IN-64-CR

280676

P-24

ON THE PARALLEL SOLUTION OF PARABOLIC EQUATIONS

E. GALLOPOULOS
Y. SAAD

May, 1989

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 89.19

NASA Cooperative Agreement Number NCC 2-387

RIACS

Research Institute for Advanced Computer Science

ON THE PARALLEL SOLUTION OF PARABOLIC EQUATIONS

E. GALLOPOULOS
Y. SAAD

May, 1989

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 89.19

NASA Cooperative Agreement Number NCC 2-387

ON THE PARALLEL SOLUTION OF PARABOLIC EQUATIONS

E. GALLOPOULOS
Center for Supercomputing Research and Development
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801
stratis@uicsrd.csrd.uiuc.edu

Y. SAAD
RIACS, Mail Stop 230-5
NASA Ames Research Center
Moffet Field, California 94035
saad@riacs.edu

ABSTRACT

We propose new parallel algorithms for the solution of linear parabolic problems. The first of these methods is based on using polynomial approximation to the exponential. It does not require solving any linear systems and is highly parallelizable. The two other methods proposed are based on Padé and Chebyshev approximations to the matrix exponential. The parallelization of these methods is achieved by using partial fraction decomposition techniques to solve the resulting systems and thus offers the potential for increased time parallelism in time dependent problems. We also present experimental results from the Alliant FX/8 and the Cray Y-MP/832 vector multiprocessors.

1. Introduction. We consider the following linear parabolic partial differential equation:

$$
\begin{aligned}
\frac{\partial u(x,t)}{\partial t} &= Lu(x,t) + s(x), \quad x \in \Omega \\
u(0,x) &= u_0, \quad \forall x \in \Omega \\
u(t,x) &= \sigma(x), \; x \in \partial\Omega, t \geq 0.
\end{aligned} \tag{1.1}
$$

where L is a second order partial differential operator of elliptic type, acting on functions defined on the open bounded set Ω. If the method of lines is used to solve (1.1), then this partial differential equation is first discretized with respect to space variables, resulting in a system of ordinary differential equations of the form

$$
\begin{aligned}
\frac{dw(t)}{dt} &= -Aw(t) + r \\
w(0) &= w_0
\end{aligned}
$$

whose solution is explicitly given by

$$
w(t) = A^{-1}r + e^{-tA}(w_0 - A^{-1}r) \tag{1.2}
$$

which simplifies to

$$
w(t) = e^{-tA}w_0 \tag{1.3}
$$

in the case of a homogeneous system ($r = 0$). Note that if we denote by $\hat{w}(t) \equiv w(t) - A^{-1}r$ and accordingly, $\hat{w}_0 \equiv w_0 - A^{-1}r$, then $\hat{w}(t)$ satisfies a homogeneous system and

$$
\hat{w}(t) = e^{-tA}\hat{w}_0 \tag{1.4}
$$

Thus, if we want to obtain the solution at time t in one single step, we would need to operate with the matrices A^{-1} and e^{-tA} on certain initial vectors. This solution faces the following difficulties:

- Computing $e^{-tA}\hat{w}_0$ may not be easy, especially for large t.
- The cost of computing $A^{-1}r$ is not negligible for more than one space dimensions.
- In many problems the operator L, as well as the forcing term s, may vary with t; If this variation is rapid the above formula is not applicable or may be very inaccurate for large t;

Note that if we denote by f the function $f(z) = (1 - e^{-z})/z$ we can rewrite the solution (1.2) as follows,

$$\begin{aligned} w(t) &= e^{-tA}w_0 + f(tA)tr \\ &= w_0 + tf(tA)[r - Aw_0] \end{aligned} \tag{1.5}$$

This removes the term $A^{-1}r$ from the expression (1.2), at the expense of dealing with the function $f(z)$ instead of e^{-z} [22]. The above expression also shows more clearly the dependence of the solution with respect to the initial condition w_0 and the forcing function r.

Assume now that instead of attempting to compute the solution at time t in one single step, we use a time-stepping procedure. At time $t + \Delta t$, the solution will depend on $w(t)$ which plays the role of w_0 in the above expressions and we get

$$\hat{w}(t + \Delta t) = e^{-\Delta t A}\hat{w}(t). \tag{1.6}$$

from (1.4), or

$$\begin{aligned} w(t + \Delta t) &= e^{-\Delta t A}w(t) + \Delta t f(\Delta t A)r \\ &= w(t) + \Delta t f(\Delta t A)[r - Aw(t)]. \end{aligned} \tag{1.7}$$

from (1.5).

We should observe that the use of the variable $\hat{w}$ in formula (1.6) requires computing $A^{-1}r$ only once and not at every step of the stepping procedure. The advantage of using (1.7) over (1.6) is therefore limited, except when A and varies with time.

In both (1.7) and (1.6), we need to compute a vector of the form $q(\Delta t A)v$, where $q(z)$ is a known analytic function in z. The basic idea for computing (1.6) and (1.7), is to find a suitable approximation $g(A)$, to the function $q(A)$ and then substitute this approximation in (1.6) or (1.7). This is complicated by the following facts. First, depending on the operator L, the type of discretization performed and the boundary conditions, A may be symmetric positive definite, or nonsymmetric. It may also be singular or nearly singular. Moreover, in typical problems A is large and sparse making the direct calculation of $q(\Delta t A)$ by usual methods prohibitive.

Note that there is no need to actually evaluate the matrix $g(\Delta t A)$. Instead all we need is be able to evaluate $g(\Delta t A)v$ inexpensively for some vector v. In this paper we show how to do this for the specific case $q(z) = e^{-z}$ which allows to solve the general problem via (1.6). For notational convenience, we will assume that after suitable scaling $\Delta t = 1$.

There are two different ways of generating approximations $g(A)v$. The first is by using polynomials. The resulting procedure will only require matrix by vector multiplications with the matrix A, and is therefore very easily

parallelizable. In Section 3, we will consider an approach of this type based on using a Krylov subspace technique.

The second class of methods consists of taking g to be a rational function of the form $r(z) = p(z)/q(z)$. In this situation, a difficulty arises when computing $r(A)v$ on parallel machines. Typically the denominator is factored as $q(z) = \prod_{i=1}^{r}(z - \lambda_i)$ and $q(A)^{-1}v$ is computed by solving the sequence of linear systems $(A - \lambda_i I)u_i = u_{i-1}$ with $u_0 = v$. This is a sequential process which can be particularly damaging, especially for one-dimensional problems where A is usually a tridiagonal matrix. Although there are many efficient methods for solving tridiagonal systems, it is clear that if we have to solve only one single system per step, we will very likely be under-utilizing the computational resources. To circumvent this sequential constraint we propose in Section 4 to use the partial fraction expansions of $r(z)$. This will transform the sequential solution of $(A - \lambda_i I)u_i = u_{i-1}$ into solving in parallel the independent linear systems $(A - \lambda_i I)u_i = v$ and then taking a linear combination of the results u_i. The advantages for one-dimensional problems are clear. For higher-dimensional problems, this allows to remove the need to parallelize each of the linear systems $(A - \lambda_i I)u_i = u_{i-1}$. In effect it offers a means for achieving parallelism in an extremely simple manner, far simpler than would be needed in optimizing the linear solves in the traditional approach. This is achieved by using high order schemes, i.e., high degree rational approximations. As a result an added benefit is that the overall amount of work is also reduced. As is stated in our conclusion, it seems that high order integration schemes in ODE methods offer a tremendous potential in a parallel processing environment.

2. Previous work. The previous discussion underscores the direct connection that exists between the topic of this paper and that of of parallel solution of ordinary differential equations (ODEs). As argued in [8], the most important situation when considering parallel methods for solving systems of ODE's is when the problem is very large, as is the case for systems resulting from a Method of Lines semi-discretization of a partial differential equation such as (1.1). We refer the reader to [13] for methods to approximate the matrix exponential, to [18,21] for polynomial approximations in parabolic problems, and to the work of Varga and co-authors for rational approximations ([22,3,2,12]). In [7,20] a method was introduced for the parallelization of Block Cyclic Reduction (BCR). The connection between the method considered in these papers and the question addressed here, is that when using BCR one must evaluate a vector of the form $q(A)v$, where q is a rational function. Partial fractions in a sequential context for time dependent problems were used in [12,19,26] and suggested in parallel complexity studies in [11]. Finally recent experiments of Reusch et al. ([17]) demonstrated re-

markable gains in efficiency and accuracy for the solution of homogeneous linear evolution equations by means of high-order diagonal Padé approximations. As will be argued in Section 4 such schemes are extremely attractive on parallel machines, when properly implemented.

3. Polynomial approximations. In this section we consider using polynomial approximation to the exponential, i.e., we seek an approximation to $e^{-A}v$ of the form

$$e^{-A}v \approx p_{m-1}(A)v \tag{3.1}$$

where p_{m-1} is a polynomial of degree $m-1$. The main attraction of polynomial based techniques is their explicit nature, that is the fact that they do not require solving linear systems. In fact the only operations required with the matrix A are multiplications with vectors, an operation that is very easy to parallelize and vectorize. On the other hand polynomial approximation cannot handle very stiff problems as well as rational approximations. As a result the trade-off is a large number of matrix by vector multiplications versus no linear systems to solve. For two-dimensional and, more importantly, for three-dimensional problems polynomial based schemes, if implemented with care can be very attractive.

There are several ways in which polynomial approximations can be found. The simplest technique is to attempt to minimize some norm of the error $e^{-z} - p_{m-1}(z)$ on a continuum in the complex plane that encloses the spectrum of A. For example, Chebyshev approximation can be used. The disadvantage of this is that it requires some approximation to the spectrum of A. In this paper we consider only approaches that do not require any information on the spectrum of A.

The approximation (3.1) to $e^{-A}v$ is an element of the Krylov subspace

$$K_m = span\{v, Av, \ldots, A^{m-1}v\}.$$

In order to manipulate vectors in K_m it is convenient to generate an orthonormal basis $V_m = [v_1, v_2, v_3, \ldots, v_m]$. We will take as initial vector $v_1 = v/\|v\|_2$ and generate the basis V_m with the well-known Arnoldi algorithm, described below.

Algorithm: Arnoldi

1. *Initialize:*
   Compute $v_1 := v/\|v\|_2$.
2. *Iterate:* Do $j = 1, 2, ..., m$
   1. Compute $w := Av_j$
   2. Compute a set of j coefficients h_{ij} so that

$$w := w - \sum_{i=1}^{j} h_{ij} v_i \tag{3.2}$$

   is orthogonal to all previous v_i's.
   3. Compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$.

By construction the above algorithm produces an orthonormal basis $V_m = [v_1, v_2, \ldots, v_m]$, of the Krylov subspace K_m. If we denote the $m \times m$ upper Hessenberg matrix consisting of the coefficients h_{ij} computed by the algorithm by H_m we have the relation

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T \tag{3.3}$$

from which we get $H_m = V_m^T A V_m$. Therefore H_m represents the projection of the linear transformation A to the subspace K_m, with respect to the basis V_m.

We can write the desired solution $x = p_{m-1}(A)v$ as $x = V_m y$ where y, is an m-vector. Ideally we would like to minimize the norm $\|e^{-A}v - V_m y\|_2$. The solution to this optimization problem is known to be

$$y_{opt} = V_m^T e^{-A} v. \tag{3.4}$$

Unfortunately, this is not computable because it involves the unknown vector $e^{-A}v$. However, if we assume that $v_1 = \beta v$ then we have $y_{opt} = \beta V_m^T e^{-A} V_m e_1$ and it is natural to approximate $V_m^T e^{-A} V_m$ by e^{-H_m}, leading to the approximation,

$$e^{-A} v \approx \beta V_m e^{-H_m} e_1 \tag{3.5}$$

This immediately raises a question concerning the quality of this approximation. A first observation is that the above approximation is exact for $m = n$. This is because in this situation $v_{m+1} = 0$ and (3.3) becomes $AV_m = V_m H_m$, where V_m is an $n \times n$ orthogonal matrix. In fact, similarly to the conjugate gradient method and the Arnoldi process, the approximation will be exact for m whenever m is larger or equal to the degree of the minimal polynomial of v_1 with respect to A. This however, is unlikely to take

place before $m = n$. More generally, the following theorem provides a rough bound on the error and establishes convergence when m increases.

THEOREM 3.1. *Let A be any matrix and let $\rho = \|A\|_2$. Then the error of the approximation (3.5) is such that*

$$\|e^{-A}v - \beta V_m e^{-H_m} e_1\|_2 \leq 2\beta \frac{\rho^m e^\rho}{m!}. \tag{3.6}$$

The proof of this result is omitted. This result as well as sharper bounds will be fully discussed in a forthcoming paper.

The theorem shows convergence of the approximation (3.5). It can also serve as a guide to choosing the step size in a time-stepping procedure. Indeed, if we were to replace A by the scaled matrix τA, then the Krylov subspace will remain the same, i.e., V_m will not change, and H_m will be scaled to τH_m. As a result the bound (3.6) becomes,

$$\|e^{-\tau A}v - \beta V_m e^{-\tau H_m} e_1\| \leq 2\beta \frac{(\tau\rho)^m e^{\tau\rho}}{m!} \tag{3.7}$$

The consequence of (3.7) is that by reducing the step-size one can always make the scheme accurate enough, without changing the dimension m.

We note that the idea of exploiting the Lanczos algorithm to evaluate terms of the exponential of Hamiltonian operators has been extensively used in Chemistry ([25]). The work in [15] is also related wherein systems of ODEs are solved by first projecting into Krylov subspaces and then solving reduced tridiagonal systems of ODEs.

So far we have not considered the important particular case where A is symmetric. As is well-known in this situation Arnoldi's algorithm simplifies into the Lanczos process, which entails a three-term recurrence. This is a result of the fact that the matrix $H_m = V_m^T A V_m$ must be symmetric and therefore tridiagonal symmetric, and so all $h_{i,j} = 0$ for $i = 1, 2, .., j-2$. However, the resulting vectors which are in theory orthogonal to each other, tend to loose their orthogonality rapidly.

From the practical point of view several problems must be addressed. For example we can mention the following issues:

1. How should one compute the vector $e^{H_m} e_1$?
2. In the case where A is symmetric, should orthogonality be enforced?

We note that for (1) we can use the methods described in the next sections efficiently since H_m is either tridiagonal or Hessenberg. If m is small enough as is the case in practical situations, then the cost of computing $e^{H_m} e_1$ will be negligible. An important observation here is that since V_m is orthogonal, the integration scheme based on the formula (3.5) is likely to inherit the stability properties of the scheme used in approximating $e^{-H_m} e_1$. For this reason it is crucial to use rational approximation.

For (2), if the matrix is nonsymmetric it is recommended to perform a modified Gram-Schmidt process with partial reorthogonalization. Selective or partial reorthogonalization can be used when A is symmetric [16].

4. Rational approximations.

4.1. Overview. As was mentioned earlier, a popular way of computing approximations to $e^{-A}v$ is via a rational approximation to the exponential, i.e.,

$$e^{-A}v \approx q_r(A)^{-1}p_m(A)v \tag{4.1}$$

The simplest approximation of this type, referred to as Padé approximation can be found by matching the Taylor series expansion of the left-hand-side and right-hand-side of (4.1) at the origin. This approximation is local, i.e., it is very accurate near the origin but may be inaccurate far away. For this reason schemes based on more global approximation have been devised [22,2]. Thus, for typical parabolic problems that involve a second order partial differential equation L that is self-adjoint elliptic, the eigenvalues of L are located in the interval $(-\infty, 0)$ and it is therefore natural to seek the best rational approximation to the function e^z on this interval, or equivalently to the function e^{-z} on the interval $(0, +\infty)$.

One of the main reasons why rational approximations have been preferred to polynomial approximations, is the better stability properties of the corresponding integration schemes. Thus, it is known that the Padé approximations to the matrix exponential give rise to unconditionally stable methods if and only if the degree m of the numerator is at most equal to the degree of the denominator ([22]).

By far the best known rational approximation to the exponential is the $(1,1)$ Padé approximation $e^z \approx (1+\frac{1}{2}z)/(1-\frac{1}{2}z)$ which when used in conjunction with (1.6) leads to the well-known Crank-Nicolson scheme. However, because of its modest accuracy, there are limitations as to how large the step size Δt can be and there might be, some large number, say, m_1, applications of formula (1.6) before the solution at the final time T is found. At the other extreme assume that one can find a highly accurate rational approximation that allows to compute $w(T)$ in just one application of (1.6). If the rational approximation is of the type (m_2, m_2) then it is very likely that $m_2 \ll m_1$, meaning that the total amount of work is far less with the more accurate scheme. Thus, the more accurate schemes have tremendous potential in the context of parallel processing precisely because of this feature. By their very nature, low order schemes do not allow for much work to be shared at every step while, as will be seen in the next section, high order schemes are easily and safely parallelizable.

4.2. The use of partial fraction expansions. Let the rational approximation to the exponential of e^{-z} be of the form

$$R_{m,r}(z) = \frac{p_m(z)}{q_r(z)} \tag{4.2}$$

where we assume that $m \leq r$. Then, at each application of the scheme corresponding to (1.6) we would have to evaluate the vector

$$x = p_m(A).q_r(A)^{-1}v \tag{4.3}$$

There are several ways in which one can compute $q_r(A)^{-1}v$. One economical procedure involves factoring the polynomial $q_r(z)$ as

$$q_r(z) = \prod_{i=1}^{r}(z - \lambda_i) \tag{4.4}$$

and then solving the successive linear systems

$$(A - \lambda_i I)u_i = u_{i-1}, i = 1, 2, ..., r \tag{4.5}$$

with $u_0 = v$. The final result is the desired solution. One then needs to multiply the result by $p_m(A)$. In fact, several modifications to the aforementioned scheme have been proposed in the literature to avoid this last extra step [4]. Incidentally, we should mention that partial fraction expansion formulations can be used to explain many of these efficient implementations of time stepping procedures. This is discussed in the note [6].

Clearly, a significant difficulty with the use of (4.5) is that it is a sequential process. System number i must be solved before system $i + 1$ since its solution will be the right hand side of the next system.

An alternative approach used in [7] in a different context is to resort to the partial fraction expansion of (4.2), namely,

$$R_{m,r}(z) = \frac{\pi_m}{\kappa_r} + \sum_{i=1}^{r}\frac{\alpha_i}{z - \lambda_i} \tag{4.6}$$

where

$$\alpha_i = \frac{p_m(\lambda_i)}{q'_r(\lambda_i)} \tag{4.7}$$

and π_m and κ_r are the leading coefficients of the polynomials p_m and q_r respectively.

With this expansion the algorithm for computing (4.3) becomes:

Algorithm:

1. For $i = 1, 2, \ldots, r$ solve $(A - \lambda_i I)x_i = v$ in parallel.
2. Compute $x = \frac{\pi_m}{\kappa_r} v + \sum_{i=1}^{r} \alpha_i x_i$.

Note that for the usual approach, schemes with repeated poles (e.g. the *restricted* schemes [14]) have often been preferred because they involve fewer factorizations with the standard techniques when direct methods are used. These factorizations are very expensive for 2-D and 3-D problems. These schemes cannot be used with our approach since we need to have distinct poles.

As we can see, the method is very well suited for systems offering *hierarchical parallelism* realized with multicluster architectures ([10]). For problems in two and three space dimensions, we could use a rational approximation generating as many independent systems as there are clusters. The solution of each system could then proceed independently in each cluster. We thus see the interesting phenomenon that not only numerical considerations but also the amount of available parallelism will drive the choice of the order of the approximation.

4.3. Padé Approximation. In this section we outline the procedure using Padé approximation to the exponential. Given the degrees of the numerator and denominator, it is easy to automatically generate the rational function as a pair of two polynomials both given in power form. More precisely, the coefficients $\pi_j, j = 0, ..., m$ of the polynomial p_m and $\kappa_j, j = 0, ..., r$ of the polynomial q_r are explicitly given by [22]:

$$\begin{aligned} \pi_j &= (-1)^j \frac{(r+m-j)!m!}{(r+m)!j!(m-j)!}, j = 0, \ldots m \\ \kappa_j &= \frac{(m+r-j)!r!}{(m+r)!j!(r-j)!}, j = 0, \ldots r. \end{aligned} \tag{4.8}$$

Then we need to compute the roots of the denominator. This we do by some standard polynomial rootfinder. Once the roots are computed one then needs to compute the coefficients α_i of the partial fraction expansion (4.6) using formula (4.7). For high degree polynomials, numerical difficulties both in evaluating accurately the roots and in computing α_i by formula (4.7) are to be expected.

4.4. Chebyshev Approximation. When using Chebyshev approximation, one must first decide on which region the best rational function must be computed. In this paper we only consider the best uniform approximation to e^{-z} for $z \in (0, \infty)$. Unlike the Padé case, the coefficients of the numerator and denominator polynomials are not available analytically and must be computed as the solution of an optimization problem. This can lead

to a fairly involved procedure, requiring the use of a Remez type algorithm. We preferred instead to use directly the very accurate values from [1], where the polynomial coefficients are provided for up to the degree 30. Once these coefficients are input, we proceed as before, calling a polynomial rootfinder and evaluating the partial fraction coefficients.

The big advantage of Chebyshev methods, stressed in the work by Varga, is the ability to use large step size. In fact, when A is Hermitian, a relation of the form

$$\|w(t) - w_{m,r}\|_2 \leq \Lambda_{m,r} \|w_0 - A^{-1}r\|_2 \; \forall t \geq 0$$

holds, where $w_{m,r}$ is the solution computed with an (m,r) order Chebyshev approximation, and $\Lambda_{m,r}$ are constants converging to zero *geometrically* (see [3] and [1] for a list of $\Lambda_{r,r}$'s).

Although such a technique will have its limitations for time-varying coefficients and boundary conditions, it can often produce excellent results as is demonstrated in our experiments. Thus by combining the large time step, together with the problem decoupling for parallelism, we obtain a very efficient procedure in the sense that fast convergence, low error and efficient exploitation of the parallel resources are achieved.

4.5. Handling complex poles. The partial fractions in the decomposition could involve complex shifts of the operator A. These complex shifts come in conjugate pairs and correspond to complex poles of the rational function under consideration, that is the roots of the (real) denominator. Similar problems occur elsewhere in linear algebra, e.g. in the course of the QR algorithm [24, Section 41]. Since the coefficients of the corresponding principal parts in the partial fraction expansion also have conjugate coefficients, complex arithmetic can be avoided completely by writing the rational function as a sum of fractions whose denominators can contain quadratic factors. Each quadratic factor corresponds to a product of the form $(x - \lambda_i)(x - \bar{\lambda}_i)$ for all roots λ of the denominator having non-zero imaginary parts. This is just a case of incomplete partial fraction decomposition (see [9, Section 7.1]).

Some drawbacks to this technique are the need to form A^2, the extra computations due to the first order expansion coefficients and the need to store the quadratic factors.

In case A is banded, however, the formation of A^2 means a doubling of the bandwidth, and will result in an increase of data locality when working with A. If the corresponding matrix operations are designed carefully (e.g. using blocking as is done in BLAS3) increased efficiency will result on architectures with hierarchical memories. A numerical drawback is due to the squaring of the (possibly already large) condition of the matrix factors with the ensuing drawbacks in the application of iterative methods.

A simpler way of dealing with complex poles is to observe that the expansion coefficients α_i associated with two complex conjugate pairs must be conjugate. Then we can write

$$\alpha_i(A-\lambda_i I)^{-1}x + \bar{\alpha}_i(A-\bar{\lambda}_i I)^{-1}x = 2\Re[\alpha_i(A-\lambda_i I)^{-1}x] \tag{4.9}$$

This requires solving one complex system as opposed to two. It has the advantage of requiring less storage and fewer arithmetic operations than with the squaring approach. Moreover, data locality is also preserved through the use of complex arithmetic.

The above discussion addresses only the use of direct solvers. For 2-D and, more importantly, for 3-D problems, iterative procedures become attractive and we would like next to discuss how complex poles can be handled in this case. The first observation to be made is that we can again exploit the fact that the poles usually come in conjugate pairs. Thus, we can use the conjugate gradient technique to solve a system of the form $(A-\lambda I)(A-\bar{\lambda}I)x = f$, which will involve no complex arithmetic in the CG iteration. Indeed, the only operations that are needed with this matrix are matrix by vector multiplications of the form $w = (A-\lambda I)(A-\bar{\lambda}I)v$ which can be performed inexpensively in real arithmetic when v is real as $w = |\lambda|^2 v + A(A-2\Re(\lambda)I)v$. Moreover, the storage requirement is also not affected since only A is needed. Note that this is not equivalent to the normal equations approach. Preconditioning can also be easily retrofitted in this scheme. Indeed, the ICC(0) preconditioners require only an extra diagonal of data. This extra diagonal is complex and can be easily constructed. Using extra fill-in is, however, troublesome since all of L and U matrices must be treated as complex. Once the preconditioning $M = LU$ has been built then the CG iteration can be performed with the matrix $M^{-1}(A-\lambda I)\bar{M}^{-1}(A-\bar{\lambda}I)$ in real arithmetic. We should note that the scheme described here represents the simplest, certainly not the best, of a number of possible options. In particular, there are methods which will not be described here, that do not involve the matrix $|\lambda|^2 v + A(A-2\Re(\lambda)I)$ but the original matrix A. We also mention that the problem of solving complex linear systems of the form $(A-\lambda I)x = f$ has been addressed by Freund who devised special iteration schemes [5].

5. Numerical experiments. In this section we will describe a few tests to illustrate the behavior of the schemes described in this paper. In particular we do not compare here the polynomial approach with the rational approximation approach. Further experiments will be presented in a forthcoming report. Except where noted, our tests were conducted on an Alliant FX/8 vector multiprocessor using 64 bit floating-point arithmetic.

The test problem is issued from the discretization of

$$u_t = u_{xx},\ x \in (0,1)$$

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

using 22 grid points, yielding a matrix of size n=20. The initial conditions are chosen once the matrix is discretized, in such a way that the solution is known for all t. More precisely,

$$u(0,x_j) = \sum_{k=1}^{n} \frac{1}{k} \sin \frac{jk\pi}{n+1} \tag{5.1}$$

Note that the vector $\{\sin \frac{jk\pi}{n+1}\}_{j=1,..,n}$ is an eigenvector of the discretized operator. In order to decouple from the influence of errors due to spatial discretization, for all experiments in this section we take the solution of the semi-discrete problem $u_t = -Au$ to be the true solution.

We begin by illustrating the behavior of the polynomial approximation to the exponential. First, we would like to show how the accuracy of the polynomial scheme varies as m varies but Δt is fixed. We take $\Delta t = 0.01$ and let m vary from 1 to 20. The infinity-norm of the error between the exact result $e^{-A\Delta t}w_0$ and the approximation obtained from using the Arnoldi process as described in Section 3 is computed and scaled by the infinity norm of the exact solution. These relative error norms are then plotted in Figure 5 versus the subspace dimensions m. Notice that as is expected for $m = 20$ the error is zero up to the machine accuracy and the errors induced by the computation of $e^{-H}e_1$, since the approximation (3.5) is exact in this situation. In the tests dealing with polynomial approximation, the vector $e^{-\Delta t H_m}e_1$ is computed to very high accuracy, by compounding Taylor series expansions of degree 10. The composition is done by scaling H_m by a scalar δ in such a way that the spectral radius of δH_m is less than $1/2$. Thus, the evaluation of $e^{-\Delta t H_m}e_1$ may require a large number of successive evaluations of vectors of the form $e^{-\delta \Delta t H_m}e$. A more elaborate implementation using rational approximations as suggested earlier is under way.

Next we fix the dimension of the Krylov subspace to $m = 10$ and let Δt vary from $\Delta t = 0.005$ to $\Delta t = 0.1$ with an increment of 0.005. The relative error norms are plotted in Figure 2. Notice how the accuracy deteriorates at once instead of progressively.

We now consider a three-dimensional version of the previous test problem, i.e., we discretize the problem

$$\begin{aligned} u_t &= u_{xx} + u_{yy} + u_{zz},\ x,y,z \in (0,1) \\ u &= 0 \ \text{ on the boundary} \end{aligned}$$

using 17 grid points in each direction, yielding a matrix of size $N = 15^3 = 3375$. The experiment we now describe was performed on a Cray Y-MP/832

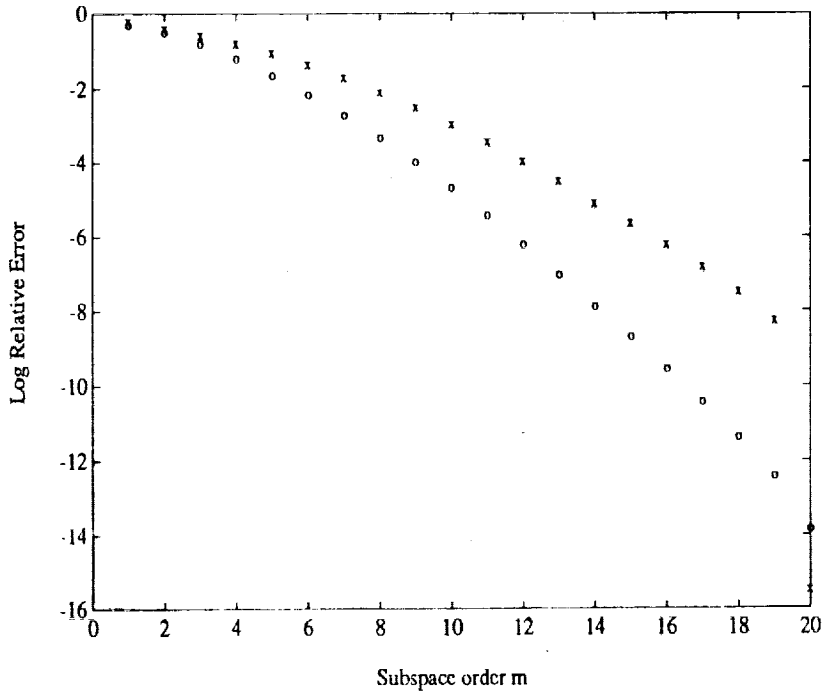

FIG. 1. *Behavior of polynomial approximation for* $\Delta t = 0.005$ *(o) and* $\Delta t = 0.01$ *(x) as the degree* m *varies.*

(8 processors). Again the initial conditions are chosen once the matrix is discretized, in such a way that the solution is known for all t. We take

$$u(0, x_i, y_j, z_k) = \sum_{i',j',k'=1}^{n} \frac{1}{i'+j'+k'} \sin\frac{ii'\pi}{n+1} \sin\frac{jj'\pi}{n+1} \sin\frac{kk'\pi}{n+1} \tag{5.2}$$

The above expression is simply an explicit linear combination of the eigenvectors of the discretized operator.

The purpose of this experiment is to illustrate the efficiency of using high accuracy schemes versus low accuracy schemes. This point was stressed earlier and constitutes one of the main motivations for this paper. As will be seen later the same conclusions also hold for the methods based on rational approximations to the exponential.

Assume that we want to integrate the above equation between t=0 and t=0.1, and achieve an error-norm at $t = 0.1$ which is less than $\epsilon = 10^{-10}$. Here by error-norm we mean the 2-norm of the absolute error. We can

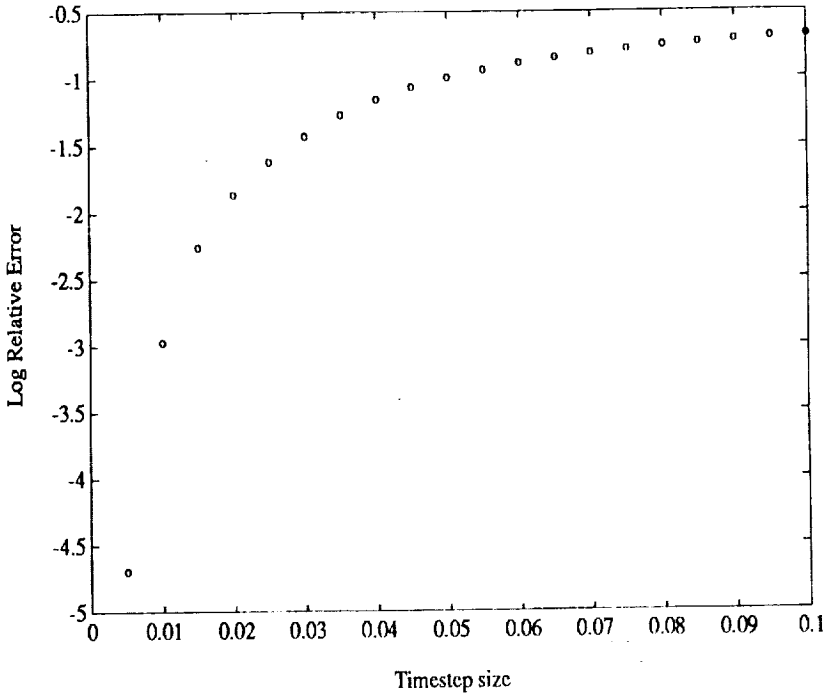

FIG. 2. *Behavior of polynomial approximation for* $m = 10$ *as timestep* Δt *varies.*

vary both the degree m and the time-step Δt. In a normal procedure we would first choose a degree m and then try to determine the maximum Δt allowed to achieve the desirable error level. However, for convenience, we proceed in the opposite way: we first select a step-size Δt and then determine the minimum m that is needed to achieve the desirable error level. Here the vector $e^{-H_m}e_1$ was computed via the diagonalization of the tridiagonal matrix H_m using EISPACK's routine IMTQL2. This is clearly not the most efficient technique since H_m is tridiagonal. What is shown in Table 1 is the various time steps chosen (column 1) and the minimum values of m (column 2) to achieve an absolute error less than $\epsilon = 10^{-10}$ at t=0.1. We show in the third column the total number of matrix-by-vector multiplications required to complete the integration. The times required to complete the integration on a Cray Y-MP are shown in two parts in columns 4 and 5. Since we used an inefficient algorithm to compute $e^{-H_m}e_1$ we showed the total time for performing this operation (denoted by Time_m) separately in column 5. The remaining time denoted by Time_N and shown in column 4, represents the time for performing the Arnoldi process and the linear combinations of the

TABLE 1

Performance of the polynomial scheme with varying accuracy on the Cray Y-MP/832.

Δt	m	M-vec's	Time_N	Time_m	$\|Error\|_2$
0.5000E-04	6	12006	0.8264E+01	0.3840E+00	0.3808E-10
0.1000E-03	7	7007	0.4779E+01	0.2459E+00	0.1298E-10
0.5000E-03	10	2010	0.1329E+01	0.9062E-01	0.1338E-10
0.1000E-02	12	1200	0.7968E+00	0.6331E-01	0.1946E-10
0.5000E-02	20	400	0.2593E+00	0.3479E-01	0.7336E-10
0.1000E-01	26	260	0.1646E+00	0.2939E-01	0.5304E-10
0.2000E-01	33	165	0.1030E+00	0.2487E-01	0.9857E-10
0.3000E-01	39	156	0.9504E-01	0.2856E-01	0.6247E-10
0.4000E-01	44	132	0.8154E-01	0.2847E-01	0.4098E-10
0.5000E-01	49	98	0.5879E-01	0.2446E-01	0.5787E-10
0.1000E+00	69	69	0.4134E-01	0.3170E-01	0.7494E-10

Arnoldi vectors. Since for all tests $m \leq 69$, and H_m is tridiagonal symmetric, one can expect that with an efficient algorithm the total time for evaluating the vector $e^{-H_m}e_1$ will represent a small portion of the total execution time, given the size of this problem. However, as is indicated by the last entry of the table, this time may become nonnegligible compared with the time Time_N as m increases, if an inefficient algorithm is used, even though the total number of operations involved is much smaller than that in the rest of the computation. Another point is that the matrix is symmetric, so we have used a Lanczos algorithm to generate the $v_i's$ instead of the full Arnoldi algorithm. No reorthogonalization of any sort was performed. The matrix consists of 7 diagonals, so the matrix by vector products are performed by diagonals resulting in a very effective use of the vector capabilities of the YMP. Based on the time Time_N for the last entry of the table, we have estimated that the average Mflops rate reached (excluding the calculation of $\exp\{-H_m\}e_1$) was around 161. This is achieved with virtually no code optimization.

Note the very rapid decrease in the total number of matrix by vector products required. The ratio between the lowest degree $m = 6$ and the highest degree $m = 69$ is 174. The corresponding ratio between the two times is roughly 200. The case $m = 69$ can achieve the desired accuracy in just one step, i.e., with $\Delta t = 0.1$. On the other hand for $m = 6$ a time-step of $\Delta t = 0.00005$ must be taken resulting in a total of 2000 steps. We should point out that we are restricting ourselves to a constant time-step, but more efficient variable time stepping procedures are likely to reduce the total number of steps needed. From the result of Theorem 3.1, these

observations come with no surprise. In effect, increasing the dimension of the Krylov subspace, will increase the accuracy in such a way that a much larger ρ, i.e., a larger Δt, can quickly be afforded.

We next test the rational approximations described in Section 4. The program asks for the *type* (Padé or Chebyshev) and *order* of the diagonal rational approximation. In the Padé case, the coefficients of the numerator and denominator polynomials in the rational function are numerically evaluated from (4.8). In the Chebyshev case the coefficients are taken directly from [1]. Subsequently, the IMSL routine ZRPOLY (based on the Jenkins-Traub algorithm) is used to compute the roots and poles of the rational approximation. The partial fraction coefficients are then computed from (4.7). The new solution is found by solving the independent complex tridiagonal systems in parallel and combining the results. In our algorithm we take advantage of the feature of the decomposition alluded to in (4.9). In this fashion, for an approximation of degree r we only need to solve $\lfloor \frac{r}{2} \rfloor$ complex systems and $\lceil \frac{r}{2} \rceil - \lfloor \frac{r}{2} \rfloor$ real systems (corresponding to the possible real root).

Figure 3 shows the behavior of the error in one time-step for Padé and Chebyshev diagonal approximations of orders 1, 10, 4 and 14 as the steplength Δt varies. Thus, it is the rational approximation analogue of Figure 2. The case of diagonal Padé order 1 is of course identical to the Crank-Nicolson scheme, so that figure indicates us the behavior of a standard method in comparison to the high-order methods we are proposing.

Judging from the errors, it would seem that one could discard the Padé schemes as inferior. It is known however that Chebyshev approximation reaches maximum error at 0, exactly where Padé does best. An example of the better behavior of Padé for small steps can be seen in Figure 4, where $\Delta t = 0.0005$ to 0.01. We note that there exist techniques to avoid this error behavior of Chebyshev approximation [23].

To test the performance of rational approximation method we used the 1-dimensional problem but with $u(0, x_j)$ taken to be the j^{th} component $\sin \frac{j\pi}{n+1}$ of the eigenvector of A corresponding to the eigenvalue of smallest modulus. The dimension n was chosen to be 98. Diagonal approximation was used throughout ($m = r$). When $m = 1$ real arithmetic was used. The objective was to integrate from 0 to $T \in [1, 1 + \Delta t]$ so that the maximum error ($\|error\|_\infty$) at T is less than $\epsilon = 10^{-9}$. The optimal Δt_{op} is the maximum Δt which achieves error tolerance ϵ at T. This is difficult to compute exactly and we determined numerically an approximation Δt_{op} so that an underestimation will add only a minimal amount of iterations. We only show the results for the Padé case with degrees 1, 2, 4, 6, 8. From Section 4.5 there are 1, 1, 2, 3, and 4 tridiagonal systems to be solved per step (using LU). Our results are summarized in Table 2 and Figure 5.

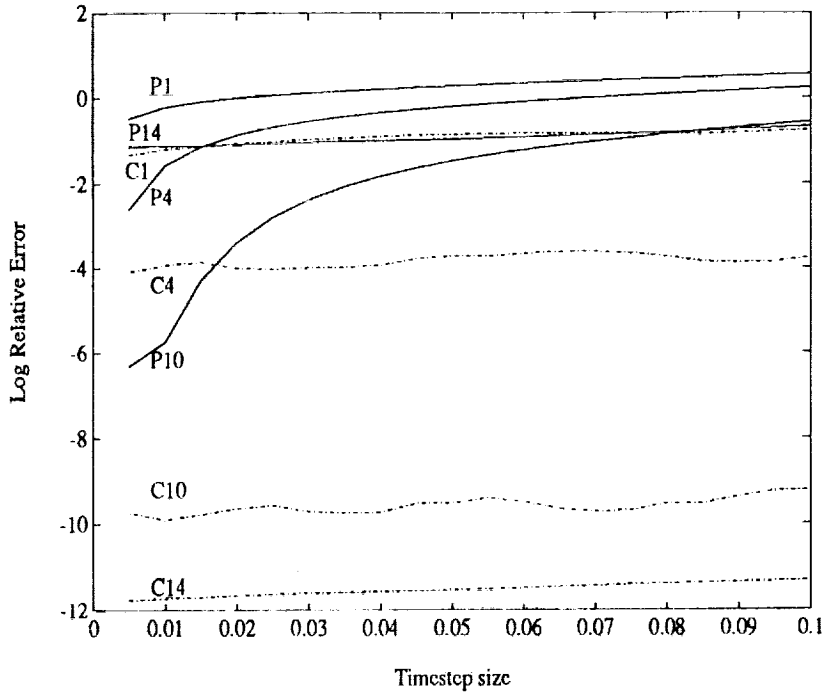

FIG. 3. *Behavior of rational approximation: Pm, Cm are the curves for m th order Padé and Chebyshev respectively as Δt varies.*

The experiment demonstrates the following crucial facts:

1. Crank-Nicolson on 8 CEs achieves a speedup of 1.43 over its 1 CE run.
2. The 8th order scheme achieves a speedup of 3.5 (for 4 or more CEs) over its 1 CE run.
3. The 8th order scheme achieves a speedup of 167 over Crank-Nicolson.

Item (1) shows the difficulty of the low order scheme to profit from parallel processing. Item (2) shows the considerably better behavior of the higher order schemes due to the use of partial fractions. Item (3) shows the excellent behavior and potential of high degree methods compared with standard low order schemes.

6. Concluding remarks. We have proposed three parallel techniques for solving parabolic equations. The first method based on Krylov subspaces is the easiest to implement. It has the advantage of not requiring any solution of linear systems. On the other hand it is basically an explicit method and may be inefficient for very stiff problems. The other two methods rely on a

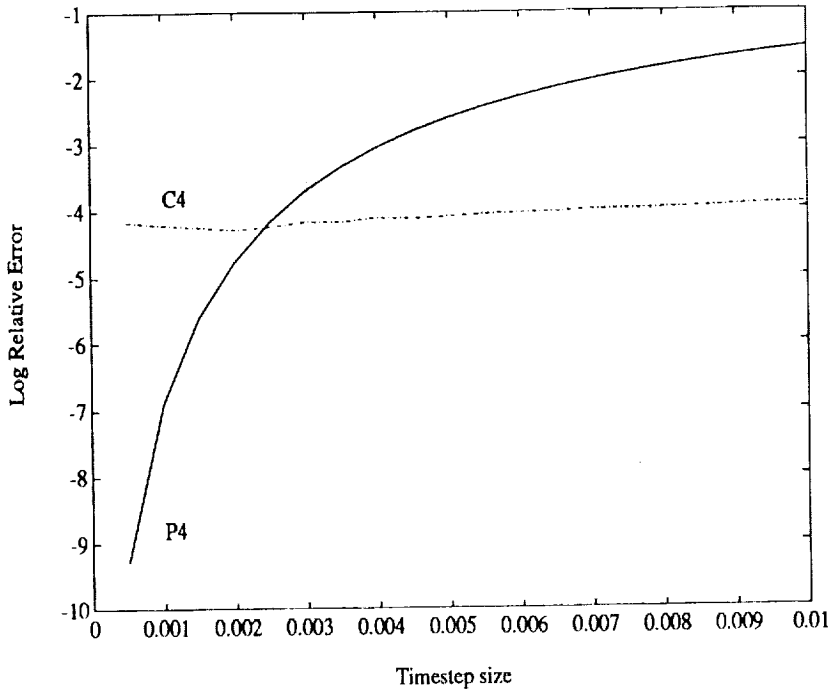

FIG. 4. *Behavior of fourth ($m = 4$) order Padé (P4) and Chebyshev (C4) rational approximations as Δt varies.*

rational approximation to the exponential. The basic idea of their parallel implementations is to resort to partial fraction expansions. This transforms the basic problem of solving a linear system with a product of matrices into that of solving independent linear systems.

We would like to conclude with two comments, placing ourselves in the more general framework of the parallel solution of systems of Ordinary Differential Equations. First, it is becoming apparent that explicit methods will regain interest with parallel processing. These methods are particularly appealing for three-dimensional problems, especially in conjunction with highly accurate schemes. Second, high order integration methods seem to be important in ODE methods, in order to achieve parallelism. In one-dimensional problems they are mandatory since each step requires solving a tridiagonal system, with little room for parallelization. For two or three dimensional problems, the use of the techniques based on partial fraction expansions described in this paper, allow us to bypass the need to parallelize the sparse linear system solvers which are difficult to optimize on supercomputers.

TABLE 2
Performance of different degree Padé schemes on the Alliant FX/8.

Δt	m	Steps	Time
0.4910E-03	1	2037	0.1838E+01
0.1950E-01	2	52	0.1740E+00
0.1600E+00	4	7	0.2590E-01
0.4000E+00	6	3	0.1320E-01
0.5000E+00	8	2	0.1110E-01

Acknowledgement. The research of the first author was supported by the National Science Foundation under Grants No. US NSF-MIP-8410110, US NSF DCR85-09970, US NSF CCR-8717942 and by AT&T Grant AT&T AFFL67Sameh. The research of the second author was supported by NASA under USRA Grant No. NCC 2-387.

REFERENCES

[1] A. J. CARPENTER, A. RUTTAN, AND R. S. VARGA. Extended numerical computations on the 1/9 conjecture in rational approximation theory. In P. R. Graves-Morris, E. B. Saff, and R. S. Varga, editors, *Rational Approximation and Interpolation*, pages 383–411, Springer-Verlag, Berlin, 1984.

[2] J. C. CAVENDISH, W. E. CULHAM, AND R. S. VARGA. A comparison of Crank-Nicolson and Chebyshev rational methods for numerically solving linear parabolic equations. *J. Comput. Phys.*, 10:354–368, 1972.

[3] W. J. CODY, G. MEINARDUS, AND R. S. VARGA. Chebyshev rational approximations to e^{-x} in $(0, +\infty)$ and applications to heat-conduction problems. *J. Approx. Theory*, 2(1):50–65, March 1969.

[4] G. FAIRWEATHER. A note on the efficient implementation of certain Padé methods for linear parabolic problems. *BIT*, 18:106–109, 1978.

[5] R. FREUND. *On conjugate gradient type methods and polynomial preconditioners for a class of complex non-Hermitian matrices.* Technical Report 88-44, RIACS, NASA Ames Research Center, 1989.

[6] E. GALLOPOULOS. *A partial fraction decomposition approach to improved efficiency of some parabolic solvers.* Technical Report 874, Center for Supercomputing Research and Development, May 1989.

[7] E. GALLOPOULOS AND Y. SAAD. *Parallel block cyclic reduction algorithm for the fast solution of elliptic equations.* Technical Report 659, Center for Supercomputing Research and Development, April 1987.

[8] C. W. GEAR. *The potential for parallelism in ordinary differential equations.* Technical Report R-86-1246, Department of Computer Science, University of Illinois at Urbana Champaign, Feb. 1986.

[9] P. HENRICI. *Applied and Computational Complex Analysis.* Volume 1, Wiley, 1974.

[10] D. J. KUCK, E. S. DAVIDSON, D. L. LAWRIE, AND A. H SAMEH. Parallel supercomputing today and the Cedar approach. *Science*, 231:967–974, February 1986.

[11] H. T. KUNG. New algorithms and lower bounds for the parallel evaluation of certain rational expressions and recurrences. *J. Assoc. Comput. Mach.*, 23(2):252–261,

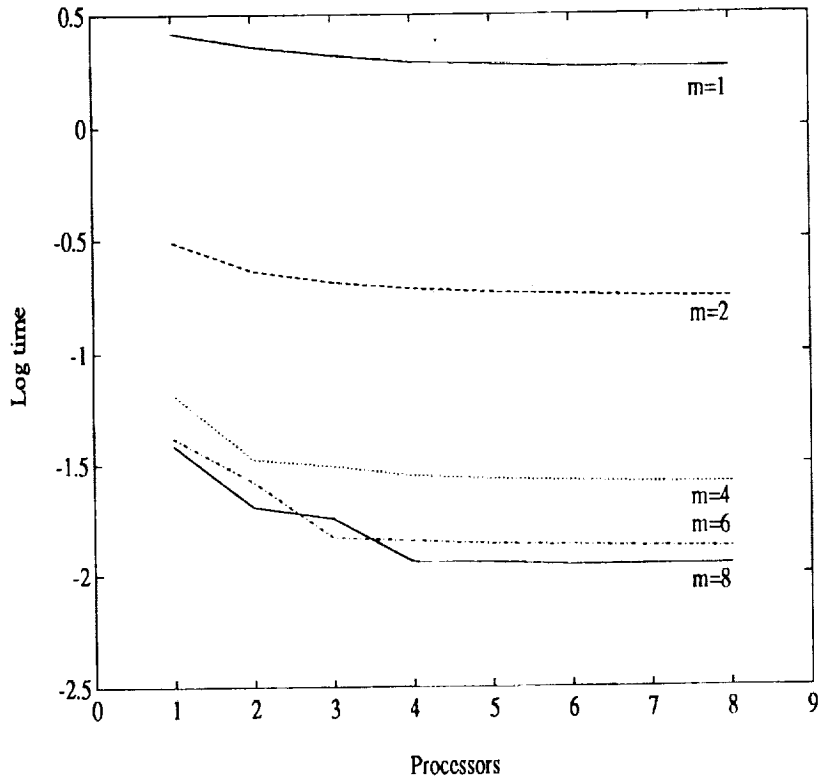

FIG. 5. *Performance of rational Padé approximation of degree m on the Alliant FX/8 varying the number of processors.*

April 1976.

[12] J. D. LAWSON AND D. A. SWAYNE. High-order near best uniform approximations to the solution of heat conduction problems. In *Proc. IFIP Congress 80 - Information Processing 80*, pages 741–746, North Holland, New York, 1980.

[13] C. MOLER AND C. VAN LOAN. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Rev.*, 20(4):801–836, October 1978.

[14] S. P. NØRSETT. Restricted Padé approximations to the exponential function. *SIAM J. Numer. Anal.*, 15(5):1008–1029, Oct. 1978.

[15] B. NOUR-OMID. Applications of the Lanczos algorithm. *Comp. Phy. Comm.*, 53, 1989.

[16] B. N. PARLETT. *The Symmetric Eigenvalue Problem.* Prentice Hall, Englewood Cliffs, 1980.

[17] M. F. REUSCH, L. RATZAN, N. POMPHREY, AND W. PARK. Diagonal Padé approximations for initial value problems. *SIAM J. Sci. Statist. Comput.*, 9(5):829–838, September 1988.

[18] M. J. SCHAEFER. *A polynomial based iterative method for linear parabolic equations.* Technical Report 661, Center for Supercomputing Research and Development, May 1987.

[19] D. A. SWAYNE. Matrix operations with rational functions. In *Proc. 7th Manitoba Conf. Numerical Mathematics*, pages 581–589, Utilitas Mathematica, Winnipeg, Manitoba, 1977.

[20] R. A. SWEET. A parallel and vector cyclic reduction algorithm. *SIAM J. Sci. Statist. Comput.*, 9(4):761–765, July 1988.

[21] H. TAL-EZER. Spectral methods in time for parabolic problems. *SIAM J. Numer. Anal.*, 26(1):1-11, Feb. 1989.

[22] R. S. VARGA. On higher order stable implicit methods for solving parabolic partial differential equations. *J. Math. Phys.*, 40:220–231, 1961.

[23] R. S. VARGA. Some results in approximation theory with applications to numerical analysis. In B. E. Hubbard, editor, *Numerical Solution of Partial Differential Equations – II*, pages 623–649, Academic Press, New York, 1971.

[24] J. H. WILKINSON. *The Algebraic Eigenvalue Problem.* Oxford University Press, 1965.

[25] R. E. WYATT. The recursive residue generation method. *Adv. Chem. Phy. Chap. 5*, 1988.

[26] V. ZAKIAN. Properties of I_{MN} and J_{MN} approximants and applications to numerical inversion of Laplace transforms and initial value problems. *J. Math. Anal. Applic.*, 50:191-222, 1975.