2}})$$

$$= \frac{1}{2}\left[\frac{u_{i-1,n} - 2u_{i,n} + u_{i+1,n}}{h^2} + \frac{u_{i-1,n+1} - 2u_{i,n+1} + u_{i+1,n+1}}{h^2}\right] + O(h^2+k^2)$$

Rearranging and getting $\lambda = \frac{k}{h^2}$ gives the desired algorithm

$$(2.13) \qquad \frac{1}{2}\lambda(U_{i-1,n+1} + U_{i+1,n+1}) - (\lambda+1)U_{i,n+1}$$

$$= (\lambda-1)U_{i,n} - \frac{1}{2}\lambda(U_{i-1,n} + U_{i+1,n}) + kf(x_i, t_{n+\frac{1}{2}})$$

This is of the form 

This method, like the Classical Implicit Method, requires the implicit solution of a set of simultaneous linear equations.

Extension of Algorithms for Problems of Type $u_{xx} = u_t$ to Problems of Type $u_{xx} = u_t + f(x, t)$

The algorithms constructed in the previous section are simple extensions of algorithms given in most texts for the problem $u_{xx} = u_t$ to the problem $u_{xx} = u_t + f(x, t)$. The total error for the Classical Explicit Method is $O(h^2+k)$. When $\lambda = \frac{1}{6}$, and $f(x, t) = 0$, the order becomes $O(h^4+k^2)$. To obtain the $O(h^4+k^2)$ accuracy for

the case when $f(x, t) \neq 0$, we assume that the necessary partial derivatives are continuous, so that the following conditions hold:

$$u_{xxxx} = u_{xxt} + f_{xx} \tag{2.14}$$

$$u_{tt} = u_{xxt} - f_t \tag{2.15}$$

$$u_{xxxx} = u_{tt} + f_t + f_{xx} \tag{2.16}$$

Using formulas (2.14), (2.15) and (2.11) we get

$$\frac{u_{i,n+1}-u_{i,n}}{k} + f(x_i, t_n) - \frac{u_{i-1,n}-2u_{i,n}+u_{i+1,n}}{h^2} \tag{2.17}$$

$$= \left[\frac{k}{2}(u_{xxt}-f_t) - \frac{h^2}{12}(u_{xxt}+f_{xx})\right] + O(h^4+k^2).$$

Set

$$\lambda = \frac{1}{6}, \quad \text{i.e.,} \quad \frac{k}{2} = \frac{h^2}{12}$$

to get

$$\frac{u_{i,n+1}-u_{i,n}}{k} \tag{2.18}$$

$$= \frac{u_{i-1,n}-2u_{i,n}+u_{i+1,n}}{h^2} - f(x_i, t_n) - \frac{k}{2} f_t - \frac{h^2}{12} f_{xx} + O(h^4+k^2)$$

Rearranging and multiplying by k gives

$$u_{i,n+1} = \lambda(u_{i-1,n}+u_{i+1,n}) + (1-2\lambda)u_{i,n} \tag{2.19}$$

$$- k\left[f(x_i, t_n) + \frac{k}{2} f_t(x_i, t_n) + \frac{h^2}{12} f_{xx}(x_i, t_n)\right] + kO(h^4+k^2).$$

Next approximate f_t and f_{xx} by

$$f_t(x_i, t_n) = \frac{f(x_i, t_{n+1}) - f(x_i, t_n)}{k} + O(k) \tag{2.20}$$

$$f_{xx}(x_i, t_n) = \frac{f(x_{i-1}, t_n) - 2f(x_i, t_n) + f(x_{i+1}, t_n)}{h^2} + O(h^2) \tag{2.21}$$

Substituting (2.20) and (2.21) into (2.19) gives

$$\begin{aligned} u_{i,n+1} &= \lambda(u_{i-1,n} + u_{i+1,n}) + (1-2\lambda)u_{i,n} \\ &\quad - k\left[f(x_i, t_n) + \frac{1}{2}[f(x_i, t_{n+1}) - f(x_i, t_n)]\right. \\ &\quad \left. + \frac{1}{12}[f(x_{i-1}, t_n) - 2f(x_i, t_n) + f(x_{i+1}, t_n)]\right] + O(h^4 + k^2) \end{aligned}$$

and simplifying yields

$$\begin{aligned} u_{i,n+1} &= \lambda(u_{i-1,n} + u_{i+1,n}) + (1-2\lambda)u_{i,n} \\ &\quad - k\left[\frac{1}{3}f(x_i, t_n) + \frac{1}{12}[f(x_{i-1}, t_n) + f(x_{i+1}, t_n)] + \frac{1}{2}f(x_i, t_{n+1})\right] \\ &\quad + O(h^4 + k^2) \end{aligned} \tag{2.22}$$

Thus the algorithm to be used is

$$\begin{aligned} U_{i,n+1} &= \lambda(U_{i-1,n} + U_{i+1,n}) + (1-2\lambda)U_{i,n} \\ &\quad - k\left[\frac{1}{3}f(x_i, t_n) + \frac{1}{12}[f(x_{i-1}, t_n) + f(x_{i+1}, t_n)] + \frac{1}{2}f(x_i, t_{n+1})\right] \end{aligned} \tag{2.23}$$

In a similar manner it is possible to extend other higher order methods for use on problems of the type $u_{xx} = u_t + f(x, t)$ without lowering the order of the accuracy.

Convergence, Accuracy and Stability of Net Methods

Place a rectangular net of the type described in Chapter I over the rectangular region R. Let $U(x_i, t_n)$ represent the values of the solution of the difference equations at the point (x_i, t_n). Let $u(x, t)$ be the solution of the partial differential equation problem at the point (x, t). We shall say that the net method converges to the solution $u(x, t)$ on the net if $|u(x_i, t_n) - U(x_i, t_n)| \to 0$ uniformly at all grid points (x_i, t_n), $i = 1, 2, \ldots, I-1$, $n = 1, 2, \ldots, N-1$, as $h \to 0$ and $k \to 0$.

The order of the total error for most of the methods tested is $O(h^2+k)$ or $O(h^4+k^2)$. If $\lambda = \frac{k}{h^2}$ is kept constant as h and k are reduced, then $O(h^2+k)$ is equivalent to $O(h^2)$, and $O(h^4+k^2)$ is equivalent to $O(h^4)$. For this reason some texts give the error as $O(h^2)$ or $O(h^4)$. For a few of the methods, the error is of order $O(h^2+k^2)$. One question to be answered is whether methods supposedly of higher order, say $O(h^4+k^2)$, actually achieve this order of accuracy in practice. According to the test results, the answer is, yes, for methods of order $O(h^4+k^2)$ provided the function $u(x, t)$ has partial derivatives with respect to x and t of

sufficiently high order. It was not possible to get a higher order of accuracy than $O(h^4+k^2)$ even though in one case the order was theoretically $O(h^6+k^3)$. The testing was done using a Burroughs B5500 computer which uses 39 binary digits for the magnitude of a number. The maximum relative error for an arithmetic operation is 2^{-38}.

A net method is considered to be stable provided the errors introduced at a given time step do not increase at an unacceptable rate as the time increases. Otherwise, the method is said to be unstable. Generally speaking, most methods are stable or unstable depending on whether $\lambda = \frac{k}{h^2}$ is less than some constant which depends on the method. A few implicit methods are stable for all values of λ. Such methods are said to be unconditionally stable.

The Solution of the Simultaneous Linear Equations for Implicit Net Methods

The matrix corresponding to the set of linear equations to be solved is usually tridiagonal. It would be possible to solve the system by an iterative method. From the standpoint of efficiency, however, Gauss elimination is the best method known.

The set of equations is of the form

$$
\begin{aligned}
& b_1 x_1 + c_1 x_2 = N_1 \\
& a_2 x_1 + b_2 x_2 + c_2 x_3 = N_2 \\
& \vdots \qquad \vdots \qquad \vdots \\
& a_j x_{j-1} + b_j x_j + c_j x_{j+1} = N_j, \quad j = 2, \ldots, I-2 \\
& \vdots \qquad \vdots \qquad \vdots \\
& a_{I-1} x_{I-2} + b_{I-1} x_{I-1} = N_{I-1}
\end{aligned}
$$

The algorithm is as follows: Set

$$
\beta_1 = b_1
$$

$$
\beta_i = b_i - \frac{a_i c_{i-1}}{\beta_{i-1}}, \quad i = 2, 3, \ldots, I-1
$$

$$
D_1 = \frac{N_1}{\beta_1}, \quad D_i = \frac{N_i - a_i D_{i-1}}{\beta_i}, \quad i = 2, 3, \ldots, I-1
$$

This is followed by the back substitution

$$
x_{I-1} = D_{I-1}; \quad x_i = D_i - \frac{c_i D_{i+1}}{\beta_i}, \quad i = I-2, I-3, \ldots, 1
$$

For most implicit methods, the tridiagonal matrix is invariant. Only the right hand side N_j changes from one time step to the next. Therefore, the β_i's need to be calculated only once. This method is very efficient, and it has been shown that the roundoff errors do not grow unacceptably large (see [1] and [2]). The number of operation required to solve the system of $I-1$ equations is proportional

to I-1. Finally, the I-1 dimensional tridiagonal matrix can be stored in a 3 x (I-1) dimensioned array, so storage is not a serious problem. This algorithm was used for testing the implicit methods.

Speed of Net Methods

The calculation of the values $U_{i, n+1}$; i = 1, 2, . . . , I-1 from the values $U_{i, n}$; i = 0, 1, 2, . . . , I requires I-1 explicit calculations for explicit methods. For implicit methods, the solution of I-1 linear equations is required. If one uses the algorithm given in the previous section, then the number of operations, and hence the amount of computer time required, is proportional to I-1. A certain amount of bookkeeping and other miscellaneous operations will be necessary regardless of the size of h. Therefore, if h is large, the computer time required will be somewhat greater than expected when compared to the computer time required for small values of h. For both explicit and implicit methods, then, the computer time is roughly proportional to I-1, where I = 1/h. Since the computer time required is the same for all time steps, the total computer time is proportional to the number of time steps N. Combining the two results shows that the computer time required is proportional to (I-1) x N, or, for large I and N, proportional to I x N, where $I = \frac{1}{h}$ and $N = \frac{1}{k}$. If h is divided by R, the number of x steps and the computer time required are multiplied by R.

Similarly, if the t step size k is divided by R, the number of t steps and the computer time required are multiplied by R. For a method with error of order $O(h^2+k^2)$, it is necessary to divide h by R and k by R in order to divide the error by R^2. This multiplies the computer time required by R^2. Therefore, to divide the error by R, it is necessary to multiply the computer time required by R. If the error has order $O(h^2+k)$, it is necessary to divide h by R and k by R^2 in order to divide the error by R^2. This multiplies the number of x steps by R, the number of t steps by R^2 and the computer time by R^3. Therefore, to divide the error by R, the computer time must be multiplied by $R^{3/2}$. Using the same kind of reasoning for methods with errors of order $O(h^4+k^2)$, the computer time must be multiplied by R^3 in order to divide the error by R^4. Therefore, to divide the error by R, the computer time must be multiplied by $R^{3/4}$.

This analysis gives an idea of the time that can be saved when using higher order methods to achieve a certain required accuracy. These considerations were used in trying to compare the methods for efficiency.

Efficiency of Net Methods

Efficiency refers to the amount of computer time required by different methods to achieve the same accuracy. Efficiency depends

on the order of the error of the method, the coefficients of the error terms (which in general depend on derivatives and are unknown), and upon the number of calculations required for the particular method.

III. THE SCOPE AND METHODS OF THE THESIS

The Partial Differential Equation Problems Used for Testing

After much consideration, we have chosen three problems for testing. Problems one and two are of the form $u_{xx} = u_t$. Problem three is of the form $u_{xx} = u_t + f(x, t)$. The unit square was used for the domain of the function $u(x, t)$. We considered the following problembs.

Problem One

$$u_{xx} = u_t, \quad 0 < x < 1, \quad 0 < t \leq 1$$

$$u(x, 0) = x(2-x)$$

$$u(0, t) = 0, \quad u(1, t) = 1$$

Referring to the discussion of explicit solutions to the problem in the Introduction, it is seen that

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

where

$$a_n = 2\int_0^1 x(2-x) \sin(n\pi x)dx$$

$$= \frac{4}{(n\pi)^3}[1-(-1)^n]$$

The series converges very fast, and the results from the computer summation are more accurate than the eleven decimal digits used for comparison.

Problem Two

$$u_{xx} = u_t, \quad 0 < x < 1, \quad 0 < t \leq 1$$

$$u(0, t) = (0.3)e^{-5t} \cos(27t) + (0.7)$$

$$u(1, t) = e^{-3t}, \quad u(x, 0) = 1$$

Let

$$\mu(t) = (0.3)e^{-5t} \cos(27t) + (0.7)$$

$$\nu(t) = e^{-3t}$$

and

$$w(x, t) = (1-x) \cdot \mu(t) + x \cdot \nu(t)$$

and form

$$v(x, t) = u(x, t) - w(x, t)$$

Then continuing as in Chapter I,

$$v_{xx} = v_t + (1-x)\mu'(t) + x\nu'(t),$$

$$v(0, t) = v(1, t) = 0$$

and

$$v(x, 0) = u(x, 0) - [(1-x) \cdot \mu(0) + x \cdot \nu(0)]$$

or

$$v(x, 0) = 1 - [(1-x) \cdot 1 + x \cdot 1] = 0$$

Therefore

$$v(x, t) = v_t + f(x, t),$$

where

$$f(x, t) = (1-x) \cdot \mu'(t) + x \cdot \nu'(t)$$

From Chapter I,

$$v(x, t) = \sum_{n=1}^{\infty} a_n(t) \sin (n\pi x)$$

where

$$a_n(t) = -\int_0^t e^{-(n\pi)^2(t-\tau)} \beta_n(\tau) d\tau,$$

with

$$\beta_n(\tau) = 2\int_0^1 f(x, \tau) \sin (n\pi x) dx.$$

For this particular problem

$$f(x, \tau) = (1-x)[-5e^{-5\tau} \cos (27\tau) - 27e^{-5\tau} \sin (27\tau)](-3xe^{-3\tau}).$$

After considerable manipulation

$$a_n(t) = \frac{2}{n\pi}\left[\frac{(0.3)e^{-5t}}{[(n\pi)^2-5]^2+729}[5[((n\pi)^2-5)\cos(27t)+27\sin(27t)] + 27[((n\pi)^2-5)\sin(27t)-27\cos(27t)]] - \frac{3(-1)^n e^{-3t}}{[(n\pi)^2-3]} - \frac{(0.3)e^{-(n\pi)^2 t}}{[((n\pi)^2-5)^2+729]}[5((n\pi)^2-5)-729] + \frac{3(-1)^n e^{-(n\pi)^2 t}}{((n\pi)^2-3)}\right].$$

Finally, after $v(x, t)$ has been calculated using the above expression for $a_n(t)$, it must be added to $w(x, t) = (1-x)[(0.3)e^{-5t}\cos(27t)+(0.7)] + xe^{-3t}$. The error in computing $u(x, t)$ is the error resulting from the truncation of the infinite sum

$$\sum_{n=1}^{\infty} a_n(t) \sin(n\pi x)$$

to a finite number of terms. As n becomes large, the last two terms in parentheses in the expression for $a_n(t)$ become insignificant. For $n \geq 200$ they can be completely disregarded. Sixteen hundred terms of the infinite sum were used.

Multiply each of the first two terms within the parentheses by $\frac{2}{n\pi}$ outside the parentheses and denote the two products as $a_n^{(1)}(t)$ and $a_n^{(2)}(t)$ respectively. It can be shown that

$$\sum_{n=1601}^{\infty} a_n^{(1)}(t) \sin(n\pi x) < 2e^{-5t} \times 10^{-9}$$

and

$$\sum_{n=1601}^{\infty} a_n^{(2)}(t) \sin(n\pi x) < (0.7)e^{-3t} \times 10^{-9}$$

for $x = (0.1, (0.2), \ldots, (0.9)$. Each sum can be shown to behave like an alternating series. Because of this fact, the error is smaller than might be expected. Using these estimates and proceeding in a

standard manner, we find that for $t = .1, .5$ and 1.0, the error caused by truncation of the series after 1600 terms gives us the table:

t	error bound
.1	2.3×10^{-9}
.5	9×10^{-10}
1.0	6×10^{-10}

Problem Three

$$u_{xx} = u_t + x(1-x)e^{-t}[\cos(t^2)+2t\sin(t^2)] - 2\cos(t^2)e^{-t}$$

$$u(0, t) = 0, \quad u(1, t) = 0, \quad u(x, 0) = x(1-x)$$

The solution is

$$u(x, t) = \cos(t^2)e^{-t}x(1-x)$$

My reason for choosing these three problems follows: Problem one gives a function $u(x, t)$ which has no great fluctuations and is, in fact, rather smooth. On the other hand, problem two gives a function which oscillates rapidly near the boundary $x = 0$. It should put more stress on the various methods.

Problem three was chosen because its solution satisfies $u_{xx} = u_t + f(x, t)$ where $f(x, t) \neq 0$, which is a somewhat more general problem than problems one and two. Most of the methods were extended to apply to this type of problem. None of the problems have

rapidly varying initial conditions. It was found that variations here are quickly damped out, and after a very short time, they were no longer felt.

Procedures Used During Testing

The unit square was covered by a net of 100 squares of length (0. 1) and width (0. 1). This gives 90 grid points at which the function $U(x, t)$ was to be calculated. Also, there are 31 boundary points. The various methods were run on a Burroughs 5500 computer. The program, written in ALGOL, was run using various time and distance step sizes. Only the values for $U(x, t)$ at the 90 grid points and at the 31 boundary points were printed. Also cards were punched with the calculated values at the time values . 1, . 2, . 5 and 1. 0. Thus $9 \times 4 = 36$ computed values were punched onto cards. In some cases, the specified grid points were not exactly reached because of the time step size. In such cases, the printouts and punched cards were output at time values as close as possible to the specified grid points. Eleven significant figures were punched for each calculated value of $U(x, t)$. The amount of computer time required for each run was also printed. No one else used the B5500 at the same time, so that the timing results would be accurate. The printouts were analyzed and the stability of the methods verified for various ratios of $\frac{k}{h^2}$. Then, the punched cards were used as input to a CDC 3600 computer

program, written in double precision FORTRAN. This program calculated the correct values for the function and compared these values with the values on punched cards which had been computed by one of the methods to be analyzed. The errors in the values on the punched cards were printed and analyzed. Finally, the maximum errors for each run were recorded. From these results, the methods were compared according to their efficiency.

Limitations of the Results

Only three problems were used to test the various methods. Problems one and two are of the type, $u_{xx} = u_t$. The solution to problem one is a well behaved function. The solution to problem two is a function which fluctuates rapidly with changes in the time at points near the line $x = 0$. The methods performed similarly on these two problems, and it is likely that they would perform about the same on most problems of this type.

It was necessary to extend the various algorithms for use on problem three, of type $u_{xx} = u_t + f(x, t)$ where $f(x, t) \neq 0$. Most of the methods required only simple and obvious modification. A few methods were difficult to extend, and some of these were not tried on problem three. Original attempts to extend the methods of higher order in the obvious way caused the order of accuracy to be lowered. Subsequently, three of the higher order methods were extended in

such a way that the order of accuracy was not reduced. Therefore, fewer methods were tried on problem three than on the other two problems. This, and the fact that only one problem of this type was tested, tend to make the results less conclusive for problem three.

However, all the results taken together, do seem to give a pretty clear picture to the user of what to expect and how the methods are likely to perform.

IV. THE METHODS TESTED

Method 1. Classical Explicit Method

This is the simplest net method known. In general, the order of the error is $O(h^2+k)$. If the problem is of the type $u_{xx} = u_t$, then by setting $\lambda = \frac{k}{h^2} = \frac{1}{6}$, it is possible to increase this order to $O(h^4+k^2)$ (see Chapter II). Under these conditions it is the most efficient of all the net methods tested. It is also the simplest to program. For problems of type $u_{xx} = u_t$ it is highly recommended. It should not usually be inconvenient to set $k = \frac{1}{6}h^2$. If it is inconvenient for some reason, then there are better methods to use.

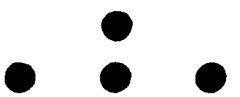
The simplest extension of the algorithm of method 1 for use on problems of type $u_{xx} = u_t + f(x, t)$ is given in Formula (4. 1) below.

When this algorithm is used on problems of that type, setting $\lambda = \frac{1}{6}$ no longer increases the order of the accuracy from $O(h^2+k)$ to $O(h^4+k^2)$. It is possible, however, to extend the method in such a way that $O(h^4+k^2)$ is the order of accuracy if λ is set to $\frac{1}{6}$. The details are given in Chapter II under Extension of Algorithms for Problems of Type $u_{xx} = u_t + f(x, t)$ to Problems of Type $u_{xx} = u_t$ where Formula (2. 23) is constructed. This algorithm, referred to as Method 1*, is reproduced here as Formula (4. 2). Using this algorithm with $\lambda = \frac{1}{6}$, the method compared favorably with the other two higher order methods tested on problem three. It did not

outperform Method seven, however, and it is not clear which of the two methods is better. Method 1 is stable for $\lambda \leq \frac{1}{2}$. Since the method is more efficient for smaller values of λ, this is not as great a disadvantage as is sometimes believed.

In summary, method 1 is recommended for problems of type $u_{xx} = u_t$ and Method 1* with $\lambda = \frac{1}{6}$ performs well on problems of type $u_{xx} = u_t + f(x, t)$. On more general parabolic equations it will not usually be possible to achieve an increase in accuracy by tricks such as setting $\lambda = \frac{1}{6}$. Also, the stability restriction on λ may be troublesome.

Algorithms:



(4. 1) Method 1.

$$U_{i, n+1} = \lambda(U_{i-1, n}+U_{i+1, n}) + (1-2\lambda)U_{i, n} - kf(x_i, t_n)$$

(4. 2) Method 1*

$$U_{i, n+1} = \lambda(U_{i-1, n}+U_{i+1, n}) + (1-2\lambda)U_{i, n}$$

$$- k[\frac{1}{3}f(x_i, t_n)+\frac{1}{12}[f(x_{i-1}, t_n)+f(x_{i+1}, t_n)]+\frac{1}{2}f(x_i, t_{n+1})]$$

Method 2. Classical Implicit Method

This is the simplest of the so-called implicit methods. As with all implicit methods, a set of linear algebraic equations must be solved at each time step. The order of the error is $O(h^2+k)$. The

only real advantage of the method is its unconditional stability. During the testing, $\lambda = \frac{k}{h^2}$ was set to 5.0 and to 1.0; both are rather large. Undoubtedly, greater efficiency would be achieved with smaller values of λ. However, the error is of the same order as that of the Classical Explicit Method, and it is slower than the latter method. The method seems to have little to recommend it.

Algorithm: ● ● ● ●

(4.3) $$-\lambda(U_{i-1,\,n+1}+U_{i+1,\,n+1}) + (1+2\lambda)U_{i,\,n+1} = U_{i,\,n} - kf(x_i, t_{n+1})$$

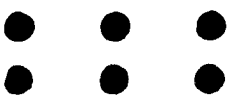
Method 3. Crank-Nicholson Method

The Crank-Nicholson Method is an implicit method requiring the solution of a set of simultaneous linear equations at each time step. It is a very well known and widely used method. The method has an error of order $O(h^2+k^2)$. Furthermore, this order does not depend on the fact that $u_{xx} = u_t$; therefore, this order of accuracy is maintained on problems of type $u_{xx} = u_t + f(x, t)$. On problems of type $u_{xx} = u_t$ and $u_{xx} = u_t + f(x, t)$, which are the subject of this thesis, the method is fairly good. Nevertheless, if $u(x, t)$ has partial derivatives of sufficiently high order, the test results show that we can get better efficiency with higher order accurate methods such

as Method 6 and Method 1* with $\lambda = \frac{1}{6}$. Therefore, if $u(x, t)$ is sufficiently smooth, and computer time is important, there are better methods. Otherwise, it is an excellent method to use. On more general parabolic partial differential equation problems, where it is difficult to develop higher order methods, it is of much value because of its order of accuracy and its unconditional stability.

Method 3 is unconditionally stable. On a method of order $O(h^2+k^2)$ such as this one, it is the quantity $\frac{k}{h}$ rather than $\frac{k}{h^2}$ which assumes prime importance (provided the method is unconditionally stable). Choose a suitable value for $\frac{k}{h}$ and keep it constant as the step sizes are reduced.

Algorithm:



$$\frac{\lambda}{2}(U_{i-1,\,n+1}+U_{i+1,\,n+1}) - (1+\lambda)U_{i,\,n+1}$$

$$= -\frac{\lambda}{2}(U_{i-1,\,n}+U_{i+1,\,n}) + (\lambda-1)U_{i,\,n} + kf(x_i, t_{n+\frac{1}{2}}) \tag{4.4}$$

Method 4. Alternate Explicit and Implicit Methods

This method, which is described in Saul'yev [6], involves the alternate use of Methods 1 and 2 (Classical Explicit and Classical Implicit Methods). The error is of order $O(h^2+k^2)$. The method is unconditionally stable. The method behaves much like the Crank-Nicholson Method. In fact, for $\frac{k}{h} = .1$ it was more efficient on

problems one and two, but a little less efficient on problem three. When the problem is generalized to $u_{xx} = u_t + f(x, t)$ the order of the error does not decrease. There seems to be no reason to choose Method 4 over the Crank-Nicholson Method which has the same order of accuracy. For efficiency, it is possible to do better with higher order methods.

Algorithm: Use Methods 1 and 2 alternately. That is, use the Classical Explicit Method for odd numbered time steps, and the Classical Implicit Method for even numbered time steps.

Method 5

Method 5 is the result of adding Methods 10 and 11 (see below) and transposing all terms from the new time step $(n+1)k$ to the same side of the equation. The method appears in Saul'yev [6] as Formula (8.16). Method 5 is an implicit method which requires the solution of a set of simultaneous linear equations. Method 5 makes use of a parameter α, $(0 \leq \alpha \leq 1)$. The choice of this parameter has considerable effect on the performance of the method. The method is stable for $\lambda = \frac{k}{h^2} \leq \frac{1}{2(1-\alpha)}$, and the accuracy also depends on the value of α. For problems of type $u_{xx} = u_t$ the algorithm is as follows:

$$(4.5)\qquad -\alpha(U_{i-1,n+1}+U_{i+1,n+1}) + 2(\omega+\alpha)U_{i,n+1}$$

$$-(2-\alpha)(U_{i-1,n}+U_{i+1,n}) + 2(2-\omega-\alpha)U_{i,n} = 0,$$

where

$$\omega = 1/\lambda \quad \text{and} \quad 0 \leq \alpha \leq 1.$$

We shall restrict our attention for this method to the problem $u_{xx} = u_t$. Formula (4.5) is suitable for computation, but in order to analyze Method 5 further, divide that algorithm by $2h^2$ to get

$$-\frac{\alpha}{2h^2}(U_{i-1,n+1}+U_{i+1,n+1}) + (\frac{1}{k}+\frac{\alpha}{h^2})U_{i,n+1}$$

$$-(\frac{1}{h^2}-\frac{\alpha}{2h^2})(U_{i-1,n}+U_{i+1,n}) + (\frac{2}{h^2}-\frac{1}{k}-\frac{\alpha}{h^2})U_{i,n} = 0$$

Rearrangement gives

$$\frac{U_{i,n+1}-U_{i,n}}{k} - \frac{U_{i+1,n}-2U_{i,n}+U_{i-1,n}}{h^2}$$

$$+\frac{\alpha}{2}(\frac{U_{i+1,n}-2U_{i,n}+U_{i-1,n}}{h^2}) - \frac{\alpha}{2}(\frac{U_{i+1,n+1}-2U_{i,n+1}+U_{i-1,n+1}}{h^2}) = 0$$

If we take the limit of this expression as $h \to 0$, $k \to 0$, the result is

$$\frac{\partial u}{\partial t}(x_i, t_n) - \frac{\partial^2 u}{\partial x^2}(x_i, t_n) + \frac{\alpha}{2}\frac{\partial^2 u}{\partial x^2}(x_i, t_n) - \frac{\alpha}{2}\frac{\partial^2 u}{\partial x^2}(x_i, t_{n+1}) = 0$$

This breakdown shows the true nature of Method 5.

$$\frac{\partial^2 u}{\partial x^2}(x_i, t_{n+1})$$

is subtracted from

$$\frac{\partial^2 u}{\partial x^2}(x_i, t_n)$$

and the result multiplied by $\frac{a}{2}$. a is adjusted to cancel errors in the approximations to the equation

$$\frac{\partial u}{\partial t}(x_i, t_n) - \frac{\partial^2 u}{\partial x^2}(x_i, t_n) = 0.$$

On page 93, Saul'yev [6] gives a formula for the error term for Method t (Formula 8.16). Tedious calculation gave the following expression for the error:

$$\text{(4.6)} \qquad \text{Error} = \left(\frac{h^2}{12} - \frac{k}{2} + \frac{ak}{2}\right)\frac{\partial^2 u}{\partial t^2}(x_i, t_n)$$

$$+ \left(\frac{h^4}{360} - \frac{k^2}{6} + \frac{ak^2}{4} + \frac{ah^2k}{24}\right)\frac{\partial^3 u}{\partial t^3}(x_i, t_n) + O(h^6 + k^3 + h^4k^2)$$

Saul'yev's formula gives the error in terms of

$$\frac{\partial^2 u}{\partial t^2}(x_i, t_{n+1}) \quad \text{and} \quad \frac{\partial^3 u}{\partial t^3}(x_i, t_{n+1}).$$

Otherwise, the first term is the same. Setting $a = 1 - \frac{\omega}{6}$, where $\omega = \frac{h^2}{k}$, causes the terms within the first pair of parentheses to

vanish. This increases the order of accuracy from $O(h^2+k)$ to $O(h^4+k^2)$. The normal error for Method 5 is $O(h^2+k)$. The testing verified these facts.

Next, with α still set to $1 - \frac{\omega}{6}$, set $\omega = 2\sqrt{5}$. This causes the terms within the second pair of parentheses to vanish in Formula (4. 6) as well as in Saul'yev's version of the error formula. This increases the order of accuracy to $O(h^6+k^3)$. Henceforth, with $\alpha = 1 - \frac{\omega}{6}$, $\omega \neq 2\sqrt{5}$, Method 5 will be referred to as Method 5*. With $\alpha = 1 - \frac{\omega}{6}$, $\omega = 2\sqrt{5}$, Method 5 will be referred to as Method 5**.

Unfortunately, computer runs did not show an accuracy of $O(h^6+k^3)$ for Method 5**. The order was $O(h^4+k^2)$, the same as for Method 5*; however Method 5** was more accurate and efficient. Perhaps the error in the computer representation of $\sqrt{5}$ as well as roundoff errors, reduced the order of accuracy. Single precision arithmetic was used. Each arithmetic operation has a maximum relative error of 2^{-38}. Method 5 is a good method to use on problems of type $u_{xx} = u_t$. With $\alpha = 1 - \frac{\omega}{6}$, $\omega = 2\sqrt{5}$ (Method 5**), it is surpassed in efficiency only by Method 1 with $\lambda = \frac{1}{6}$. In practice it would not usually be convenient to set $\frac{h^2}{k} = \omega = 2\sqrt{5}$ (or rather the computer's closest approximation to that value). For maximum efficiency $\omega = \frac{h^2}{k}$ should be set as close as possible to that value, however. With $\alpha = 1 - \frac{\omega}{6}$ (Method 5*), the method is equivalent to

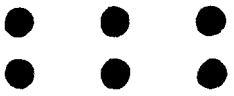
Method 6. Since the form of that algorithm is a little simpler, it may as well be used. Its efficiency is almost the same. Of course Method 1 with $\lambda = \frac{1}{6}$ is better than either of these methods on problems of type $u_{xx} = u_t$. Method 6 is also easy to extend to problems of type $u_{xx} = u_x + f(x, t)$, so because of this and for several other reasons, it is advantageous to choose it instead of Method 5.

Method 6

Method 6 is an implicit method which is unconditionally stable. The algorithm is discussed in [1] and [5] for problems of type $u_{xx} = u_t$, and the accuracy is of order $O(h^4+k^2)$. It is possible to extend the method in such a way that the method has the same order of accuracy for problems of the type $u_{xx} = u_t + f(x, t)$. The algorithm, as presented in the two references, is equivalent to Method 5 (Formula (8.16) in Saul'yev [6]), with $\alpha = 1 - \frac{\omega}{6}$. On problems of type $u_{xx} = u_t$ Method 6 should achieve maximum efficiency with $\lambda = \frac{k}{h^2} = \frac{1}{2\sqrt{5}}$ although this value for λ was not tried. Like Method 5, Method 6 is an excellent one as far as efficiency is concerned, and it is a little simpler to program than Method 5. On problems one and two of the type $u_{xx} = u_t$, the smallest value of λ tested was (0.5). In view of its equivalence to Method 5 and its nearly equal speed, it is clear from the test results for Method 5, that setting $\lambda = \frac{1}{2\sqrt{5}}$ would cause Method 6 to be as efficient as any method tested except

for Method 1 with $\lambda = \frac{1}{6}$ (Classical Explicit).

On problem three of type $u_{xx} = u_t + f(x, t)$ it performed as well as any of the methods tested. Only one setting of λ was tried on problem three ($\lambda = .25$). This should be a good setting of λ although it is not clear that $\lambda = \frac{1}{2\sqrt{5}}$ is necessarily optimum on problems of this type. The extension of Method 6 for use on this problem was especially simple. On problems of the type $u_{xx} = u_t + f(x, t)$, small values for λ seem to give the most efficiency for Method 6, and indeed this holds true for all other methods tested with accuracy of order $O(h^4+k^2)$ or $O(h^2+k)$. Therefore, to achieve a high degree of accuracy without losing efficiency, it is necessary to reduce k much faster than h. For this reason, the unconditional stability of Method 6 cannot be fully exploited. The method would be just as useful on problems of type $u_{xx} = u_t + f(x, t)$ if it were only stable for values of $\lambda \leq c$ for some constant $c \geq 1$. (On more general problems, stability considerations can assume more importance, however.) The lesson here is that stability restrictions on problems of type $u_{xx} = u_t$ and $u_{xx} = u_t + f(x, t)$ are important only when considered together with the order of accuracy.

Algorithm: 

As presented in Douglas [1] for use on problems of type $u_{xx} = u_t$, the algorithm is as follows:

$$\frac{U_{i,n+1}-U_{i,n}}{k} = \frac{1}{2}\left(1-\frac{1}{6\lambda}\right)[\Delta^2 x U_{i,n+1}] + \frac{1}{2}\left(1+\frac{1}{6\lambda}\right)[\Delta^2 x U_{i,n}]$$

where

$$\lambda = \frac{k}{h^2}\ ; \quad \Delta^2 x U_{i,n} = \frac{U_{i+1,n}-2U_{i,n}+U_{i-1,n}}{h^2}$$

The following algorithm is in a more usable form, and it is to be used on problems of type $u_{xx} = u_t + f(x, t)$.

$$(4.7)\qquad \left(\frac{5}{6}+\lambda\right)U_{i,n+1}+\left(\frac{1}{12}-\frac{\lambda}{2}\right)(U_{i-1,n+1}+U_{i+1,n+1})$$

$$= \left(\frac{5}{6}-\lambda\right)U_{i,n} + \left(\frac{1}{12}+\frac{\lambda}{2}\right)(U_{i+1,n}-U_{i-1,n})$$

$$- k\left[f(x_i, t_{n+\frac{1}{2}}) + \frac{h^2}{12} f_{xx}(x_i, t_{n+\frac{1}{2}})\right]$$

Since explicit calculation of $f_{xx}(x_i, t_{n+\frac{1}{2}})$ is undesirable, it can be approximated by

$$\frac{f(x_{i+1}, t_{n+\frac{1}{2}})-2f(x_i, t_{n+\frac{1}{2}})+f(x_{i-1}, t_{n+\frac{1}{2}})}{h^2}$$

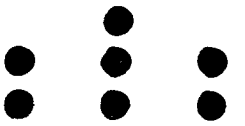
with error of order $O(h^2)$. This gives the following algorithm which is satisfactory to use for actual computation:

$$(4.8)\qquad \left(\frac{5}{6}+\lambda\right)U_{i,n+1} + \left(\frac{1}{12}-\frac{\lambda}{2}\right)(U_{i-1,n+1}+U_{i+1,n+1})$$

$$= \left(\frac{5}{6}-\lambda\right)U_{i,n} + \left(\frac{1}{12}+\frac{\lambda}{2}\right)(U_{i+1,n}-U_{i-1,n})$$

$$- k\left[\frac{5}{6}f(x_i, t_{n+\frac{1}{2}}) + \frac{1}{12}(f(x_{i-1}, t_{n+\frac{1}{2}}) + f(x_{i+1}, t_{n+\frac{1}{2}}))\right]$$

This is an implicit method and requires the solution of a set of simultaneous linear equations.

Method 7 

Method 7 is the only multi-step method tested. It appears as Formula (8. 30 in Saul'yev [6]. The method is explicit and is stable for $\lambda = \frac{k}{h^2} \leq \frac{1}{3}$. For the moment, let us restrict ourselves to problems of type $u_{xx} = u_t$. Expansion by Taylor series gives

$$\frac{u_{i,n+1} - u_{i,n}}{k} - \frac{u_{i-1,n} - 2u_{i,n} + u_{i+1,n}}{h^2}$$

$$= u_t(x_i, t_n) - u_{xx}(x_i, t_n) + \left(\frac{k}{2} u_{tt}(x_i, t_n) - \frac{h^2}{12} u_{xxxx}(x_i, t_n)\right)$$

$$+ \left(\frac{k^2}{6} u_{ttt}(x_i, t_n) - \frac{h^4}{360} u_{xxxxxx}(x_i, t_n)\right) + O(h^6 + k^3)$$

Since $u_{xx} = u_t$, Method 7 approximates $u_{tt} = u_{xxxx} = u_{xxt}$ by

$$\frac{\Delta^2_x U_{i,n+1} - \Delta^2_x u_{i,n}}{k}$$

with the error of order $O(h^2 + k)$. Here

$$\Delta^2_x U_{i,n} = \frac{u_{i+1,n} - 2u_{i,n} + u_{i-1,n}}{h^2} .$$

Using this approximation the sum of the two terms within the first set

of parentheses is set to zero. If u_{tt} and u_{xxxx} in the Taylor expansion are approximated in this manner, the resulting error is of the form $\frac{k}{2} \cdot O(h^2+k) - \frac{h^2}{12} \cdot O(h^2+k)$. If $\lambda = \frac{k}{h^2}$ is set equal to $\frac{1}{2\sqrt{15}}$, then the next term of the Taylor series

$$(\frac{k^2}{6} u_{ttt}(x_i, t_n) - \frac{h^4}{360} u_{xxxxxx}(x_i, t_n))$$

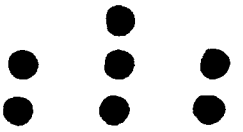
vanishes. For this reason, it would be expected that setting $\lambda = \frac{1}{2\sqrt{15}}$ would result in maximum efficiency. Of course, this irrational number can never be represented exactly on a computer, and it would be inconvenient to use such a value for λ even if it were possible. Values of $\lambda = \frac{1}{8}$ and $\lambda = \frac{1}{6}$ were tried, and the method performed efficiently for these settings of λ as expected. ($\frac{1}{8}$ is close to $\frac{1}{2\sqrt{15}}$.) This method gave good results on problems one and two of type $u_{xx} = u_t$. Here, Method 7 competed on equal terms with Methods 5 and 6, yielding only slightly in efficiency to Method 1 with $\lambda = \frac{1}{6}$. All four of these methods have accuracy of order $O(h^4+k^2)$.

Method 7 was extended for use on problems of type $u_{xx} = u_t + f(x, t)$. The order of accuracy remained the same, $O(h^4+k^2)$. The manner in which this was done is very similar to the analogous extension of Method 1 to Method 1* given in Chapter II. Although the order of Method 7 remained $O(h^4+k^2)$, Method 6 and Method 1*, with $\lambda = \frac{1}{6}$, were both more efficient on problem three

which is of type $u_{xx} = u_t + f(x, t)$.

Two definite drawbacks to Method 7 and other multi-step methods are the increased programming required for bookkeeping the multiple time steps, and the need for a "starter" method to calculate the values of $U_{i,1}$, $i = 1, 2, \ldots, I-1$ (i.e., the first time step). This programming difficulty is small, and since Method 7 is explicit in nature, no simultaneous linear equations need to be solved. Method 5* (Method 5 with $\alpha = -\frac{\omega}{6}$) was used for the starter method. Its order of accuracy is $O(h^4+k^2)$, the same as for Method 7. Method 6 would have been a little better to use as the starter method because it is slightly less complicated. To maintain the $O(h^4+k^2)$ accuracy, it is necessary to use a starter method with the same order of accuracy as the multip-step method. It is necessary to program both the multi-step method and the starter method, and in this case the starter method was implicit.

In summary, Method 7 is a good method as far as efficiency is concerned, but it is unstable for $\lambda > \frac{1}{3}$, and it has the disadvantage of being a multi-step method; therefore, a starter method is required and the programming is a little more involved. There seems to be no reason to choose it over Method 6 or Method 1, with $\lambda = \frac{1}{6}$.

Algorithm: 

$$(4.9)\quad U_{i,n+1} = (\frac{7}{6} - 3\lambda)U_{i,n} + [(3\lambda - \frac{1}{6})(U_{i-1,n} + U_{i+1,n})$$

$$- (\lambda - \frac{1}{6})(U_{i-1,n-1} - 2U_{i,n-1} + U_{i+1,n-1})]/2$$

$$- k[f(x_i, t_n) + \frac{h^2}{12} f_{xx}(x_i, t_n) + \frac{k}{2} f_t(x_i, t_n)]$$

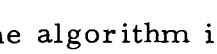
Since it is usually not practical to calculate f_{xx} and f_t explicitly, they can be approximated by formulas similar to (2.6) and (2.1) in Chapter II without reducing the accuracy order of Method 7. The result is the following algorithm which is suitable for computation.

$$(4.10)\quad U_{i,n+1} = (\frac{7}{6} - 3\lambda)U_{i,n} + [(3\lambda - \frac{1}{6})(U_{i-1,n} + U_{i+1,n})$$

$$- (\lambda - \frac{1}{6})(U_{i-1,n-1} - 2U_{i,n-1} + U_{i+1,n-1})]/2$$

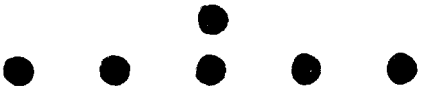
$$- k[\frac{1}{3} f(x_i, t_n) + \frac{1}{12}(f(x_{i-1}, t_n) + f(x_{i+1}, t_n)) + \frac{1}{2} f(x_i, t_{n+1})]$$

Method 8

Method 8 appears as Formula (3.40) in Saul'yev [6]. It is an explicit method. Both this method and Method 9 use five grid points at time $t = nk$ to calculate each value of $U_{i,n+1}$, $i = 2, 3, \ldots, I-2$. The form of the algorithm is  . This means that some other means must be used to calculate $U_{1,n+1}$ and $U_{I-1,n+1}$ near the ends of the x axis. During the testing, Method 2

(Classical Implicit) was used in an explicit manner to accomplish this. It has an error of order $O(h^2+k)$. Method 8 has an error of order asymptotic to $O(h^2+k)$. The method is stable for $\frac{k}{h^2} \leq 2.0$. The method was tested using values of $\lambda = 2.0,\ 1.0$ and 0.5. The greatest efficiency was achieved at $\lambda = 0.5$. In comparison with other methods this one did rather poorly. The necessity to use some special method to calculate $U_{1,\,n+1}$ and $U_{I-1,\,n+1}$ is also inconvenient.

Algorithm:



(4.11)

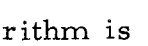
$$U_{i,\,n+1} = (1-2\lambda+\frac{3}{4}\lambda^2)U_{i,\,n} + \lambda(1-\frac{\lambda}{2})(U_{i-1,\,n}+U_{i+1,\,n})$$

$$+ \frac{1}{8}\lambda^2(U_{i-2,\,n}+U_{i+2,\,n}) - kf(x_i, t_n) \quad \text{for} \quad 2 \leq i \leq I-2$$

$$U_{1,\,n+1} = \frac{U_{1,\,n}+\lambda(U_{0,\,n+1}+U_{2,\,n+1})}{1+2\lambda} - \frac{kf(x_1, t_{n+1})}{1+2\lambda}$$

$$U_{I-1,\,n+1} = \frac{U_{I-1,\,n}+\lambda(U_{I-2,\,n+1}+U_{I,\,n+1})}{1+2\lambda} - \frac{kf(x_{I-1}, t_{n+1})}{1+2\lambda}$$

Method 9

Method 9 is presented as Formula (3.42) in Saul'yev [6]. It is an explicit method which uses five grid points at time $t = nk$ to calculate each value of $U_{i,\,n+1}$; $i = 2, 3, \ldots, I-2$. The form of the algorithm is  . In this respect it is like Method 8. Because

of this, some special means must be used to calculate $U_{1, n+1}$ and $U_{I-1, n+1}$ near the ends of the x axis.

The algorithm presented on page 42 of Saul'yev [6] is erroneous. This fact was discovered from the incorrect results of computer runs using that algorithm. The algorithm is supposed to be equivalent to the following:

$$\frac{\Delta_t U_{i, n}}{k} = \frac{\delta^2 x U_{i, n}}{h^2} - \frac{\delta^4 x U_{i, n}}{12h^2}$$

(in Saul'yev's notation)

where

$$\Delta_t U_{i, n} = U_{i, n+1} - U_{i, n}$$

and

$$\delta x^2 U_{i, n} = \frac{U_{i+1, n} - 2U_{i, n} + U_{i-1, n}}{h^2}$$

$$\delta^4 x U_{i, n} = \frac{U_{i+2, n} - 4U_{i+1, n} + 6U_{i, n} - 4U_{i-1, n} + U_{i-2, n}}{h^4}$$

This gives the following correct algorithm:

(4.12)
$$U_{i, n+1} = (1 - 2.5\lambda)U_{i, n} + \frac{4}{3}\lambda(U_{i-1, n} + U_{i+1, n}) - \frac{\lambda}{12}(U_{i-2, n} + U_{i+2, n}) - kf(x_i, t_n)$$

where

$$\lambda = \frac{k}{h^2}$$

Saul'yev gives the following algorithm which is incorrect:

$$U_{i,\,n+1} = (1-3\frac{k}{h^2})U_{i,\,n} + \frac{2}{3}\frac{k}{h^2}(U_{i-1,\,n}+U_{i+1,\,n})$$

$$+ \frac{1}{12}\frac{k}{h^2}(U_{i-2,\,n}+U_{i+2,\,n})$$

The method is stable for $\lambda = \frac{k}{h^2} \leq \frac{3}{8}$. The error has order $O(h^4+k)$. Method 2 (Classical Implicit) was used to calculate $U_{1,\,n+1}$ and $U_{I-1,\,n+1}$. It is used in an explicit way. The error has order $O(h^2+k)$.

Algorithm for Extreme Nodes:

$$U_{1,\,n+1} = \frac{U_{1,\,n}+\lambda(U_{0,\,n+1}+U_{2,\,n+1})}{1+2\lambda} - \frac{kf(x_1,\,t_{n+1})}{1+2\lambda}$$

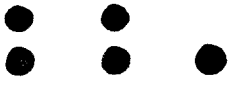
$$U_{I-1,\,n+1} = \frac{U_{I-1,\,n}+\lambda(U_{I-2,\,n+1}+U_{I,\,n+1})}{1+2\lambda} - \frac{kf(x_{I-1},\,t_{n+1})}{1+2\lambda}$$

Testing did verify the order of the error as $O(h^4+k)$; however for the values of h and k tried, the method did not perform particularly well in comparison with some other methods. Also, the problem of computing $U_{1,\,n+1}$ and $U_{I-1,\,n+1}$ is a complicating factor.

Methods 10 and 11. The Left and Right Asymmetric Methods

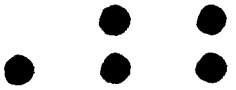
Both these methods are presented in Saul'yev [6]. They are not recommended for use by themselves, but they are used as a basis for

more complex methods possessing greater accuracy. Both methods are explicit. Method 10 has the stencil  . Method 11 has the stencil  . Method 10 starts at the leftmost grid point and proceeds to the right in its calculations. Method 11 starts at the rightmost grid point and proceeds to the left in its calculations. Method 10 was tested and it proved to be very poor to use by itself. According to Saul'yev, the total error has order $O(h)$, assuming $\lambda = \frac{k}{h^2}$ is kept constant. Methods 10 and 11 make use of a parameter α, $0 \leq \alpha \leq 1$, which is used in calculations. The method is stable for $\omega = \frac{h^2}{k} \geq \max\left(\frac{3}{2} - \alpha, 1 - \alpha + \sqrt{1-\alpha}\right)$. Saul'yev [6] shows in Formulas (3.9) and (3.10) that if $\alpha = 1$, then the method becomes unconditionally stable. This is true, but is of little value, because increasing the ratio $\frac{k}{h^2}$ increases the error. Furthermore, for a given ratio of $\frac{k}{h^2}$, it turns out that the largest errors occur for $\alpha = 1$. Furthermore Methods 10 and 11 are inefficient.

Algorithm: 

Method 10 (Left Asymmetric)

$$(4.13)\quad U_{i,n+1} = \frac{1}{\omega+\alpha}\left[\alpha U_{i-1,n+1} + (1-\alpha)U_{i-1,n} + U_{i+1,n} - (2-\omega-\alpha)U_{i,n}\right]$$

Algorithm: 

Method 11 (Right Asymmetric)

$$(4.14)\quad U_{i,n+1} = \frac{1}{\omega+\alpha}\left[\alpha U_{i+1,n+1} + (1-\alpha)U_{i+1,n} + U_{i-1,n} - (2-\omega-\alpha)U_{i,n}\right]$$

where

$$\omega = \frac{h^2}{k}, \quad \text{and} \quad 0 \leq \alpha \leq 1$$

in both (4. 13) and (4. 14).

Method 12. Method of Mean Arithmetic

This method consists in averaging the results of Methods 10 and 11 (Left and Right Asymmetric Methods) at each time step. Method 12 is therefore an explicit method. Method 12 is described in Reference [6]. The order of the error is $O(ch+h^2)$ where c is larger near the end nodes (near $x = 0$ and $x = 1$), and is smaller or zero at the mid-point $x = .5$. Because of this, the error reduces faster at the center than at the endpoints, and the reduction of the error is more unpredictable than it is for most of the other methods. Method 12 uses the parameter α, $0 \leq \alpha \leq 1$, as it is a combination of Methods 10 and 11. The stability requirement depends on α. The method is stable for $\lambda = \frac{k}{h^2} \leq \frac{1}{2(1-\alpha)}$. Therefore, as α approaches 1, the method is stable for larger and larger values of λ. Unfortunately, the error also increases with α. Testing was done for $\alpha = \frac{3}{4}$ and for $\alpha = \frac{1}{2}$ on problems one and two only. Much better efficiency was attained with $\alpha = \frac{1}{2}$ than with $\alpha = \frac{3}{4}$. This method is only fair in comparison with other methods and there is no good reason for using it.

Algorithm: Given the values $U(x_i, t_n)$, $i = 0, 1, 2, \ldots, I$, calculate $U(x_i, t_{n+1})$, $i = 1, 2, \ldots, I-1$ using Method 10 first and then Method 11. Add the results and divide by 2.

Multi-Step Methods in General

Method 7 was the only multi-step method tested; therefore no conclusive results about them can be drawn. Most multi-step methods listed in texts have an accuracy of order $O(h^4+k^2)$. Since single step methods exist with this same order of accuracy, there seems to be no reason to choose such a multi-step method. The multi-step methods are generally more complicated, and they require a single step starter method as well. Two multi-step methods, given as Formulas (8.35) and (8.36) in Saul'yev [6] are supposed to have an accuracy of order $O(h^6+k^3)$ for all settings of λ which give stability (the stability requirements are not given). In fact, for $\lambda = \frac{1}{\sqrt{840}}$, the theoretical accuracy is $O(h^8+k^4)$. It is not known whether such accuracy can be achieved in practice since neither method was tested. Perhaps it is possible if double precision arithmetic operations are used. In that case, it will still be necessary to use a single step starter method of the same order of accuracy. If such a starter method is available, then why not use it exclusively? It might be possible to find some other means to start the calculations without a single step starter method and without reducing the order of accuracy.

If so, then such multi-step methods might be very useful. Formulas (8. 35) and (8. 36) are designed for use on problems of type $u_{xx} = u_t$. They would be difficult to extend to problems of type $u_{xx} = u_t + f(x, t)$ as they are quite complicated. Finally, such high orders of accuracy are only possible if the function $u(x, t)$ has partial derivatives of high order. In practice, this may not be the case.

V. CONCLUSIONS AND SUMMARY

Which Method to Use?

The testing showed that the order of accuracy of the various methods tested agreed with their theoretical order of accuracy in most cases. This was true of methods of order $O(h^4+k^2)$ and for methods of lower order. It was possible to extend three methods of order $O(h^4+k^2)$ for problems of type $u_{xx} = u_t$ to problems of type $u_{xx} = u_t + f(x, t)$ and to retain an accuracy of order $O(h^4+k^2)$. Method 5** was supposed to be of order $O(h^6+k^3)$, but in practice the order was $O(h^4+k^2)$ using single precision arithmetic (relative error $\leq 2^{-38}$). There are several methods of order $O(h^4+k^2)$ and many methods of lower orders of accuracy which were not tested. Nevertheless, from the standpoint of efficiency, it is clear that the higher order methods are better and should be used whenever possible. However, if $u(x, t)$ does not possess partial derivatives of sufficiently high order, it will not be possible for higher order methods to achieve a higher order of accuracy. On the other hand it is not necessary to resort to multi-step methods to achieve an order of accuracy of $O(h^4+k^2)$. Several single step methods give this same order of accuracy and are to be preferred. Of the methods tested, from the standpoint of efficiency and simplicity, the two methods which seem best are Method 1, with $\lambda = \frac{1}{6}$ (Classical Explicit) and Method

6 with λ set as close to $\frac{1}{2\sqrt{5}}$ as is convenient. Both methods have accuracy of order $O(h^4+k^2)$. It was possible to extend their algorithms for use on problems of type $u_{xx} = u_t + f(x, t)$ without reducing the order of accuracy. The extended Method 1 is referred to as Method 1*. On problems one and two of type $u_{xx} = u_t$, Method 1, with $\lambda = \frac{1}{6}$, was more efficient. Since it is considerably simpler and it is explicit rather than implicit, like Method 7, it is to be preferred. If it is inconvenient to set $\lambda = \frac{1}{6}$, then Method 7 could be chosen. On problem three of type $u_{xx} = u_t + f(x, t)$, the two methods were almost equally efficient. Because of its simplicity and explicit nature, Method 1*, with $\lambda = \frac{1}{6}$ is slightly preferable.

It should be emphasized here that the problems being investigated are a special type of parabolic partial differential equation with constant coefficients. When more general problems are to be solved, the methods recommended here may no longer be worthwhile. In fact, it will usually be impossible to extend Method 1, with $\lambda = \frac{1}{6}$, (Classical Explicit) for use on more general problems while maintaining an accuracy of order $O(h^4+k^2)$ Method 6 can be extended in many cases to more general problems, see [1]. The Crank-Nicholson Method, which has order $O(h^2+k^2)$, does not do well on the problems investigated if it is compared to the methods of higher order. Nevertheless, it is unconditionally stable, and can be extended for use on much more general equations including nonlinear equations without

reducing the order of its accuracy.

It should be noted that higher order accurate methods of order $O(h^4+k^2)$ depend on the fact that the derivatives u_{ttt} and u_{xxxxxx} exist and the the derivatives u_{tt} and u_{xxxx} are continuous (so that $u_{tt} = u_{xxxx} = u_{txx} = u_{xxt}$). If this is not the case then the order will be reduced to $O(h^2+k)$, and the errors will be larger than for some of the simpler methods of a lower order of accuracy. In this case, the method which seems best from the standpoint of efficiency, stability, and simplicity is the widely used Crank-Nicholson Method (Method 3).

The Importance of Stability and the Choice of Step Size

In some cases it is best from the standpoint of efficiency, to set $\lambda = \frac{k}{h^2}$ to some special value, such as $\frac{1}{6}$ for the Classical Explicit Method. In other cases there is no known optimum setting. One thing that became apparent during the testing was that in all cases, the best settings of $\frac{k}{h^2}$ (or $\frac{k}{h}$) were less than the maximum values of λ necessary for stability. This indicates that stability requirements are important only when considered together with the order of accuracy of a particular method. For example, Method 6 is unconditionally stable, but since the order is $O(h^4+k^2)$, with maximum efficiency for $\frac{k}{h^2} = \frac{1}{2\sqrt{5}}$, k must be reduced much faster than h in order to achieve good accuracy without excessive use of computer

time. In general, even when no best value of $\frac{k}{h^2}$ or $\frac{k}{h}$ is known, it still is best to choose some value and to keep it constant as h and k are reduced. This guarantees that errors in terms of h and errors in terms of k will both be reduced at the same rate. If this is not done, one or the other of the errors will be reduced much faster than the other until it is relatively insignificant. Further reduction of it will increase computer time, but will not reduce the total error much. In fact, the testing showed that for most methods, including the two methods recommended in the previous section, the error reduces predictably, at least until it becomes very small.

BIBLIOGRAPHY

1. Douglas, Jim Jr. A survey of numerical methods for parabolic differential equations. In: Advances in Computers, ed. by Franz L. AIT, A. D. Booth, and R. E. Meagher. Vol. 2. New York, Academic, 1961. p. 1-54.

2. Forsythe, George E. and Wolfgang R. Wasow. Finite-difference methods for partial differential equations. New York, John Wiley, 1960. 144 p.

3. John, F. Advanced numerical analysis. New York, New York University, 1956. 194 p. (N. Y. U. Lecture Notes. Vol. 51)

4. Koshlyakov, N. S., M. M. Smirnov and E. B. Gliner. Differential equations of mathematical physics, tr. by Scripta Technica. Amsterdam, North-Holland, 1964. 701 p.

5. Richtmyer, Robert D and K. W. Morton. Difference methods for initial-value problems. 2d ed. New York, Interscience, 1967. 405 p.

6. Saul'yev, V. K. Integration of equations of parabolic type by the method of nets, tr. by G. J. Tee. New York, MacMillan, 1964. 346 p.

7. Smirnov, M. M. Second-order partial differential equations, tr. by Scripta Technica. New York, Stechert-Hafner, 1966. 192 p.

8. Smith, G. D. Numerical solution of partial differential equations. New York, Oxford University, 1965. 179 p.

9. Tychonov, A. N. and A. A. Samavski. Partial differential equations of mathematical physics, tr. by S. Radding. Vol. 1. San Francisco, Holden-Day, 1964. 380 p.

10. Varga, Richard S. Matrix iterative analysis. Englewood Cliffs, New Jersey, Prentice Hall, 1962. 322 p.

APPENDICES

APPENDIX A

Charts for Comparison of Methods

The charts compare the efficiency of the various methods. Efficiency refers to the amount of computer time required to achieve a certain accuracy. The methods are listed by number. The most efficient methods in each category are listed at the top of the charts and the less efficient methods are listed below them. In general, the methods of higher order will be more efficient. An attempt was made to compare methods of different orders of accuracy on the charts. This was done by listing two methods of different orders directly across from each other in their corresponding columns if they had about the same efficiency for the step sizes used in the testing. Of course, in this case, it is to be expected that with further reduction of the step sizes, the method of higher order will become more efficient. The step sizes and the accuracy attained are listed in Appendix B. As an example, in Chart IV for problem two, Method 3 with $\frac{k}{h} = 1.0$ is listed directly across from Method 1 with $\lambda = .125$. This indicates that they are of about the same efficiency for the step sizes used during the testing. Methods 10 and 11 (Left and Right Asymmetric) do not appear on the comparison charts. Their order is $O(h)$ for λ constant, and they are the poorest methods tested.

Chart I. Orders of the Error Terms and Stability Requirments

Method No.	Stability requirement	Error order on problems of type $u_{xx} = u_t$	Error order on problems of type $u_{xx} = u_t + f(x, t)$
(1)	$\lambda \leq \frac{1}{2}$	$O(h^4+k^2)$ for $\lambda = \frac{1}{6}$ $O(h^2+k)$ for $\lambda \neq \frac{1}{6}$	$O(h^2+k)$
(1*)	$\lambda \leq \frac{1}{2}$		$O(h^4+k^2)$ for $\lambda = \frac{1}{6}$ $O(h^2+k)$ for $\lambda \neq \frac{1}{6}$
(2)	stable	$O(h^2+k)$	$O(h^2+k)$
(3)	stable	$O(h^2+k^2)$	$O(h^2+k^2)$
(4)	stable	$O(h^2+k^2)$	$O(h^2+k^2)$
(5) $a \neq 1-\frac{\omega}{6}$	$\lambda \leq \frac{1}{2(1-a)}$	$O(h^2+k)$	
(5*) $a = 1-\frac{\omega}{6}$ $\omega \neq 2\sqrt{5}$	stable	$O(h^4+k^2)$	
(5**) $a = 1-\frac{\omega}{6}$ $\omega = 2\sqrt{5}$	stable	$O(h^6+k^3)$*	
(6)	stable	$O(h^4+k^2)$	$O(h^4+k^2)$
(7)	$\lambda \leq \frac{1}{3}$	$O(h^4+k^2)$	$O(h^4+k^2)$
(8)	$\lambda \leq 2.0$	$O(h^2+k)$**	$O(h^2+k)$**
(9)	$\lambda \leq \frac{3}{8}$	$O(h^4+k)$	
(12)	$\lambda \leq \frac{1}{2(1-a)}$	$O(ch+h^2)$	

* In practice the error was of order $O(h^4+k^2)$, but the efficiency was best for $a = 1 - \frac{\omega}{6}$, $\omega = 2\sqrt{5}$.

** The error order is asymptotic to $O(h^2+k)$.

Chart II. Time Factors

For a given order of the error term, the computer time required must be approximately multiplied by the time factor in order to divide the total error by R.

Error order	Time factor
$O(h^2+k)$	$R^{3/2}$
$O(h^4+k)$	$R^{5/4}$
$O(h^2+k^2)$	R
$O(h^4+k^2)$	$R^{3/4}$
$O(h^6+k^3)$	$R^{1/2}$

Chart III. Comparison of Methods--Problem One ($u_{xx} = u_t$)

$O(h^4+k^2)$	$O(h^2+k^2)$	$O(h^2+k)$	$O(ch+h^2+k)$
(1) $\lambda = \frac{1}{6}$			
(7) $\lambda = \frac{1}{6}$			
(5**)			
(7) $\lambda = \frac{1}{8}$			
(7) $\lambda = .25$			
(6) $\lambda = .5$			
(5*) $\lambda = 1.0$			
(6) $\lambda = 1.0$			
(5*) $\lambda = 2.0$	(3) $\frac{k}{h} = .5$		
	(4) $\frac{k}{h} = .1$		
	(3) $\frac{k}{h} = 1.0$	(5) $\alpha = \frac{3}{4}$, $\lambda = .5$	
	(3) $\frac{k}{h} = .1$	(5) $\alpha = \frac{3}{4}$, $\lambda = 1.0$	(12) $\alpha = \frac{1}{2}$, $\lambda = .5$
	(4) $\frac{k}{h} = .5$		(12) $\alpha = \frac{1}{2}$, $\lambda = 1.0$
		(1) $\lambda = .25$	
		(5) $\alpha = \frac{1}{2}$, $\lambda = .25$	
		(8)* $\lambda = .5$	
		(5) $\alpha = \frac{3}{4}$, $\lambda = 2.0$	
		(1) $\lambda = .5$	
		(8)* $\lambda = 1.0$	
		(5) $\alpha = \frac{1}{2}$, $\lambda = 1.0$	(12) $\alpha = \frac{3}{4}$, $\lambda = .5$
		(9)**	
		(2) $\lambda = 1.0$	(12) $\alpha = \frac{3}{4}$, $\lambda = 1.0$
		(2) $\lambda = 5.0$	

* Error order is asymptotic to $O(h^2+k)$

** Error order is $O(h^4+k)$ where * and ** refer to asterisks outside parentheses only.

Chart IV. Comparison of Methods--Problem Two ($u_{xx} = u_t$)

$O(h^4+k^2)$	$O(h^2+k^2)$	$O(h^2+k)$	$O(ch+h^2+k)$
(1) $\lambda = \frac{1}{6}$			
(5**)			
(7) $\lambda = \frac{1}{8}$			
(7) $\lambda = \frac{1}{6}$			
(7) $\lambda = .25$			
(6) $\lambda = .5$			
(5*) $\lambda = 1.0$			
(6) $\lambda = 1.0$			
(5*) $\lambda = 2.0$			
	(4) $\frac{k}{h} = .1$		
	(3) $\frac{k}{h} = .1$	(5) $a = \frac{3}{4}$, $\lambda = .5$	(12) $a = \frac{1}{2}$, $\lambda = .5$
	(3) $\frac{k}{h} = .5$	(5) $a = \frac{3}{4}$, $\lambda = 1.0$	(12) $a = \frac{1}{2}$, $\lambda = 1.0$
	(3) $\frac{k}{h} = 1.0$	(1) $\lambda = .125$	
	(4) $\frac{k}{h} = .5$	(5) $a = \frac{1}{2}$, $\lambda = .5$	
		(8)* $\lambda = .5$	
		(5) $a = \frac{3}{4}$, $\lambda = 2.0$	
		(1) $\lambda = .5$	
		(5) $a = \frac{1}{2}$, $\lambda = 1.0$	
		(9)**	
		(8)* $\lambda = 1.0$	(12) $a = \frac{3}{4}$, $\lambda = .5$
		(2) $\lambda = 1.0$	
		(2) $\lambda = 5.0$	(12) $a = \frac{3}{4}$, $\lambda = 1.0$

* Error order is asymptotic to $O(h^2+k)$

** Error order is $O(h^4+k)$ where * and ** refer to asterisks outside parentheses only.

Chart V. Comparison of Methods--Problem Three ($u_{xx} = u_t + f(x, t)$

$O(h^4+k^2)$	$O(h^2+k^2)$	$O(h^2+k)$
(6) $\lambda = .25$		
(1*) $\lambda = \frac{1}{6}$		
	(3) $\frac{k}{h} = .1$	
	(4) $\frac{k}{h} = .1$	
	(3) $\frac{k}{h} = .5$	
	(4) $\frac{k}{h} = .5$	
(7) $\lambda = .25$	(3) $\frac{k}{h} = 1.0$	
		(1) $\lambda = .25$
		(1) $\lambda = \frac{1}{6}$
		(1) $\lambda = .5$
		(8)* $\lambda = .5$
		(8)* $\lambda = 1.0$
		(2) $\lambda = 1.0$
		(2) $\lambda = 5.0$

* Error order is asymptotic to $O(h^2+k)$.

Where * refers to the asterisks outside the parentheses only.

Not all the methods were tested on problem three.

APPENDIX B

The Test Results

The charts that follow are arranged first by problem number and next by method. The unit square was the domain used for the problems. I and N are the number of distance and time steps resepctively. Time is given in 1/60 second units. The maximum errors are given in the form d.ddE^{-e} which represents d.dd x 10^{-e}. For example, 1.95E-3 means 1.95 x 10^{-3}. The underlined numbers give the x coordinate at which the maximum error, in magnitude, occurred. The numbers .1, .2, .5, and 1.0 refer to the time. As an example consider

I	N	Times	.1	.2	.5	1.0
10	20	12	1.80E-3(.1 & .9)	9.22E-4(.3 & .7)	1.25E-4(.2 & .8)	631E-6(.2 &.8)

There were 10 distance steps, (h = .1), and 20 time steps, (k = .05). The run took 12/60 seconds (excluding the time to print and punch data). At time t = .5 the maximum error occurred at x = .2 and x = .8. This error was of magnitude 1.25 x 10^{-4}. The time of 12/60 was determined by subtracting $\frac{60}{60}$ = 1.0 seconds from the printed time of 72/60 seconds. This is a good approximation to the time required to print the data and to punch the cards.

Problem One. Test Results

Method 1--Classical Explicit

λ	I	N	Time	.1	.2	.5	1.0
1/6	10	600	195	2.22E-6(.5)	1.79E-7(.5)	9.07E-8(.5)	1.89E-9(.5)
1/6	20	2400	1229	1.38E-7(.5)	1.11E-8(.5)	5.70E-9(.5)	1.89E-10(.5)
1/6	30	5400	4683	2.71E-8(.5)	2.15E-9(.5)	1.21E-9(.5)	1.46E-10(.5)
.125	10	800	238	1.93E-4(.5)	1.46E-4(.5)	1.91E-5(.5)	2.76E-7(.5)
.25	10	400	121	3.96E-4(.5)	2.93E-4(.5)	3.76E-5(.5)	5.35E-7(.5)
.25	20	1600	866	9.78E-5(.5)	7.29E-5(.5)	9.41E-6(.5)	1.35E-7(.5)
.5	10	200	54	1.57E-3(.48.6)	1.16E-3(.5)	1.47E-4(.5)	2.04E-6(.5)
.5	20	800	408	3.96E-4(.5)	2.93E-4(.5)	3.76E-5(.5)	5.35E-7(.5)

Method 2--Classical Implicit

1.0	10	100	65	5.22E-3(.5)	4.01E-3(.5)	5.65E-4(.5)	9.35E-6(.5)
1.0	20	400	494	1.35E-3(.5)	1.01E-3(.5)	1.34E-4(.5)	2.00E-6(.5)
1.0	30	900	1640	6.03E-4(.5)	4.52E-4(.5)	5.90E-5(.5)	8.63E-7(.5)
5.0	10	20	23	1.98E-2(.5)	1.66E-2(.5)	2.95E-3(.5)	7.60E-5(.5)
5.0	20	80	98	5.73E-3(.5)	4.42E-3(.5)	6.26E-4(.5)	1.05E-5(.5)
5.0	40	320	778	1.49E-3(.5)	1.12E-3(.5)	1.49E-4(.5)	2.22E-6(.5)

Method 3--Crank-Nicholson

.1	10	100	72	7.01E-4(.5)	5.26E-4(.5)	6.90E-5(.5)	1.01E-6(.5)
.1	20	200	285	1.76E-4(.5)	1.31E-4(.5)	1.70E-5(.5)	2.46E-7(.5)
.1	40	400	1168	4.40E-5(.5)	3.28E-5(.5)	4.25E-6(.5)	6.13E-8(.5)
.5	10	20	12	1.80E-3(.18.9)	9.22E-4(.38.7)	1.25E-4(.28.8)	6.31E-6(.28.5)
.5	40	80	237	7.01E-5(.5)	5.34E-5(.5)	6.90E-6(.5)	9.79E-8(.5)
.5	160	320	3677	4.40E-6(.5)	3.33E-6(.5)	4.31E-7(.5)	5.41E-9(.5)
.5	240	480	8301	1.96E-6(.5)	1.49E-6(.6)	1.99E-7(.5)	1.03E-8(.6)
1.0	40	40	114	4.26E-4(.8)	3.31E-4(.5)	4.14E-5(.5)	1.49E-6(.5)
1.0	80	80	456	1.07E-4(.5)	8.08E-5(.5)	1.04E-5(.5)	1.49E-7(.5)

Method 4--Alternate Explicit-Implicit

$\frac{k}{h}$	I	N	Time	.1	.2	.5	1.0
.1	10	100	41	4.79E-4(.5)	3.55E-4(.5)	4.64E-4(.5)	6.76E-7(.5)
.1	20	200	176	1.19E-4(.5)	8.82E-5(.5)	1.14E-5(.5)	1.65E-7(.5)
.1	40	400	690	2.97E-5(.5)	2.20E-5(.5)	2.85E-6(.5)	4.04E-8(.5)
.5	10	20	8	9.43E-2(.28.8)	8.09E-3(.5)	1.88E-3(.5)	4.86E-4(.18.9)
.5	40	80	140	4.26E-4(.28.8)	3.31E-4(.5)	4.14E-5(.5)	1.49E-6(.18.9)
.5	160	320	2177	2.68E-5(.5)	2.02E-5(.5)	2.61E-6(.5)	3.68E-8(.5)
.5	240	480	4880	1.19E-5(.5)	8.96E-6(.5)	1.16E-6(.5)	1.80E-8(.5)

Method 5--Formula 8.16 [4]; $a = \frac{1}{2}$

λ	I	N	Time	.1	.2	.5	1.0
1.0	10	100	72	1.66E-3(.4&.6)	1.25E-3(.4&.6)	1.90E-4(.4&.6)	1.46E-5(.4&.6)
1.0	20	400	582	4.00E-4(.5)	3.00E-4(.5)	4.07E-5(.5)	1.37E-6(.5)
.5	10	200	143	4.10E-4(.5)	3.04E-4(.5)	3.90E-5(.5)	5.54E-7(.5)
.5	20	800	1166	9.87E-5(.5)	7.35E-5(.5)	9.50E-6(.5)	1.36E-7(.5)
.25	10	400	292	1.90E-4(.5)	1.43E-4(.5)	1.87E-5(.5)	2.71E-7(.5)

Method 5--Formula 8.16[4]; $a = \frac{3}{4}$

2.0	10	50	27	1.87E-3(.5)	1.42E-3(.5)	2.69E-4(.5)	8.42E-5(.4&.6)
2.0	20	200	286	4.10E-4(.5)	3.06E-4(.5)	4.47E-5(.5)	5.79E-6(.5)
2.0	40	800	2307	9.87E-5(.5)	7.37E-5(.5)	9.85E-6(.5)	4.61E-7(.5)
1.0	10	100	71	4.68E-4(.5)	3.47E-4(.5)	4.45E-5(.5)	6.32E-7(.5)
1.0	20	400	574	1.02E-4(.5)	7.62E-5(.5)	9.84E-6(.5)	1.41E-7(.5)
.5	10	200	138	1.76E-4(.5)	1.33E-4(.5)	1.73E-5(.5)	2.51E-7(.5)

Method 5*--Formula 8.16 [6]; $a = 1 - \frac{\omega}{6}$, $\omega = \frac{1}{\lambda}$

1.0	10	100	80	7.65E-5(.5)	5.60E-5(.5)	7.13E-6(.5)	1.02E-7(.5)
1.0	20	400	609	4.77E-6(.5)	3.50E-6(.5)	4.46E-7(.5)	6.11E-9(.5)
2.0	10	50	36	3.02E-4(.4&.6)	2.29E-4(.5)	2.94E-5(.5)	4.19E-7(.5)
2.0	20	200	304	1.91E-5(.5)	1.43E-5(.5)	1.84E-6(.5)	2.67E-8(.5)
2.0	40	800	2450	1.19E-6(.5)	8.93E-7(.5)	1.16E-7(.5)	3.11E-9(.5)

Method 5**--Formula 8.16 [6]; $a = 1 - \frac{\omega}{6}$, $\omega = \frac{1}{\lambda}$, $\lambda = \frac{1}{2\sqrt{5}}$

λ	I	Time step size	Time	.100623	.201246	.500879	1.0
.2236	10	.002236	341	3.89E-6(.5)	1.44E-6(.5)	7.40E-8(.5)	3.57E-10(.4)
				.100064	.200128	.500320	
.2236	20	.000559	2728	2.44E-7(.5)	9.17E-8(.5)	5.63E-9(.5)	9.60E-10(.6)
				.100126	.200003	.500134	
.2236	30	.000248	8949	4.76E-8(.5)	1.74E-8(.5)	4.03E-10(.5)	4.82E-10(.5)

Actually $\lambda = \frac{1}{2\sqrt{5}} = .2236068$ to 7 decimal places.

Method 6--Formula 7.4 [1]

λ	I	N	Time	.1	.2	.5	1.0
1.0	10	100	81	7.65E-5(.5)	5.60E-5(.5)	7.13E-6(.5)	1.02E-7(.5)
1.0	20	400	706*	4.77E-6(.5)	3.50E-6(.5)	4.46E-7(.5)	6.11E-9(.5)
1.0	30	900	2430*	9.42E-7(.5)	6.91E-7(.5)	6.74E-8(.5)	6.11E-10(.5)
.5	10	200	190*	1.92E-5(.5)	1.29E-5(.5)	1.56E-6(.5)	2.18E-8(.5)
.5	20	800	1410*	1.20E-6(.5)	8.08E-7(.5)	9.70E-8(.5)	7.57E-10(.5)

* Not run on block time, time may be too slow.

Method 7--Formula 8.30 [6]

λ	I	N	Time	.1	.2	.5	1.0
1/6	10	600	354	2.22E-6(.5)	1.78E-7(.5)	9.10E-8(.5)	2.08E-9(.5)
1/6	20	2400	2775	1.37E-7(.5)	1.03E-8(.5)	6.61E-9(.5)	1.15E-9(.6)
1/6	30	5400	9631	2.55E-8(.5)	3.49E-10(.8)	3.30E-9(.5)	2.24E-9(.6)
1/8	10	800	510	3.66E-6(.5)	1.27E-6(.5)	5.18E-8(.5)	1.46E-10(.6)
1/8	20	3200	3889	2.28E-7(.5)	7.85E-8(.5)	1.99E-9(.5)	1.28E-9(.5)
.25	10	400	242	4.53E-6(.5)	5.00E-6(.5)	7.69E-7(.5)	1.18E-8(.5)
.25	20	1600	1932	2.91E-7(.5)	3.14E-7(.5)	4.83E-8(.5)	1.38E-9(.5)
.25	40	6400	15260	2.03E-8(.5)	2.20E-8(.5)	5.60E-9(.5)	2.75E-9(.6)
2/7	10	350	204	8.98E-6(.5)	8.45E-6(.5)	1.22E-6(.5)	1.83E-8(.5)
2/7	20	1400	1636	5.75E-7(.5)	5.29E-7(.5)	7.62E-8(.5)	1.69E-9(.5)
Method 8--Formula 3.40 [6]							
.5	10	200	93	6.56E-4(.5)	5.58E-4(.5)	7.68E-5(.5)	1.11E-6(.5)
.5	20	800	730	2.23E-4(.5)	1.73E-4(.5)	2.28E-5(.5)	3.28E-7(.5)
1.0	10	100	42	2.35E-3(.5)	1.88E-3(.5)	2.99E-4(.5)	4.21E-5(.5)
1.0	20	400	372	6.56E-4(.5)	4.97E-4(.5)	6.22E-5(.5)	1.64E-6(.5)
1.0	40	1600	3231	1.69E-4(.5)	1.27E-4(.5)	1.65E-5(.5)	2.60E-7(.5)
2.0	10	50	19	6.46E-3(.5)	4.63E-3(.48.6)	5.53E-4(.5)	6.83E-6(.5)
2.0	20	200	173	1.57E-3(.5)	1.16E-3(.5)	1.47E-4(.5)	2.04E-6(.5)
2.0	40	800	1623	3.96E-4(.5)	2.93E-4(.5)	3.76E-5(.5)	5.34E-7(.5)
Method 9--Formula 3.42 [6]							
.25	10	400	169	8.38E-4(.5)	6.93E-4(.5)	9.37E-5(.5)	1.34E-6(.5)
.25	20	1600	1310	2.71E-4(.5)	2.08E-4(.5)	2.73E-5(.5)	3.92E-7(.5)
1/3	10	300	134	1.15E-3(.5)	9.41E-4(.5)	1.26E-4(.5)	1.78E-6(.5)
1/12	20	4800	4331	8.65E-5(.5)	6.76E-5(.5)	8.97E-6(.5)	1.30E-7(.5)
Method 10--Left Asymmetric; $\alpha = \frac{1}{2}$							
.5	10	200	86	2.14E-3(.8)	9.71E-4(.7)	7.50E-5(.7)	8.29E-7(.5)
.5	20	800	644	1.02E-3(.8)	4.33E-4(.8)	2.81E-5(.7)	2.76E-7(.7)
Method 10--Left Asymmetric; $\alpha = \frac{3}{4}$							
1.0	10	100	45	6.01E-3(.2)	2.61E-3(.7)	2.09E-4(.5)	2.35E-6(.6)
1.0	20	400	318	2.96E-3(.2)	1.23E-3(.8)	8.06E-5(.7)	8.06E-7(.7)
2.0	20	200	160	5.82E-3(.8)	2.60E-3(.7)	2.08E-4(.7)	2.33E-6(.6)
2.0	40	800	1273	2.96E-3(.2)	1.23E-3(.8)	8.06E-5(.7)	8.06E-7(.7)
Method 10--Left Asymmetric; $\alpha = 1$							
1.0	10	100	42	9.05E-3(.2)	3.34E-3(.2)	2.13E-4(.7)	2.28E-6(.7)
1.0	30	900	1041	2.72E-3(.2)	1.06E-3(.2)	5.85E-5(.8)	5.02E-7(.7)

Method 12--Method of Mean Arithmetic; $a = \frac{1}{2}$

λ	I	N	Time	.1	.2	.5	1.0
1.0	10	100	80	1.78E-4(.5)	1.46E-4(.28.8)	3.84E-5(.5)	6.60E-6(.5)
1.0	20	400	693	9.91E-5(.5)	7.13E-5(.5)	7.62E-6(.5)	2.40E-7(.5)
1.0	30	900	2391	6.29E-5(.5)	4.66E-5(.5)	6.29E-6(.5)	1.59E-7(.5)
.5	10	200	173	6.17E-5(.18.9)	2.04E-5(.5)	2.34E-6(.5)	3.24E-8(.5)
.5	20	800	1384	2.96E-5(.5)	2.18E-5(.5)	2.80E-6(.5)	4.29E-8(.5)
.25	10	400	351	3.03E-4(.5)	2.27E-4(.5)	2.95E-4(.5)	4.28E-7(.5)

Method 12--Method of Mean Arithmetic; $a = \frac{3}{4}$

1.0	10	100	82	3.09E-3(.5)	2.29E-3(.5)	3.06E-4(.5)	4.74E-6(.5)
1.0	20	400	695	1.07E-3(.5)	8.07E-4(.5)	1.05E-4(.5)	1.29E-6(.5)
1.0	30	900	2399	4.78E-4(.5)	3.56E-4(.5)	4.64E-5(.5)	6.75E-7(.5)
.5	10	200	173	1.14E-3(.5)	8.42E-4(.5)	1.10E-4(.5)	1.63E-6(.5)
2.0	10	50	38	1.02E-2(.5)	8.20E-3(.5)	1.29E-3(.48.6)	9.96E-5(.5)
2.0	20	200	343	3.71E-3(.5)	2.79E-3(.5)	3.72E-4(.5)	1.52E-6(.5)
2.0	40	800	2788	1.08E-3(.5)	8.07E-4(.5)	1.05E-4(.5)	1.29E-6(.5)

Problem Two. Test Results

Method 1--Classical Explicit

λ	I	N	Time	.1	.2	.5	1.0
1/6	10	600	285	4.96E-6(.2)	3.73E-6(.2)	1.02E-6(.5)	8.70E-8(.6)
1/6	20	2400	1798	3.11E-7(.2)	2.37E-7(.2)	5.80E-8(.5)	5.80E-9(.6)
1/6	30	5400	5493	6.11E-8(.2)	4.60E-8(.2)	6.10E-8(.2)	1.40E-9(.5)
.125	10	800	360	3.70E-4(.5)	1.72E-4(.4)	6.97E-5(.6)	1.08E-5(.6)
.25	10	400	183	7.45E-4(.5)	3.42E-4(.4)	1.18E-4(.7)	2.13E-5(.6)
.25	20	1600	1106	1.84E-4(.5)	8.56E-5(.4)	2.93E-5(.7)	5.33E-6(.6)
.5	10	200	87	2.96E-3(.5)	1.37E-3(.4)	4.79E-4(.7)	8.49E-5(.6)
.5	20	800	539	7.45E-4(.5)	3.42E-4(.4)	1.18E-4(.7)	2.13E-5(.5)

Method 2--Classical Implicit

1.0	10	100	75	9.66E-3(.5)	4.52E-3(.4)	1.58E-3(.6)	3.09E-4(.6)
1.0	20	400	540	2.53E-3(.5)	1.18E-3(.4)	4.06E-4(.6)	7.53E-5(.6)
1.0	30	900	1756	1.14E-3(.5)	5.29E-4(.4)	1.81E-4(.6)	3.33E-5(.6)
5.0	10	20	13	3.46E-2(.5)	1.72E-2(.5)	6.19E-3(.6)	1.47E-3(.6)
5.0	20	80	108	1.06E-2(.5)	4.99E-3(.4)	1.74E-3(.6)	3.42E-4(.6)
5.0	40	320	813	2.79E-3(.5)	1.31E-3(.4)	4.48E-4(.6)	8.34E-5(.6)

Method 3--Crank-Nicholson

$\frac{k}{h}$	I	N	Time	.1	.2	.5	1.0
.1	10	100	83	1.25E-3(.6)	7.10E-4(.5)	2.06E-4(.7)	3.93E-5(.6)
.1	20	200	311	3.15E-4(.6)	1.80E-4(.5)	5.12E-5(.7)	9.75E-6(.6)
.1	40	400	1212	7.87E-5(.6)	4.51E-5(.5)	1.28E-5(.7)	2.43E-6(.6)
.5	10	20	15	9.46E-3(.2)	4.92E-3(.7)	9.49E-4(.4)	6.34E-5(.2)
.5	40	80	240	5.05E-4(.3)	2.54E-4(.7)	5.32E-5(.5)	3.32E-6(.6)
.5	160	320	3670	3.14E-5(.3)	1.58E-5(.7)	3.30E-6(.6)	2.04E-7(.6)
.5	240	480	8260	1.39E-5(.2)	7.00E-6(.7)	1.47E-6(.5)	9.56E-8(.6)
1.0	40	40	116	2.22E-3(.2)	9.45E-4(.7)	2.53E-4(.5)	2.17E-5(.6)
1.0	80	80	465	5.47E-4(.3)	2.37E-4(.7)	6.22E-5(.5)	5.32E-6(.6)

Method 4--Alternate Explicit-Implicit

.1	10	100	58	9.86E-4(.2)	1.05E-3(.6)	1.20E-4(.7)	2.79E-5(.6)
.1	20	200	198	2.52E-4(.2)	2.63E-4(.6)	2.91E-5(.7)	6.88E-6(.6)
.1	40	400	742	6.32E-5(.2)	6.58E-5(.6)	7.23E-6(.7)	1.72E-6(.6)
.5	10	20	7	3.80E-2(.2)	3.52E-2(.1)	1.16E-2(.1)	2.71E-3(.1)
.5	20	80	147	2.22E-3(.3)	9.45E-4(.7)	2.53E-4(.5)	2.17E-5(.6)
.5	160	320	2184	1.36E-4(.3)	5.89E-5(.7)	1.55E-5(.5)	1.32E-6(.6)
.5	240	480	4859	6.04E-5(.3)	2.62E-5(.7)	6.87E-6(.5)	5.88E-7(.6)

Method 5--Formula 8.16 [6]; $\alpha = \frac{1}{2}$

λ	I	N	Time	.1	.2	.5	1.0
1.0	10	100	83	3.15E-3(.6)	1.35E-3(.3)	5.46E-4(.7)	9.61E-5(.7)
1.0	20	400	648	7.57E-4(.5)	3.38E-4(.4)	1.16E-4(.7)	2.06E-5(.6)
.5	10	200	181	7.88E-4(.5)	3.41E-4(.4)	1.24E-4(.6)	2.20E-5(.6)
.5	20	800	1293	1.87E-4(.5)	8.55E-5(.4)	2.97E-5(.6)	5.38E-6(.6)
.25	10	400	359	3.60E-4(.5)	1.72E-4(.5)	5.82E-5(.7)	1.06E-5(.6)

Method 5--Formula 8.16 [6]; $\alpha = \frac{3}{4}$

2.0	10	50	47	3.91E-3(.4)	1.66E-3(.3)	7.63E-4(.6)	1.89E-4(.5)
2.0	20	200	322	7.95E-4(.5)	3.40E-4(.4)	1.19E-4(.6)	1.62E-5(.5)
2.0	40	800	2472	1.87E-4(.5)	8.54E-5(.4)	2.93E-5(.6)	5.01E-6(.6)
1.0	10	100	86	9.60E-4(.5)	3.72E-4(.3)	1.53E-4(.6)	2.49E-5(.6)
1.0	20	400	662	1.98E-4(.5)	8.51E-5(.4)	3.14E-5(.6)	5.56E-6(.6)
.5	10	200	183	3.17E-4(.5)	1.77E-4(.5)	5.27E-5(.7)	9.93E-6(.6)

Method 5*--Formula 8.16 [6]; $\alpha = 1 - \frac{\omega}{6}$, $\omega = \frac{1}{\lambda}$

1.0	10	100	88	3.43E-4(.3)	1.42E-4(.7)	3.94E-5(.5)	3.63E-6(.6)
1.0	20	400	668	2.14E-5(.3)	8.81E-6(.7)	2.45E-6(.5)	2.25E-7(.6)
2.0	10	50	44	1.45E-3(.3)	5.86E-4(.7)	1.66E-4(.5)	1.53E-5(.6)
2.0	20	200	331	8.83E-5(.3)	3.64E-5(.7)	1.02E-5(.5)	9.41E-7(.6)
2.0	40	800	2563	5.52E-6(.3)	2.11E-6(.7)	6.40E-7(.5)	6.01E-8(.6)

Method 5**--Formula 8.16 [6]; $\alpha = 1 - \frac{\omega}{6}$, $\omega = \frac{1}{\lambda}$, $\lambda = \frac{1}{2\sqrt{5}}$

λ	I	Time step size	Time	.100628	.201246	.500879	1.0
.2236	10	.002236	407	4.51E-6(.5)	1.51E-6(.5)	9.92E-8(.4)	1.34E-7(.9)
				.100064	.2000128	.500320	
.2236	20	.000559	2991	2.89E-7(.5)	9.81E-8(.5)	1.86E-8(.9)	9.69E-9(.9)
				.100126	.200003	.500134	
.2236	30	.000248	9730	7.54E-8(.9)	4.70E-8(.9)	1.79E-8(.9)	4.59E-9(.9)

Method 6--Formula 7.4 [1]

λ	I	N	Time	.1	.2	.5	1.0
1.0	10	100	96*	3.43E-4(.3)	1.42E-4(.7)	3.94E-5(.5)	3.63E-6(.6)
1.0	20	400	777*	2.14E-5(.3)	8.81E-6(.7)	2.45E-6(.5)	2.25E-7(.6)
1.0	30	900	2489*	4.24E-6(.3)	1.75E-6(.7)	4.86E-6(.5)	4.42E-8(.6)
.5	10	200	202*	7.48E-5(.3)	3.07E-5(.7)	8.20E-6(.5)	7.60E-7(.6)
.5	20	800	1491*	4.68E-6(.3)	1.91E-6(.7)	5.13E-7(.5)	4.72E-8(.6)

Method 7--Formula 8.30 [4]

λ	I	N	Time	.1	.2	.5	1.0
1/6	10	600	445	4.96E-6(.2)	3.74E-6(.2)	1.02E-6(.5)	8.67E-8(.6)
1/6	20	2400	3144	3.10E-7(.2)	2.38E-7(.2)	6.42E-8(.5)	6.00E-9(.7)
1/6	30	5400	10484	6.01E-8(.2)	4.85E-8(.2)	1.52E-8(.5)	2.80E-9(.6)
1/8	10	800	619	3.88E-6(.6)	1.57E-6(.3)	2.17E-7(.5)	1.37E-8(.5)
1/8	20	3200	4357	3.40E-7(.6)	1.00E-7(.3)	1.50E-8(.5)	2.00E-9(.9)
.25	10	400	323	3.66E-5(.3)	1.60E-5(.7)	4.79E-6(.7)	4.42E-7(.6)
.25	20	1600	2177	2.28E-6(.3)	9.62E-7(.7)	2.98E-7(.5)	2.78E-8(.6)
.25	40	6400	16251	1.41E-7(.3)	6.15E-8(.2)	2.12E-8(.5)	4.00E-9(.6)
Method 8--Formula 3.40 [6]							
.5	10	200	121	1.80E-3(.1)	5.33E-4(.1)	2.53E-4(.1)	4.11E-5(.5)
.5	20	800	861	4.27E-4(.5)	1.80E-4(.4)	7.03E-5(.6)	1.27E-5(.6)
1.0	10	100	64	4.40E-3(.5)	1.59E-3(.4)	7.84E-4(.6)	1.67E-4(.6)
1.0	20	400	429	1.24E-3(.5)	5.44E-4(.4)	2.05E-4(.6)	3.94E-5(.6)
1.0	40	1600	3472	3.19E-4(.5)	1.46E-4(.4)	5.12E-5(.7)	9.35E-6(.6)
2.0	10	50	27	1.23E-2(.6)	5.96E-3(.4)	2.03E-3(.6)	3.35E-4(.6)
2.0	20	200	204	2.96E-3(.6)	1.37E-3(.4)	4.79E-4(.7)	8.49E-5(.6)
2.0	40	800	1744	7.45E-4(.5)	3.42E-4(.4)	1.18E-4(.7)	2.13E-5(.6)
Method 9--Formula 3.42 [6]							
.25	10	400	230	1.61E-3(.5)	5.23E-4(.4)	2.78E-4(.6)	5.08E-5(.6)
.25	20	1600	1557	5.17E-4(.5)	2.23E-4(.4)	8.44E-5(.6)	1.53E-5(.6)
1/3	10	300	183	2.21E-3(.5)	9.39E-4(.4)	3.78E-4(.6)	6,87E-5(.6)
1/12	20	4800	5043	1.65E-4(.5)	6.86E-5(.4)	2.74E-5(.6)	4.99E-6(.5)
Method 10--Left Asymmetric; $\alpha = \frac{1}{2}$							
.5	10	200	117	5.42E-3(.3)	7.36E-3(.3)	3.50E-3(.5)	6.88E-4(.5)
.5	20	800	770	2.91E-3(.3)	3.76E-3(.3)	1.80E-3(.5)	3.52E-4(.5)
Method 10--Left Asymmetric; $\alpha = \frac{3}{4}$							
1.0	10	100	51	1.66E-2(.3)	2.23E-2(.3)	1.04E-2(.5)	2.07E-3(.5)
1.0	20	400	390	8.77E-3(.3)	1.13E-2(.3)	5.36E-3(.5)	1.06E-3(.5)
2.0	20	200	195	1.66E-2(.3)	2.22E-2(.3)	1.05E-2(.5)	2.08E-3(.5)
2.0	40	800	1404	8.80E-3(.3)	1.13E-2(.3)	5.36E-3(.5)	1.06E-3(.5)
Method 10--Left Asymmetric; $\alpha = 1$							
1/2	10	20	10	1.25E-1(.4)	1.41E-1(.6)	7.43E-2(.3)	1.28E-2(.4)
1/2	160	320	2137	1.25E-1(.4)	1.41E-1(.6)	7.47E-2(.3)	1.28E-2(.4)
1.0	10	100	54	2.37E-2(.3)	3.00E-2(.3)	1.41E-2(.5)	2.82E-3(.5)
1.0	20	400	380	1.20E-2(.3)	1.52E-2(.3)	7.17E-3(.5)	1.42E-3(.5)

Method 12--Method of Mean Arithmetic; $\alpha = \frac{1}{2}$

λ	I	N	Time	.1	.2	.5	1.0
1.0	10	100	98	5.29E-4(.1)	7.33E-4(.2)	1.19E-4(.1)	1.50E-5(.1)
1.0	20	400	758	1.82E-4(.6)	1.10E-4(.6)	3.77E-5(.7)	6.57E-6(.6)
1.0	30	900	2527	1.14E-4(.5)	5.08E-5(.5)	1.91E-5(.7)	3.53E-6(.6)
.5	10	200	204	1.19E-4(.1)	1.70E-4(.2)	3.25E-5(.1)	2.73E-6(.1)
.5	20	800	1509	5.36E-5(.6)	2.70E-5(.5)	9.71E-6(.7)	1.69E-6(.6)
.25	10	400	410	5.67E-4(.5)	2.60E-4(.4)	9.31E-5(.7)	1.69E-5(.6)

Method 12--Method of Mean Arithmetic; $\alpha = \frac{3}{4}$

1.0	10	100	100	5.48E-3(.5)	2.65E-3(.5)	9.79E-4(.7)	1.80E-4(.6)
1.0	20	400	761	1.85E-3(.5)	8.21E-4(.5)	3.09E-4(.7)	5.61E-5(.6)
1.0	30	900	2537	8.93E-4(.5)	3.98E-4(.4)	1.47E-4(.7)	2.65E-5(.6)
.5	10	200	202	2.06E-3(.5)	9.66E-4(.5)	3.51E-4(.7)	6.39E-5(.6)
2.0	10	50	54	1.83E-2(.6)	8.19E-3(.6)	3.66E-3(.7)	7.57E-4(.7)
2.0	20	200	383	6.84E-3(.5)	2.97E-3(.5)	1.18E-3(.7)	2.23E-4(.7)
2.0	40	800	2930	2.02E-3(.5)	9.00E-4(.4)	3.32E-4(.7)	6.08E-5(.6)

Problem Three. Test Results

Method 1--Classical Explicit

λ	I	N	Time	.1	.2	.5	1.0
1/6	10	600	1776	1.26E-5(.5)	1.48E-5(.5)	4.33E-6(.5)	2.47E-6(.5)
1/6	20	2400	13565	3.14E-6(.5)	3.69E-6(.5)	1.07E-6(.5)	6.12E-7(.5)
1/6	30	5400	---	1.39E-6(.5)	1.64E-6(.5)	4.76E-7(.5)	------
.125	10	800	2381	9.46E-6(.5)	1.11E-5(.5)	3.25E-6(.5)	1.85E-6(.5)
.25	10	400	1182	1.90E-5(.5)	2.23E-5(.5)	6.48E-6(.5)	3.71E-6(.5)
.5	10	200	589	3.83E-5(.5)	4.48E-5(.5)	1.29E-5(.5)	7.41E-6(.5)
.5	20	800	4487	9.44E-6(.5)	1.11E-5(.5)	3.21E-6(.5)	1.84E-6(.5)

Method 1*--Classical Explicit Extended

1/6	10	60	1883	9.30E-8(.5)	1.08E-7(.5)	2.30E-8(.5)	1.21E-8(.5)
1/6	20	2400	14444	5.76E-9(.5)	6.75E-9(.5)	1.42E-9(.5)	7.86E-10(.5)

Method 2--Classical Implicit

1.0	10	100	65	5.22E-3(.5)	4.01E-3(.5)	5.64E-4(.5)	9.35E-6(.5)
1.0	20	400	494	1.35E-3(.5)	1.01E-3(.5)	1.34E-4(.5)	2.00E-6(.5)
1.0	30	900	1640	6.03E-4(.5)	4.52E-4(.5)	5.90E-5(.5)	8.63E-7(.5)
5.0	10	20	23	1.98E-2(.5)	1.66E-2(.5)	2.95E-3(.5)	7.60E-5(.5)
5.0	20	80	98	5.73E-3(.5)	4.42E-3(.5)	6.26E-4(.5)	1.05E-5(.5)
5.0	40	320	778	1.49E-3(.5)	1.12E-3(.5)	1.49E-4(.5)	2.22E-6(.5)

Method 3--Crank-Nicholson

$\frac{k}{h}$	I	N	Time	.1	.2	.5	1.0
.1	10	100	338	1.67E-6(.5)	1.93E-6(.5)	4.16E-7(.5)	2.18E-7(.5)
.1	20	200	1297	4.18E-7(.5)	4.84E-7(.5)	1.04E-7(.5)	5.43E-8(.5)
.1	40	400	5108	1.05E-7(.5)	1.21E-7(.5)	2.59E-8(.5)	1.36E-8(.5)
.5	10	20	62	4.25E-5(.5)	4.89E-5(.5)	1.04E-5(.5)	5.62E-6(.5)
.5	40	80	1024	2.62E-6(.5)	3.03E-6(.5)	6.46E-7(.5)	3.40E-7(.5)
.5	160	320	16127	1.63E-7(.5)	1.89E-7(.5)	4.02E-8(.5)	2.13E-8(.5)
1.0	40	40	506	1.05E-5(.5)	1.21E-5(.5)	2.59E-6(.5)	1.37E-6(.5)
1.0	80	80	2007	2.62E-6(.5)	3.03E-6(.5)	6.46E-7(.5)	3.40E-7(.5)

Method 4--Alternate Explicit-Implicit

.1	10	100	305	8.30E-7(.5)	1.46E-6(.5)	1.76E-6(.5)	1.16E-6(.5)
.1	20	200	1179	2.07E-7(.5)	3.62E-7(.5)	4.38E-7(.5)	2.28E-7(.5)
.1	40	400	4644	5.19E-8(.5)	9.07E-8(.5)	1.09E-7(.5)	7.20E-8(.5)
.5	10	20	54	2.10E-5(.5)	3.69E-5(.5)	4.46E-5(.5)	2.87E-5(.5)
.5	40	80	916	1.29E-6(.5)	2.27E-6(.5)	2.73E-6(.5)	1.80E-6(.5)
.5	160	320	14569	8.07E-8(.5)	1.41E-7(.5)	1.70E-7(.5)	1.13E-7(.5)

Method 6--Formula 7.4 [1]

λ	I	N	Time	.1	.2	.5	1.0
.25	10	400	1477	1.05E-7(.5)	1.21E-7(.5)	2.58E-8(.5)	1.36E-8(.5)
.25	20	1600	11162	6.24E-9(.5)	7.15E-9(.5)	1.28E-9(.5)	1.01E-9(.5)
Method 7--Formula 8.30 [6]							
.25	10	400	1394	2.96E-6(.5)	1.33E-6(.5)	1.31E-7(.5)	4.22E-8(.5)
.25	20	1600	28766*	1.85E-5(.5)	8.31E-8(.5)	8.35E-9(.5)	2.59E-9(.5)
Method 8--Formula 3.40 [6]							
.5	10	200	711	3.64E-5(.5)	4.23E-5(.5)	1.21E-5(.5)	6.91E-6(.5)
.5	20	800	5175	9.38E-6(.5)	1.10E-5(.5)	3.19E-6(.5)	1.82E-6(.5)
1.0	10	100	353	7.50E-5(.5)	8.66E-5(.5)	2.44E-5(.5)	1.41E-5(.5)
1.0	20	400	2583	1.89E-5(.5)	2.21E-5(.5)	6.39E-6(.5)	3.66E-6(.5)
Method 9--Formula 3-42 [6]							
.25	10	400	1426	1.76E-5(.5)	2.04E-5(.5)	5.76E-6(.5)	3.30E-6(.5)
.25	20	1600	10231	4.66E-6(.5)	5.47-6(.5)	1.58E-6(.5)	9.04E-6(.5)

* This must be erroneous, the figures should have been about 8 x 1394 = 11, 152. Perhaps someone else was using the computer without my knowledge. I will assume 11, 152 was the time used.

APPENDIX C

Programming the Methods

The program used for testing the methods was written in Burroughs Extended ALGOL, and it was run on the Burroughs 5500 computer. The program was intended to test a number of methods, and hence, it is rather long and involved. It is designed to solve the problem $u_{xx} = u_t + f(x, t)$. When the program is used on problems of type $u_{xx} = u_t$, $f(x, t)$ must be set to 0. The most important part of the program is reproduced here. This involves the calculation of $U(x, t)$ from the values given or calculated at the previous time step. For the one multi-step method, it is assumed that the starter method has already been used on the first time step. It is assumed that there are M x steps and that $U(x, t)$ is known at the $M + 1$ grid points of the previous time step. The $M + 1$ values are stored in the one dimensional array A, (A[0] to A[M]). The M-1 values to be computed are to be stored into the one dimensional array B, (B[1] to B[M-1]). The values of $U(x, t)$ at the end points of the new time interval (B[0] and B[M]) must have been previously set. The values of $f(x, t)$ at the appropriate points in the time interval, must be calculated by the program and stored into the array FN and/or the array FN2. If $f(x, t)$ must be calculated at only one time t for each time step, only FN is used. Methods 1* and 7

require the calculation of $f(x, t)$ at two different times. In that case, $f(x_i, t_n)$, $i = 1, M-1$ is stored into FN, and $f(x_i, t_{n+1})$, $i = 1, M-1$ is stored into FN2.

The programs for the methods to be listed here are for the problem $u_{xx} = u_t + f(x, t)$, with two exceptions; (1) the program for Method 5 is for the problem $u_{xx} = u_t$ only. (2) When Method 6 is used, a change to PROCEDURE IMPLICIT is required for problems of type $u_{xx} = u_t + f(x, t)$. The revised PROCEDURE IMPLICIT is reproduced after the program for Method 6.

When the methods were tested on problems one and two of type $u_{xx} = u_t$, $f(x, t)$ was set to zero. $f(x, t)$ is stored in the array FN. This was done rather than to reprogram the methods for this special problem. In practice, the methods would be reprogrammed. For the explicit type methods the changes are obvious. For the implicit type methods the last parameter in the various calls to PROCEDURE IMPLICIT would be removed and PROCEDURE IMPLICIT would be simplified.

C is a 2 dimensional array consisting of three columns of elements. It is used to hold the tridiagonal matrix used in the solution of the set of simultaneous linear equations used with implicit methods. Q is a one dimensional array. It is used to hold the elements of the right hand side for the linear equations. E is a one dimensional array used to hold the values of $U(x, t)$ two time steps before the

present one. It is only needed for multi-step methods.

If a maximum of 1000 x steps are allowed, the Extended ALGOL array declaration would be as follows:
REAL ARRAY A [0:1000], B[0:1000], E[0:1000], Q[0:1000], C[0:1000, 0:2], FN[0:1000], FN2[0:1000]. After the testing was completed, it was discovered that the array declaration C[0:2, 0:1000] should have been used instead of C[0:1000, 0:2]. This would require minor reprogramming of the methods. The row and column designators would need to be reversed. The only reason for the change is that the array declaration used would result in gross inefficiency in a multi-programming environment on the B5500. The testing was not performed in a multi-programming environment so the test results are valid.

MULTI is an integer variable which indicates a multi-step method when set to 1. Y represents α (ALPHA). G represents λ and must have previously been set to $\frac{k}{h^2}$. W represents ω and must have previously been set to $\frac{h^2}{k}$. FACTOR represents the factor for $f(x, t)$ to be used in the right hand side of the simultaneous linear equations. The Procedure TRIDIAG will solve a set of tridiagonal linear equations. The Procedure IMPLICIT will set up the required tridiagonal matrix, and then it will call Procedure TRIDIAG to solve the system of equations.

Procedures TRIDIAG and IMPLICIT are reproduced here.

```
PROCEDURE TRIDIAG (C);
REAL ARRAY C[0, 0];
BEGIN
Q[1] ← Q[1]/C[1, 1];
FOR I ← 2 STEP 1 UNTIL M-1 DO
BEGIN
IF P = 1 OR (P ≤ 2 AND MULTI = 1) THEN
COMMENT "C ONLY HAS TO BE SET UP ONCE";
    BEGIN
        C[I, 1] ← C[I, 1] - C[I, 0] x C[I-1, 2]/C[I-1, 1];
    END;
    Q[I] ← (Q[I] - Q[I-1] x C[I, 0])/C[I, 1];
END;
B[M-1] ← Q[M-1];
FOR I ← M-2 STEP -1 UNTIL 1 DO
BEGIN
    B[I] ← Q[I] - B[I+1] x C[I, 2]/C[I, 1];
END;
END;
COMMENT "THE FOLLOWING CODE SETS UP THE REQUIRED
            TRIDIAGONAL MATRIX;
PROCEDURE IMPLICIT (D, E, F, G, FACTOR);
VALUE D, E, F, FACTOR;
REAL D, E, F, G, FACTOR;
BEGIN
FOR I ← 1 STEP 1 UNTIL M-1 DO
BEGIN
IF P = 1 OR (P ≤ 2 AND MULTI = 1) THEN
```

```
COMMENT "ONLY THE RIGHT HAND SIDE Q NEEDS TO BE SET UP
         AT EVERY TIME STEP";
BEGIN
    C[I, 1] ← D;
    C[I, 0] ← E;
    C[I, 2] ← F;
END;
Q[I] ← G + FACTOR x FN[I];
END;
Q[1] ← Q[1] - E x B[0];
Q[M-1] ← Q[M-1] - F x B[M];
TRIDIAG(C); COMMENT "NOW SOLVE THIS MATRIX";
END;
```

Only a few of the best methods are reproduced here. These are Methods 1, 1*, 3, 5, 6 and 7. Method 1 is of order $O(h^2+k)$ in general. If λ, i.e. G, is set to $\frac{1}{6}$, the order is $O(h^4+k^2)$ on problems of type $u_{xx} = u_t$, but remains $O(h^2+k)$ on problems of type $u_{xx} = u_t + f(x, t)$, $f(x, t) \neq 0$. Method 1* is of order $O(h^2+k)$ in general. If λ, i.e. G, is set to $\frac{1}{6}$, the order is $O(h^4+k^2)$ on problems of type $u_{xx} = u_t + f(x, t)$.

Method 3 is of order $O(h^2+k^2)$. If $u(x, t)$ is not sufficiently smooth to achieve higher orders on some of the other methods, then this is probably the best method available.

Method 5 is of order $O(h^2+k)$. If $a = 1 - \frac{\omega}{6}$, the order is $O(h^4+k^2)$ on problems of type $u_{xx} = u_t$. On problems of this type

the best efficiency is achieved with $Y = 1 - \frac{\omega}{6}$ and W set close to $2\sqrt{5}$ · $(W = \omega = \frac{h^2}{k})$.

Method 6 is of order $O(h^4+k^2)$. The best efficiency is achieved with λ, i.e. G, set close to $\frac{1}{2\sqrt{5}}$ for problems of type $u_{xx} = u_t$.

Method 7 is of order $O(h^4+k^2)$. The best efficiency is achieved on problems of type $u_{xx} = u_t$ with λ, i.e. G, set close to $\frac{1}{2\sqrt{15}}$ The computer algorithm for Method 1 (Classical Explicit Method) corresponding to Formula (4.1) is as follows:

```
AA:  FOR I ← 1 STEP 1 UNTIL M - 1 DO
        B[I] ← G x (A[I+1] + A[I-1]) + (1-2xG)xA[I] - K x FN[I];
        GO TO SAV ;
```

FN [I] contains (x_i, t_n). For maximum efficiency G, i.e., λ, would be set to $\frac{1}{6}$ on problems of type $u_{xx} = u_t$. Method 1 was extended to Method 1* which gives accuracy of order $O(h^4+k^2)$ on problems of type $u_{xx} = u_t + f(x, t)$ with $G = \frac{1}{6}$. (Here G represents λ). The computer algorithm corresponding to Formula (4.2) is as follows:

```
AB:   FOR I ← 1 STEP 1 UNTIL M - 1 DO
      B[I] ← Gx(A[I+1] + A[I-1]) + (1-2xG)xA[I]
-Kx(FN[I]/3 + (FN[I-1] + FN[I+1])/12 + FN2[I]/2);
      GO TO SAV;
```

FN [I] contains $f(x_i, t_n)$ and FN2 [I] contains $f(x_i, t_{n+1})$. For maximum efficiency G would be set to $\frac{1}{6}$. The computer algorithm for Method 3 (Crank-Nicholson Method) is as follows:

```
AC:     IMPLICIT (-G-1, G/2, G/2, -(G/2)x(A[I-1] + A[I+1])
    + (G-1) x A[I], K);
        GO TO SAV;
```

FN [I] contains $f(x_i, t_{n+\frac{1}{2}})$. This corresponds to Formula (4.4).

The computer algorithm for Method 5 for problems of type $u_{xx} = u_t$ is as follows:

```
AF:     IMPLICIT (2x(W+Y), -Y, -Y, (2-Y)x(A[I-1] + A[I+1])
    -2x(2-W-Y) x A[I])
        GO TO SAV;
```

This corresponds to Formula (4.5). For best efficiency, $Y = 1 - \frac{W}{6}$. W is near $2\sqrt{5}$. (Y represents α and W represents $\omega = \frac{1}{G}$).
The computer algorithm for Method 6 is as follows:

```
AP:     IMPLICIT (5/6 + G, 1/12 - G/2, 1/12 - G/2,
        (5/6 - G) x A[I]+(1/12 + G/2)x(A[I-1] + A[I+1]), -K);
        GO TO SAV;
```

This corresponds to Formula (4.8). It is also necessary to make a change to PROCEDURE IMPLICIT when Method 6 is used (unless a

problem of type $u_{xx} = u_t$ is being solved). The revised coding for IMPLICIT is as follows:

```
PROCEDURE IMPLICIT (D, E, F, G, FACTOR);
VALUE D, E, F, FACTOR;
REAL  D, E, F, G, FACTOR;
BEGIN
FOR I ← 1 STEP 1 UNTIL M-1 DO
BEGIN
IF P = 1 OR (P ≤ 2 AND MULTI = 1) THEN
COMMENT "C ONLY HAS TO BE SET UP ONCE";
BEGIN
        C[I, 1] ← D;
        C[I, 0] ← E;
        C[I, 2] ← F;
END;
Q[I] ← G + FACTOR x (5x FN[I]/6 + (FN[I-1] + FN[I+1])/12);
END;
Q[1] ← Q[1] - E x B[0];
Q[M-1] ← Q[M-1] - F x B[M];
TRIDIAG(C); COMMENT "NOW SOLVE THIS MATRIX";
END;
```

FN [I] contains $f(x_i, t_{n+\frac{1}{2}})$. For best efficiency G is set close to $\frac{1}{2\sqrt{5}}$. (G represents λ). The computer algorithm for Method 7 is as follows:

```
AH:     FOR I ← 1 STEP 1 UNTIL M - 1 DO
            B[I] ← (7/6 - 3xG) x A[I] + ((3xG - 1/6)x(A[I-1]
```

```
            + A[I+1])-(G - 1/6)x(E[I-1] - 2xE[I] + E[I+1]))/2
-Kx(FN[I]/3 + (FN[I-1] + FN[I+1])/12 + FN2[I]/2);
            GO TO SAV;
```

FN [I] contains $f(x_i, t_n)$ and FN2 [I] contains $f(x_i, t_{n+1})$. This corresponds to Formula (4.10). The routine to save the data from one time step to the next is coded as follows:

```
SAV:
   IF MULTI = 1 THEN
        BEGIN
        FOR I ← 0 STEP 1 UNTIL M DO
           E[I] ← A[I];
           COMMENT "NECESSARY ONLY FOR MULTISTEP
                    METHODS";
        END;
        FOR I ← 0 STEP 1 UNTIL M DO
           A[I] ← B[I];
```

From here the program will return to do computations for the next time step.