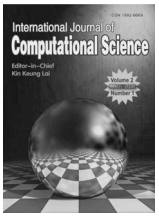
International Journal of Computational Science
1992-6669 (Print) 1992-6677 (Online) www.gip.hk/ijcs



Implicitly Defined High-Order Operator Splittings for Parabolic and Hyperbolic Variable-Coefficient PDE Using Modified Moments

James V. Lambers*

Department of Energy Resources Engineering, Stanford University,
367 Panama St Rm 065, Stanford, CA 94305-2220 USA
lambers@stanford.edu

Abstract. This paper presents a reformulation of Krylov Subspace Spectral (KSS) Methods, which build on the many contributions of Gene Golub, et al. pertaining to moments and Gaussian quadrature in the spectral domain in order to produce high-order accurate approximate solutions to variable-coefficient time-dependent PDE, including both parabolic and hyperbolic problems. This reformulation serves two useful purposes. First, it improves the numerical stability of these methods by removing cancellation arising from the approximation of certain derivatives by finite differences by computing these derivatives analytically. Second, it reveals that KSS methods are actually high-order operator splittings that are defined implicitly, in terms of derivatives of the nodes and weights of Gaussian quadrature rules with respect to a parameter. Efficient algorithms for computing these derivatives are provided, as well as the first application of KSS methods to systems of coupled PDE.

Keywords: spectral methods, Gaussian quadrature, variable-coefficient, Lanczos method, stability, heat equation, wave equation.

* Corresponding Author. Email: lambers@stanford.edu.

1 Introduction

Consider the following initial-boundary value problem in one space dimension,

$$u_t + L(x,D)u = 0 \quad \text{on } (0,2\pi)\times(0,\infty), \tag{1}$$

$$u(x,0) = f(x), \quad 0 < x < 2\pi, \tag{2}$$

with periodic boundary conditions

$$u(0,t) = u(2\pi,t), \quad t > 0. \tag{3}$$

The operator L is a second-order differential operator of the form

$$L(x,D)u = -pD^2u + q(x)u, \tag{4}$$

where $D = \frac{\partial}{\partial x}$, p is a positive constant and $q(x)$ is a nonnegative (but nonzero) smooth function. It follows that L is self-adjoint and positive definite. The exact solution can be represented using a Fourier series if $q(x)$ is constant, but here we concern ourselves exclusively with the solution of variable-coefficient problems, for which numerical methods are necessary.

In [13], [14] a class of methods, called Krylov subspace spectral (KSS) methods, was introduced for the purpose of solving time-dependent, variable-coefficient problems such as this one. These methods are based on the application of techniques developed by Golub and Meurant in [6], originally for the purpose of computing elements of the inverse of a matrix, to elements of the matrix exponential of an operator. It has been shown in these references that KSS methods, by employing different approximations of the solution operator for each Fourier component of the solution, achieve higher-order accuracy in time than other Krylov subspace methods (see, for example, [11]) for stiff systems of ODE. However, because these methods approximate derivatives of bilinear forms using finite differences, they are prone to numerical instability. In this paper, we address this issue, and also demonstrate that the resulting modified KSS methods are actually operator splittings.

Section 2 reviews the main properties of KSS methods, including algorithmic details and results concerning local accuracy. The main idea behind these methods is that each Fourier component of the solution is obtained from a *perturbation* of a *frequency-dependent* Krylov subspace in the direction of the initial data, instead of a single Krylov subspace generated from the data. It follows that KSS methods can be reformulated as high-order operator splittings that are implicitly defined in terms of directional derivatives of nodes and weights of Gaussian quadrature rules. This reformulation, and a description of the resulting splittings, is presented in Section 3. Generalization to the second-order wave equation, and systems of coupled PDEs, are described in Sections 4 and 5, respectively. In Section 6, various generalizations and future directions are discussed.

2 Krylov Subspace Spectral Methods

We begin with a review of the main aspects of KSS methods. Let $S(t) = \exp[-Lt]$ represent the exact solution operator of the problem (1), (2), (3), and let $\langle \cdot, \cdot \rangle$ denote the standard inner product of functions defined on $[0, 2\pi]$,

$$\langle f(x), g(x) \rangle = \int_0^{2\pi} \overline{f(x)} g(x)\, dx. \tag{5}$$

Krylov subspace spectral methods, introduced in [13], [14], use Gaussian quadrature on the spectral domain to compute the Fourier components of the solution. These methods are time-stepping algorithms that compute the solution at time $t_1, t_2, \ldots$, where $t_n = n\Delta t$ for some choice of Δt. Given the computed solution $\tilde{u}(x, t_n)$ at time t_n, the solution at time t_{n+1} is computed by approximating the Fourier components that would be obtained by applying the exact solution operator to $\tilde{u}(x, t_n)$,

$$\hat{u}(\omega, t_{n+1}) = \left\langle \frac{1}{\sqrt{2\pi}} e^{i\omega x}, S(\Delta t)\tilde{u}(x, t_n) \right\rangle. \tag{6}$$

KSS methods approximate these components with higher-order temporal accuracy than traditional spectral methods and time-stepping schemes. We briefly review how these methods work.

We discretize functions defined on $[0, 2\pi]$ on an N-point uniform grid with spacing $\Delta x = 2\pi/N$. With this discretization, the operator $L(x, D)$ and the solution operator $S(\Delta t)$ can be approximated by $N \times N$ matrices that represent linear operators on the space of grid functions, and the quantity (6) can be approximated by a bilinear form

$$\hat{u}(\omega, t_{n+1}) \approx \hat{\mathbf{e}}_\omega^H S_N(\Delta t)\mathbf{u}(t_n), \tag{7}$$

where

$$[\hat{\mathbf{e}}_\omega]_j = \frac{1}{\sqrt{2\pi}} e^{i\omega j \Delta x}, \quad [\mathbf{u}(t_n)]_j = u(j\Delta x, t_n), \tag{8}$$

and

$$S_N(t) = \exp[-L_N t], \quad [L_N]_{jk} = \sum_{\mu=0}^{m} a_\mu(j\Delta x)[D_N^\mu]_{jk} \tag{9}$$

where D_N is a discretization of the differentiation operator D that is defined on the space of grid functions. Our goal is to approximate (7) by computing an approximation to

$$[\hat{\mathbf{u}}^{n+1}]_\omega = \hat{\mathbf{e}}_\omega^H \mathbf{u}(t_{n+1}) = \hat{\mathbf{e}}_\omega^H S_N(\Delta t)\mathbf{u}(t_n). \tag{10}$$

In [6] Golub and Meurant describe a method for computing quantities of the form

$$\mathbf{u}^T f(A)\mathbf{v}, \tag{11}$$

where $\mathbf{u}$ and $\mathbf{v}$ are N-vectors, A is an $N \times N$ symmetric positive definite matrix, and f is a

smooth function. Our goal is to apply this method with $A = L_N$ where L_N was defined in (9), $f(\lambda) = \exp(-\lambda t)$ for some t, and the vectors $\mathbf{u}$ and $\mathbf{v}$ are derived from $\hat{\mathbf{e}}_\omega$ and $\mathbf{u}(t_n)$.

The basic idea is as follows: since the matrix A is symmetric positive definite, it has real eigenvalues

$$b = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N = a > 0, \tag{12}$$

and corresponding orthogonal eigenvectors $\mathbf{q}_j$, $j = 1, \ldots, N$. Therefore, the quantity (11) can be rewritten as

$$\mathbf{u}^T f(A)\mathbf{v} = \sum_{\ell=1}^{N} f(\lambda_\ell)\mathbf{u}^T\mathbf{q}_j\mathbf{q}_j^T\mathbf{v}. \tag{13}$$

We let $a = \lambda_N$ be the smallest eigenvalue, $b = \lambda_1$ be the largest eigenvalue, and define the measure $\alpha(\lambda)$ by

$$\alpha(\lambda) = \begin{cases} 0, & \text{if } \lambda < a \\ \sum_{j=i}^{N} \alpha_j\beta_j, & \text{if } \lambda_i \leq \lambda < \lambda_{i-1}, \quad \alpha_j = \mathbf{u}^T\mathbf{q}_j, \quad \beta_j = \mathbf{q}_j^T\mathbf{v}, \\ \sum_{j=i}^{N} \alpha_j\beta_j, & \text{if } b \leq \lambda \end{cases} \tag{14}$$

If this measure is positive and increasing, then the quantity (11) can be viewed as a Riemann-Stieltjes integral

$$\mathbf{u}^T f(A)\mathbf{v} = I[f] = \int_a^b f(\lambda)\, d\alpha(\lambda). \tag{15}$$

As discussed in [2], [4], [5], [6], the integral $I[f]$ can be bounded using either Gauss, Gauss-Radau, or Gauss-Lobatto quadrature rules, all of which yield an approximation of the form

$$I[f] = \sum_{j=1}^{K} w_j f(t_j) + \sum_{j=1}^{M} v_j f(z_j) + R[f], \tag{16}$$

where the nodes t_j, $j = 1, \ldots, K$, and z_j, $j = 1, \ldots, M$, as well as the weights w_j, $j = 1, \ldots, K$, and v_j, $j = 1, \ldots, M$, can be obtained using the symmetric Lanczos algorithm if $\mathbf{u} = \mathbf{v}$, and the unsymmetric Lanczos algorithm if $\mathbf{u} \neq \mathbf{v}$ (see [9]).

In the case $\mathbf{u} \neq \mathbf{v}$, there is the possibility that the weights may not be positive, which destabilizes the quadrature rule (see [1] for details). Therefore, it is best to handle this case by rewriting (11) using decompositions such as

$$\mathbf{u}^T f(A)\mathbf{v} = \frac{1}{\delta}[\mathbf{u}^T f(A)(\mathbf{u} + \delta\mathbf{v}) - \mathbf{u}^T f(A)\mathbf{u}], \tag{17}$$

where δ is a small constant. Guidelines for choosing an appropriate value for δ can be found in [14, section 2.2].

Employing these quadrature rules yields the following basic process (for details see [13], [14]) for computing the Fourier coefficients of $\mathbf{u}(t_{n+1})$ from $\mathbf{u}(t_n)$. It is assumed that when the Lanczos

algorithm (symmetric or unsymmetric) is employed, $M+K$ iterations are performed to obtain the $M+K$ quadrature nodes and weights.

for $\omega = -N/2+1,\ldots,N/2$
  Choose a scaling constant δ_ω
  Compute $u_1 \approx \hat{\mathbf{e}}_\omega^H S_N(\Delta t)\hat{\mathbf{e}}_\omega$
    using the symmetric Lanczos algorithm
  Compute $u_2 \approx \hat{\mathbf{e}}_\omega^H S_N(\Delta t)(\hat{\mathbf{e}}_\omega + \delta_\omega \mathbf{u}^n)$
    using the unsymmetric Lanczos algorithm
  $[\hat{\mathbf{u}}^{n+1}]_\omega = (u_2 - u_1)/\delta_\omega$
end

It should be noted that the constant δ_ω plays the role of δ in the decomposition (17), and the subscript ω is used to indicate that a different value may be used for each wave number $\omega = -N/2+1,\ldots,N/2$. Also, in the presentation of this algorithm in [14], a polar decomposition is used instead of (17), and is applied to sines and cosines instead of complex exponential functions.

This algorithm has high-order temporal accuracy, as indicated by the following theorem. Let $BL_N([0,2\pi]) = \text{span}\{\, e^{-i\omega x} \,\}_{\omega=-N/2+1}^{N/2}$ denote a space of bandlimited functions with at most N non-zero Fourier components.

Theorem 1. *Let L be a self-adjoint m-th order positive definite differential operator on $C_p([0, 2\pi])$ with coefficients in $BL_N([0,2\pi])$. Let $f \in BL_N([0,2\pi])$, and let $M = 0$. Then the preceding algorithm, applied to the problem (1), (2), (3), is consistent; i.e.*

$$[\hat{\mathbf{u}}^1]_\omega - \hat{u}(\omega, \Delta t) = O(\Delta t^{2K}),$$

for $\omega = -N/2+1,\ldots,N/2$.

Proof. See [14, Lemma 2.1, Theorem 2.4].

The preceding result can be compared to the accuracy achieved by an algorithm described by Hochbruck and Lubich in [11] for computing $e^{A\Delta t}\mathbf{v}$ for a given matrix A and vector $\mathbf{v}$ using the unsymmetric Lanczos algorithm. As discussed in [11], this algorithm can be used to compute the solution of some ODEs without time-stepping, but this becomes less practical for ODEs arising from a semi-discretization of problems such as (1), (2), (3), due to their stiffness. In this situation, it is necessary to either use a high-dimensional Krylov subspace, in which case reorthogonalization is required, or one can resort to time-stepping, in which case the local temporal error is only $O(\Delta t^K)$, assuming a K-dimensional Krylov subspace. Regardless of which remedy is used, the computational effort needed to compute the solution at a fixed time T increases substantially.

The difference between KSS methods and the approach described in [11] is that in the former, a different K-dimensional Krylov subspace is used for each Fourier component, instead of the same subspace for all components as in the latter. As can be seen from numerical results comparing the two approaches in [14], using the same subspace for all components causes a loss of accu-

racy as the number of grid points increases, whereas KSS methods do not suffer from this phenomenon.

Unfortunately, the difference quotient used to compute each Fourier component of the solution can be numerically unstable if δ_ω is chosen too small, but this parameter must also be chosen small enough to ensure stability of the quadrature rules. Furthermore, stability analysis is difficult to carry out. We will now address these difficulties, and realize an additional benefit in the process: operator splittings that are high-order accurate in time, and, though explicit, possess favorable stability properties.

3 Reformulation of KSS Methods as Splittings

From the algorithm given in the preceding section, we see that each Fourier component $[\hat{\mathbf{u}}^{n+1}]_\omega$ approximates the derivative

$$\frac{d}{d\delta_\omega}\left[\hat{\mathbf{e}}_\omega^H(\hat{\mathbf{e}}_\omega+\delta_\omega\mathbf{u}^n)\mathbf{e}_1^T\exp[T_\omega(\delta_\omega)\Delta t]\mathbf{e}_1\right]\Bigg|_{\delta_\omega=0} \tag{18}$$

where $T_\omega(\delta_\omega)$ is the tridiagonal matrix output by the unsymmetric Lanczos algorithm applied to the matrix L_N with starting vectors $\hat{\mathbf{e}}_\omega$ and $(\hat{\mathbf{e}}_\omega+\delta_\omega\mathbf{u}^n)$ (which reduces to the symmetric Lanczos algorithm for $\delta_\omega=0$). In this section, we will compute these derivatives analytically. This leads to an improved algorithm, henceforth referred to as the "new formulation" of KSS methods, over the "original formulation", described in [16].

3.1 High-Order Splittings

For a given δ_ω, let $\lambda_{\omega,j}$, $j=1,\ldots,K$, be the nodes of the K-point Gaussian rule obtained by applying the unsymmetric Lanczos algorithm to L_N with starting vectors $\hat{\mathbf{e}}_\omega$ and $(\hat{\mathbf{e}}_\omega+\delta_\omega\mathbf{u}^n)$. Let $w_{\omega,j}$, $j=1,\ldots,K$, be the corresponding weights. Then, letting $\delta_\omega\to 0$, we obtain the following, assuming all required derivatives exist:

$$\begin{aligned}
[\hat{\mathbf{u}}^{n+1}]_\omega &= \hat{\mathbf{e}}_\omega^H\mathbf{u}^{n+1}\\
&= \frac{d}{d\delta_\omega}\left[\hat{\mathbf{e}}_\omega^H(\hat{\mathbf{e}}_\omega+\delta_\omega\mathbf{u}^n)\mathbf{e}_1^T\exp[-T_\omega(\delta_\omega)\Delta t]\mathbf{e}_1\right]\Bigg|_{\delta_\omega=0}\\
&= \frac{d}{d\delta_\omega}\left[\hat{\mathbf{e}}_\omega^H(\hat{\mathbf{e}}_\omega+\delta_\omega\mathbf{u}^n)\sum_{k=1}^K w_j e^{-\lambda_j\Delta t}\right]\Bigg|_{\delta_\omega=0}\\
&= \hat{\mathbf{e}}_\omega^H\mathbf{u}^n\sum_{k=1}^K w_j e^{-\lambda_j\Delta t}+\sum_{k=1}^K w_j' e^{-\lambda_j\Delta t}-\Delta t\sum_{k=1}^K w_j\lambda_j' e^{-\lambda_j\Delta t}
\end{aligned} \tag{19}$$

where the $'$ denotes differentiation with respect to δ_ω, and evaluation of the derivative at $\delta_\omega=0$.

Equivalently, these derivatives are equal to the length of $\mathbf{u}^n$ times the directional derivatives of the nodes and weights, as functions defined on $\mathbb{R}^N$, in the direction of $\mathbf{u}^n$, and evaluated at the origin.

It should be noted that in the above expression for $[\hat{\mathbf{u}}^{n+1}]$, the nodes and weights depend on the wave number ω, but for convenience, whenever a fixed Fourier component is being discussed, the dependence of the nodes and weights on ω is not explicitly indicated.

It can be shown that the derivatives of the nodes and weights at $\delta_\omega = 0$ are Fourier components of pseudodifferential operators applied to $\mathbf{u}^n$. For example, when $K = 1$, the one node is

$$\lambda_1(\delta_\omega) = \frac{\hat{\mathbf{e}}_\omega^H L_N(\hat{\mathbf{e}}_\omega + \delta_\omega \mathbf{u}^n)}{\hat{\mathbf{e}}_\omega^H(\hat{\mathbf{e}}_\omega + \delta_\omega \mathbf{u}^n)} \tag{20}$$

and therefore

$$\lambda_1(0) = \hat{\mathbf{e}}_\omega^H(L_N - \lambda_1(0)I)\mathbf{u}^n. \tag{21}$$

Formulas for the derivatives of the nodes and weights in the case $K = 2$ are given in [12], but for $K > 2$, these derivatives are far more difficult to express in terms of L_N, as the nodes are roots of a polynomial of degree K.

It follows that by considering all Fourier components together, we find that KSS methods are actually high-order operator splittings "in disguise". These splittings have the form

$$\exp[-L(x,D)\Delta t] \approx \sum_{k=1}^{K} W_k(x,D)e^{-C_k(x,D)\Delta t}[I - \Delta t V_k(x,D)] \tag{22}$$

where K is the number of quadrature nodes, and the operators $C_k(x,D)$ and $W_k(x,D)$ are diagonal in the basis of trial functions (e.g., a constant-coefficient operator when using Fourier series). In fact, their eigenvalues are the k th nodes and weights, respectively. The operators $V_k(x,D)$ have the form

$$V_k(x,D) = C_k'(x,D) + \Delta t^{-1} W_k(x,D)^{-1} W_k'(x,D) \tag{23}$$

where the Fourier components of $C_k'(x,D)\mathbf{u}^n$ and $W_k'(x,D)\mathbf{u}^n$ are the derivatives of the nodes and weights with respect to δ_ω at $\delta_\omega = 0$. Because $e^{-C_k(x,D)\Delta t} \to I$ linearly as $\Delta t \to 0$, and $\sum_{j=1}^{K} W_k(x,D) = I$, it follows that the terms in $V_k(x,D)$ of order $O(\Delta t^{-1})$ cancel and pose no difficulty.

In the simplest case of $K = 1$, we obtain the splitting

$$\exp[-L(x,D)\Delta t] \approx e^{-C(x,D)t} P_N[I - tV(x,D)] \tag{24}$$

where P_N is the orthogonal projection onto $BL_N([0,2\pi])$, $C(x,D)$ is the constant-coefficient differential operator obtained by averaging the coefficients of $L(x,D)$, and $V(x,D) = L(x,D) - C(x,D)$. This amounts to a first-order accurate, two-stage splitting in $u(x,t_{n+1})$ is obtained by applying the forward Euler method to the problem $u_t + V(x,D)u = 0$, and then the constant-coefficient problem $u_t + C(x,D)u = 0$ is solved exactly with initial data $P_N[I - \Delta t V(x,D)]u(x,t_n)$.

As shown in [12] for this, splittings such as this facilitate stability analysis of KSS methods,

and such analysis demonstrates that KSS methods represent a "best-of-both-worlds" compromise between explicit and explicit time-stepping methods, as they possess the stability of implicit methods, but like explicit methods, they do not require solution of large systems of equations.

Unlike splitting such as the Strang splitting (see [18]), the stages of the splitting (22) cannot easily be described in terms of the coefficients of $L(x,D)$, because they are defined in terms of the nodes and weights of quadrature rules, which do not have a simple relation to the recursion coefficients, as they come from the eigenvalues and eigenvectors of Jacobi matrices. Nevertheless, these splittings can still be implemented efficiently, as the action of the operators $V_k(x,D)$ on a given grid function can be computed from the derivatives of the nodes and weights. We now discuss how to compute these derivatives.

3.2 Derivatives of the Nodes and Weights

From the Lanczos algorithm, $T_\omega(\delta_\omega)$ has the structure

$$T_\omega(\delta_\omega) = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{K-2} & \alpha_{K-1} & \beta_{K-1} \\ & & & \beta_{K-1} & \alpha_K \end{bmatrix}, \tag{25}$$

where all entries are functions of δ_ω. Because the nodes and weights are obtained from the eigenvalues and eigenvectors of this matrix, it is desirable to use these relationships to develop efficient algorithms for computing the derivatives of the nodes and weights in terms of those of the recursion coefficients. We will first describe such algorithms, and then we will explain how the derivatives of the recursion coefficients can be computed.

The nodes are the eigenvalues of $T_\omega(\delta_\omega)$. Because $T_\omega(0)$ is Hermitian, it follows that there exists a unitary matrix Q_ω^0 such that

$$T_\omega(0) = Q_\omega^0 \Lambda_\omega(0)[Q_\omega^0]^H. \tag{26}$$

The eigenvalues of $T_\omega(0)$ are distinct (see [9]). Because the eigenvalues are continuous functions of the entries of the matrix, they continue to be distinct for δ_ω sufficiently small, and therefore $T_\omega(\delta)$ remains diagonalizable. It follows that we can write

$$T_\omega(\delta_\omega) = Q_\omega(\delta_\omega)\Lambda_\omega(\delta_\omega)Q_\omega(\delta_\omega)^{-1}, \tag{27}$$

where $Q_\omega(0) = Q_\omega^0$. Differentiating (27) with respect to δ_ω and evaluating at $\delta_\omega = 0$ yields

$$\operatorname{diag}(\Lambda_\omega'(0)) = \operatorname{diag}\left(Q_\omega(0)^H T_\omega'(0) Q_\omega(0)\right), \tag{28}$$

since all other terms that arise from application of the product rule vanish on the diagonal. Therefore, for each ω, the derivatives of the nodes $\lambda_1,\ldots,\lambda_K$ are easily obtained by applying a similar-

ity transformation to the matrix of the derivatives of the recursion coefficients, $T'_\omega(0)$, where the transformation involves a matrix, $Q_\omega(0)$, that must be computed anyway to obtain the weights.

To compute the derivatives of the weights, we consider the equation

$$(T_\omega(\delta_\omega) - \lambda_j I)\mathbf{w}_j(\delta_\omega) = 0, \quad j = 1,\ldots,K, \tag{29}$$

where $\mathbf{w}_j(\delta_\omega)$ is an eigenvector of $T_\omega(\delta_\omega)$ with eigenvalue λ_j, normalized to have unit 2-norm. First, we differentiate this equation with respect to δ_ω and evaluate at $\delta_\omega = 0$. Then, we delete the last equation and eliminate the last component of $\mathbf{w}_j(0)$ and $\mathbf{w}'_j(0)$ using the fact that $\mathbf{w}_j(0)$ must have unit 2-norm. The result is a $(K-1)\times(K-1)$ system where the matrix is the sum of a tridiagonal matrix and a rank-one update. This matrix is independent of the solution $\mathbf{u}^n$, while the right-hand side is not. After solving this simple system, as well as a similar one for the left eigenvector corresponding to λ_j, we can obtain the derivative of the weight w_j from the first components of the two solutions. It should be noted that although $T_\omega(0)$ is Hermitian, $T_\omega(\delta_\omega)$ is, in general, complex symmetric, which is why the system corresponding to the left eigenvector is necessary.

3.3 Derivatives of the Recursion Coefficients

Let A be a symmetric positive definite $n\times n$ matrix and let $\mathbf{r}_0$ be an n-vector. Suppose that we have already carried out the symmetric Lanczos iteration to obtain orthogonal vectors $\mathbf{r}_0,\ldots,\mathbf{r}_K$ and the Jacobi matrix

$$T_K = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{K-2} & \alpha_{K-1} & \beta_{K-1} \\ & & & \beta_{K-1} & \alpha_K \end{bmatrix}. \tag{30}$$

Now, we wish to compute the entries of the modified matrix

$$\hat{T}_K = \begin{bmatrix} \hat{\alpha}_1 & \hat{\beta}_1 & & & \\ \hat{\beta}_1 & \hat{\alpha}_2 & \hat{\beta}_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \hat{\beta}_{K-2} & \hat{\alpha}_{K-1} & \hat{\beta}_{K-1} \\ & & & \hat{\beta}_{K-1} & \hat{\alpha}_K \end{bmatrix} \tag{31}$$

that results from applying the unsymmetric Lanczos iteration with the same matrix A and the initial vectors $\mathbf{r}_0$ and $\mathbf{r}_0 + \mathbf{f}$, where $\mathbf{f}$ is a given perturbation.

In [16], an iteration that produces $\hat{T}_K$ was presented, based on algorithms from [7]. It was used to efficiently obtain the recursion coefficients needed to approximate $\hat{\mathbf{e}}_\omega^H S_N(\Delta t)(\hat{\mathbf{e}}_\omega + \delta_\omega \mathbf{u}^n)$ from

those used to approximate $\hat{\mathbf{e}}_\omega^H S_N(\Delta t)\hat{\mathbf{e}}_\omega$. It was shown that with an efficient implementation of this algorithm in MATLAB, KSS methods are a viable option for solving parabolic problems when compared to MATLAB's built-in ODE solvers, even though the former are explicit and the latter are implicit.

In [12], this algorithm was used for a different purpose. From the expressions for the entries of $\hat{T}_K$, the derivatives of the recursion coefficients α_j, $j = 1,\ldots,K$, and β_j, $j = 1,\ldots,K-1$, can be obtained by setting $\mathbf{r}_0 = \hat{\mathbf{e}}_\omega$ and $\mathbf{f} = \delta_\omega \mathbf{u}^n$. By differentiating the recurrence relations that define $\hat{T}_K$ with respect to δ_ω and evaluating at $\delta_\omega = 0$, we obtain the following algorithm.

Algorithm 1. *Let $T_K(\delta)$ be the tridiagonal matrix produced by the unsymmetric Lanczos iteration with left initial vector $\mathbf{r}_0$ and right initial vector $\mathbf{r}_0 + \delta\mathbf{f}$. Let $\mathbf{r}_0,\ldots,\mathbf{r}_K$ be the $K+1$ unnormalized Lanczos vectors associated with $T_K(0)$. Given T_K, as defined in (30), whose entries are those of $T_K(\delta)$ evaluated at $\delta = 0$, the following algorithm generates the tridiagonal matrix T_K' whose entries are the derivatives of the entries of $T_K(\delta)$ with respect to δ, and evaluated at $\delta = 0$.*

$\beta_{-1} = 0$
$\mathbf{q}_{-1} = 0$
$\mathbf{q}_0 = \mathbf{f}$
$[\beta_0^2]' = \mathbf{r}_0^H \mathbf{q}_0$
$s_0 = \frac{1}{\beta_0}$
$t_0' = -\frac{[\beta_0^2]'}{\beta_0^2}$
$d_0' = 0$
for $j = 1,\ldots,K$
  $\alpha_j' = s_{j-1}\mathbf{r}_j^H \mathbf{q}_{j-1} + d_{j-1}'\beta_{j-2}$
  if $j < K$ **then**
    $d_j' = (d_{j-1}'\beta_{j-2} - \alpha_j')/\beta_{j-1}$
    $\mathbf{q}_j = (A - \alpha_j I)\mathbf{q}_{j-1} - \beta_{j-1}^2 \mathbf{q}_{j-2}$
    $[\beta_j^2]' = t_{j-1}'\beta_j^2 + s_{j-1}\mathbf{r}_j^H \mathbf{q}_j$
    $s_j = s_{j-1}/\beta_j$
    $t_j' = t_{j-1}' - \frac{[\beta_j^2]'}{\beta_j^2}$
  end
end

It is worth noting that because $\mathbf{r}_0 = \hat{\mathbf{e}}_\omega$, for each $\omega = -N/2+1,\ldots,N/2$, the inner products $\mathbf{r}_i^H \mathbf{q}_j$ can be computed for all ω simultaneously using appropriate FFTs. As shown in [16], [12], the resulting time-stepping algorithm, in either the original or the new formulation, requires $O(N \log N)$ floating-point operations per time step.

3.4 Consistency

We now prove that the reformulated KSS method maintains the same order of temporal accuracy as the original method. For simplicity, we assume that the recursion coefficients are computed by refining the spatial grid in order to compute all Fourier coefficients resulting from pointwise multiplication of grid functions, as is done in the original KSS algorithm presented in [14].

Theorem 2. *Let $N_K = 2^K N$, for a positive integer K, and let L be a self-adjoint positive definite differential operator of the form (4), with $q \in BL_N([0,2\pi])$. Assume that for $\omega = -N/2+1,\ldots,N/2$, the entries of $T_\omega(0)$ are computed using an N_K-point uniform grid. Let $\mathbf{f}$ is the discretization of $f(x) \in BL_N([0,2\pi])$ on an N-point uniform grid, and let $\mathbf{u}$ be the result of a single time step of a KSS method with step length Δt and K-node Gaussian rules used to compute each Fourier component of $\mathbf{u}$. Then*

$$\left|[\hat{\mathbf{u}}]_\omega - \left\langle e^{i\omega x}, e^{-L(x,D)\Delta t} f(x)\right\rangle\right| = O(\Delta t^{2K}). \tag{32}$$

Proof. Let $E_\omega(\Delta t)$ be the error in the Fourier component with frequency ω and time step Δt. Then, due to the $(2K-1)$-degree accuracy of Gaussian quadrature, we have

$$\begin{aligned}
E_\omega(\Delta t) &= [\hat{\mathbf{u}}]_\omega - \left\langle e^{i\omega x}, e^{-L(x,D)\Delta t} f(x)\right\rangle \\
&= \frac{d}{d\delta_\omega}\left[\hat{\mathbf{e}}_\omega^H(\hat{\mathbf{e}}_\omega+\delta_\omega\mathbf{f})\mathbf{e}_1^T \exp[-T_\omega(\delta_\omega)\Delta t]\mathbf{e}_1\right]\Big|_{\delta_\omega=0} - \hat{\mathbf{e}}_\omega^H e^{-L_N\Delta t}\mathbf{f} \\
&= \sum_{m=0}^{\infty}\frac{(-\Delta t)^m}{m!}\left\{\left[\hat{\mathbf{e}}_\omega^H(\hat{\mathbf{e}}_\omega+\delta_\omega\mathbf{f})\mathbf{e}_1^T T_\omega(\delta_\omega)^m\mathbf{e}_1\right]' - \hat{\mathbf{e}}_\omega^H L_N^m\mathbf{f}\right\} \\
&= \sum_{m=0}^{2K-1}\frac{(-\Delta t)^m}{m!}\left\{\left[\hat{\mathbf{e}}_\omega^H\mathbf{f}\mathbf{e}_1^T T_\omega(\delta_\omega)^m\mathbf{e}_1 + \left(\mathbf{e}_1^T T_\omega(\delta_\omega)^m\mathbf{e}_1\right)'\right] - \hat{\mathbf{e}}_\omega^H L_N^m\mathbf{f}\right\} + \\
&\quad \sum_{m=2K}^{\infty}\frac{(-\Delta t)^m}{m!}\left\{\left[\hat{\mathbf{e}}_\omega^H(\hat{\mathbf{e}}_\omega+\delta_\omega\mathbf{f})\mathbf{e}_1^T T_\omega(\delta_\omega)^m\mathbf{e}_1\right]' - \hat{\mathbf{e}}_\omega^H L_N^m\mathbf{f}\right\} \\
&= \sum_{m=0}^{2K-1}\frac{(-\Delta t)^m}{m!}\left\{\left[\hat{\mathbf{e}}_\omega^H\mathbf{f}\hat{\mathbf{e}}_\omega^H L_N^m\hat{\mathbf{e}}_\omega + \left(\frac{\hat{\mathbf{e}}_\omega^H L_N^m(\hat{\mathbf{e}}_\omega+\delta_\omega\mathbf{f})}{\hat{\mathbf{e}}_\omega^H(\hat{\mathbf{e}}_\omega+\delta_\omega\mathbf{f})}\right)'\right] - \hat{\mathbf{e}}_\omega^H L_N^m\mathbf{f}\right\} + \\
&\quad \sum_{m=2K}^{\infty}\frac{(-\Delta t)^m}{m!}\left\{\left[\hat{\mathbf{e}}_\omega^H(\hat{\mathbf{e}}_\omega+\delta_\omega\mathbf{f})\mathbf{e}_1^T T_\omega(\delta_\omega)^m\mathbf{e}_1\right]' - \hat{\mathbf{e}}_\omega^H L_N^m\mathbf{f}\right\} \\
&= \sum_{m=0}^{2K-1}\frac{(-\Delta t)^m}{m!}\left\{\left[\hat{\mathbf{e}}_\omega^H\mathbf{f}\hat{\mathbf{e}}_\omega^H L_N^m\hat{\mathbf{e}}_\omega + \hat{\mathbf{e}}_\omega^H L_N^m\mathbf{f} - \hat{\mathbf{e}}_\omega^H L_N^m\hat{\mathbf{e}}_\omega\hat{\mathbf{e}}_\omega^H\mathbf{f}\right] - \hat{\mathbf{e}}_\omega^H L_N^m\mathbf{f}\right\} + \\
&\quad \sum_{m=2K}^{\infty}\frac{(-\Delta t)^m}{m!}\left\{\left[\hat{\mathbf{e}}_\omega^H(\hat{\mathbf{e}}_\omega+\delta_\omega\mathbf{f})\mathbf{e}_1^T T_\omega(\delta_\omega)^m\mathbf{e}_1\right]' - \hat{\mathbf{e}}_\omega^H L_N^m\mathbf{f}\right\} \\
&= \sum_{m=2K}^{\infty}\frac{(-\Delta t)^m}{m!}\left\{\left[\hat{\mathbf{e}}_\omega^H(\hat{\mathbf{e}}_\omega+\delta_\omega\mathbf{f})\mathbf{e}_1^T T_\omega(\delta_\omega)^m\mathbf{e}_1\right]' - \hat{\mathbf{e}}_\omega^H L_N^m\mathbf{f}\right\}
\end{aligned}$$

□

3.5 Numerical Results

In [16], the practicality of the original formulation of KSS methods was demonstrated for parabolic problems by comparisons with MATLAB's stiff ODE solvers for various spatial and temporal mesh sizes. Comparisons with additional existing methods, including the Crank-Nicolson method, other finite-difference methods, and a third-order two-stage ODE solution method due to Hochbruck and Lubich (see [11]), have been detailed in [10], [12] and [14].

Here, we perform similar comparisons with the new formulation presented earlier in this section. In addition, we compare the original formulation with its reformulation. We set $p = 1$ in (4), and construct the coefficient $q(x)$ of L and initial data $f(x)$ by a procedure described in [14] that randomly generates and damps Fourier coefficients so that both functions possess 3 continuous derivatives. The output of this procedure is

$$\begin{aligned} L(x,D) &= -D^2 + q(x) \\ q(x) &= 0.47082 - 0.06299\cos x - 0.017279\sin x - 0.0054404\cos 2x + \\ &\quad 0.0015179\sin 2x - 0.0016807\cos 3x + 0.00061292\sin 3x - \\ &\quad 0.00066623\cos 4x - 0.00022637\sin 4x + \tilde{q}(x), \\ f(x) &= 0.49005 + 0.099808\cos x + 0.01893\sin x + 0.0122\cos 2x \\ &\quad + 0.0015349\sin 2x + 0.0020287\cos 3x - 5.3264e-006\sin 3x + \tilde{f}(x) \end{aligned}$$

where

$$\frac{\|\tilde{f}(x)\|_{L^2}}{\|f(x)\|_{L^2}} \le 10^{-3}, \quad \frac{\|\tilde{q}(x)\|_{L^2}}{\|q(x)\|_{L^2}} \le 10^{-3}.$$

We then apply both formulations to approximate the solution to (1), (2), (3) at $t = 1$, using 2-node Gaussian rules for each Fourier component of $u(x,t)$. We also solve this problem using two implicit ODE solvers provided in MATLAB that are designed for such stiff problems: ode23s and ode23tb. The algorithms implemented by this solver are described in [17].

Figure 1 shows that the two KSS formulations return virtually identical results, as expected. It should be noted that the original formulation uses $\delta_\omega = 10^{-5}$ for all ω, but as the number of quadrature nodes K is increased, it is advisable to use a smaller value of δ_ω, especially for higher frequencies, as higher powers of L_N are applied to the perturbation $\mathbf{u}^n$. We also see that both KSS methods exhibit greater accuracy and higher order of accuracy than either MATLAB solver.

Furthermore, they achieve such accuracy with the same time steps as those used by the implicit solvers, which far exceed the CFL limit for the forward Euler method, even though they are explicit. The error estimates, obtained from the 2-norm of the difference of successive approximations since no exact solution is available, are listed in Table 1. The MATLAB code used to generate these results, and all other results in this paper, is available at [15].

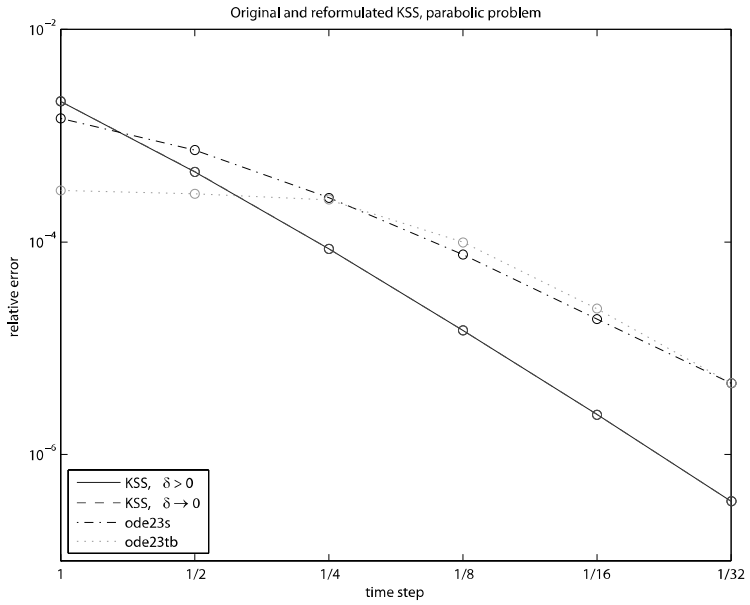


Fig. 1. Estimates of relative error in the approximate solution of problem (1), (2), (3) at $t = 1$, computed with the original KSS method from [16] with $\delta_\omega = 10^{-5}$ (solid curve), the reformulated KSS method (dashed curve), the MATLAB solver ode23s (dotted-dashed curve) and ode23tb (dotted curve). In both KSS methods, 2-point Gaussian quadrature rules are used, and $N = 64$ grid points for all methods.

Table 1. Estimates of relative error in approximate solutions to (1), (2), (3) with $L(x, D)$ as in (4), with $p = 1$. The third column lists errors for the original formulation of a KSS method from [16], the new formulation of a KSS method presented in Section 3, and the MATLAB stiff ODE solvers ode23s and ode23tb. Both KSS methods use 2-node Gaussian rules to compute each Fourier coefficient, and for all methods, $N = 64$ grid points are used to represent the solution.

Method	Δt	Error	Order
KSS(2), $\delta > 0$	1	0.0021	2.5
	1/2	0.00046	
	1/4	8.6e-005	
	1/8	1.5e-005	
	1/16	2.4e-006	
	1/32	3.6e-007	
KSS(2), $\delta \to 0$	1	0.0021	2.5
	1/2	0.00046	
	1/4	8.6e-005	
	1/8	1.5e-005	
	1/16	2.4e-006	
	1/32	3.6e-007	

ode23s	1	0.0014	1.82
	1/2	0.00073	
	1/4	0.00026	
	1/8	7.6e-005	
	1/16	1.9e-005	
	1/32	4.7e-006	
ode23tb	1	0.00031	1.91
	1/2	0.00028	
	1/4	0.00025	
	1/8	9.9e-005	
	1/16	2.4e-005	
	1/32	4.7e-006	

4 Application to the Wave Equation

In this section we apply KSS methods developed in [13] to the problem

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} + Lu = 0 & \text{in } (0,2\pi)\times R, \\ u(0,t) = u(2\pi,t) & \text{on } R, \end{cases} \tag{33}$$

with the initial conditions

$$u(x,0) = f(x), \quad u_t(x,0) = g(x), \quad x \in (0,2\pi), \tag{34}$$

where, as before, the operator L is as described in (4).

4.1 Structure of the Solution

A spectral representation of the operator L allows us the obtain a representation of the solution operator (the *propagator*) in terms of the sine and cosine families generated by L by a simple functional calculus. Introduce

$$R_1(t) = L^{-1/2}\sin(t\sqrt{L}) := \sum_{n=1}^{\infty} \frac{\sin(t\sqrt{\lambda_n})}{\sqrt{\lambda_n}} \langle \varphi_n^*, \cdot \rangle \varphi_n, \tag{35}$$

$$R_0(t) = \cos(t\sqrt{L}) := \sum_{n=1}^{\infty} \cos(t\sqrt{\lambda_n}) \langle \varphi_n^*, \cdot \rangle \varphi_n, \tag{36}$$

where $\lambda_1, \lambda_2, \ldots$ are the (positive) eigenvalues of L, and $\varphi_1, \varphi_2, \ldots$ are the corresponding eigenfunctions. Then the propagator of (33) can be written as

$$P(t) = \begin{bmatrix} R_0(t) & R_1(t) \\ -L\,R_1(t) & R_0(t) \end{bmatrix}. \tag{37}$$

The entries of this matrix, as functions of L, indicate which functions are the integrands in the Riemann-Stieltjes integrals used to compute the Fourier components of the solution.

4.2 Solution Using KSS Methods

We briefly review the use of KSS methods for solving (33), first outlined in [10].

Since the exact solution $u(x,t)$ is given by

$$u(x,t) = R_0(t)f(x) + R_1(t)g(x), \tag{38}$$

where $R_0(t)$ and $R_1(t)$ are defined in (35), (36), we can obtain $[\mathbf{u}^{n+1}]_\omega$ by approximating each of the quadratic forms

$$c_\omega^+(t) = \left\langle \hat{\mathbf{e}}_\omega, R_0(\Delta t)[\hat{\mathbf{e}}_\omega + \delta_\omega \mathbf{u}^n] \right\rangle \tag{39}$$

$$c_\omega^-(t) = \left\langle \hat{\mathbf{e}}_\omega, R_0(\Delta t)\hat{\mathbf{e}}_\omega \right\rangle \tag{40}$$

$$s_\omega^+(t) = \left\langle \hat{\mathbf{e}}_\omega, R_1(\Delta t)[\hat{\mathbf{e}}_\omega + \delta_\omega \mathbf{u}_t^n] \right\rangle \tag{41}$$

$$s_\omega^-(t) = \left\langle \hat{\mathbf{e}}_\omega, R_1(\Delta t)\hat{\mathbf{e}}_\omega \right\rangle, \tag{42}$$

where δ_ω is a nonzero constant. It follows that

$$[\hat{\mathbf{u}}^{n+1}]_\omega = \frac{c_\omega^+(t) - c_\omega^-(t)}{\delta_\omega} + \frac{s_\omega^+(t) - s_\omega^-(t)}{\delta_\omega}. \tag{43}$$

Similarly, we can obtain the coefficients $\tilde{v}_\omega$ of an approximation of $u_t(x,t)$ by approximating the quadratic forms

$$c_\omega^+(t)' = -\left\langle \hat{\mathbf{e}}_\omega, LR_1(\Delta t)[\hat{\mathbf{e}}_\omega + \delta_\omega \mathbf{u}^n] \right\rangle \tag{44}$$

$$c_\omega^-(t)' = -\left\langle \hat{\mathbf{e}}_\omega, LR_1(\Delta t)\hat{\mathbf{e}}_\omega \right\rangle \tag{45}$$

$$s_\omega^+(t)' = \left\langle \hat{\mathbf{e}}_\omega, R_0(\Delta t)[\hat{\mathbf{e}}_\omega + \delta_\omega \mathbf{u}_t^n] \right\rangle \tag{46}$$

$$s_\omega^-(t)' = \left\langle \hat{\mathbf{e}}_\omega, R_0(\Delta t)\hat{\mathbf{e}}_\omega \right\rangle. \tag{47}$$

As noted in [13], this approximation to $u_t(x,t)$ does not introduce *any* error due to differentiation of our approximation of $u(x,t)$ with respect to t –the latter approximation can be differenti-

ated *analytically*.

It follows from the preceding discussion that we can compute an approximate solution $\tilde{u}(x,t)$ at a given time T using the following algorithm.

Algorithm 2. (KSS for the wave equation, original formulation) *Given functions* $c(x)$, $f(x)$, *and* $g(x)$ *defined on the interval* $(0,2\pi)$, *and a final time* T, *the following algorithm from [10] computes a function* $\tilde{u}(x,t)$ *that approximately solves the problem (33), (34) from* $t=0$ *to* $t=T$.

$t=0$
while $t<T$ **do**
  Select a time step Δt
  $f(x)=\tilde{u}(x,t)$
  $g(x)=\tilde{u}_t(x,t)$
  for $\omega=-N/2+1$ **to** $N/2$ **do**
    Choose a nonzero constant δ_ω
    Compute the quantities $c_\omega^+(\Delta t)$, $c_\omega^-(\Delta t)$, $s_\omega^+(\Delta t)$, $s_\omega^-(\Delta t)$,
      $c_\omega^+(\Delta t)'$, $c_\omega^-(\Delta t)'$, $s_\omega^+(\Delta t)'$, *and* $s_\omega^-(\Delta t)'$
    $\tilde{u}_\omega(\Delta t)=\frac{1}{\delta_\omega}(c_\omega^+(\Delta t)-c_\omega^-(\Delta t))+\frac{1}{\delta_\omega}(s_\omega^+(\Delta t)-s_\omega^-(\Delta t))$
    $\tilde{v}_\omega(\Delta t)=\frac{1}{\delta_\omega}(c_\omega^+(\Delta t)'-c_\omega^-(\Delta t)')+\frac{1}{\delta_\omega}(s_\omega^+(\Delta t)'-s_\omega^-(\Delta t)')$
  end
  $\tilde{u}(x,t+\Delta t)=\sum_{\omega=1}^{N}\hat{\mathbf{e}}_\omega(x)\tilde{u}_\omega(\Delta t)$
  $\tilde{u}_t(x,t+\Delta t)=\sum_{\omega=1}^{N}\hat{\mathbf{e}}_\omega(x)\tilde{v}_\omega(\Delta t)$
  $t=t+\Delta t$
end

In this algorithm, each of the quantities inside the **for** loop are computed using K quadrature nodes. The nodes and weights are obtained in exactly the same way as for the parabolic problem (1), (2), (3). It should be noted that although 8 bilinear forms are required for each wave number ω, only three sets of nodes and weights need to be computed, and then they are used with different integrands.

4.3 Reformulation

Following the reformulation of KSS methods presented in Section 3, we let $\delta_\omega \to 0$ to obtain

$$\begin{bmatrix} \hat{\mathbf{u}}^{n+1} \\ \hat{\mathbf{u}}_t^{n+1} \end{bmatrix}_\omega = \left(\sum_{k=1}^{K} w_k \begin{bmatrix} \cos(\sqrt{\lambda_k}t) & \frac{1}{\sqrt{\lambda_k}}\sin(\sqrt{\lambda_k}t) \\ -\sqrt{\lambda_k}\sin(\sqrt{\lambda_k}t) & \cos(\sqrt{\lambda_k}t) \end{bmatrix} \right) \begin{bmatrix} \hat{\mathbf{e}}_\omega^H \mathbf{u}^n \\ \hat{\mathbf{e}}_\omega^H \mathbf{u}_t^n \end{bmatrix} +$$
$$\sum_{k=1}^{K} \begin{bmatrix} \cos(\sqrt{\lambda_k}t) & \frac{1}{\sqrt{\lambda_k}}\sin(\sqrt{\lambda_k}t) \\ -\sqrt{\lambda_k}\sin(\sqrt{\lambda_k}t) & \cos(\sqrt{\lambda_k}t) \end{bmatrix} \begin{bmatrix} w_k' \\ \tilde{w}'_k \end{bmatrix} -$$
$$\sum_{k=1}^{K} w_k \frac{t}{2\sqrt{\lambda_k}} \begin{bmatrix} \sin(\sqrt{\lambda_k}t) & -\frac{1}{\sqrt{\lambda_k}}\cos(\sqrt{\lambda_k}t) \\ \sqrt{\lambda_k}\cos(\sqrt{\lambda_k}t) & \sin(\sqrt{\lambda_k}t) \end{bmatrix} \begin{bmatrix} \lambda_k' \\ \tilde{\lambda}'_k \end{bmatrix} - \tag{48}$$
$$w_k \begin{bmatrix} 0 & \frac{1}{2(\lambda_k)^{3/2}}\sin(\sqrt{\lambda_k}t) \\ \frac{1}{2\sqrt{\lambda_k}}\sin(\sqrt{\lambda_k}t) & 0 \end{bmatrix} \begin{bmatrix} \lambda_k' \\ \tilde{\lambda}'_k \end{bmatrix}$$

where λ_k' and w_k' are the derivatives of the nodes and weights, respectively, in the direction of $\mathbf{u}^n$, and $\tilde{\lambda}'_k$ and $\tilde{w}'_k$ are the derivatives in the direction of $\mathbf{u}_t^n$.

To discuss local truncation error, we first recall a result concerning the accuracy of each component of the approximate solution produced by Algorithm 2.

Theorem 3. *Assume that $f(x)$ and $g(x)$ satisfy (3), and let $u(x,\Delta t)$ be the exact solution of (33), (34) at $(x,\Delta t)$, and let $\tilde{u}(x,\Delta t)$ be the approximate solution computed by Algorithm 2. Then*

$$|\langle \hat{\mathbf{e}}_\omega, u(\cdot,\Delta t) - \tilde{u}(\cdot,\Delta t)\rangle| = O(\Delta t^{4K}) \tag{49}$$

where K is the number of quadrature nodes used in Algorithm 2.

Proof. See [10].

The approach used in the proof of Theorem 2 can be applied to the reformulated KSS method (48) to show that it achieves the same order of accuracy in time as Algorithm 2.

In the case $K = 1$, (48) reduces to

$$\begin{bmatrix}\hat{\mathbf{u}}^{n+1}\\ \hat{\mathbf{u}}_t^{n+1}\end{bmatrix}_\omega = \begin{bmatrix} c_\omega & \frac{1}{\sqrt{\alpha_1}}s_\omega \\ -\sqrt{\alpha_1}s_\omega & c_\omega \end{bmatrix}\begin{bmatrix}\hat{\mathbf{e}}_\omega^H\mathbf{u}^n\\ \hat{\mathbf{e}}_\omega^H\mathbf{u}_t^n\end{bmatrix} - \frac{\Delta t}{2\sqrt{\alpha_1}}\begin{bmatrix} s_\omega & -\frac{1}{\sqrt{\alpha_1}}c_\omega \\ \sqrt{\alpha_1}c_\omega & s_\omega \end{bmatrix}\begin{bmatrix}\hat{\mathbf{e}}_\omega^H V_N\mathbf{u}^n\\ \hat{\mathbf{e}}_\omega^H V_N\mathbf{u}_t^n\end{bmatrix} - \begin{bmatrix} 0 & \frac{1}{2(\alpha_1)^{3/2}}s_\omega \\ \frac{1}{2\sqrt{\alpha_1}}s_\omega & 0 \end{bmatrix}\begin{bmatrix}\hat{\mathbf{e}}_\omega^H V_N\mathbf{u}^n\\ \hat{\mathbf{e}}_\omega^H V_N\mathbf{u}_t^n\end{bmatrix} \tag{50}$$

where we use the splitting $L(x,D) = C(x,D) + V(x,D)$ as in Section 3, with corresponding spectral discretizations L_N, C_N and V_N. The first two terms in (50) yield the Fourier component $[\hat{\mathbf{v}}^1]_\omega$ of the exact solution at time Δt to the constant-coefficient problem

$$\frac{\partial^2 v}{\partial t^2} + C(x,D)v = 0, \tag{51}$$

$$v(x,0) = u(x,t_n) + \frac{\Delta t}{2}P_N C(x,D)^{-1}V(x,D)u_t(x,t_n), \tag{52}$$

$$v_t(x,0) = u_t(x,t_n) - \frac{\Delta t}{2}P_N V(x,D)u(x,t_n). \tag{53}$$

In [12], it was shown that this splitting, which is 3rd-order accurate in time, is unconditionally stable when $q(x)$ is bandlimited.

4.4 Numerical Results

We now compare the original formulation of KSS methods as presented in [16] with its reformulation as presented in Section 3, with the appropriate integrands used in place of $e^{-\lambda t}$. We construct the operator L and initial data $f(x)$ and $g(x)$ as in Section 3.5, to obtain

$$L(x,D) = -D^2 + q(x) \tag{54}$$

$$q(x) \approx 0.47089 - 0.06438\cos x + 0.016734\sin x - 0.002807\cos 2x + 0.0030454\sin 2x - 0.0012447\cos 3x + 0.00042544\sin 3x \tag{55}$$

$$f(x) \approx 0.49005 + 0.099808\cos x + 0.01893\sin x + 0.0122\cos 2x + 0.0015349\sin 2x + 0.0020287\cos 3x - 5.3264e-006\sin 3x \tag{56}$$

$$g(x) \approx 0.46063 + 0.06299\cos x + 0.017279\sin x + 0.0054404\cos 2x - 0.0015179\sin 2x + 0.0016807\cos 3x - 0.00061292\sin 3x + 0.00066623\cos 4x + 0.00022637\sin 4x \tag{57}$$

We then apply both formulations to approximate the solution to (33), (34) at $t = 1$, using 2-node Gaussian rules for each Fourier component of $u(x,t)$ and $u_t(x,t)$. As before, $N = 64$ grid points are used.

Because this KSS method is 7th-order accurate in time, very high accuracy is achieved, but as $\Delta t \to 0$, this order of convergence is not maintained by the original formulation due to catastrophic cancellation in the final step of the computation of each component. Because this subtraction is not performed in the new formulation as an implicitly-defined operator splitting, the order of convergence is maintained by this reformulation for smaller values of Δt, as we can see in Figure 2. The error estimates are also listed in Table 2.

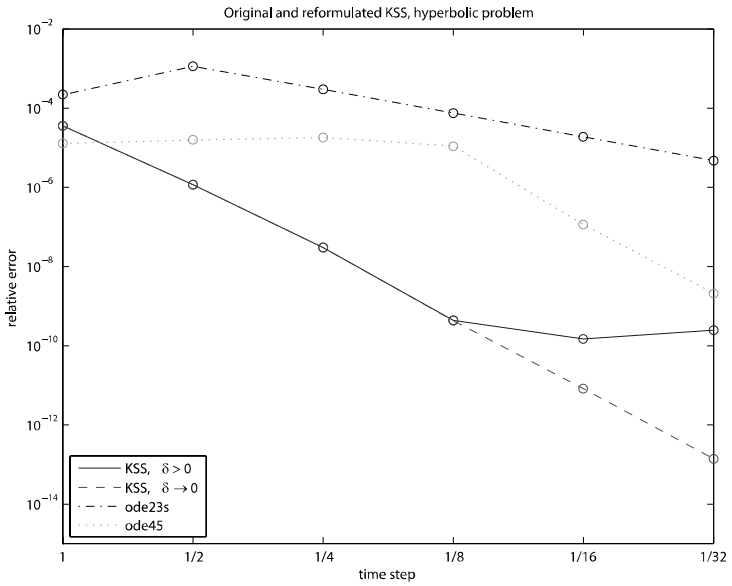


Fig. 2. Estimates of relative error in the approximate solution of problem (33), (34) at $t = 1$, computed with the original KSS method from [16] with $\delta_\omega = 10^{-5}$ (solid curve), the reformulated KSS method (dashed curve), the MATLAB solver ode23s (dotted-dashed curve), and ode45 (dotted curve). In both KSS methods, 2-node Gaussian quadrature rules are used, and $N = 64$ grid points are used for all methods.

Figure 2 and Table 2 also report the results of solving this problem using two of MATLAB's ODE solvers, ode23s and ode45, a higher-order accurate explicit solver that, while not a practical option for a parabolic problem, is much better suited to this hyperbolic one. While ode45 exhibits slightly higher-order accuracy in this case, it is unable to achieve reasonable accuracy for larger time steps, and this threshold decreases as the number of grid points increases due to the stiffness

of the problem, while KSS methods do not exhibit such sensitivity, even though they too are explicit. Table 3 illustrates this difference in behavior by solving the same problem with $N = 128$ grid points.

Table 2. Estimates of relative error in approximate solutions to (33), (34) with $L(x, D)$ as in (4), with $p = 1$ and $q(x)$ as defined in (55). The third column lists errors for the original formulation of a KSS method from [16], the new formulation of a KSS method presented in Section 3, and the MATLAB solvers ode23s and ode45. Both KSS methods use 2-node Gaussian rules to compute each Fourier coefficient, and for all methods, $N = 64$ grid points are used to represent the solution.

Method	Δt	Error	Order
KSS(2), $\delta > 0$	1	3.6e-005	5.44
	1/2	1.2e-006	
	1/4	3e-008	
	1/8	4.4e-010	
	1/16	1.5e-010	
	1/32	2.5e-010	
KSS(2), $\delta \to 0$	1	3.6e-005	5.59
	1/2	1.2e-006	
	1/4	3e-008	
	1/8	4.3e-010	
	1/16	8.3e-012	
	1/32	1.4e-013	
ode23s	1	0.00022	1.98
	1/2	0.0012	
	1/4	0.0003	
	1/8	7.5e-005	
	1/16	1.9e-005	
	1/32	4.7e-006	
ode45	1	1.3e-005	6.18
	1/2	1.6e-005	
	1/4	1.8e-005	
	1/8	1.1e-005	
	1/16	1.1e-007	
	1/32	2.1e-009	

Table 3. Estimates of relative error in approximate solutions to (33), (34) with $L(x,D)$ as in (4), with $p=1$ and $q(x)$ as defined in (55). The third column lists errors for the new formulation of a KSS method presented in Section 3, and the MATLAB solver ode45. The KSS method uses 2-node Gaussian rules to compute each Fourier coefficient, and for botoh methods, $N=128$ grid points are used to represent the solution.

Method	Δt	Error
KSS(2), $\delta \to 0$	1	2.2e-005
	1/2	6.8e-007
	1/4	2.0e-008
	1/8	5.0e-010
	1/16	6.3e-012
	1/32	1.2e-013
ode45	1	2.8e-005
	1/2	1.8e-005
	1/4	2.8e-005
	1/8	3.0e-005
	1/16	1.4e-005
	1/32	1.7e-008

5 Generalization to Systems of Equations

Now, we consider the following initial-boundary value problem in one space dimension,

$$\frac{\partial \mathbf{u}}{\partial t}(x,t)+L(x,D)\mathbf{u}(x,t)=0, \quad 0<x<2\pi, \quad t>0, \tag{58}$$

$$\mathbf{u}(x,0)=\mathbf{f}(x), \quad 0<x<2\pi, \tag{59}$$

with periodic boundary conditions

$$\mathbf{u}(0,t)=\mathbf{u}(2\pi,t), \quad t>0, \tag{60}$$

where $\mathbf{u}:[0,2\pi]\times[0,\infty)\to R^n$ for $n>1$, and $L(x,D)$ is an $n\times n$ matrix where the (i,j) entry is a differential operator $L_{ij}(x,D)$ of the form

$$L_{ij}(x,D)u(x)=\sum_{\mu=0}^{m_{ij}} a_{\mu}^{ij}(x)D^{\mu}u, \quad D=\frac{d}{dx}, \tag{61}$$

with spatially varying coefficients a_{μ}^{ij}, $\mu=0,1,\ldots,m_{ij}$.

Generalization of KSS methods to a system of the form (58) can proceed as follows. For $i,j=1,\ldots,n$, let $\overline{L}_{ij}(D)$ be the constant-coefficient operator obtained by averaging the coefficients of $L_{ij}(x,D)$ over $[0,2\pi]$. Then, for each wave number ω, we define $L(\omega)$ be the matrix

with entries $\overline{L}_{ij}(\omega)$, i.e., the symbols of $\overline{L}_{ij}(D)$ evaluated at ω. Next, we compute the Schur decomposition of $L(\omega)$ for each ω. For $j = 1,\ldots,n$, let $\mathbf{q}_j(\omega)$ be the Schur vectors of $L(\omega)$. Then, we define our trial and test functions by $\mathbf{q}_j(\omega) \otimes e^{i\omega x}$.

As in the scalar case, this approach yields $O(\Delta t^{2K})$ local temporal error, where K is the number of nodes in each Gaussian quadrature rule. Furthermore, although the quadrature rules described in [6], and employed in KSS methods, were designed for use with self-adjoint positive semi-definite operators, the same high-order accuracy can also be achieved with non-self-adjoint differential operators, provided that the Lanczos iteration does not break down, which is unlikely in practice, considering that small values of K are typically used. Justification for the application of the Lanczos algorithm to construct Gaussian quadrature rules for Riemann-Stieltjes integrals of the form (15) in the non-self-adjoint case is presented in [13, section 6.1.2].

The recursion coefficients, nodes and weights can be computed in the same manner as in the scalar, self-adjoint case, with obvious modifications to account for the fact that the matrix $T_\omega(\delta_\omega)$, for each ω, is no longer Hermitian. The result is an implicitly-defined operator splitting of the form (22) that, as in the scalar case, is the limit of a KSS method, based on its original formulation, as $\delta_\omega \to 0$ for each ω.

We apply this approach to a parabolic system of the form (58) with $n = 2$. Using the same function-generation procedure from [14] that we applied in Section 3.5, we construct the operator $L(x,D)$ from (61) as follows. For $i = 1,2$, $L_{ii}(x,D)$ is set equal to a self-adjoint positive definite second-order differential operator whose zeroth-order coefficient is constructed to have the smoothness of a function with 3 continuous derivatives, and $L_{i,3-i}(x,D)$ set equal to a first-order differential operator with variable coefficients that are constructed so as to possess 4 continuous derivatives. We have

$$\begin{aligned} L_{11}(x,D) &= a_2(x)D^2 + a_0(x) \\ a_0(x) &\approx 0.50344 - 0.075855\cos x + 0.060748\sin x - 0.0069633\cos 2x + \\ &\quad 0.0059539\sin 2x - 0.00070443\cos 3x + 2.8555e-005\sin 3x \\ a_2(x) &= -0.39894 \end{aligned} \tag{62}$$

$$\begin{aligned} L_{12}(x,D) &= b_1(x)D + b_0(x) \\ b_0(x) &\approx 0.48617 - 0.080102\cos x + 0.023861\sin x - 0.0032964\cos 2x + \\ &\quad 0.0006793\sin 2x \\ b_1(x) &\approx -0.51544 - 0.0776\cos x + 0.09141\sin x - 0.0007574\cos 2x + \\ &\quad 0.0035344\sin 2x \end{aligned} \tag{63}$$

$$\begin{aligned} L_{21}(x,D) &= c_1(x)D + c_0(x) \\ c_0(x) &\approx 0.52236 - 0.12017\cos x + 0.0090074\sin x - 0.0021617\cos 2x + \\ &\quad 0.0011406\sin 2x \\ c_1(x) &\approx -0.48283 - 0.039285\cos x + 0.076048\sin x - 0.00068367\cos 2x + \\ &\quad 0.0024259\sin 2x \end{aligned} \tag{64}$$

$$
\begin{aligned}
L_{22}(x,D) &= d_2(x)D^2 + d_0(x) \\
d_0(x) &\approx 0.48568 - 0.0061309\cos x + 0.094417\sin x - 0.0069907\cos 2x + \\
&\quad 0.0022355\sin 2x - 0.00038766\cos 3x + 0.0014394\sin 3x \\
d_2(x) &= -0.39894
\end{aligned}
\tag{65}
$$

and initial data

$$
\begin{aligned}
\mathbf{f}(x) &= \begin{bmatrix} f(x) \\ g(x) \end{bmatrix} \\
f(x) &= 0.54202 + 0.070953\cos x - 0.12256\sin x + 0.0061862\cos 2x - \\
&\quad 0.0011921\sin 2x + 0.0012855\cos 3x - 0.00029609\sin 3x \\
g(x) &= 0.48023 + 0.039719\cos x - 0.073462\sin x + 0.0010172\cos 2x - \\
&\quad 0.0019871\sin 2x + 0.0012392\cos 3x - 0.0010306\sin 3x.
\end{aligned}
\tag{66}
$$

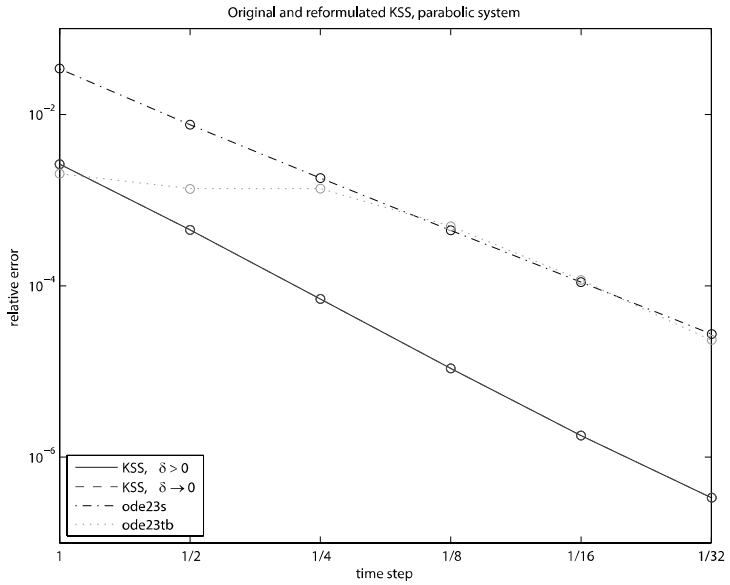


Fig. 3. Estimates of relative error in the approximate solution of problem (33), (34) at $t = 1$, computed with the original KSS method from [16] with $\delta_\omega = 10^{-5}$ (solid curve), the reformulated KSS method (dashed curve), the MATLAB solver ode23s (dotted-dashed curve) and ode23tb (dotted curve). In both KSS methods, 2-node Gaussian quadrature rules are used, and $N = 64$ grid points are used for all methods.

In Table 4, the relative error, estimated by taking the vector 2-norm of the difference between successive approximations, is given for both the original formulation of a 2-node KSS method, as presented in [16], and the new formulation of the same method presented in Section 3. As desired, the results for the two KSS methods are nearly identical, and as in the scalar case, superior accu-

racy and order of convergence is obtained compared to the two MATLAB solvers.

Table 4. Estimates of relative error in approximate solutions to (58), (59), (60) with $n = 2$ and $L_{ij}(x, D) = 0$ as defined in (62)-(65). The third column lists errors for the original formulation of a KSS method from [16], the new formulation of a KSS method presented in Section 3, and the MATLAB stiff ODE solvers ode23s and ode23tb. Both KSS methods use 2-node Gaussian rules to compute each Fourier coefficient, and for all methods, $N = 64$ grid points are used to represent each component of the solution $\mathbf{u}(t)$.

Method	Δt	Error	Order
KSS(2), $\delta > 0$	1	0.0026	2.59
	1/2	0.00045	
	1/4	7e-005	
	1/8	1.1e-005	
	1/16	1.8e-006	
	1/32	3.4e-007	
KSS(2), $\delta \to 0$	1	0.0026	2.59
	1/2	0.00045	
	1/4	7e-005	
	1/8	1.1e-005	
	1/16	1.8e-006	
	1/32	3.4e-007	
ode23s	1	0.034	2.06
	1/2	0.0075	
	1/4	0.0018	
	1/8	0.00044	
	1/16	0.00011	
	1/32	2.7e-005	
ode23tb	1	0.002	1.96
	1/2	0.0014	
	1/4	0.0014	
	1/8	0.00049	
	1/16	0.00012	
	1/32	2.3e-005	

6 Discussion

In this concluding section, we consider various generalizations of the problems and methods considered in this paper.

6.1 Higher Space Dimension

In [16], it is demonstrated how to compute the recursion coefficients α_j and β_j for operators of the form $Lu = -p\Delta u + q(x,y)u$, and the expressions are straightforward generalizations of the expressions given in [12] for the one-dimensional case. The derivatives of the nodes and weights can then be computed in exactly the same way as in the one-dimensional case, resulting in an implicitly-defined high-order operator splitting of the form (22).

6.2 Greater Variation in Coefficients

In [10], [14], it is shown that the consistency results in Theorems 1 and 3 apply when the leading coefficient is also dependent on x. In [12], it is shown that the stability results for the case of $K = 1$ can be generalized to operators with variable leading coefficients by means of a simple homogenizing transformation. In [16], it is demonstrated when the leading coefficient varies with x, KSS methods achieve the same accuracy as for problems in which only lower-order coefficients vary, but this accuracy depends on the smoothness of the coefficient. Furthermore, it is shown there that discontinuous coefficients pose difficulties, due to the loss of information resulting from truncation of Fourier series. Current work is focusing on the use of reprojection, such as Freud reprojection presented in [3], to compute the recursion coefficients with greater accuracy.

6.3 Summary

We have demonstrated that for both parabolic and hyperbolic variable-coefficient PDE, a reformulation of KSS methods that eliminates the need to perturb quadrature rules not only improves their numerical stability, but also reveals that these methods are actually implicitly-defined high-order operator splittings. Although the stages of the splitting are not easily described, efficient algorithms for computing appropriate derivatives of the nodes and weights allow efficient implementation of these methods. We have also shown that KSS methods can also be readily generalized to systems of coupled PDE through an appropriate choice of trial functions, even if the operator $L(x,D)$ is not self-adjoint. Future work will explore the analysis, and enhancement, of KSS methods through closer examination of the splittings they define. Also, because of the improved numerical stability of these methods, using a larger number of nodes is now more viable than in the original formulation.

References

1. Atkinson, K.: *An Introduction to Numerical Analysis*, 2nd Ed. Wiley. (1989)

2. Dahlquist, G., Eisenstat, S. C., Golub, G. H.: Bounds for the Error of Linear Systems of Equations Using the Theory of Moments. *Journal of Mathematical Analysis and Applications* 37 (1972) 151-166.
3. Gelb, A., Tanner, J.: Robust Reprojection Methods for the Resolution of the Gibbs Phenomenon. *Applied and Computational Harmonic Analysis* 20 (2006) 3-25.
4. Golub, G. H.: Some Modified Matrix Eigenvalue Problems. *SIAM Review* 15 (1973) 318-334.
5. Golub, G. H.: Bounds for Matrix Moments. *Rocky Mountain Journal of Mathematics* 4 (1974) 207-211.
6. Golub, G. H., Meurant, G.: Matrices, Moments and Quadrature. *Proceedings of the 15th Dundee Conference*, June-July 1993, Griffiths, D. F., Watson, G. A. (eds.), Longman Scientific & Technical. (1994)
7. Golub, G. H., Gutknecht, M. H.: Modified Moments for Indefinite Weight Functions. *Numerische Mathematik* 57 (1989) 607-624.
8. Golub, G. H., van Loan, C. F.: *Matrix Computations*, 3rd Ed. Johns Hopkins University Press. (1996)
9. Golub, G. H, Welsch, J.: Calculation of Gauss Quadrature Rules. *Math. Comp.* 23 (1969) 221-230.
10. Guidotti, P., Lambers, J. V., Sølna, K.: Analysis of 1-D Wave Propagation in Inhomogeneous Media. *Numerical Functional Analysis and Optimization* 27 (2006) 25-55.
11. Hochbruck, M., Lubich, C.: On Krylov Subspace Approximations to the Matrix Exponential Operator. *SIAM Journal of Numerical Analysis* 34 (1996) 1911-1925.
12. Lambers, J. V.: Derivation of High-Order Spectral Methods for Time-dependent PDE Using Modified Moments. *Electronic Transactions on Numerical Analysis* 28 (2007). to appear.
13. Lambers, J. V.: Krylov SubspaceMethods for Variable-Coefficient Initial-Boundary Value Problems. Ph.D. Thesis, Stanford University, SCCM Program, 2003. Available at http://sccm.stanford.edu/pub/sccm/theses/James Lambers.pdf
14. Lambers, J. V.: Krylov Subspace Spectral Methods for Variable-Coefficient Initial-Boundary Value Problems. *Electronic Transactions on Numerical Analysis* 20 (2005) 212-234.
15. Lambers, J. V.: MATLAB Implementation of Algorithms Described in This Paper. http://www.stanford.edu/_lambers/splittings
16. Lambers, J. V.: Practical Implementation of Krylov Subspace Spectral Methods. *Journal of Scientific Computing* 32 (2007) 449-476.
17. Shampine, L. F., Reichelt, M. W.: The MATLAB ODE Suite. *SIAM Journal of Scientific Computing* 18 (1997) 1-22.
18. Strang, G.: On the Construction and Comparison of Difference Schemes. *SIAM Journal of Numerical Analysis* 5 (1968) 506-517.