Polynomial Collocation Using a Domain Decomposition Solution to Parabolic PDE's via the Penalty Method and Explicit/Implicit Time Marching

Kelly Black[1]

Abstract

A domain decomposition method is examined to solve a time dependent parabolic equation. The method employs an orthogonal polynomial collocation technique on multiple subdomains. The subdomain interfaces are approximated with the aid of a penalty method. The time discretization is implemented in an explicit/implicit finite difference method. The subdomain interface is approximated using an explicit Dufort-Frankel method while the interior of each subdomain is approximated using an implicit backwards Euler's method. The principal advantage to the method is the direct implementation on a distributed computing system with a minimum of interprocessor communication. Theoretical results are given for Legendre polynomials while computational results are given for Chebyshev polynomials. Results are given for both a single processor computer and a distributed computing system.

Key Words: domain decomposition, parallel processing, orthogonal polynomials.
AMS(MOS) Subject Classifications: 65C20, 65D99, 65L20, 65M70, 65N12, 65P05.

(1) Introduction We examine a domain decomposition method for solving a parabolic equation using orthogonal polynomials. The model for the parabolic problem is a heat equation:

$$\begin{array}{rcll} u_t & = & u_{xx} - k^2 u & x \in \Omega \ , \ t > 0 \\ u(x,0) & = & f(x) & x \in \Omega \\ u(x,t) & = & 0 & x \in \partial\Omega \ , \ t > 0 \end{array} \tag{1.1}$$

The treatment of the multi-domain approach and the subdomain interfaces has received a great deal of attention. Methods that have enforced strict continuity and smoothness conditions are among the many candidates that have been examined. While the solutions such methods generate are accurate and robust they require a global knowledge of the approximation. The approximation across all of the subdomains is used to insure the smoothness necessary. Examples of these techniques can be found in [7].

Requiring that the approximation be considered across all of the subdomains can be a drawback. The principal thrust of our work focuses on applications to high speed parallel computing on distributed memory machines. The resources that are expended on the techniques described above is expensive on such machines. In order to get around this we examine methods that use more relaxed requirements on the boundary. Rather than enforce strict smoothness conditions the method places a penalty on approximations that are not smooth. The method examined is similar to the penalty method in [4], however, we examine the technique for parabolic problems and retain a time dependency. Other techniques that utilize a relaxation on the boundary include [3, 10].

The spatial approximation technique that we use is a Chebyshev collocation method. Orthogonal polynomial techniques suffer from many drawbacks. For example, the condition numbers of the matrices generated tend to be large; the condition number for the Chebyshev

[1] Center for Research in Scientific Computation, North Carolina State University, Box 8205, Raleigh, NC 27695-8205. (black@crsc1.math.ncsu.edu)
Work supported by DARPA grant N-00014-91-J-4016 and AFOSR Grant No. 90-0093.

second derivative matrix on a grid of size N is $O(N^4)$ compared to $O(N^2)$ for a finite difference method.

Despite the disadvantages orthogonal polynomials do give accurate approximations when the solutions are smooth. When the solutions have up to p continuous derivatives the accuracy of the approximation is of the order N^{-p}. More importantly the solutions generated can be calculated using a smaller number of points than other techniques such as finite difference and finite element methods. Because the equation examined is a heat equation the solution approximated tends to an infinitely smooth solution and orthogonal polynomials offer an accurate estimation of the solution.

Finally, the most unique aspect of the work demonstrated here is the method of the time discretization. The method presented is an explicit/implicit method. The subdomain interfaces are calculated using the explicit Dufort-Frankel method scheme and the interior of the subdomains are calculated using an implicit backwards Euler scheme.

(2) Chebyshev and Legendre Polynomial Collocation Before examining the penalty method the basic properties of the Chebyshev and Legendre polynomials are examined. The collocation formulation and the second derivative matrices for both are examined.

(2.1) Chebyshev Polynomials The first class of orthogonal polynomials examined are the Chebyshev polynomials of the first kind. Chebyshev polynomials are generated by the weight $w(x) = \frac{1}{\sqrt{1-x^2}}$. The advantage to these polynomials is that the Gauss-Lobatto quadrature is known in a closed form.

The Gauss-Lobatto quadrature can be calculated in a closed form [2]:

$$\begin{aligned} x_l &= \cos\frac{\pi l}{N} \\ w_l &= \begin{cases} \frac{\pi}{2N} & l = 0, N \\ \frac{\pi}{N} & 1 \le l \le N-1 \end{cases} \end{aligned} \tag{2.1}$$

The collocation formula can be calculated in a straightforward manner. The approximations are projected onto the finite space of all polynomials of degree less than or equal to N and the space is denoted $\mathbb{P}_N$. Note that the projection operator yields the unique polynomial interpolating the Gauss-Lobatto quadrature points, and the extrema of the Chebyshev polynomial $T_N(x)$ coincide with the Gauss-Lobatto quadrature points. Since $T_N(x)$ is a polynomial and continuous it's derivative is zero on the grid points. The Lagrange polynomial that interpolates the points δ_{ij} can be written in terms of $T_N(x)$:

$$t_j(x) = \frac{(-1)^j}{c_j N^2} \frac{T_N'(x)(1-x^2)}{(x-x_j)} \tag{2.2}$$

The collocation projection operator yields the interpolating polynomial on the grid points and is written in terms of $t_j(x)$ [1]:

$$P_N f(x) = \sum_{j=0}^{N} f(x_j) t_j(x) \tag{2.3}$$

Derivatives of $P_N f$ are calculated by finding the derivatives of $t_j(x)$. When the projection

is written as a column vector, the derivatives can be written as a matrix multiplication:

$$\frac{d^2}{dx^2}P_N f(x_l) = \sum_{j=0}^{N} f(x_j)t_l''(x_l) \tag{2.4}$$

$$\frac{d^2}{dx^2}\begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) \end{bmatrix} = D_N^2 \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) \end{bmatrix}$$

The Chebyshev second derivative matrix is found by evaluating the second derivative of the interpolating polynomial, $t_j(x)$, on the Gauss-Lobatto quadrature [6]:

$$D_N^2 = [d_{lj}^2] = \begin{cases} \frac{N^4-1}{15} & l = j; l = 0, N \\ -\frac{(N^2-1)(1-x_j^2)+3}{3(1-x_j^2)^2} & l = j; 1 \le j \le N-1 \\ \frac{2}{3}\frac{(-1)^j}{c_j}\frac{(2N^2+1)(1-x_j)-6}{(1-x_j)^2} & l = 0; 1 \le j \le N \\ \frac{2}{3}\frac{(-1)^{N+j}}{c_j}\frac{(2N^2+1)(1+x_j)-6}{(1+x_j)^2} & l = N; 0 \le j \le N-1 \\ \frac{(-1)^{(l+j)}}{c_j}\frac{x_l^2+x_l x_j-2}{(1-x_l^2)(x_l-x_j)^2} & 1 \le l \le N-1; 0 \le j \le N \\ & l \ne j \end{cases} \tag{2.5}$$

The second derivative matrix is modified slightly in applications to accommodate boundary conditions. The top and the bottom rows are replaced with zeros. This is used to solve the heat equation with zero boundary conditions. In their work, Gottlieb & Lustman showed that the matrix is stable and the eigen values are real, distinct, and negative [6, p. 5]. These results were further generalized in [8] for generalized boundary conditions.

(2.2) Legendre Polynomials One simple weight function is the identity, $w(x) = 1$. Polynomials generated by this weight are known as the Legendre polynomials and are denoted as $L_k(x)$. Unfortunately the Gauss-Lobatto quadrature is not known in a closed form. The quadrature has to be calculated and the grid points are denoted as x_j and the weights are denoted as w_j for $0 \le j \le N$. The collocation derivatives can be calculated using the Lagrange polynomial that interpolates the points δ_{ij}:

$$l_j(x) = (-1)^j w_j \frac{L_N'(x)(1-x^2)}{x - x_j} \tag{2.6}$$

The result is the second derivative collocation matrix is written in terms of the associated Lagrange polynomial:

$$D_N^2 = [l_{ij}^2] = [\frac{d^2}{dx^2} l_i(x_j)] \tag{2.7}$$

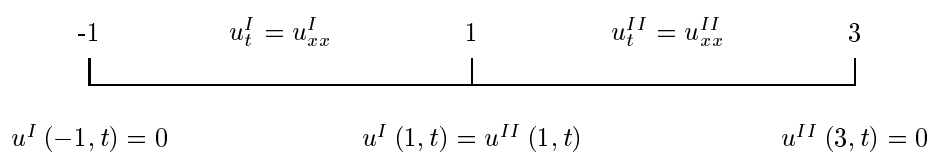
The derivative matrices are established using the same method as the Chebyshev derivative matrices. The second derivative Legendre matrix has similar properties to that of the corresponding Chebyshev matrix. The order of the matrix is $O(N^4)$ and all of the eigen values are real, negative, and distinct [11, p. 313].

(3) The Penalty Method in One Dimension The penalty method is introduced for problems in one dimension. First an example using two subdomains is presented including stability restrictions. Also the stability conditions for the general case are presented.

(3.1) Example of Two Subdomains The penalty method adds a penalty for solutions that are not smooth during the time stepping. At each time step the derivatives on the subdomain interfaces are calculated by averaging the approximations on either side of the interface. Along with this a proportion of the difference between the first derivatives between the subdomain interfaces is added as a penalty. The problem we examine is a heat equation with Dirichlet boundaries:

$$\begin{array}{rcll} u_t(x,t) & = & u_{xx}(x,t) & x \in \Omega, t > 0 \\ u(x,0) & = & f(x) & x \in \Omega, t = 0 \\ u(x,t) & = & 0 & x \in \partial\Omega, t > 0 \end{array} \tag{3.1}$$

Figure 1: The domain $[-1,3]$ divided into two subdomains



The first example will examine a heat equation on two domains. In order to simplify the analysis a simple domain is chosen and the number of collocation points is the same on both subdomains. The domain is $[-1,3]$ which is chosen because it can be easily divided into two subdomains: subdomain I is $[-1,1]$ and subdomain II is $[1,3]$ (figure (1)). Within subdomains I and II the functions u^I and u^{II} are solutions to the heat equation. On the boundaries of the full domain the Dirichlet boundaries are enforced. A stability condition can be found by examining the time derivative of a weak energy. The stability is shown for a Legendre collocation method and the quadrature used is the Gauss-Lobatto quadrature.

The equations generated can be rewritten as a system:

$$\vec{u} = \vec{u}(x,t) = \begin{pmatrix} u^I(x,t) \\ u^{II}(2-x,t) \end{pmatrix} \qquad x \in [-1,1] \tag{3.2}$$

Note that the approximation u^{II} is mapped so that the value of x is translated and "flipped" onto $[-1,1]$. This way it is mapped onto a single domain by flipping it over and it can be written as a system of equations. This has the advantage that it maintains the subdomain interface within the mapping which makes it easier to apply the subdomain interface boundary conditions.

A few important properties arise from the system:

$$\begin{array}{rcl} \vec{u}(-1,t) & = & \begin{pmatrix} u^I(-1,t) \\ u^{II}(3,t) \end{pmatrix} \\ \vec{u}_x(1,t) & = & \begin{pmatrix} u_x^I(-1,t) \\ -u_x^{II}(3,t) \end{pmatrix} \end{array} \tag{3.3}$$

$$\vec{u}_{xx}(1,t) = \begin{pmatrix} u^I_{xx}(-1,t) \\ u^{II}_{xx}(3,t) \end{pmatrix}$$

The system is solved by using an orthogonal polynomial method. The resulting analysis begins by showing the stability of this system for Legendre polynomials approximations. The system is approximated using Legendre polynomials and the Gauss-Lobatto quadrature is employed. The weights on the grid are denoted as w^I_i and w^{II}_i for subdomains I and II respectively. The values for the approximations on the grid are denoted u^I_i and u^{II}_i where $u^I_i = P_{N^I} u^I(x_i,t)$ and $u^{II}_i = P_{N^{II}} u^{II}(x_i,t)$.

In the spacial discretization a penalty method is employed to approximate the solution at the subdomain interface. First the second derivative is approximated by averaging the second derivatives across the subdomain interface: $\sigma^I (u^I_0)_{xx} + \sigma^{II}(u^{II}_{N^{II}})_{xx}$. Next, a penalty is added that is in proportion to the difference between the first derivatives at the subdomain interface: $\lambda((u^{II}_{N^{II}})_x - (u^I_0)_x)$. The values for σ^I,σ^{II}, and λ are found in the following stability analysis. The result is an approximation for the time derivative at the subdomain interface:

$$\begin{aligned} (u^I_0)_t = (u^{II}_{N^{II}})_t &= \sigma^I (u^I_0)_{xx} + \sigma^{II}(u^{II}_{N^{II}})_{xx} \\ &\quad + \lambda((u^{II}_{N^{II}})_x - (u^I_0)_x) \end{aligned} \tag{3.4}$$

This equation is examined in terms of the collocation method. Note that the collocation method yields a solution on the grid points [4]:

$$\begin{aligned} u^I_{N^I} &= 0 \\ (u^I_i)_t &= (u^I_i)_{xx} \qquad 1 \le i \le N^I - 1 \\ (u^I_0)_t w^I_0 + (u^{II}_{N^{II}})_t w^{II}_{N^{II}} &= (u^I_0)_{xx} w^I_0 + (u^{II}_{N^{II}})_{xx} w^{II}_{N^{II}} \\ &\quad + (u^I_0)_x w^I_0 - (u^{II}_{N^{II}})_x w^{II}_{N^{II}} \\ (u^{II}_i)_t &= (u^{II}_i)_{xx} \qquad 1 \le i \le N^{II} - 1 \\ u^{II}_{N^{II}} &= 0 \end{aligned} \tag{3.5}$$

With these equations and for $\phi^I \in \mathbb{P}_{N^I}$ and $\phi^{II} \in \mathbb{P}_{N^{II}}$ the following systems are defined[4]:

$$\begin{aligned} \sum_{i=1}^{N^I} \phi^I_i (u^I_i)_t w^I_i &= \sum_{i=1}^{N^I} \phi^I_i (u^I_i)_{xx} w^I_i \\ \phi^I_0 (u^I_0)_t w^I_0 + \phi^{II}_{N^{II}} (u^{II}_{N^{II}})_t w^{II}_{N^{II}} &= \phi^I_0 (u^I_0)_{xx} w^I_0 + \phi^{II}_{N^{II}} (u^{II}_{N^{II}})_{xx} w^{II}_{N^{II}} \\ &\quad + \phi^I_0 (u^I_0)_x w^I_0 - \phi^{II}_{N^{II}} (u^{II}_{N^{II}})_x w^{II}_{N^{II}} \\ \sum_{i=0}^{N^{II}-1} \phi^{II}_i (u^{II}_i)_t w^{II}_i &= \sum_{i=0}^{N^{II}-1} \phi^{II}_i (u^{II}_i)_{xx} w^{II}_i \end{aligned} \tag{3.6}$$

The sums on the right hand side of equations (3.6) are collected into one sum:

$$\begin{aligned} \sum_{i=0}^{N^I} \phi^I_i (u^I_i)_t w^I_i + \sum_{i=0}^{N^{II}} \phi^{II}_i (u^{II}_i)_t w^{II}_i &= \sum_{i=0}^{N^I} \phi^I_i (u^I_i)_{xx} w^I_i + \sum_{i=0}^{N^{II}} \phi^{II}_i (u^{II}_i)_{xx} w^{II}_i \\ &\quad + \phi^I_0 (u^I_0)_x w^I_0 - \phi^{II}_{N^{II}} (u^{II}_{N^{II}})_x w^{II}_{N^{II}} \end{aligned} \tag{3.7}$$

$$
\begin{aligned}
&= \int_{-1}^{1} \phi^I u^I_{xx} dx + \int_{1}^{3} \phi^{II} u^{II}_{xx} dx \qquad (3.8)\\
&\quad + \phi^I_0 (u^I_0)_x w^I_0 - \phi^{II}_{N^{II}} (u^{II}_{N^{II}})_x w^{II}_{N^{II}} \\
&= -\int_{-1}^{1} \phi^I_x u^I_x dx - \int_{1}^{3} \phi^{II}_x u^{II}_x dx \qquad (3.9)
\end{aligned}
$$

The solution is written as a column vector on the Legendre Gauss-Lobatto points:

$$
P_N \vec{u} = \vec{u}_N = \left(u^I_{N^I} \cdots u^I_0, u^{II}_{N^{II}} \cdots u^{II}_0 \right)^T \qquad (3.10)
$$

The derivatives are written in terms of the collocation matrix from equation (2.7). The top and the bottom rows of the second derivative matrix are modified such that the matrix mirrors the interface conditions:

$$
\begin{aligned}
\hat{D}^2_{N^I} &= [\hat{d}^2_{ij}] \qquad (3.11)\\
&= \begin{cases} 0 & i = N^I \\ l^2_{ij} & 1 \le i < N^I, 0 \le j \le N^I \\ \frac{w^I}{w^I_0 + w^{II}_0} l^2_{N^I j} + \frac{1}{w^I_0 + w^{II}_0} l'_{N^I}(x_j) & i = 0, 0 \le j \le N^I \end{cases} \\
\hat{D}^2_{N^{II}} &= [\hat{d}^2_{ij}] \qquad (3.12)\\
&= \begin{cases} \frac{w^I}{w^I_0 + w^{II}_0} l^2_{N^{II} j} + \frac{1}{w^I_0 + w^{II}_0} l'_{N^{II}}(x_j) & i = N^{II}, 0 \le j \le N^{II} \\ l^2_{ij} & 1 \le i < N^{II}, 0 \le j \le N^{II} \\ 0 & i = 0 \end{cases}
\end{aligned}
$$

The resulting operator for the penalty method is written in terms of the two matrices [4]:

$$
\begin{aligned}
S = [s_{ij}] &= \begin{bmatrix} \hat{D}^2_{N^I} & 0 \\ 0 & \hat{D}^2_{N^{II}} \end{bmatrix} \qquad (3.13)\\
(\vec{u}_N)_t &= S \vec{u}_N \qquad (3.14)
\end{aligned}
$$

This system is equivalent to the system in (3.6) where the weights given in equation (3.4) are set according to the Legendre Gauss-Lobatto weights:

$$
\begin{aligned}
\sigma^I &= \frac{w^I_0}{w^I_0 + w^{II}_0} \qquad (3.15)\\
\sigma^{II} &= \frac{w^{II}_0}{w^I_0 + w^{II}_0} \\
\lambda &= \frac{1}{w^I_0 + w^{II}_0}
\end{aligned}
$$

Theorem 3.1.1 *Given the system of equations in equation (3.6) the discretization by the Legendre polynomial collocation method is stable for a different number of grid points in the two subdomains. Subdomain I has N^I grid points and subdomain II has N^{II} grid points.*

Proof: For the test function ϕ substitute $\vec{u}_N$. Since each $w^I_i, w^{II}_i > 0$ and from equation (3.7) the result from equation (3.9) gives a bound on the time derivative:

$$
\frac{1}{2}\partial_t \left(\sum_{i=0}^{N^I} (u^I_i)^2 w^I_i + \sum_{i=0}^{N^{II}} (u^{II}_i)^2 w^{II}_i \right) = \sum_{i=0}^{N^I} u^I_i (u^I_i)_t w^I_i + \sum_{i=0}^{N^{II}} u^{II}_i (u^{II}_i)_t w^{II}_i \qquad (3.16)
$$

$$
\begin{aligned}
&= -\int_{-1}^{1} (\vec{u}_N)_x (\vec{u}_N)_x^T \, dx \\
&\leq 0
\end{aligned}
$$

The last step follows since the integrand is positive. □

(3.2) The Generalized Penalty Method A more generalized stability condition can be found by looking at more than two subdomains with varying grids. The averaging weights are denoted as σ_0^l and σ_1^l. σ_0^l is the averaging weight for the right side of the domain l while σ_1^l is the averaging weight for the left side of domain l. Also, the penalty is different for each subdomain interface. The penalties are denoted λ^l for the subdomain interface on the right side of domain l.

For a given subdomain, l, the penalty method gives the following equation on the interior of a subdomain:

$$
(u_{N^l}^l(x,t))_t = (u_{N^l}^l(x,t))_{xx} \qquad x \in (2l-3, 2l-1) \qquad l = 1, 2, \cdots, L \tag{3.17}
$$

The averaging and penalty weights give a condition on the subdomain interfaces for $t > 0$:

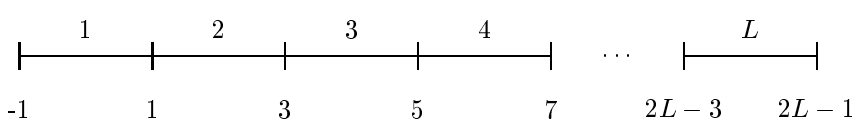
$$
\begin{aligned}
\partial_t (u_{N^l}^l(2l-3,t)) &= \sigma_1^l (u_{N^l}^l(2l-3,t))_{xx} + \sigma_0^{l-1} (u_{N^{l-1}}^{l-1}(2l-3,t))_{xx} \\
&\quad + \lambda^l ((u_{N^l}^l(2l-3,t))_x - (u_{N^{l-1}}^{l-1}(2l-3,t))_x) \qquad l = 1, 2, \cdots, L
\end{aligned} \tag{3.18}
$$

$$
\begin{aligned}
(u_{N^l}^l(2l-1,t))_t &= \sigma_0^l (u_{N^l}^l(2l-1,t))_{xx} + \sigma_1^{l+1} (u_{N^{l+1}}^{l+1}(2l-1,t))_{xx} \\
&\quad + \lambda^l ((u_{N^{l+1}}^{l+1}(2l-1,t))_x - (u_{N^l}^l(2l-1,t))_x) \qquad l = 0, 1, \cdots, L-1
\end{aligned} \tag{3.19}
$$

$$
u_{N^1}^1(-1,t) = u_{N^L}^L(2L-1,t) = 0 \tag{3.20}
$$

$$
u_{N^l}^l(x,0) = P_{N^l} f(x) \qquad x \in (-1, 2L-1) \tag{3.21}
$$

Figure 2: Domain Divided into L Subdomains for the One Dimensional Case



Theorem 3.2.1 *Given the system of equations defined by equation (3.17) with boundary conditions given by equations (3.18), (3.19), (3.20), and (3.21) the discretization by the Legendre polynomial collocation method is stable for L subdomains with varying numbers of grid points per subdomain. The number of grid points for a domain l is denoted as N^l and the Legendre weights are denoted w_i^l. Stability is insured for averaging weights and penalties set dependent on the quadrature:*

$$\begin{aligned}
\sigma_0^l &= \frac{w_0^l}{w_0^l + w_0^{l+1}} \\
\sigma_1^l &= \frac{w_0^l}{w_0^l + w_0^{l-1}} \\
\lambda^l &= \frac{1}{w_0^l + w_0^{l+1}}
\end{aligned}$$

Proof: The proof proceeds as before by examining the time derivative of the weak energy and showing that it is bounded. The subdomains in question are assumed to be uniform with length equal to 2 (figure (2)).

The subdomains are folded so as to preserve the subdomain interfaces and the solution, $\vec{u}_N$, is written as a vector. Assuming that L is an even number the solution is written in vector form:

$$\vec{u}_N(x,t) = \begin{pmatrix} u_{N^1}^1(x,t) \\ u_{N^2}^2(2-x,t) \\ u_{N^3}^3(4+x,t) \\ u_{N^4}^4(6-x,t) \\ \vdots \\ u_{N^L}^L(2L-2-x,t) \end{pmatrix} \qquad x \in [-1,1] \tag{3.22}$$

The elements within the vector satisfy the conditions given by equation (3.17) and the vector solves the heat equation:

$$(\vec{u}_N)_t = (\vec{u}_N)_{xx} \qquad x \in (-1,1) \tag{3.23}$$

The boundary conditions for the equation are given by equations (3.18), (3.19), (3.20), and (3.21).

The averaging and penalty weights yield a similar collocation matrix as in equation (3.13) and the results from equations (3.9) and (3.7) still hold when extended to the general case. Just as in theorem 3.1.1 the time derivative of the weak energy can be shown to be bounded and the method is stable. □

(4) Time Discretization

The spatial approximations used are an orthogonal polynomial collocation method. The spatial discretizations are written in terms of families of orthogonal polynomials. The time discretizations, on the other hand, are made using standard finite difference techniques. The two techniques we use are the Dufort-Frankel method to solve for the solution on the subdomain interface and the implicit backwards Euler method to solve for the solution on the interior of each subdomain. The two methods are combined in an explicit/implicit method.

(4.1) The Explicit/Implicit Method The penalty method can be implemented using an explicit/implicit technique. The interior of the subdomains are approximated using an implicit technique. In particular, a backwards Euler scheme is used. However, to use the implicit method an a priori knowledge of the subdomain interface is needed. This information is calculated using an explicit method. Working together the two methods are used to calculate a solution on each subdomain.

First the explicit method is used to calculate a solution on each subdomain interface. This is done with the penalty method via the Dufort-Frankel scheme. Once a future value is known on the interface the interior is calculated using an implicit method.

The advantage to such an approach is to take advantage of the inherent stability of an implicit scheme. Domain decomposition is an ideal way to exploit this. The principle disadvantage of an implicit scheme is the difficulty in inverting the associated matrices. Using a domain decomposition method, however, allows the use of a smaller grid on each individual subdomain. Consequently, smaller matrices are generated within each subdomain and the matrices can be inverted in parallel.

(4.2) The Explicit Method Before the implicit method is used to calculate the value of the interior of a subdomain the future value on the subdomain interface is calculated using an explicit method. On the subdomain interface the future time step is calculated using the Dufort-Frankel method:

$$\frac{u^{n+1} - u^{n-1}}{2\triangle t} = Su^n - \gamma \frac{1}{\triangle x^2}(u^{n+1} - 2u^n + u^{n-1}) \tag{4.1}$$

The operator S is the penalty method approximation at the subdomain interface (equations (3.18) and (3.19)) and $\triangle x$ is calculated at the endpoints, $\triangle x = x_0 - x_1$. The Dufort-Frankel scheme has been shown to be unconditionally stable when used on the interior of a single domain problem [5] and in particular for the Chebyshev approximation [6].

The scheme is consistent with the heat equation if the ratio $\frac{\triangle t}{\triangle x}$ goes to zero. The method is time accurate with order $O(\triangle t^2) + O\left(\left(\frac{\triangle t}{\triangle x}\right)^2\right)$. One way to maintain accuracy is to make the ratio $\frac{\triangle t}{\triangle x^2}$ constant, which will insure that the ratio $\frac{\triangle t}{\triangle x} \to 0$. The scheme is consistent with a wave equation [5]:

$$u_t + \gamma \left(\frac{\triangle t}{\triangle x}\right)^2 u_{tt} = u_{xx} \tag{4.2}$$

The new equation retains the same steady state solution as the heat equation. γ is chosen such that the scheme is unconditionally stable [5].

(4.3) The Implicit Method Once the future value at the subdomain interface is known the interior of a subdomain is calculated using an implicit method. In the one-dimensional case this is done using a backwards Euler equation. In the two-dimensional case, however, a splitting method is used. The method is solved in an implicit manner in alternating directions.

(4.3.1) One Dimensional Implicit Method On the interior of the domain the heat equation is approximated using a backwards Euler equation:

$$\frac{u^{n+1} - u^n}{\triangle t} = D_N^2 u^{n+1} \tag{4.3}$$

Where D_N^2 is the second derivative operator on $\mathbb{P}_N$. In the case examined we use a Chebyshev second derivative collocation matrix. As shown in [6, p. 5] the Chebyshev second derivative matrix has real, negative, and distinct eigen values when used with Dirichlet boundary conditions.

(4.3.2) Two Dimensional Implicit Method Implementing an implicit method in the two dimensional case requires solving a larger system than the one dimensional case. In order to reduce the complexity of the problem a splitting method is used. In the splitting method an implicit solution is sought at each time step but in alternating directions [9, p. 180].

In the two dimensional case, in one step an implicit solution is sought in the y-direction while in the following step an implicit solution is sought in the x-direction:

$$\begin{aligned} &\text{Step } n: && \frac{u^{n+\frac{1}{2}}-u^n}{\frac{\triangle t}{2}} = D_x^2(u^n)^T + D_y^2 u^{n+\frac{1}{2}} \\ &\text{Step } n+\tfrac{1}{2}: && \frac{(u^{n+1})^T-(u^{n+\frac{1}{2}})^T}{\frac{\triangle t}{2}} = D_x^2(u^{n+1})^T + D_y^2 u^{n+\frac{1}{2}} \end{aligned} \tag{4.4}$$

In practice, for grids of size N_x in the x-direction and size N_y in the y-direction, the second derivative matrices are of size $N_x \mathrm{x} N_x$ and $N_y \mathrm{x} N_y$ for the x and y derivative matrices, respectively. The approximation u^n is a $N_x \mathrm{x} N_y$ matrix and the inverse operation takes place on a column by column basis. The resulting operation proceeds in each of the columns of the matrix u^n.

The analysis of the splitting method proceeds in a different manner. First the approximation u^n is arranged in a vector. The entries are formed by arranging the column such that the first N_x entries are from the first row of the matrix u^n and the next N_x entries are from the second row of u^n. This process proceeds until all of the rows of u^n have been assigned.

The derivative matrices are constructed accordingly:

$$D_x^2 = \begin{bmatrix} D_x^2 & 0 & & 0 \\ 0 & D_x^2 & & 0 \\ & & \ddots & \\ 0 & 0 & & 0 \\ 0 & 0 & & D_x^2 \end{bmatrix} \tag{4.5}$$

The second derivative matrix in the y direction is constructed to approximate the y derivative and so the entries must skip over the rows of x values. To display this we first define a matrix, $\tilde{D_y^2}$:

$$\left(\tilde{D_y^2}\right)_{i,j} = \begin{bmatrix} d_{i,j}^2 & 0 & & 0 \\ 0 & d_{i,j}^2 & & 0 \\ & & \ddots & \\ 0 & 0 & & d_{i,j}^2 \end{bmatrix} \tag{4.6}$$

The entries $d_{i,j}^2$ correspond to the entries (i,j) in the one-dimensional second derivative collocation matrix.

The second derivative matrix is constructed using this new matrix:

$$D_y^2 = \begin{bmatrix} \left(\tilde{D_y^2}\right)_{0,0} & \left(\tilde{D_y^2}\right)_{0,1} & \cdots & \left(\tilde{D_y^2}\right)_{0,N_y} \\ \left(\tilde{D_y^2}\right)_{1,0} & \left(\tilde{D_y^2}\right)_{1,1} & \cdots & \left(\tilde{D_y^2}\right)_{1,N_y} \\ & \vdots & & \\ \left(\tilde{D_y^2}\right)_{N_y,0} & \left(\tilde{D_y^2}\right)_{N_y,1} & \cdots & \left(\tilde{D_y^2}\right)_{N_y,N_y} \end{bmatrix} \tag{4.7}$$

The steps in equation (4.4) are first solved in terms of the highest time level:

$$\begin{aligned} &\text{Step } n: && (I - \tfrac{\triangle t}{2} D_y^2) u^{n+\frac{1}{2}} = (I + \tfrac{\triangle t}{2} D_x^2) u^n \\ &\text{Step } n+\tfrac{1}{2}: && (I - \tfrac{\triangle t}{2} D_x^2) u^{n+1} = (I + \tfrac{\triangle t}{2} D_y^2) u^{n+\frac{1}{2}} \end{aligned} \tag{4.8}$$

The approximation of $u^{n+\frac{1}{2}}$ from step n gives an implicit system:

$$u^n = (I - \frac{\triangle t}{2} D_y^2)^{-1} (I + \frac{\triangle t}{2} D_x^2) u^{n-1} \tag{4.9}$$

Substituting this result into step $n + \frac{1}{2}$ for the approximation $u^{n+\frac{1}{2}}$ gives an approximation for u^{n+1}:

$$u^{n+1} = (I - \frac{\triangle t}{2} D_x^2)^{-1} (I + \frac{\triangle t}{2} D_y^2) (I - \frac{\triangle t}{2} D_y^2)^{-1} (I + \frac{\triangle t}{2} D_x^2) u^n \tag{4.10}$$

Adopting the notation of [9, p. 113] this gives a matrix operator:

$$T = (I - \frac{\triangle t}{2} D_x^2)^{-1} (I + \frac{\triangle t}{2} D_y^2) (I - \frac{\triangle t}{2} D_y^2)^{-1} (I + \frac{\triangle t}{2} D_x^2) \tag{4.11}$$

Again, as shown in [6, p. 5], the Chebyshev second derivative operator is stable and the splitting method is a stable method.

(5) Results Results are presented for four different cases. The results of the penalty method are given for the penalty method in one-dimension with a single processor and for a distributed machine. Results are also given for the two-dimensional case for both the single processor and the distributed machines. The single processor machine examined is an IBM RISC/6000 and the distributed machine is an Intel iPSC/i860 with thirty-two nodes.

For the one-dimensional case the following equation is examined:

$$\begin{aligned} u_t &= u_{xx} - k^2 u && x \in (0,16),\ t > 0 \\ u(x,0) &= \sin(\omega \tfrac{2\pi}{16} x) + e^{kx} && x \in [0,16] \\ u(0,t) &= 1 && t > 0 \\ u(16,t) &= e && t > 0 \end{aligned} \tag{5.1}$$

For the two-dimensional case the following equation is examined:

$$\begin{aligned} u_t &= u_{xx} + u_{yy} - 2k^2 u && (x,y) \in (0,16)\text{x}(0,16),\ t > 0 \\ u(x,y,0) &= \sin(\omega \tfrac{2\pi}{16} x) \sin(\omega \tfrac{2\pi}{16} y) + e^{k(x+y)} && (x,y) \in (0,16)\text{x}(0,16) \\ u(x,0,0) &= e^{kx} && x \in (0,16) \\ u(x,16,0) &= e^{k(x+16)} && x \in (0,16) \\ u(0,y,0) &= e^{ky} && y \in [0,16] \\ u(16,y,0) &= e^{k(y+16)} && y \in [0,16] \end{aligned} \tag{5.2}$$

For each of the test runs, the tables give the number of subdomains, the number of grid points within each subdoman, the number of steps required to reach steady state, the L^1 and the L^∞ errors, the ratio $\frac{\triangle t}{\triangle x^2}$ or $\triangle t\left(\frac{1}{\triangle x^2}+\frac{1}{\triangle y^2}\right)$, and the time required. Note that the $\triangle t$ given in the tables coresponds to the $\frac{\triangle t}{2}$ used in equation (4.4).

(5.1) Single processor in One Dimension Results are presented for the one-dimensional case. The values for the initial conditions are $\omega = 5$ and $k = \frac{1}{16}$. The cases presented are from runs on a single processor machine. Tables (1) and (2) are for test runs where the time step, $\triangle t$, is constant while tables (3) and (4) modify the time step so that the ratio $\frac{\triangle t}{\triangle x^2}$ is constant ($\triangle x$ is taken to be the minimum distance between grid points). Tables (2) and (4) are the ideals and are used to compare the results of the penalty method.

In table (1) the increase in the number of subdomains requires an increase in the number of iterations to solve for the steady state. However, once the ratio $\frac{\triangle t}{\triangle x^2}$ becomes small enough the number of iterations is reduced. This is demonstrated in table (3) where the number of iterations decreases. Once the number of subdomains is more than 16, though, more iterations are required.

Figure (3) represents the times given in tables (1) and (2). Figure (4) is from tables (3) and (4). In figure (3) the time required to calculate the steady state solution decreases consistently but the rate of decrease is not as rapid as for the ideal case. In figure (4), however, the rate of decrease is steeper than the ideal case until the number of subdomains is more than 16.

Table 1: Single processor test run. (1-D Case) $\triangle t$ constant: $\gamma = 0.2$

Calculated Boundaries
Frequency = 5.0 k = 0.0625 dt = 0.125

# X Dom	Grid Points In X	# Steps	L^1 Error	L^∞ Error	$\frac{\triangle t}{\triangle x^2}$	Elapsed Time (msec.)
1	256	20	1.07e-05	1.67e-05	3.44e+05	39100
2	128	20	1.06e-05	1.67e-05	8.61e+04	5080
4	64	1820	2.68e-05	4.08e-05	2.15e+04	5310
8	32	2510	1.78e-05	3.70e-05	5.39e+03	3710
16	16	2020	2.63e-05	4.18e-05	1.35e+03	1550
32	8	1320	1.12e-05	1.91e-05	3.45e+02	860

(5.2) Distributed Machine in One Dimension Results are presented for the one-dimensional case. The values for the initial conditions are $\omega = 5$ and $k = \frac{1}{16}$. The cases presented are from runs on a distributed machine. Tables (5) through (7) are for test runs with constant time step $\triangle t$. Tables (8) through (10) are for test runs with constant ratio $\frac{\triangle t}{\triangle x^2}$. In tables (6) and (9) the true time dependent solution is enforced at the subdomain interface but the interprocessor communication is retained. In tables (7) and (10), however, the true solution is enforced at the subdomain interface and the interprocessor communication is retained. The later case is an ideal and is used to measure the overhead associated with communication costs.

Table 2: Single processor test run. (1-D Case) $\triangle t$ constant with true boundaries.

True Boundaries
Frequency = 5.0 k = 0.0625 dt = 0.125

# X Dom	Grid Points In X	# Steps	L^1 Error	L^∞ Error	$\frac{\triangle t}{\triangle x^2}$	Elapsed Time (msec.)
1	256	20	1.07e-05	1.67e-05	3.44e+05	38930
2	128	20	1.06e-05	1.67e-05	8.61e+04	5120
4	64	70	1.78e-05	2.79e-05	2.15e+04	850
8	32	30	4.88e-06	1.08e-05	5.39e+03	140
16	16	10	6.96e-06	1.67e-05	1.35e+03	20
32	8	10	5.17e-10	1.27e-09	3.45e+02	10

Figure 3: Graph of the time required on the single processor machine to solve the one dimensional problem. $\triangle t$ constant

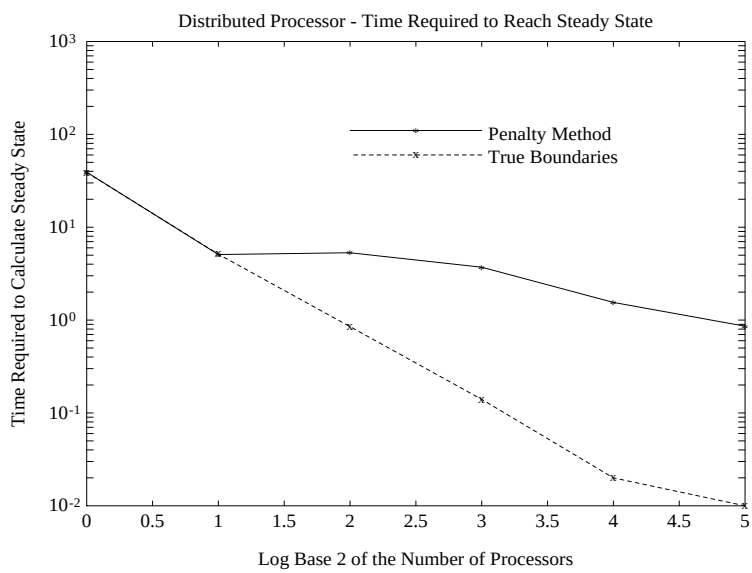


Table 3: Single processor test run. (1-D Case) $\frac{\triangle t}{\triangle x^2}$ constant: $\gamma = 0.2$

Calculated Boundaries
Frequency = 5.0 k = 0.0625 $\frac{\triangle t}{\triangle x^2} = 100.0$

# X Dom	Grid Points In X	# Steps	L^1 Error	L^∞ Error	$\triangle t$	Elapsed Time (msec.)
1	256	50000	5.79e-04	9.10e-04	3.63e-05	480180
2	128	17680	3.18e-05	5.00e-05	1.45e-04	87870
4	64	46300	3.18e-05	5.00e-05	5.80e-04	117560
8	32	2190	3.18e-05	4.99e-05	2.32e-03	3250
16	16	170	5.42e-06	1.20e-05	9.23e-03	150
32	8	360	1.88e-05	4.65e-05	3.62e-02	240

Table 4: Single processor test run. (1-D Case) $\frac{\triangle t}{\triangle x^2}$ constant.

True Boundaries
Frequency = 5.0 k = 0.0625 $\frac{\triangle t}{\triangle x^2} = 100.0$

# X Dom	Grid Points In X	# Steps	L^1 Error	L^∞ Error	$\triangle t$	Elapsed Time (msec.)
1	256	50000	5.79e-04	9.10e-04	3.63e-05	479510
2	128	17680	3.18e-05	5.00e-05	1.45e-04	85980
4	64	21210	3.18e-05	4.99e-05	5.80e-04	52890
8	32	1680	2.17e-05	4.81e-05	2.32e-03	2470
16	16	120	1.42e-05	3.42e-05	9.23e-03	110
32	8	20	3.98e-08	9.44e-08	3.62e-02	20

Figure 4: Graph of the time required on the single processor machine to solve the one-dimensional problem. $\frac{\triangle t}{\triangle x^2}$ constant

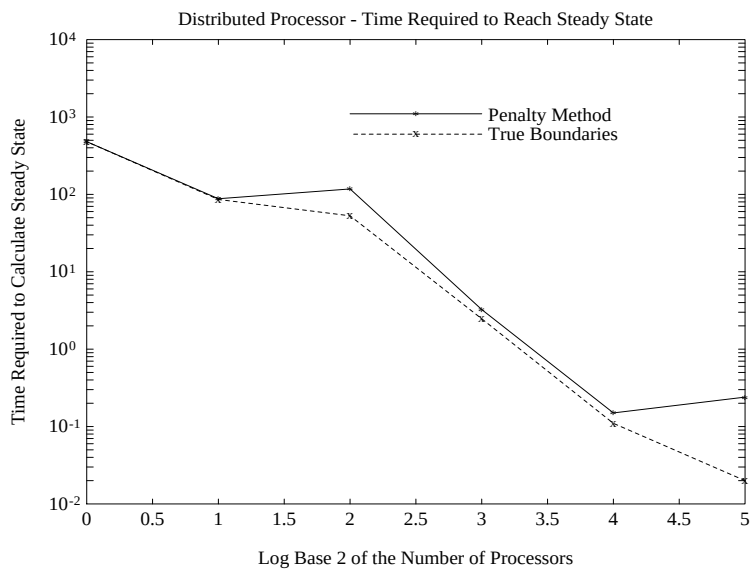


Figure (5) is a plot of the times from tables (5) through (7). Figure (6) is a plot of the times from tables (8) through (10). The rate of rate of decrease in time required in figure (5) is not as steep as the ideal. In figure (6), however, the rate of time savings is greater but as the numbers of subdomains goes beyond 16 the time required increases.

Table 5: Distributed processor test run. (1-D Case) $\triangle t$ constant, $\gamma = 0.2$

Calculated Boundaries
Frequency = 5.0 k = 0.0625 dt = 0.125

# X Domains	Points Per Domain	# Steps	L^1 Error	L^∞ Error	$\frac{\triangle t}{\triangle x^2}$	Elapsed Time (Sec.)
1	256	20	1.07e-05	1.67e-05	3.44e+05	26.99
2	128	20	1.06e-05	1.67e-05	8.61e+04	3.99
4	64	1820	2.68e-05	4.08e-05	2.15e+04	4.91
8	32	2510	1.78e-05	3.70e-05	5.39e+03	2.93
16	16	2020	2.63e-05	4.18e-05	1.35e+03	1.55
32	8	1320	1.12e-05	1.91e-05	3.45e+02	0.87

Table 6: Distributed processor test run. (1-D Case) $\triangle t$ constant, true boundaries

True Boundaries With Communication
Frequency = 5.0 k = 0.0625 dt = 0.125

# X Domains	Points Per Domain	# Steps	L^1 Error	L^∞ Error	$\frac{\triangle t}{\triangle x^2}$	Elapsed Time (Sec.)
1	256	20	1.07e-05	1.67e-05	3.44e+05	27.00
2	128	20	1.06e-05	1.67e-05	8.61e+04	3.99
4	64	70	1.78e-05	2.79e-05	2.15e+04	0.74
8	32	30	4.88e-06	1.08e-05	5.39e+03	0.13
16	16	10	6.96e-06	1.67e-05	1.35e+03	0.03
32	8	10	5.17e-10	1.27e-09	3.45e+02	0.01

Table 7: Distributed processor test run. (1-D Case) $\triangle t$ constant, no communication

True Boundaries Without Communication
Frequency = 5.0 k = 0.0625 dt = 0.125

# X Domains	Points Per Domain	# Steps	L^1 Error	L^∞ Error	$\frac{\triangle t}{\triangle x^2}$	Elapsed Time (Sec.)
1	256	20	1.07e-05	1.67e-05	3.44e+05	27.00
2	128	20	1.06e-05	1.67e-05	8.61e+04	3.97
4	64	70	1.78e-05	2.79e-05	2.15e+04	0.70
8	32	30	4.88e-06	1.08e-05	5.39e+03	0.11
16	16	10	6.96e-06	1.67e-05	1.35e+03	0.02
32	8	10	5.17e-10	1.27e-09	3.45e+02	0.01

Table 8: Distributed processor test run. (1-D Case) $\frac{\triangle t}{\triangle x^2}$ constant, $\gamma = 0.2$

Calculated Boundaries
Frequency = 5.0 k = 0.0625 $\frac{\triangle t}{\triangle x^2} = 100.0$

# X Domains	Points Per Domain	# Steps	L^1 Error	L^∞ Error	$\triangle t$	Elapsed Time (Sec.)
1	256	50000	5.79e-04	9.10e-04	3.63e-05	1203.13
2	128	17680	3.18e-05	5.00e-05	1.45e-04	126.40
4	64	46300	3.18e-05	5.00e-05	5.80e-04	111.81
8	32	2190	3.18e-05	4.99e-05	2.32e-03	2.68
16	16	170	5.42e-06	1.20e-05	9.23e-03	0.15
32	8	360	1.88e-05	4.65e-05	3.62e-02	0.25

Figure 5: Graph of the time required on the distributed processor machine to solve the one-dimensional problem. $\triangle t$ constant

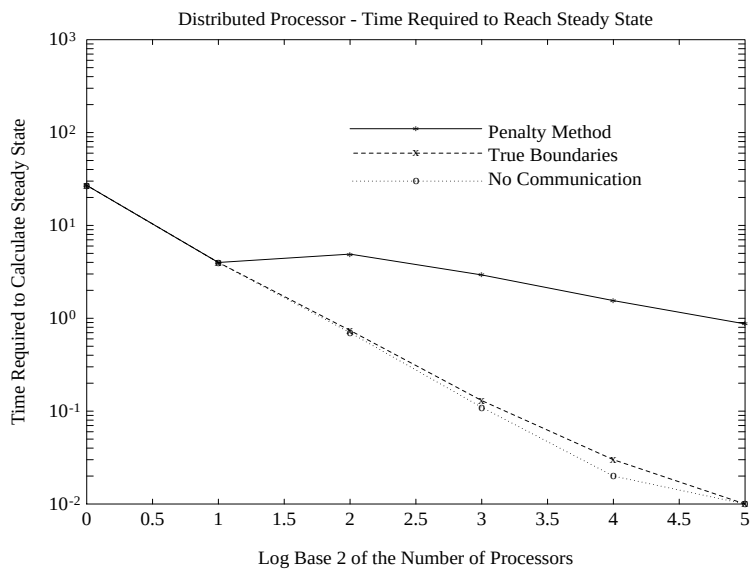


Table 9: Distributed processor test run. (1-D Case) $\frac{\triangle t}{\triangle x^2}$ constant, true boundaries

True Boundaries With Communication
Frequency = 5.0 k = 0.0625 $\frac{\triangle t}{\triangle x^2} = 100.0$

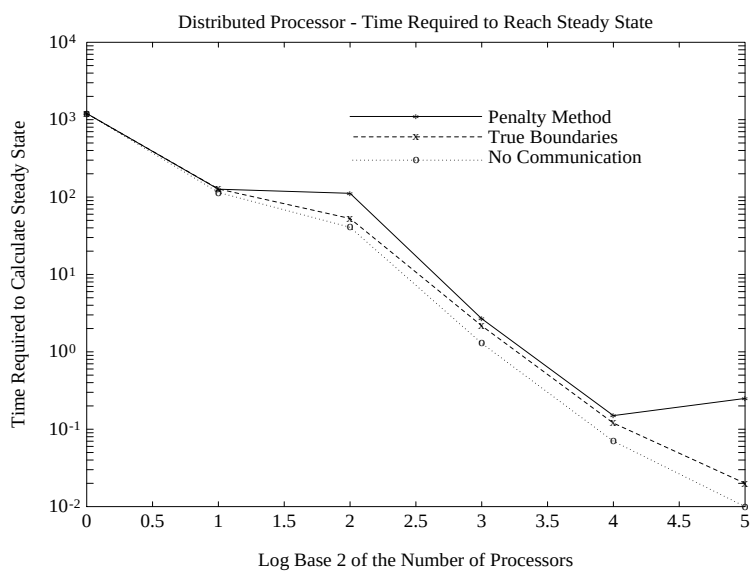
# X Domains	Points Per Domain	# Steps	L^1 Error	L^∞ Error	$\triangle t$	Elapsed Time (Sec.)
1	256	50000	5.79e-04	9.10e-04	3.63e-05	1200.52
2	128	17680	3.18e-05	5.00e-05	1.45e-04	127.36
4	64	21210	3.18e-05	4.99e-05	5.80e-04	52.87
8	32	1680	2.17e-05	4.81e-05	2.32e-03	2.17
16	16	120	1.42e-05	3.42e-05	9.23e-03	0.12
32	8	20	3.98e-08	9.44e-08	3.62e-02	0.02

Table 10: Distributed processor test run. (1-D Case) $\frac{\triangle t}{\triangle x^2}$ constant, no communication

True Boundaries Without Communication
Frequency = 5.0 k = 0.0625 $\frac{\triangle t}{\triangle x^2} = 100.0$

# X Domains	Points Per Domain	# Steps	L^1 Error	L^∞ Error	$\triangle t$	Elapsed Time (Sec.)
1	256	50000	5.79e-04	9.10e-04	3.63e-05	1200.52
2	128	17680	3.18e-05	5.00e-05	1.45e-04	114.96
4	64	21210	3.18e-05	4.99e-05	5.80e-04	40.72
8	32	1680	2.17e-05	4.81e-05	2.32e-03	1.29
16	16	120	1.42e-05	3.42e-05	9.23e-03	0.07
32	8	20	3.98e-08	9.44e-08	3.62e-02	0.01

Figure 6: Graph of the time required on the distributed processor machine to solve the one-dimensional problem. $\frac{\triangle t}{\triangle x^2}$ constant



(5.3) Single Processor in Two Dimensions Results are presented for the two-dimensional case. The values for the initial conditions are $\omega = 1$ and $k = \frac{1}{16}$. The cases presented are from runs on a single processor machine. Tables (11) and (12) are for trial runs from the single processor case and both use a constant time step. Table (11) is a trial in which the penalty method is employed while table (12) is a trial in which the true time dependent solution is enforced at the subdomain interface.

Figure (7) is a plot of the times from tables (11) and (12). Once the number of subdomains increases beyond four the amount of time required begins to grow. The time required to calculate the extra iterations grows faster than the time savings of calculating the smaller matrices. There were difficulties implementing the penalty method in two dimensions. To avoid instabilities the values at the subdomain interfaces were first calculated using the penalty method and the final value was calculated by averaging the new approximation and the two previous time levels.

Table 11: Single processor test run. (2-D Case) $\triangle t$ constant: $\gamma = 10.0$

Calculated Boundarys
Frequency = 1.0 k = 0.0625 dt = 0.0125

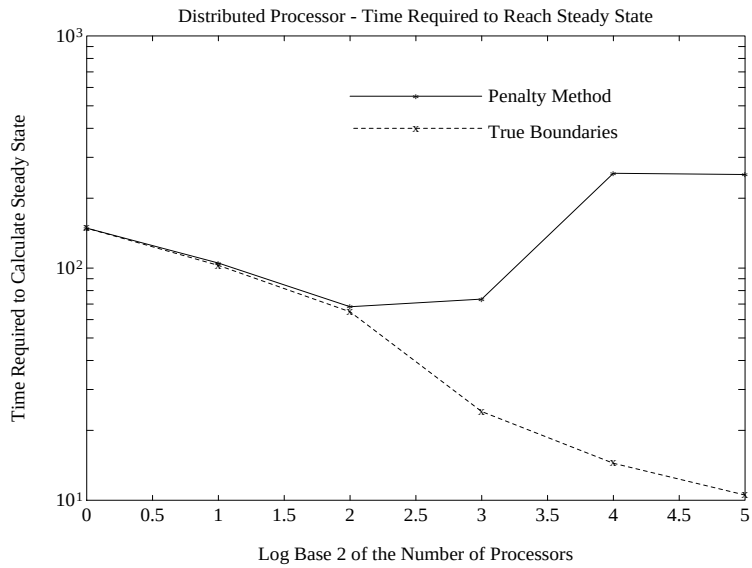
# X Dom	# Y Dom	Grid Points In X	Grid Points In Y	# Steps	L^1 Error	L^∞ Error	$\triangle t\left(\frac{1}{\triangle x^2}+\frac{1}{\triangle y^2}\right)$	Elapsed Time (msec.)
1	1	64	64	1920	2.00e-04	4.92e-04	2.69e+02	148630
1	2	64	32	1920	2.00e-04	4.93e-04	1.68e+02	105140
2	2	32	32	1920	2.00e-04	4.94e-04	6.74e+01	68170
2	4	32	16	2460	1.76e-04	4.90e-04	4.22e+01	73530
4	4	16	16	10000	7.48e-05	9.44e-04	1.69e+01	256060
4	8	16	8	10000	7.70e-04	4.28e-03	1.06e+01	252980

Table 12: Single processor test run. (2-D Case) $\triangle t$ constant, true boundaries

True Boundaries
Frequency = 1.0 k = 0.0625 dt = 0.0125

# X Dom	# Y Dom	Grid Points In X	Grid Points In Y	# Steps	L^1 Error	L^∞ Error	$\triangle t\left(\frac{1}{\triangle x^2}+\frac{1}{\triangle y^2}\right)$	Elapsed Time (msec.)
1	1	64	64	1920	2.00e-04	4.92e-04	2.69e+02	148600
1	2	64	32	1920	2.00e-04	4.93e-04	1.68e+02	102730
2	2	32	32	1920	2.00e-04	4.93e-04	6.74e+01	64830
2	4	32	16	800	1.92e-04	4.73e-04	4.22e+01	24100
4	4	16	16	520	1.97e-04	4.85e-04	1.69e+01	14430
4	8	16	8	340	1.68e-04	4.42e-04	1.06e+01	10530

Figure 7: Graph of the time required on the single processor machine to solve the two-dimensional problem. $\triangle t$ constant



(5.4) Distributed Machine in Two Dimensions Results are presented for the two-dimensional case. The values for the initial conditions are $\omega = 1$ and $k = \frac{1}{16}$. The cases presented are from runs on a distributed machine. Tables (13) through (15) are for trial runs on the distributed machine. Table (13) is a trial in which the penalty method is emplyed, table (14) is a trial in which the true time dependent solution is enforced at the subdomain interface but the interprocessor communication is retained, and table (15) is a trial using the true solution at the subdomain interfaces with no interprocessor communication.

Figure (8) is a plot of the times from tables (13) through (15). Here the amount of time required decreases until the number of subdomains becomes greater than eight. As the number of subdomains increases to four the rate of decrease in the time required matches that of the ideal case.

(6) Acknowledgments Special thanks go to David Gottlieb of Brown University who initiated this work and thanks for his guidance and support. Also, the computing system used is the iPSC/2 860 at ICASE, NASA Langley. Thanks go to the staff at ICASE for allowing the use of the machine and for their guidance and support.

Table 13: Distributed processor test run. (2-D Case) $\triangle t$ constant: $\gamma = 10.0$

Calculated Boundarys
Frequency = 1.0 k = 0.0625 dt = 0.0125

# X Dom	# Y Dom	Grid Points In X	Grid Points In Y	# Steps	L^1 Error	L^∞ Error	$\triangle t \left(\frac{1}{\triangle x^2} + \frac{1}{\triangle y^2}\right)$	Elapsed Time (sec .)
1	1	64	64	2000	1.45e-04	3.59e-04	2.69e+02	454.62
1	2	64	32	2000	1.46e-04	3.62e-04	1.68e+02	193.52
2	2	32	32	2000	1.46e-04	3.62e-04	6.74e+01	86.35
2	4	32	16	2600	1.22e-04	4.19e-04	4.22e+01	53.61
4	4	16	16	11800	5.95e-05	4.73e-04	1.69e+01	127.43
4	8	16	8	12000	8.34e-04	3.48e-03	1.06e+01	75.91

Table 14: Distributed processor test run. (2-D Case) $\triangle t$ constant, true boundaries

True Boundaries with Communication
Frequency = 1.0 k = 0.0625 dt = 0.0125

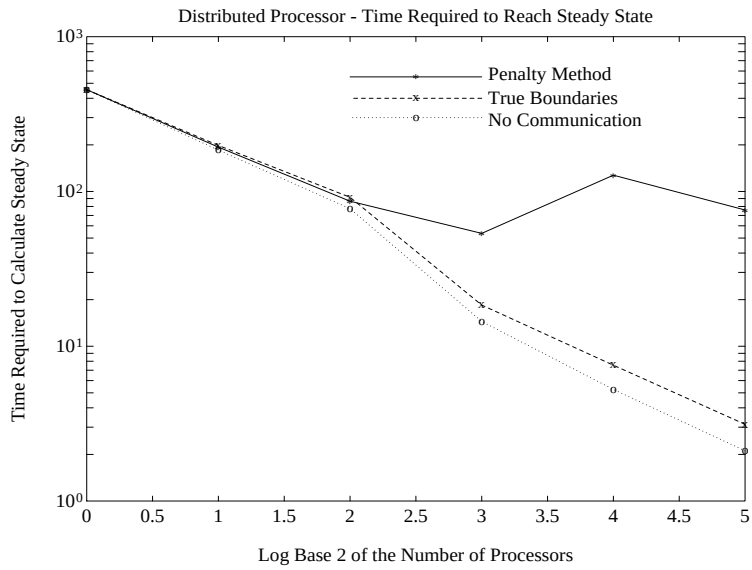
# X Dom	# Y Dom	Grid Points In X	Grid Points In Y	# Steps	L^1 Error	L^∞ Error	$\triangle t \left(\frac{1}{\triangle x^2} + \frac{1}{\triangle y^2}\right)$	Elapsed Time (sec .)
1	1	64	64	2000	1.45e-04	3.59e-04	2.69e+02	454.62
1	2	64	32	2000	1.46e-04	3.59e-04	1.68e+02	198.36
2	2	32	32	2000	1.46e-04	3.60e-04	6.74e+01	91.70
2	4	32	16	800	1.93e-04	4.76e-04	4.22e+01	18.45
4	4	16	16	600	5.79e-05	1.43e-04	1.69e+01	7.54
4	8	16	8	400	3.86e-05	1.01e-04	1.06e+01	3.13

Table 15: Distributed processor test run. (2-D Case) $\triangle t$ constant, no communication

True Boundaries with no communication
Frequency = 1.0 k = 0.0625 dt = 0.0125

# X Dom	# Y Dom	Grid Points In X	Grid Points In Y	# Steps	L^1 Error	L^∞ Error	$\triangle t \left(\frac{1}{\triangle x^2} + \frac{1}{\triangle y^2}\right)$	Elapsed Time (sec .)
1	1	64	64	2000	1.45e-04	3.59e-04	2.69e+02	454.53
1	2	64	32	2000	1.46e-04	3.59e-04	1.68e+02	185.05
2	2	32	32	2000	1.46e-04	3.60e-04	6.74e+01	77.42
2	4	32	16	800	1.93e-04	4.76e-04	4.22e+01	14.46
4	4	16	16	600	5.79e-05	1.43e-04	1.69e+01	5.26
4	8	16	8	400	3.86e-05	1.01e-04	1.06e+01	2.11

Figure 8: Graph of the time required on the distributed processor machine to solve the two-dimensional problem. $\triangle t$ constant



References

[1] Canuto, C., Hussaini, M.Y., Quarteroni, A., and Zang, T.A. *Spectral Methods in Fluid Dynamics*. Springer-Verlag, New York, 1988.

[2] Davis, Philip J. and Rabinowitz, Philip. *Methods of Numerical Integration*. Academic Press, inc. (Harcourt Brace Jovanovich), New York, second edition, 1984.

[3] Funaro, Daniele. Domains Decomposition Methods for Pseudo Spectral Approximations. Technical report, Univeritá degli studi di Pavia, Strada Nuova 65, 27100 Pavia, Italy. Research supported in part by AFOSR Grant 85-0303.

[4] Funaro, Daniele. A Multidomain Spectral Approximation of Elliptic Equations. *Numerical Methods for P.D.E.*, 2:187–205, 1986.

[5] Gottlieb, David and Gustafsson, Bertil. Generalized Dufort-Frankel Methods for Parabolic Initial-Boundary-Value Problems. ICASE Report no. 75-5, ICASE, NASA Langley Research Center, February 1975.

[6] Gottlieb, David and Lustman, Liviu. The Dufort-Frankel Chebyshev Method for Parabolic Initial Boundary Value Problems. ICASE Report No. 81-42, ICASE, NASA Langley Research Center, December 1981.

[7] Gottlieb, David and Hirsh, Richard S. Parallel Pseudospectral Domain Decomposition Techniques. *Journal of Scientific Computing*, 4(4):309–325, December 1989.

[8] Gottlieb, David and Lustman, Liviu. The Spectrum of the Chebyshev Collocation Operator for the Heat Equation. *Siam Journal of Numerical Analysis*, 20(5):909–921, October 1983.

[9] Marchuk, G.I. *Methods of Numerical Mathematics*. Springer-Verlag, New York, 1975.

[10] Quarteroni, Alfio. Domain Decomposition Methods for Systems of Conservation Laws: Spectral Collocation Approximations. ICASE Report no. 89-5, ICASE, NASA Langley Research Center, January 1989.

[11] Vandeven, Hervé. On the Eigenvalues of Second-Order Spectral Differential Operators. In Canuto, Claudio, and Quarteroni, Alfio, editor, *Spectral and High Order Methods for Partial Differential Equations*, pages 313–318. North-Holland, June 1989. Proceedings of the ICOSAHOM 89 Conference.