SIAM Review
Vol. 55, No. 3, pp. 462–489

A Fast and Well-Conditioned Spectral Method*

Sheehan Olver†
Alex Townsend‡

Abstract. A spectral method is developed for the direct solution of linear ordinary differential equations with variable coefficients and general boundary conditions. The method leads to matrices that are almost banded, and a numerical solver is presented that takes $\mathcal{O}(m^2n)$ operations, where m is the number of Chebyshev points needed to resolve the coefficients of the differential operator and n is the number of Chebyshev coefficients needed to resolve the solution to the differential equation. We prove stability of the method by relating it to a diagonally preconditioned system that has a bounded condition number, in a suitable norm. For Dirichlet boundary conditions, this implies stability in the standard 2-norm. An adaptive QR factorization is developed to efficiently solve the resulting linear system and automatically choose the optimal number of Chebyshev coefficients needed to represent the solution. The resulting algorithm can efficiently and reliably solve for solutions that require as many as a million unknowns.

Key words. spectral method, ultraspherical polynomials, adaptive direct solver

AMS subject classifications. 65N35, 65L10, 33C45

DOI. 10.1137/120865458

1. Introduction.

Spectral methods are an important tool in scientific computing and engineering for solving differential equations (see, for instance, [6, 21, 23, 44, 45]). Although the computed solutions can converge superalgebraically to the solution of the differential equation, conventional wisdom states that spectral methods lead to dense, ill-conditioned matrices. In this paper, we introduce a spectral method that continues to converge superalgebraically to the solution, but only requires solving an almost banded, well-conditioned linear system.

Throughout, we consider the family of linear differential equations on $[-1, 1]$:

$$\mathcal{L}u = f \qquad \text{and} \qquad \mathcal{B}u = \mathbf{c}, \tag{1.1}$$

where $\mathcal{L}$ is an Nth order linear differential operator

$$\mathcal{L}u = a^N(x)\frac{\mathrm{d}^N u}{\mathrm{d}x^N} + \cdots + a^1(x)\frac{\mathrm{d}u}{\mathrm{d}x} + a^0(x)u,$$

$\mathcal{B}$ denotes K boundary conditions (Dirichlet, Neumann, etc.), $\mathbf{c} \in \mathbb{C}^K$, and $a^0, \ldots, a^N$, f are suitably smooth functions on $[-1, 1]$. We make the assumption that the dif-

*Received by the editors February 9, 2012; accepted for publication (in revised form) January 2, 2013; published electronically August 8, 2013.
http://www.siam.org/journals/sirev/55-3/86545.html

†School of Mathematics and Statistics, The University of Sydney, Sydney, Australia (Sheehan.Olver@sydney.edu.au).
‡Mathematical Institute, 24-29 St Giles', Oxford, OX1 3LB, UK (townsend@maths.ox.ac.uk).

ferential equation is not singular; i.e., the leading variable coefficient $a^N(x)$ does not vanish on the interval $[-1, 1]$.

Within spectral methods, there is a subdivision between collocation methods and coefficient methods: the former construct matrices operating on the values of a function at, say, Chebyshev points, and the latter construct matrices operating on coefficients in a basis, say, of Chebyshev polynomials. Here there is a common belief that collocation methods are more adaptable to differential equations with variable coefficients [4, section 9.2], i.e., when $a^0(x), \ldots, a^N(x)$ are not constant. However, the spectral coefficient method that we construct is equally applicable to differential equations with variable coefficients.

For example, the first order differential equation (chosen so the entries in the resulting linear system are integers)

$$\frac{\mathrm{d}u}{\mathrm{d}x} + 4xu = 0 \qquad \text{and} \qquad u(-1) = c \tag{1.2}$$

results in our spectral method forming the *almost banded* $n \times n$ linear system

$$\begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 & \cdots & (-1)^{n-1} \\ & 2 & & -1 & & & & \\ 2 & & 2 & & -1 & & & \\ & 1 & & 3 & & -1 & & \\ & & \ddots & & \ddots & & \ddots & \\ & & & 1 & & n-3 & & -1 \\ & & & & 1 & & n-2 & \\ & & & & & 1 & & n-1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \\ \vdots \\ \vdots \\ \\ u_{n-1} \end{pmatrix} = \begin{pmatrix} c \\ 0 \\ 0 \\ \\ \vdots \\ \vdots \\ \\ 0 \end{pmatrix}. \tag{1.3}$$

We then approximate the solution to (1.2) as

$$u(x) \approx \sum_{k=0}^{n-1} u_k T_k(x),$$

where T_k is the Chebyshev polynomials (of the first kind) of degree k. Moreover, the stability of solving (1.3) is directly related to that of a diagonally preconditioned matrix system that has 2-norm condition number bounded above by 53.6 for all n.

Our method is based on the following:

1. Representing derivatives in terms of *ultraspherical polynomials*. This results in diagonal differentiation matrices.
2. Representing conversion from Chebyshev polynomials to ultraspherical polynomials by a *banded* operator.
3. Representing multiplication for variable coefficients by banded operators in coefficient space. This is achieved by approximating the variable coefficients by a truncated Chebyshev (and thence ultraspherical) series.
4. Imposing the boundary conditions using *boundary bordering*; that is, K rows of the linear system are used to impose K boundary conditions. For (1.2), the last row of the linear system has been replaced and then permuted to the first row. This allows for a convenient solution basis to be used for general boundary conditions.

5. Using an *adaptive QR factorization* to determine the optimal truncation denoted by n_{opt}. We develop a sparse representation for the resulting dense matrix, allowing for efficient back substitution.

The resulting stable method requires only $\mathcal{O}(n_{\text{opt}})$ operations to solve the resulting linear systems and calculate the Chebyshev coefficients of the solution. (A further $\mathcal{O}(n_{\text{opt}} \log n_{\text{opt}})$ operations would be required to convert from coefficients to values at n_{opt} Chebyshev points.) The algorithm calculates the Chebyshev coefficients of the solution (as well as its derivatives) to machine precision. A striking application of the method is to singularly perturbed differential equations, where the bandwidth of the linear system is uniformly bounded, and the accuracy is unaffected by the extremely large condition numbers.

This paper is organized as follows. We first provide an overview of existing spectral methods. In section 2, we construct the method for first order differential equations and apply it to problems with oscillatory and stiff coefficients. In the third section, we extend the approach to higher order differential equations by using ultraspherical polynomials, and present numerical results of the method applied to challenging second order and higher order differential equations. In the fourth section, we prove stability and convergence of the method in high order norms, which reduce to the standard 2-norm for Dirichlet boundary conditions. In section 5, we present a fast, stable algorithm to solve the almost banded linear systems in $\mathcal{O}(n_{\text{opt}})$ operations, where n_{opt} is automatically determined, allowing for the efficient solution of differential equations that require as many as a million unknowns (see Figure 5.2). In the final section we describe directions for future research.

Remark 1. The MATLAB and C++ code used for the numerical results is available from [35].

1.1. Existing Techniques.

Chebyshev polynomials or, more generally, Jacobi polynomials have been abundantly used to construct spectral coefficient methods. In this section we briefly survey some existing techniques.

1.1.1. Tau-method.

The tau-method was originally proposed by Lanczos [28] and is analogous to representing differentiation as an operator on Chebyshev coefficients [4]. The resulting linear system, which is constructed by using recurrence relationships between Chebyshev polynomials, is always dense—even for differential equations with constant coefficients—and is typically ill-conditioned. The boundary conditions are imposed by replacing the last few rows of the linear system with entries constraining the solution's coefficients.

This method was made popular and extended by Ortiz and his colleagues [23, 38] and can be made into an automated black-box solver, but this is more due to how the boundary conditions are imposed rather than the specifics of this approach. Therefore, we will use the same technique called *boundary bordering* for the boundary conditions.

1.1.2. Basis Recombination.

An alternative to boundary bordering is to impose the boundary conditions by *basis recombination*. That is, for our simple example (1.2), by computing the coefficients in the expansion

$$u(x) = cT_0(x) + \sum_{k=1}^{\infty} \tilde{u}_k \phi_k(x),$$

where

$$\phi_k(x) = \begin{cases} T_k(x) - T_0(x), & k \text{ even}, \\ T_k(x) + T_0(x), & k \text{ odd}. \end{cases}$$

The basis is chosen so that $\phi_k(-1) = 0$, and there are many other possible choices.

Basis recombination runs counter to an important observation of Orszag [37]: Spectral methods in a convenient basis (like Chebyshev) can outperform an awkward problem-dependent basis. The problem-dependent basis may be theoretically elegant, but convenience is worth more in practice. In detail, the benefits of using boundary bordering instead of standard basis recombination include the following: (1) the solution is always computed in the convenient and orthogonal Chebyshev basis; (2) a fixed basis means we can automatically apply recurrence relations between Chebyshev polynomials (and later, ultraspherical polynomials) to construct multiplication matrices for incorporating variable coefficients; (3) the structure of the linear systems does not depend on the boundary condition(s), allowing for a fast, general solver; and (4) while basis recombination can result in well-conditioned linear systems, the solution can be in terms of an unstable basis for very high order boundary conditions.

These issues can be mitigated for special boundary conditions—such as Dirichlet—by using alternative recombinations; see [41, 42, 43, 44]. Such basis recombinations can be incorporated into our approach for variable coefficients and adaptive truncation to achieve many of the advantages of our method. However, we use boundary bordering to avoid limitations on which boundary conditions are permitted.

There is one negative consequence of boundary bordering: it results in nonsymmetric matrices—even for self-adjoint differential operators—potentially doubling the computational cost of solving the resulting linear system when using direct methods. However, our focus is on general differential equations that are not necessarily self-adjoint.

1.1.3. Petrov–Galerkin Methods. Petrov–Galerkin methods solve linear ODEs using different bases for representing the solution (trial basis) and the right-hand side of the equation (test basis). The boundary conditions determine the trial basis [31], and the boundary conditions associated to the dual differential equation can determine the test basis [42]. Shen has constructed bases for second, third, fourth and higher odd order differential equations with constant coefficients so that the resulting matrices are banded or easily invertible [42, 43]. Moreover, it was shown that very specific variable coefficients preserve this matrix structure [44]. Related bases were constructed by Doha and Abd-Elhameed using ultraspherical polynomials [15, 16, 17]. These methods can be very efficient, but the method for imposing the boundary conditions is difficult to automate.

Our method will also use two different bases: the Chebyshev and ultraspherical polynomial bases. This choice preserves sparsity as do those considered in [17, 31, 43], but our bases will depend only on the order of the differential equation, and not the boundary conditions.

1.1.4. Integral Reformulation. Integral reformulation expresses the solution's Nth derivative in a Chebyshev expansion, and solves for those coefficients before integrating back to obtain the coefficients of the solution [11, 24, 50]. The same idea was previously advocated by Clenshaw [9] and a comparison was made to the tau-method by Fox [22], who concluded that Clenshaw's method was more accurate. In the case of constant coefficients, an alternative approach is to integrate the differential

equation itself, which results in a banded system [47]. Recently, integration reformulation has received considerable attention in the literature because it can construct well-conditioned linear systems [19, 29, 47].

Clenshaw's method was a motivating example for (F. W. J.) Olver's algorithm [33, 30]: a method for choosing the optimal truncation parameter n_{opt} for calculating solutions to recurrence relationships (i.e., banded linear systems). The original paper considered dense boundary rows to handle the boundary conditions for Clenshaw's method. In section 5.3 we develop the adaptive QR factorization that is a generalization of Olver's algorithm. The original approach was based on adaptive Gaussian elimination without pivoting, with a proof of numerical stability for sufficiently large n. By replacing LU with QR, we achieve stability for all n.

Many of the ideas that we develop in this paper are equally applicable to integral reformulation, as the resulting matrices are also almost banded. However, imposing boundary conditions requires additional variables related to the integration constants, considerably complicating the representation of multiplication. Moreover, integral reformulation does not maintain sparsity for *mixed operators* such as

$$\frac{\mathrm{d}}{\mathrm{d}x}a(x)\frac{\mathrm{d}u}{\mathrm{d}x},$$

as the integration will no longer annihilate the outer differentiation, which corresponds to a dense operator. Our approach avoids such issues.

1.1.5. Preconditioned Iterative Methods. Another approach to obtain well-conditioned linear systems is preconditioning, with the most common being motivated by a finite difference stencil [36], the operator representing the ODE with all its variable coefficients set to constants [3, 44], a finite element approximation [5, 14], or integration preconditioners [26, 20]. Once preconditioned, iterative solvers can be employed along with the fast Fourier transform, which can be more efficient than dense solvers in certain situations [4, 6, 44]. While specially designed iterative methods can outperform direct methods for specific problems, they do not provide the generality and robustness that we require.

Differential equations with highly oscillatory or nonanalytic variable coefficients are one example where iterative methods may be necessary to achieve computational efficiency, as the multiplication operators have very large bandwidth in coefficient space, resulting in essentially dense linear systems. Iterative methods based on matrix-vector products avoid this growth in complexity, and hence can be more efficient.

However, the matrix-vector product has to be user-supplied code that is unlikely to be optimized in terms of memory caching or memory allocation, and hence significant benefits over direct methods appear only for problems that require large n [49]. Moreover, direct methods provide better stability and robustness [24]. Finally, while the computational efficiency of our approach is lost for large bandwidth variable coefficients, numerical stability is maintained, as demonstrated in the second example of section 2.5.

2. Chebyshev Polynomials and First Order Differential Equations. For pedagogical reasons, we begin by solving first order differential equations of the form

$$u'(x) + a(x)u(x) = f(x) \qquad \text{and} \qquad u(-1) = c, \tag{2.1}$$

where $a : [-1, 1] \to \mathbb{C}$ and $f : [-1, 1] \to \mathbb{C}$ are continuous functions with bounded variation. The continuity assumption ensures that (2.1) has a unique continuously differentiable solution on the unit interval [39], while bounded variation ensures a unique

representation as a uniformly convergent Chebyshev expansion [32, Theorem 5.7]. An exact representation of a continuous function $g(x)$ with bounded variation is

$$g(x) = \sum_{k=0}^{\infty} g_k T_k(x),\ g_0 = \frac{2}{\pi}\int_{-1}^{1} \frac{g(x)}{\sqrt{1-x^2}}\mathrm{d}x,\ g_k = \frac{1}{\pi}\int_{-1}^{1} \frac{g(x)T_k(x)}{\sqrt{1-x^2}}\mathrm{d}x, \tag{2.2}$$

where $T_k(x) = \cos\left(k\cos^{-1}(x)\right)$. One way to approximate $g(x)$ is to truncate (2.2) after the first m terms, to obtain the polynomial (of degree at most $n-1$)

$$g_{\mathrm{trunc}}(x) = \sum_{k=0}^{m-1} g_k T_k(x). \tag{2.3}$$

The m coefficients $\{g_k\}$ can be obtained by numerical quadrature in $\mathcal{O}(m^2)$ operations. A second approach is to interpolate $g(x)$ at m Chebyshev points—that is, $\cos(k\pi/(m-1))$ for $k = 0, 1, \ldots, m-1$—to obtain the polynomial

$$g_{\mathrm{interp}}(x) = \sum_{k=0}^{m-1} \tilde{g}_k T_k(x). \tag{2.4}$$

In this case, the m coefficients $\{\tilde{g}_k\}$ can be computed with a fast cosine transform in just $\mathcal{O}(m\log m)$ operations. The coefficients $\{\tilde{g}_k\}$ and $\{g_k\}$ are closely related by an aliasing formula stated by Clenshaw and Curtis [10].

Usually, in practice, $g(x)$ will be many times differentiable on $[-1,1]$, and then (2.3) and (2.4) converge uniformly to $g(x)$ at an algebraic rate as $m \to \infty$. Moreover, the convergence is spectral (superalgebraic) when g is infinitely differentiable, which improves to being exponential when g is analytic in a neighborhood of $[-1,1]$. If g is entire, the convergence rate improves further to being superexponential.

Mathematically, we seek the Chebyshev coefficients of the truncation (2.3) of $a(x)$ and $f(x)$ that are defined by the integrals (2.2). However, we use the coefficients in the polynomial interpolant (2.4) as an approximation, since the coefficients match to machine precision for sufficiently large m. The degree of the polynomial used to approximate $a(x)$ will later be closely related to the bandwidth of the almost banded linear system associated to (2.1). In practice, the optimal truncation can be calculated adaptively by continually doubling the truncation size until convergence is observed.

In what follows, we will assume that we are given the truncated Chebyshev coefficients of the variable coefficients $a^N, \ldots, a^0$ and f. It is a significant feature of the numerical approximation that we calculate these expansions only once and independently of the number n of coefficients needed to represent the solution u. Because this computation is independent of n, we do not include it in the computational cost estimates. Including this cost results in a total of $O(n+m\log m)$ operations, where m is the number of Chebyshev coefficients needed to resolve all the variable coefficients. Thus, if u has a comparable truncation to f, the total computational cost will be dominated by expanding f and hence will be effectively $O(n\log n)$ operations. For singularly perturbed problems, the truncation length of u will dominate the truncation length of f, and hence the $O(n)$ term will dominate the computational cost.

The spectral method in this paper solves for the coefficients of the solution in a Chebyshev series. In order to achieve this, we need to be able to represent differentiation, $u'(x)$, and multiplication, $a(x)u(x)$, in terms of operators on coefficients.

2.1. First Order Differentiation Operator. The derivatives of Chebyshev polynomials satisfy

$$\frac{dT_k}{\mathrm{d}x} = \begin{cases} kC_{k-1}^{(1)}, & k \geq 1, \\ 0, & k = 0, \end{cases} \tag{2.5}$$

where $C_{k-1}^{(1)}(x)$ is the Chebyshev polynomial of the second kind of degree $k-1$. We use $C^{(1)}$ instead of U to highlight the connection to ultraspherical polynomials, which are key to extending the method to higher order differential equations (see section 3).

Now, we use (2.5) to derive a simple expression for the derivative of a Chebyshev series. Suppose that $u(x)$ is given by the Chebyshev series

$$u(x) = \sum_{k=0}^{\infty} u_k T_k(x). \tag{2.6}$$

Differentiating scales the coefficients and changes the basis:

$$u'(x) = \sum_{k=1}^{\infty} k u_k C_{k-1}^{(1)}(x).$$

In other words, the vector of coefficients of the derivative in a $C^{(1)}$ series is given by $\mathcal{D}_0\mathbf{u}$, where $\mathcal{D}_0$ is the differentiation operator

$$\mathcal{D}_0 = \begin{pmatrix} 0 & 1 & & & \\ & & 2 & & \\ & & & 3 & \\ & & & & \ddots \end{pmatrix},$$

and $\mathbf{u}$ is the (infinite) vector of Chebyshev coefficients for $u(x)$. Note that this differentiation operator is sparse, in stark contrast to the classic differentiation operator in spectral collocation methods [4].

2.2. Multiplication Operator for Chebyshev Series. In order to handle variable coefficients of the form $a(x)u(x)$ in (2.1), we need to represent the multiplication of two Chebyshev series as an operator on coefficients. To this end, we express $a(x)$ and $u(x)$ in terms of their Chebyshev series

$$a(x) = \sum_{j=0}^{\infty} a_j T_j(x) \qquad \text{and} \qquad u(x) = \sum_{k=0}^{\infty} u_k T_k(x)$$

and multiply these two expressions together to obtain

$$a(x)u(x) = \sum_{j=0}^{\infty}\sum_{k=0}^{\infty} a_j u_k T_j(x)T_k(x) = \sum_{k=0}^{\infty} c_k T_k(x),$$

desiring an explicit form for $\mathbf{c} = (c_0, c_1, \dots,)^\top$. By Proposition 2.1 of [1] we have that

$$c_k = \begin{cases} a_0u_0 + \frac{1}{2}\sum_{l=1}^{\infty} a_l u_l, & k = 0, \\ \frac{1}{2}\sum_{l=0}^{k-1} a_{k-l}u_l + a_0u_k + \frac{1}{2}\sum_{l=1}^{\infty} a_l u_{l+k} + \frac{1}{2}\sum_{l=0}^{\infty} a_{l+k}u_l, & k \geq 1, \end{cases} \tag{2.7}$$

and in terms of operators $\mathbf{c} = \mathcal{M}_0[a]\mathbf{u}$, where $\mathcal{M}_0[a]$ is a Toeplitz plus an almost Hankel operator given by

$$\mathcal{M}_0[a] = \frac{1}{2}\left[\begin{pmatrix} 2a_0 & a_1 & a_2 & a_3 & \dots \\ a_1 & 2a_0 & a_1 & a_2 & \ddots \\ a_2 & a_1 & 2a_0 & a_1 & \ddots \\ a_3 & a_2 & a_1 & 2a_0 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & \dots \\ a_1 & a_2 & a_3 & a_4 & \iddots \\ a_2 & a_3 & a_4 & a_5 & \iddots \\ a_3 & a_4 & a_5 & a_6 & \iddots \\ \vdots & \iddots & \iddots & \iddots & \iddots \end{pmatrix}\right].$$

At first glance, it appears that the multiplication operator and any truncation of it are dense. However, since $a(x)$ is continuous with bounded variation, we are able to uniformly approximate $a(x)$ with a finite number of Chebyshev coefficients to any desired accuracy. That is, for any $\epsilon > 0$ there exists an $m \in \mathbb{N}$ such that

$$\left\| a(x) - \sum_{k=0}^{m-1} a_k T_k(x) \right\|_{L_\infty([-1,1])} < \epsilon.$$

As long as m is large enough, to all practical purposes, we can use the truncated Chebyshev series to replace $a(x)$. Recall that in our implementation we approximate this truncation by the polynomial interpolant of the form (2.4). Hence, the $n \times n$ principal part of $\mathcal{M}_0[a]$ is banded with bandwidth m for $n > m$. Moreover, m can be surprisingly small when $a(x)$ is analytic or many times differentiable.

There is still one ingredient absent: The operator $\mathcal{D}_0$ returns coefficients in a $C^{(1)}$ series, whereas the operator $\mathcal{M}_0[a]$ returns coefficients in a Chebyshev series; hence, $\mathcal{D}_0 + \mathcal{M}_0[a]$ is meaningless. In order to correct this, we require an operator that maps coefficients in a Chebyshev series to those in a $C^{(1)}$ series.

2.3. Conversion Operator for Chebyshev Series. The Chebyshev polynomials $T_k(x)$ can be written in terms of the $C^{(1)}$ polynomials by the recurrence relation

$$T_k = \begin{cases} \frac{1}{2}\left(C_k^{(1)} - C_{k-2}^{(1)}\right), & k \geq 2, \\ \frac{1}{2}C_1^{(1)}, & k = 1, \\ C_0^{(1)}, & k = 0. \end{cases} \tag{2.8}$$

A more general form of (2.8), which we will use later, is given in [34]. Suppose that $u(x)$ is given by the Chebyshev series (2.6). Then, using (2.8), we have

$$\begin{aligned} u(x) = \sum_{k=0}^{\infty} u_k T_k(x) &= u_0 C_0^{(1)}(x) + \frac{1}{2}u_1 C_1^{(1)}(x) + \frac{1}{2}\sum_{k=2}^{\infty} u_k \left(C_k^{(1)}(x) - C_{k-2}^{(1)}(x)\right) \\ &= \left(u_0 - \frac{1}{2}u_2\right) C_0^{(1)}(x) + \sum_{k=1}^{\infty} \frac{1}{2}\left(u_k - u_{k+2}\right) C_k^{(1)}(x). \end{aligned}$$

Hence, the $C^{(1)}$ coefficients for $u(x)$ are $\mathcal{S}_0\mathbf{u}$, where $\mathcal{S}_0$ is the conversion operator

$$\mathcal{S}_0 = \begin{pmatrix} 1 & & -\frac{1}{2} & & \\ & \frac{1}{2} & & -\frac{1}{2} & \\ & & \frac{1}{2} & & -\frac{1}{2} \\ & & & \ddots & & \ddots \end{pmatrix}$$

and $\mathbf{u}$ is the vector of Chebyshev coefficient for $u(x)$. Note that this conversion operator, and any truncation of it, is sparse and banded.

2.4. Discretization of the System. Now, we have all the ingredients to solve any differential equation of the form (2.1). First, we can represent the differential operator as

$$\mathcal{L} := \mathcal{D}_0 + \mathcal{S}_0\mathcal{M}_0[a],$$

which takes coefficients in a Chebyshev series to those in a $C^{(1)}$ series. Due to this fact, the right-hand side $f(x)$ must be expressed in terms of its coefficients in a $C^{(1)}$ series. Hence, we can represent the differential equation (2.1), without its boundary conditions, as

$$\mathcal{L}\mathbf{u} = \mathcal{S}_0\mathbf{f},$$

where $\mathbf{u}$ and $\mathbf{f}$ denote the vectors of coefficients in the Chebyshev series of the form (2.2) for $u(x)$ and $f(x)$, respectively.

We truncate the operators to derive a practical numerical scheme, which corresponds to applying the $n \times \infty$ projection operator given by

$$\mathcal{P}_n = (I_n, \mathbf{0}), \tag{2.9}$$

where I_n is the square identity matrix of size n. Truncating the differentiation operator to $\mathcal{P}_n\mathcal{D}_0\mathcal{P}_n^\top$ results in an $n\times n$ matrix with a zero last row. This observation motivates us to impose the boundary condition by replacing the last row of $\mathcal{P}_n\mathcal{L}\mathcal{P}_n^\top$. We take the convention of permuting this boundary row to the first row so that the linear system is close to upper triangular. That is, in order to obtain an approximate solution to (2.1) we solve the system

$$A_n \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \end{pmatrix} = \begin{pmatrix} c \\ \\ \left(\mathcal{P}_{n-1}\mathcal{S}_0\mathcal{P}_n^\top\right)\left(\mathcal{P}_n\mathbf{f}\right) \\ \\ \end{pmatrix} \tag{2.10}$$

with

$$A_n = \begin{pmatrix} T_0(-1) & T_1(-1) & \dots & T_{n-1}(-1) \\ & \mathcal{P}_{n-1}\mathcal{L}\mathcal{P}_n^\top & & \end{pmatrix}.$$

(Note that $T_k(-1) = (-1)^k$.) The solution $u(x)$ is then approximated by the computed solution,

$$\tilde{u}(x) = \sum_{k=0}^{n-1} u_k T_k(x).$$

Remark 2. Traditionally, one generates the spectral system by discretizing each operator individually; i.e., we would make the approximation

$$\mathcal{P}_{n-1}\mathcal{L}\mathcal{P}_n^\top \approx \mathcal{P}_{n-1}\mathcal{D}_0\mathcal{P}_n^\top + \left(\mathcal{P}_{n-1}\mathcal{S}_0\mathcal{P}_n^\top\right)\left(\mathcal{P}_n\mathcal{M}[a]\mathcal{P}_n^\top\right).$$

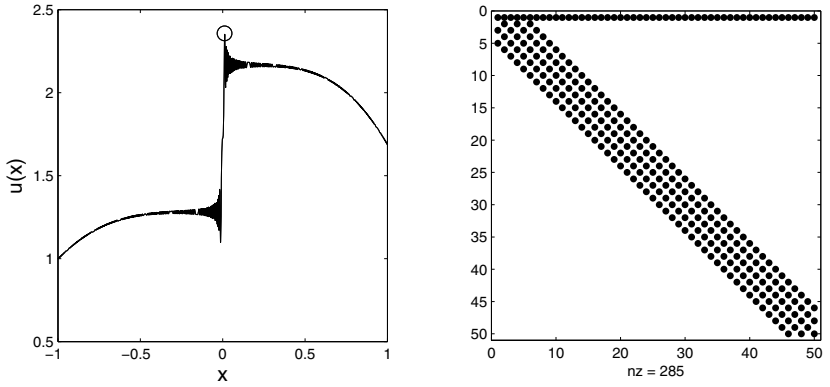

Fig. 2.1 *Left: The highly oscillatory solution to* (2.11) *with the maximum of the solution computed and marked by a circle. Right: Sparsity structure of the almost banded linear system, with the number of nonzero entries denoted by* `nz`.

However, we can generate $\mathcal{P}_{n-1}\mathcal{L}\mathcal{P}_n^\top$ precisely via

$$\mathcal{P}_{n-1}\mathcal{L}\mathcal{P}_n^\top = \mathcal{P}_{n-1}\mathcal{D}_n\mathcal{P}_n^\top + \left(\mathcal{P}_{n-1}\mathcal{S}_0\mathcal{P}_{n+m}^\top\right)\left(\mathcal{P}_{n+m}\mathcal{M}[a]\mathcal{P}_n^\top\right).$$

This means that each row of the operator can be generated exactly and allows us to develop the adaptive QR algorithm in section 5. Moreover, exact truncations are of critical importance for infinite-dimensional linear algebra and for approximating spectra and pseudospectra [25].

2.5. Numerical Examples. The traditional approach solves the linear system (2.10) for progressively larger n, and terminates the process when the tail of the solution falls below relative magnitude of machine epsilon. This is the procedure employed in Chebfun [46, 18]. Instead the adaptive QR algorithm of section 5 can be used to find the optimal truncation and solve for the solution simultaneously.

For the first example, we consider the linear differential equation

$$u'(x) + x^3u(x) = 100\sin\left(20{,}000x^2\right), \qquad u(-1) = 0, \tag{2.11}$$

which has a highly oscillatory forcing term. The exact solution is

$$u(x) = e^{-\frac{x^4}{4}}\left(\int_{-1}^{x} 100e^{\frac{t^4}{4}}\sin\left(20{,}000t^2\right)dt\right).$$

The computed solution is a polynomial of degree 20,391 and uniformly approximates the exact solution to essentially machine precision. The computed solution is of very high degree because at least 2 coefficients are required per oscillation—the *Nyquist rate*. In Figure 2.1 we plot the computed oscillatory solution and a realization of the matrix when $n = 50$ to show the almost banded structure of the linear system.

The approximate solution is expressed in terms of a Chebyshev basis, which is convenient for further manipulation. For example, its maximum is 2.3573 (circled in Figure 2.1 (left)), its integral is 3.2879, and the equation $u(x) - 1.3 = 0$ has 113 solutions in $x \in [-1, 1]$.

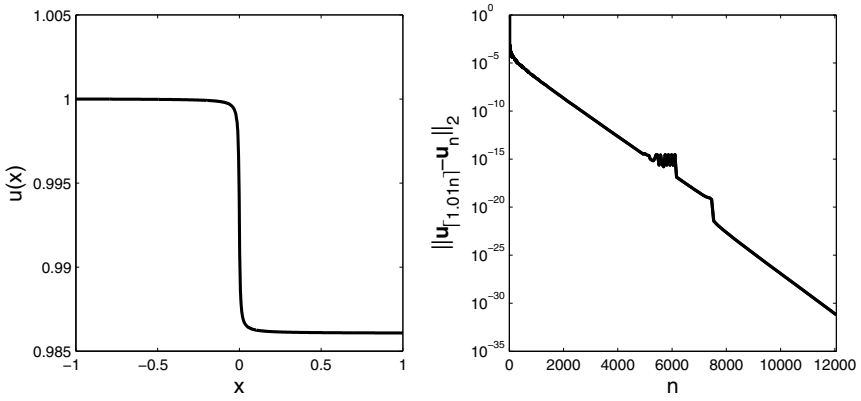

Fig. 2.2 *Left: The computed solution to* (2.12) *with* $a = 5 \times 10^4$. *Right: Plot of the Cauchy error for the solution coefficients, which shows the* 2*-norm difference between the coefficients of the approximate solution when solving an* $n \times n$ *and an* $\lceil 1.01n \rceil \times \lceil 1.01n \rceil$ *matrix system.*

As a second example we consider the linear differential equation

$$u'(x) + \frac{1}{ax^2 + 1} u(x) = 0, \qquad u(-1) = 1. \tag{2.12}$$

We take $a = 5 \times 10^4$, and the variable coefficient can be approximated to roughly machine precision by a polynomial of degree 7,350. Therefore, in this example, a large n is required in the linear system (2.10) before it is banded. The exact solution to (2.12) is

$$u(x) = \exp\left(-\frac{\tan^{-1}(\sqrt{a}x) + \tan^{-1}(\sqrt{a})}{\sqrt{a}} \right),$$

which can be approximated to machine precision by a polynomial of degree 5,377, determined by interpolating at Chebyshev points. On the other hand, the computed solution $\tilde{u}(x)$ is a polynomial of degree 5,093 such that

$$\left(\int_{-1}^{1} (u(x) - \tilde{u}(x))^2 \, \mathrm{d}x \right)^{\frac{1}{2}} = 2.86 \times 10^{-15}.$$

The solution contains a thin boundary layer and a singularity in the complex plane that is close in proximity to the $[-1, 1]$. This means that the associated Berstein ellipse is restricted, and therefore a large degree polynomial is required to approximate the solution (see, for example, [12]). A plot of the solution and the Cauchy error are included in Figure 2.2. The Cauchy error plot confirms that the solution is, up to machine precision, independent of the number of coefficients in its expansion for $n \geq 5{,}100$.

3. Ultraspherical Polynomials and High Order Differential Equations. We now generalize the approach to high order differential equations of the form

$$\sum_{\lambda=0}^{N} a^{\lambda}(x) \frac{\mathrm{d}^{\lambda} u(x)}{\mathrm{d}x^{\lambda}} = f(x) \qquad \text{on } [-1, 1], \tag{3.1}$$

with general boundary conditions $\mathcal{B}u = \mathbf{c}$. We assume that the boundary operator $\mathcal{B}$ is given in terms of the Chebyshev coefficients of u. For example, for Dirichlet conditions

$$\mathcal{B} = \begin{pmatrix} T_0(-1) & T_1(-1) & T_2(-1) & \cdots \\ T_0(1) & T_1(1) & T_2(1) & \cdots \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 & -1 & \cdots \\ 1 & 1 & 1 & 1 & \cdots \end{pmatrix},$$

and for Neumann conditions

$$\mathcal{B} = \begin{pmatrix} T_0'(-1) & T_1'(-1) & T_2'(-1) & \cdots \\ T_0'(1) & T_1'(1) & T_2'(1) & \cdots \end{pmatrix} = \begin{pmatrix} 0 & 1 & -4 & \cdots & (-1)^{k+1}k^2 & \cdots \\ 0 & 1 & 4 & \cdots & k^2 & \cdots \end{pmatrix}.$$

We can also impose less standard boundary conditions; e.g., we can impose that the solution integrates to a constant by using the Clenshaw–Curtis weights [10] (computable in $\mathcal{O}(n \log n)$ operations [48]) or it evaluates to a fixed constant at a point in $(-1, 1)$. In fact, because we are using boundary bordering, any boundary condition that depends linearly on the solution's coefficients can be imposed in an automated manner.

The approach of the first order method relied on three relations: differentiation (2.5), multiplication (2.7), and conversion (2.8). To generalize the spectral method to higher order differential equations we use similar relations, now in terms of higher order ultraspherical polynomials.

The ultraspherical (or Gegenbauer) polynomials $C_0^{(\lambda)}(x), C_1^{(\lambda)}(x), \ldots$ are a family of polynomials orthogonal with respect to the weight

$$(1 - x^2)^{\lambda - \frac{1}{2}}.$$

We will use only ultraspherical polynomials for $\lambda = 1, 2, \ldots$, defined uniquely by normalizing the leading coefficient so that

$$C_k^{(\lambda)}(x) = \frac{2^k(\lambda)_k}{k!} x^k + \mathcal{O}(x^{k-1}),$$

where $(\lambda)_k = \frac{(\lambda+k-1)!}{(\lambda-1)!}$ denotes the *Pochhammer symbol*. In particular, the ultraspherical polynomials with $\lambda = 1$ are the Chebyshev polynomials of the second kind that we denote by $C^{(1)}$.

Importantly, ultraspherical polynomials satisfy an analogue of (2.5) [34],

$$\frac{dC_k^{(\lambda)}}{\mathrm{d}x} = \begin{cases} 2\lambda C_{k-1}^{(\lambda+1)}, & k \geq 1, \\ 0, & k = 0, \end{cases} \qquad \lambda \geq 1. \tag{3.2}$$

Moreover, they also satisfy an analogue to (2.8),

$$C_k^{(\lambda)} = \begin{cases} \frac{\lambda}{\lambda+k}\left(C_k^{(\lambda+1)} - C_{k-2}^{(\lambda+1)}\right), & k \geq 2, \\ \frac{\lambda}{\lambda+1} C_1^{(\lambda+1)}, & k = 1, \\ C_0^{(\lambda+1)}, & k = 0, \end{cases} \qquad \lambda \geq 1. \tag{3.3}$$

Suppose that $u(x)$ is represented as the Chebyshev series (2.6). Then, for $\lambda = 1, 2, \ldots$, (2.5) implies that

$$\frac{\mathrm{d}^\lambda u(x)}{\mathrm{d}x^\lambda} = \sum_{k=1}^{\infty} k u_k \frac{\mathrm{d}^{\lambda-1} C_{k-1}^{(1)}(x)}{\mathrm{d}x^{\lambda-1}}.$$

By applying the relation (3.2) $\lambda - 1$ times we obtain

$$\frac{\mathrm{d}^\lambda u(x)}{\mathrm{d}x^\lambda} = 2^{\lambda-1}(\lambda-1)! \sum_{k=\lambda}^{\infty} k u_k C_{k-\lambda}^{(\lambda)}(x).$$

This means that the λ-order differentiation operator takes the form

$$\mathcal{D}_\lambda = 2^{\lambda-1}(\lambda-1)! \begin{pmatrix} \overbrace{0 \quad \cdots \quad 0}^{\lambda \text{ times}} & \lambda & & & \\ & & \lambda+1 & & \\ & & & \lambda+2 & \\ & & & & \ddots \end{pmatrix}.$$

The λ-order differentiation operator that does not change bases can be constructed from recurrence relations [16], and this can be used to construct spectral methods [15]. However, differentiation operators constructed in such a way are not sparse.

In the process of differentiation, $\mathcal{D}_\lambda$ converts coefficients in a Chebyshev series to coefficients in a $C^{(\lambda)}$ series. Moreover, using (3.3) the operator that converts coefficients in $C^{(\lambda)}$ series to those in a $C^{(\lambda+1)}$ series is given by

$$\mathcal{S}_\lambda = \begin{pmatrix} 1 & -\frac{\lambda}{\lambda+2} & & & \\ \frac{\lambda}{\lambda+1} & & -\frac{\lambda}{\lambda+3} & & \\ & \frac{\lambda}{\lambda+2} & & -\frac{\lambda}{\lambda+4} & \\ & & \ddots & & \ddots \end{pmatrix}.$$

As before, we also require a multiplication operator, but this time representing the multiplication between two $C^{(\lambda)}$ ultraspherical series.

3.1. Multiplication Operator for Ultraspherical Series. In order to handle the variable coefficients in (3.1), we must represent multiplication of two ultraspherical series in coefficient space. Given two functions

$$a(x) = \sum_{j=0}^{\infty} a_j C_j^{(\lambda)}(x) \qquad \text{and} \qquad u(x) = \sum_{k=0}^{\infty} u_k C_k^{(\lambda)}(x),$$

we have

$$a(x)u(x) = \sum_{j=0}^{\infty}\sum_{k=0}^{\infty} a_j u_k C_j^{(\lambda)}(x) C_k^{(\lambda)}(x). \tag{3.4}$$

To obtain a $C^{(\lambda)}$ series for $a(x)u(x)$ we use the linearization formula given by Carlitz [7] that takes the form

$$C_j^{(\lambda)}(x) C_k^{(\lambda)}(x) = \sum_{s=0}^{\min(j,k)} c_s^\lambda(j,k) C_{j+k-2s}^{(\lambda)}(x), \tag{3.5}$$

where

$$c_s^\lambda(j,k) = \frac{j+k+\lambda-2s}{j+k+\lambda-s} \frac{(\lambda)_s (\lambda)_{j-s} (\lambda)_{k-s}}{s!(j-s)!(k-s)!} \frac{(2\lambda)_{j+k-s}}{(\lambda)_{j+k-s}} \frac{(j+k-2s)!}{(2\lambda)_{j+k-2s}}. \tag{3.6}$$

We substitute (3.5) into (3.4) and rearrange the summation signs to obtain

$$a(x)u(x) = \sum_{j=0}^{\infty} \left(\sum_{k=0}^{\infty} \sum_{s=\max(0,k-j)}^{k} a_{2s+j-k} c_s^{\lambda}(k, 2s+j-k) u_k \right) C_j^{(\lambda)}(x). \tag{3.7}$$

From (3.7) the (j,k) entry of the multiplication operator representing the product of $a(x)$ in a $C^{(\lambda)}$ series is

$$\mathcal{M}_{\lambda}[a]_{j,k} = \sum_{s=\max(0,k-j)}^{k} a_{2s+j-k} c_s^{\lambda}(k, 2s+j-k), \qquad j,k \geq 0.$$

In practice, $a(x)$ will be approximated by a truncation of its $C^{(\lambda)}$ series,

$$a(x) \approx \sum_{j=0}^{m-1} a_j C_j^{(\lambda)}(x), \tag{3.8}$$

and with this approximation the matrix $\mathcal{P}_n \mathcal{M}_{\lambda}[a] \mathcal{P}_n^{\top}$ is banded with bandwidth m for $n > m$. The expansion (3.8) can be computed by approximating the first m Chebyshev coefficients in the Chebyshev series for $a(x)$ and then applying a truncation of the conversion operator $\mathcal{S}_{\lambda-1} \cdots S_0$.

The formula for $c_s^{\lambda}(j,k)$, (3.6), cannot be used directly to form $\mathcal{M}_{\lambda}[a]$ due to arithmetic overflow issues that arise for $j,k \geq 70$. Instead, we cancel terms in the numerator and denominator of (3.6) and match up the remaining terms of similar magnitude to obtain an equivalent, but more numerically stable, formula:

$$\begin{aligned} c_s^{\lambda}(j,k) = {} & \frac{j+k+\lambda-2s}{j+k+\lambda-s} \times \prod_{t=0}^{s-1} \frac{\lambda+t}{1+t} \times \prod_{t=0}^{j-s-1} \frac{\lambda+t}{1+t} \\ & \times \prod_{t=0}^{s-1} \frac{2\lambda+j+k-2s+t}{\lambda+j+k-2s+t} \times \prod_{t=0}^{j-s-1} \frac{k-s+1+t}{k-s+\lambda+t}. \end{aligned} \tag{3.9}$$

All the fractions are of magnitude $\mathcal{O}(1)$ in size, and hence $\mathcal{P}_n \mathcal{M}_{\lambda}[a] \mathcal{P}_n^{\top}$ can be formed in floating point arithmetic. For the purposes of computational speed, we apply (3.9) only once per entry, and use the recurrence relation

$$\begin{aligned} c_{s+1}^{\lambda}(j,k+2) = c_s^{\lambda}(j,k) & \times \frac{j+k+\lambda-s}{j+k+\lambda-s+1} \times \frac{\lambda+s}{s+1} \\ & \times \frac{j-s}{\lambda+j-s-1} \times \frac{2\lambda+j+k-s}{\lambda+j+k-s} \times \frac{k-s+\lambda}{k-s+1} \end{aligned}$$

to generate all the other terms required.

Remark 3. In the special case when $\lambda = 1$, the multiplication operator $\mathcal{M}_1[a]$ can be decomposed as a Toeplitz operator plus a Hankel operator.

3.2. Discretization of the System. We now have everything in place to be able to solve high order differential equations of the form (3.1). First, we can represent the differential operator as

$$\mathcal{L} := \mathcal{M}_N[a^N] \mathcal{D}_N + \sum_{\lambda=1}^{N-1} \mathcal{S}_{N-1} \cdots \mathcal{S}_{\lambda} \mathcal{M}_{\lambda}[a^{\lambda}] \mathcal{D}_{\lambda} + \mathcal{S}_{N-1} \cdots \mathcal{S}_0 \mathcal{M}_0[a^0],$$

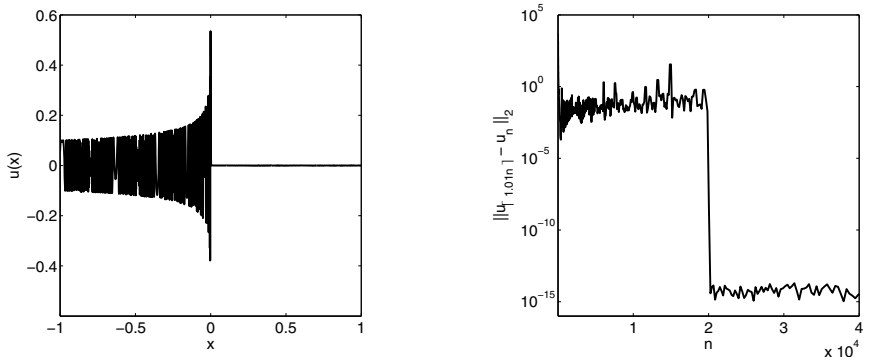

Fig. 3.1 *Left: The highly oscillatory solution to* (3.10) *with* $\epsilon = 10^{-9}$. *Right: The Cauchy error for the solution coefficients. The plot shows the* 2*-norm difference between the coefficients of the approximate solution when solving an* $n \times n$ *and* $\lceil 1.01n \rceil \times \lceil 1.01n \rceil$ *matrix system.*

which takes coefficients in a Chebyshev series to those in a $C^{(N)}$ series. Due to this fact, the right-hand side $f(x)$ must be expressed in terms of its coefficients in a $C^{(N)}$ series. Moreover, we impose the K boundary conditions on the solution by replacing the last K rows of $\mathcal{P}_n \mathcal{L} \mathcal{P}_n^\top$ and permute these to the first K rows. That is, in order to obtain an approximate solution to (3.1) we solve the system

$$A_n \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \end{pmatrix} = \begin{pmatrix} \mathbf{c} \\ \\ \mathcal{P}_{n-K} \mathcal{S}_{N-1} \cdots \mathcal{S}_0 \mathbf{f} \\ \\ \end{pmatrix},$$

where

$$A_n = \begin{pmatrix} \mathcal{B} \mathcal{P}_n^\top \\ \\ \mathcal{P}_{n-K} \mathcal{L} \mathcal{P}_n^\top \\ \\ \end{pmatrix}$$

and $\mathbf{f}$ is again a vector containing the Chebyshev coefficients of the right-hand side f. The solution $u(x)$ is then approximated by the n-term Chebyshev series:

$$u(x) \approx \sum_{k=0}^{n-1} u_k T_k(x).$$

3.3. Numerical Examples. For the first example we consider the Airy differential equation

$$\epsilon u''(x) - x u(x) = 0 \qquad \text{with} \qquad u(-1) = \operatorname{Ai}\left(-\sqrt[3]{\frac{1}{\epsilon}} \right), \; u(1) = \operatorname{Ai}\left(\sqrt[3]{\frac{1}{\epsilon}} \right), \tag{3.10}$$

where $\operatorname{Ai}(\cdot)$ is the Airy function of the first kind.

In Figure 3.1 we take $\epsilon = 10^{-9}$ and plot the computed solution that is a polynomial of degree 20,003. The exact solution to (3.10) is the scaled Airy function,

$$u(x) = \operatorname{Ai}\left(\sqrt[3]{\frac{1}{\epsilon}} x \right).$$

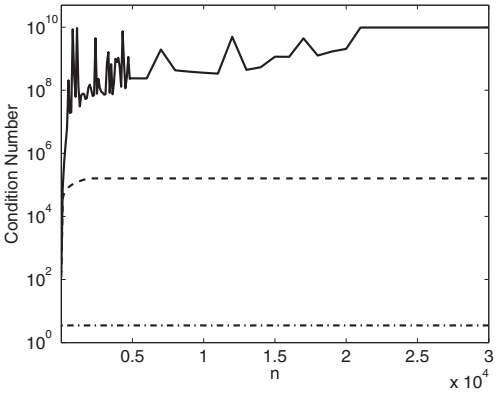

Fig. 3.2 *Plot of the condition number of the linear systems to solve* (3.10) *against the size of the discretization for* $\epsilon = 1 \times 10^{-9}$ *(solid),* $\epsilon = 1 \times 10^{-4}$ *(dashed), and* $\epsilon = 1$ *(dot-dashed). For* $\epsilon = 1$ *we employ the diagonal preconditioner presented in section* 4*. The plot demonstrates that the* 2*-norm condition number is bounded from above. The observed error in the solution is considerably better than what is suggested by the bounding constants, as seen in Figure* 3.1*.*

Letting $\tilde{u}(x)$ denote the computed solution, we have

$$\left(\int_{-1}^{1} (u(x) - \tilde{u}(x))^2 \right)^{1/2} = 2.44 \times 10^{-12},$$

which is surprisingly good when compared with the ill-conditioning inherent in this singularly perturbed differential equation. Numerically we witness that the spectral method delivers much better accuracy than standard bounds based on the condition number would suggest. The Cauchy error plot in Figure 3.1 indicates that the solution coefficients themselves are resolved to essentially machine precision for $n \geq 20{,}000$. In Figure 3.2 we show numerical evidence that with a simple diagonal preconditioner, which we analyze in the next section, the condition number of the linear systems formed are bounded for all n. Later, for $\epsilon = 1$, we also show in Figure 5.3 that the derivatives of the solution are well approximated.

For the second example we consider the boundary layer problem,

$$\epsilon u''(x) - 2x \left(\cos(x) - \frac{8}{10} \right) u'(x) + \left(\cos(x) - \frac{8}{10} \right) u(x) = 0, \tag{3.11}$$

with boundary conditions

$$u(-1) = u(1) = 1.$$

Perturbation theory shows that the solution has two boundary layers at $\pm \cos^{-1}(8/10)$, both of width $\mathcal{O}(\epsilon^{1/4})$. In Figure 3.3 we take $\epsilon = 10^{-7}$. The computed solution is of degree 15,394 and it is confirmed by the Cauchy error plot that the solution is well resolved.

For the last example we consider the high order differential equation

$$u^{(10)}(x) + \cosh(x) u^{(8)}(x) + x^2 u^{(6)}(x) + x^4 u^{(4)}(x) + \cos(x) u^{(2)}(x) + x^2 u(x) = 0,$$

with boundary conditions

$$u(-1) = u(1) = 0,\ u'(-1) = u'(1) = 1,\ u^{(k)}(\pm 1) = 0, \qquad 2 \leq k \leq 4.$$

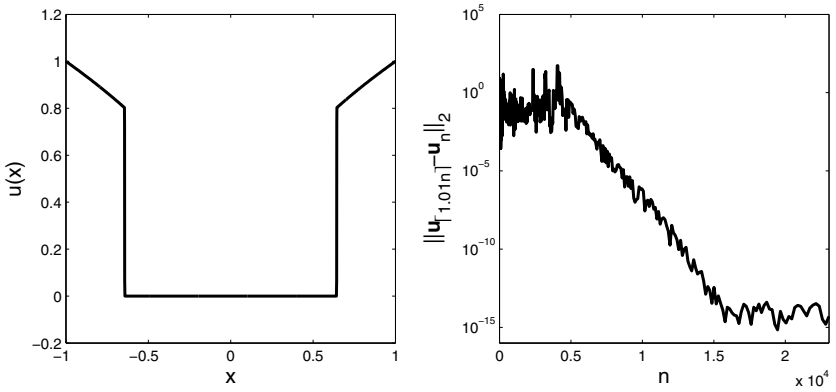

Fig. 3.3 *Left: The solution to the boundary layer problem* (3.11) *with* $\epsilon = 10^{-7}$. *Right: The Cauchy error in the* 2*-norm for the solution coefficients.*

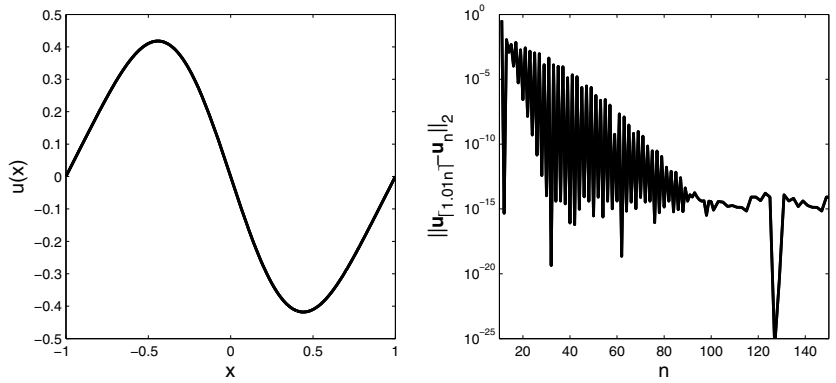

Fig. 3.4 *Left: Plot of the solution to the* 10*th order differential equation. Right: Plot of the Cauchy error in the* 2*-norm for the solution coefficients.*

This is far from a practical example, and an exact solution seems difficult to construct. Instead, we note that if $u(x)$ is the solution, then it is odd; that is, $u(x) = -u(-x)$. Our method does not impose such a condition, and therefore we can use it along with the Cauchy error to gain confidence in the computed solution. The computed solution $\tilde{u}(x)$ is of degree 55 and plotted in Figure 3.4. Moreover, the computed solution is odd to about machine precision,

$$\left(\int_{-1}^{1} \left(\tilde{u}(x) + \tilde{u}(-x) \right)^2 \right)^{1/2} = 1.252 \times 10^{-14}.$$

4. Stability and Convergence. The 2-norm condition number of a matrix $A \in \mathbb{C}^{n \times n}$ is defined as

$$\kappa(A) = \|A\|_2 \left\|A^{-1}\right\|_2.$$

Without any preconditioning $\kappa(A_n)$ grows proportionally with n, which is significantly better than the typical growth of $\mathcal{O}(n^{2N})$ in the condition number for the

standard tau and collocation methods (see section 4.3 of [6]). However, the accuracy seen in practice even outperforms this: the backward error is consistent with a numerical method with bounded condition number. Later, we will show that a trivial, diagonal preconditioner that scales the columns results in a linear system with a bounded condition number. However, we observe that even without preconditioning the linear systems can be solved to the same accuracy. We explain this with the following proposition that the stability of QR is not affected by column scaling.

PROPOSITION 4.1. *Suppose T is a diagonal matrix. Solving $A\mathbf{c} = \mathbf{b}$ using QR (with Givens rotations) is stable if QR applied to solve $AT\mathbf{q} = \mathbf{b}$ is stable and $\|T\|_\infty \leq 1$.*

Proof. This follows immediately from the invariance of Givens rotations to column scaling and the stability of back substitution; see [27, pp. 374]. ∎

4.1. A Diagonal Preconditioner and Compactness. Throughout this section we assume that $a^N(x) = 1$ (otherwise, divide through by the coefficient on the highest order term, assuming it is nonsingular). We make the restriction that $K = N$; that is, the Nth order differential equation has exactly N boundary conditions. When $K > N$ it is more appropriate to choose a nondiagonal preconditioner, but we do not analyze that situation here.

We show that there exists a diagonal preconditioner so that the preconditioned system has bounded condition number in high order norms (Definition 4.2). For Dirichlet boundary conditions the preconditioned system has bounded condition number in the 2-norm.

Define the diagonal preconditioner by

$$\mathcal{R} = \frac{1}{2^{N-1}(N-1)!}\operatorname{diag}\left(\overbrace{1,\ldots,1}^{N \text{ times}}, \frac{1}{N}, \frac{1}{N+1}, \ldots\right).$$

In practice, we observe that many other diagonal preconditioners also give a bounded condition number, and it is likely that there are preconditioners that give better practical bounds on the backward error, but only by a constant factor.

The analysis of (1.1) will follow from the fact that, on suitably defined spaces,

$$\begin{pmatrix}\mathcal{B}\\ \mathcal{L}\end{pmatrix}\mathcal{R} = I + \mathcal{K}$$

for a compact operator $\mathcal{K}$, where $\mathcal{L}$ is the Nth order differential operator and $\mathcal{B}$ is a boundary operator representing N boundary conditions. To achieve this we need to be precise about the spaces these operators act on. Since we are working in coefficient space, we consider the problems as defined in ℓ^2_λ spaces, which correspond to Sobolev spaces of the transformed function $u(\cos\theta)$.

DEFINITION 4.2. *The space $\ell^2_\lambda \subset \mathbb{C}^\infty$ is defined as the Banach space with norm*

$$\|\mathbf{u}\|_{\ell^2_\lambda} = \sqrt{\sum_{k=0}^{\infty} |u_k|^2 (k+1)^{2\lambda}} < \infty.$$

We now show that the preconditioned operator is a compact perturbation of the identity. The spaces ℓ^2_λ such that this will hold true will depend on the smallest D such that $\mathcal{B} : \ell^2_D \to \mathbb{C}^K$ is bounded. As examples, Dirichlet conditions have bounded nondecaying entries, hence $\mathcal{B} : \ell^2_1 \to \mathbb{C}^K$ is bounded. On the other hand, first order

Neumann conditions have linearly growing entries, and thus $\mathcal{B} : \ell^2_2 \to \mathbb{C}^K$ is bounded. Each additional derivative used in the boundary condition will increase D by one.

LEMMA 4.3. *Suppose that the boundary operator $\mathcal{B} : \ell^2_D \to \mathbb{C}^K$ is bounded.*[1] *Then*

$$\begin{pmatrix} \mathcal{B} \\ \mathcal{L} \end{pmatrix} \mathcal{R} = I + \mathcal{K},$$

where $\mathcal{K} : \ell^2_\lambda \to \ell^2_\lambda$ is compact for $\lambda = D-1, D, \ldots$.

Proof. First, note that

$$\begin{pmatrix} \mathcal{B} \\ \mathcal{L} \end{pmatrix} = \begin{pmatrix} 2^{N-1}(N-1)!\mathcal{P}_N \\ \mathcal{D}_N \end{pmatrix} + \begin{pmatrix} \mathcal{B} - 2^{N-1}(N-1)!\mathcal{P}_N \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \mathcal{S}_{N-1}\mathcal{M}_{N-1}[a^{N-1}]\mathcal{D}_{N-1} + \cdots + \mathcal{S}_{N-1}\cdots\mathcal{S}_1\mathcal{M}_1[a^1]\mathcal{D} + \mathcal{S}_{N-1}\cdots\mathcal{S}_0\mathcal{M}[a^0] \end{pmatrix},$$

where $\mathcal{P}_N : \ell^2_\lambda \to \mathbb{C}^N$ is the $N \times \infty$ projection operator (2.9).

Second, we remark that $\mathcal{R} : \ell^2_\lambda \to \ell^2_{\lambda+1}$, and hence $\mathcal{B}\mathcal{R} : \ell^2_\lambda \to \mathbb{C}^N$ is bounded for $\lambda = D-1, D, \ldots$. Furthermore, $\mathcal{S}_k : \ell^2_\lambda \to \ell^2_{\lambda+1}$ is bounded for $k = 1, 2, \ldots$, and so is $\mathcal{D}_N : \ell^2_\lambda \to \ell^2_{\lambda-1}$. It follows that

$$\begin{pmatrix} 2^{N-1}(N-1)!\mathcal{P}_N \\ \mathcal{D}_N \end{pmatrix} \mathcal{R} = I : \ell^2_\lambda \to \ell^2_\lambda.$$

Since $\begin{pmatrix} \mathcal{B} - 2^{N-1}(N-1)!\mathcal{P}_N \\ 0 \end{pmatrix} \mathcal{R} : \ell^2_\lambda \to \ell^2_\lambda$ for $\lambda = D-1, D, \ldots$ is bounded and has finite rank, it is compact. Note that $\mathcal{R}$ is compact as an operator $\mathcal{R} : \ell^2_\lambda \to \ell^2_\lambda$, as are $\mathcal{S}_{N-1}, \ldots, \mathcal{S}_1$ (since $\mathcal{S}_k\mathcal{R}^{-1} : \ell^2_\lambda \to \ell^2_\lambda$ are bounded and $\mathcal{R}$ is compact). It follows that the last term

$$\begin{pmatrix} 0 \\ \mathcal{S}_{N-1}\cdots\mathcal{S}_0\mathcal{M}[a^0] \end{pmatrix} \mathcal{R} : \ell^2_\lambda \to \ell^2_\lambda$$

is compact, since $\mathcal{M}[a^0]$ and $\mathcal{S}_0$ are also bounded. Finally, each of the intermediate terms are compact since $\begin{pmatrix} 0 \\ \mathcal{D}_k \end{pmatrix} \mathcal{R} : \ell^2_\lambda \to \ell^2_\lambda$ is bounded and $\mathcal{S}_k$ are compact.

We have shown that the preconditioned operator is the identity plus a sum of compact operators and hence a compact perturbation of the identity. ∎

The compactness of $\mathcal{K}$ allows us to show well-conditioning and convergence.

LEMMA 4.4. *Suppose that $\begin{pmatrix} \mathcal{B} \\ \mathcal{L} \end{pmatrix} : \ell^2_{\lambda+1} \to \ell^2_\lambda$ is an invertible operator for some $\lambda \in \{D-1, D, \ldots\}$. Then, as $n \to \infty$,*

$$\|A_n R_n\|_{\ell^2_\lambda} = \mathcal{O}(1) \quad \text{and} \quad \|(A_n R_n)^{-1}\|_{\ell^2_\lambda} = \mathcal{O}(1)$$

for the diagonal matrix $R_n = \mathcal{P}_n \mathcal{R} \mathcal{P}_n^\top$ and truncated spectral matrix

$$A_n = \mathcal{P}_n \begin{pmatrix} \mathcal{B} \\ \mathcal{L} \end{pmatrix} \mathcal{P}_n^\top.$$

Proof. Since $\mathcal{R} : \ell^2_\lambda \to \ell^2_{\lambda+1}$ is invertible, we have that $I + \mathcal{K} : \ell^2_\lambda \to \ell^2_\lambda$ is invertible. The lemma follows since $\mathcal{K}$ is compact,

$$A_n R_n = \mathcal{P}_n (I + \mathcal{K}) \mathcal{R}^{-1} \mathcal{P}_n^\top \mathcal{P}_n \mathcal{R} \mathcal{P}_n^\top = I_n + \mathcal{P}_n \mathcal{K} \mathcal{P}_n^\top,$$

and $\mathcal{P}_n^\top \mathcal{P}_n \mathcal{K} \mathcal{P}_n^\top \mathcal{P}_n$ converges in norm to $\mathcal{K}$. ∎

[1] The norm we impose on $\mathbb{C}^K$ does not matter since the space is finite-dimensional, and hence all norms are equivalent.

4.2. Convergence. Denote the coefficients of the exact solution by $\mathbf{u}$, and note that vector $\mathcal{P}_n^\top \mathcal{P}_n \mathbf{u}$ agrees with $\mathbf{u}$ for the first n coefficients and thereafter has zero entries. We show that our numerical scheme converges at the same rate as $\mathcal{P}_n^\top \mathcal{P}_n \mathbf{u}$ converges to $\mathbf{u}$.

THEOREM 4.5. *Suppose* $\mathbf{f} \in \ell^2_{\lambda-N+1}$ *for some* $\lambda \in \{D-1, D, \ldots\}$, *and that* $\binom{\mathcal{B}}{\mathcal{L}} : \ell^2_{\lambda+1} \to \ell^2_\lambda$ *is an invertible operator. Define*

$$\mathbf{u}_n = A_n^{-1} \mathcal{P}_n \begin{pmatrix} \mathbf{c} \\ \mathcal{S}_{N-1} \cdots \mathcal{S}_0 \mathbf{f} \end{pmatrix}.$$

Then

$$\|\mathbf{u} - \mathcal{P}_n^\top \mathbf{u}_n\|_{\ell^2_{\lambda+1}} \le C \|\mathbf{u} - \mathcal{P}_n^\top \mathcal{P}_n \mathbf{u}\|_{\ell^2_{\lambda+1}} \to 0.$$

Proof. Let $\mathbf{v}_n = R_n^{-1} \mathbf{u}_n$ and $\mathbf{v} = \mathcal{R}^{-1} \mathbf{u}$. First note that

$$\begin{pmatrix} \mathcal{B} \\ \mathcal{L} \end{pmatrix} \mathbf{u} = \begin{pmatrix} \mathcal{B} \\ \mathcal{L} \end{pmatrix} \mathcal{R} \mathcal{R}^{-1} \mathbf{u} = (I + \mathcal{K}) \mathbf{v} = \begin{pmatrix} \mathbf{c} \\ \mathcal{S}_{N-1} \cdots \mathcal{S}_0 \mathbf{f} \end{pmatrix} \in \ell^2_\lambda$$

and that

$$\mathbf{v}_n = (A_n R_n)^{-1} \mathcal{P}_n (I + \mathcal{K}) \mathbf{v}.$$

Moreover, since $A_n R_n = \mathcal{P}_n (I + \mathcal{K}) \mathcal{P}_n^\top$,

$$\mathcal{P}_n \mathbf{v} = (A_n R_n)^{-1} \mathcal{P}_n (I + \mathcal{K}) \mathcal{P}_n^\top \mathcal{P}_n \mathbf{v},$$

and thus we have

$$\begin{aligned} \mathbf{v} - \mathcal{P}_n^\top \mathbf{v}_n &= \mathbf{v} - \mathcal{P}_n^\top \mathcal{P}_n \mathbf{v} + \mathcal{P}_n^\top (A_n R_n)^{-1} \mathcal{P}_n (I + \mathcal{K}) \mathcal{P}_n^\top \mathcal{P}_n \mathbf{v} \\ &\qquad - \mathcal{P}_n^\top (A_n R_n)^{-1} \mathcal{P}_n (I + \mathcal{K}) \mathbf{v} \\ &= (I - \mathcal{P}_n^\top (A_n R_n)^{-1} \mathcal{P}_n (I + \mathcal{K}))(\mathbf{v} - \mathcal{P}_n^\top \mathcal{P}_n \mathbf{v}). \end{aligned}$$

Finally, we use the fact that $\|\mathcal{R}^{-1} \mathbf{u}\|_{\ell^2_\lambda} \le \frac{1}{N} \|\mathbf{u}\|_{\ell^2_{\lambda+1}}$ and $\|\mathcal{R} \mathbf{v}\|_{\ell_{\lambda+1}} \le (N+1) \|\mathbf{v}\|_{\ell^2_\lambda}$ to bound the error in the solution by the error in the Chebyshev series of u and its truncation,

$$\begin{aligned} \|\mathbf{u} - \mathcal{P}_n^\top \mathbf{u}_n\|_{\ell^2_{\lambda+1}} &\le (N+1) \|\mathbf{v} - \mathcal{P}_n^\top \mathbf{v}_n\|_{\ell^2_\lambda} \\ &\le (N+1) \left[1 + \left\|(A_n R_n)^{-1}\right\|_{\ell^2_\lambda} \left(1 + \|\mathcal{K}\|_{\ell^2_\lambda}\right)\right] \left\|\mathbf{v} - \mathcal{P}_n^\top \mathcal{P}_n \mathbf{v}\right\|_{\ell^2_\lambda} \\ &\le C \left\|\mathbf{u} - \mathcal{P}_n^\top \mathcal{P}_n \mathbf{u}\right\|_{\ell^2_{\lambda+1}}. \end{aligned}$$

Since $\mathbf{u} \in \ell^2_{\lambda+1}$ we know that $\left\|\mathbf{u} - \mathcal{P}_n^\top \mathcal{P}_n \mathbf{u}\right\|_{\ell^2_{\lambda+1}} \to 0$ as $n \to \infty$. ∎

5. Fast Linear Algebra for Almost Banded Matrices. The spectral method we have described requires the solution of a linear system $Ax = b$, where $A \in \mathbb{C}^{n \times n}$. The matrix A is banded, with $m_{\mathrm{R}} = \mathcal{O}(m)$ nonzero superdiagonals and $m_{\mathrm{L}} = \mathcal{O}(m)$ nonzero subdiagonals (so that $m = m_{\mathrm{L}} + m_{\mathrm{R}} + 1$), except for the first K dense boundary rows. The typical structure of A is shown in Figure 5.1. Here, we describe a stable algorithm to solve $Ax = b$ in $\mathcal{O}(m^2 n)$ operations and with space requirement $\mathcal{O}(mn)$.

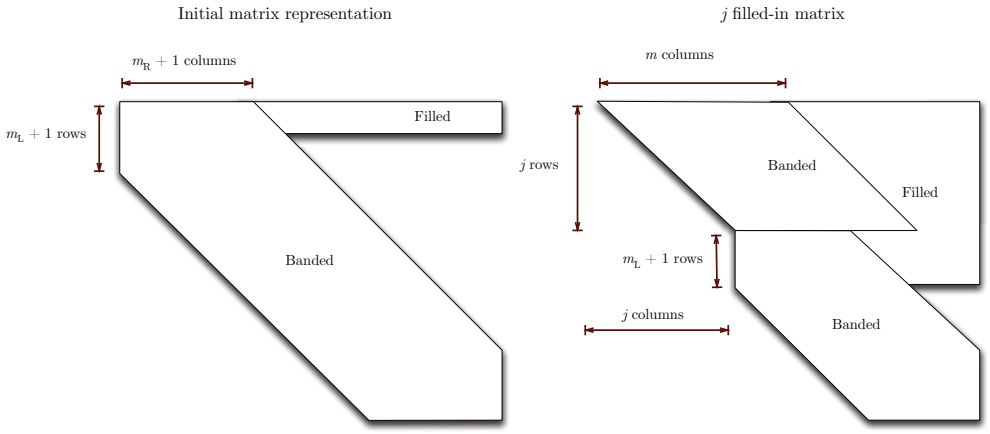

Fig. 5.1 *Structure of operators during the QR factorization. Left: The structure of the original differential operator. Right: A j filled-in matrix obtained after upper triangularizing the first j columns.*

We will solve $Ax = b$ by computing a QR factorization using Givens rotations. However, the resulting upper triangular part will be dense because of *fill-in* caused by the boundary rows. We will show that these dense rows can still be represented sparsely, and that the resulting upper triangular matrix can be solved in $\mathcal{O}(m^2 n)$ operations.

Remark 4. Alternatively, the $A = QRP^*$ factorization can be constructed in $\mathcal{O}(m^2 n)$ operations by applying Givens rotations to the left and the right, which prevents the boundary row(s) causing fill-in [8]. However, with this factorization it is unclear if the optimal truncation n_{opt} and the solution can be determined simultaneously.

5.1. QR Factorization for Filled-In Matrices. Represent the matrix A after the jth stage of the QR factorization, where the jth column has been completely reduced, by B. We claim that B has the form of a *j filled-in matrix*.

DEFINITION 5.1. *B is a j filled-in matrix if, for $k = 1, \ldots, j$, the kth row of B has the form*

$$\mathbf{e}_k^\top B = \Bigg(\overbrace{0, \ldots, 0}^{k-1 \text{ times}}, \overbrace{B_{k,k}, \ldots, B_{k,k+m-1}}^{\text{banded terms}}, \overbrace{\mathbf{b}_k^\top \mathcal{B}_{1:K,k+m:n}}^{\text{fill-in terms}} \Bigg), \tag{5.1}$$

where $\mathbf{b}_k \in \mathbb{C}^K$. Furthermore, every row $k = j+1, \ldots, j+m_{\mathrm{L}}+1$ has the form

$$\mathbf{e}_k^\top B = \Bigg(\overbrace{0, \ldots, 0}^{j \text{ times}}, \overbrace{B_{k,j+1}, \ldots, B_{k,k+m_{\mathrm{R}}}}^{\text{banded terms}}, \overbrace{\mathbf{b}_k^\top \mathcal{B}_{1:K,k+m_{\mathrm{R}}+1:n}}^{\text{fill-in terms}} \Bigg).$$

The remaining rows have the form

$$\mathbf{e}_k^\top B = \Bigg(\overbrace{0, \ldots, 0}^{k-m_{\mathrm{L}}-1 \text{ times}}, \overbrace{B_{k,k-m_{\mathrm{L}}}, \ldots, B_{k,k+m_{\mathrm{R}}}}^{\text{banded terms}}, \overbrace{0, \ldots, 0}^{n-k-m_{\mathrm{R}} \text{ times}} \Bigg). \tag{5.2}$$

In Figure 5.1 we show the structure of a j filled-in matrix. Essentially, it is a banded matrix where the top-right part can be filled with linear combinations of the boundary rows. Note that each row of a filled-in matrix takes at most $m+K$ entries to represent, a bound that is independent of j. Since the QR factorization only applies Givens rotations on the left (i.e., linear combinations of rows), and the initial matrix is a 0 filled-in matrix, the elimination results in j filled-in matrices. This means that throughout the elimination, the matrix can be represented with $\mathcal{O}(mn)$ storage.

In section 5.3, we perform QR factorization adaptively on the operator. This is successful because the representation as a j filled-in matrix is independent of the number of columns, and the rows below the $(j+m_{\mathrm{L}})$th are unchanged from the original operator, and hence can be added during the factorization.

5.2. Back Substitution for Filled-In Matrices. After we have performed n stages of Givens rotations, the first n rows are of the form (5.1) and hence are *upper triangular*. Thus, we can now perform back substitution by truncating the right-hand side. The last m rows consist only of banded terms, and standard back substitution is used to calculate $u_{n-m+1}, \ldots, u_n$. We then note that the kth row imposes the condition

$$B_{k,k}u_k = c_k - \sum_{s=1}^{m} B_{k,k+s}u_{k+s} - \mathbf{b}_k^\top \sum_{s=k+m+1}^{n} \mathcal{B}_{1:K,s}u_s$$

on the solution. We can thus obtain an $\mathcal{O}(mn)$ back substitution algorithm by the following.

Algorithm 1 Back substitution for filled-in matrices.

$\mathbf{p}_{n-m} = \mathbf{0}$ (i.e., a vector of K zeros)
for $k = n - m - 1 \to 1$ **do**
  $\mathbf{p}_k = u_{k+m+1}\mathcal{B}_{1:K,k+m+1} + \mathbf{p}_{k+1}$
  $u_k = \frac{1}{B_{k,k}}\left(c_k - \sum_{s=1}^{m} B_{k,k+s}u_{k+s} - \mathbf{b}_k^\top \mathbf{p}_k\right)$
end for

This is mathematically equivalent to standard back substitution. However, the reduced number of operations decreases the accumulation of round-off error.

5.3. Optimal Truncation. One does not know a priori how many coefficients n_{opt} are required to resolve the solution to relative machine precision. A straightforward algorithm for finding n_{opt} is to progressively increase the discretization size until the difference in the computed coefficients is below a given threshold. Each time the discretization is increased all the coefficients are solved for again. We will present an alternative approach that can compute the optimal truncation and the solution's coefficients simultaneously.

For the simple equation

$$u' + u = f \qquad \text{and} \qquad u(1) = c,$$

the truncation of the operator is tridiagonal with a single dense boundary row. This is equivalent to an *inhomogeneous three-term recurrence boundary value problem*. The problem of adaptively truncating such recurrence relationships is solvable by (F. W. J.) Olver's algorithm [33]. The central idea is to apply row reduction (without pivoting) adaptively. The row reduction applied to the right-hand side, when combined with a

concurrent adaptive computation of the homogeneous solutions to the recurrence relationship, allowed an explicit bound for the *relative error in solution*. An alternative (and simpler) bound for the *absolute error* was obtained in examples. The case of dense boundary rows, with an application to the Clenshaw method [9] as motivation, was also considered. Adaptation of the algorithm to higher order difference equations was developed by Lozier [30].

We adapt this approach to our case by incorporating it into the QR factorization of subsection 5.1. By using QR factorization in place of Gaussian elimination, we avoid potential numerical stability issues of the original Olver's algorithm in the low order coefficients. The key observation is that the boundary terms $\mathcal{B}$ and the rows of the form (5.2) can be evaluated lazily, with bounded computational cost per entry. Thus the truncation parameter n is not involved in the proposed QR factorization algorithm (only in the back substitution), and hence the optimal truncation n_{opt} can be found adaptively.

Represent the Givens rotations that reduces the first n columns of $\mathcal{A}$ by the orthonormal operator $Q_n \in \mathbb{C}^{(n+m_{\text{L}})\times(n+m_{\text{L}})}$, so that

$$\begin{pmatrix} Q_n^\star & \\ & I \end{pmatrix} \mathcal{A} = \begin{pmatrix} R_n & \mathcal{F} \\ & \mathcal{W} \end{pmatrix},$$

where $R_n \in \mathbb{C}^{n\times n}$ is upper triangular. Our right-hand side is

$$\begin{pmatrix} Q_n^\star & \\ & I \end{pmatrix} \begin{pmatrix} \mathbf{c} \\ \mathcal{S}_{N-1}\cdots\mathcal{S}_0\mathbf{f} \end{pmatrix} = \begin{pmatrix} \mathbf{r}_{1:n} \\ \mathbf{r}_{n+1:n+m_{\text{L}}} \\ \mathbf{0} \end{pmatrix},$$

where we use the fact that $\mathbf{f}$ has only a finite number of nonzero entries (due to the initial truncation of its Chebyshev expansion) and assume that $n+m_{\text{L}}$ is greater than the number of nonzero entries. Thus our numerical approximation $\mathbf{u}_n = R_n^{-1}\mathbf{r}_{1:n}$ has the *forward error*

$$\begin{pmatrix} Q_n^\star & \\ & I \end{pmatrix} \mathcal{A} \begin{pmatrix} \mathbf{u}_n \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} \mathbf{r}_{1:n} \\ \mathbf{r}_{n+1:n+m_{\text{L}}} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} R_n\mathbf{u}_n \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} \mathbf{r}_{1:n} \\ \mathbf{r}_{n+1:n+m_{\text{L}}} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{r}_{n+1:n+m_{\text{L}}} \\ \mathbf{0} \end{pmatrix}.$$

Since we calculate $\mathbf{r}_{n+1:n+m_{\text{L}}}$ during the algorithm, we know the error in residual *exactly*. Having a bound on $\|\mathcal{A}^{-1}\|$ allows us to subsequently bound the *error in solution*, i.e., $\|\mathbf{u} - \mathcal{P}_n^\top \mathbf{u}_n\|$. This can be improved further by adapting the procedure of [33, 30], which implicitly calculates an alternative bound based on the homogeneous solutions of the difference equation as part of the algorithm.

In Figure 5.2, we apply the adaptive QR factorization to solve the Airy equation of example (3.10) with $\epsilon = 1, 10^{-1}, 10^{-2}, \ldots, 10^{-13}$, as well as

$$u'' + (7 + 2x + 6x^2)u = \sum_{k=0}^{\nu} T_k(x) \quad \text{for} \quad u(-1) = 1, \quad u(1) = 1, \tag{5.3}$$

$$u'' + (\cos x)\, u = \sum_{k=0}^{\nu} T_k(x) \quad \text{for} \quad u(-1) = 1, \quad u(1) = 1 \tag{5.4}$$

for increasing values of ν, up to 2 million. In the last example, we replace $\cos x$ with its 13-point Chebyshev interpolating polynomial. We plot the number of seconds

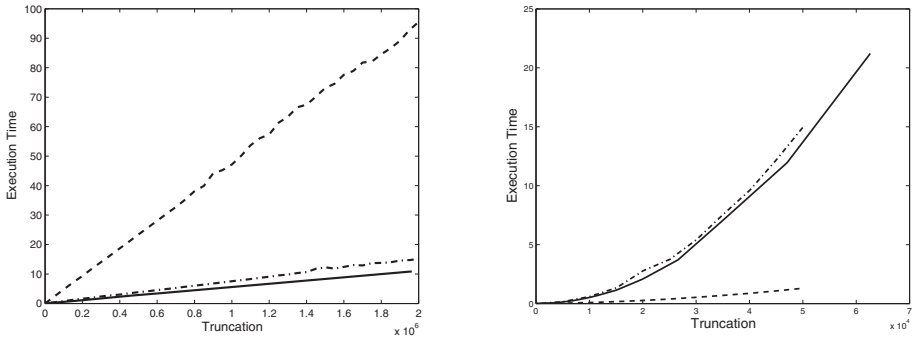

Fig. 5.2 *Time versus optimal truncation, showing linear computational cost for the* C++ *implementation (left) and the MATLAB implementation (right). Examples are* (3.10) *(solid),* (5.3) *(dot-dashed), and* (5.4) *(dashed).*

the calculation takes versus the adaptively calculated optimal truncation n_{opt}. This demonstrates the $\mathcal{O}(n_{\text{opt}})$ complexity of the algorithm, and the fact that the algorithm easily scales to more than a million unknowns. It also shows that, while $\mathcal{O}(n_{\text{opt}})$ complexity is maintained, the computational cost does increase with the bandwidth of the variable coefficient. In the Airy example (3.10), the time taken for $\epsilon = 10^{-13}$ is less than 11 seconds,[2] with the n_{opt} calculated to be approximately 2 million. For example (5.4), a calculation resulting in n_{opt} being 2 million increases the timing to 95 seconds.

On the right of Figure 5.2 we plot the timing of MATLAB's built-in sparse LU solver applied to the truncated equation, with $n = n_{\text{opt}}$ prespecified. Even without the added difficulty of calculating n_{opt}, the computational cost grows faster than $\mathcal{O}(n)$, prohibiting its usefulness for extremely large n.

Remark 5. We calculated Figure 5.2 using C++. There is a great deal of room for optimizing the implementation, as we do not use GPU, parallel, or vector processing units.

5.4. Linear Algebra Stability in Higher Order Norms. We first remark that if $\mathcal{B}$ is a bounded operator from $\ell_1^2 \to \ell_0^2$, such as Dirichlet boundary conditions, then the results of section 4 prove that the *preconditioned* linear system has bounded 2-norm condition number as $n \to \infty$. Because the QR factorization is computed using Givens rotations that are stable in ℓ_0^2 [27], as is backward substitution, we see that the linear algebra scheme applied to the preconditioned operator is stable, and has $\mathcal{O}(m^2 n)$ complexity.

The results of section 4 also show convergence and well-conditioning in high order norms. One would expect numerical round-off in QR factorization to destroy this convergence property. However, in practice, this is not the case. In Figure 5.3, we solve the standard Airy equation as a two-point boundary value problem:

$$u'' - xu = 0, \quad u(-1) = \text{Ai}(-1) \text{ and } u(1) = \text{Ai}(1).$$

We witness convergence in higher order norms as well. In other words, the computed solution has a fast decaying tail, and all of the *absolute* error is in the low order coefficients.

[2]CPU times were calculated on a 2011 iMac, with a 2.7 GHz Intel Core i5 CPU.

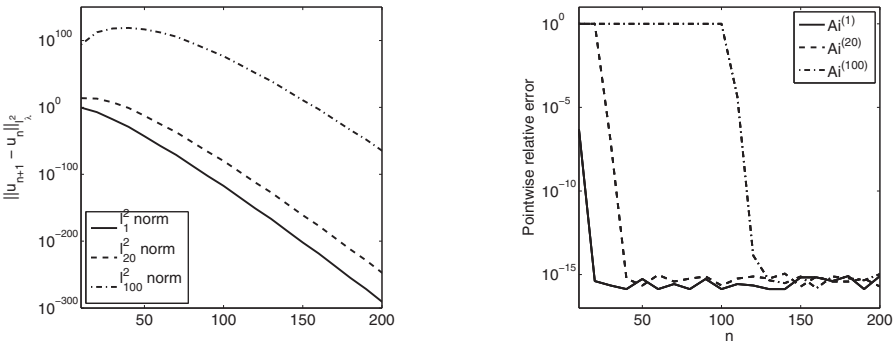

Fig. 5.3 *Left: The Cauchy error of the solution coefficients measured in the ℓ_1^2-norm (solid), ℓ_{20}^2-norm (dashed), and ℓ_{100}^2-norm (dot-dashed). Right: The relative error in derivatives of the solution at $x = -\frac{1}{2}$ for the 1st (solid), 20th (dashed), and 100th (dot-dashed) derivative. This plot shows that for sufficiently large discretizations very high derivatives of the solution can be resolved.*

Convergence in higher order norms implies convergence of derivatives, and this is verified by differentiating the computed solution by applying $\mathcal{S}_0^{-1}\mathcal{D}_1$ repeatedly. (We note that the banded, upper triangular nature of $\mathcal{S}_0$ means that its inverse applied to a vector is computable in $\mathcal{O}(n)$ operations; after all, this corresponds to differentiating without converting bases, which can be done in $\mathcal{O}(n)$ operations [32].) We compare the computed derivatives with the true derivatives of the Airy function $\mathrm{Ai}^{(p)}$ at a single point, $x = -1/2$, and witness convergence for $p = 1, 5$, and 20.

We note that the stability of the QR algorithm in higher order norms appears to follow from Q being almost banded: it is banded along the superdiagonal, and decays exponentially along the subdiagonals. In the case where the boundary conditions are such that A itself is banded, exponential decay in $Q = AR^{-1}$ follows since R^{-1} has exponential decay, due to R being banded and well-conditioned [13].

Since ℓ_λ^2 is a Hilbert space, Givens rotations can be constructed with the relevant inner product, resulting in orthogonal operations (i.e., with condition number one) in ℓ_λ^2. The stability of such an algorithm in ℓ_λ^2 follows immediately. With this modification, we have an $\mathcal{O}(m^2 n)$ stable algorithm that is guaranteed to converge in higher order norms.

Remark 6. While we have discussed the convergence in higher order norms with Dirichlet boundary conditions, the exact same logic applies to the convergence and stability observed with higher order boundary conditions, such as Neumann conditions.

6. Future Work. We have designed a spectral method that achieves $\mathcal{O}(n)$ computational cost, stability, and generality for solving linear ODEs. We determined the optimal truncation adaptively using the QR factorization. We believe that the ideas introduced in this paper will serve as a basis for future spectral methods.

An exciting generalization of this work will be to higher dimensions following a similar approach based on boundary recombination for the Helmholtz equation [41]. Adapting our method to rectangular domains, using tensor products of ultraspherical polynomials, results in a tensor product of almost banded matrices. Constructing an adaptive QR factorization to such matrices will be crucial for achieving competitive

computational costs, and for optimally choosing the number of unknowns needed in each dimension.

Using the theory of [40], there are potentially generalizations of ultraspherical polynomials to deltoid domains. What is less clear is how the results would be generalizable to other domains, such as the triangle.

For problems with boundary layers, or localized oscillatory behavior, it can be more efficient to subdivide the unit interval, in order to minimize the total number of unknowns. This will be of fundamental importance for PDEs, where solutions typically have singularities at corners. In the one-dimensional case, it is straightforward to incorporate subdivision by representing the operators as block matrices, with additional boundary rows to impose continuity. Whether the adaptive QR factorization can be easily generalized to such matrices is less clear.

Finally, an extension of this work is to nonlinear differential equations, of the form

$$\mathcal{B}u = \mathbf{0} \qquad \text{and} \qquad \mathcal{L}u + g(u) = f.$$

Our approach can be incorporated into an infinite-dimensional Newton iteration, à la [2, 18]. The Newton iteration takes the form

$$u_{k+1} = u_k - \begin{pmatrix} \mathcal{B} \\ \mathcal{L} + g'(u_k) \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{0} \\ \mathcal{L}u_k + g(u_k) - f \end{pmatrix}.$$

Since the linear operator that is inverted involves the solution itself, the bandedness of multiplication is lost, at least when naïvely implemented. It may be possible to overcome this difficulty by using the fact that the linearization used in the Newton iteration need not be accurate to machine precision. Moreover, the decay in the coefficients of the operator can combine with decay in the solution; hence the entries of the operator can be truncated more aggressively while maintaining accuracy. However, even with a dense representation of the operator, the stability of the resulting algorithm is preserved in initial numerical experiments.

Acknowledgments. We thank P. Gonnet, discussions with whom led to the observation of well-conditioning of coefficient methods, which initiated the research of this paper. We also thank the rest of the Chebfun team, including T. A. Driscoll, N. Hale, and L. N. Trefethen for valuable feedback. We thank J. P. Boyd and the anonymous reviewers for very useful comments and suggestions. We finally thank D. Lozier and F. W. J. Olver for discussions and references related to Olver's algorithm.

REFERENCES

[1] G. Baszenski and M. Tasche, *Fast polynomial multiplication and convolutions related to the discrete cosine transform*, Linear Algebra Appl., 252 (1997), pp. 1–25.

[2] A. Birkisson and T. A. Driscoll, *Automatic Fréchet differentiation for the numerical solution of boundary-value problems*, ACM Trans. Math. Software, 38 (2012), 26.

[3] C. M. Bender and S. A. Orszag, *Advanced Mathematical Methods for Scientists and Engineers*, McGraw–Hill, New York, 1978.

[4] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, Dover, Mineola, NY, 2001.

[5] C. Canuto and A. Quarteroni, *Preconditioned minimal residual methods for Chebyshev spectral calculations*, J. Comput. Phys., 60 (1985), pp. 315–337.

[6] C. Canuto, *Spectral Methods: Fundamentals in Single Domains*, Springer, Berlin, 2006.

[7] L. Carlitz, *The product of two ultraspherical polynomials*, Proc. Glasgow Math. Assoc., 5 (1961), pp. 76–79.

[8] S. Chandrasekaran and M. Gu, *Fast and stable algorithms for banded plus semiseparable systems of linear equations*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 373–384.
[9] C. W. Clenshaw, *The numerical solution of linear differential equations in Chebyshev series*, Math. Proc. Cambridge Philos. Soc., 53 (1957), pp. 134–149.
[10] C. W. Clenshaw and A. R. Curtis, *A method for numerical integration on an automatic computer*, Numer. Math., 2 (1960), pp. 197–205.
[11] E. A. Coutsias, T. Hagstrom, and D. Torres, *An efficient spectral method for ordinary differential equations with rational function coefficients*, Math. Comp., 65 (1996), pp. 611–635.
[12] P. J. Davis, *Interpolation and Approximation*, Dover, New York, 1975.
[13] S. Demko, W. F. Moss, and P. W. Smith, *Decay rates for inverse of band matrices*, Math. Comp., 43 (1984), pp. 491–499.
[14] M. Deville and E. Mund, *Chebyshev pseudospectral solution of second-order elliptic equations with finite element preconditioning*, J. Comput. Phys., 60 (1985), pp. 517–533.
[15] E. H. Doha and W. M. Abd-Elhameed, *Efficient spectral-Galerkin algorithms for direct solution of second-order equations using ultraspherical polynomials*, SIAM J. Sci. Comput., 24 (2002), pp. 548–571.
[16] E. H. Doha, *On the construction of recurrence relations for the expansion and connection coefficients in series of Jacobi polynomials*, J. Phys. A, 37 (2004), pp. 657–675.
[17] E. H. Doha and W. M. Abd-Elhameed, *Efficient spectral ultraspherical-dual-Petrov–Galerkin algorithms for the direct solution of* $(2n+1)$*th-order linear differential equations*, Math. Comput. Simulation, 79 (2009), pp. 3221–3242.
[18] T. A. Driscoll, F. Bornemann, and L. N. Trefethen, *The chebop system for automatic solution of differential equations*, BIT, 48 (2008), pp. 701–723.
[19] T. A. Driscoll, *Automatic spectral collocation for integral, integro-differential, and integrally reformulated differential equations*, J. Comput. Phys., 229 (2010), pp. 5980–5998.
[20] E. M. E. Elbarbary, *Integration preconditioning matrix for ultraspherical pseudospectral operators*, SIAM J. Sci. Comput., 28 (2006), pp. 1186–1201.
[21] B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, Cambridge, UK, 1998.
[22] L. Fox, *Chebyshev methods for ordinary differential equations*, Comput. J., 4 (1962), pp. 318–331.
[23] D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*, SIAM, Philadelphia, 1977.
[24] L. Greengard, *Spectral integration and two-point boundary value problems*, SIAM J. Numer. Anal., 28 (1991), pp. 1071–1080.
[25] A. C. Hansen, *Infinite dimensional numerical linear algebra; theory and applications*, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci., 466 (2008) pp. 3539–3559.
[26] J. S. Hesthaven, *Integration preconditioning of pseudospectral operators. I. Basic linear operators*, SIAM J. Numer. Anal., 35 (1998) pp. 1571–1593.
[27] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.
[28] C. Lanczos, *Trigonometric interpolation of empirical and analytical functions*, J. Math. Phys., 17 (1938) pp. 123–199.
[29] J.-Y. Lee and L. Greengard, *A fast adaptive numerical method for stiff two-point boundary value problems*, SIAM J. Sci. Comput., 18 (1997) pp. 403–429.
[30] D. W. Lozier, *Numerical Solution of Linear Difference Equations*, NBSIR Technical Report 80-1976, National Bureau of Standards, 1980.
[31] H. Ma and W. Sun, *A Legendre–Petrov–Galerkin and Chebyshev collocation method for third-order differential equations*, SIAM J. Numer. Anal., 38 (2000), pp. 1425–1438.
[32] J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials*, Chapman & Hall/CRC, Boca Raton, FL, 2003.
[33] F. W. J. Olver, *Numerical solution of second-order linear difference equations*, J. Res. Nat. Bur. Standards Sect. B, 71 (1967), pp. 111–129.
[34] F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, *NIST Handbook of Mathematical Functions*, Cambridge University Press, Cambridge, UK, 2010.
[35] S. Olver and A. Townsend, https://github.com/dlfivefifty/Ultraspherical.
[36] S. A. Orszag, *Spectral methods for problems in complex geometries*, J. Comput. Phys., 37 (1980), pp. 70–92.
[37] S. A. Orszag, *Accurate solution of the Orr–Sommerfeld stability equation*, J. Fluid Mech., 50 (1980), pp. 689–703.
[38] E. L. Ortiz, *The tau method*, SIAM J. Numer. Anal., 6 (1969), pp. 480–492.

[39] P. D. Ritger and N. J. Rose, *Differential Equations with Applications*, Dover, 2000.
[40] B. N. Ryland and H. Z. Munthe-Kaas, *On multivariate Chebyshev polynomials and spectral approximations on triangles*, in Spectral and High Order Methods for Partial Differential Equations, Lect. Notes Comput. Sci. Eng. 76, Springer, Berlin, Heidelberg, 2011, pp. 19–41.
[41] J. Shen, *Efficient spectral-Galerkin method* II. *Direct solvers of second- and fourth-order equations using Chebyshev polynomials*, SIAM J. Sci. Comput., 16 (1995), pp. 74–87.
[42] J. Shen, *A new dual-Petrov–Galerkin method for third and higher odd-order differential equations: Application to the KdV equation*, SIAM J. Numer. Anal., 41 (2003), pp. 1595–1619.
[43] J. Shen and L. L. Wang, *Legendre and Chebyshev dual-Petrov–Galerkin methods for hyperbolic equations*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 3785–3797.
[44] J. Shen, T. Tang, and L. L. Wang, *Spectral Methods: Algorithms, Analysis and Applications*, Springer, Heidelberg, 2009.
[45] L. N. Trefethen, *Spectral Methods in MATLAB*, SIAM, Philadelphia, 2000.
[46] L. N. Trefethen et al., *Chebfun Version* 4.2, The Chebfun Development Team, 2011, http://www.maths.ox.ac.uk/chebfun/.
[47] D. Viswanath, *Spectral Integration of Linear Boundary Value Problems*, preprint, arXiv: 1205.2717, 2012.
[48] J. Waldvogel, *Fast construction of the Fejér and Clenshaw–Curtis quadrature rules*, BIT, 46 (2006), pp. 195–202.
[49] J. A. Weideman and S. C. Reddy, *A MATLAB differentiation matrix suite*, ACM Trans. Math. Software, 26 (2000), pp. 465–519.
[50] A. Zebib, *A Chebyshev method for the solution of boundary value problems*, J. Comput. Phys., 53 (1984), pp. 443–455.