[Re] A Fast and Well-Conditioned Spectral Method: Dense Linear Algebra? In This Economy?

Kia Ghods
kia.ghods@princeton.edu
GitHub Repository: https://github.com/kiaghods/spectral-method

Contents

1 Introduction

Spectral methods have become an essential tool in the arsenal of a numerical analyst. Broadly, spectral methods solve differential equations by approximating an unknown function $u(x)$ as a truncated expansion in *global* basis functions (e.g., Fourier modes or orthogonal polynomials), and then solving for the resulting coefficient vector. Historically, the term "spectral" comes from harmonic analysis: one seeks to represent u by its expansion in eigenfunctions (modes) of a suitable operator, i.e., by its "spectrum" in an orthogonal basis.

For sufficiently smooth solutions, spectral discretizations often exhibit *super-algebraic* (and frequently near-exponential) convergence as the number of modes increases. However, classical formulations can suffer from poor conditioning and dense linear systems, which may compromise numerical stability and efficiency at high resolution. The paper of interest, "A Fast and Well-Conditioned Spectral Method" (S. Olver and Townsend, 2013), aims to rectify this tension by developing a spectral method that is both fast and well-conditioned, yielding linear systems with favorable structure that can be solved stably and efficiently.

In this paper, we will consider the family of linear ODE boundary value problems (BVPs) on $[-1, 1]$:

$$\mathcal{L}u = f \quad \text{and} \quad \mathcal{B}u = \mathbf{c} \tag{1}$$

where $\mathcal{L}$ is an N-th order linear differential operator

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

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

1.1 Banded and almost-banded matrices

We briefly recall the relevant notion of sparsity. A matrix $A \in \mathbb{C}^{m\times n}$ is called *banded* with bandwidth p if $A_{ij} = 0$ whenever $|i - j| > p$. Equivalently, nonzeros are confined to a diagonal band of width $2p + 1$ around the main diagonal.
In this work, the discretizations that arise are typically *almost banded*: all but a fixed number (independent of n) of rows are banded with a small bandwidth, while a small number of additional rows (introduced by boundary conditions) may be dense. The almost banded structure is crucial; it retains the efficiency of banded linear algebra while accommodating boundary constraints via boundary bordering.

2 Discretization in Coefficient Space

2.1 Chebyshev expansions

We approximate the unknown solution u by a truncated Chebyshev series

$$u(x) \approx \sum_{k=0}^{n-1} u_k T_k(x), \qquad \mathbf{u} := (u_0, \ldots, u_{n-1})^\top \in \mathbb{C}^n, \tag{2}$$

where $\{T_k\}_{k\geq 0}$ are the Chebyshev polynomials of the first kind on $[-1, 1]$. The goal is to rewrite the action of the differential operator $\mathcal{L}$ on u as a structured linear operator acting on the coefficient vector $\mathbf{u}$.

2.2 Sparse differentiation via a change of basis

A central observation is that differentiation is sparse in coefficient space provided we allow the polynomial basis to change. Using the identity

$$\frac{\mathrm{d}}{\mathrm{d}x} T_k(x) = k\, U_{k-1}(x),$$

where U_k are Chebyshev polynomials of the second kind (equivalently, the ultraspherical family with parameter $\lambda = 1$), the derivative of a Chebyshev expansion naturally admits an expansion in an ultraspherical basis:

$$u'(x) = \sum_{k\geq 1} k u_k\, C^{(1)}_{k-1}(x),$$

where we follow S. Olver and Townsend (2013) and write $C^{(1)}_j$ for the $\lambda = 1$ ultraspherical polynomials (so $C^{(1)}_j \equiv U_j$ under this convention).

Consequently, the first-derivative operator $\mathcal{D}_0$ mapping Chebyshev coefficients to ultraspherical $(\lambda = 1)$ coefficients is sparse: for $\mathbf{u} = \begin{pmatrix} u_0 & \dots & u_{n-1} \end{pmatrix}^\top$,

$$(\mathcal{D}_0 \mathbf{u})_k = (k+1)\, u_{k+1}, \qquad k = 0, \dots, n-2,$$

i.e., $\mathcal{D}_0 \in \mathbb{C}^{(n-1)\times n}$ is a shifted diagonal matrix,

$$\mathcal{D}_0 = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 & n-1 \end{pmatrix}.$$

Higher derivatives are obtained by composing sparse differentiation operators across successive ultraspherical bases, which is the first ingredient leading to an almost-banded discretization.

2.3 Conversion operators between ultraspherical bases

Differentiation maps coefficients from one ultraspherical family to the next, so in order to *add* different terms in $\mathcal{L}u$ we must be able to express all components in a common basis. For $\lambda \geq 0$, let $C^{(\lambda)}_k$ denote the ultraspherical polynomial of degree k with parameter λ (with $C^{(0)}_k \equiv T_k$ and $C^{(1)}_k \equiv U_k$ under the convention of S. Olver and Townsend (2013)).

The Chebyshev-to-$C^{(1)}$ conversion $\mathcal{S}_0$. The Chebyshev polynomials satisfy the identity

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

and therefore, if $u(x) = \sum_{k\geq 0} u_k T_k(x)$, regrouping terms yields a $C^{(1)}$-series

$$u(x) = \sum_{k\geq 0} v_k C_k^{(1)}(x), \qquad \mathbf{v} = \mathcal{S}_0 \mathbf{u},$$

with

$$v_0 = u_0 - \frac{1}{2}u_2, \qquad v_k = \frac{1}{2}(u_k - u_{k+2}),\ k \geq 1.$$

Equivalently, the conversion operator $\mathcal{S}_0$ is the sparse, banded matrix

$$\mathcal{S}_0 = \begin{pmatrix} 1 & 0 & -\frac{1}{2} & 0 & 0 & \cdots \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & \cdots \\ 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}.$$

General conversion $\mathcal{S}_\lambda$ for $\lambda \geq 1$. More generally, ultraspherical polynomials satisfy

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

which induces a linear map $\mathbf{v} = \mathcal{S}_\lambda \mathbf{u}$ from $C^{(\lambda)}$ coefficients to $C^{(\lambda+1)}$ coefficients. In particular, regrouping coefficients gives

$$v_0 = u_0 - \frac{\lambda}{\lambda+2}u_2, \qquad v_k = \frac{\lambda}{\lambda+k}u_k - \frac{\lambda}{\lambda+k+2}u_{k+2},\ k \geq 1,$$

so $\mathcal{S}_\lambda$ is sparse and banded, with nonzeros on the main diagonal and the second superdiagonal:

$$\mathcal{S}_\lambda = \begin{pmatrix} 1 & 0 & -\dfrac{\lambda}{\lambda+2} & 0 & 0 & \cdots \\ 0 & \dfrac{\lambda}{\lambda+1} & 0 & -\dfrac{\lambda}{\lambda+3} & 0 & \cdots \\ 0 & 0 & \dfrac{\lambda}{\lambda+2} & 0 & -\dfrac{\lambda}{\lambda+4} & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}.$$

2.4 Multiplication operators in coefficient space

To handle variable coefficients $a(x)$ multiplying $u(x)$ (and its derivatives), we introduce multiplication operators acting on coefficient vectors. We begin with the Chebyshev ($\lambda = 0$) case. Suppose

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

Then $\mathbf{c} = \mathcal{M}_0[a]\mathbf{u}$, where $\mathcal{M}_0[a]$ is a Toeplitz plus an almost-Hankel operator:

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

At first glance $\mathcal{M}_0[a]$ (and its truncations) appear dense. However, if a is smooth then its Chebyshev coefficients decay rapidly, and truncating

$$a(x) \approx \sum_{j=0}^{m-1} a_j T_j(x)$$

implies that the $n \times n$ principal part of $\mathcal{M}_0[a]$ is banded with bandwidth m for $n > m$.

Remark (general λ). The same idea extends to ultraspherical series: for each $\lambda \geq 1$ one defines $\mathcal{M}_\lambda[a]$ by

$$a(x)\Big(\sum_{k\geq 0} u_k C_k^{(\lambda)}(x)\Big) = \sum_{j\geq 0} v_j C_j^{(\lambda)}(x), \qquad \mathbf{v} = \mathcal{M}_\lambda[a]\mathbf{u},$$

and with a truncation of the $C^{(\lambda)}$ expansion of a, the projected matrix becomes banded.

2.5 Assembling the discretized operator

With $\mathcal{D}_\lambda$, $\mathcal{S}_\lambda$, and $\mathcal{M}_\lambda[\cdot]$ in hand, we can now write the action of $\mathcal{L}$ on u as a single structured linear operator on the Chebyshev coefficient vector $\mathbf{u}$.

Warm-up: first-order example. Consider the first-order ODE

$$u'(x) + a(x)u(x) = f(x).$$

The derivative term u' naturally lives in the $\lambda = 1$ ultraspherical basis, while the product $a(x)u(x)$ is most naturally formed in the Chebyshev basis and then converted. In coefficient space this becomes

$$\underbrace{\mathcal{D}_0}_{\text{Cheb}\to C^{(1)}} \mathbf{u} + \underbrace{\mathcal{S}_0}_{\text{Cheb}\to C^{(1)}} \underbrace{\mathcal{M}_0[a]}_{\text{multiply in Cheb}} \mathbf{u} = \underbrace{\mathcal{S}_0}_{\text{Cheb}\to C^{(1)}} \mathbf{f},$$

i.e.,

$$\Big(\mathcal{D}_0 + \mathcal{S}_0\mathcal{M}_0[a]\Big)\mathbf{u} = \mathcal{S}_0\mathbf{f}.$$

This already exhibits the main structural phenomenon: $\mathcal{D}_0$ is sparse, $\mathcal{S}_0$ is banded, and (after truncating a) $\mathcal{M}_0[a]$ is banded, so the resulting discretization is almost banded.

General N-th order operator. For the N-th order differential operator

$$\mathcal{L}u = \sum_{k=0}^{N} a^k(x)\, \frac{\mathrm{d}^k u}{\mathrm{d}x^k},$$

each derivative $\frac{\mathrm{d}^k u}{\mathrm{d}x^k}$ is represented in the $C^{(k)}$ ultraspherical basis via the sparse product

$$\mathcal{D}_{k-1} \cdots \mathcal{D}_0\, \mathbf{u}.$$

Multiplication by a^k is then applied in that same basis using $\mathcal{M}_k[a^k]$, and finally converted up to a common target basis, which we take to be $C^{(N)}$, using a chain of conversion operators. Concretely, define for each $k \le N$ the conversion-to-$C^{(N)}$ map

$$\mathcal{S}_{k\to N} \;:=\; \begin{cases} \mathcal{S}_{N-1}\mathcal{S}_{N-2}\cdots\mathcal{S}_k, & k \le N-1, \\ I, & k = N. \end{cases}$$

(so $\mathcal{S}_{N\to N} = I$ and $\mathcal{S}_{0\to N} = \mathcal{S}_{N-1}\cdots\mathcal{S}_0$). Then the full coefficient-space discretization of $\mathcal{L}$ is

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

where for $k = 0$ we interpret the empty product $\mathcal{D}_{-1}\cdots\mathcal{D}_0$ as the identity. The right-hand side must be represented in the same $C^{(N)}$ basis:

$$\mathbf{f}_N := \mathcal{S}_{0\to N}\, \mathbf{f}.$$

Thus the interior discretization takes the form

$$\mathcal{L}_N \mathbf{u} = \mathbf{f}_N.$$

Remark: Why a common basis? The reason we convert all terms to the same $C^{(N)}$ basis is simple: we need to *add* the contributions from different derivative orders. Since

$$\mathcal{L}u = \sum_{k=0}^{N} a^k(x)\frac{\mathrm{d}^k u}{\mathrm{d}x^k},$$

each $\frac{\mathrm{d}^k u}{\mathrm{d}x^k}$ naturally lives in the $C^{(k)}$ basis after differentiation. Without a common target basis, we would be attempting to add coefficient vectors that represent expansions in different polynomial families, which is an operation that is not well-defined. By converting everything to $C^{(N)}$, we obtain a single coefficient vector in a single basis, which can then be equated to the right-hand side (also expressed in $C^{(N)}$).

2.6 Imposing boundary conditions via boundary bordering

To impose K boundary conditions $\mathcal{B}u = \mathbf{c}$, we append (or replace) K rows of the truncated system with rows encoding the boundary functionals acting on the Chebyshev coefficients of u. Equivalently, we form a finite-dimensional system $A\mathbf{u} = \mathbf{b}$ by taking a sufficiently large truncation of $\mathcal{L}_N$ and then *bordering* it with K boundary rows. The resulting matrix is *almost banded*: it is banded in the interior rows (coming from $\mathcal{L}_N$) with a small number of dense boundary rows (coming from $\mathcal{B}$).

Example: Dirichlet and Neumann conditions. If $u(x) = \sum_{k=0}^{n-1} u_k T_k(x)$, then since $T_k(1) = 1$ and $T_k(-1) = (-1)^k$,

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

so Dirichlet boundary rows are

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \end{pmatrix}, \qquad \begin{pmatrix} 1 & -1 & 1 & \cdots & (-1)^{n-1} \end{pmatrix}.$$

For Neumann data, using $T_k'(x) = kU_{k-1}(x)$ with $U_{k-1}(1) = k$ and $U_{k-1}(-1) = (-1)^{k-1}k$ yields

$$u'(1) = \sum_{k=1}^{n-1} k^2 u_k, \qquad u'(-1) = \sum_{k=1}^{n-1} (-1)^{k-1} k^2 u_k,$$

so the corresponding boundary rows are

$$\begin{pmatrix} 0 & 1^2 & 2^2 & \cdots & (n-1)^2 \end{pmatrix}, \qquad \begin{pmatrix} 0 & 1^2 & -2^2 & \cdots & (-1)^{n-2}(n-1)^2 \end{pmatrix}.$$

3 Fast and Well-Conditioned Linear Algebra

The coefficient-space discretization in the previous section produces a finite linear system

$$A_n \mathbf{u} \approx \mathbf{b}_n,$$

where A_n is *almost banded*: the interior rows come from the ultraspherical discretization of $\mathcal{L}$ and remain banded (after truncating coefficient multipliers), while a small number K of boundary rows are dense. The point of S. Olver and Townsend (2013) is that we can solve such systems stably *and* in essentially linear time (up to bandwidth factors), without destroying the sparse structure.

3.1 Conditioning and a diagonal preconditioner

The projection operator. Throughout this section, we work with truncated (finite-dimensional) versions of the infinite-dimensional operators introduced earlier. The truncation is performed via the $n \times \infty$ projection operator

$$P_n = (I_n, 0),$$

where I_n is the $n \times n$ identity matrix and the zero block represents an $n \times \infty$ matrix of zeros. Applying P_n to an infinite coefficient vector $\begin{pmatrix} u_0 & u_1 & u_2 & \dots \end{pmatrix}^\top$ simply extracts the first n entries, yielding $\begin{pmatrix} u_0 & u_1 & \dots & u_{n-1} \end{pmatrix}^\top \in \mathbb{C}^n$. The adjoint P_n^* embeds a finite vector back into the infinite-dimensional space by padding with zeros.

Conditioning concerns. A recurring issue in classical spectral discretizations is that the linear systems become poorly conditioned as n grows. For standard tau and collocation methods, the 2-norm condition number typically grows as $O(n^{2N})$ for an N-th order differential equation (Canuto et al. 2006), which can erase the benefits of spectral accuracy in finite precision arithmetic.
The key stability result of S. Olver and Townsend (2013) is twofold:

- Even without preconditioning, the condition number growth for the ultraspherical discretization is significantly milder; only $O(n)$ rather than polynomial in n.
- Moreover, a simple *diagonal* (column-scaling) preconditioner can yield a system with *bounded* condition number as $n \to \infty$ in appropriate norms, and in particular in the standard 2-norm for Dirichlet boundary conditions.

The diagonal preconditioner. For an N-th order differential equation (1) with $K = N$ boundary conditions and assuming the leading coefficient satisfies $a^N(x) = 1$ (otherwise, divide through by a^N), the diagonal preconditioner is defined as

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

Its finite $n \times n$ truncation is

$$R_n = P_n R P_n^* = \frac{1}{2^{N-1}(N-1)!}\ \mathrm{diag}\left(1, \ldots, 1, \frac{1}{N}, \frac{1}{N+1}, \ldots, \frac{1}{n-1}\right) \in \mathbb{C}^{n \times n}.$$

The scaling factors grow mildly as $1/k$ for $k \geq N$, which reflects the natural decay rate of Chebyshev coefficients for smooth solutions.

Coefficient-space norms and functional analysis. The stability analysis in S. Olver and Townsend (2013) takes place in weighted ℓ^2 spaces (denoted ℓ^2_λ in their notation), defined by

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

These spaces correspond to Sobolev-type regularity: if $u(x) = \sum_{k \geq 0} u_k T_k(x)$, then $\mathbf{u} \in \ell^2_\lambda$ roughly means that u and its first λ derivatives are square-integrable. The parameter λ depends on the highest-order derivative appearing in the boundary conditions: for example, Dirichlet conditions require $\lambda = 1$ (so that evaluation functionals are bounded on ℓ^2_1), while Neumann conditions require $\lambda = 2$.

With this functional-analytic setup, the preconditioned discretized operator can be written as

$$A_n R_n = P_n \begin{pmatrix} \mathcal{B} \\ \mathcal{L} \end{pmatrix} R P_n^* = P_n (I + \mathcal{K}) P_n^*,$$

where $\mathcal{K} : \ell^2_\lambda \to \ell^2_\lambda$ is a *compact* operator for appropriate λ. Compact perturbations of the identity are Fredholm operators: they are invertible if and only if they have trivial kernel, and their finite-rank truncations $P_n(I+\mathcal{K})P_n^*$ inherit stable invertibility with uniformly bounded condition number as $n \to \infty$.

Practical takeaway. The diagonal matrix R rescales the coefficient directions so that high-order terms (which naturally have smaller magnitude) are normalized to be comparable with low-order terms. This prevents the ill-conditioning that would otherwise arise from the disparity in coefficient scales. In practice, one solves

$$A_n R_n \mathbf{v} = \mathbf{b}, \qquad \text{then recovers} \quad \mathbf{u} = R_n \mathbf{v}.$$

The result is a linear system with $\kappa_2(A_n R_n) = O(1)$ (for Dirichlet problems), ensuring that QR factorization and back-substitution remain numerically stable even for very large n.

3.2 QR for almost-banded matrices via *filled-in* structure

Why QR factorization? We solve the linear system $A_n \mathbf{u} = \mathbf{b}_n$ using QR factorization via Givens rotations rather than LU decomposition with partial pivoting. The reasons are threefold:

1. **Stability:** QR factorization by Givens rotations is backward stable without any pivoting, with a backward error bounded by a small multiple of machine epsilon (Higham 2002). In contrast, for almost-banded matrices with dense boundary rows, partial pivoting in LU can introduce significant fill-in and may not guarantee stability.

2. **Column scaling invariance:** The diagonal preconditioner R scales the columns of A_n. Givens rotations (which act on rows) are unaffected by column scaling, so the stability of QR applied to $A_n R_n$ is identical to that of A_n (S. Olver and Townsend 2013, Proposition 4.1). This is not true for LU, where column scaling can affect pivot selection and numerical behavior.

3. **Adaptive truncation:** The QR algorithm naturally exposes a residual tail that allows us to determine the optimal truncation n_{opt} on the fly (see §3.4). This adaptive feature is difficult to achieve with LU-based approaches.

For these reasons, we adopt QR factorization as the core linear algebra workhorse.

Givens rotations. A Givens rotation $G(i, j, \theta)$ is a sparse orthogonal matrix that acts as the identity except in rows i and j, where it performs a plane rotation:

$$G(i,j,\theta) = \begin{pmatrix} I_{i-1} & & & & \\ & \cos\theta & & -\sin\theta & \\ & & I_{j-i-1} & & \\ & \sin\theta & & \cos\theta & \\ & & & & I_{n-j} \end{pmatrix}.$$

Given two entries a and b in positions (i, k) and (j, k) of a matrix, we choose θ such that

$$c = \cos\theta = \frac{a}{\sqrt{a^2 + b^2}}, \qquad s = \sin\theta = \frac{-b}{\sqrt{a^2 + b^2}},$$

so that applying $G(i, j, \theta)$ from the left zeros out the (j, k) entry.

QR factorization via Givens rotations. Standard QR factorization proceeds column-by-column: for the k-th column, apply Givens rotations $G(k, k+1, \theta_1), G(k+1, k+2, \theta_2), \ldots$ to zero all subdiagonal entries. The product $Q = G_1^\top G_2^\top \cdots G_N^\top$ is orthogonal and $R = G_N \cdots G_2 G_1 A$ is upper triangular.

The filled-in structure. If A_n were purely banded, each Givens rotation would preserve bandedness and the algorithm would be straightforward. The problem is the K boundary rows: naive elimination can create fill-in far to the right, destroying sparsity.
The key insight is that after j columns have been reduced, the intermediate matrix B can be represented as a j *filled-in matrix* (Definition 5.1 in the original paper). Specifically:

- The first j rows are upper triangular with bandwidth m, plus a *rank-K tail* that can be written as a linear combination of the K boundary rows.
- The next $m_L + 1$ rows are banded with a similar rank-K tail.
- All remaining rows are purely banded (unchanged from the original operator).

Concretely, each row $k \le j$ has the form

$$e_k^\top B = (\underbrace{0, \ldots, 0}_{k-1}, \underbrace{B_{k,k}, \ldots, B_{k,k+m-1}}_{\text{banded}}, \quad \underbrace{\mathbf{b}_k^\top B_{1:K,k+m:n}}_{\text{fill-in as boundary combo}} \quad),$$

where $\mathbf{b}_k \in \mathbb{C}^K$ encodes how the k-th row's tail is a linear combination of the boundary rows.
Because Givens rotations are applied only on the left (row operations), this representation is preserved throughout elimination: each row operation mixes two rows, and if both are linear combinations of the boundary rows in their tails, so is the result.

Storage and cost. Each row is described using only $O(m + K)$ numbers (bandwidth m plus K boundary-combination coefficients), so the entire QR process requires $O(mn)$ storage. The arithmetic cost is $O(m^2 n)$ operations — essentially banded QR complexity, unaffected by the boundary rows.

3.3 Back substitution that respects boundary-induced fill-in

After n stages of Givens rotations, the leading $n \times n$ block R_n is upper triangular. However, the boundary-induced fill-in means that naive back substitution would still require $O(mn + Kn)$ operations per row.

Algorithm. The trick is to maintain an auxiliary vector $\mathbf{p}_k \in \mathbb{C}^K$ that accumulates the contribution of the boundary rows to the "tail" of the solution. Working backwards from $k = n$ to $k = 1$:

Algorithm 1 Back substitution for filled-in matrices

```
$\mathbf{p}_{n-m} \leftarrow \mathbf{0}$                                    ▷ Initialize tail accumulator
**for** $k = n - m - 1$ **down to** $1$ **do**
    $\mathbf{p}_k \leftarrow u_{k+m+1} B_{1:K,k+m+1} + \mathbf{p}_{k+1}$                  ▷ Update tail
    $u_k \leftarrow \frac{1}{B_{k,k}} \left( c_k - \sum_{s=1}^{m} B_{k,k+s} u_{k+s} - \mathbf{b}_k^\top \mathbf{p}_k \right)$
**end for**
```

This is mathematically equivalent to standard back substitution but uses only $O(mn)$ arithmetic operations and reduces round-off accumulation in practice by avoiding explicit formation of the dense tail.

3.4 Adaptive QR and optimal truncation

A major practical question is: *how large should n be?* You typically do not know n_{opt} (the truncation resolving the solution to near machine precision) ahead of time.

A naive approach increases n and resolves from scratch until coefficients stabilize—expensive and wasteful. Instead, S. Olver and Townsend (2013) adapts Olver-style ideas (F. Olver 1967) to the present setting by growing the QR factorization itself. The key observations are:

- the filled-in representation is independent of the number of columns, so new columns/rows can be appended without recomputing earlier work;
- many operator entries can be evaluated lazily with bounded cost per entry;
- the QR process produces, essentially for free, an *exact residual tail* whose norm directly measures the truncation error.

Residual tail and forward error. Concretely, after reducing the first n columns via Givens rotations Q_n^* (where $Q_n \in \mathbb{C}^{(n+m_L)\times(n+m_L)}$), we obtain

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

where $R_n \in \mathbb{C}^{n\times n}$ is upper triangular. Applying the same transformations to the right-hand side gives

$$\begin{pmatrix} Q_n^* & \\ & I \end{pmatrix} \begin{pmatrix} \mathbf{c} \\ \mathcal{S}_{0\to N}\mathbf{f} \end{pmatrix} = \begin{pmatrix} \mathbf{r}_{1:n} \\ \mathbf{r}_{n+1:n+m_L} \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{b} := \begin{pmatrix} \mathbf{c} \\ \mathcal{S}_{0\to N-1}\mathbf{f} \end{pmatrix}$$

where we note that $\mathbf{f}$ has finitely-many nonzero entries and we assume $n + m_L$ is greater than the number of nonzero entries. Solving $R_n\mathbf{u}_n = \mathbf{r}_{1:n}$ yields the computed coefficient vector $\mathbf{u}_n \in \mathbb{C}^n$. The *forward error* in this context is the error $\|\mathbf{u} - P_n^*\mathbf{u}_n\|$ in the infinite-dimensional coefficient space, where $\mathbf{u}$ is the exact solution (as an infinite vector) and $P_n^*\mathbf{u}_n$ is the computed solution padded with zeros.

The key observation is that the residual

$$\begin{pmatrix} Q_n^* & \\ & I \end{pmatrix} \mathcal{A} \begin{pmatrix} \mathbf{u}_n \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} Q_n^* & \\ & I \end{pmatrix} \mathbf{b} = \begin{pmatrix} \mathbf{0} \\ -\mathbf{r}_{n+1:n+m_L} \\ \mathbf{0} \end{pmatrix}$$

is *explicitly computable*: the tail $\mathbf{r}_{n+1:n+m_L}$ is produced during the QR factorization at no additional cost. Moreover, since Q_n^* is orthogonal, we have

$$\|\mathcal{A}P_n^*\mathbf{u}_n - \mathbf{b}\|_2 = \|\mathbf{r}_{n+1:n+m_L}\|_2.$$

Provided $\mathcal{A}^{-1}$ is well-conditioned (which is guaranteed by the diagonal preconditioner), the forward error satisfies

$$\|\mathbf{u} - P_n^*\mathbf{u}_n\|_2 \lesssim \|\mathcal{A}^{-1}\|_2 \cdot \|\mathbf{r}_{n+1:n+m_L}\|_2.$$

Thus, by monitoring $\|\mathbf{r}_{n+1:n+m_L}\|_2$ as we incrementally increase n, we can determine n_{opt} on the fly: stop when the residual tail falls below a desired tolerance (e.g., 10^{-14} for near-machine-precision accuracy). This adaptive strategy avoids the cost of repeated full solves and automatically adjusts to the smoothness of the solution.

Summary remarks

- **Fast:** the discretization is almost banded, and QR/back-substitution exploit a filled-in representation, giving near-linear complexity in n (up to bandwidth factors).
- **Stable:** QR with Givens rotations is backward stable, and (optionally) a diagonal column-scaling preconditioner yields bounded conditioning in the appropriate norms.
- **Automatic resolution:** adaptive QR exposes an explicit residual tail, enabling an efficient determination of n_{opt} without repeated full solves.

4 Numerical Experiments

We replicate a subset of the experiments in S. Olver and Townsend (2013), focusing on (i) coefficient convergence via Cauchy error, (ii) conditioning with/without diagonal preconditioning, and (iii) stiffness via boundary layers.

4.1 Airy equation: oscillatory regime and Cauchy error

We consider the singularly perturbed Airy BVP

$$\varepsilon u''(x) - xu(x) = 0, \qquad u(\pm 1) = \mathrm{Ai}\left(\pm\frac{1}{\varepsilon^{1/3}}\right), \tag{3}$$

whose exact solution is $u(x) = \mathrm{Ai}\big(\frac{x}{\varepsilon^{1/3}}\big)$. Following Fig. 3.1 of S. Olver and Townsend (2013), we take $\varepsilon = 10^{-9}$ and plot (i) the computed solution and (ii) the coefficient Cauchy error $\|\mathbf{u}_{\lceil 1.01n\rceil} - \mathbf{u}_n\|_2$ as a function of n in Fig. 1.

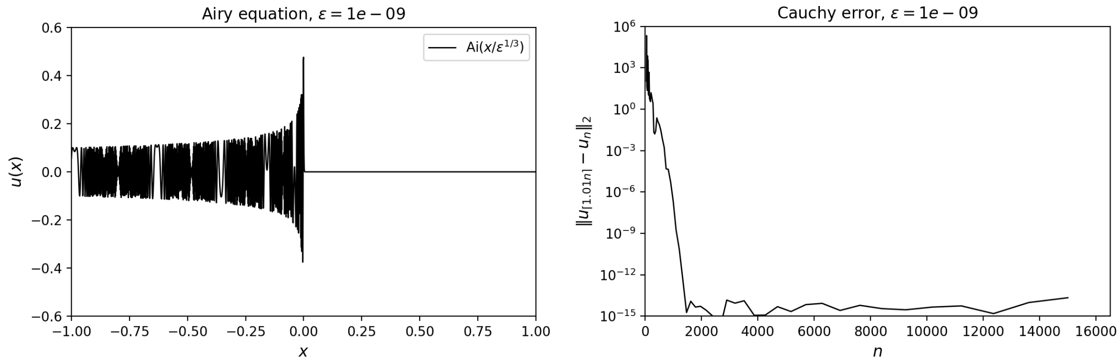

Figure 1: Airy BVP (3) with $\varepsilon = 10^{-9}$: (left) solution; (right) Cauchy error in coefficient space.

4.2 Conditioning and diagonal preconditioning

We next study $\kappa_2(A_n)$ for the Airy discretization as a function of n for $\varepsilon \in \{10^{-9}, 10^{-4}, 1\}$, reproducing the qualitative behavior of Fig. 3.2 in the original paper, and show this in our own Fig. 2.

For $\varepsilon = 1$ we additionally plot the preconditioned condition number $\kappa_2(A_n R_n)$, where R is the diagonal preconditioner from §3.1:

$$R = \frac{1}{2^{N-1}(N-1)!}\,\mathrm{diag}(\underbrace{1,\dots,1}_{N\text{ times}}, \tfrac{1}{N}, \tfrac{1}{N+1}, \dots), \qquad R_n = P_n R P_n^*.$$

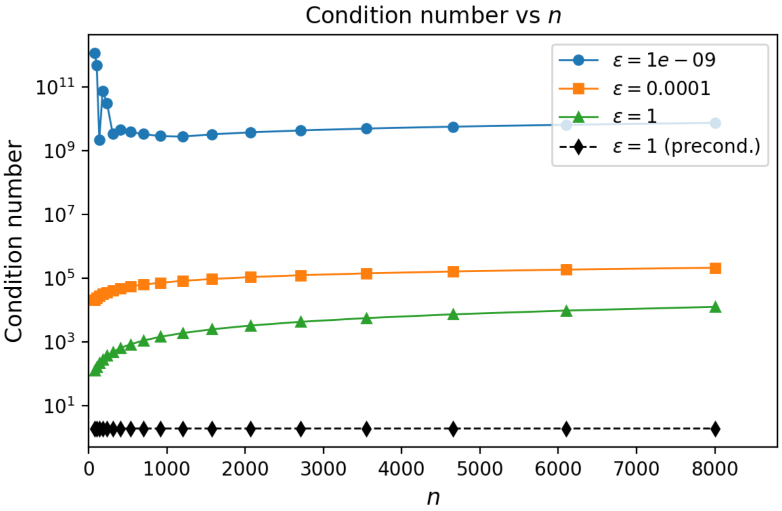

Figure 2: Condition number vs discretization size n for the Airy BVP (3); with/without diagonal preconditioning.

4.3 Boundary layers

We solve the boundary layer BVP (Eq. (3.11) in S. Olver and Townsend (2013))

$$\varepsilon u'' - 2x\Big(\cos x - \tfrac{8}{10}\Big)u' + \Big(\cos x - \tfrac{8}{10}\Big)u = 0, \qquad u(-1) = u(1) = 1, \tag{4}$$

with $\varepsilon = 10^{-7}$ and plot the solution and coefficient Cauchy error as in Fig. 3.3 from the original paper; we show this in our own Fig. 3. We note that the solution has two boundary layers at $\pm\cos^{-1}(8/10)$ both of width $\mathcal{O}(\epsilon^{1/4})$.

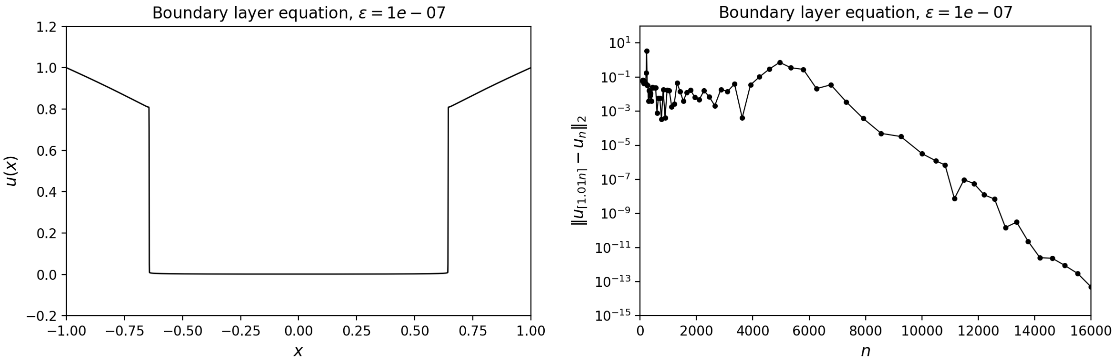

Figure 3: Boundary layer BVP (4) with $\varepsilon = 10^{-7}$: (left) solution; (right) Cauchy error in coefficient space.

Across these examples, the key diagnostics are (i) rapid decay of the coefficient tail, indicating spectral resolution, and (ii) controlled conditioning under diagonal scaling. Once the fast QR for almost-banded matrices is in place, the same experiments can be pushed to the regime $n \gg 10^5$ where dense solvers are infeasible.

5 Conclusion

This report reviewed the ultraspherical (spectral) discretization of S. Olver and Townsend (2013), whose central contribution is to reconcile spectral accuracy with numerically stable and efficient linear algebra. By expressing differentiation in a sequence of ultraspherical bases, the method represents derivatives by sparse (shifted-diagonal) operators and uses banded conversion operators to combine terms in a common basis. After truncating smooth variable coefficients, multiplication becomes banded as well, so the interior discretization remains banded and boundary conditions contribute only a fixed number of dense rows via boundary bordering. This almost-banded structure is exactly what enables the paper's fast, stable QR-based solver and its adaptive selection of the truncation size. Empirically, the numerical examples exhibit rapid coefficient convergence (spectral resolution) even in stiff regimes such as singular perturbations and boundary layers, while conditioning can be controlled further via simple diagonal scaling. Together, these ideas provide a practical blueprint for bringing "spectral accuracy" into settings where dense linear algebra would otherwise be the bottleneck.

Acknowledgements

This work was completed as part of *MAT 321 - Numerical Analysis and Scientific Computing* at Princeton University. I thank Professor Marc Aurèle Gilles for teaching me the ways of numerics.

Compute

Experiments were conducted on a MacBook Pro with an Apple M2 Max chip and 32 GB of RAM. All code was written in Python (`numpy/scipy/matplotlib`) and run locally. For each boundary value problem, we used an n-mode Chebyshev (coefficient-space) discretization and increased n until the coefficient tail stabilized, reporting both solution plots and convergence diagnostics (Cauchy error in coefficient space). Figures were generated directly from the Python scripts and saved as static PNGs for inclusion in this report.

References

Canuto, Claudio et al. (2006). *Spectral Methods: Fundamentals in Single Domains*. Springer Berlin Heidelberg. ISBN: 9783540307266. DOI: 10.1007/978-3-540-30726-6. URL: http://dx.doi.org/10.1007/978-3-540-30726-6.

Higham, Nicholas J. (Jan. 2002). *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics. ISBN: 9780898718027. DOI: 10.1137/1.9780898718027. URL: http://dx.doi.org/10.1137/1.9780898718027.

Olver, F.W.J. (Apr. 1967). "Numerical solution of second-order linear difference equations". In: *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics* 71B.2 and 3, p. 111. ISSN: 0022-4340. DOI: 10.6028/jres.071b.018. URL: http://dx.doi.org/10.6028/jres.071b.018.

Olver, Sheehan and Alex Townsend (2013). "A Fast and Well-Conditioned Spectral Method". In: *SIAM Review* 55.3, pp. 462–489. DOI: 10.1137/120865458. eprint: https://doi.org/10.1137/120865458. URL: https://doi.org/10.1137/120865458.