IMA Journal of Numerical Analysis (2015) Page 1 of 25
doi:10.1093/imanum/dru062

Rectangular spectral collocation

Tobin A. Driscoll
Department of Mathematical Sciences, University of Delaware, Newark, DE 19716, USA
driscoll@udel.edu
http://www.math.udel.edu/∼driscoll/

and

Nicholas Hale*
Department of Mathematical Sciences, University of Stellenbosch, Stellenbosch, South Africa
*Corresponding author: nickhale@sun.ac.za
http://www.dip.sun.ac.za/∼nhale/

[Received on 29 August 2014; revised on 16 December 2014]

Boundary conditions in spectral collocation methods are typically imposed by removing some rows of the discretized differential operator and replacing them with others that enforce the required conditions at the boundary. A new approach based upon resampling differentiated polynomials into a lower-degree subspace makes differentiation matrices, and operators built from them, rectangular without any row deletions. Then, boundary and interface conditions can be adjoined to yield a square system. The resulting method is both flexible and robust, and avoids ambiguities that arise when applying the classical row deletion method outside of two-point scalar boundary-value problems. The new method is the basis for ordinary differential equation solutions in Chebfun software, and is demonstrated for a variety of boundary-value, eigenvalue and time-dependent problems.

Keywords: spectral method; collocation; boundary conditions; Chebyshev.

1. Introduction

Chebyshev spectral collocation, or pseudospectral, methods have a well-established presence in the numerical solution of differential equations of Boyd (2001), Canuto *et al.* (1988), Fornberg (1996) and Trefethen (2000). For problems with smooth or piecewise smooth solutions, they offer rapid convergence (typically exponential) in the number of degrees of freedom, and can be used in conjunction with asymptotically fast and stable algorithms such as the FFT and barycentric interpolation formula of Berrut & Trefethen (2004) and Higham (2004). Compared with other formulations of spectral methods, such as Galerkin approximations, they offer convenience in the presence of variable coefficients and nonlinearities.

The paradigm of spectral collocation for differential equations is straightforward. Functions, including the unknown solution of a differential equation, are represented as finite vectors of their values at certain points. Each such vector implicitly defines a global interpolant, and operations on functions are replaced by the same operations applied to the corresponding interpolants. Because representations of fixed length N are used for every function in the formulation, linear operators are replaced by $N \times N$ matrices, such as those that stand in for differentiation and integration operators. After all, such

operations are discretized, the result is an $N \times N$ algebraic system of equations, in the case of a scalar boundary-value problem, or an N-dimensional eigenvalue problem, or system of ordinary differential equations (ODEs) in other contexts.

When solving a two-point boundary-value problem (TPBVP), two more side conditions must be applied to the solution. The classical solution is to remove the top and bottom rows of the discretized operator and replace them with discrete versions of the boundary conditions. This strategy has proved successful in a wide variety of problems. However, the precise nature of how the ODE is imposed is a curious one, involving discretization of the residual at the $N - 2$ interior points of an N-point second-kind Chebyshev grid.

The original motivation for our considering the rectangular point of view was the implementation of differential equation solvers in Chebfun (Driscoll *et al.*, 2014), which aims to enable fully automatic numerical computation using Chebyshev polynomial interpolants. When one considers how to discretize and solve as wide a variety as possible of higher-order problems, systems of equations, nonseparated and global side conditions, and so on, it quickly becomes clear that the row replacement strategy is an *ad hoc* one outside of the familiar two-point, second-order problem. For example, the imposition of boundary conditions by row replacement in fourth-order problems requires rows other than those at the boundary to be removed.

Various ways to address this situation have been suggested. Individual classical problems may be solved by clever manipulation of the underlying polynomial representation (Trefethen, 2000, Chapter 14), which is difficult to generalize. The problem can be fixed at least somewhat more generally by the fictitious point manipulations of Fornberg (2006), requiring a rewrite of algorithms and having unknown numerical stability. A different type of approach is to soften the boundary conditions to penalty terms (Hesthaven & Gottlieb, 1996), which introduces free parameters that may be difficult to pin down in general.

In this work, we propose to dispense with row replacements and instead map between representations by values at different standard Chebyshev grids. In an mth-order differential operator, the map is between Chebyshev grids of lengths $N + m$ and N. Because we break with the tradition of row replacements, the traditional preference for Chebyshev grids that include boundary points no longer seems important, and we advise the use of first-kind points (i.e., Chebyshev–Gauss rather than Chebyshev–Gauss–Lobatto—see equations (3.5) and (3.3), respectively), which are more convenient in some cases and essential for our implementation of the method of lines.

In Section 2, we provide some further motivation as to why well-trodden ideas of square differentiation matrices and row replacement might need revisiting, and in Section 3 we show a practical and stable way to implement rectangular operators as an extra step following traditional collocation manipulations. In Section 4, we describe how to use rectangular methods to solve boundary-value problems of considerable generality, and in Sections 5 and 6 extend the technique to eigenvalue and time-dependent problems. In Section 7, we discuss implementation issues and demonstrate the results on challenging examples. Finally, in Section 8, the method is shown to have strong connections to other forms of spectral methods.

2. The rectangularity of differentiation and integration

The basis for our algorithms is a view that differentiation and integration by square matrices is unnatural for the polynomial interpolants that underlie spectral methods, as these operations change the degree of a polynomial.

TABLE 1 *Singular values of matrices representing successive differentiation and integration in finite dimensions. In the first two columns are results using traditional square collocation matrices. In the last two columns are results for rectangular forms of the operators. The last column is particularly notable: integration followed by differentiation yields the identity operator, which is not the case for the traditional square methods.*

Q_5D_5	D_5Q_5	$\tilde{Q}_{5,4}\tilde{D}_{4,5}$	$\tilde{D}_{5,6}\tilde{Q}_{6,5}$
2.23606797749979	1.24096736459909	2.23606797749979	1.00000000000000
1.00000000000000	1.00000000000000	1.00000000000000	1.00000000000000
1.00000000000000	1.00000000000000	1.00000000000000	1.00000000000000
1.00000000000000	1.00000000000000	1.00000000000000	1.00000000000000
0.00000000000000	0.00000000000000	0.00000000000000	1.00000000000000

To illustrate the significance of this viewpoint, let D_N and Q_N be standard Chebyshev spectral differentiation and integration matrices, as defined in Weideman & Reddy (2000) and Greengard (1991), respectively. (The integration matrix is chosen so that the resulting antiderivative is zero at the left endpoint, $x = -1$.) The first two columns of Table 1 show the singular values for the products Q_5D_5 and D_5Q_5. Neither of these products yields an identity matrix. In the former case, two singular values are different from unity because differentiation irrevocably destroys the constant term; the choice of integration constant means that one singular value is zero. The results for the latter case, D_5Q_5, are more disquieting. The essential problem here is that exact integration of a fourth-degree polynomial creates a term that the five-point interpolant cannot represent accurately, causing it to be aliased to a polynomial of degree 3 (Trefethen, 2012).[1]

Now suppose that $\tilde{D}_{N-1,N}$ is an $N-1 \times N$ differentiation matrix that maps a representation on N Chebyshev points (and thus underlying polynomial interpolant of degree less than N) to a representation on $N-1$ points (and an interpolant of degree $< N-1$). Similarly, let $\tilde{Q}_{N+1,N}$ be an integration matrix that maps from N points to $N+1$ points. The construction of these matrices is considered in Section 3.3; both allow the exact expression of the underlying operators starting from the original N-point discretization. The last two columns of Table 1 show the singular values for the 5×5 products $\tilde{Q}_{5,4}\tilde{D}_{4,5}$ and $\tilde{D}_{5,6}\tilde{Q}_{6,5}$. The former is no different from that in the square case, because the lost information is inherent to the exact process of differentiation. The latter product, however, is now the identity matrix, as we should expect.

The inherent rectangularity of differentiation in particular is subtly acknowledged in the imposition of boundary or other side conditions for differential equations using the row replacement or 'boundary bordering' strategy (see Boyd, 2001, Section 6.4; Trefethen, 2000, p. 135). Each side constraint boils down to an algebraic statement about polynomial interpolants. But the square representation of the differential equation, when combined with the algebraic constraints, exceeds the number of available degrees of freedom. The usual response is to delete some equations from the original square system—typically, those representing the differential equation at boundary points. In the case of a second-order TPBVP, for instance, one ODE equation is deleted at each boundary so that the total system remains square.

[1] The choice of truncation versus aliasing of the highest degree term in spectral integration is arbitrary, but in neither case is the original information perfectly retained.

3. Spectral resampling matrices

3.1 *The barycentric resampling matrix*

A key ingredient in our suggested approach is an efficient and stable means to transform a polynomial interpolant, as represented by a vector of its values at a set of points, to its representation as values on a different set of points. We call this process the *resampling* of a polynomial interpolant. The barycentric interpolation formula (Berrut & Trefethen, 2004) is an ideal tool for this task. Given a set of points $\mathbf{x} = \{x_k\}_{k=1}^N$ and the barycentric weights

$$w_k = \prod_{\substack{l=1 \\ l \neq k}}^{N} (x_k - x_l)^{-1}, \quad k = 1, \ldots, N, \tag{3.1}$$

the unique polynomial p_{N-1} of degree $< N$ interpolating the data $\{(x_j, f_j)\}_{j=1}^N$ is given by

$$p_{N-1}(y) = \frac{\sum_{k=1}^N (w_k/(y - x_k)) f_k}{\sum_{l=1}^N (w_l/(y - x_l))}. \tag{3.2}$$

Since the interpolant depends linearly on the values $\{f_j\}$, the evaluation of p_{N-1} at a set of points $\mathbf{y} = \{y_j\}_{j=1}^{N-m}$ can be expressed as the matrix–vector multiplication

$$p_{N-1}(\mathbf{y}) = P_{N,-m} p_{N-1}(\mathbf{x}),$$

where $P_{N,-m} \in \mathbb{R}^{N-m \times N}$ is the *barycentric resampling matrix*

$$(P_{N,-m})_{j,k} = \begin{cases} \dfrac{w_k}{y_j - x_k} \left(\displaystyle\sum_{l=1}^{N} \frac{w_l}{y_j - x_l} \right)^{-1} : & y_j \neq x_k, \\ 1 : & y_j = x_j. \end{cases}$$

The vector $P_{N,-m} p_{N-1}(\mathbf{x})$ then defines a unique polynomial interpolant p_{N-m-1} of the data $\{(y_j, p_{N-1}(y_j))\}_{j=1}^{N-m}$ if x_k and f_k in (3.1) and (3.2) are replaced by y_j and $p_{N-1}(y_j)$, respectively.

Note that, for $m > 0$, resampling results in an irrevocable loss of information, as the new vector of function values is shorter than the original; see Section 3.2. We call this effect *downsampling*. One can also apply resampling for $m < 0$, or *upsampling*, in which case the original polynomial interpolant is preserved, but represented on a larger set of points than is required.

Conditioning and stability properties of P follow directly from results in Higham (2004), and the matrix $P_{N,-m}$ can be easily computed in MATLAB with just a few lines of code, as shown in Fig. 1.

The most widely used spectral methods for nonperiodic problems are those based on Chebyshev polynomials and Chebyshev points (Fornberg, 1996; Trefethen, 2000; Boyd, 2001), and it is these methods on which we focus for the remainder of this paper. (However, everything we discuss is equally valid in the case of collocation with, say, Legendre polynomials and points.) Here, and throughout, we use the term *Chebyshev points* to mean the extrema of first-kind Chebyshev polynomials T_{N-1} on $[-1, 1]$ (sometimes referred to as Chebyshev–Gauss–Lobatto or second-kind points),

$$x_k = \cos\left(\frac{(j-1)\pi}{N-1} \right), \quad k = 1, \ldots, N. \tag{3.3}$$

```
function P = barymat(y, x, w)
    P = bsxfun(@minus, y, x');          % All y(j)-x(k)
    P = bsxfun(@rdivide, w' ,P);        % All w(k)/(y(j)-x(k))
    P = bsxfun(@rdivide, P, sum(P, 2)); % Normalization
    P(isnan(P)) = 1;                    % Remove NaNs
end
```

FIG. 1. MATLAB code to generate a barycentric resampling matrix. The built-in MATLAB function `bsxfun` applies the given operator (`minus`, etc.) in an element-by-element fashion across each column or row of the input vectors.

The barycentric weights $\mathbf{w} = \{w_k\}_{k=1}^N$ for these points are known explicitly,

$$w_k = \begin{cases} \frac{1}{2}, & k = 1, \\ (-1)^k, & k = 2, \ldots, N-1, \\ (-1)^N/2, & k = N. \end{cases} \tag{3.4}$$

We shall also make use of first-kind Chebyshev points, $\hat{\mathbf{x}} = \{\hat{x}_j\}_{j=1}^{N-1}$ (known sometimes as Chebyshev 'roots' or 'nodes'), which are the roots of the Chebyshev polynomial $T_{N-1}(x)$. These are given explicitly by

$$\hat{x}_j = \cos\left(\frac{(2j-1)\pi}{2(N-1)}\right), \quad j = 1, \ldots, N-1. \tag{3.5}$$

3.2 *Resampling and aliasing*

The phenomenon of aliasing in Chebyshev polynomials is well understood (Trefethen, 2012, Chapter 4), and here we demonstrate its relation to the downsampling process described above. We begin by quoting a result for second-kind discretizations.

THEOREM 3.1 (Trefethen, 2012, Theorem 4.1) For any $N \geqslant 1$ and $0 \leqslant k \leqslant N$, the following Chebyshev polynomials take the same values on the $(N+1)$-point second-kind Chebyshev grid:

$$T_k, \quad T_{2N\pm k}, \quad T_{4N\pm k}, \quad T_{6N\pm k}, \quad \ldots.$$

One can show a similar result for Chebyshev polynomials at first-kind points.

THEOREM 3.2 For any $N \geqslant 1$ and $0 \leqslant k \leqslant N$, the following Chebyshev polynomials take the same values on the N-point first-kind Chebyshev grid:

$$T_k, \quad -T_{2N\pm k}, \quad T_{4N\pm k}, \quad -T_{6N\pm k}, \quad \ldots.$$

Proof. The proof is similar to that of Theorem 3.1, but with the observation that $\exp(\pm(2j+1)i\pi l) = (-1)^l$ for any integers j and l. □

Now, consider downsampling a polynomial

$$p_{N-1} = \sum_{k=0}^{N-1} a_k T_k(x)$$

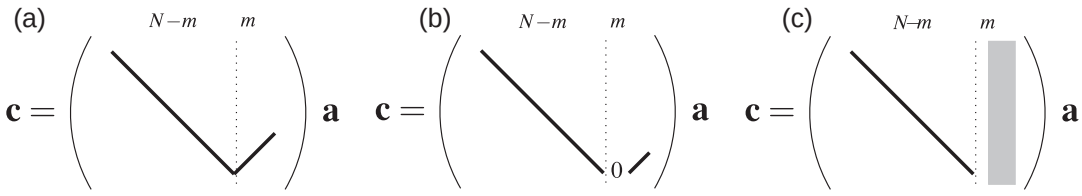

FIG. 2. Illustration of the aliased Chebyshev coefficients in downsampling a polynomial of degree $N-1$ via interpolation at $N-m$ Chebyshev points of the second-kind (a), first-kind (b) and the N-point second-grid with m end-points removed (i.e., row replacement) (c). Compare with equations (3.6) and (3.7).

to the degree $N-m-1$ polynomial

$$p_{N-m-1} = \sum_{k=0}^{N-m-1} c_k T_k(x)$$

by interpolating at $N-m$ points (where, for simplicity, we restrict $m < N/2$). It follows from Theorem 3.1 and (Trefethen, 2012, Theorem 4.2) that

$$c_k = \begin{cases} a_k, & k = 0, \ldots, N-2m-2, \text{ and } N-m-1, \\ a_k + a_{2(N-m)-(k+1)}, & k = N-2m-1, \ldots, N-m-2, \end{cases} \tag{3.6}$$

if the $N-m$ points are second-kind Chebyshev points, and from Theorem 3.2 that

$$c_k = \begin{cases} a_k, & k = 0, \ldots, N-2m, \\ a_k - a_{2(N-m)-(k-1)}, & k = N-2m+1, \ldots, N-m-1, \end{cases} \tag{3.7}$$

for first-kind Chebyshev points. The aliased coefficients of equations (3.6) and (3.7) are illustrated graphically in Fig. 2. Importantly, the zero in the $(N-m, N-m+1)$ entry for the first-kind point interpolation leads to the following corollary.

COROLLARY 3.3 The downsampling of $p_{N-1} = \sum_{k=0}^{N-1} a_k T_k(x)$ to a grid of $N-1$ Chebyshev points of the first kind is equivalent to truncating the highest-order Chebyshev coefficient a_{N-1}.

Proof. The result follows immediately from Theorem 3.2 and Fig. 2(b), and is also evident from the fact that the Chebyshev polynomial $T_{N-1}(x)$ is aliased to zero (by definition!) on the new grid. □

Conversely, we see in Fig. 2(c) that interpolating at the original N-point grid with m points at the boundary removed, as in the traditional row replacement strategy, aliases the high-order coefficients to *all* lower degree coefficients (proof omitted).

3.3 *Rectangular operators*

We can now use the resampling matrices from Section 3.1 to form rectangular versions of the differentiation and integration operators. We begin with differentiation. The standard square Chebyshev spectral differentiation matrix D_N maps function values of a polynomial defined at $\{x_k\}_{k=1}^N$, the second-kind Chebyshev points (3.3), to the values of its derivative on this same grid (Trefethen, 2000, Chapter 6).

For an mth-order derivative, $(D_N)^m$ is applied.[2] Left-multiplying $(D_N)^m$ by the downsampling matrix $P_{N,-m}$, with $\{w_k\}_{k=1}^N$ from (3.4) and $\{y_j\}_{j=1}^{N-m}$ as the first-kind Chebyshev grid (3.5), resamples the polynomial values at $N-m$ points. Hence, $P_{N,-m}(D_N)^m \in \mathbb{R}^{(N-m)\times N}$ is a rectangular mth-order differentiation matrix, mapping values of a polynomial defined at an N-point second-kind Chebyshev grid to the values of its derivative on an $N-m$ point first-kind Chebyshev grid.[3] This particular application of downsampling does not result in the loss of any information because the result of multiplying by $(D_N)^m$ is the discretization of a polynomial of degree $< N-m$, which can be represented exactly on $N-m$ points.

For indefinite integration, let Q_N be the standard Chebyshev spectral integration matrix on an N-point second-kind grid, as defined in Greengard (1991), for example. It will not be sufficient to upsample the result of applying Q_N, for the map from N points to N points cannot preserve information about increasing the degree of the original interpolant. Instead, to properly integrate a polynomial represented on an N-point grid, we have to upsample *first* to $N+m$ points, and then apply square integration on the larger grid; $(Q_{N+m})^m P_{N,m} \in \mathbb{R}^{(N+m)\times N}$ is the appropriate rectangular integration operator. Hence, the matrix $\tilde{D}_{5,6}\tilde{Q}_{6,5}$ whose singular values are reported in Table 1 can be written as

$$\tilde{D}_{5,6}\tilde{Q}_{6,5} = P_{6,-1}D_6Q_6P_{5,1},$$

which produces correct results for all vectors representing interpolants on five points; i.e., the 5×5 identity matrix.

We note that using (3.6) or (3.7), one could apply $P_{N,-m}$ using discrete cosine transforms, which may be more efficient when N is large. Furthermore, one could replace the matrices in Fig. 2 with a truncated identity to form a resampling matrix which truncates rather than aliases. However, we consider the idea of downsampling to the interpolant on a smaller Chebyshev grid to be the easiest to motivate conceptually, and a detailed comparison is beyond the scope of this paper. We also note that it is possible to form the rectangular differentiation matrices $P_{N,-m}(D_N)^m$ directly (Xu & Hale, 2014), but our aim here is to demonstrate that the collocation matrices representing more general operators can be easily adapted by premultiplying by $P_{N,-m}$, as discussed in the next section.

4. Rectangular collocation for boundary-value problems

In this section, we show how resampling and rectangular differentiation can be used to solve boundary-value problems. For scalar problems, the process is clear and unambiguous, unlike for the boundary bordering method. For systems of equations the process is a little more complicated, but the ambiguity is still removed. We begin with the simplest cases and then add complexity.

4.1 *Linear, scalar problems*

4.1.1 *First-order* An initial value problem of the form

$$a(x)u'(x) + b(x)u(x) = f(x), \tag{4.1}$$

$$u(-1) = c \tag{4.2}$$

[2] In practice, higher-order differentiation matrices may be constructed from explicit formulae rather than by taking successive powers of D_N (Weideman & Reddy, 2000).

[3] One might choose other grids for $\mathbf{y} \in \mathbb{R}^{N-m}$, such as another second-kind grid, but in Section 6 we will point out a major advantage of first-kind points for the discretization of a partial differential equation (PDE).

can be discretized in the Chebyshev collocation method as

$$A\mathbf{u} := [\mathrm{diag}(\mathbf{a})D_{N+1} + \mathrm{diag}(\mathbf{b})]\mathbf{u} = \mathbf{f}, \tag{4.3}$$

$$B\mathbf{u} := [1 \quad 0 \quad 0 \quad \cdots \quad 0]\mathbf{u} = c. \tag{4.4}$$

We use boldface letters to indicate a vector obtained by discretizing at $N+1$ Chebyshev nodes; thus, $\mathbf{x} = [x_1, \ldots, x_{N+1}]^{\mathrm{T}}$, $\mathbf{a} = [a(x_1), \ldots, a(x_{N+1})]^{\mathrm{T}}$, etc. The reason for using the notation of $N+1$ rather than N points for u is to collocate the ODE at N points, which will become important in Section 4.4.

The standard row replacement or boundary bordering strategy is to solve the $(N+1) \times (N+1)$ system

$$\begin{bmatrix} B \\ R_{N+1,-1}A \end{bmatrix} \mathbf{u} = \begin{bmatrix} c \\ R_{N+1,-1}f(\mathbf{x}) \end{bmatrix}, \tag{4.5}$$

where, in this instance, one makes the particular choice of $R_{N+1,-1}$ as

$$R_{N+1,-1} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ & & & \ddots & \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}, \tag{4.6}$$

i.e., the $N \times (N+1)$ matrix that deletes the first row of any right multiplicand. It is arguably natural to choose this row, which corresponds to collocation of the ODE system at the left boundary, in order to replace it with a side condition based on the known function value at the same location. Typically, the trivial algebra of simply replacing $u(0)$ by c is done before machine computation, reducing the system to a square one of size N.

In our formulation, the approach is to solve equation (4.5) with resampling rather than row deletion; that is, to let $R_{N+1,-1} = P_{N+1,-1}$. Note that while the final linear algebraic system is again $(N+1) \times (N+1)$, the enforcement of the ODE (4.1) is at a different, smaller set of Chebyshev nodes than the set on which the solution $\mathbf{u}$ is defined. This perhaps stretches the meaning of the term 'collocation', but a representation of the ODE on a set of nodes native to the entire interval should be more appropriate than an approximation that deletes one node (or more) from a complete set.

A more concrete advantage of the rectangular approach emerges when we consider auxiliary conditions to the problem different from (4.2) in the row replacement strategy. If $u(1)$ is given rather than $u(-1)$, it is natural to let $R_{N+1,-1}$ be derived from the identity with its last row deleted, rather than (4.6). But matters are less clear with the nonseparable boundary condition

$$u(-1) + u(1) = 0, \tag{4.7}$$

or even a global condition, like

$$\int_{-1}^{1} u(x)\,\mathrm{d}x = 0. \tag{4.8}$$

These conditions are straightforward to discretize: one only has to replace the matrix B in (4.4) with

$$B = [1 \quad 0 \quad \cdots \quad 0 \quad 1]$$

for (4.7), and with a vector of Clenshaw–Curtis quadrature weights in the case of (4.8). However, it now becomes unclear which row in A should be replaced. This ambiguity is removed when resampling is used; no rows are replaced!

To understand what effect the row deletion choice may have, it is illustrative to consider the trivial example with $A = D_{N+1}$, i.e., the antiderivative problem $u' = f$. Dropping the subscript for the moment, it is well known that D obeys the antisymmetry property $D = -JDJ$, where J is a flipping matrix obtained by reversing the columns (or rows) of an identity. Thus, we can relate deletion of the first row to deletion of the last row via

$$\begin{aligned}\begin{bmatrix} e_2^{\mathrm{T}} \\ \vdots \\ e_{N+1}^{\mathrm{T}} \end{bmatrix} D = -\begin{bmatrix} e_2^{\mathrm{T}} \\ \vdots \\ e_{N+1}^{\mathrm{T}} \end{bmatrix} JD_N J = -\begin{bmatrix} e_N^{\mathrm{T}} \\ \vdots \\ e_1^{\mathrm{T}} \end{bmatrix} D_N J \\ = -\left(J \begin{bmatrix} e_2^{\mathrm{T}} \\ \vdots \\ e_{N+1}^{\mathrm{T}} \end{bmatrix} \right) DJ \\ = J \left(-\begin{bmatrix} e_N^{\mathrm{T}} \\ \vdots \\ e_1^{\mathrm{T}} \end{bmatrix} D \right) J.\end{aligned}$$

Hence, if we augment the problem with a boundary row such as in (4.5), the two types of discretization are related via a similarity transformation with the reflection matrix

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ 0 & & J & \\ 0 & & & \end{bmatrix},$$

and the eigenvalues of the two deletion cases are negatives of one another.[4]

In contrast, the rectangular projection strategy uses $R_{N+1,-1} = P_{N+1,-1}$. It is easy to see the symmetry relation $P_{N+1,-1} = JP_{N+1,-1}J$. Continuing the antiderivative example without subscripts on D and P, we find that $PD = (JPJ)(-JDJ) = -J(PD)J$, so that the eigenvalues of the matrix in (4.5) are symmetric with respect to the origin. Figure 3 compares the eigenvalues of row replacements with resampling for the antiderivative operator $A = D_{N+1}$ with side condition (4.7). (For details on how the side condition is incorporated into the eigenvalue computation, see Section 5.) Only in the resampling discretization do the eigenvalues have the same symmetry as those of the exact operator.

4.1.2 *Higher-order problems* Now, consider a linear problem of order m,

$$\begin{aligned} a_m(x)u^{(m)}(x) + \cdots + a_1(x)u'(x) + a_0(x)u(x) = f(x), \\ \ell_i(u, \ldots, u^{(m-1)}) = c_i, \quad i = 1, \ldots, m, \end{aligned} \tag{4.9}$$

[4] This relationship may be immaterial for the boundary-value problem, but it can be decisive in the stability of a method of lines discretization of a PDE.

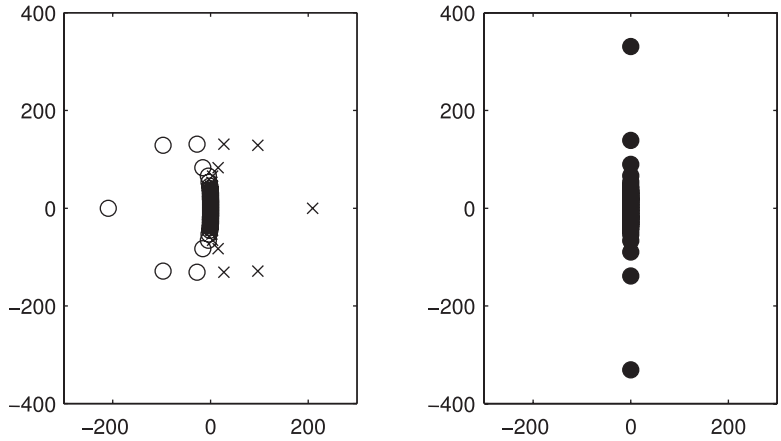

FIG. 3. Eigenvalues of discretizations on $N = 40$ points of the operator u' subject to $u(1) + u(-1) = 0$. On the left, the results using deletion of the first row (crosses) or last row (circles) to make room for the side condition. On the right, the results using resampling. Only in the resampling case are the eigenvalues purely imaginary, as for the exact operator.

where each ℓ_i is a linear functional. Again, for reasons important in Section 4.4, we will discretize the solution u at $N + m$ points.

In the commonplace case with $m = 2$ and one condition at each boundary, the obvious strategy for row replacement is to replace the first and last row of the collocation matrix. For $m > 2$, though, while it is straightforward to delete some m rows of an identity to get an analogue of (4.6), there is little clarity in whether deleting multiple rows from either the top or bottom is a sound idea. In fact, one finds various alternative strategies in the literature for particular problems (see, for example, Trefethen (2000, Chapter 14)).

The rectangular differentiation method, in contrast, proceeds unambiguously as before. The linear equations to be solved can be expressed as

$$\begin{bmatrix} B \\ P_{N+m,-m}A \end{bmatrix} \mathbf{u} = \begin{bmatrix} c \\ P_{N+m,-m}f(\mathbf{x}) \end{bmatrix},$$

where A is the standard (square) collocation representation of the operator in equation (4.9), the ith row of $B \in \mathbb{C}^{m,N+m}$ is an $(N + m)$-point discretization of the functional ℓ_i, and c is an m-vector of the c_is.

4.2 *Nonlinear problems*

Nonlinearity in a scalar boundary-value problem introduces no significant complications for either row replacement or rectangular formulations. Suppose

$$h(u) = f(x), \quad -1 < x < 1, \tag{4.10}$$

$$g_i(u) = c_i, \quad i = 1, \ldots, m, \tag{4.11}$$

where h is a nonlinear operator of differential order m, and each g_i is a nonlinear functional of differential order up to $m - 1$. We replace $u(x)$ by its $(N + m)$-vector discretization $\mathbf{u}$, and each derivative $u^{(k)}(x)$ by $D^k_{N+m}\mathbf{u}$, in order to create an N-vector $\mathbf{h}(\mathbf{u})$.

The natural formulation of (4.10) is the nonlinear algebraic system

$$R_{N+m,-m}(\mathbf{h}(\mathbf{u}) - \mathbf{f}) = 0,$$

where $R_{N+m,-m}$ is either a truncated identity for row replacement, as in (4.6), or a resampling matrix $P_{N+m,-m}$. Perturbing the unknown to $\mathbf{u} + \boldsymbol{\delta}$ and linearizing, as is the basis of Newton-like solvers, one obtains

$$R_{N+m,-m}\mathbf{h}'(\mathbf{u})\boldsymbol{\delta} = R_{N+m,-m}[\mathbf{f} - \mathbf{h}(\mathbf{u})], \tag{4.12}$$

where $\mathbf{h}'$ is a square $N + m$ Jacobian matrix. For example, in $u'' + uu' = 1$, we have $h(u) = u'' + uu'$ and $\mathbf{h}(\mathbf{u}) = D^2_{N+m}\mathbf{u} + \mathbf{u} \circ (D_{N+m}\mathbf{u})$, where $\circ$ is the Hadamard (elementwise) product. Then, it is straightforward to show that

$$\mathbf{h}'(\mathbf{u}) = D^2_{N+m} + \mathrm{diag}(D_{N+m}\mathbf{u}) + \mathrm{diag}(\mathbf{u})D_{N+m}, \tag{4.13}$$

where diag() creates a diagonal matrix using the entries of the given vector along the main diagonal.

Equation (4.12) is identical to the rectangular discretization of the linear ODE

$$h'(u)\delta(x) = f(x) - h(u),$$

where h' is the Fréchet derivative of h. Continuing the example above, we find $h'(u)\delta(x) = \delta''(x) + u'(x)\delta(x)u(x)\delta'(x)$, whose $N + m$ point discretization for δ again leads to the matrix of (4.13). Thus, the transformations of linearization and discretization are commutative at fixed $N + m$.

The boundary conditions (4.11) are handled just as in the linear case. They are discretized on $N + m$ points and do not need to be resampled (since they result in scalars, not functions). Each boundary condition can be linearized, if necessary, in order to yield additional rows to be adjoined to (4.12), leading to a square algebraic system. Linear boundary conditions are imposed correctly on the solution after a single linearization step, whereas nonlinear conditions are satisfied only in the iterative limit of the Newton solver.

4.3 *Piecewise smooth problems*

While spectral methods converge geometrically fast when the underlying solution to an ODE is analytic in some neighbourhood of its domain, any discontinuities in the solution or one of its derivatives reduce the rate to algebraic. However, if the singularities are isolated and their location is known, one can patch together piecewise Chebyshev interpolants to maintain geometric convergence (Driscoll & Fornberg, 1998). Furthermore, it can sometimes be advantageous to use such a piecewise discretization (i.e., domain decomposition) even when the solution is smooth (Driscoll & Weideman, 2014).

Consider, as an example,

$$a(x)u'(x) + b(x)u(x) = |x|,$$
$$u(-1) = c.$$

With some fairly weak assumptions on a and b, it is evident that u will in general have a discontinuity in its second derivative at $x = 0$, and therefore a global Chebyshev interpolant of u will converge only algebraically. Instead, let us discretize the solution on two scaled and shifted $(N + 1)$-point Chebyshev grids: $\mathbf{x_L}$ on $[-1, 0]$ and $\mathbf{x_R}$ on $[0, 1]$. Using the appropriately scaled differentiation matrices

$D_L = D_R = 2D_{N+1}$, we can discretize the ODE by

$$A_L\mathbf{u_L} = [\mathrm{diag}(a(\mathbf{x_L}))D_L + \mathrm{diag}(b(\mathbf{x_L}))]\mathbf{u_L} = f(\mathbf{x_L}) = \mathbf{f_L},$$
$$A_R\mathbf{u_R} = [\mathrm{diag}(a(\mathbf{x_R}))D_R + \mathrm{diag}(b(\mathbf{x_R}))]\mathbf{u_R} = f(\mathbf{x_R}) = \mathbf{f_R},$$

and so arrive at the block system

$$\begin{bmatrix} A_L & \\ & A_R \end{bmatrix} \begin{bmatrix} \mathbf{u_L} \\ \mathbf{u_R} \end{bmatrix} = \begin{bmatrix} \mathbf{f_L} \\ \mathbf{f_R} \end{bmatrix}.$$

Since both A_L and A_R are first-order operators, we reduce the row size of each by one and add the boundary condition at $x = -1$ as in equation (4.4) to find the $(2N+1) \times (2N+2)$ system

$$\begin{bmatrix} B_L & & \\ R_{N+1,-1}A_L & & \\ & & R_{N+1,-1}A_R \end{bmatrix} \begin{bmatrix} \mathbf{u_L} \\ \mathbf{u_R} \end{bmatrix} = \begin{bmatrix} c \\ R_{N+1,-1}\mathbf{f_L} \\ R_{N+1,-1}\mathbf{f_R} \end{bmatrix}.$$

As before, the choice of row replacement versus resampling boils down to the definition of $R_{N+1,-1}$. The final constraint required to square up the system comes from enforcing continuity of u at $x = 0$. Since both $\mathbf{x_L}$ and $\mathbf{x_R}$ contain 0, this can be achieved by adding the *continuity conditions* row

$$C = [e_{N+1}^{\mathrm{T}} \quad -e_1^{\mathrm{T}}] = [0 \quad \cdots \quad 0 \quad 1 \quad -1 \quad 0 \quad \cdots \quad 0],$$

so that

$$\begin{bmatrix} B_L & & \\ e_{N+1}^{\mathrm{T}} & & -e_1^{\mathrm{T}} \\ R_{N+1,-1}A_L & & \\ & & R_{N+1,-1}A_R \end{bmatrix} \begin{bmatrix} \mathbf{u_L} \\ \mathbf{u_R} \end{bmatrix} = \begin{bmatrix} c \\ 0 \\ R_{N+1,-1}\mathbf{f_L} \\ R_{N+1,-1}\mathbf{f_R} \end{bmatrix}.$$

Note that the differential equation implicitly enforces continuity of $u'(x)$.

More generally, if A were an mth-order ODE, once the m given boundary conditions have been applied to the projected matrix, the system is still $(2N+m) \times (2N+2m)$. Therefore, to square it up, we enforce continuity of the zeroth to $(m-1)$th derivatives of the solution, in addition to the m boundary conditions. For example, if $m = 3$, then

$$C = \begin{bmatrix} e_{N+1}^{\mathrm{T}} & -e_1^{\mathrm{T}} \\ e_{N+1}^{\mathrm{T}}D_L & -e_1^{\mathrm{T}}D_R \\ e_{N+1}^{\mathrm{T}}D_L^2 & -e_1^{\mathrm{T}}D_R^2 \end{bmatrix}.$$

Furthermore, this approach can be used to enforce desired jumps in the derivatives, rather than continuity, by specifying a nonzero in the appropriate row of the right-hand side vector.

4.4 *Systems of equations*

As with scalar equations, it suffices to restrict our attention to linear systems of equations, as nonlinear systems will be solved by successive linear approximations. Consider first the first-order system

$$A_1(x)u'(x) + A_0(x)u(x) = f(x), \quad x \in [-1, 1],$$
$$Lu = c,$$

where $u(x), f(x), c \in \mathbb{C}^d$, $A_k(x) \in \mathbb{C}^{d\times d}$ and L is a functional mapping $u(x)$ into $\mathbb{C}^d$. We assume the rows of L to be linearly independent and that the system is without singular points (i.e., that $A_1(x)$ is nonsingular at every $x \in [-1, 1]$). As in the scalar case, each component function of $u(x)$ is collocated at $N + 1$ points. With the understanding that, for any vector-valued function $w(x)$,

$$\mathbf{w} = \begin{bmatrix} w(x_1) \\ w(x_2) \\ \vdots \\ w(x_{N+1}) \end{bmatrix} \in \mathbb{C}^{(N+1)d},$$

and

$$\mathsf{A}_k = \begin{bmatrix} \operatorname{diag}(\mathbf{a}_{11}^{(k)}) & \cdots & \operatorname{diag}(\mathbf{a}_{1d}^{(k)}) \\ \vdots & & \vdots \\ \operatorname{diag}(\mathbf{a}_{d1}^{(k)}) & \cdots & \operatorname{diag}(\mathbf{a}_{dd}^{(k)}) \end{bmatrix} \in \mathbb{C}^{(N+1)d\times(N+1)d},$$

for $k = 0, 1$, the discretization of the ODE is compactly written as

$$\mathsf{A}_1(I_d \otimes D_{N+1})\mathbf{u} + \mathsf{A}_0\mathbf{u} = \mathbf{f}. \tag{4.14}$$

Here, I_d is the $d \times d$ identity matrix, and $\otimes$ is the Kronecker product.

In order to reduce dimensions so that d boundary conditions may be imposed, we reduce the dimension of each block row of the system by one; that is, we left-multiply (4.14) by $(I_d \otimes R_{N+1,-1})$. The discrete form of the auxiliary condition $Lu = c$ is then prepended to make a square $(N + 1)d \times (N + 1)d$ linear system to solve for $\mathbf{u}$.

Now, consider the mth-order system $Au = A_m(x)u^{(m)}(x) + \cdots + A_0(x)u(x) = f(x)$, with $A_m(x)$ nonsingular at every point. One could use the standard transformation of introducing extra variables to represent the derivatives $u_j', \ldots, u_j^{(m_j)}, j = 1, \ldots, d$, and rewrite the problem as a first-order system in md variables. However, to avoid an unnecessarily large system of equations, we prefer a direct discretization of the original high-order equation.

For convenience, denote by m_j the maximum derivative of u_j (the jth component of u) appearing in A; that is, at least one element in the jth column of $A_{m_j}(x)$ is nontrivial. A well-posed problem will specify $m = \sum_{j=1}^{d} m_j$ independent boundary/side conditions. If we apply the principle that, in each row of the system, the output dimension from each column should be the same, then we arrive at the following approach:

$$\begin{bmatrix} P_{N+m_1,-m_1}\mathsf{A}_{11} & \cdots & P_{N+m_d,-m_d}\mathsf{A}_{1d} \\ \vdots & & \vdots \\ P_{N+m_1,-m_1}\mathsf{A}_{d1} & \cdots & P_{N+m_d,-m_d}\mathsf{A}_{dd} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_d \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_d \end{bmatrix},$$

where, for each $j = 1, \ldots, d$, variable u_j and all the $\mathsf{A}_{i,j}$ are discretized at $(N + m_j)$ second-kind points, while f_j is discretized at N Chebyshev points of the first-kind. The method can be viewed as discretizing each column of A using $N + m_j$ points in the usual square fashion before projecting down to size N. The resulting matrix in (4.14) is of size $N \times (N + m)$ and can be squared up with the m supplemental conditions before solving for u.

5. Rectangular collocation for eigenvalue ODE problems

Resampling can also be used in ODE eigenvalue problems having the generalized form

$$\begin{aligned} &a_m(x)u^{(m)}(x) + \cdots + a_1(x)u'(x) + a_0(x)u(x) \\ &\quad = \lambda[b_m(x)u^{(m)}(x) + \cdots + b_1(x)u'(x) + b_0(x)u(x)], \qquad (5.1) \\ &\ell_i(u, \ldots, u^{(m-1)}) = 0, \quad i = 1, \ldots, m, \qquad (5.2) \end{aligned}$$

where each ℓ_i is a linear functional. For simplicity, we assume that the coefficient functions are smooth. Extensions to the system and piecewise smooth cases follow from the same considerations as for boundary-value problems.

We again use standard collocation to define the matrices

$$\begin{aligned} A &= \operatorname{diag}(\mathbf{a}_m)D_{N+m}^m + \cdots + \operatorname{diag}(\mathbf{a}_1)D_{N+m} + \operatorname{diag}(\mathbf{a}_0), \\ B &= \operatorname{diag}(\mathbf{b}_m)D_{N+m}^m + \cdots + \operatorname{diag}(\mathbf{b}_1)D_{N+m} + \operatorname{diag}(\mathbf{b}_0). \end{aligned} \qquad (5.3)$$

We also use standard $(N+m)$-point discretization to define an $m \times (N+m)$ matrix L whose ith row is defined by

$$e_i^{\mathrm{T}} L\mathbf{u} = \ell_i(\mathbf{u}, D_{N+m}\mathbf{u}, \ldots, D_{N+m}^m\mathbf{u}) \qquad (5.4)$$

for all $\mathbf{u} \in \mathbb{C}^{N+m}$.

The row replacement strategy, as with boundary-value problems, is clearest in the case of a second-order, two-point problem. We will apply Schur complementation in order to eliminate degrees of freedom using the auxiliary conditions. For simplicity, suppose we rearrange and partition $\mathbf{u}$, L, A and B so that

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_b \\ \mathbf{u}_i \end{bmatrix}, \quad \mathbf{u}_b = \begin{bmatrix} u_1 \\ u_{N+2} \end{bmatrix},$$

$$[L_i \quad L_b]u = 0, \quad A = \begin{bmatrix} A_{bb} & A_{bi} \\ A_{ib} & A_{ii} \end{bmatrix}, \quad B = \begin{bmatrix} B_{bb} & B_{bi} \\ B_{ib} & B_{ii} \end{bmatrix}.$$

We apply $L\mathbf{u} = 0$ in order to solve for $\mathbf{u}_b$:

$$\mathbf{u}_b = -L_b^{-1}L_i\mathbf{u}_i.$$

Now, disregarding the b-rows of the system, we arrive at the $N \times N$ generalized eigensystem

$$[A_{ib} \quad A_{ii}] \begin{bmatrix} -L_b^{-1}L_i \\ I_N \end{bmatrix} \mathbf{u}_i = \lambda [B_{ib} \quad B_{ii}] \begin{bmatrix} -L_b^{-1}L_i \\ I_N \end{bmatrix} \mathbf{u}_i. \qquad (5.5)$$

The system (5.5) has a notable simplification in the important case where $B = I_{N+2}$; i.e., when the right-hand side of (5.1) is just $\lambda u(x)$ and the eigenvalue problem is standard, not generalized. In this case, $B_{ib} = 0$ and $B_{ii} = I_N$, so the right-hand side of (5.5) is simply a truncated identity. That is, the discrete eigenvalue problem is a generalized (rather than standard) one only when the original ODE eigenvalue problem is also a generalized one.

The resampling rectangular discretization of (5.1)–(5.2), on the other hand, is simply

$$\begin{bmatrix} L \\ P_{N+m,-m}A \end{bmatrix} \mathbf{u} = \lambda \begin{bmatrix} 0 \\ P_{N+m,-m}B \end{bmatrix} \mathbf{u}.$$

This result takes the form of an $(N+m)\times(N+m)$ generalized eigenvalue problem, even when the original problem is not generalized. The resampling formulation therefore requires the QZ algorithm (Golub & Van Loan, 2012) rather than the QR algorithm in all cases.

As we saw in Fig. 3, the row replacement method does not always produce eigenvalues that have the same qualitative properties, such as symmetries, as the original operator. In addition, the Schur complementation technique is less clear, nonunique, and of unknown numerical stability in problems with order greater than two or atypical auxiliary conditions. We feel that the simplicity and unity of the resampling strategy is worth the extra computational effort in the context of a general-purpose algorithm.

6. Rectangular collocation for time-dependent PDEs

6.1 *Linear, constant coefficient PDE*

We begin with the important special case of a linear, constant coefficient problem:

$$\begin{gathered} \frac{\partial u}{\partial t} = a_m u^{(m)}(x,t) + \cdots + a_1 u'(x,t) + a_0 u(x,t), \quad t > 0, \\ \ell_i(u,\ldots,u^{(m-1)}) = 0, \quad i = 1,\ldots,m, \\ u(0) \text{ given}, \end{gathered}$$

where we continue to use superscript notation for derivatives with respect to x, and the ℓ_i are constant linear functionals. Extensions to piecewise solutions and to systems are as described in Section 4, but the extension to nonlinearity is given differently below.

As with (5.3)–(5.4), we use powers of the spectral differentiation matrix to define the discrete linear equations

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = A\mathbf{u}, \tag{6.1}$$

$$L\mathbf{u} = 0, \tag{6.2}$$

where $A \in \mathbb{C}^{(N+m)\times(N+m)}$ and $L \in \mathbb{C}^{m\times(N+m)}$. Because it is a linear, constant coefficient ODE, we want to express the solution to (6.1) using a matrix exponential. However, the constraints (6.2) are a complicating factor. Suppose we define $\tilde{\mathbf{u}} = R_{N,-m}\mathbf{u}$ as reduced variables resulting from either row removal or resampling. We can recover $\mathbf{u}$ from $\tilde{\mathbf{u}}$ by joining this definition with the side constraints and solving:

$$\begin{bmatrix} L \\ R_{N+m,-m} \end{bmatrix} \mathbf{u} = \begin{bmatrix} 0 \\ I_N \end{bmatrix} \tilde{\mathbf{u}}$$

$$\Rightarrow \quad \mathbf{u} = E\tilde{\mathbf{u}}, \quad E = \begin{bmatrix} L \\ R_{N+m,-m} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ I_N \end{bmatrix}. \tag{6.3}$$

Multiplying both sides of (6.1) on the left by $R_{N,-m}$, we get the $(N-m)\times(N-m)$ linear system

$$\frac{\mathrm{d}\tilde{\mathbf{u}}}{\mathrm{d}t} = R_{N+m,-m}A\mathbf{u} = R_{N+m,-m}AE\tilde{\mathbf{u}} \;\Rightarrow\; \tilde{\mathbf{u}}(t) = \exp(tR_{N+m,-m}AE)\tilde{\mathbf{u}}(0).$$

Combining this with (6.3) and the definition of $\tilde{\mathbf{u}}$, we arrive at

$$\mathbf{u}(t) = E\exp(tR_{N+m,-m}AE)R_{N+m,-m}\mathbf{u}(0).$$

As we saw with eigenvalues, row removal can lead to a modest algebraic speedup compared with resampling. When $R_{N+m,-m}$ is a truncated identity, then all but the top m rows of (6.3) are trivial, and $\mathbf{u}$ can be recovered from $\tilde{\mathbf{u}}$ by solving an $m\times m$ system through complementation. However, we recall from Fig. 3 that the choice of deleted rows can have effects on the eigenvalues that completely change the qualitative character of the solution. In addition, the cost of computing E (or its action on a vector) is typically far less than computing the matrix exponential.

6.2 *General PDEs and DAEs*

For more general PDEs,

$$\begin{aligned} \frac{\partial u}{\partial t} &= F(x,t,u), \\ g(t,u) &= 0, \end{aligned} \tag{6.4}$$

where F is the mth-order differential in x and $g\in\mathbb{C}^m$, a method of lines approach is useful. If as usual we denote an $(N+m)$-point discretization using boldface letters, the simplest expression of (6.4) is

$$\begin{bmatrix} 0 \\ R_{N+m,-m} \end{bmatrix} \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \begin{bmatrix} g(t,\mathbf{u}) \\ R_{N+m,-m}\mathbf{F}(\mathbf{x},t,\mathbf{u}) \end{bmatrix}. \tag{6.5}$$

Equation (6.5) is a differential–algebraic equation (DAE). As such, it requires a stiff integrator. Traditionally, in row replacement where $R_{N+m,-m}$ is a truncated identity, one can again appeal to complementation and perform algebra to make (6.5) an ODE which might be treated by an explicit integrator. For semiexplicit problems, this algebra is essentially performed by a DAE solver automatically. Given that Chebyshev differential operators of order m on N points have eigenvalues of size $O(N^{2m})$ (Weideman & Trefethen, 1988), we do not regard the requirement of a stiff solver as a severe imposition on the great majority of problems. In exchange, we get a simple and robust formulation for a very wide variety of PDEs and auxiliary conditions.

We do point out one important caution when using resampling via $R_{N,-m} = P_{N,-m}$: the points used to define the range of $P_{N,-m}$ must not include points at which conditions are imposed via g—typically, boundary points. Doing so creates two potentially different conditions to satisfy at a single point, and since surprisingly many initial-boundary-value problems are supplied with initial conditions that are incompatible with the boundaries (Flyer & Fornberg, 2003), they will raise the DAE index and usually cause the solver to fail. This observation is a major part of our motivation for using first-kind Chebyshev points throughout the collocation process, as they avoid boundaries altogether.

In Section 7.5, we demonstrate this approach for the KdV-like Rosenau equation, which has a mixed time–space derivative term, u_{xxxxt}.

7. Examples

In this section, we demonstrate how rectangular spectral collocation can be used in solving various problems of the types discussed in Sections 4–6.

Chebfun is an open-source software system for numerical computing with functions (Driscoll *et al.*, 2014). Its mathematical basis is piecewise polynomial interpolation with Chebyshev polynomials, and in particular the software has extensive capabilities for dealing with linear and nonlinear differential and integral operators based upon Chebyshev spectral methods. The use of rectangular collocation as a means to enforce boundary conditions in BVPs, ODE eigenvalue problems, and PDEs has been present in the system since the release of version 4.2 in March 2012 (Driscoll *et al.*, 2014), and a large suite of tests and examples has consistently been used to validate the correctness of the approach.

Below, we give some 'manual' implementations, in which we construct the required matrices step by step, and also examples of Chebfun syntax that automates such computations. In the automated computations, Chebfun uses automatic differentiation to linearize nonlinear operators and apply a damped Newton iteration, and adapts the discretization size in linear problems to attempt to fully resolve the solution to high accuracy. Rather than discuss these features in detail here, we refer the reader to Birkisson & Driscoll (2012) and Driscoll *et al.* (2008), respectively. All the examples in this section use Chebfun version 5.1.0.

7.1 *Piecewise smooth BVP*

As our first example, we consider a second-order ODE with a discontinuous coefficient in the reaction term,

$$\begin{aligned} 0.001u'' + \operatorname{sgn}(x)u &= 1, \quad x \in [-1,1], \\ u(-1) &= 0, \\ \int_{-1}^{1} u(x)\,\mathrm{d}x &= 0. \end{aligned} \tag{7.1}$$

To solve manually, we proceed as described in Section 4.3 and discretize with smooth interpolants on $[-1,0]$ and $[0,1]$ coupled together by continuity conditions at $x=0$. The first step is to define differentiation matrices and relevant grids for the solution and the ODE.

```
e = 0.001; N = 42;                          % Choose eps and N.
% Square, scaled differentiation matrix for 2nd-kind points:
D = diffmat(N+2); D = 2*D;
D2 = D^2;                                   % 2nd-order derivative.
% Nodes for solution, plus quadrature and barycentric weights:
[xFunc, CCW, BaryW] = chebpts(N+2, [0 1], 2);
xDE = chebpts(N, [0, 1], 1);                % 1st-kind grid for ODE.
```

The resampling matrix is computed in a single line:

```
P = barymat(xDE, xFunc, BaryW);             % Resampling matrix.
```

Next, we create the system matrix and resample it down to $2N$ total points.

```
I = eye(N+2);                               % Identity operator.
A = blkdiag(e*D2 - I, e*D2 + I);            % Square operator.
PA = blkdiag(P, P) * A;                     % Rectangular operator.
```

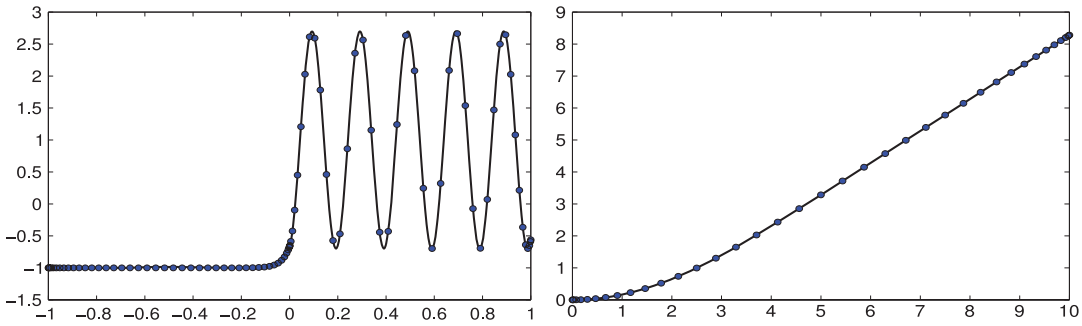

FIG. 4. Solutions of reaction–diffusion equation (7.1) (left) and Blasius equation (7.3) (right).

Then, we discretize the boundary conditions at the function's nodes:

```
z = zeros(1, N+2);
B = [ 1 z(2:N+2),  z   ;              % Dirichlet constraint.
      CCW       ,  CCW ];             % Integral constraint.
```

The Clenshaw–Curtis quadrature weights in `CCW` were computed above with the grid `xFunc`. To complete the operator, we append C^1 continuity conditions at the interface.

```
B = [ B                          ;
      z(1:N+1) 1, -1 z(2:N+2) ;       % Continuity of u at x = 0.
      D(N+2,:)  , -D(1,:)     ];      % Continuity of u' at x = 0.
M = [ B ; PA ];                       % Complete system matrix.
```

The right-hand side of the linear system starts out with a 1 for the Dirichlet condition row, followed by three zeros, and then the discretization of the ODE forcing term (here, identically 1) at the ODE grid.

```
rhs = [ -1 ; 0 ; 0 ; 0 ;          % BCs & continuity constraints.
        ones(2*N, 1) ];           % RHS of ODE.
u = M \ rhs;                      % Solve on the function grid.
xSol = chebpts([N+2, N+2], [-1, 0, 1]);
plot(xSol, u, '.b');              % Plot the solution.
```

The solution is shown in Fig. 4 (left). In Chebfun syntax, the process can be applied fully automatically as follows:

```
A = chebop( @(x,u) e*diff(u,2) + sign(x).*u, [-1, 1] );
A.bc = @(x,u) [ u(-1) + 1 ; sum(u) ]; % Append boundary conditions.
u = A \ 1;                            % Solve.
hold on, plot(u, 'k'), hold off       % Plot the solution.
```

7.2 *Nonlinear system*

As our next example, we consider the Blasius equation on a truncated domain:

$$\begin{aligned} f'''(x) + \tfrac{1}{2} f(x) f''(x) = 0, \quad x \in [0, 10], \\ f(0) = f'(0) = 0, \quad f'(10) = 1. \end{aligned} \tag{7.2}$$

Although it is not necessary to do so, we rewrite (7.2) as a system of two equations in order to demonstrate the method of Section 4.4:

$$\begin{aligned} u'' - v = 0, \\ v' + \tfrac{1}{2} uv = 0, \\ u(0) = u'(0) = 0, \quad u'(10) = 1. \end{aligned} \tag{7.3}$$

After linearization, we get successive systems for corrections $\tilde{u}$ and $\tilde{v}$ to current estimates u_c and v_c, respectively, in the form

$$\begin{aligned} \tilde{u}'' - \tilde{v} = r_1, \\ \tilde{v}' + \tfrac{1}{2}(u_c \tilde{v} + \tilde{u} v_c) = r_2, \\ \tilde{u}(0) = \tilde{u}'(0) = \tilde{u}'(10) = 0. \end{aligned}$$

Here, r_1 and r_2 represent residuals in the ODEs when inserting the current approximations u_c and v_c. We have assumed that u_c satisfies the boundary conditions (in any event, one Newton step will correct them perfectly due to their linearity).

Since $\tilde{u}$ now appears with a second derivative term, and $\tilde{v}$ only with a first, we discretize them on $N+2$ and $N+1$ points, respectively. The rectangular block operator before appending boundary conditions is

$$\begin{bmatrix} P_{N+2,-2}\, D^2_{N+2} & -P_{N+1,-1} \\ \tfrac{1}{2} P_{N+2,-2} V & P_{N+1,-1}(D_{N+1} + \tfrac{1}{2} U) \end{bmatrix}, \tag{7.4}$$

where

$$V = P_{N+1,1}\, \mathrm{diag}(\mathbf{v}_c), \quad U = P_{N+2,-1}\, \mathrm{diag}(\mathbf{u}_c). \tag{7.5}$$

The resampling in (7.5) is needed to translate between the different grids used to represent the two functions. The matrix in (7.4) has dimension $2N \times (2N+3)$; hence, appending three boundary rows will produce a square matrix.

In Chebfun, the entire process (including adaptive search for N and linearization) is applied through the following commands.

```
% Define the nonlinear operator on the interval [0 10]:
N = chebop( @(x,u,v) [ diff(u,2) - v ; diff(v) + .5*u.*v ], [0, 10] );
N.lbc = @(u,v) [ u ; diff(u) ]; % Add left boundary conditions.
N.rbc = @(u,v) diff(u) - 1;     % Add right boundary condition.
uv = N \ 0;                     % Solve with backslash.
plot(uv{1}, '.-k')              % Plot the solution.
```

The result is shown in Fig. 4 (right) and is accurate to around 10 digits.

7.3 *ODE eigenvalues*

The harmonic, rotationally symmetric vertical displacement of a perfectly elastic circular drum whose density is $\rho(r)$ is described by the Bessel equation

$$u''(r) + r^{-1}u'(r) = -\omega^2\rho(r)u(r), \quad u'(0) = 0, \;\; u(1) = 0,$$

where ω is the temporal frequency of the vibration. The problem is to be solved for $0 < r < 1$, which makes the $1/r$ term problematic, so we multiply through by r:

$$ru''(r) + u'(r) = -\omega^2 r\rho(r)u(r), \quad u'(0) = 0, \;\; u(1) = 0. \tag{7.6}$$

Equation (7.6) is in the form of a generalized eigenproblem $Au = \lambda Bu$. For the case of a uniform material with $\rho \equiv 1$, its expression in Chebfun is

```
A = chebop( @(r,u) r.*diff(u,2) + diff(u), [0, 1], 'neumann', 0 );
B = chebop( @(r,u) -r.*u, [0, 1] );
```

The relevant command is `eigs`, which, as for large matrices, finds a requested number of eigenvalues closest to a given value (or those most easily resolved, if no reference point is given).

```
[V, D] = eigs(A, B, 6, 0); % Find 6 eigenvalues closest to zero.
omega = sqrt(diag(D)).
```

The values of ω should be roots of the Bessel function J_0:

```
>> besselj(0, omega)
ans =
  -1.4041e-10
   7.6527e-11
  -8.7578e-12
   9.1255e-13
  -4.3800e-14
  -1.0307e-13
```

These results are well in line with the default error tolerance of 10^{-10} for ODE problems in Chebfun. We can visualize the first four eigenfunctions on the disc using Chebfun2 (Townsend & Trefethen, 2013), as shown in Fig. 5.

```
x = chebfun2( @(r,theta) r.*cos(theta), [0, 1, -pi, pi] );
y = chebfun2( @(r,theta) r.*sin(theta), [0, 1, -pi, pi] );
for k = 1:4
    v = chebfun2( @(r,theta) V{k}(r), [0, 1, -pi, pi] );
    v = v/sign(v(0,0)); % Normalize.
    subplot(1, 4, k), surf(x, y, v), axis equal square off
end
```

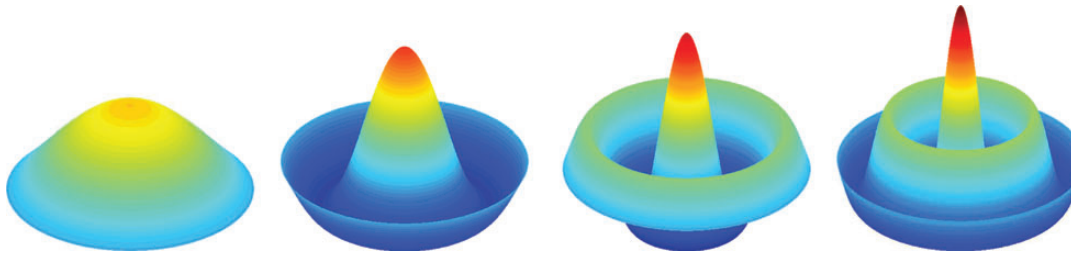

FIG. 5. First four axisymmetric vibration modes for a circular drum. See the text in Section 7.3 for related code.

7.4 *Operator exponential*

Consider the Black–Scholes evolution equation for a European call option,

$$\begin{aligned} v_t(s,t) &= \frac{\sigma^2}{2} s^2 v_{ss}(s,t) + rsv(s,t)_s - rv, \quad s > 0, \quad t > 0, \\ v(0,t) &= 0, \quad v(t,s) \to s \text{ as } s \to \infty, \end{aligned} \tag{7.7}$$

where v is the value of the option as a function of the price s of the underlying security. Suppose the option has a strike price of 50 at the initial time (in our time-reversed formulation), so that $v(s,0) = v_0 = \max(s-50, 0)$. We will truncate the domain at $s = \Sigma$ and let $v'(t, \Sigma) = 1$ be the right boundary condition in place of the asymptotic condition.

```
d = [0, 500];          % Choose a large interval to approximate infinity.
s = chebfun('s', d);   % Linear chebfun of stock price.
v0 = max(0, s - 50);   % Value at t = 0.
```

The system (7.7) has the form $v_t = Av$, $Bv = q$.

```
sigma = .45; r = 0.03; % std. dev. and risk-free interest rate
A = chebop(@(s,v) sigma^2/2*s.^2.*diff(v,2) + r*S.*diff(v) - r*v, d);
A.lbc = 0; A.rbc = @(v) diff(v) - 1; % Boundary conditions.
```

The boundary conditions are not homogeneous, whereas the method of Section 6.1 requires homogeneous conditions. However, there is a general method to avoid this limitation. Suppose $u(s)$ satisfies $Au = 0$, $Bu = q$. Then, $(v-u)_t = v_t = Av = A(v-u)$, and $B(u-v) = 0$. Hence, one can propagate $w = v - u$ with homogeneous conditions in combination with the solution of a boundary-value problem for u.

```
u = A\0;      % u satisfies u_t = A*u = 0, B*u = q.
w0 = v0 - u; % Adjusted variable - satisfies w_t = A*w, B*w = 0.
A.rbc = 0;    % Homogenous BCs.
```

We can now evaluate at any time directly (not by marching) using the operator exponential implemented by `expm`. The results are shown in Fig. 6.

```
plot(v0, [40, 60], 'LineWidth', 2), hold on
% Advance in time to t = 0.1, 0.2, ..., 0.5 using EXPM():
for t = 0.1:0.1:0.5
    w = expm(A, t, w0);
```

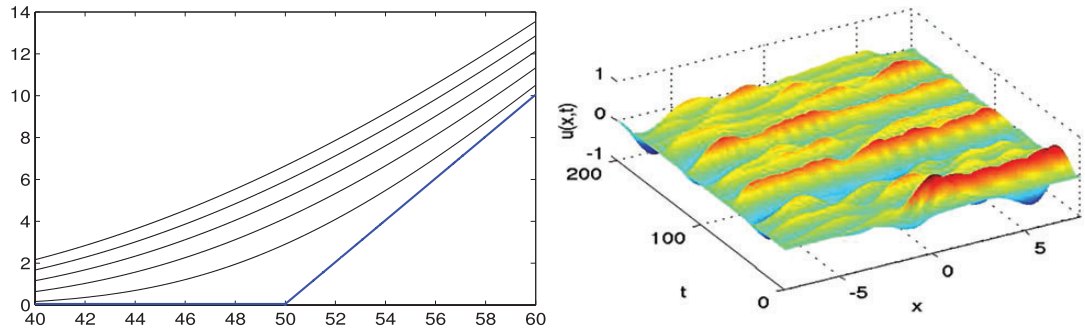

FIG. 6. Solutions of Black–Scholes equation (7.7) (left) and Rosenau equation (7.8) (right).

```
    v = w + u;
    plot(v, [40, 60], 'k')
end
```

The reported value of the option at a stock price of 55, for instance, is found to be `w(55) + u(55)`, which is 9.8499. Because the initial condition for w is piecewise smooth, the solution found by Chebfun at all times will also be piecewise smooth; at $t = 0.5$ in our example, it is represented by polynomials of degrees 28 and 43 to the left and right of $s = 50$, respectively. However, computationally the solution is very smooth, with a numerical jump of $< 10^{-8}$ in the second derivative at the interface.

7.5 *Nonlinear PDE*

As our final example, we consider the KdV-like 1D Rosenau equation

$$\begin{aligned} u_t + u_{xxxxt} &= -(u+u^2)_x, \quad x \in [-8,8], \quad t > 0, \\ u(\pm 1, t) &= u_{xx}(\pm 1, t) = 0, \quad u(x,0) = \operatorname{sech}(x). \end{aligned} \tag{7.8}$$

We pose this problem as a differential–algebraic equation, $Mu_t = f(t,u)$, with 'mass operator' $M = (I + (\partial x)^4)$. We begin by setting up a discretization of this operator to a mass matrix.

```
d = [-8, 8]; N = 80;                              % Domain and N.
M = chebop( @(u) u + diff(u, 4)/2, d );           % Mass operator.
massmat = [ matrix(M, N) ;  zeros(4, N+4) ]; % Mass matrix.
```

The last four rows are reserved for the boundary conditions, $0 = 0 \cdot u_t = Bu$. The Chebfun command `matrix` detects that M is a fourth-order operator and returns an $N \times (N+4)$ matrix accordingly.

Next we need to provide a discretization of the nonlinear term $f(t,u)$. This is less straightforward, because it has only a first-order derivative and `matrix` would produce it at size $N \times (N+1)$. We will cheat a bit by adding a tiny fourth-order term to fool `matrix` into doing the right thing for our context.

```
% N x (N+4) rectangular differentiation matrix:
D = matrix( chebop( @(u) diff(u) + eps*diff(u, 4), d ), N );
```

Now, we form the $4 \times (N + 4)$ boundary condition matrix B.

```
I = eye(N+4);                                   % Identity.
D2 = diffmat(N+4, 2)/64;                        % 2nd derivative.
B = [ I([1,N+4],:) ; D2([1,N+4],:) ];           % BCs.
```

Finally, we are ready to define a function that computes u_t:

```
function F = timederiv(t,u)
  u_t = -D * (u + u.^2);              % Time derivative.
  F = [ u_t ; B*u ];                  % Append BCs.
end
```

In order to solve the PDE, we just define an initial condition and call the DAE solver, `ode15s`. We convert the solution to a 2D chebfun for plotting the result (see Fig. 6).

```
xFunc = chebpts(N+4, d, 2);                     % Chebyshev grid.
u0 = sech(xFunc);                               % Initial condition.
t = chebpts(400, [0, 200]);                     % Output times.
odeopt = odeset('mass', massmat, 'masssing', 'yes');
[t, u] = ode15s(timederiv, t, u0, odeopt);      % Solve.
U = chebfun2(u, [d, t(1), t(400)]); plot(U); % Plot.
```

The simulation takes less than half a second to run on a laptop.

8. Connections with other methods

We do not claim that the problems solved by our approach cannot be solved by any other collocation formulation by experts. Rather, our intent was to describe a systematic approach that can be implemented reliably in a fully automated fashion even for unusual problem types. Unlike many descriptions in the literature, our approach eliminates all decisions and manual or specialized manipulations in the discretization process.

The key concept behind the rectangular collocation approach we describe is to discretize each equation, rather than each solution function, at the same set of Chebyshev nodes. Due to the natural rectangularity of differentiation, a consequence is that the solution functions are discretized at more nodes than the equations, and different variables may be discretized at different numbers of nodes. This idea is not clearly acknowledged by spectral collocation using row replacement. Our formulation expresses the residual of a differential equation in a fully standard, rather than endpoint-truncated, interpolation form.

Other successful spectral methods effectively employ variations on the key idea of clearly separating the space of the solution from the space of the residual: spectral integral formulations of differential equations (Greengard, 1991; Driscoll, 2010), and ultraspherical spectral methods (Olver & Townsend, 2013), for example. In the tau spectral method (Lanczos, 1938; Fox, 1962), the problem (4.1)–(4.2) is approached by expressing the solution in the form $u(x) = \sum_{k=0}^{N} u_k T_k(x)$. This expression is substituted into (4.1), which is evaluated in spectral form (i.e., matching of polynomial coefficients). However, in order to make room for also imposing (4.2) in spectral form, the differential equation is imposed only up through degree $N - 1$.

The mathematical relationship between rectangular collocation and tau discretization is entirely determined by how (4.1) is evaluated—specifically, in how the multiplication terms $a(x)u'(x)$ and

$b(x)u(x)$ are reduced to degree $N-1$. If the coefficient functions are approximated to degree $N-1$ and Toeplitz/Hankel multiplication operators are created to represent the necessary convolution, as in Baszenski & Tasche (1997) and Olver & Townsend (2013), for example, then the difference from the rectangular collocation method is in aliasing of coefficients of index order $O(N)$, a difference that vanishes exponentially fast for smooth functions. In particular, when the coefficient functions are constant, proofs of tau convergence (Fox, 1962) should be straightforward to convert to proofs of rectangular collocation convergence.

The chief practical difference between tau and rectangular collocation is that, in the former, differentiation and integration are simple and sparse, whereas multiplication is messier and dense; in the latter case, the situation is exactly the opposite. Collocation is a bit more straightforward when it comes to unusual boundary conditions. Traditionally, collocation is also very different in nonlinear problems, where a nonlinear discretization is solved by standard optimization, but as mentioned in Section 4.2 we approach nonlinear problems by linearization *before* discretization, which greatly diminishes the difference from tau methods.

Another interesting and apparently general approach to problems with unusual boundary conditions is the fictitious point method (Fornberg, 2006). Here, again the number of points at which the solution is represented is greater, by the number of boundary conditions, than the number of points at which the equation is imposed. However, the additional nodes are not found as part of a standard Chebyshev grid, but instead are fictitious nodes outside the domain of the function. The discretized boundary conditions are then used to eliminate the direct influence in those extra nodes, as in Schur complementation. In the end, a square linear system is solved at the standard Chebyshev nodes. In theory, the specific locations of the fictitious nodes are irrelevant, because they all generate the same underlying interpolant before being eliminated, but in practice a little care is needed to preserve numerical stability. One also must use a procedure for generating differentiation matrices on arbitrary node sets (Fornberg, 1998; Weideman & Reddy, 2000). In practice, one should expect little difference from the rectangular collocation and tau approaches. However, it is built upon a theoretically unclear notion of convergence of approximations using interpolation nodes outside the domain, and it has a few extra steps that make it less convenient than the other two methods.

In conclusion, we have introduced a novel means of incorporating boundary conditions in Chebyshev spectral methods, which removes the ambiguity of the usual 'boundary bordering' approach in the face of nonstandard constraints or higher-order problems. We described how this new approach could easily be used to modify existing techniques for systems of ODEs, ODE eigenvalue problems and PDEs, and demonstrated this with numerous examples. The links noted above to other spectral methods in the literature, combined with the idea of linearization in continuous function space to effectively limit all discrete problems to linear ones, suggest that the variety of methods available may overlap a great deal in their theoretical underpinnings, differing substantially only in their implementations.

Acknowledgements

We thank Alfa Heryudono for bringing the Rosenau example in Section 7.5 to our attention, and The Chebfun Team for numerous helpful discussions.

Funding

This work was supported by The MathWorks, Inc. and by King Abdullah University of Science and Technology (KAUST), award KUK-C1-013-04.

References

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

Berrut, J. P. & Trefethen, L. N. (2004) Barycentric lagrange interpolation. *SIAM Rev.*, **46**, 501–517.

Birkisson, A. & Driscoll, T. A. (2012) Automatic Fréchet differentiation for the numerical solution of boundary-value problems. *ACM Trans. Math. Software*, **38**, 26:1–26:29.

Boyd, J. P. (2001) *Chebyshev and Fourier Spectral Methods*, 2nd edn. New York: Dover.

Canuto, C., Hussaini, M. Y., Quarteroni, A. & Zang, T. A. (1988) *Spectral Methods in Fluid Dynamics*. Berlin: Springer.

Driscoll, T. A. (2010) Automatic spectral collocation for integral, integro-differential, and integrally reformulated differential equations. *J. Comp. Phys.*, **229**, 5980–5998.

Driscoll, T. A., Bornemann, F. & Trefethen, L. N. (2008) The chebop system for automatic solution of differential equations. *BIT Numer. Math.*, **48**, 701–723.

Driscoll, T. A. & Fornberg, B. (1998) A block pseudospectral method for Maxwell's equations: I. One-dimensional case. *J. Comput. Phys.*, **140**, 1–19.

Driscoll, T. A., Hale, N. & Trefethen, L. N. (eds) (2014) *Chebfun Guide*, 1st edn. Oxford, UK: Pafnuty Publications.

Driscoll, T. A. & Weideman, J. A. C. (2014) Optimal domain splitting for interpolation by Chebyshev polynomials. *SIAM J. Num. Analy.*, **52**, 1913–1927.

Flyer, N. & Fornberg, B. (2003) On the nature of initial-boundary value solutions for dispersive equations. *SIAM J. Appl. Math*, **64**, 546–564.

Fornberg, B. (1996) *A Practical Guide to Pseudospectral Methods*. Cambridge: Cambridge University Press.

Fornberg, B. (1998) Calculation of weights in finite difference formulas. *SIAM Rev.*, **40**, 685–691.

Fornberg, B. (2006) A pseudospectral fictitious point method for high-order initial-boundary value problems. *SIAM J. Sci. Comput.*, **28**, 1716–1729.

Fox, L. (1962) Chebyshev methods for ordinary differential equations. *Comput. J.*, **4**, 318–331.

Golub, G. H. & Van Loan, C. F. (2012) *Matrix Computations*, 3rd edn. Baltimore, MD: JHU Press.

Greengard, L. (1991) Spectral integration and two-point boundary value problems. *SIAM J. Numer. Anal.*, **28**, 1071–1080.

Hesthaven, J. S. & Gottlieb, D. (1996) A stable penalty method for the compressible Navier–Stokes equations: I. open boundary conditions. *SIAM J. Sci. Comput.*, **17**, 579–612.

Higham, N. (2004) The numerical stability of barycentric Lagrange interpolation. *IMA J. Numer. Anal.*, **24**, 547–556.

Lanczos, C. (1938) Trigonometric interpolation of empirical and analytical functions. *J. Math. Phys.*, **17**, 123–199.

Olver, S. & Townsend, A. (2013) A fast and well-conditioned spectral method. *SIAM Rev.*, **55**, 462–489.

Townsend, A. & Trefethen, L. N. (2013) An extension of Chebfun to two dimensions. *SIAM J. Sci. Comput.*, **35**, C495–C518.

Trefethen, L. N. (2000) *Spectral Methods in MATLAB*. Philadelphia: Society for Industrial and Applied Mathematics.

Trefethen, L. N. (2012) *Approximation Theory and Approximation Practice*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Weideman, J. A. C. & Reddy, S. C. (2000) A MATLAB differentiation matrix suite. *ACM Trans. Math. Softw.*, **26**, 465–519.

Weideman, J. A. C. & Trefethen, L. N. (1988) The eigenvalues of second-order spectral differentiation matrices. *SIAM J. Numer. Anal.*, **25**, 1279–1298.

Xu, K. & Hale, N. (2014) Explicit construction of rectangular differentiation matrices (Submitted).