COMM. MATH. SCI.
Vol. 3, No. 3, pp. 335–372

# APPLICATION OF WEEKS METHOD FOR THE NUMERICAL INVERSION OF THE LAPLACE TRANSFORM TO THE MATRIX EXPONENTIAL*

PATRICK O. KANO†, MOYSEY BRIO‡, AND JEROME V. MOLONEY§

**Abstract.** Weeks method is a well established algorithm for the numerical inversion of scalar Laplace space functions. In this paper, we extend the method to the inversion of matrix functions of a single time variable and assess the qualities of this approach. To illustrate and quantify our discussion, we compute the matrix exponential by means of an FFT based algorithm. Particular attention is paid to a comparison of algorithms for the automated selection of two tuning parameters. In addition to selection algorithms from the literature, we introduce a pseudospectra based approach for the particular case of the matrix exponential. Finally, applications involving both pathological matrices and the numerical solution of differential equations highlight the utility of the method.

**Key words.** Weeks Method, Matrix Exponential, Nonparaxial Beam Propagation Equation

**MSC Number.** 65R10, 44A10, 35Q60

## 1. Introduction

In this paper, we present an approach to calculate a matrix function from its Laplace space representation. To perform the numerical inversion of the Laplace transform, we extend the well established scalar Weeks method to matrix functions parameterized by time. This entails a matrix formulation, an analysis of the errors, and a systematic comparison of selection algorithms for the method's parameters. Also novel is a pseudospectra based approach to the selection of these two tuning parameters. To illustrate and quantify our discussion, we focus upon the computation of the matrix exponential from its corresponding Laplace space function [23].

Due to its inherent ill-posedness, the numerical inversion of the Laplace transform

$$f(t) = \frac{1}{2\pi i}\int_{\Gamma} e^{st}F(s)ds \tag{1.1}$$

is a long standing problem. This fact is attested to by the existence of numerous algorithms for this procedure [7, 40]. One perspective is that the various approaches to the numerical inversion are different regularization techniques.

From experimentation and review, four main algorithms for numerical Laplace transform inversion have proved to be of use, the Post-Widder formula [2, 40, 41], Fourier Series Expansion [9], Talbot's method [31], and the Weeks method [1, 36, 37]. Each has found a domain of application corresponding to the ability of the algorithm to invert certain classes of Laplace space functions. To highlight the strengths and weaknesses of the Weeks method, we briefly discuss these other algorithms.

The distinct property of the Post-Widder formula (1.2) is that it requires sampling

*Received: November 30, 2004; accepted (in revised version): May 18, 2005. Communicated by Anna Gilbert.

†Arizona Center for Mathematical Sciences, Department of Mathematics, University of Arizona, 617 N. Santa Rita, Tucson, AZ 85721, USA (pkano@math.arizona.edu).

‡Arizona Center for Mathematical Sciences, Department of Mathematics, University of Arizona, 617 N. Santa Rita, Tucson, AZ 85721, USA (brio@math.arizona.edu).

§Arizona Center for Mathematical Sciences, Department of Mathematics, University of Arizona, 617 N. Santa Rita, Tucson, AZ 85721, USA (jml@acms.arizona.edu).

of the Laplace space function $F(s)$ only on the real line.

$$f(t) = \lim_{n\to\infty} \frac{(-1)^n}{n!}\left(\frac{n}{t}\right)^{n+1} F^{(n)}\left(\frac{n}{t}\right) \tag{1.2}$$

The difficulty of computing high order derivatives $F^{(n)}$ can be mitigated by using finite differences. This gives rise to the well-known Gaver functionals

$$\phi_n(t) = n\frac{\ln(2)}{t}\binom{2n}{n}\sum_{j=0}^{n}(-1)^j\binom{n}{j}F\left((n+j)\frac{\ln(2)}{t}\right) \tag{1.3}$$
$$f(t) = \lim_{n\to\infty}\phi_n(t).$$

Unfortunately, this method suffers from very slow convergence $|f(t)-\phi_n(t)|\sim 1/n$ as $n\to\infty$, so that a series accelerator is demanded for any practical computation. Recent work [41] has established the utility of the Wynn-rho algorithm for the acceleration. Another drawback to a Post-Widder based approach is its sensitivity to roundoff errors. This is due to the potentially large coefficients in the Gaver functionals. Some success using high precision variables to subdue the roundoff error has been reported [2]. Although notable packages exist [4], the present lack of a standardized library for arbitrary precision variables for low level languages however makes this approach cumbersome.

The Fourier Series method is excellent for dissipative problems [11] and widely used in the hydrological community to perform long time integration of weekly damped systems. For these dissipative problems, it is capable of providing spectral accuracy in time similar to Fourier (FFT) spectral accuracy in space [6, 28, 30]. As such, it is typically formulated for real time domain functions $f(t)$. It utilizes the standard Bromwich contour $s=\sigma+iy$ familiar from the analytic inversion of the Laplace transform. The result is a weighted Fourier transform and after numerical integration, a Fourier series. While certainly one of the faster numerical inversion methods, this approach has the disadvantage that the solution is not well determined for $t\approx 0$ and $t\approx 2T$ where $T$ is the fundamental frequency of the Fourier expansion. It also requires a series acceleration due to the highly oscillatory nature of the integrand. Typically, one uses the quotient-difference algorithm, which, as is true for any acceleration method, is unstable for certain classes of coefficients [9, 38].

Talbot's method applies a deformed contour to the integration of the complex inversion integral (1.1). It maps the standard Bromwich contour to one which opens toward the negative real axis $s(\theta)=\sigma+\lambda\theta(i\nu+\cot(\theta))$ where $\lambda,\nu,\sigma\in\mathbb{R}$ and $\theta\in(-\pi,\pi)$. It thus inherently assumes that the physical system damps highly oscillatory terms and is not appropriate for purely conservative problems. The nontrivial task of developing software to select optimal values for these parameters has been undertaken for scalar functions [24]. Simplified alternatives to Talbot's contour have also been proposed [2].

Weeks method has the principal advantage over these three other methods of returning an analytic formula for the time domain function. In particular, it assumes that a smooth function [1] $f(t)$ can be well approximated by an expansion in terms of Laguerre polynomials $L_n(t)$

$$f(t)\approx e^{\sigma t}\sum_{n=0}^{N-1}a_n e^{-bt}L_n(2bt). \tag{1.4}$$

It is thus highly efficient for multiple evaluations in the time domain. Computing the function at a new time with these other methods requires essentially restarting the numerical inversion procedure. Furthermore, it does not inherently utilize dissipation and is equally applicable to real and complex time domain functions. It can also be computed using double precision variables. Despite these strengths and its popularity for scalar functions, little work has been done to extend it to matrix-valued functions. Part of this may be due to the difficulty of choosing appropriate values for the expansion parameters $\sigma$ and $b$, which can vary dramatically with Laplace space function. In this paper, we attempt to address many of these issues.

The present text is organized as follows. In the first section, we extend Weeks method to the computation of the time domain function $f(t)$ for a general matrix function $F(s)$. We provide midpoint and FFT algorithms for the coefficients $a_n$ and an estimate of the errors following the recent work by Weideman for the scalar Weeks method [37]. Once the theoretical apparatus has been established, we apply it to the particular case of the matrix exponential.

In the second portion, we focus upon the algorithms for the selection of $\sigma$ and $b$. We have investigated three approaches taken from the literature, Weeks original suggestions, a direct minimization of the Cauchy estimate for the truncation error, and a total error minimization. We also introduce a fourth algorithm based on the concept of the pseudospectrum [35]. In all these cases, we select $\sigma$ and $b$ based on the properties of the matrix as a single mathematical entity instead of performing the more expensive element-by-element parameter selection.

Illustrations of the accuracy and stability of Weeks method for the matrix exponential are provided in the third section. Our examples include both matrices with pathological spectra and matrices which arise in the numerical simulation of PDEs. The first differential equation is the usual advection and diffusion equation, which has been solved with great success by other numerical Laplace transform methods, chiefly the Fourier series method [30]. The second PDE is a dispersive equation, the scalar nonparaxial beam propagation equation from computational photonics [42]. The solutions to this equation are complex and are typically propagated by conservative algorithms. It thus presents an interesting challenge to any numerical Laplace transform inversion procedure. Since one typically encounters sparse and dense matrices, our examples include both.

Finally, to conclude, we summarize some of the main points of this paper and provide suggestions for future work.

## 2. Weeks Method

Weeks method is one of the most well known algorithms for the numerical inversion of a scalar Laplace space function [1, 7, 12, 37] [1]. Its popularity is due primarily to the fact that it returns an explicit expression for the time domain function. In particular, Weeks method assumes that a smooth function of bounded exponential growth $f(t)$, given by the inverse Laplace transform

$$f(t) = \frac{1}{2\pi i} \int_{\Gamma} e^{st} F(s) ds \tag{2.1}$$

where $\Gamma$ is a contour in the complex plane, can be expressed as the limit of an expan-

[1]The notation in this paper follows the conventions used by Weideman [37].

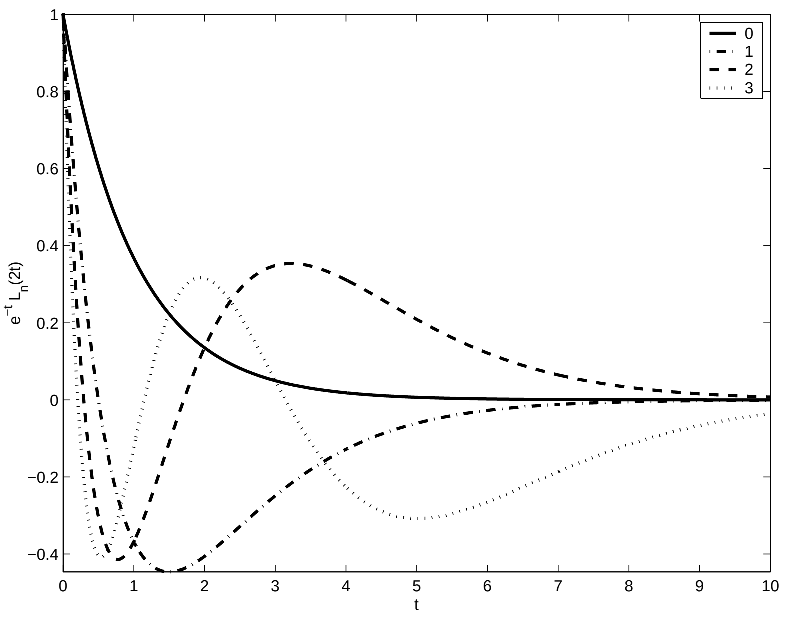

FIG. 2.1. *Laguerre polynomials*

sion in scalar Laguerre polynomials

$$f_N(t) = e^{\sigma t} \sum_{n=0}^{N-1} a_n e^{-bt} L_n(2bt). \tag{2.2}$$

The functions $L_n(x)$ for $n \geq 0$ are defined by the equation

$$L_n(x) = \frac{e^x}{n!} \frac{d^n}{dx^n}(e^{-x}x^n) \tag{2.3}$$

on $x \in (0,\infty)$. Clearly, the coefficients $a_n$, which may be scalars, vectors, or matrices, contain the information particular to the Laplace space function $F(s)$ and may be complex if $f(t)$ is complex. More importantly, these coefficients are time independent so that $f(t)$ can be evaluated at multiple times from a single set of coefficients.

The two free scaling parameters $\sigma$ and $b$ in the expansion must be selected according to the constraints that $b > 0$ and $\sigma > \sigma_0$, where $\sigma_0$ is the abscissa of convergence. The restriction of $b$ to positive values ensures that the weighted Laguerre polynomials $e^{-bt}L_n(2bt)$ are well behaved for large $t$, Fig. 2.1. More importantly, this condition implies that $|e^{-bt}L_n(2bt)| < 1$. The convergence of the series is uniform [13].

In this section, we present an algorithm to compute the coefficients $a_n$ for a general vector or matrix function $F(s)$ and an analysis of the errors for a fixed $\sigma$ and $b$. The case that $F(s) = (sI - A)^{-1}$ which corresponds to $e^{At}$ is illustrated in section (2.3). For clarity, we discuss the case that the coefficients $a_n$ are matrices. In a high level language such as MATLAB however, we were able to compute vector and matrix coefficients with only minor changes to our codes.

**2.1. General Function $F(s)$.** The computation of the coefficients begins with an integration in the complex plane

$$f(t)=\frac{1}{2\pi i}\int_{\Gamma} e^{st}F(s)ds. \tag{2.4}$$

If one chooses the Bromwich contour $\Gamma(s)=\sigma+iy$, with $\sigma>\sigma_0$, $y\in\mathbb{R}$

$$f(t)=\frac{e^{\sigma t}}{2\pi}\int_{-\infty}^{\infty} e^{iyt}F(\sigma+iy)dy \tag{2.5}$$

and assumes the expansion

$$f(t)=e^{\sigma t}\sum_{n=0}^{\infty} a_n e^{-bt}L_n(2bt) \tag{2.6}$$

then equating the two expressions yields

$$\sum_{n=0}^{\infty} a_n e^{-bt}L_n(2bt)=\frac{1}{2\pi}\int_{-\infty}^{\infty} e^{iyt}F(\sigma+iy)dy. \tag{2.7}$$

It is known that the weighted Laguerre coefficients have the Fourier representation [37]

$$e^{-bt}L_n(2bt)=\frac{1}{2\pi}\int_{-\infty}^{\infty} e^{iyt}\frac{(iy-b)^n}{(iy+b)^{n+1}}dy \tag{2.8}$$

Performing the appropriate substitution, assuming it is possible to interchange the sum and integral, and equating integrands thus leaves

$$\sum_{n=0}^{\infty} a_n \frac{(iy-b)^n}{(iy+b)^{n+1}}=F(\sigma+iy). \tag{2.9}$$

The functions $\frac{(iy-b)^n}{(iy+b)^{n+1}}$ form a complete, orthogonal basis in $L_2(\mathbb{R})$ [18]. In principle, one could try to use directly the orthogonality of the basis to determine $a_n$. However, these functions are high oscillatory and thus not amendable to numerical integration. One method of recourse to this problem is to apply a Möbius transformation to map $s$ to a new complex variable $w$

$$w=\frac{s-\sigma-b}{s-\sigma+b}. \tag{2.10}$$

For the Bromwich contour $s=\sigma+iy$

$$w=\frac{iy-b}{iy+b} \tag{2.11}$$

so that $s$ is mapped to the unit circle $|w|=1$. A more subtle result of this transformation is that the singularities of $F(s)$ in the half-plane $\sigma<\sigma_0$ are mapped to the exterior of the unit circle in the $w$ plane. An exception to this occurs when $F(s)$ has a singularity at infinity which then maps onto the unit circle [7]. In the following we will assume that the singularities of $F(s)$ occur in a finite region of the complex plane.

The Laplace space function corresponding to the matrix exponential belongs to this class of functions.

With the Möbius transformation and using $y=\frac{ib(w+1)}{w-1}$ along the Bromwich contour, equation (2.9) can now be expressed as

$$\sum_{n=0}^{\infty} a_n w^n = (iy+b)F(\sigma+iy) \tag{2.12}$$

$$\sum_{n=0}^{\infty} a_n w^n = \frac{2b}{1-w}F\left(\sigma-b\frac{w+1}{w-1}\right). \tag{2.13}$$

In this form we see that the coefficients $a_n$ of the original expansion (2.2) are also the coefficients of a Maclaurin series. The radius of convergence $R$ is strictly greater than unity due to our selection of functions $F(s)$ which do not have a singularity at infinity. Furthermore, within this radius, the power series converges uniformly.

Cauchy's integral theorem [3] provides a method to compute $a_n$. Since the function is analytic inside the radius of convergence $R>1$, the integration can be performed along the unit circle $w=e^{i\theta}$

$$a_n = \frac{1}{2\pi i}\int_{|w|=1} \frac{1}{w^{n+1}} \frac{2b}{1-w} F(\sigma - b\frac{w+1}{w-1})dw \tag{2.14}$$

$$a_n = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{-in\theta} \frac{2b}{1-e^{i\theta}} F(\sigma - b\frac{e^{i\theta}+1}{e^{i\theta}-1})d\theta \tag{2.15}$$

Numerically the evaluation of the integral can be computed very accurately using the midpoint rule; $\theta_m=\frac{m\pi}{M}$, where $n=0,\ldots,N-1$ and $m=-M,\ldots,M-1$.

$$a_n \approx \frac{e^{-in\pi/(2M)}}{2M}\sum_{m=-M}^{M-1} e^{-in\theta_m}\frac{2b}{1-e^{i\theta_{m+1/2}}}F\left(\sigma-b\frac{e^{i\theta_{m+1/2}}+1}{e^{i\theta_{m+1/2}}-1}\right) \tag{2.16}$$

Implementation of this formula for the coefficients (2.16) requires consideration, particularly, with respect to memory and computing costs.

If there is sufficient memory to store the function evaluated along the contour then a large enhancement in speed can be obtained by using the fast Fourier transform [37]. The summation (2.16) with $N=M$ can be performed in $O(Nlog(N))$ operations as opposed to $O(N^2)$ required by the direct application of the midpoint rule. The coefficients corresponding to negative indices which result from the FFT algorithm are not needed for the summation and thus can be neglected.

For large matrices, it may be too demanding on memory to store a series of evaluations of $F(s)$. In this case, it is more efficient to overwrite the Laplace space function after each evaluation at a point in the contour. Because the midpoint rule converges quickly for a periodic function, a reasonable choice of $M$ is $N=M$.

If the matrix function $F(s)$ is dense, the evaluations of the Laplace space function can dominate the computations. It is at this point that the use of a preliminary Schur decomposition, although not essential, may be advantageous. Every square matrix $A$ has a complex Schur factorization of the form $A=QTQ^H$ where $T$ is a triangular matrix whose diagonal elements are the eigenvalues of $A$ and $Q$ is a unitary matrix whose columns are not necessarily however the eigenvectors [14]. With this representation, $f(A)=Qf(T)Q^H$. After a preliminary factorization one might suppose to simply compute the function of the triangular matrix and proceed without Weeks method. Indeed, Parlett's algorithm provides one with a method to compute the $f(T)$; for $i<j$,

$$f(T)_{ij} = t_{ij}\frac{f_{ii}-f_{jj}}{t_{ii}-t_{jj}} + \sum_{k=i+1}^{j-1}\frac{f_{ik}t_{kj}-t_{ik}f_{kj}}{t_{ii}-t_{jj}}. \tag{2.17}$$

The resulting matrix is also triangular. This method is in fact implemented in MATLAB's command *funm* for an arbitrary function of a matrix. One of the major advantages of this approach is speed. After the initial Schur decomposition is computed for a square $p\times p$ matrix in $0(p^3)$, the Parlett algorithm can be performed very quickly. The problem with the Schur-Parlett approach however can also be seen from equation (2.17). When the eigenvalues of $A$ are repeated or sufficiently close, the computation of $f_{ij}$ can not be performed reliably. This problem was originally addressed in Parlett's paper [26] and in more recent works by Higham and Davies [8, 19]. One can see from the level of sophistication of the papers however that the resolution of the issue is by no means settled. One must in general resort to another method to deal with the matrices corresponding to degenerate eigenvalues.

Also, one might suggest a series acceleration method for the coefficients in the Maclaurin series. However, from our investigations, we have found that the gains in accuracy do not justify the increased computation time. The potential instability of a nonlinear series acceleration also brings robustness into question.

Finally, once the coefficients have been computed and the parameters selected, it is necessary to perform the Laguerre expansion. A naive approach is to generate the Laguerre polynomials using the recurrence relation,

$$(n+1)L_{n+1}(x) = (2n+1-x)L_n(x) - nL_{n-1}(x) \tag{2.18}$$

with starting values

$$L_0(x) = 1$$
$$L_1(x) = 1-x$$

multiply by the coefficients, and compute the sum

$$f(t) = e^{(\sigma-b)t}\sum_{n=0}^{\infty} a_n L_n(2bt). \tag{2.19}$$

The Laguerre polynomials however can be large for increasing $n$ and thus lead to an unstable summation. A stable method which does not require explicit evaluation of the Laguerre polynomials is the backward Clenshaw algorithm [29].

## 2.2. Error Estimate.

Up to this point we have not been concerned with the accuracy of Weeks method. One of the stronger arguments for using it instead of another numerical Laplace transform inversion method is that a straight-forward estimate of the errors can be performed using standard methods of numerical analysis [37]. In contrast, an analysis of the errors in Talbot's or the Fourier series methods is significantly more involved [7, 25, 38]. The estimates we are able to obtain from our analysis are fundamental to a proper selection of the tuning parameters $\sigma$ and $b$.

We start by postulating that a smooth function $f(t)$

$$f(t) = e^{\sigma t}\sum_{n=0}^{\infty} a_n e^{-bt} L_n(2bt) \tag{2.20}$$

is approximated by the computed function $\tilde{f}(t)$

$$\tilde{f}(t) = e^{\sigma t} \sum_{n=0}^{N-1} a_n (1+\epsilon) e^{-bt} L_n(2bt) \tag{2.21}$$

where $\epsilon$ is on the order of the numerical precision. We note that there are three sources of error:

- *Discretization Error:* Due to the discrete sampling along the unit circle in the integral for the coefficients $a_n$.
- *Roundoff Error:* Due to the finite precision of the computations.
- *Truncation Error:* Due to the finite number of Laguerre expansion coefficients.

Here we will ignore the discretization error introduced in the computation of the coefficients since the integration (2.16) is computed spectrally accurately by the midpoint rule on the unit circle. That is, we assume that the coefficients in the approximation are exactly the coefficients in the true expansion except for the roundoff error. Typically one would expect to be able to neglect the roundoff error as well, however since it is multiplied by $e^{(\sigma-b)t}$ it may add significantly to the total error.

Despite the fact that the contributions from the roundoff error may be considerable, it is the truncation error which typically dominates. Although potentially expensive, a computational estimate can be obtained by calculating twice the number of coefficients than that used in the Laguerre expansion and then defining the truncation error from the tail coefficients.

Using these ideas, the total error including the truncation and roundoff can be computed as follows. If we let $\|\cdot\|_F$ denote the Frobenius norm, then

$$\frac{\|f(t) - \tilde{f}(t)\|_F}{e^{\sigma t}} \le \|\bar{T}\|_F + \|\bar{R}\|_F \tag{2.22}$$

where the truncation matrix $\bar{T}$ and roundoff matrix $\bar{R}$ are

$$\bar{T} = \sum_{n=N}^{\infty} a_n e^{-bt} L_n(2bt) \tag{2.23}$$

$$\bar{R} = \epsilon \left( \sum_{n=0}^{N-1} a_n e^{-bt} L_n(2bt) \right) \tag{2.24}$$

and $a_n$ are matrices. Applying elementary analysis and using the indices, $j = 0, \ldots, J-1$ and $k = 0, \ldots, K-1$, to denote the position $(j,k)$ in the matrix

$$\|\bar{T}\|_F = \sqrt{\sum_{j=0}^{J-1} \sum_{k=0}^{K-1} |T_{jk}|^2} \tag{2.25}$$

$$|T_{jk}|^2 = \left| \sum_{n=N}^{\infty} a_{njk} e^{-bt} L_n(2bt) \right|^2. \tag{2.26}$$

Since for $t>0$ and $n\geq 0$, $|e^{-bt}L_n(2bt)|<1$

$$|T_{jk}|^2 \leq \sum_{n=N}^{\infty} |a_{njk}|^2 \tag{2.27}$$

$$\|\bar{T}\|_F \leq \sqrt{\sum_{j=0}^{J-1}\sum_{k=0}^{K-1}\sum_{n=N}^{\infty} |a_{njk}|^2} \tag{2.28}$$

$$\|\bar{T}\|_F \leq \sqrt{\sum_{n=N}^{\infty} \|a_n\|_F^2}. \tag{2.29}$$

A similar computation can be performed for the roundoff error which together with the truncation yields the total error

$$E_{total} \leq e^{\sigma t}\left(\sqrt{\sum_{j=0}^{J-1}\sum_{k=0}^{K-1}\sum_{n=N}^{\infty} |a_{njk}|^2} + \epsilon\sqrt{\sum_{j=0}^{J-1}\sum_{k=0}^{K-1}\sum_{n=0}^{N-1} |a_{njk}|^2}\right) \tag{2.30}$$

or written more succinctly

$$E_{total} \leq e^{\sigma t}\left(\sqrt{\sum_{n=N}^{\infty} \|a_n\|_F^2} + \epsilon\sqrt{\sum_{n=0}^{N-1} \|a_n\|_F^2}\right). \tag{2.31}$$

This above computational method for estimating the truncation error is often appropriate. It however does not provide any additional insight into the convergence properties of the Laguerre expansion. Instead, the role that the radius of convergence plays in the *rate* of convergence of the Maclaurin series can be determined from the the Cauchy estimate for the truncation error. To compute this estimate consider a circular contour of radius $r\in(1,R)$ such that the function $F(\sigma - b\frac{w+1}{w-1})$ is analytic

$$a_n = \frac{1}{2\pi i}\int_{|w|=r} \frac{1}{w^{n+1}}\frac{2b}{1-w}F\left(\sigma - b\frac{w+1}{w-1}\right)dw \tag{2.32}$$

$$G(w) = \frac{2b}{1-w}F\left(\sigma - b\frac{w+1}{w-1}\right) \tag{2.33}$$

$$\|a_n\|_F \leq \frac{1}{2\pi}\int_{|w|=r} \frac{\|G(w)\|_F}{|w^{n+1}|}dw \tag{2.34}$$

$$\|G(w)\|_F \leq \max_{|w|=r}\|G(w)\|_F \equiv K(r) \tag{2.35}$$

$$\|a_n\|_F \leq \frac{1}{2\pi}\frac{K(r)}{r^{n+1}}\int_{|w|=r} dw \tag{2.36}$$

$$\|a_n\|_F \leq \frac{K(r)}{r^n} \tag{2.37}$$

$$\|\bar{T}\|_F \leq \sum_{n=N}^{\infty}\|a_n\|_F \tag{2.38}$$

$$\|\bar{T}\|_F \leq \frac{K(r)}{r^{N-1}(r-1)}. \tag{2.39}$$

Thus we see, that since $r \in (1,R)$ the coefficients always decay geometrically. For $R \approx 1$ however the coefficients decay slowly and the truncation error is large, while for $R \gg 1$, they decay more quickly.

**2.3. Application to the Matrix Exponential.** An illustration of the general formalism of section (2.2) is presented here for the matrix exponential. It becomes clear that this is a natural application of the inverse Laplace transform when one considers the general definition of a function $f$ of a matrix $A$

$$f(A) \equiv \frac{1}{2\pi i} \int_\Gamma f(s)(sI-A)^{-1} ds. \tag{2.40}$$

If $f(s) = e^{st}$, then

$$e^{At} \equiv \frac{1}{2\pi i} \int_\Gamma e^{st}(sI-A)^{-1} ds. \tag{2.41}$$

The Laplace transform function corresponding to the matrix exponential is obviously the resolvent matrix $F(s) = (sI-A)^{-1}$. With this fact, the calculations of the previous section proceed as follows

$$e^{At} = \frac{e^{\sigma t}}{2\pi} \int_{-\infty}^{\infty} e^{iyt}\left((\sigma+iy)I-A\right)^{-1} dy \tag{2.42}$$

$$a_n = \frac{1}{2\pi i} \int_{|w|=1} \frac{1}{w^{n+1}} \frac{2b}{1-w} \left((\sigma - b\frac{w+1}{w-1})I - A\right)^{-1} dw \tag{2.43}$$

$$a_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-in\theta} \frac{2b}{1-e^{i\theta}} \left((\sigma - b\frac{e^{i\theta}+1}{e^{i\theta}-1})I - A\right)^{-1} d\theta. \tag{2.44}$$

The coefficients are computed approximately from the sum

$$a_n \approx \frac{e^{-in\pi/(2M)}}{2M} \sum_{m=-M}^{M-1} e^{-in\theta_m} \frac{2b}{1-e^{i\theta_{m+1/2}}} \left((\sigma - b\frac{e^{i\theta_{m+1/2}}+1}{e^{i\theta_{m+1/2}}-1})I - A\right)^{-1}. \tag{2.45}$$

We choose $N = M$. With the substitution of a preliminary complex Schur factorization $A = QTQ^H$ and multiplication by an initial condition vector $\vec{u}(0)$

$$e^{At}\vec{u}(0) = \frac{Q}{2\pi i} \int_\Gamma e^{st}(sI-T)^{-1}(Q^H \vec{u}(0)) ds \tag{2.46}$$

we can define new coefficients

$$\tilde{a}_n \approx \frac{e^{-in\pi/(2M)}}{2M} \sum_{m=-M}^{M-1} e^{-in\theta_m} \frac{2b}{1-e^{i\theta_{m+1/2}}} \left((\sigma - b\frac{e^{i\theta_{m+1/2}}+1}{e^{i\theta_{m+1/2}}-1})I - T\right)^{-1} \tilde{u}(0) \tag{2.47}$$

where $\tilde{u}(0) = Q^H \vec{u}(0)$. This quantity is of course computed only once for each determination of the coefficients $\tilde{a}_n$. The inversions of $(sI-T)$ are now of a triangular matrix which can be performed efficiently by back substitution. Even though the number of operations to perform the Schur decomposition of a square $p \times p$ matrix is $O(p^3)$, the fact that back substitution is $O(p^2)$ more than makes up for the time that would have been spent on direct inversions. The only exception may be for the case when $A$ is initially sparse. More importantly for the proper selection of the parameters, a preliminary Schur decomposition has the advantage of providing the entire spectrum of $F(s)=(sI-A)^{-1}$. This follows from the fact that the eigenvalues of $A$, which are the poles of $F(s)$, are also the diagonal elements of the triangular matrix $T$.

Once all the coefficients have been computed the Clenshaw algorithm is used to perform the summation

$$e^{At}\vec{u}(0)\approx Qe^{(\sigma-b)t}\sum_{n=0}^{N-1}\tilde{a}_n L_n(2bt). \tag{2.48}$$

## 3. Parameter Selection Algorithms

Up to this point, we have shown that the theory and error analysis for the scalar Weeks method can be extended in a straight forward manner to vector or matrix functions. This analysis assumed however predetermined optimal values for $\sigma$ and $b$. The proper selection of these tuning parameters is crucial for the success of Weeks method and also the most difficult aspect of the computation.

Various algorithms have been proposed to automate the selection of $\sigma$ and $b$ for the scalar Weeks method. One could in principal use these for each element in a matrix valued function parameterized by time. For large matrices however this would become prohibitively expensive. A more sensible approach is to treat the matrix function as a single mathematical entity and determine optimal parameters based on its properties. In this paper, we thus extend three methods from the literature for scalar functions and add one of our own based on the concept of the pseudospectrum for the particular case of the matrix exponential. The three from the literature are Weeks original approach [36], a maximization of the radius of convergence (MinMax) approach [12], and a total error minimization proposed by Weideman [37]. One of the main issues besides accuracy and efficiency that these methods must address is the relationship between $\sigma$ and $b$. A straight forward two dimensional search of the $(\sigma,b)$ plane is virtually impossible for matrix functions due to sheer computational expense. Thus, one must consider how to relate $\sigma$ and $b$ and thereby reduce the search to only one of the parameters. For real time domain scalar functions, prescriptions for determining an optimal $b$ for a chosen $\sigma$ have been derived [12]. However, since we are interested in potentially complex time domain functions, we will not utilize these considerations. Finally, in keeping with the previous discussion, we describe the algorithms for a general function $F(s)$ and then focus upon the inverse Laplace transform of the resolvent matrix $F(s)=(sI-A)^{-1}$ for illustration.

**3.1. Weeks Original Approach.** In his original 1966 paper [36], Weeks offers semi-empirically determined expressions for $\sigma$ and $b$. His suggestions provide a starting point for an investigation of the optimal parameter selection. Using our present notation where

$$f(t)\approx e^{(\sigma-b)t}\sum_{n=0}^{N-1}a_n L_n(2bt) \tag{3.1}$$

Weeks suggests the values $\sigma=\max(0,\sigma_0+\frac{1}{t})$ and $b=\frac{N}{2t}$ where $N$ is the number of expansion coefficients. The selection for $\sigma$ is purely empirical with the additional constraint that $\sigma>0$. His justification for $b$ however comes from a consideration of the behavior of the weighted Laguerre polynomials $e^{-bt}L_n(2bt)$. For $2bt>4n$, the $n$th Laguerre polynomial decays exponentially and thus, Weeks assumes, would not be expected to approximate the solution well. Weeks thus considers only $2bt\leq 4N$ which is satisfied if we choose $b = \frac{N}{2t}$. Since this value for $b \propto N$ can be potentially large, we also look at the case when $b$ is fixed at $b = 1/2$. The value of $\sigma$ is still chosen according to Weeks prescription.

**3.2. MinMax Approach.** Weeks original suggestions for $\sigma$ and $b$ are somewhat unsatisfying in that they do not rely upon an analysis of the errors or knowledge of the spectrum of $F(s)$ beyond the location of the abscissa of convergence. An alternative is to neglect roundoff and discretization errors and attempt to minimize the truncation error from the Cauchy estimate [12]

$$\|\bar{T}\|_F \leq \frac{K(r)}{r^{N-1}(r-1)}. \tag{3.2}$$

We surmise that maximizing the radius of convergence

$$R = \min_{\lambda} \left| \frac{s-\sigma-b}{s-\sigma+b} \right| \tag{3.3}$$

where $\lambda$ are the singularities, will lead to a minimization of the truncation error.

For a general function $F(s)$ this type of maximization requires knowledge of at least the singularity which is mapped closest to the origin in the $w$ plane and thus defines the radius of convergence. For the case of the matrix exponential where $F(s) = (sI - A)^{-1}$, we have already seen that all of the singularities can be obtained from a complex Schur factorization. Thus, $R$ can be determined unambiguously for each $(\sigma, b)$ from a mapping of the eigenvalues of $A$ and maximized without the need to evaluate $F(s)$. In some simple cases it is also possible to analytically determine the critical singularity. An example is the matrix exponential of a matrix $A$ with purely imaginary eigenvalues $\lambda = iy$. In this case

$$R^2 = \min_{\lambda} \frac{(\sigma+b)^2 + y^2}{(\sigma-b)^2 + y^2}. \tag{3.4}$$

The function $\frac{(\sigma+b)^2+y^2}{(\sigma-b)^2+y^2}$ has a maximum at $y = 0$ and decreases monotonically as $|y| \to \infty$. Thus, for matrices with only imaginary eigenvalues, $R$ is determined by the eigenvalue with the largest absolute value.

Concerning the implementation, we have tested two approaches. First, we compute $R$ on a discretized $(\sigma, b)$ grid and then maximize over the computed values. This can be performed in MATLAB using only a few lines of vectorized code. A second alternative is to utilize a two-parameter search algorithm such as the Nelder-Mead simplex method in MATLAB's command *fminsearch*. One difficulty with any search algorithm is of course the choice of starting values which will avoid local minima. For the Laplace transform, a more important issue though is the need to constrain the values of $\sigma > \sigma_0$. Although a minimum may exist for $\sigma < \sigma_0$, this value for $\sigma$ leads to an unphysical time domain function.

**3.3. Weideman's Approach.** The previous two approaches have the distinct advantage of not requiring potentially expensive evaluations of $F(s)$. Neither however directly utilize an estimate for the total error and thus presumably may fail to yield values of $\sigma$ and $b$ which control both the truncation and roundoff errors. Weideman [37] has shown that the numerical inversion of a scalar function $F(s)$ using Weeks method can be performed robustly if one minimizes the total error (2.31). This direct approach is clearly the most expensive and a full two parameter search is probably prohibitively costly for most matrix-valued functions $F(s)$. This is particularly true since the truncation error is estimated by doubling the number of expansion coefficients. Fortunately, again for the matrix exponential, we can utilize the spectrum to define $b$ as the value which maximizes the radius of convergence $R(\sigma,b)$ for a given $\sigma$. This allows a one parameter minimization of equation (2.31) over $\sigma$.

**3.4. Pseudospectra Approach.** Our last approach is a compromise between minimization of the total error used in Weideman's algorithm and the blind maximization of the radius of convergence. We propose to relate $\sigma$ and $b$ by a maximization of the radius of convergence and then perform a minimization of the Cauchy estimate for the truncation error as a function of $\sigma$. We thus ignore the roundoff and discretization errors. This in turn requires us to compute

$$K(r) = \max_{\theta} \|G(w(\theta))\|_F \tag{3.5}$$

in the estimate

$$\|\bar{T}\|_F \leq \frac{K(r)}{r^{N-1}(r-1)} \tag{3.6}$$

For some functions $F(s)$ it may be possible to determine analytically estimates for $K(r)$ or compute the norm without directly computing $G(w)$ on circular contours. The matrix exponential provides an excellent example where this is true.

In the case of the matrix exponential

$$K(r) = \max_{|w|=r} \left|\frac{2b}{1-w}\right| \left\| \left( (\sigma - b\frac{w+1}{w-1})I - A \right)^{-1} \right\|_F . \tag{3.7}$$

It is a known fact of matrix norms that

$$\|A\|_F \leq \sqrt{p}\|A\|_2 \tag{3.8}$$

where $p$ is the dimension of the square matrix $A$. With this, we can replace the Frobenius norm

$$K(r) \leq \sqrt{p} \max_{|w|=r} \left|\frac{2b}{1-w}\right| \left\| \left( (\sigma - b\frac{w+1}{w-1})I - A \right)^{-1} \right\|_2 . \tag{3.9}$$

The constant $\sqrt{p}$ is not a function of the parameters $\sigma$ and $b$ nor the radius $r$ and is therefore not involved in the minimization of $K(r)$. For this reason, we drop this factor in the following discussion.

Our motivation for the change of norm is the efficiency with which the two-norm of the resolvent matrix can be estimated. The quantity is well known in numerical methods and is related to the concept of the pseudospectrum of a matrix [34, 35]

$$\Lambda_\epsilon(A) = [s \in \mathbb{C} | \gamma_{min}(sI - A) \leq \epsilon] . \tag{3.10}$$

where $\gamma_{min}$ is the smallest singular value of the matrix $(sI - A)$. In particular,

$$\|(sI-A)^{-1}\|_2^2 = \frac{1}{\gamma_{min}(sI-A)} \tag{3.11}$$

Computing the full singular value decomposition of $(sI - A)$ to determine the norm is hardly advantageous but there are many algorithms for computing the *minimum* singular value. The method used here to accelerate the computation of $\|(sI-A)^{-1}\|_2$ is the "triangularization+inverse Lanczos iteration" by Lui [21]. The concept behind this method is the fact that for $A=QTQ^H$

$$\|(sI-A)^{-1}\|_2^2=\lambda_{\min}\left[(sI-A)^H(sI-A)\right]=\lambda_{\min}\left[(sI-T)^H(sI-T)\right] \tag{3.12}$$

where $\lambda_{\min}$ is the smallest eigenvalue. For our implementation of the Weeks method, the Schur decomposition has already been performed and the number of inversions required for the power method is often less than required by doubling the number coefficients to estimate the tail of the Laguerre expansion.

One last important detail is a reasonable choice of $r$. We have tested two approaches. In one we choose $r=R$ if $R<1.1$, else $r=1.1$. This selection is arbitrary but leads to an overestimate of error rather than an underestimate.

A second algorithm, which avoids choosing a value for $r$ is to maximize $\|((\sigma+iy)I-A)^{-1}\|_2$ over $y\in\mathbb{R}$..

### 3.5. Practical Considerations for Parameter Selection.

**3.5.1. Parameter Plane Selection Region.** One issue to address is the proper selection of a portion of the $(\sigma,b)$ plane over which to scan for the optimal parameters. We have already established that $\sigma>\sigma_0$ and that $b>0$. This is insufficient however since we can not approach the singularities too closely in a numerical method without introducing large quantities and roundoff errors. On the other hand, one can not be too far away from the singularities or else the true value of the Laplace function may fall below the smallest representable number in our computations. In this case, one would be sampling noise. Also, if $1\ll\sigma_0<\sigma$, then the exponential in Weeks method can be large enough that the errors are enhanced to levels which destroy the feasibility of the method.

In our simulations we have found that the following range of parameters leads to robust calculations. We use $b\in(0,5+|\sigma_{min}|)$ where $\sigma\in(\sigma_{min},\sigma_{max})$. $\sigma_{min}$ and $\sigma_{max}$ are chosen according to the prescription

- if $|\sigma_0|<0.1$
  - $\sigma_{min}=1$
  - $\sigma_{max}=20$
- else
  - if $1<|\sigma_0|/20$
    - if $\sigma_0>0$
      - $\sigma_{min}=1.05\sigma_0$
      - $\sigma_{max}=10\sigma_0$
    - else
      - $\sigma_{min}=\sigma_0+|\sigma_0|/20$
      - $\sigma_{max}=\sigma_0+10|\sigma_0|$
  - else
    - if $\sigma_0>0$
      - $\sigma_{min}=\sigma_0+1$
      - $\sigma_{max}=10\sigma_0+1$
    - else
      - $\sigma_{min}=\sigma_0+1$
      - $\sigma_{max}=\sigma_0+1+10|\sigma_0|$.

The only exception is for the MinMax full two parameter search. In this case, we use MATLAB's function *fminsearch* which requires an initial guess for $\sigma$ and $b$. Here we use $b = 2.5 + |\sigma_{min}/2|$ and $\sigma = \sigma_{min} + (k|\sigma_{max} - \sigma_{min}|)/10)$, where $k = 1, 2, \ldots$ and is allowed to vary so as to restart *fminsearch* until an optimal $\sigma$ is reached which is greater than $\sigma_0$.

**3.5.2. Matrix Exponential Scaling.** Another issue to address is scaling. Many of the practical algorithms for computing the matrix exponential reply upon the property $e^{At} = (e^{\frac{At}{2^n}})^{2^n}$ [14, 23]. The expectation is that the errors introduced by computing the scaled matrix, followed by repeated squaring will be less than direct application of the algorithm to the full matrix. The effect of the scaling and squaring on the Weeks method is also analyzed in this paper.

With respect to the practical implementation of the scaling, one has a variety of options. First, since the coefficients $a_n$ are time independent, the scaled exponential for any scaling $\kappa \in \mathbb{C}$ can be computed with the coefficients for the full matrix $A$

$$e^{(\kappa A)t} = e^{A(\kappa t)} \approx e^{\sigma(\kappa t)} \sum_{n=0}^{N-1} e^{-b(\kappa t)} a_n(\sigma, b, A) L_n(2b(\kappa t)). \tag{3.13}$$

This approach is particularly attractive for exponential time differencing with an arbitrary time step.

Equivalently, one can directly multiply the matrix $A$ and compute coefficients using the selection algorithms for the scaled matrix

$$e^{(\kappa A)t} \approx e^{\sigma t} \sum_{n=0}^{N-1} e^{-bt} a_n(\sigma, b, \kappa A) L_n(2bt). \tag{3.14}$$

The $\sigma$ and $b$ in this sum are chosen with respect to $\kappa A$ and are thus typically not the same as in equation (3.13).

One aspect of the coefficients $a_n$ which applies to both these sums is their homogeneity

$$a_n(\sigma, b, A) = a_n(\kappa\sigma, \kappa b, \kappa A). \tag{3.15}$$

Thus, in principle, one could replace the coefficients $a_n(\sigma, b, A)$ with any scaled equivalent. There is hence an infinitude of possible scalings. An interesting topic which we have not pursued in detail is a procedure for the automated selection of the scaling parameter $\kappa$. Instead, in this paper, we have chosen to not generally use squaring/scaling. In the cases where we do scale, we have opted to multiply the matrix $A$ by $\kappa$ and perform the sum using equation (3.14). In the PDE examples, choosing a smaller step size for $e^{A\delta t}$ and propagating with this smaller value clearly has the same effect as scaling/squaring.

## 4. Numerical Examples

In this section of the paper, we present examples to illustrate Weeks method and elucidate the behavior of the parameter selection algorithms. To begin, we compute the full matrix exponential for two pathological matrices from MATLAB's gallery, the Hanowa matrix and the Pei matrix. Both have a spectrum which challenges the accuracy and stability of the Weeks method. Although useful as test cases, they are unlikely however to arise in a practical computation. We therefore also study the exponential for matrices from the advection-diffusion equation and the scalar nonparaxial beam propagation equation from computational photonics [42].

The computation of the matrix exponential for the large matrices in the numerical simulation of a PDE requires special consideration. Typically, even for sparse matrices, the matrix exponential is dense. In two or more dimensions the memory requirements demanded by this dense matrix are often untenable. Since we wish to compare the solution from Weeks method with one computed using the explicit matrix exponential from MATLAB's function *expm1*, we have limited ourselves to one dimension. Additionally, we compute the vector $e^{A\delta t}\vec{u}(0)$ with each step $\delta t$ directly using Weeks method instead of explicitly forming the matrix exponential.

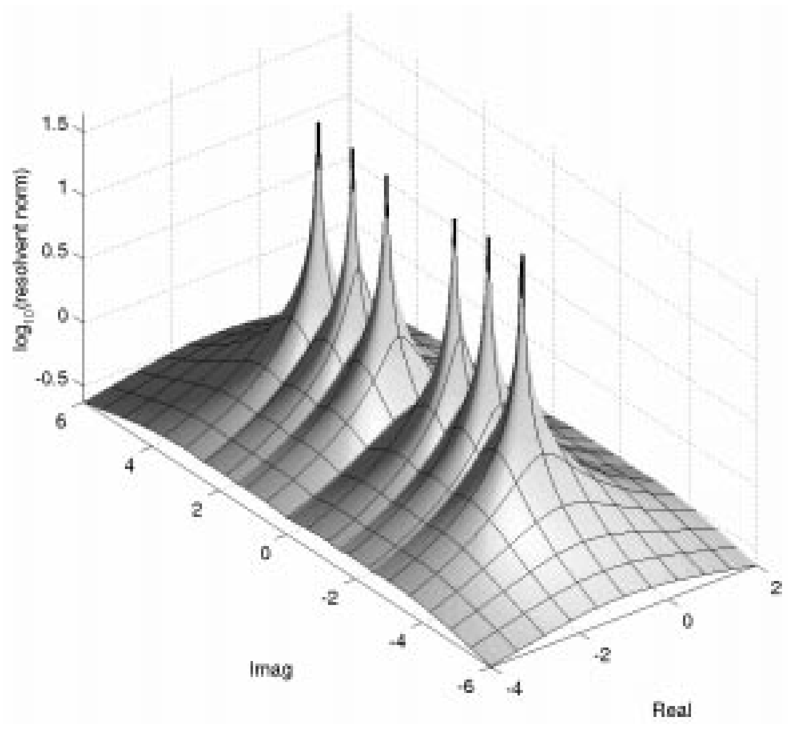

FIG. 4.1. *Hanowa matrix resolvent norm*

**4.1. Pathological Matrices.** One of MATLAB's more useful features is a gallery of test matrices. Included in this collection are the Hanowa and the Pei matrix. In both cases we have considered a square $6\times 6$ matrix. The spectrum of the Pei matrix is particularly interesting due to its degeneracy; all the eigenvalues except one have a value of 1, the other is equal to $N+1$ where $N$ is the dimension of the square Pei matrix. The Hanowa matrix in contrast has eigenvalues with negative real part, $\Re(\lambda)=-1$, and nonzero imaginary parts. Fig. 4.1 and 4.2 are plots of the two-norm of the resolvent matrix $\|(sI-A)^{-1}\|_2$ generated by the Eigtool program for MATLAB [39]. The location of the singularities at the eigenvalues is evident.

Fig. 4.3 and 4.4 depict the actual measured maximum relative error of the nonzero elements in the Hanowa and Pei matrix exponentials as a function of $\sigma$ and $b$ for 32 coefficients. The 'true solution' is defined as the matrix exponential given by the MATLAB function *expm1*. For normal matrices, the Padé/Scaling approach utilized by MATLAB is typically reliable [14]. To note from these figures is the large basin in the $(\sigma,b)$ plane where the error approaches the precision of the computations. The width of the basin is proportional to the number of coefficients. The rate of

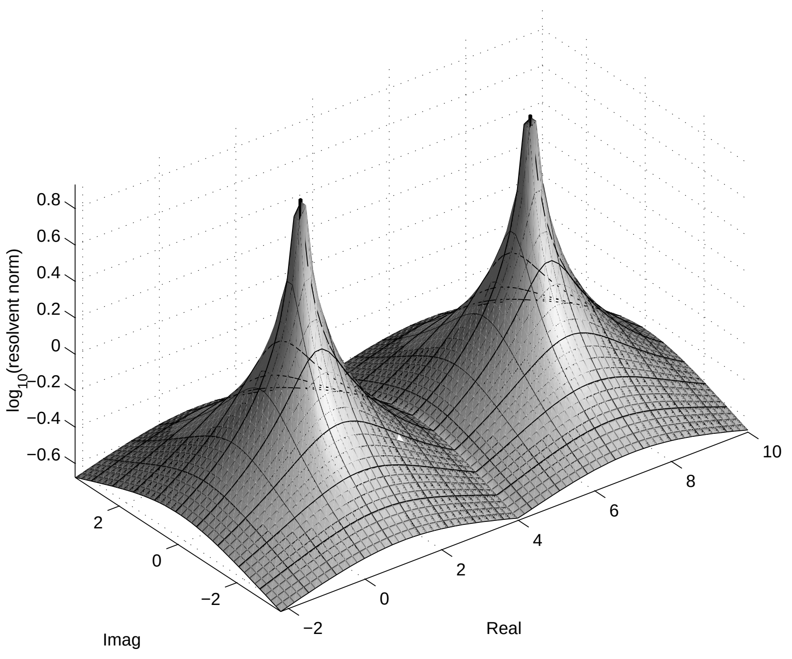

FIG. 4.2. *Pei matrix resolvent norm*

convergence as a function of the number of coefficients is given in Table 4.1 and 4.2.

Also plotted in Fig. 4.3 and 4.4 are the locations of the parameters obtained from the seven selection algorithms without scaling the matrix. The corresponding values are given in Table 4.3 and 4.4. The scaling referred to in the table is the value of $n$ in $(e^{A/n})^n$. The computation time reported is the time required by the FFT algorithm for the coefficients and includes the time to compute an estimate of the error. From this information it is clear that a squaring and scaling algorithm is a highly effective acceleration technique for Weeks method. The increased accuracy greatly outweighs the need to square the matrix. Also interesting is the agreement between Weideman's method and the pseudospectra methods on the location of the optimal parameters.

Shown in Table 4.5 and 4.6 is the accuracy of the absolute error estimates derived in section (2.2). The parameters are computed using the Weideman method. The computation times required by the midpoint and FFT methods are also contrasted. As expected, the FFT approach requires considerably less time for a large number of coefficients than the direct midpoint summation. Thus, unless otherwise stated, all simulations in this paper are performed using the FFT algorithm.

Finally, from the comparison of the computation times required by each of the selection algorithms, Table 4.7 and 4.8, one sees that while Weideman's method is the most robust it is also the most expensive. This is particularly problematic for the large numbers of inversions which one is likely to use in a practical computation. The pseudospectra approach based on the minimum singular value has a slower rate of convergence than Weideman's method but yields the same accuracy for the large numbers of coefficients required for a spectrally accurate computation. It is also considerably faster. Surprisingly, if the errors do not need to be less than $10^{-10}$, then the simple blind maximization of the radius of convergence is surprisingly well

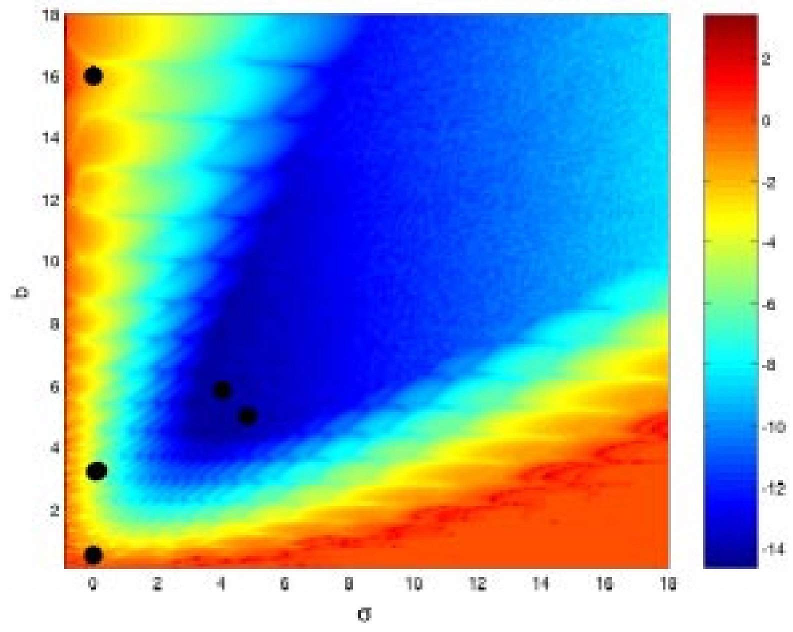

FIG. 4.3. *Hanowa matrix measured relative error*

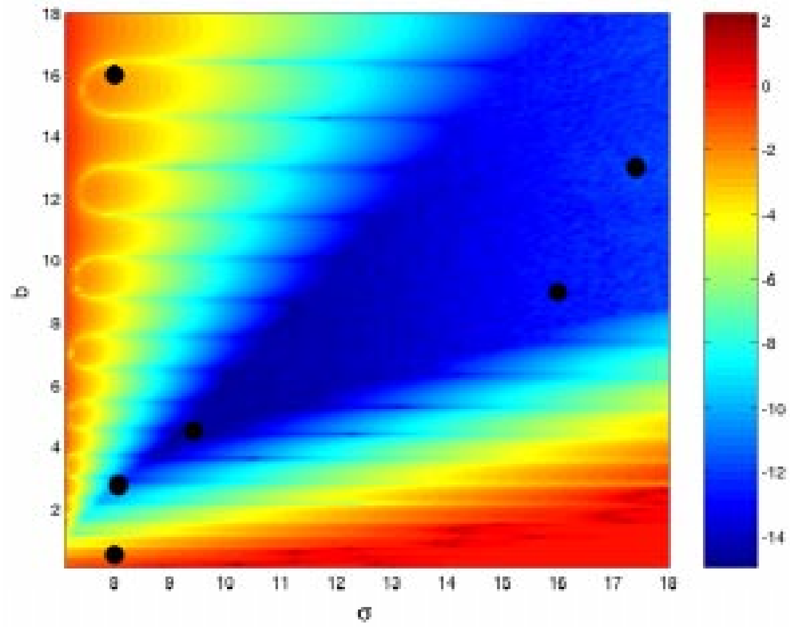

FIG. 4.4. *Pei matrix measured relative error*

suited. It provides reasonable accuracy and efficiency. The intelligent search in the $(\sigma,b)$ plane using MATLAB's *fminsearch* function is not robust due to the fact that the algorithm is not limited to values of $\sigma > \sigma_0$ and $b > 0$. Rather, a constrained optimization algorithm is required.

**4.2. Advection-Diffusion Equation: Dense Matrices.** The advection-diffusion equation

$$\frac{\partial u}{\partial t} = v\frac{\partial u}{\partial x} + d\frac{\partial^2 u}{\partial x^2} \tag{4.1}$$

with small diffusion $d << 1$ is notoriously difficult to simulate accurately. Countless algorithms have been proposed beyond standard finite differences [6, 28, 30, 32]. The numerical Laplace transform approach using the Fourier series algorithm is one of the more popular for long time, small diffusion, hydrological and geological simulations of this equation. This method however adds artificial diffusion due to the nonlinear series acceleration algorithm and is not applicable to the purely advective limit. Weeks method does not suffer from this restriction and can thus be used to propagate conservative equations spectrally accurately.

With periodic boundaries on the domain $x \in [0, 2\pi)$, the advection-diffusion equation (4.1) can be diagonalized with a Fourier spatial transform. However, to demonstrate our algorithm we use the method of lines to establish a system of ODEs

$$\frac{d\vec{u}}{dt} = vD\vec{u} + dD^2\vec{u} \tag{4.2}$$

where $D$ is the differentiation matrix. Since we use a spectrally accurate propagator in time it is sensible to use the same for the differentiation matrix. For this reason, we choose the Toeplitz Fourier $p \times p$ differentiation matrix [35] with initial column

$$D_j = \begin{cases} 0 & j = 0(mod p) \\ \frac{(-1)^j}{2}\cot\left(\frac{j\delta x}{2}\right) & j \neq 0(mod p). \end{cases} \tag{4.3}$$

This matrix is dense and thus we will use a preliminary Schur decomposition. Since the solution must be smooth to utilize the spectral accuracy of the differentiation matrix, we assume an initial condition $\vec{u}(x, t = 0) = \sin(x - \pi)$ for purely advective simulations and an initial condition $\vec{u}(x, t = 0) = 1 + \sin(x - \pi)$ for the diffusion simulations. Both of these functions have a single period over the simulation domain $[0, 2\pi)$. Since we use 16 points in the differentiation matrix, the resolution of the initial condition is 16 points per wavelength. The advection speed is 1, while the diffusion coefficient is $10^{-3}$.

The results of our simulations are shown in Table 4.9,4.10,4.11,4.12 for the purely advective and 4.13 for the purely diffusive case. The error reported is the absolute error of the Weeks method solution compared to the solution computed using a Fourier series. The Fourier series contains 256 elements. It is spectrally accurate in the case of the diffusion equation and exact for the advection equation. In Fig. 4.5 and 4.7 are the errors using the Weideman algorithm for the parameter selection.

The apparent broadening of the width of the line for the advection case is due to the growth of small oscillations in the error. The difference in the two-norm of the initial condition and the solution at time $t$

$$\eta = \left| \|\vec{u}(t)\|_2 - \|\vec{u}(0)\|_2 \right| \tag{4.4}$$

for the advection equation is shown in Fig. 4.6. The simulation time reported in these figures and in the tables includes the contributions from the error estimate

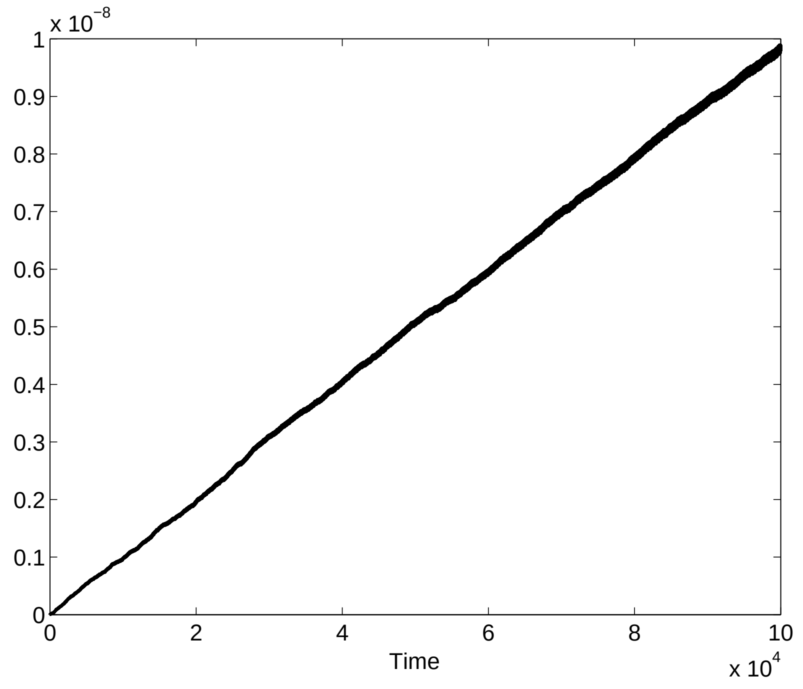

FIG. 4.5. *Advection equation absolute error. 128 coefficients $\delta t = 5$ $\sigma = 13.00$ $b = 17.94$. Computation time=*2.29 *hours.*

using equation (2.31), the storage of the Weeks method solution at each time step, and the time to compute the solution using the matrix exponential from MATLAB's *expm1*. Streamlining the calculations to involve only the propagation by the Weeks method and not store the solution at each time step yields simulation times of 699 seconds for the advection equation and 6.464 seconds for the diffusion equation. An analysis of the distribution of the computation time reveals that the majority of the time is spent in the inversion of the triangular matrix in the calculation of the expansion coefficients. In comparison, propagating the identical problem using the matrix exponential from MATLAB results in simulation times of 14.1 seconds and 0.0908 seconds for the advection and diffusion equations, respectively.

The dependence of the advection equation on the time step for Weeks method can be easily understood from the considerations of the radius of convergence $R$ in section (3.2). In the special case that $\sigma = b$, a reasonable assumption for a matrix with purely imaginary eigenvalues, $R^2 = 1 + \left(\frac{2\sigma}{y_{max}}\right)^2$. For the differentiation matrix from the center differences approximation to $\frac{d}{dx}$, the largest eigenvalue is proportional to $\frac{\delta t}{\delta x}$. Solving for $\delta t$, one obtains a stability-like condition

$$\delta t \propto \frac{\sigma \delta x}{\sqrt{(R^2 - 1)}} \tag{4.5}$$

From this equation, we can see that for a larger radius of convergence $R$ and presumably smaller error and for finer spatial grids $\delta x$, $\delta t$ must also be smaller.

Table 4.1. *Relative Error: Hanowa 6*

| N | Weeks:N | Weeks:b Half | MinMax:Grid | MinMax:Search | Pseudo:rmin | Pseudo:Y | Weideman |
|---|---|---|---|---|---|---|---|
| 2 | 1.00 | 1.49 | 27.28 | – | 2.44 | 2.08 | 2.08 |
| 4 | 0.479 | 0.825 | 1.42 | – | 0.313 | 0.330 | 0.660 |
| 8 | 0.154 | 2.59 | $7.96 \cdot 10^{-3}$ | – | 0.115 | 0.133 | $1.50 \cdot 10^{-3}$ |
| 16 | 0.0137 | 0.133 | $2.76 \cdot 10^{-7}$ | – | $2.33 \cdot 10^{-3}$ | $2.85 \cdot 10^{-3}$ | $8.22 \cdot 10^{-11}$ |
| 32 | 0.0215 | 0.117 | $1.49 \cdot 10^{-14}$ | – | $3.15 \cdot 10^{-6}$ | $5.31 \cdot 10^{-6}$ | $1.10 \cdot 10^{-14}$ |
| 64 | 0.0154 | $2.46 \cdot 10^{-3}$ | $1.54 \cdot 10^{-14}$ | – | $1.92 \cdot 10^{-11}$ | $6.84 \cdot 10^{-11}$ | $5.58 \cdot 10^{-15}$ |
| 128 | $5.24 \cdot 10^{-3}$ | $5.90 \cdot 10^{-6}$ | $1.65 \cdot 10^{-14}$ | – | $4.88 \cdot 10^{-15}$ | $5.03 \cdot 10^{-15}$ | $4.89 \cdot 10^{-15}$ |

Table 4.2. *Relative Error: Pei 6*

| N | Weeks:N | Weeks:b Half | MinMax:Grid | MinMax:Search | Pseudo:rmin | Pseudo:Y | Weideman |
|---|---|---|---|---|---|---|---|
| 2 | 0.222 | 0.767 | 1.31 | 11.99 | 0.724 | 0.724 | 0.714 |
| 4 | 0.0145 | 0.155 | 0.548 | 13.80 | 0.102 | 0.102 | 0.112 |
| 8 | $5.43 \cdot 10^{-3}$ | 0.519 | 9.45 | 0.0839 | $4.09 \cdot 10^{-2}$ | $4.09 \cdot 10^{-2}$ | $2.98 \cdot 10^{-4}$ |
| 16 | $8.75 \cdot 10^{-4}$ | 0.0722 | $2.33 \cdot 10^{-11}$ | $7.50 \cdot 10^{-7}$ | $4.42 \cdot 10^{-6}$ | $4.41 \cdot 10^{-6}$ | $9.99 \cdot 10^{-12}$ |
| 32 | $3.98 \cdot 10^{-3}$ | $7.87 \cdot 10^{-3}$ | $4.28 \cdot 10^{-13}$ | $4.73 \cdot 10^{-13}$ | $6.56 \cdot 10^{-12}$ | $6.52 \cdot 10^{-12}$ | $3.07 \cdot 10^{-15}$ |
| 64 | $2.57 \cdot 10^{-3}$ | $2.11 \cdot 10^{-4}$ | $7.74 \cdot 10^{-13}$ | $3.02 \cdot 10^{-13}$ | $3.27 \cdot 10^{-15}$ | $2.65 \cdot 10^{-15}$ | $2.65 \cdot 10^{-15}$ |
| 128 | $2.19 \cdot 10^{-3}$ | $1.97 \cdot 10^{-8}$ | $7.78 \cdot 10^{-13}$ | $2.05 \cdot 10^{-13}$ | $2.96 \cdot 10^{-15}$ | $2.77 \cdot 10^{-15}$ | $2.43 \cdot 10^{-15}$ |

Table 4.3. *Weeks Method Parameters: Hanowa 6 for 32 Coefficients*

| Scaling | | Weeks:N | Weeks:b Half | MinMax:Grid | MinMax:Search | Pseudo:rmin | Pseudo:Y | Weideman |
|---|---|---|---|---|---|---|---|---|
| 1 | $\sigma$ | 0 | 0 | 4.80 | – | 0.134 | 0.0665 | 4.03 |
| | $b$ | 16 | 1/2 | 5.00 | – | 3.21 | 3.18 | 5.84 |
| | Time (sec) | 0.0330 | 0.0328 | 0.0656 | – | 0.454 | 0.393 | 0.474 |
| | Relative Error | 0.0215 | 0.117 | $1.49\cdot10^{-14}$ | Failed | $3.15\cdot10^{-6}$ | $5.31\cdot10^{-6}$ | $1.10\cdot10^{-15}$ |
| 2 | $\sigma$ | 1/2 | 1/2 | 5.20 | – | 0.560 | 0.554 | 1.80 |
| | $b$ | 16 | 1/2 | 5.50 | – | 1.85 | 1.84 | 2.76 |
| | Time (sec) | 0.0580 | 0.0426 | 0.0625 | – | 0.452 | 0.414 | 0.424 |
| | Relative Error | 0.0575 | $4.221\cdot10^{-5}$ | $1.36\cdot10^{-14}$ | Failed | $2.37\cdot10^{-9}$ | $2.51\cdot10^{-9}$ | $5.18\cdot10^{-15}$ |
| 4 | $\sigma$ | 3/4 | 3/4 | 3.25 | 0.776 | 0.803 | 0.803 | 0.897 |
| | $b$ | 16 | 1/2 | 3.60 | 1.13 | 1.28 | 1.28 | 1.36 |
| | Time (sec) | 0.0418 | 0.0430 | 0.0512 | 0.0727 | 0.380 | 0.349 | 0.436 |
| | Relative Error | 0.106 | $1.406\cdot10^{-8}$ | $5.74\cdot10^{-15}$ | $3.20\cdot10^{-15}$ | $8.90\cdot10^{-15}$ | $8.90\cdot10^{-15}$ | $7.25\cdot10^{-15}$ |
| 8 | $\sigma$ | 7/8 | 7/8 | 2.08 | 0.977 | 0.932 | 0.944 | 0.945 |
| | $b$ | 16 | 1/2 | 2.20 | 1.15 | 1.10 | 1.10 | 1.10 |
| | Time (sec) | 0.0417 | 0.0522 | 0.0473 | 0.071 | 0.343 | 0.317 | 0.382 |
| | Relative Error | 0.205 | $1.436\cdot10^{-12}$ | $4.93\cdot10^{-15}$ | $4.57\cdot10^{-15}$ | $1.29\cdot10^{-14}$ | $6.02\cdot10^{-15}$ | $6.02\cdot10^{-15}$ |
| 16 | $\sigma$ | 15/16 | 15/16 | 5.90 | – | 1.04 | 1.04 | 1.04 |
| | $b$ | 16 | 1/2 | 6.00 | – | 1.11 | 1.10 | 1.10 |
| | Time (sec) | 0.0418 | 0.0439 | 0.127 | – | 0.758 | 0.656 | 0.883 |
| | Relative Error | 0.178 | $8.10\cdot10^{-14}$ | $7.68\cdot10^{-14}$ | Failed | $8.80\cdot10^{-15}$ | $1.38\cdot10^{-14}$ | $1.24\cdot10^{-14}$ |

TABLE 4.4. *Weeks Method Parameters: Pei 6 for 32 Coefficients*

| Scaling | | Weeks:N | Weeks:b Half | MinMax:Grid | MinMax:Search | Pseudo:rmin | Pseudo:Y | Weideman |
|---|---|---|---|---|---|---|---|---|
| 1 | $\sigma$ | 8 | 8 | 17.40 | 15.99 | 8.07 | 8.07 | 9.42 |
| | $b$ | 16 | 1/2 | 13.00 | 8.99 | 2.75 | 2.75 | 4.52 |
| | Time (sec) | 0.0356 | 0.0340 | 0.288 | 0.0561 | 0.882 | 0.995 | 0.908 |
| | Relative Error | $3.98 \cdot 10^{-3}$ | $7.87 \cdot 10^{-3}$ | $4.28 \cdot 10^{-13}$ | $4.73 \cdot 10^{-13}$ | $6.56 \cdot 10^{-12}$ | $6.52 \cdot 10^{-12}$ | $3.07 \cdot 10^{-15}$ |
| 2 | $\sigma$ | 9/2 | 9/2 | 11.6 | 4.50 | 4.56 | 4.56 | 4.78 |
| | $b$ | 16 | 1/2 | 9.50 | 36.00 | 2.09 | 2.09 | 2.34 |
| | Time (sec) | 0.0421 | 0.0422 | 0.139 | 0.070 | 0.854 | 0.765 | 1.04 |
| | Relative Error | $7.98 \cdot 10^{-3}$ | $8.44 \cdot 10^{-6}$ | $1.75 \cdot 10^{-14}$ | $4.83 \cdot 10^{-15}$ | $3.23 \cdot 10^{-15}$ | $3.12 \cdot 10^{-15}$ | $3.07 \cdot 10^{-15}$ |
| 4 | $\sigma$ | 11/4 | 11/4 | 8.75 | 4.83 | 2.81 | 2.81 | 2.79 |
| | $b$ | 16 | 1/2 | 7.70 | 3.08 | 1.65 | 1.65 | 1.64 |
| | Time (sec) | 0.0424 | 0.0423 | 0.0849 | 0.071 | 0.662 | 0.593 | 0.863 |
| | Relative Error | 0.016 | $2.70 \cdot 10^{-8}$ | $1.20 \cdot 10^{-14}$ | $2.68 \cdot 10^{-15}$ | $2.15 \cdot 10^{-15}$ | $2.50 \cdot 10^{-15}$ | $2.49 \cdot 10^{-15}$ |
| 8 | $\sigma$ | 15/8 | 15/8 | 7.28 | 2.60 | 1.93 | 1.93 | 1.94 |
| | $b$ | 16 | 1/2 | 6.80 | 2.47 | 1.37 | 1.37 | 1.37 |
| | Time (sec) | 0.0422 | 0.0425 | 0.060 | 0.072 | 0.498 | 0.452 | 0.557 |
| | Relative Error | 0.0323 | $1.72 \cdot 10^{-10}$ | $5.81 \cdot 10^{-14}$ | $2.49 \cdot 10^{-15}$ | $2.92 \cdot 10^{-15}$ | $2.49 \cdot 10^{-15}$ | $2.92 \cdot 10^{-15}$ |
| 16 | $\sigma$ | 23/16 | 23/16 | 5.34 | 1.86 | 1.49 | 1.49 | 1.50 |
| | $b$ | 16 | 1/2 | 5.10 | 1.80 | 1.22 | 1.22 | 1.23 |
| | Time (sec) | 0.042 | 0.0424 | 0.050 | 0.073 | 0.426 | 0.385 | 0.469 |
| | Relative Error | 0.0656 | $1.17 \cdot 10^{-12}$ | $2.39 \cdot 10^{-14}$ | $8.47 \cdot 10^{-15}$ | $8.45 \cdot 10^{-15}$ | $2.29 \cdot 10^{-15}$ | $4.83 \cdot 10^{-15}$ |

TABLE 4.5. *Absolute Error Estimate: Hanowa 6*

| N | Estimate | | Measured | | Computation Time (sec) | |
|---|---|---|---|---|---|---|
| | FFT | Midpoint | FFT | Midpoint | FFT | Midpoint |
| 2 | 2.67 | 2.67 | 0.305 | 0.305 | 0.239 | 0.216 |
| 4 | 0.903 | 0.903 | 0.0631 | 0.0631 | 0.159 | 0.184 |
| 8 | $8.97 \cdot 10^{-3}$ | $8.97 \cdot 10^{-3}$ | $2.01 \cdot 10^{-4}$ | $2.01 \cdot 10^{-4}$ | 0.235 | 0.390 |
| 16 | $2.60 \cdot 10^{-9}$ | $2.60 \cdot 10^{-9}$ | $1.86 \cdot 10^{-11}$ | $1.57 \cdot 10^{-11}$ | 0.310 | 1.02 |
| 32 | $2.53 \cdot 10^{-14}$ | $1.94 \cdot 10^{-13}$ | $1.28 \cdot 10^{-15}$ | $2.99 \cdot 10^{-15}$ | 0.474 | 3.35 |
| 64 | $2.44 \cdot 10^{-15}$ | $4.52 \cdot 10^{-14}$ | $1.78 \cdot 10^{-15}$ | $2.41 \cdot 10^{-15}$ | 1.36 | 12.51 |
| 128 | $1.39 \cdot 10^{-15}$ | $4.71 \cdot 10^{-14}$ | $1.78 \cdot 10^{-15}$ | $2.06 \cdot 10^{-15}$ | 6.35 | 52.38 |

TABLE 4.6. *Absolute Error Estimate: Pei 6*

| N | Estimate | | Measured | | Computation Time (sec) | |
|---|---|---|---|---|---|---|
| | FFT | Midpoint | FFT | Midpoint | FFT | Midpoint |
| 2 | $2.93 \cdot 10^{3}$ | $2.93 \cdot 10^{3}$ | $1.32 \cdot 10^{2}$ | $1.32 \cdot 10^{2}$ | 0.433 | 0.443 |
| 4 | $5.71 \cdot 10^{2}$ | $5.71 \cdot 10^{2}$ | 20.78 | 20.78 | 0.441 | 0.517 |
| 8 | 3.70 | 3.70 | 0.055 | 0.055 | 0.420 | 0.706 |
| 16 | $5.24 \cdot 10^{-7}$ | $5.72 \cdot 10^{-7}$ | $1.85 \cdot 10^{-9}$ | $8.24 \cdot 10^{-9}$ | 0.408 | 1.49 |
| 32 | $8.29 \cdot 10^{-12}$ | $6.35 \cdot 10^{-11}$ | $5.69 \cdot 10^{-13}$ | $1.34 \cdot 10^{-12}$ | 0.908 | 7.13 |
| 64 | $2.11 \cdot 10^{-12}$ | $4.46 \cdot 10^{-11}$ | $4.83 \cdot 10^{-13}$ | $6.49 \cdot 10^{-13}$ | 2.77 | 26.71 |
| 128 | $1.82 \cdot 10^{-12}$ | $6.65 \cdot 10^{-11}$ | $6.25 \cdot 10^{-13}$ | $6.54 \cdot 10^{-13}$ | 13.58 | 104.34 |

Table 4.7. *Computation Time (sec): Hanowa 6*

| N | Weeks:N | Weeks:b Half | MinMax:Grid | MinMax:Search | Pseudo:rmin | Pseudo:Y | Weideman |
|---|---|---|---|---|---|---|---|
| 2 | 0.0128 | 0.0131 | 0.0510 | – | 0.469 | 0.418 | 0.239 |
| 4 | $5.15 \cdot 10^{-3}$ | $5.34 \cdot 10^{-3}$ | 0.0378 | – | 0.400 | 0.366 | 0.159 |
| 8 | $8.45 \cdot 10^{-3}$ | $8.39 \cdot 10^{-3}$ | 0.0410 | – | 0.376 | 0.371 | 0.235 |
| 16 | 0.0154 | 0.0153 | 0.0482 | – | 0.407 | 0.376 | 0.310 |
| 32 | 0.0330 | 0.0328 | 0.0656 | – | 0.454 | 0.393 | 0.474 |
| 64 | 0.109 | 0.105 | 0.138 | – | 0.500 | 0.467 | 1.367 |
| 128 | 0.480 | 0.488 | 0.514 | – | 0.888 | 0.841 | 6.35 |

Table 4.8. *Computation Time (sec): Pei 6*

| N | Weeks:N | Weeks:b Half | MinMax:Grid | MinMax:Search | Pseudo:rmin | Pseudo:Y | Weideman |
|---|---|---|---|---|---|---|---|
| 2 | $3.80 \cdot 10^{-3}$ | $3.98 \cdot 10^{-3}$ | 0.290 | 0.0270 | 0.854 | 0.759 | 0.433 |
| 4 | $5.71 \cdot 10^{-3}$ | $5.48 \cdot 10^{-3}$ | 0.260 | 0.0290 | 0.861 | 0.756 | 0.441 |
| 8 | $8.66 \cdot 10^{-3}$ | $9.16 \cdot 10^{-3}$ | 0.263 | 0.0319 | 0.882 | 0.769 | 0.420 |
| 16 | 0.0161 | 0.0165 | 0.270 | 0.0428 | 0.874 | 0.783 | 0.408 |
| 32 | 0.0356 | 0.0340 | 0.288 | 0.0561 | 0.882 | 0.995 | 0.908 |
| 64 | 0.114 | 0.125 | 0.362 | 0.128 | 0.961 | 1.62 | 2.77 |
| 128 | 0.510 | 0.488 | 0.740 | 0.504 | 1.39 | 1.73 | 13.58 |

TABLE 4.9. *Advection Error: $N = 32$, Simulation Time = 100, $\delta T = 10$*

| | Weeks:N | Weeks:b Half | MinMax:Grid | MinMax:Search | Pseudo:rmin | Pseudo:Y | Weideman |
|---|---|---|---|---|---|---|---|
| $\sigma$ | 1 | 1 | 20 | 2.99 | 8.16 | 1.06 | 1.24 |
| $b$ | 16 | 1/2 | 10 | 2.99 | 13.10 | 6.01 | 6.19 |
| Time (sec) | 8.44 | 11.21 | 9.45 | 8.42 | 11.84 | 12.93 | 9.52 |
| Absolute Error | 0.178 | 0.990 | Failed | 0.222 | 15.71 | 0.256 | 0.148 |

TABLE 4.10. *Advection Error: $N = 32$, Simulation Time = 100, $\delta T = 1$*

| | Weeks:N | Weeks:b Half | MinMax:Grid | MinMax:Search | Pseudo:rmin | Pseudo:Y | Weideman |
|---|---|---|---|---|---|---|---|
| $\sigma$ | 1 | 1 | 12.2 | 2.99 | 1.62 | 1.05 | 9.78 |
| $b$ | 16 | 1/2 | 10 | 2.99 | 6.58 | 6.01 | 12.04 |
| Time (sec) | 13.78 | 15.10 | 16.48 | 16.58 | 16.67 | 17.79 | 14.81 |
| Absolute Error | 0.347 | $5.14 \cdot 10^{-6}$ | $3.21 \cdot 10^{-11}$ | $3.77 \cdot 10^{-14}$ | $3.71 \cdot 10^{-6}$ | $6.46 \cdot 10^{-4}$ | $2.56 \cdot 10^{-12}$ |

TABLE 4.11. *Advection Error: $N = 128$, Simulation Time = 100, $\delta T = 10$*

| | Weeks:N | Weeks:b Half | MinMax:Grid | MinMax:Search | Pseudo:rmin | Pseudo:Y | Weideman |
|---|---|---|---|---|---|---|---|
| $\sigma$ | 1 | 1 | 20 | 2.99 | 12.47 | 1.06 | 19.94 |
| $b$ | 64 | 1/2 | 10 | 2.99 | 17.41 | 6.01 | 24.90 |
| Time (sec) | 12.62 | 6.29 | 8.67 | 7.11 | 14.82 | 11.29 | 9.23 |
| Absolute Error | 0.0244 | 0.979 | Failed | $3.80 \cdot 10^{-8}$ | $8.32 \cdot 10^{-12}$ | $9.05 \cdot 10^{-6}$ | $7.02 \cdot 10^{-9}$ |

TABLE 4.12. *Advection Error: $N = 128$, Simulation Time $= 100$, $\delta T = 1$*

| | Weeks:N | Weeks:b Half | MinMax:Grid | MinMax:Search | Pseudo:rmin | Pseudo:Y | Weideman |
|---|---|---|---|---|---|---|---|
| $\sigma$ | 1 | 1 | 12.2 | 2.99 | 1.56 | 1.05 | 2.04 |
| $b$ | 64 | 1/2 | 10 | 2.99 | 6.52 | 6.01 | 7.01 |
| Time (sec) | 28.14 | 24.64 | 28.10 | 27.15 | 26.71 | 28.27 | 27.95 |
| Absolute Error | 0.216 | $4.11 \cdot 10^{-13}$ | $1.65 \cdot 10^{-11}$ | $3.78 \cdot 10^{-13}$ | $3.69 \cdot 10^{-13}$ | $3.69 \cdot 10^{-13}$ | $3.71 \cdot 10^{-13}$ |

TABLE 4.13. *Diffusion Error: $N = 32$, Simulation Time $= 1000$, $\delta T = 10$*

| | Weeks:N | Weeks:b Half | MinMax:Grid | MinMax:Search | Pseudo:rmin | Pseudo:Y | Weideman |
|---|---|---|---|---|---|---|---|
| $\sigma$ | 1 | 1 | 9.8 | 2.90 | 1.04 | 1.05 | 1.06 |
| $b$ | 16 | 1/2 | 10 | 3.15 | 1.27 | 1.27 | 1.28 |
| Time (sec) | 14.21 | 15.75 | 16.24 | 20.38 | 16.28 | 18.90 | 18.20 |
| Relative Error | 0.496 | $7.06 \cdot 10^{-14}$ | $2.19 \cdot 10^{-12}$ | $4.17 \cdot 10^{-14}$ | $4.35 \cdot 10^{-14}$ | $3.39 \cdot 10^{-14}$ | $3.10 \cdot 10^{-14}$ |
| Absolute Error | 0.660 | $7.33 \cdot 10^{-14}$ | $1.84 \cdot 10^{-12}$ | $5.07 \cdot 10^{-14}$ | $4.13 \cdot 10^{-14}$ | $2.80 \cdot 10^{-14}$ | $2.87 \cdot 10^{-14}$ |

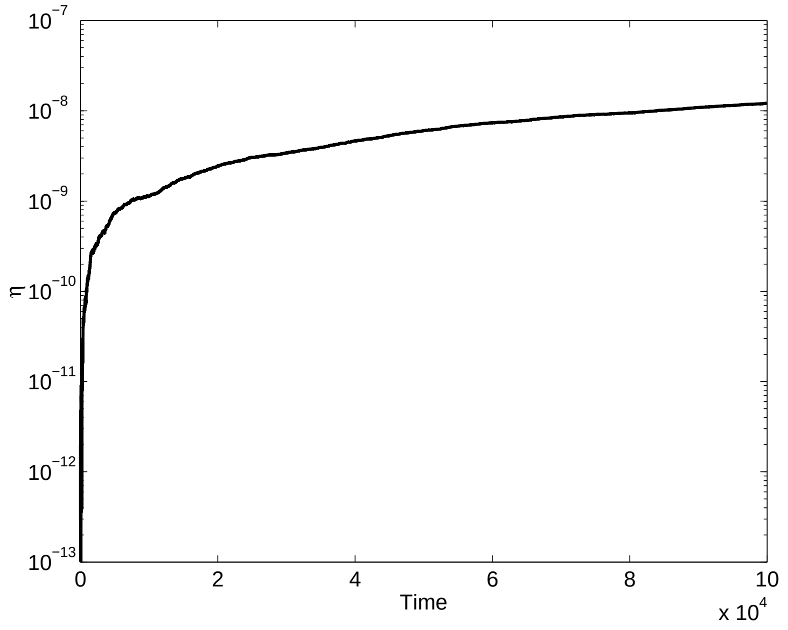

Fig. 4.6. *Advection equation solution norm difference. 128 coefficients* $\delta t = 5$ $\sigma = 13.00$ $b = 17.94$. *Computation time*=2.29 *hours.*

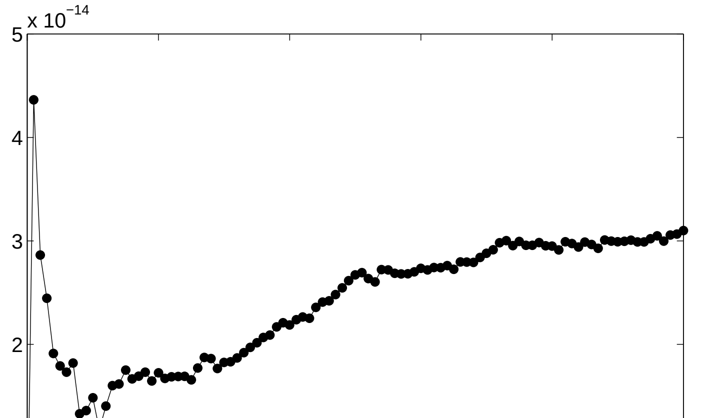

Fig. 4.7. *Diffusion equation relative error. 32 coefficients* $\delta t = 10$ $\sigma = 1.06$ $b = 1.28$. *Computation time*=18.20 *seconds.*

**4.3. Parallel 1-1 Pade' Nonparaxial Beam Propagation: Sparse Matrices.** Our second example stems from the field of computational optics. The nonparaxial scalar beam propagation equation

$$\frac{\partial u}{\partial z} = i\beta u - i\sqrt{\left(\frac{\partial^2}{\partial x^2} + k_0^2 n^2(x,y,z)\right)}u \tag{4.6}$$

is commonly used to study photonic structures. It arises from a slowly varying envelope assumption for the electric field

$$\vec{E} = ue^{i(\omega t - \beta z)}\vec{y} \tag{4.7}$$

and subsequent substitution into the electric field vector wave equation. The parameters $\beta$, $k_0$ and $n(x,y,z)$ are the propagation constant, free space wavenumber $k_0 = \frac{\omega}{c}$, where $c$ is the speed of light, and the refractive index, which describes the shape of the simulated structure, respectively [20, 42].

The refractive index profile which we are interested in is shown in Fig. 4.8. This structure is referred to as a multi-mode interference (MMI) coupler [27]. One of its uses is to separate an optical signal arriving from the single port into multiple signals. To accomplish this, the physical dimensions of the cavity and the indices of refraction must be precisely tuned to generate an interference pattern at the outports. This particular example is taken from the recent dissertation by Greedy [16]. The dimensions are given in Table 4.14.

TABLE 4.14. *MMI Parameters*

| Parameter | Value |
|---|---|
| $\lambda$ | $1.064\mu m$ |
| $n_{mmi}$ | 3.44746 |
| $n_{background}$ | 1.0 |
| Inport Length | $25\mu m$ |
| Outport Length | $50\mu m$ |
| Base Length | $564.5\mu m$ |
| Port Width | $4.4\mu m$ |
| Base Width | $26.4\mu m$ |
| Outport Center-Center Distance | $6.6\mu m$ |

For the initial condition we choose the lowest order mode of the inport. The propagation constant which corresponds to the solution is approximately $\beta \approx 20.346$. For the propagation, we choose $\delta z = 0.5\mu m$ and $\delta x = 0.25\mu m$.

With the problem constructed, we now apply Weeks method to its solution. Begin by replacing the spatial derivatives in

$$\frac{\partial u}{\partial z} = i\beta u - i\sqrt{\left(\frac{\partial^2}{\partial x^2} + k_0^2 n^2(x,y,z)\right)}u \tag{4.8}$$

with second order centered differences to discretize space

$$\frac{d\vec{u}}{dz} = i\beta\vec{u} - i\sqrt{(D + k_0^2\bar{n}^2)}\vec{u}. \tag{4.9}$$

The boundary conditions are chosen to be perfectly reflecting (Dirichlet). The goal is thus to solve this system of linear ordinary differential equations accurately. If we now assume that the index of refraction is constant over a $z$-step, as it is for the piecewise defined MMI structure, and Laplace transform equation (4.9), then

$$s\hat{u}(s) - \vec{u}(0) = i\beta\hat{u}(s) - i\sqrt{(D + k_0^2\bar{n}^2)}\hat{u}(s) \tag{4.10}$$

$$\hat{u}(s) = \left[sI - i\left(\beta - \sqrt{(D + k_0^2\bar{n}^2)}\right)\right]^{-1}\vec{u}(0) \tag{4.11}$$

$$\hat{u}(s) = \left[sI - i\beta(I - \sqrt{I + A})\right]^{-1}\vec{u}(0) \tag{4.12}$$

where

$$A = \frac{D + k_0^2\bar{n}^2 - \beta^2 I}{\beta^2}. \tag{4.13}$$

Clearly $\hat{u}(s) = (sI - M)^{-1}\vec{u}(0)$ with $M = i\beta(I - \sqrt{(I + A)})$ is the Laplace transform of the matrix exponential $e^{(M\delta z)}$.

$M$ itself involves the square root of a matrix. Fortunately, this computation is a well-studied problem with many possible algorithms [19, 43]. For our illustrative purposes, a simple 1-1 Padé approximation to the square root provides sufficient accuracy

$$M \approx \frac{-2i\beta A}{4I + A} \equiv L^{-1}R. \tag{4.14}$$

Note that $L$ and $R$ are still sparse matrices. Instead of explicitly generating $L^{-1}R$ and forming the Schur decomposition, which will create a dense matrix, we opt for a formalism which utilizes the sparsity of $L$ and $R$ and is amendable to parallelization. Namely, since $I = L^{-1}L$, we can write

$$\hat{u} \approx \left[sI - L^{-1}R\right]^{-1}\vec{u}(0) \tag{4.15}$$

$$\hat{u} \approx (sL - R)^{-1}(L\vec{u}(0)). \tag{4.16}$$

$L\vec{u}(0)$ need only be computed once and the matrix to invert $(sL - R)^{-1}$ is sparse. Furthermore, since the inversions are independent, the sum in the computation of the coefficients

$$\vec{a}_n \approx \frac{e^{-in\pi/(2M)}}{2M}\sum_{m=-M}^{M-1} e^{-in\theta_m}\frac{2b}{1 - e^{i\theta_{m+1/2}}}\left[\left(\sigma - b\frac{e^{i\theta_{m+1/2}} + 1}{e^{i\theta_{m+1/2}} - 1}\right)L - R\right]^{-1}(L\vec{u}(0)) \tag{4.17}$$

can be dispersed over parallel processors. The partial results are then passed to a central processor which computes the sum for $\vec{u}(z)$ by the midpoint or FFT algorithms.

Due to the lack of a standard parallel implementation of MATLAB, it has been necessary to program in C. Our particular code utilizes the Message-Passing Interface (MPI) [17] to manage the parallelization over a cluster of workstations. To verify the accuracy of the simulations a series MATLAB script also has been written.

Fig. 4.9 and 4.10 are simulations of the MMI structure in Fig. 4.8, using MATLAB's *sqrtm* and *expm1* functions and the 1-1 Padé approximant for the square root with Weeks method, respectively. The anticipated interference effect is clearly seen

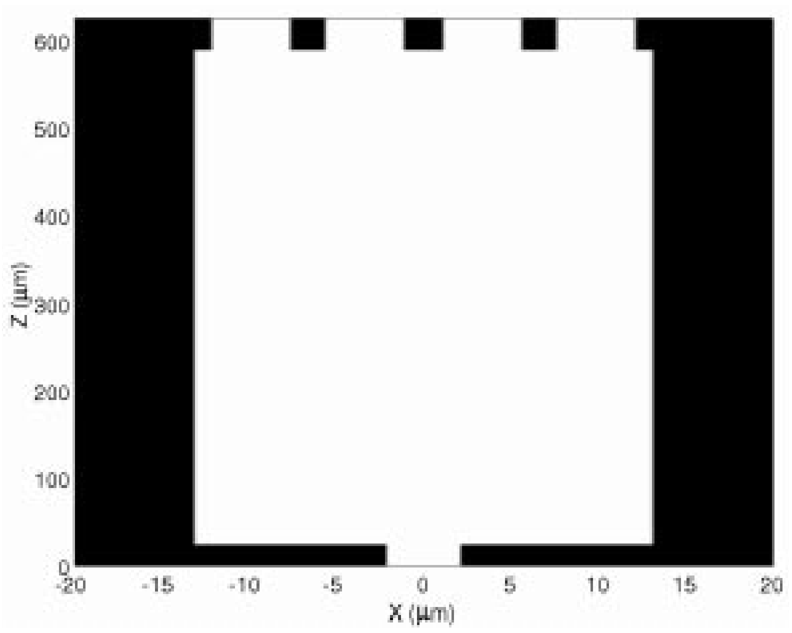

FIG. 4.8. *Multi-mode coupler*

in both figures. The parameters for the Weeks method are fixed at the values from Weideman's algorithm for the structure at $z=0$, $\sigma\approx3.898$ and $b\approx8.400$ and the sum is performed using 64 coefficients. The magnitude of the solution at $z=625\mu m$ computed using the MATLAB and C codes with the 1-1 Padé is plotted in Fig. 4.11. Also plotted in this figure is the 'exact' solution using the full square root matrix and exponential matrix from MATLAB. This simulation required approximately 40.4 seconds.

TABLE 4.15. *Parallel Computation Time:* 64 *Coefficients,* 500 *Steps*

| Number Processors | Computation Time (sec) | |
|---|---|---|
| | Midpoint | FFT |
| 1 | 716 | 23 |
| 2 | 444 | 50 |
| 4 | 241 | 53 |

Shown in Table 4.15 is the time required for the simulation of the MMI structure as a function of the number of processors. The sum is performed using the midpoint and FFT methods with 64 coefficients. The total number of inversions is 256. A factor of 2 arises from the fact that the coefficients $a_n$ are computed using a sum from $m=-N,\ldots,N-1$. Another factor of 2 arises from doubling the number of coefficients to compute an estimate of the error. Using $\delta x=0.5\mu m$ on a width of $40\mu m$, the matrix has a total of 160 elements. The matrices involved in the computation are however tridiagonal and thus can be efficiently inverted using the Thomas algorithm [29]. The

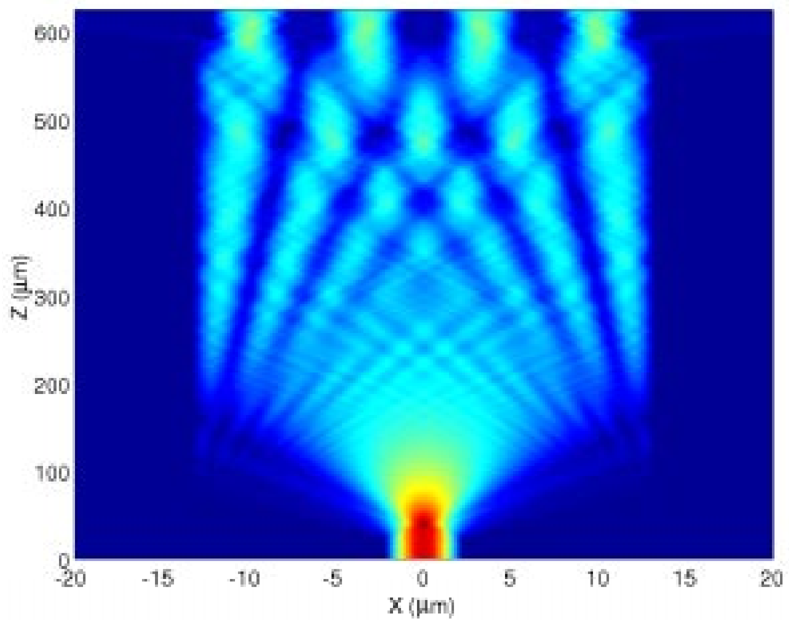

Fig. 4.9. *Multi-mode coupler intensity: exact square root matrix*

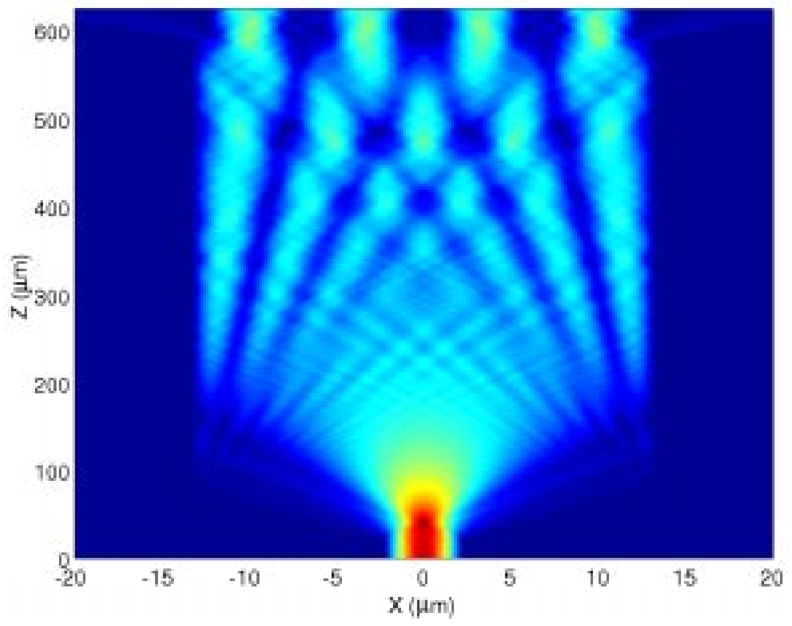

Fig. 4.10. *Multi-mode coupler intensity: 1-1 Padé approximation*

time is the number of seconds required to advance 500 steps. As expected, the FFT algorithm is considerably faster than the midpoint. Comparing computation times

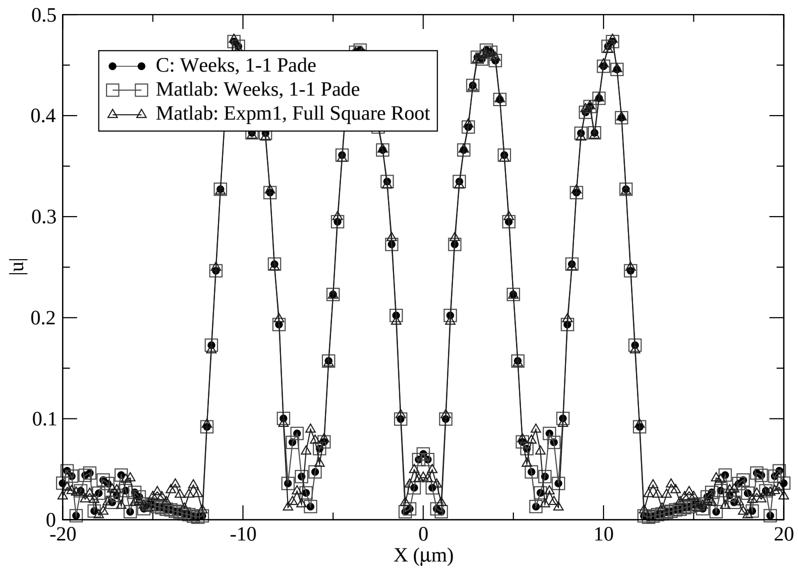

FIG. 4.11. *Intensity at 625 $\mu m$: Padé approximation accuracy*

716/23≈31.1 to the operation counts

$$\frac{N^2}{NlnN} \approx \frac{160}{ln(160)} \approx 31.5 \tag{4.18}$$

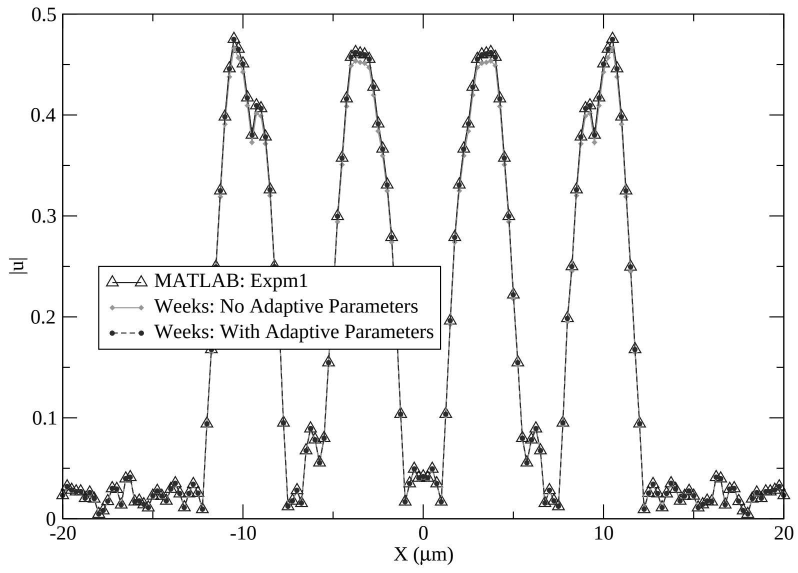

FIG. 4.12. *Intensity at 625 $\mu m$: adaptive $(\sigma,b)$ parameters analysis*

TABLE 4.16. *Parameter Algorithms for the MMI Structure*

| $Z(\mu m)$ | Parameters | Weeks:N | Weeks:b Half | MinMax:Grid | MinMax:Search | Pseudo:rmin | Pseudo:Y | Weideman |
|---|---|---|---|---|---|---|---|---|
| 0 | $\sigma$ | 1 | 1 | 20 | 4.80 | 4.02 | 1.05 | 5.86 |
|  | $b$ | 32 | 1/2 | 10 | 3.00 | 8.98 | 6.01 | 11.83 |
| 25 | $\sigma$ | 1 | 1 | 20 | 4.80 | 3.91 | 1.05 | 12.01 |
|  | $b$ | 32 | 1/2 | 10 | 3.00 | 8.86 | 6.01 | 16.95 |
| 590 | $\sigma$ | 1 | 1 | 20 | 4.80 | 3.81 | 1.05 | 11.84 |
|  | $b$ | 32 | 1/2 | 10 | 3.00 | 8.77 | 6.01 | 16.79 |
| Absolute Error $z = 625\mu m$ |  | 12.28 | Failed | $1.19 \cdot 10^{-3}$ | Failed | 0.44 | Failed | $4.25 \cdot 10^{-4}$ |

Table 4.17. *Weideman Method vs Number of Coefficients*

| $N$ | $Z(\mu m)$ | $\sigma$ | $b$ | Absolute Error at $625\mu m$ |
|---|---|---|---|---|
| | 0 | 13.26 | 18.20 | |
| 32 | 25 | 19.94 | 24.90 | $2.63 \cdot 10^{-2}$ |
| | 590 | 19.94 | 24.89 | |
| | 0 | 5.86 | 11.83 | |
| 64 | 25 | 12.01 | 16.95 | $4.25 \cdot 10^{-4}$ |
| | 590 | 11.84 | 16.79 | |
| | 0 | 3.02 | 7.96 | |
| 128 | 25 | 6.53 | 11.47 | $9.16 \cdot 10^{-5}$ |
| | 590 | 6.45 | 11.40 | |

Table 4.18. *Weideman Method vs Step Size*

| $\delta z(\mu m)$ | $Z(\mu m)$ | $\sigma$ | $b$ | Absolute Error at $625\mu m$ |
|---|---|---|---|---|
| | 0 | 12.49 | 17.44 | |
| 1 | 25 | 19.95 | 24.90 | 1.15 |
| | 590 | 19.94 | 24.90 | |
| | 0 | 5.86 | 11.83 | |
| 0.5 | 25 | 12.01 | 16.95 | $4.25 \cdot 10^{-4}$ |
| | 590 | 11.84 | 16.79 | |
| | 0 | 3.06 | 8.00 | |
| 0.25 | 25 | 5.69 | 10.66 | $1.20 \cdot 10^{-6}$ |
| | 590 | 5.60 | 10.54 | |
| | 0 | 1.78 | 5.38 | |
| 0.125 | 25 | 2.86 | 5.84 | $8.04 \cdot 10^{-9}$ |
| | 590 | 2.81 | 5.81 | |

one finds that the difference is due primarily to the summation algorithms. The increase in time for the FFT algorithm reflects the fact that the inversions can be performed in a time interval shorter than that needed to setup the parallelization and pass the ordered information between processors. For a dense matrix, where the inversions dominate the computations, the FFT algorithm may also benefit from parallelization.

In Fig. 4.12 we demonstrate the importance of adapting $\sigma$ and $b$ to the changes in the structure. The simulations were performed using the full square root matrix from MATLAB and not the 1-1 Padé approximant. The parameters used are from Weideman's algorithm. The actual values as well as those from the other parameter selection algorithms are provided in Table 4.16. What is surprising from this data is the failure of the two parameter selection algorithms which utilize a search routine. We see from the MinMax:Search and Pseudo:Y approaches, that even if the underlying error minimization approach is reasonable, the lack of a robust search routine can be catastrophic.

Finally, in Table 4.17 and 4.18 are the convergence analysis of Weeks method with Weideman's parameters. As expected, decreasing the time step leads to a dramatic increase in accuracy and there is a saturation in the accuracy of the method as the number of Laguerre expansion coefficients increases.

**5. Conclusions** In this paper, we have investigated the feasibility of using Weeks method to compute matrix functions. In particular, we have been interested in the evaluation of the matrix exponential in order to spectrally accurately propagate dispersive, diffusive, and advective equations. This has entailed an extension of the scalar Weeks method to those matrix-valued functions with finite singularities, an analysis of the errors and implementation costs, and the development of selection algorithms for the method's two tuning parameters $\sigma$ and $b$. Applications involving both pathological matrices and the numerical solution of the advection-diffusion and nonparaxial beam propagation equation highlight the utility of this approach.

Our conclusion from this study is that Weeks method provides a stable and accurate approach to the numerical calculation of the matrix exponential. The method is however highly sensitive to a proper choice for the two tuning parameters. With the regards to the selection algorithms, we have found that Weeks original suggestions are wholly inadequate. They cannot be relied upon for a general matrix to yield a spectrally accurate solution. A straight forward maximization of the radius of convergence of the Maclaurin series in the method is an efficient alternative. It can fail if the denominator in the expression for the radius of convergence is particularly small. By far the most robust and accurate method is the adaption of Weideman's method to matrix functions. It is also applicable to a general matrix-valued function as opposed to our own pseudospectra based approach which is limited to the matrix exponential. Weideman's algorithm is however very expensive for large matrices since it relies on a calculation of the expansion coefficients. The pseudospectra based approach has proved to be considerably less expensive than Weideman's and nearly as accurate. We have also found that, except for Weeks original suggestions, scaling and squaring the matrix exponential leads to a dramatic improvement in the method.

With regards to efficiency, at least for our implementation and a fixed time step, the Weeks method is not as fast as the scaling/Pade' approach utilized in MATLAB's *expm1* function. We have also shown that although the midpoint method for the expansion coefficients lends itself to parallelization, the fast Fourier transform algorithm is far more efficient.

To note is that there are still many possible improvements to the matrix Weeks method. One which could particularly benefit the parameter selection algorithms is a more sophisticated technique for a constrained search of the $(\sigma,b)$ plane. The examples in this paper show that although the parameter selection approach may be valid, the inability to constrain the search can lead to erroneous computations. Another improvement is to automate the scaling factor for the matrix exponential. This scaling may be particularly important for matrices with large eigenvalues. To increase the speed, we have two suggestions. The first is to apply a Krylov subspace method instead of a full Schur factorization. This will require a restructuring of the parameter selection algorithms and an analysis of the critical singularities. Also, for matrices with purely imaginary eigenvalues, we have shown that the singularity which defines the radius of convergence is the eigenvalue with the largest magnitude. It is thus not necessary to compute the expensive Schur decomposition to determine all the eigenvalues of the matrix. Instead, a significant increase in the speed of the parameter selection algorithms may be gained by computing only the largest, critical eigenvalue. Finally, we conclude by pointing out that in this paper we have focused upon the matrix exponential. The has allowed us to quantify our discussion and assess the qualities of the method for spectral propagation. However, the general theory of the Weeks method is not limited to this function. The application of the Weeks method to certain classes of matrix functions and highly nonnormal matrices may be of future interest.

**Acknowledgments.** The authors wish to thank J.A.C. Weideman for access to his MATLAB scripts for the scalar Weeks method and P.P. Valkó for making available a preprint of reference [2]. M. Brio would like to acknowledge support from the NSF grant ITR-0325097. This work was supported by the Air Force Office of Scientific Research under grant number F49620-02-1-0380.

## REFERENCES

[1] J. Abate, G. Choudhury and W. Whitt, *On the Laguerre method for numerically inverting Laplace transforms*, INFORMS Journal on Computing, 8, 413-427, 1996.

[2] J. Abate and P. P. Valkó, *Multi-precision Laplace transform inversion*, Int. J. Numer. Meth. Engng., 60, 5, 979-993, 2004.

[3] G. Arfken and H. Weber, *Mathematical Methods for Physicists*, Harcourt Academic Press, San Diego, 5th edition, 2001.

[4] David H. Bailey, Y. Hida, X. S. Li and B. Thompson, *ARPREC: An Arbitrary Precision Computing Package*, http://crd.lbl/gov/∼dhbailey/mpdist/

[5] Richard Bellman, *Numerical Inversion of the Laplace Transform: Applications to Biology, Economics, Engineering, and Physics*, American Elsevier Pub. Co., New York, 1966.

[6] A. J. Davies and D. Crann, *Parallel Laplace transform methods for boundary element solutions of diffusion-type problems*, Electronic Journal of Boundary Elements, 2, 231-238, 2002.

[7] B. Davies, *Integral Transforms and Their Applications*, Springer-Verlag, New York, 3rd edition, 2002.

[8] P. Davies and N. Higham, *A Schur-Parlett algorithm for computing matrix functions*, SIAM J. Matrix Anal. Appl., 25, 2, 464-485, 2003.

[9] F. R. De Hoog, J. H. Knight and A. N. Stokes, *An improved method for numerical inversion of Laplace transforms*, SIAM J. Sci. Stat. Comput., 3, 357-366, 1982.

[10] Gustav Doetsch, *Guide to the Applications of the Laplace and Z-Transforms*, 2nd English edition, Van Nostrand Reinhold Company, London, 1971.

[11] D. G. Duffy, *On the numerical inversion of Laplace transforms: Comparison of three new methods on characteristic problems from applications*, ACM Trans. Math. Software, 19, 333-359, 1993.

[12] G. Giunta, G. Lacetti and M. R. Rizzadri, *More on the Weeks method for the numerical inversion of the Laplace transform*, Numer. Math., 54, 193-200, 1988.

[13] G. Giunta, A. Murli and G. Schmid, *An analysis of bilinear transform polynomial methods of inversion of Laplace transforms*, Numer. Math., 69, 269-282, 1995.

[14] G. Goloub and Charles Van Loan, *Matrix Computations*, John Hopkins University Press, Baltimore, 1983.

[15] Urs. Graf, *Applied Laplace Transforms and Z-Transforms for Scientists and Engineers: A Computational Approach Using a Mathematica Package*, Birkhäuser Verlag, Basel, 2004.

[16] Stephen Greedy, *Advances in the spectral index method for the analysis of photonic integrated circuits*, Doctoral Dissertation, Univ. of Nottingham, 2002.

[17] W. Gropp, E. Lusk and A. Skjellum, *Using MPI: Portable Parallel Programming With the Message-Passing Interface*, The MIT Press, London, 1995.

[18] J. R. Higgins, *Completeness and Basis Functions of Sets of Special Functions*, Cambridge Univ. Press, Cambridge, 1977.

[19] Nicholas Higham, *Stable iterations for the matrix square root*, Numerical Algorithms, 15, 227-242, 1997.

[20] G. Lifante, *Integrated Photonics: Fundamentals*, John Wiley and Sons, San Francisco, 2003.

[21] S. H. Lui, *Computation of pseudospectra by continuation*, SIAM J. Sci. Comp., 18, 567-573, 1997.

[22] MATLAB http://www.mathworks.com

[23] C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Review, 45, 1, 3-49, 2003.

[24] A. Murli and M. Rizzardi, *Algorithm 682. Talbot's method for the Laplace inversion problem*, ACM Trans. Math. Soft., 16, 158-168, 1990.

[25] C. O'Cinneide, *Euler summation for Fourier series and Laplace transform inversion*, Stochastic Models, 13, 315-337, 1997.

[26] B. N. Parlett, *A recurrence among the elements of functions of triangular matrices*, Linear Algebra Appl., 14, 117-121, 1976.

[27] E. A. Patent, J. J. G. M van der Tol, J. J. M Binsma and M. K. Smit, *Conditions for optimum performance of unbalanced MMI couplers*, Optics Express, 9, 7, 2001.

[28] M. Popescu and W. Shyy, *Assessment of dispersion-relation-preserving and space-time CE/SE schemes for wave equations*, Numerical Heat Transfer, 42, 93-118, 2002.

[29] W. H. Press, S. A. Teukolsky, W. Vetterling and B. Flannery, *Numerical Recipes In C: The Art of Scientific Computing*, Cambridge University Press, 1992.

[30] E. A. Sudicky, *The Laplace transform Galerkin technique for efficient time-continuous solution of solute transport in double-porosity media*, Geoderma, 46, 209-232, 1990.

[31] A. Talbot, *The accurate numerical inversion of Laplace transforms*, J. Inst. Maths. Applics., 23, 97-120, 1979.

[32] Hillel. Tal-Ezer, *Spectral methods in time for hyperbolic equations*, SIAM J. Numer. Anal., 23, 1, 11-26, 1986.

[33] L. Trefethen and D. Bau III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[34] L. Trefethen, *Computation of pseudospectra*, Acta Numerica, 8, 247-295, 1999.

[35] L. Trefetehn, *Spectral Methods in MATLAB*, SIAM, Philadelphia, 2000.

[36] W. T. Weeks, *Numerical inversion of the Laplace transform using Laguerre functions*, J. Assoc. Comput. Mach., 13, 419-429, 1966.

[37] J. A. C. Weideman, *Algorithms for parameter selection in the Weeks method for inverting the Laplace transform*, SIAM J. Sci. Comput., 21, 1, 111-128, 1999.

[38] J. Wimp, *Sequence Transformations and Their Applications*, Academic Press, New York, 1981.

[39] T. G. Wright and L. N. Trefethen, *Large-scale computation of pseudospectra using ARPACK and Eigs*, SIAM J. Sci. Comp., 23, 591-605, 2001.

[40] P. P. Valkó and B. L. Vojta, The list, http://pumpjack.tamu.edu/∼valko, 2001.

[41] P. P. Valkó and J. Abate, *Comparison of sequence accelerators for the Gaver method of numerical Laplace transform inversion*, Comput. Math. Appl., 48, 629-636, 2004.

[42] Junji. Yamauchi, *Propagating Beam Analysis of Optical Waveguides*, Research Studies Press, Philadelphia, 2003.

[43] David. Yevick, *The application of complex Padè approximants to vector field propagation*, IEEE Photonics Technology Letters, 12, 12, 1041-1135, 2000.