SIAM J. SCI. COMPUT.
Vol. 41, No. 6, pp. A3728–A3756

A SPARSE SPECTRAL METHOD ON TRIANGLES*

SHEEHAN OLVER†, ALEX TOWNSEND‡, AND GEOFFREY VASIL§

Abstract. In this paper, we demonstrate that many of the computational tools for univariate orthogonal polynomials have analogues for a family of bivariate orthogonal polynomials on the triangle, including Clenshaw's algorithm and sparse differentiation operators. This allows us to derive a practical spectral method for solving linear partial differential equations on triangles with sparse discretizations. We can thereby rapidly solve partial differential equations using polynomials with degrees in the thousands, resulting in sparse discretizations with as many as several million degrees of freedom.

Key words. spectral methods, triangles, sparse matrices, partial differential equations

AMS subject classification. 65N35

DOI. 10.1137/19M1245888

1. Introduction.

Univariate orthogonal polynomials are fundamental in applied and computational mathematics. They are used for the development of quadrature rules [9], spectral theory of Jacobi operators [27], eigenvalue statistics of random matrices [6], computational approximation theory [29], and to derive spectral methods for the numerical solution of differential equations [3, 4, 18, 21, 30, 31]. On the contrary, multivariate orthogonal polynomials currently have a more limited impact in applications and computational methods, though it is an active research area with a promising future.

To demonstrate the potential practical importance of multivariate orthogonal polynomials, we show that many computational tools for univariate orthogonal polynomials can be generalized to a family of bivariate orthogonal polynomials on a triangle. These tools allow us to derive a sparse spectral method for solving general linear partial differential equations (PDEs) with Dirichlet and Neumann conditions on triangles. While the techniques are general, we demonstrate the method on the following PDEs:

$$\begin{aligned} \Delta^2 u &= f(x,y) && \text{(biharmonic)},\\ u_y &= c u_x && \text{(transport)},\\ \Delta u + V(x,y)u &= f(x,y) && \text{(variable coefficient Helmholtz)}. \end{aligned}$$

Since triangles can be mapped to each other by affine translations, and polynomials remain polynomial, we can consider a single reference triangle, without loss of generality. Throughout this paper, we select the reference triangle to be the *unit simplex*: a right-angled triangle of unit height and width, i.e., $T = \{(x,y) : 0 < x < 1, 0 < y < 1 - x\}$.

*Submitted to the journal's Methods and Algorithms for Scientific Computing section February 19, 2019; accepted for publication (in revised form) August 29, 2019; published electronically November 21, 2019.

https://doi.org/10.1137/19M1245888

Funding: The second author is partially supported by the National Science Foundation grant 1818757.

†Department of Mathematics, Imperial College, London SW7 2AZ, UK (s.olver@imperial.ac.uk).

‡Department of Mathematics, Cornell University, Ithaca, NY 14853 (townsend@cornell.edu).

§School of Mathematics & Statistics, The University of Sydney, Sydney, New South Wales, 2006, Australia (geoffrey.vasil@sydney.edu.au).

There are several different families of bivariate orthogonal polynomials on T [8]. Here, we consider a family that is built from univariate orthogonal polynomials [12]:

$$P_{n,k}(x,y) = \tilde{P}_{n-k}^{(2k+1,0)}(x)(1-x)^k \tilde{P}_k^{(0,0)}\left(\tfrac{y}{1-x}\right), \qquad n \geq k \geq 0, \tag{1.1}$$

where $\tilde{P}_k^{(a,b)}(x)$ denotes the degree k shifted Jacobi polynomial on $[0,1]$ with parameters (a,b).[1] The polynomials in (1.1) are one possible generalization on triangles of the Legendre polynomials [15, Tab. 18.3.1]. In particular, the polynomials satisfy three-term recurrence relations (see (2.5)) and are orthogonal with respect to the standard L^2 inner-product on T:

$$\iint_T P_{n,k}(x,y)P_{m,\ell}(x,y)\mathrm{d}x\mathrm{d}y = \begin{cases} \frac{1}{\pi_{n,k}}, & (n,k) = (m,\ell), \\ 0, & (n,k) \neq (m,\ell), \end{cases}$$

where $\pi_{n,k} = 2(2k+1)(n+1)$. They provide a well-conditioned basis to represent integrable functions $f \in L^2(T)$ as a series expansion,

$$f(x,y) = \sum_{n=0}^{\infty}\sum_{k=0}^{n} f_{n,k}P_{n,k}(x,y), \quad f_{n,k} = \pi_{n,k}\iint_T f(x,y)P_{n,k}(x,y)\mathrm{d}x\mathrm{d}y,$$

where the first equality above should be understood in the L^2-sense. In order to do efficient computations with functions defined on a triangle, it is important to be able to rapidly compute expansion coefficients of $f(x,y)$ so that

$$f(x,y) \approx \sum_{n=0}^{N}\sum_{k=0}^{n} a_{n,k}P_{n,k}(x,y)$$

for a selected integer N. Recently, Slevinsky developed and implemented a fast backward stable algorithm for precisely this task [22, 23], accompanied with an optimized multithreaded open-source C library [24], allowing expansions to be computationally feasible for relatively large N. This has greatly improved the practicality of spectral methods for triangular domains.

The use of bivariate orthogonal polynomials on triangles has a long history in the spectral element method and p-finite element method (p-FEM) literature [11], going back to Dubiner [7]. The polynomials in (1.1) lead to highly structured p-FEM discretization matrices for PDEs of the form $\mathcal{L}u = -\nabla\cdot(A(x,y)\nabla u)$, and when $A(x,y)$ is a constant one can derive sparse discretizations that can be generated in optimal complexity [2, 14]. Other related works from the hp-FEM community that result in sparse discretizations have been achieved using Bernstein–Bézier polynomials [1]. The present work can be viewed as a generalization of [2, 14] to strong formulations of PDEs that are not necessarily elliptic, achieving sparse matrices using orthogonal bases, leading to well-conditioned solutions of partial differential equations (PDEs). Moreover, the properties of bivariate orthogonal polynomials allows us to retain sparsity for high differential order and variable coefficient PDEs (see subsection 4.1.2). Finally, the entries of the discretization are obtained directly as the product of sparse matrices whose entries are given in terms of simple rational formulae, avoiding the need for quadrature schemes.

[1] In particular, $\tilde{P}_k^{(a,b)}(x) = P_k^{(a,b)}(2x-1)$, where $P_k^{(a,b)}$ is the degree k Jacobi polynomial on $[-1,1]$ with parameters (a,b).

Our main idea is to exploit a hierarchy of sparse recurrence relations [19] that hold between the polynomials in (1.1) and the so-called Jacobi polynomials on the triangle [8, 12]:[2]

$$P_{n,k}^{(a,b,c)}(x,y) = \tilde{P}_{n-k}^{(2k+b+c+1,a)}(x)(1-x)^k \tilde{P}_k^{(c,b)}\left(\tfrac{y}{1-x}\right), \qquad n \geq k \geq 0, \tag{1.2}$$

where $a, b, c > -1$. In a manner that is analogous to the ultraspherical spectral method [18, 28], we represent the action of partial derivatives by representing the domain and range as vectors of coefficients in different bases so that the matrix representation is sparse. For example, while $\frac{\partial}{\partial y} P_{n,k}$ for $k \geq 1$ cannot be written as a sparse vector of $P_{n,k}$ coefficients, we have $\frac{\partial}{\partial y} P_{n,k} = (k+1) P_{n-1,k-1}^{(0,1,1)}$ (see Corollary A.1). This means that the first partial derivative with respect to y has a sparse matrix representation if the range is represented as a vector of $P_{n,k}^{(0,1,1)}$ coefficients. This can be summarized as

$$u = \sum_{n=0}^{N} \sum_{k=0}^{n} a_{n,k} P_{n,k} \quad \Rightarrow \quad \frac{\partial u}{\partial y} = \sum_{n=0}^{N-1} \sum_{k=0}^{n} (k+1) a_{n+1,k+1} P_{n,k}^{(0,1,1)}.$$

Moreover, these sparse recurrence relationships form a hierarchy, in the sense that $\frac{\partial^s}{\partial y^s}$ has a sparse representation if the range is represented as a vector of $P_{n,k}^{(0,s,s)}$ coefficients, for any $s \geq 0$. Similar, but slightly more complicated, sparse recurrence relations hold for $\frac{\partial^s}{\partial x^s} P_{n,k}$ when the range is represented as vectors in $P_{n,k}^{(s,0,s)}$ coefficients for any $s \geq 0$ (see subsection 3.4).

One is also able to combine sparse representations to discretize linear PDEs. For example, the Laplacian operator $\Delta u = u_{xx} + u_{yy}$ can be represented by a sparse matrix if the range is selected to be a vector of $P_{n,k}^{(2,2,2)}$ coefficients while the domain is a vector of $P_{n,k}$ coefficients. This is because there exist sparse conversion relationships for converting between certain $P_{n,k}^{(a,b,c)}$ bases (see subsection 3.1). Figure 1 illustrates a typical schema that illustrates how sparse recurrences are combined. In the language of finite element methods, the *test* and *trial* spaces are different with a sparse embedding of the trial space in the test space.

The paper is organized as follows. In section 2, we establish some general computational tools for bivariate orthogonal polynomials such as Jacobi operators and the bivariate Clenshaw algorithm. In section 3, we specialize to (1.2), where the additional structure allows us to achieve a more efficient Clenshaw algorithm. In section 4, we employ weighted Jacobi polynomials on the triangle to solve PDEs such as a variable coefficient Helmholtz equation and a biharmonic equation with zero Dirichlet conditions. In section 5, we extend the ideas to solve linear PDEs with nonzero Dirichlet conditions, and in section 6 we demonstrate that the framework easily generalizes to systems of PDEs so that it can be used to solve the Helmholtz equation in a polygonal domain.

The appendices contain relationships and additional formulae about orthogonal polynomials on the triangle. Our spectral method depends on explicit rational recurrence relationships that the polynomials $P_{n,k}^{(a,b,c)}(x,y)$ satisfy for differentiation, weighted differentiation, and conversion, which we detail in Appendix A. Tackling Dirichlet conditions requires a modification of the basis to enable sparse restriction

[2] The polynomials $P_{n,k}^{(a,b,c)}$ for $a, b, c > -1$ satisfy $\iint_T P_{n,k}^{(a,b,c)}(x,y) P_{m,\ell}^{(a,b,c)}(x,y) x^a y^b (1-x-y)^c \mathrm{d}x\mathrm{d}y = 0$ if $n \neq m$ or $k \neq \ell$.

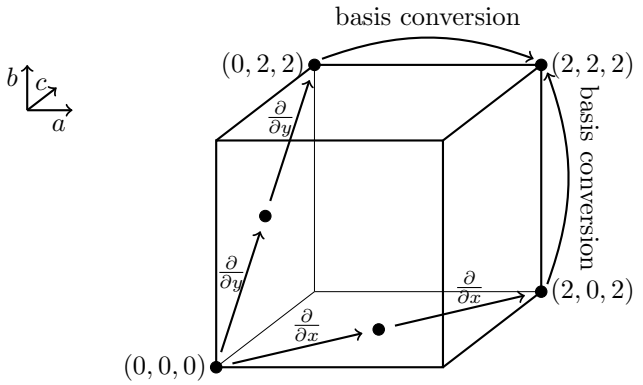

FIG. 1. *The Laplace operator acting on vectors of $P_{n,k} = P_{n,k}^{(0,0,0)}$ coefficients has a sparse matrix representation if the range is represented as vectors of $P_{n,k}^{(2,2,2)}$ coefficients. Here, the arrows indicate that the corresponding operation has a sparse matrix representation when the domain is $P_{n,k}^{(a,b,c)}$ coefficients, where (a, b, c) is at the tail of the arrow, and the range is $P_{n,k}^{(\tilde{a},\tilde{b},\tilde{c})}$ coefficients, where $(\tilde{a}, \tilde{b}, \tilde{c})$ is at the head of the arrow.*

operators, which we define as $Q_{n,k}^{(a,b,c)}$ in Appendix B. These also have explicit rational recurrence relationships for differentiation and conversion, which we derive in Appendix C.

Remark 1. An experimental version of the Julia implementation of this method, including all the examples below, is available [17].

2. Computations with bivariate orthogonal polynomials. In this section, we derive several computational tools for bivariate orthogonal polynomials such as the Jacobi operators, Clenshaw's algorithm, and multiplication operators. Later, in section 3, we specialize these tools to the Jacobi polynomials on the triangle (see (1.2)).

Consider a sequence of bivariate polynomials

$$p_{0,0}(x, y), p_{1,0}(x, y), p_{1,1}(x, y), p_{2,0}(x, y), p_{2,1}(x, y), p_{2,2}(x, y), \ldots,$$

where $\{p_{n,k}\}_{0\le k\le n\le N}$ is a basis for the space of bivariate polynomials of total degree $\le N$,[3] for any integer N. We say that such a sequence is orthogonal with respect to a nonnegative weight function $w(x, y)$ on $\Omega \subset \mathbb{R}^2$ if

$$\iint_\Omega w(x, y)p_{n,k}(x, y)p_{m,\ell}(x, y)\mathrm{d}x\mathrm{d}y = \begin{cases} d_{n,k}, & (n, k) = (m, \ell), \\ 0, & (n, k) \neq (m, \ell), \end{cases} \tag{2.1}$$

where $d_{n,k}$ are positive numbers.

It is notationally convenient to write the bivariate polynomials of the same total degree as a single vector-valued polynomial [8] as follows:

$$\mathbb{P}_n(x, y) = \begin{pmatrix} p_{n,0}(x, y) \\ \vdots \\ p_{n,n}(x, y) \end{pmatrix}.$$

[3]We say that a bivariate polynomial $q(x, y)$ is of total degree $\le N$ if $q(x, y) = \sum_{n=0}^{N}\sum_{k=0}^{n} b_{n,k}x^k y^{n-k}$ for some coefficients $b_{n,k}$.

One can then state the orthogonality condition in (2.1) more succinctly as

$$\iint_\Omega w(x,y)\mathbb{P}_m(x,y)\mathbb{P}_n(x,y)^\top \mathrm{d}x\mathrm{d}y = \begin{cases} D_n, & m=n, \\ \mathbf{0}, & m\neq n, \end{cases} \tag{2.2}$$

where D_n is the $(n+1)\times(n+1)$ diagonal matrix with entries $d_{n,k}$ for $0\le k\le n$, $\mathbf{0}$ is a matrix of all zeros of the appropriate size, and $\mathbb{P}_n(x,y)^\top$ denotes the transpose of $\mathbb{P}_n(x,y)$. The sequence of bivariate polynomials are normalized (orthonormal) if D_n is the identity matrix for all $n\ge 0$. We also use the notation

$$\mathbf{P}(x,y) = \left(\mathbb{P}_0(x,y)^\top, \mathbb{P}_1(x,y)^\top, \ldots\right)^\top$$

to encode all of the polynomials as a single infinite vector.

2.1. Bivariate function approximation. A sequence of bivariate orthogonal polynomials on $\Omega\subset\mathbb{R}^2$ can be used to approximate functions that are square integrable with respect to the associated weight function $w(x,y)$ on Ω. For example, provided $\iint_\Omega w(x,y)\left|f(x,y)\right|^2 \mathrm{d}x\mathrm{d}y<\infty$, we can write

$$f(x,y) = \sum_{n=0}^{\infty}\sum_{k=0}^{n} f_{n,k}p_{n,k}(x,y) = \sum_{n=0}^{\infty}\mathbb{P}_n(x,y)^\top \boldsymbol{f}_n = \mathbf{P}(x,y)^\top \boldsymbol{f}, \tag{2.3}$$

where $\boldsymbol{f}_n = (f_{n,0},\ldots,f_{n,n})^\top$ and $\boldsymbol{f} = (\boldsymbol{f}_0,\boldsymbol{f}_1,\ldots)^\top$ are the coefficients of the expansion. Here, the first equality in (2.3) is understood in the sense that the difference between the left- and right-hand sides is zero in the norm associated to the inner product.

The expansion coefficients in (2.3) are defined by the following integrals:

$$f_{n,k} = \frac{1}{d_{n,k}}\iint_\Omega w(x,y)f(x,y)p_{n,k}(x,y)\mathrm{d}x\mathrm{d}y, \qquad n\ge k\ge 0, \tag{2.4}$$

where $d_{n,k}$ is the orthogonality constant in (2.1). In practice, it is usually desirable for the expansion coefficients to rapidly decay, i.e., $\|\boldsymbol{f}_0\|, \|\boldsymbol{f}_1\|, \ldots$ is a rapidly decaying sequence.

2.2. Jacobi operators. In the theory of univariate orthogonal polynomials an important object is the Jacobi operator, which is a self-adjoint linear operator given by a tridiagonal matrix [27]. It is closely related to the fact that a sequence of univariate orthogonal polynomials satisfies a three-term recurrence. For example, if $p_0,p_1,\ldots$ is a sequence of univariate orthogonal polynomials, then

$$b_k p_{k+1}(x) + a_k p_k(x) + c_{k-1}p_{k-1}(x) = xp_k(x)$$

for $k\ge 1$ [26, Thm. 3.2.1] and

$$J\mathbf{P}(x) = x\mathbf{P}(x), \quad \mathbf{P}(x) = \begin{pmatrix} p_0(x) \\ p_1(x) \\ \vdots \end{pmatrix}, \qquad J = \begin{pmatrix} a_0 & b_0 & & \\ c_0 & a_1 & b_1 & \\ & c_1 & a_2 & \ddots \\ & & \ddots & \ddots \end{pmatrix}.$$

The Jacobi operator associated with $p_0,p_1,\ldots$ is the symmetric tridiagonal matrix obtained by a diagonal similarity transform of J [27]. This diagonal similarity transform corresponds precisely to the normalization factors required to orthonormalize

the sequence of univariate orthogonal polynomials. The transformation is possible provided $0 < b_k^{-1} c_k < \infty$ for all k. In particular, if $\{p_k(x)\}_{k\geq 0}$ are orthonormal, then J is a symmetric tridiagonal matrix.

A related fact that is important for designing spectral methods is that $J^\top$ can be interpreted as the "multiplication-by-x" operator, that is, if $f(x) = \boldsymbol{P}(x)^\top \boldsymbol{f}$, we have

$$xf(x) = x\boldsymbol{P}(x)^\top \boldsymbol{f} = \boldsymbol{P}(x)^\top J^\top \boldsymbol{f}.$$

In other words, $J^\top \boldsymbol{f}$ gives the coefficients of $xf(x)$.

The analogue for bivariate orthogonal polynomials is a *pair* of commuting operators J_x and J_y [8, section 3.4], which satisfy

$$J_x \mathbf{P}(x,y) = x\mathbf{P}(x,y), \qquad J_y \mathbf{P}(x,y) = y\mathbf{P}(x,y). \tag{2.5}$$

Here, J_x and J_y are block-tridiagonal operators so that

$$J_x = \begin{pmatrix} A_0^x & B_0^x & & \\ C_0^x & A_1^x & B_1^x & \\ & C_1^x & A_2^x & \ddots \\ & & \ddots & \ddots \end{pmatrix}, \qquad J_y = \begin{pmatrix} A_0^y & B_0^y & & \\ C_0^y & A_1^y & B_1^y & \\ & C_1^y & A_2^y & \ddots \\ & & \ddots & \ddots \end{pmatrix},$$

where $A_n^x, A_n^y \in \mathbb{R}^{(n+1)\times(n+1)}$, $B_n^x, B_n^y \in \mathbb{R}^{(n+1)\times(n+2)}$, and $C_n^x, C_n^y \in \mathbb{R}^{(n+2)\times(n+1)}$. When deriving spectral methods the operators J_x and J_y play an important role as they can be interpreted as operators for "multiplication-by-x" and "multiplication-by-y," respectively, that is,

$$x\mathbf{P}(x,y)^\top \boldsymbol{f} = \mathbf{P}(x,y)^\top J_x^\top \boldsymbol{f} \quad \text{and} \quad y\mathbf{P}(x,y)^\top \boldsymbol{f} = \mathbf{P}(x,y)^\top J_y^\top \boldsymbol{f}. \tag{2.6}$$

In other words, if $f(x,y) = \mathbf{P}(x,y)^\top \boldsymbol{f}$, then $J_x^\top \boldsymbol{f}$ and $J_y^\top \boldsymbol{f}$ give the coefficients of $xf(x,y)$ and $yf(x,y)$, respectively.

2.3. Recurrences and the Clenshaw algorithm. For univariate orthogonal polynomials, the three-term recurrence encoded by a Jacobi operator can be used to construct the polynomials themselves at a specified point via forward substitution. Clenshaw's algorithm is a closely related concept that allows the evaluation of a finite series expansion of univariate orthogonal polynomials at a point [5]. While it is common to interpret the three-term recurrence/Clenshaw's algorithm as recursions, we prefer to interpret them as forward/backward substitution on a lower/upper triangular system associated to the Jacobi operator as this point-of-view facilitates generalization to the bivariate setting.

Let $p_0(x), p_1(x), \ldots$ be a sequence of univariate orthogonal polynomials such that $p_0(x) = 1$, and suppose that we wish to evaluate $f(x) = \sum_{k=0}^N a_k p_k(x)$ at $x_* \in \mathbb{R}$. Since $p_0(x), p_1(x), \ldots$ satisfy a three-term recurrence of the form $b_k p_{k+1}(x) = (x - a_k)p_k(x) - c_{k-1}p_{k-1}(x)$ for $k \geq 1$ [26, Thm. 3.2.1], we find that

$$L_N(x_*) \begin{pmatrix} p_0(x_*) \\ p_1(x_*) \\ p_2(x_*) \\ \vdots \\ p_N(x_*) \end{pmatrix} = \begin{pmatrix} 1 & & & & \\ a_0 - x_* & b_0 & & & \\ c_0 & a_1 - x_* & b_1 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{N-2} & a_{N-1} - x_* & b_{N-1} \end{pmatrix} \begin{pmatrix} p_0(x_*) \\ p_1(x_*) \\ p_2(x_*) \\ \vdots \\ p_N(x_*) \end{pmatrix} = e_0, \tag{2.7}$$

where $b_0 p_1(x) = (a_0 - x)p_0(x)$ and $e_0 = (1, 0, \ldots, 0)^\top$.

Forward substitution on the lower triangular linear system in (2.7) allows one to evaluate $p_k(x_*)$ for $k \geq 0$ from which one could evaluate $f(x_*) = \sum_{k=0}^N a_k p_k(x_*)$. For stability purposes, the Clenshaw algorithm evaluates expansions more directly and can be written as

$$f(x_*) = (p_0(x_*), \ldots, p_N(x_*))^\top \boldsymbol{a} = e_0^\top \left((L_N(x_*))^{-\top} \boldsymbol{a} \right), \qquad \boldsymbol{a} = \begin{pmatrix} a_0 \\ \vdots \\ a_N \end{pmatrix}. \tag{2.8}$$

Therefore, the Clenshaw algorithm is equivalent to solving the upper triangular linear system $(L_N(x_*))^\top \boldsymbol{v} = \boldsymbol{a}$, followed by returning the first entry of $\boldsymbol{v}$. Since $L_N(x_*)$ only has three nonzero subdiagonals, the algorithm requires $\mathcal{O}(N)$ operations to evaluate $f(x_*) = \sum_{k=0}^N a_k p_k(x_*)$.

The bivariate case is more involved. Given $(x_*, y_*) \in \mathbb{R}^2$, we would like to evaluate $f(x, y) = \sum_{n=0}^N \sum_{k=0}^n a_{n,k} p_{n,k}(x, y)$ at (x_*, y_*), where (without loss of generality) we assume that $p_{0,0}(x, y) = 1$. Since there are three-term recurrence relations in both x and y (see (2.5)) we find that

$$L_N(x_*, y_*)\mathbf{P}(x_*, y_*) = \begin{pmatrix} 1 & & & \\ A_0^x - x_* I_1 & B_0^x & & \\ A_0^y - y_* I_1 & B_0^y & & \\ C_0^x & A_1^x - x_* I_2 & B_1^x & \\ C_0^y & A_1^y - y_* I_2 & B_1^y & \\ & \ddots & \ddots & \ddots \end{pmatrix} \mathbf{P}(x_*, y_*) = \begin{pmatrix} 1 \\ \mathbf{0}_{1\times 1} \\ \mathbf{0}_{1\times 1} \\ \mathbf{0}_{2\times 1} \\ \mathbf{0}_{2\times 1} \\ \vdots \end{pmatrix}, \tag{2.9}$$

where I_m is the $m \times m$ identity matrix and $\mathbf{0}_{m\times 1}$ is the zero vector of length m. Unlike the univariate case, the system is not lower triangular and so we cannot immediately invert this system via forward recurrence to find $\mathbf{P}(x_*, y_*)$.

A reformulation that allows for inversion is to multiply the system to reduce the blocks above the diagonal in (2.9) to the identity. First, note that the blocks

$$B_n = \begin{pmatrix} B_n^x \\ B_n^y \end{pmatrix} \in \mathbb{R}^{(2n+2)\times(n+2)}$$

have full column rank for $n \geq 0$ [8, Theorem 3.3.4]. Therefore, B_n has a left-inverse B_n^+ for $n \geq 0$ such that $B_n^+ B_n = I_{n+2}$. It follows that an equivalent evaluation scheme can be designed from

$$\tilde{L}_N(x_*, y_*)\mathbf{P}(x_*, y_*) = \begin{pmatrix} 1 \\ \mathbf{0}_{1\times 1} \\ \mathbf{0}_{2\times 1} \\ \vdots \end{pmatrix}, \quad \tilde{L}_N(x_*, y_*) = \begin{pmatrix} 1 & & & \\ & B_0^+ & & \\ & & B_1^+ & \\ & & & \ddots \end{pmatrix} L_N(x_*, y_*). \tag{2.10}$$

Since $\tilde{L}_N(x_*, y_*)$ is lower triangular we can construct $\mathbf{P}(x_*, y_*)$ via forward substitution.

Furthermore, a natural bivariate analogue of Clenshaw's algorithm follows from writing

$$f(x_*, y_*) = \mathbf{P}(x_*, y_*)^\top \boldsymbol{a} = \boldsymbol{e}_0^\top \left(\left(\tilde{L}_N(x_*, y_*) \right)^{-\top} \boldsymbol{a} \right).$$

Thus $f(x_*, y_*)$ can be evaluated by solving an upper triangular linear system using back substitution.

If B_n^+ are dense matrices for $n \geq 0$, then forward recurrence and Clenshaw's algorithm require $\mathcal{O}(N^3)$ operations. However, in the special case of Jacobi polynomials on the triangle, the matrices involved are sparse (see section 3) and the complexity can be reduced to $\mathcal{O}(N^2)$ operations, which is optimal.

2.4. Multiplication operators. The relations in (2.6) show that $J_x^\top$ and $J_y^\top$ are operators that represent "multiplication-by-x" and "multiplication-by-y," respectively, in the bivariate orthogonal polynomial basis. Here, we combine these operators together to construct multiplication matrices that represent multiplication by a degree d polynomial expanded as $q(x,y) = \sum_{n=0}^{d} \sum_{k=0}^{n} q_{n,k} p_{n,k}(x,y)$.

Suppose we are given a function $f(x,y) = \sum_{n=0}^{N} \sum_{k=0}^{n} a_{n,k} p_{n,k}(x,y)$, and wish to find the expansion coefficients of $g(x,y) = q(x,y)f(x,y)$, where the degrees of f and q can differ. Using $J_x^\top$ and $J_y^\top$, we find that

$$\boldsymbol{g} = M_q \boldsymbol{f}, \qquad M_q = q(J_x^\top, J_y^\top),$$

where the definition of $q(J_x^\top, J_y^\top)$ is (see [13, Def. 2.1])

$$q(J_x^\top, J_y^\top) = \sum_{n=0}^{d} \sum_{k=0}^{n} c_{nk} (J_x^\top)^{n-k} (J_y^\top)^k, \qquad q(x,y) = \sum_{n=0}^{d} \sum_{k=0}^{n} c_{nk} x^{n-k} y^k. \tag{2.11}$$

Since J_x and J_y are block-tridiagonal and each matrix-matrix product increases the block-bandwidth by one, we see that $q(J_x^\top, J_y^\top)$ is also a block-banded with upper and lower block-bandwidth d.

The expression in (2.11) is not ideal for computations when d is moderately large because of the inherent ill-conditioning in the monomial basis. It is often computationally beneficial to expand $q(x,y)$ in a bivariate orthogonal polynomial expansion and evaluate $q(J_x^\top, J_y^\top)$ using an operator-valued analogue of Clenshaw's algorithm [25, 31].

The operator-valued analogue of Clenshaw's algorithm for evaluating $q(J_x^\top, J_y^\top)$ is equivalent to the expression

(2.12)

$$M_q = (e_0 \otimes \mathcal{I})(L^{-\top} \boldsymbol{q}), \quad L = \begin{pmatrix} I_1 \otimes \mathcal{I} & & & & \\ A_0^x \otimes \mathcal{I} - I_1 \otimes J_x & B_0^x \otimes \mathcal{I} & & & \\ A_0^y \otimes \mathcal{I} - I_1 \otimes J_y & B_0^y \otimes \mathcal{I} & & & \\ C_0^x \otimes \mathcal{I} & A_1^x \otimes \mathcal{I} - I_2 \otimes J_x & B_1^x \otimes \mathcal{I} & & \\ C_0^y \otimes \mathcal{I} & A_1^y \otimes \mathcal{I} - I_2 \otimes J_y & B_1^y \otimes \mathcal{I} & & \\ & \ddots & & \ddots & \ddots \end{pmatrix},$$

where $\mathcal{I}$ is an infinite identity matrix, e_0 is the first canonical unit vector, and $\boldsymbol{q}$ is the vector of coefficients for $q(x,y)$ in the bivariate orthogonal polynomial expansion. Here we use the Kronecker product denoted $\otimes$.

In general, J_x and J_y have dense blocks so that the total number of nonzero entries in the principal $N \times N$ block matrix of $q(J_x^\top, J_y^\top)$ is $\mathcal{O}(N^3)$, and the complexity of constructing $q(J_x^\top, J_y^\top)$ using the operator-valued Clenshaw's algorithm is $\mathcal{O}(N^4)$ (where the total number of unknowns is $\mathcal{O}(N^2)$). In the case of Jacobi polynomials on the triangle, the blocks of J_x and J_y are tridiagonal and there are only $\mathcal{O}(N^2)$ nonzero entries, which can be calculated in optimal complexity.

3. Computing with Jacobi polynomials on the triangle. We now specialize the algorithmic ideas in section 2 to Jacobi polynomials on the triangle (see (1.2)). Since these polynomials have additional structure, more efficient algorithms can be designed.

We denote the Jacobi polynomials on the triangle that are orthogonal with respect to $x^a y^b (1-x-y)^c$ with $a, b, c > -1$ by

$$\mathbb{P}_n^{(a,b,c)}(x,y) = \begin{pmatrix} P_{n,0}^{(a,b,c)}(x,y) \\ \vdots \\ P_{n,n}^{(a,b,c)}(x,y) \end{pmatrix}, \qquad \mathbf{P}^{(a,b,c)}(x,y) = \begin{pmatrix} \mathbb{P}_0^{(a,b,c)}(x) \\ \mathbb{P}_1^{(a,b,c)}(x) \\ \vdots \end{pmatrix},$$

and note that series expansions in the $P_{n,k}^{(a,b,c)}$ basis can be expressed as

$$f(x,y) = \sum_{n=0}^{\infty} \sum_{k=0}^{n} f_{n,k} P_{n,k}^{(a,b,c)}(x,y) = \mathbf{P}^{(a,b,c)}(x,y)^\top \boldsymbol{f},$$

where $\boldsymbol{f}$ is the vector of $P_{n,k}^{(a,b,c)}$ coefficients for f. These expansion coefficients can be efficiently computed from samples of f by a fast, backward stable algorithm [22, 23, 24]. We further denote the shifted Jacobi polynomials on the unit interval as

$$\mathbf{P}^{(a,b)}(x) = \begin{pmatrix} \tilde{P}_0^{(b,a)}(x) \\ \tilde{P}_1^{(b,a)}(x) \\ \vdots \end{pmatrix},$$

where the alternative ordering of a and b helps to build analogies with the triangle case.

3.1. Conversion operators. An important property of Jacobi polynomials on the interval is that they have banded conversion operators, which translate between coefficients from expansion in $P_n^{(a,b)}$ to $P_n^{(a+1,b)}$ or $P_n^{(a,b+1)}$. In terms of converting expansions between bases, we can express such conversions as

$$f(x) = \mathbf{P}^{(a,b)}(x)^\top \boldsymbol{f} = \mathbf{P}^{(a+1,b)}(x)^\top S_{(a,b)}^{(a+1,b)} \boldsymbol{f} = \mathbf{P}^{(a,b+1)}(x)^\top S_{(a,b)}^{(a,b+1)} \boldsymbol{f},$$

where $S_{(a,b)}^{(a+1,b)}$ and $S_{(a,b)}^{(a,b+1)}$ are upper bidiagonal operators, with rational entries as given in [15, (18.9.5)].

Jacobi polynomials on the triangle have a similar property: we can increment either a, b, or c in the expansion by one:

$$\begin{aligned} f(x,y) &= \mathbf{P}^{(a,b,c)}(x,y)^\top \boldsymbol{f} = \mathbf{P}^{(a+1,b,c)}(x,y)^\top S_{(a,b,c)}^{(a+1,b,c)} \boldsymbol{f} \\ &= \mathbf{P}^{(a,b+1,c)}(x,y)^\top S_{(a,b,c)}^{(a,b+1,c)} \boldsymbol{f} = \mathbf{P}^{(a,b,c+1)}(x,y)^\top S_{(a,b,c)}^{(a,b,c+1)} \boldsymbol{f}. \end{aligned}$$

Each of these operators are sparse: they have block-bandwidths $(0,1)$ with diagonal blocks for $S_{(a,b,c)}^{(a+1,b,c)}$ and upper bidiagonal blocks for $S_{(a,b,c)}^{(a,b+1,c)}$ and $S_{(a,b,c)}^{(a,b,c+1)}$. The entries are rational, and can be determined in closed form by the recurrence relationships in Corollary A.3.

3.2. Constructing Jacobi operators. For Jacobi polynomials, the recurrence relationships that give rise to tridiagonal Jacobi operators, representing multiplication by x, are well known. However, the Jacobi operators can alternatively be derived via lower bidiagonal *lowering operators* $L^{(a-1,b)}_{(a,b)}$ and $L^{(a,b-1)}_{(a,b)}$ [15, (18.9.5)] that represent multiplication by x and $1-x$:

$$xf(x) = \mathbf{P}^{(a-1,b)}(x)^\top L^{(a-1,b)}_{(a,b)} \boldsymbol{f} = \mathbf{P}^{(a,b)}(x)^\top S^{(a,b)}_{(a-1,b)} L^{(a-1,b)}_{(a,b)} \boldsymbol{f}.$$

Similarly, $1-x$ is equivalent to $S^{(a,b)}_{(a,b-1)} L^{(a,b-1)}_{(a,b)}$. In other words, the Jacobi operator corresponding to multiplication by x can be constructed via

$$J^\top \equiv S^{(a,b)}_{(a-1,b)} L^{(a-1,b)}_{(a,b)} \equiv I - S^{(a,b)}_{(a,b-1)} L^{(a,b-1)}_{(a,b)}.$$

Note that the product of a lower bidiagonal operator $L^{(a-1,b)}_{(a,b)}$ and an upper bidiagonal operator $S^{(a,b)}_{(a-1,b)}$ is a tridiagonal operator, as expected.

To construct the Jacobi operators J_x and J_y for Jacobi polynomials on the triangle, we first note that there exist three lowering operators that satisfy

$$\begin{aligned} xf(x,y) &= \mathbf{P}^{(a-1,b,c)}(x,y)^\top L^{(a-1,b,c)}_{(a,b,c)} \boldsymbol{f}, \\ yf(x,y) &= \mathbf{P}^{(a,b-1,c)}(x,y)^\top L^{(a,b-1,c)}_{(a,b,c)} \boldsymbol{f}, \\ zf(x,y) &= \mathbf{P}^{(a,b,c-1)}(x,y)^\top L^{(a,b,c-1)}_{(a,b,c)} \boldsymbol{f}, \end{aligned}$$

where $z := 1-x-y$. We will use other indices to indicate multiple lowering in a row, e.g.,

$$L^{(0,0,0)}_{(1,1,1)} := L^{(0,0,0)}_{(1,0,0)} L^{(1,0,0)}_{(1,1,0)} L^{(1,1,0)}_{(1,1,1)}$$

corresponds to multiplication by xyz, where the choice for navigating the parameter tree is arbitrary.

We can construct the Jacobi operators from the lowering operators via

$$J_x^\top = S^{(a,b,c)}_{(a-1,b,c)} L^{(a-1,b,c)}_{(a,b,c)}, \qquad J_y^\top = S^{(a,b,c)}_{(a,b-1,c)} L^{(a,b-1,c)}_{(a,b,c)}.$$

Note that the entries of the lowering operators can be determined by the recurrences in Corollary A.4, and they are sparse. In particular, they have block-bandwidths $(1,0)$ and diagonal blocks for $L^{(a-1,b,c)}_{(a,b,c)}$ and lower bidiagonal blocks for $L^{(a,b-1,c)}_{(a,b,c)}$ and $L^{(a,b,c-1)}_{(a,b,c)}$. This block structure ensures that J_x is block-tridiagonal with diagonal blocks and that J_y is block-tridiagonal with tridiagonal blocks. Finally, J_x and J_y commute because the L and S operators commute due to the following relationships:

(3.1)
$$S^{(a,b,c)}_{(a-1,b,c)} L^{(a-1,b,c)}_{(a,b,c)} = L^{(a+1,b,c)}_{(a,b,c)} S^{(a,b,c)}_{(a+1,b,c)}, \quad S^{(a,b,c)}_{(a,b-1,c)} L^{(a,b-1,c)}_{(a,b,c)} = L^{(a,b+1,c)}_{(a,b,c)} S^{(a,b,c)}_{(a,b+1,c)}.$$

3.3. Implementation of Clenshaw's algorithm and multiplication operators. We now exploit the sparsity structure of J_x and J_y to get an $\mathcal{O}(N^2)$ complexity Clenshaw algorithm. In particular, using the notation of subsection 2.3, since B^x_n is diagonal and B^y_n is tridiagonal we can construct a simple left-inverse B^+_n, that is, we

have the following structure:

$$B_n = \begin{pmatrix} B_n^x \\ B_n^y \end{pmatrix} = \begin{pmatrix} \times & & & & & & \\ & \times & & & & & \\ & & \ddots & & & & \\ & & & \times & & & \\ & & & & \times & 0 & \\ \times & \times & & & & & \\ \times & \times & \times & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \times & \times & \times & & \\ & & & \times & \times & \times & \end{pmatrix}.$$

Let $B_1 = B_n^x[0:n, 0:n]$ denote the first $(n+1) \times (n+1)$ subblock of B_n^x, let $b_2 = B_n^y[n, n+1]$, and let

$$\boldsymbol{b}_1^\top = -b_2^{-1}\left(\mathbf{0}_{1\times n-2}, \frac{B_n^y[n, n-1]}{B_n^x[n-1, n-1]}, \frac{B_n^y[n, n]}{B_n^x[n, n]}\right).$$

Then, the following matrix is a pseudo-inverse of B_n:

$$B_n^+ := \begin{pmatrix} B_1^{-1} & \mathbf{0}_{n\times n-1} & \mathbf{0}_{n\times 1} \\ \boldsymbol{b}_1^\top & \mathbf{0}_{1\times n-1} & b_2^{-1} \end{pmatrix}.$$

Note that B_n^+ can be applied to a vector in $\mathcal{O}(n)$ operations. When incorporation into Clenshaw's algorithm described in section 2.3, this gives an optimal $\mathcal{O}(N^2)$ algorithm for evaluating functions. Furthermore, when incorporated into the construction of the multiplication operators (see section 2.4), we find that one can construct multiplication operators in $\mathcal{O}(N^2)$ operations.

3.4. Differentiation. Jacobi polynomials on the interval have banded recurrence relationships for their derivatives by incrementing both of the parameters, that is, we can represent

$$f'(x) = \mathbf{P}^{(a+1,b+1)}(x)^\top D_{(a,b)}^{(a+1,b+1)} \boldsymbol{f},$$

where $D_{(a,b)}^{(a+1,b+1)}$ is zero except for the first superdiagonal [15, (18.9.15)]. They also have banded recurrence relationship for their weighted derivatives that decrement the parameters:

$$\frac{\mathrm{d}}{\mathrm{d}x}[x^a(1-x)^b f(x)] = x^{a-1}(1-x)^{b-1}\mathbf{P}^{(a-1,b-1)}(x)^\top W_{(a,b)}^{(a-1,b-1)} \begin{pmatrix} f_0 \\ f_1 \\ \vdots \end{pmatrix},$$

where $W_{(a,b)}^{(a-1,b-1)}$ is zero except for the first subdiagonal [15, (18.9.16)].

These properties translate to partial derivatives of Jacobi polynomials on the triangle, that is, we have[4]

$$\begin{aligned} \frac{\partial f}{\partial x} &= \mathbf{P}^{(a+1,b,c+1)}(x,y)^\top D_{x,(a,b,c)}^{(a+1,b,c+1)} \boldsymbol{f}, \\ \frac{\partial f}{\partial y} &= \mathbf{P}^{(a,b+1,c+1)}(x,y)^\top D_{y,(a,b,c)}^{(a,b+1,c+1)} \boldsymbol{f}, \end{aligned}$$

[4]We have similar relationships for $\frac{\partial}{\partial z} := \frac{\partial}{\partial x} - \frac{\partial}{\partial y}$, but we omit these for brevity as they are not needed.

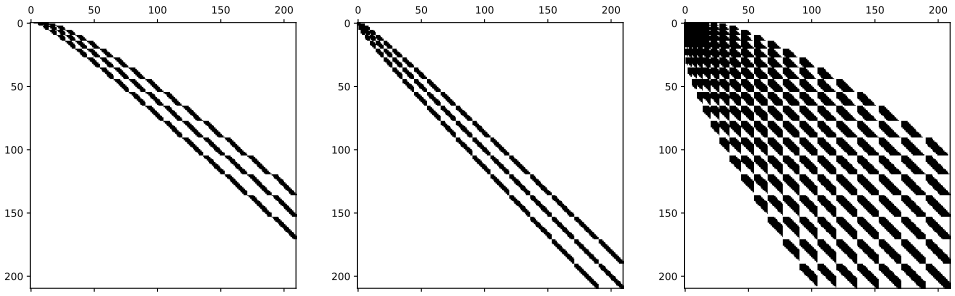

FIG. 2. *The sparsity pattern of the Laplacian* $\Delta^{(2,2,2)}_{(0,0,0)}$ *(left), the weighted Laplacian* Δ_W *(center), and the weighted variable coefficient Helmoltz operator* $\Delta_W + S^{(1,1,1)}_{(0,0,0)} v(J_x^\top, J_y^\top) L^{(0,0,0)}_{(1,1,1)}$ *with* $V(x,y) = xy^2$ *(right).*

where the entries are derived in Corollary A.1. Both $D^{(a+1,b,c+1)}_{x,(a,b,c)}$ and $D^{(a,b+1,c+1)}_{y,(a,b,c)}$ are sparse: they are block-superdiagonal, and their blocks are upper bidiagonal and superdiagonal, respectively. Similarly, for weighted differentiation we have

$$\begin{aligned}\tfrac{\partial}{\partial x}[x^a y^b z^c f(x,y)] &= x^{a-1} y^b z^{c-1} \mathbf{P}^{(a-1,b,c-1)}(x,y)^\top W^{(a-1,b,c-1)}_{x,(a,b,c)} \boldsymbol{f},\\ \tfrac{\partial}{\partial y}[x^a y^b z^c f(x,y)] &= x^a y^{b-1} z^{c-1} \mathbf{P}^{(a,b-1,c-1)}(x,y)^\top W^{(a,b-1,c-1)}_{y,(a,b,c)} \boldsymbol{f},\end{aligned}$$

where the entries are derived in Corollary A.2. Both $W^{(a-1,b,c-1)}_{x,(a,b,c)}$ and $W^{(a,b-1,c-1)}_{y,(a,b,c)}$ are also sparse matrices as they are block-subdiagonal, and their blocks are lower bidiagonal and subdiagonal, respectively.

Combining differentiation and conversion appropriately allows us to represent more complicated differential operators. For example, the Laplacian can be expressed as an operator that takes coefficients in an $\mathbf{P}^{(0,0,0)}$ expansion to coefficients in an $\mathbf{P}^{(2,2,2)}$ expansion as follows:

$$\Delta^{(2,2,2)}_{(0,0,0)} := S^{(2,2,2)}_{(2,1,2)} S^{(2,1,2)}_{(2,0,2)} D^{(2,0,2)}_{x,(1,0,1)} D^{(1,0,1)}_{x,(0,0,0)} + S^{(2,2,2)}_{(1,2,2)} S^{(1,2,2)}_{(0,2,2)} D^{(0,2,2)}_{y,(0,1,1)} D^{(0,1,1)}_{y,(0,0,0)}.$$

A simple calculation determines that this is also a sparse operator with block-bandwidths $(2,4)$ and blocks with bandwidths $(0,4)$; see Figure 2 (left).

Similarly, we can express the Laplacian as an operator from coefficients in an $xy(1-x-y)\mathbf{P}^{(1,1,1)}(x,y)$ expansion to coefficients in a $\mathbf{P}^{(1,1,1)}(x,y)$ expansion by using weighted derivatives and lowering operators:

$$\Delta_W := S^{(1,1,1)}_{(1,0,1)} D^{(1,0,1)}_{x,(0,0,0)} L^{(0,0,0)}_{(0,1,0)} W^{(0,1,0)}_{x,(1,1,1)} + S^{(1,1,1)}_{(0,1,1)} D^{(0,1,1)}_{y,(0,0,0)} L^{(0,0,0)}_{(1,0,0)} W^{(1,0,0)}_{y,(1,1,1)}. \tag{3.2}$$

This is a sparse operator with block-bandwidths $(1,2)$ and blocks with bandwidths $(2,2)$; see Figure 2 (center).

Finally, variable coefficients can be constructed by combining lowering, conversion, and Jacobi operators. For example, the variable Helmholtz operator $\Delta + v(x,y)$ can be represented as

$$\Delta_W + S^{(1,1,1)}_{(0,0,0)} v(J_x^\top, J_y^\top) L^{(0,0,0)}_{(1,1,1)}.$$

This still leads to a sparse discretization, where the block-bandwidths depend on the degree of v; see Figure 2 (right) for an example with $v(x,y) + xy^2$.

4. Solving linear PDEs with zero Dirichlet conditions. We now use the systematic approach to constructing sparse operators to solve PDEs. We construct the operators using BlockBandedMatrices.jl [16], which enables fast multiplication of block-banded matrices with banded blocks by building on BLAS. We then convert the representation to a SuiteSparse compatible sparse matrix format, for matrix factorization and solves.

4.1. Zero Dirichlet conditions. To solve PDEs with vanishing Dirichlet conditions, we use the weighted basis

$$xy(1-x-y)\mathbf{P}^{(1,1,1)}(x,y).$$

For higher-order equations like the biharmonic equation we consider vanishing Dirichlet and Neumann conditions using the weighted basis

$$x^2y^2(1-x-y)^2\mathbf{P}^{(2,2,2)}(x,y).$$

4.1.1. Example 1: Poisson equation. Consider Poisson's equation on a triangle with zero Dirichlet conditions, i.e.,

$$u_{xx}+u_{yy}=f(x,y), \quad (x,y)\in T, \qquad u|_{\partial T}=0.$$

We reduce this equation to a truncation of

$$\Delta_W \boldsymbol{u}=\boldsymbol{f},$$

where the coefficients of $f(x,y)=\mathbf{P}^{(1,1,1)}(x,y)^\top \boldsymbol{f}$ are determined using [22] as implemented in [24]. In Figure 3 (left), we depict the solution for a specific choice of $f(x,y)$. In Figure 3 (right), we show the construction time[5] of the matrix (using BlockBandedMatrices.jl [16], execution time for an LU factorization, and the solve time (using SuiteSparse via Julia 1.1's SparseArrays.jl). We observe that the construction (roughly) requires an optimal $\mathcal{O}(N^2)$ operations, while the factorization and solution time are observed to (roughly) cost an almost-optimal $\mathcal{O}(N^3)$ operations. The same complexities are observed for PDEs on rectangles using a Chebyshev-based spectral method [10].

In Figure 4, we show the norms of each block of calculated coefficients of the approximation for four right-hand sides with $N=999$. Note that the rate of decay in the coefficients is a proxy for the rate of convergence of the computed solution. The behavior of the right-hand side at the corners has an impact on the convergence rate; in particular, if f and its derivatives vanish at the corners, then we observe faster convergence of the solution. The behavior at the origin is particularly important as the Laplacian of the basis $xy(1-x-y)P_{n,k}^{(1,1,1)}(x,y)$ always vanishes at the origin. While we only observe algebraic convergence for the first three examples (that is, we do not achieve spectral convergence as $N\to\infty$), the rate of convergence is fairly fast, achieving machine precision accuracy when $f(x,y)$ vanishes at the origin with around 10,000 unknowns. Furthermore, the last example shows spectral convergence for a Gaussian bump function, which up to machine precision vanishes to all orders at the corners. Finally, over-resolving the solution does not result in the error plateauing at machine precision, which means our discretization slightly improves the regularity of the data, similar to the ODE case in [18].

[5]Timings are performed on an iMac 2017 with a 4 core 3.8 GHz Intel Core i5, using Julia v1.1 compiled with MKL BLAS. Note that the default OpenBLAS is slower for banded matrix operations. Both MKL BLAS and SuiteSparse use 4 threads.

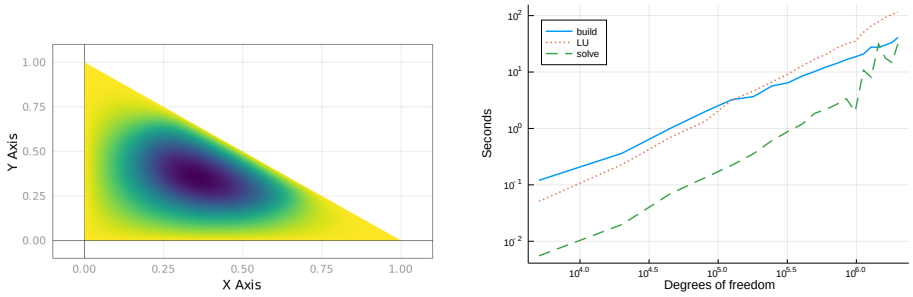

FIG. 3. *Left: The computed solution to* $\Delta u = f$ *with zero boundary conditions and* $f(x,y) = 1 + \operatorname{erf}(5(1 - 10((x-1/2)^2 + (y-1/2)^2)))$. *Right: the time in seconds to build the discretization, calculate its LU factorization using SuiteSparse, and solve the system.*

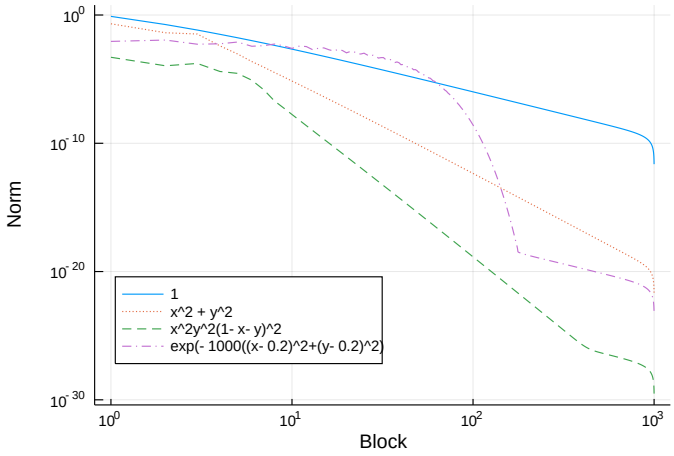

FIG. 4. *The norm of the blocks of the calculated coefficients for four functions, for* $N = 1000$, *i.e., with* $500k$ *degrees of freedom. The rate in decay serves as a proxy for the error in the computed solution. We see that in the first three cases we have algebraic convergence, with the convergence rate improving when the function vanishes to higher order at the corners. The last example shows spectral convergence for a Gaussian bump.*

4.1.2. Example 2: Variable coefficient Helmholtz equation with forcing terms. Now, consider a variable coefficient Helmholtz equation with zero Dirichlet conditions, i.e.,

$$u_{xx} + u_{yy} + k^2 v(x,y)u = xye^x, \quad (x,y) \in T, \qquad u|_{\partial T} = 0.$$

We first approximate $v(x,y)$ by a polynomial [24] and then use the operator-valued Clenshaw's algorithm to construct $v(J_x^\top, J_y^\top)$. We obtain the following discretization:

$$\Delta_W + k^2 S^{(1,1,1)}_{(0,0,0)} v(J_x^\top, J_y^\top) L^{(0,0,0)}_{(1,1,1)},$$

where $J_x^\top$ and $J_y^\top$ are the Jacobi operators for $\mathbf{P}^{(1,1,1)}$ and

$$L^{(0,0,0)}_{(1,1,1)} = L^{(0,0,0)}_{(0,0,1)} L^{(0,0,1)}_{(0,1,1)} L^{(0,1,1)}_{(1,1,1)}, \qquad S^{(1,1,1)}_{(0,0,0)} = S^{(1,1,1)}_{(0,1,1)} S^{(0,1,1)}_{(0,0,1)} S^{(0,0,1)}_{(0,0,0)}.$$

In Figure 5 we depict the solution for $k = 100$ and plot the timings for construction, factorization, and solution for k between 100 and 300, using polynomials of

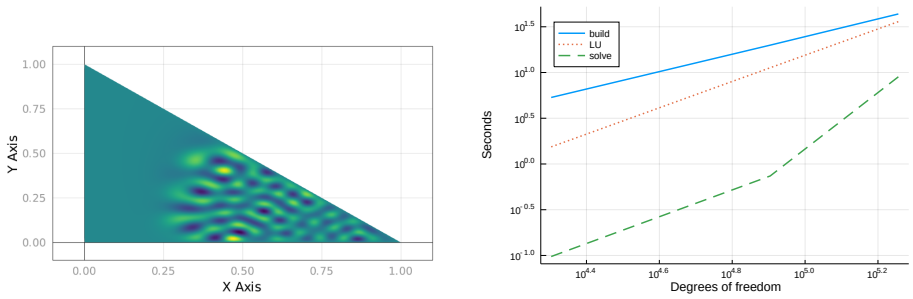

FIG. 5. *Left: The computed solution to* $(\Delta + k^2 v(x,y))u = xye^x$ *with zero Dirichlet conditions and* $v(x,y) = 1 - (3(x-1)^2 + 5y^2)$. *Right: The execution time to build the discretization, calculate its LU factorization using SuiteSparse, and solve the linear system.*

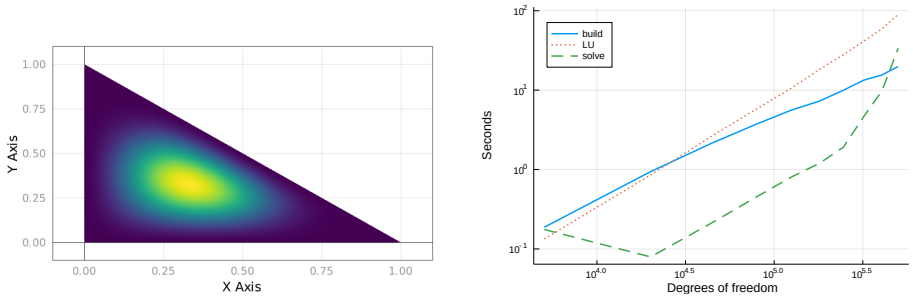

FIG. 6. *Left: The solution to* $\Delta^2 u = f$ *with zero Dirichlet conditions and* $f(x,y) = 1 + \mathrm{erf}\,(5(1 - 10((x-1/2)^2 + (y-1/2)^2)))$. *Right: The execution time to build the discretization, calculate its LU factorization using SuiteSparse, and solve the linear system.*

degree $2k$. The build time depends only on the discretization size, so we observe an $\mathcal{O}(k^2)$ cost.

4.1.3. Example 3: The biharmonic equation. The same technique for constructing a sparse representation of the Laplacian Δ translates to the biharmonic operator Δ^2, though now we must use a basis that satisfies both zero Dirichlet and Neumann conditions. We can represent the Laplacian as a map from coefficients in an $x^2y^2(1-x-y)^2\mathbf{P}^{(2,2,2)}(x,y)$ expansion to coefficients in an $\mathbf{P}^{(0,0,0)}$ expansion by using weighted differentiation and lowering operators:

$$\Delta_{W^2} := L^{(0,0,0)}_{(0,1,0)} W^{(0,1,0)}_{x,(1,1,1)} L^{(1,1,1)}_{(1,2,1)} W^{(1,2,1)}_{x,(2,2,2)} + L^{(0,0,0)}_{(1,0,0)} W^{(1,0,0)}_{y,(1,1,1)} L^{(1,1,1)}_{(2,1,1)} W^{(2,1,1)}_{y,(2,2,2)}.$$

Hence, the biharmonic operator can be sparsely represented as a map from coefficients in an $x^2y^2(1-x-y)^2\mathbf{P}^{(2,2,2)}(x,y)$ expansion to coefficients in an $\mathbf{P}^{(2,2,2)}$ expansion. This is simply given by $\Delta^{(2,2,2)}_{(0,0,0)}\Delta_{W^2}$.

In Figure 6 we depict a solution to the biharmonic equation and show that the build time grows linearly with respect to the number of degrees of freedom employed to discretize the solution.

5. Nonzero Dirichlet conditions. To handle general nonzero Dirichlet boundary conditions, we wish to construct restriction operators that are sparse operators. To facilitate this, we use a basis where most elements of the basis vanish at the boundary. We take the weighted basis $x^a y^b (1-x-y)^c \mathbf{P}^{(a,b,c)}$, where a, b, c are integers, and

augment it with additional polynomials so that the basis can represent all bivariate polynomials. This is essentially the same procedure as in [11], but we do it in a way that preserves the sparsity of the restriction operators. Appendix B gives the definition of $Q_{n,k}^{(a,b,c)}(x,y)$, where $a,b,c \in \{0,1\}$, which is the basis we use to construct sparse discretizations. Here, most of $Q_{n,k}^{(1,b,c)}(0,y)$, most of $Q_{n,k}^{(a,1,c)}(x,0)$, and most of $Q_{n,k}^{(a,b,1)}(x,1-x)$ vanish.

Remark 2. Formally, $Q_{n,k}^{(a,b,c)}(x,y)$ can be thought of as $P_{n,k}^{(-a,-b,-c)}(x,y)$, which is made precise in [32] during the construction of the polynomials $J_{n,k}^{(a,b,c)}(x,y)$. However, the construction in [32] is normalized in a way that leads to underflow in double precision computing, and we find it simpler to define our own basis $Q_{n,k}^{(a,b,c)}(x,y)$ in an ad hoc way.

5.1. Derivative and conversion operators. Partial derivatives and conversion operators $\boldsymbol{Q}^{(a,b,c)}(x,y)$ are similar to those derived for $\boldsymbol{P}^{(a,b,c)}(x,y)$. Using the formulas in Corollary C.1, we can construct conversion operators that convert from one-edge bases to $\boldsymbol{P}^{(0,0,0)}$:

$$\begin{aligned} f(x,y) &= \boldsymbol{Q}^{(1,0,0)}(x,y)^\top \boldsymbol{f} = \boldsymbol{P}^{(0,0,0)}(x,y)^\top \tilde{S}_{(1,0,0)}^{(0,0,0)} \boldsymbol{f}, \\ f(x,y) &= \boldsymbol{Q}^{(0,1,0)}(x,y)^\top \boldsymbol{f} = \boldsymbol{P}^{(0,0,0)}(x,y)^\top \tilde{S}_{(0,1,0)}^{(0,0,0)} \boldsymbol{f}, \\ f(x,y) &= \boldsymbol{Q}^{(0,0,1)}(x,y)^\top \boldsymbol{f} = \boldsymbol{P}^{(0,0,0)}(x,y)^\top \tilde{S}_{(0,0,1)}^{(0,0,0)} \boldsymbol{f}. \end{aligned}$$

Note that each operator is block-upper bidiagonal, with diagonal or upper bidiagonal blocks. Similarly, Corollary C.2 derives sparse conversion operators from two-edge bases to one-edge bases, which we denote by $\tilde{S}_{(1,1,0)}^{(1,0,0)}$, $\tilde{S}_{(1,1,0)}^{(0,1,0)}$, $\tilde{S}_{(1,0,1)}^{(1,0,0)}$, etc. Finally, Corollary C.3 derives sparse conversion operators from the three-edge basis to any of the two-edge bases, which we denote by $\tilde{S}_{(1,1,1)}^{(1,1,0)}$, $\tilde{S}_{(1,1,1)}^{(1,0,1)}$, and $\tilde{S}_{(1,1,1)}^{(0,1,1)}$. We can clearly compose these operators together to convert from, say, $\boldsymbol{Q}^{(1,1,1)}$ to $\boldsymbol{P}^{(0,0,0)}$. For this purpose, we can define

$$\tilde{S}_{(1,1,1)}^{(0,0,0)} := \tilde{S}_{(1,0,0)}^{(0,0,0)} \tilde{S}_{(1,1,0)}^{(1,0,0)} \tilde{S}_{(1,1,1)}^{(1,1,0)}.$$

There are always several paths through the parameter space to convert one basis into another; however, any path that is chosen results in the same final conversion operator.

The same principle is true for derivatives, though for our purposes it suffices to restrict our attention to derivatives of the two-edge bases. Corollary C.4 gives us the entries for sparse (block-superdiagonal with at most bidiagonal blocks) operators that satisfy

$$\begin{aligned} \frac{\partial f}{\partial x} &= \frac{\partial}{\partial x} \boldsymbol{Q}^{(1,0,1)}(x,y)^\top \boldsymbol{f} = \mathbf{P}^{(0,0,0)}(x,y)^\top \tilde{D}_{x,(1,0,1)}^{(0,0,0)} \boldsymbol{f}, \\ \frac{\partial f}{\partial y} &= \frac{\partial}{\partial y} \boldsymbol{Q}^{(0,1,1)}(x,y)^\top \boldsymbol{f} = \mathbf{P}^{(0,0,0)}(x,y)^\top \tilde{D}_{y,(0,1,1)}^{(0,0,0)} \boldsymbol{f}. \end{aligned}$$

General partial derivative operators can be constructed by combining conversion and derivative operators. For example, we can successfully construct the Laplacian from $\boldsymbol{Q}^{(1,1,1)}$ to $\boldsymbol{P}^{(1,1,1)}$ as

$$\tilde{\Delta} := S_{(1,0,1)}^{(1,1,1)} D_{x,(0,0,0)}^{(1,0,1)} \tilde{D}_{x,(1,0,1)}^{(0,0,0)} \tilde{S}_{(1,1,1)}^{(1,0,1)} + S_{(0,1,1)}^{(1,1,1)} D_{y,(0,0,0)}^{(0,1,1)} \tilde{D}_{y,(0,1,1)}^{(0,0,0)} \tilde{S}_{(1,1,1)}^{(0,1,1)}.$$

This is a sparse operator with block-bandwidths and sub-blockbandwidths equal to $(1, 4)$.

5.2. Restriction operators. The definitions of $Q_{n,k}^{(1,0,0)}(x, y)$, $Q_{n,k}^{(0,1,0)}(x, y)$, and $Q_{n,k}^{(0,0,1)}(x, y)$ each have a simple restriction formula to one of the three edges of the triangle:

$$\begin{aligned}
Q_{n,n}^{(1,0,0)}(0, y) &= \tilde{P}_n(y) \qquad \text{and} \qquad Q_{n,k}^{(1,0,0)}(0, y) = 0 \quad \text{for} \quad k = 0, \dots, n-1,\\
Q_{n,0}^{(0,1,0)}(x, 0) &= \tilde{P}_n(x) \qquad \text{and} \qquad Q_{n,k}^{(0,1,0)}(x, 0) = 0 \quad \text{for} \quad k = 1, \dots, n,\\
Q_{n,0}^{(0,0,1)}(x, 1-x) &= \tilde{P}_n(x) \qquad \text{and} \qquad Q_{n,k}^{(0,0,1)}(x, 1-x) = 0 \quad \text{for} \quad k = 1, \dots, n.
\end{aligned}$$

In other words, the restriction operator from expansion in $Q_{n,k}^{(1,0,0)}(x, y)$ to Legendre expansion on the edge from $(0, 0)$ to $(0, 1)$ is a block-banded operator, where the blocks themselves are very sparse: each block has precisely one nonzero entry. Similarly, the other two bases give restriction operators to the other edges. We denote these restriction operators as R_x, R_y, and R_z, respectively. They are given by

$$\begin{aligned}
f(0, y) &= \boldsymbol{Q}^{(1,0,0)}(0, y)^\top \boldsymbol{f} = \boldsymbol{P}(y)^\top R_x \boldsymbol{f},\\
f(x, 0) &= \boldsymbol{Q}^{(0,1,0)}(0, y)^\top \boldsymbol{f} = \boldsymbol{P}(x)^\top R_y \boldsymbol{f},\\
f(x, 1-x) &= \boldsymbol{Q}^{(0,0,1)}(x, 1-x)^\top \boldsymbol{f} = \boldsymbol{P}(x)^\top R_z \boldsymbol{f},
\end{aligned}$$

where $\boldsymbol{P}(x) := \mathbf{P}^{(0,0)}(x)^\top$ are the shifted Legendre polynomials.

For the full Dirichlet operator, we need to restrict to all three edges. Thus we can construct restriction operators from $\boldsymbol{Q}^{(1,1,1)}$ to the boundary, where the boundary bases are piecewise mapped Legendre polynomials. This restriction operator can be calculated by combining conversion and the one-edge restrictions as follows:

$$R := \begin{pmatrix} R_x \tilde{S}_{(1,1,0)}^{(1,0,0)} \tilde{S}_{(1,1,1)}^{(1,1,0)} \\ R_y \tilde{S}_{(1,1,0)}^{(0,1,0)} \tilde{S}_{(1,1,1)}^{(1,1,0)} \\ R_z \tilde{S}_{(1,0,1)}^{(0,0,1)} \tilde{S}_{(1,1,1)}^{(1,0,1)} \end{pmatrix}.$$

This operator is also sparse as each component is a product of sparse operators.

5.3. The τ-method. An issue we must deal with is boundary data with discontinuities at the corners. Consider, for example, the Laplace equation with Dirchlet conditions:

$$\Delta u = 0, \qquad u|_{x=0} = f, \qquad u|_{x=0} = g, \qquad \text{and} \qquad u|_{z=0} = h.$$

If the boundary data has discontinuities, that is, $f(0, 0) \neq g(0, 0)$, $f(0, 1) \neq h(0, 1)$, or $g(1, 0) \neq h(1, 0)$, then the solution itself will have an arg-like singularity: e.g., near the origin we have the local behavior

$$u(x, y) \sim (g(0, 0) - f(0, 0)) \frac{2}{\pi} \arg(x + \mathrm{i}y) + f(0, 0).$$

Approximating $u(x, y)$ by polynomials is, therefore, limited as they impose continuity. Other PDEs like the Helmholtz equation have similar behavior when the boundary data has discontinuities.

To overcome this issue, we adapt the Lanczos τ-method; see [20] for an overview. The Lanczos τ-method is a device to produce invertible systems for polynomial spectral methods by augmenting the equations with polynomial correction terms, which also provide error control by measuring the magnitude of the correction term. In our context we use it to capture discontinuities by augmenting the boundary data with corrections of the form

$$\Delta u = 0, \qquad u|_{x=0} = f + \tau_1, \qquad u|_{y=0} = g + \tau_2, \qquad \text{and} \qquad u|_{z=0} = h.$$

That is, we add constants τ_1 and τ_2 to our discretization:

$$\begin{pmatrix} \boldsymbol{e}_0 & 0 & R_x \tilde{S}^{(1,0,0)}_{(1,1,0)} \tilde{S}^{(1,1,0)}_{(1,1,1)} \\ 0 & \boldsymbol{e}_0 & R_y \tilde{S}^{(0,1,0)}_{(1,1,0)} \tilde{S}^{(1,1,0)}_{(1,1,1)} \\ 0 & 0 & R_z \tilde{S}^{(0,0,1)}_{(1,0,1)} \tilde{S}^{(1,0,1)}_{(1,1,1)} \\ 0 & 0 & \tilde{\Delta} \end{pmatrix} \begin{pmatrix} \tau_1 \\ \tau_2 \\ \boldsymbol{u} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{g} \\ \boldsymbol{h} \\ 0 \end{pmatrix}.$$

Now, in our examples below we actually have mathematically continuous boundary data; however, round-off errors mean our boundary data is slightly discontinuous. The τ correction terms give a way of capturing this discontinuity without destroying the regularity of u. When the solution is resolved the τ terms are, therefore, negligible, and we can use the approximation of u on its own.

Note that it is possible to add additional τ correction terms to make the system invertible, but this is a more technical task and hence we prefer to use a QR decomposition to solve the resulting rectangular linear system in a least squares sense. This does incur a substantial performance penalty, as SuiteSparse's QR decomposition is significantly slower than its LU decomposition.

5.3.1. Example 4: Laplace's equation. Consider Laplace's equation with prescribed Dirichlet data:

$$u_{xx} + u_{yy} = 0, \qquad u(0,y) = f(y), \quad u(x,0) = g(x), \quad u(x,1-x) = h(x).$$

Expanding $f(x)$, $g(y)$, and $h(x)$ in Legendre series leads to a system of equations satisfied by u, that is,

$$\begin{pmatrix} R \\ \tilde{\Delta} \end{pmatrix} \boldsymbol{u} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{g} \\ \boldsymbol{h} \\ 0 \end{pmatrix},$$

where $\boldsymbol{u}$ are the coefficients of $u(x,y)$ in the basis $\boldsymbol{Q}^{(1,1,1)}$, $\boldsymbol{f}$ are the Legendre coefficients of $f(y)$, $\boldsymbol{g}$ are the Legendre coefficients of $g(x)$, and $\boldsymbol{h}$ are the Legendre coefficients of $h(x)$. We augment this system with τ corrections, which are ultimately ignored in the approximation of the solution.

In Figure 7 we plot the calculated coefficients for $N = 999$ for three choices of boundary data: $\mathrm{e}^x \cos y$, x^2, and $x^3(1-x)^3(1-y)^3$. The first two examples exhibit algebraic decay, with the rate of decay dictated by the number of derivatives matching at the corners. The last example has a smooth solution ($\mathrm{e}^x \cos y$ is harmonic), and we see that the algorithm achieves superalgebraic convergence and is stable for large N. We also note that evaluating the approximation is exact to within an accuracy of 3×10^{-16} compared to the exact solution at the arbitrary point $(x,y) = (0.1, 0.2)$.

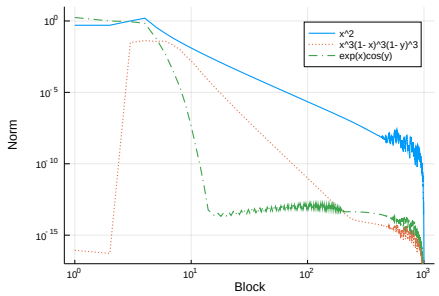

FIG. 7. *The norm of the blocks of the calculated coefficients to the solution of* $\Delta u = 0$ *with specified Dirichlet boundary conditions with* $N = 1000$. *The first two examples show algebraic convergence, with faster convergence when there is more continuity at the corners. The third example shows spectral convergence when the solution is smooth.*

5.3.2. Example 5: Transport equation. Nothing in this framework depends on the PDE being elliptic. Here, we consider the transport equation given by

$$u_y = cu_x.$$

Information travels at a rate and direction dictated by c, and depending on its value we need either one or two edges to uniquely determine the solution. If $0 \le c \le 1$, the solution is uniquely determined from the boundary on the bottom, and hence we use the basis $Q^{(0,1,0)}$. If $c > 1$, then information is coming in from the right, so we use the basis $Q^{(0,1,1)}$ on the bottom and hypotenuse edges. If $c < 0$, then information comes in from the left and we use the basis $Q^{(1,1,0)}$ on the bottom and left edges. In Figure 8 we depict the three solutions, using degree $N = 100$ polynomial approximations.

6. Systems of PDEs. Systems of PDEs can be handled in a straightforward way by concatenating their blocks. As an example, we can solve the Poisson equation with Neumann conditions by re-expressing the PDE as a first-order system: writing $v = u_x$ expressed in the basis $\boldsymbol{Q}^{(1,0,1)}$, and $w = u_y$ expressed in the basis $\boldsymbol{Q}^{(0,1,1)}$, the system becomes

$$\begin{pmatrix} 0 & -R_x \tilde{S}^{(1,0,0)}_{(1,0,1)} & 0 \\ 0 & 0 & -R_y \tilde{S}^{(0,1,0)}_{(0,1,1)} \\ 0 & R_z \tilde{S}^{(0,0,1)}_{(1,0,1)} & R_z \tilde{S}^{(0,0,1)}_{(0,1,1)} \\ D^{(1,0,1)}_{x,(0,0,0)} & -S^{(1,0,1)}_{(0,0,0)} \tilde{S}^{(0,0,0)}_{(1,0,1)} & 0 \\ D^{(0,1,1)}_{y,(0,0,1)} & 0 & -S^{(0,1,1)}_{(0,0,0)} \tilde{S}^{(0,0,0)}_{(0,1,1)} \\ 0 & \tilde{D}^{(0,0,0)}_{x,(1,0,1)} & \tilde{D}^{(0,0,0)}_{y,(0,1,1)} \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \boldsymbol{f} \end{pmatrix}.$$

6.1. Example 6: Helmholtz equation in a polygon. Note that being able to handle systems of PDEs in this manner also allows us to solve on polygonal domains that are partitioned into triangular elements. For example, consider the Helmholtz equation

$$u_{xx} + u_{yy} + k^2 u = 0$$

on the polygonal domain with the vertices $(0,0)$, $(1,0)$, $(1,1)$, $(0,2)$, $(0,1)$, and $(-1,1.5)$. We can decompose this domain into four triangles and represent the solution as well as its first derivatives in orthogonal polynomial expansions. This leads

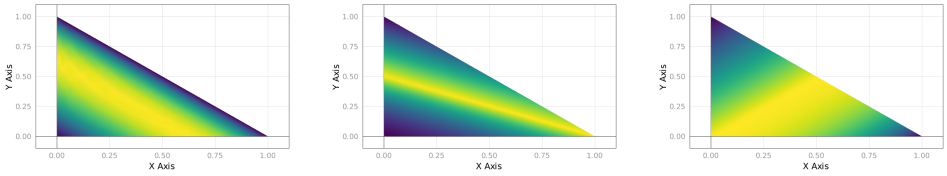

FIG. 8. *Left: The solution to* $u_y = u_x$ *using the basis* $\boldsymbol{Q}^{(0,1,0)}$ *with boundary condition* $u(x,0) = x(1-x)\mathrm{e}^x$ *imposed on the bottom. Middle: The solution to* $u_y = 2u_x$ *using the basis* $\boldsymbol{Q}^{(0,1,1)}$ *with boundary condition* $u(x,0) = x\mathrm{e}^{x-1}$ *imposed on the bottom and* $u(x,1-x) = x$ *on the hypotenuse. Right: The solution to* $u_y = -u_x$ *using the basis* $\boldsymbol{Q}^{(1,1,0)}$ *with boundary condition* $u(x,0) = (1-x)\mathrm{e}^x$ *imposed on the bottom and* $u(0,y) = 1-y$ *on the left.*

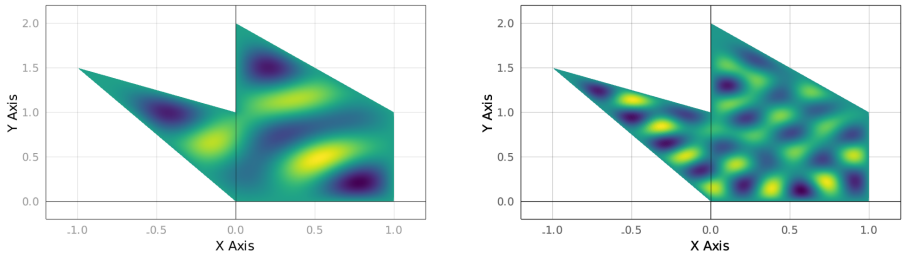

FIG. 9. *The solution to* $(\Delta^2 + k^2)u = 0$ *with Dirichlet boundary conditions fixed to one, for* $k = 10$ *(left) and* $k = 20$ *(right), using degree* $N = 100$ *polynomials in each triangle.*

to a system of $4 \times 3 = 12$ PDEs. We then impose continuity of the value and the normal derivative across the interfaces of each element, exploiting the fact that the restriction operator maps to the same basis of Legendre polynomials. (The orientation may be different, but reversing orientation of Legendre expansions corresponds to multiplying by a diagonal matrix that swaps the signs of every other coefficient.) The discretization of the PDE system is sparse, and the complexity of building the matrices is an optimal $\mathcal{O}(N^2)$ using degree N polynomials within each element. We show the success of this approach in Figure 9 for $k = 10$ (left) and $k = 20$ (right), using degree $N = 100$ polynomials.[6]

7. Conclusions. We have shown that bivariate orthogonal polynomials can lead to sparse discretizations of general linear PDEs on triangles with Dirichlet and Neumann boundary conditions. Instead of quadrature, we use sparse recurrence relationships combined with specialized linear algebra routines, allowing optimal complexity for building the linear systems. Multiple triangles can be patched together to solve PDEs on polygonal domains.

An extension is to tetrahedra in three dimensions and higher. We expect this to be straightforward because the definitions of orthogonal polynomials on higher dimensional simplices is very similar to the two-dimensional (2D) case. In three dimensions, we can use the following polynomials:

$$P^{(a,b,c,d)}_{n,k,j}(x,y,z) := P^{(a,b,2j+c+d+1)}_{n-j,k}(x,y)(1-x-y)^j P^{(d,c)}_j\left(\frac{z}{1-x-y}\right),$$

[6]The implementation is currently unoptimized, and it takes 90.5s to build the sparse 63,024 × 61,812 matrix, 2.5s to calculate its QR factorization, and 0.7s to solve the system using the factorization.

which are orthogonal with respect to $x^a y^b z^c (1-x-y-z)^d$ on the unit three-dimensional (3D) simplex. The most time-consuming part of such an extension is deriving the recurrences relationships. Note that in three dimensions and higher the sparsity of our construction is useful even for small discretization sizes, as a degree N dense discretization (e.g., arising from collocation) would require calculating $\mathcal{O}(N^6)$ entries, where the proposed construction would require an optimal $\mathcal{O}(N^3)$ operations.

We used direct solvers via SuiteSparse to solve the resulting discretizations, which is fairly efficient with even millions of unknowns. However, to push the methodology further we will need robust iterative methods and the development of preconditioners. It is not yet clear how to design preconditioners in this setting.

Finally, we mention that these techniques can be combined with polynomial mappings to solve PDEs on more exotic geometries. This combined with domain decomposition could provide an effective way of achieving high accuracy solution of PDEs on geometries with piecewise-smooth boundaries.

Appendix A. Recurrence relationships for Jacobi polynomials on the triangle. Here, we outline the recurrence relationships for $P_{n,k}^{(a,b,c)}(x,y)$ that we employ, which were previously derived in [19, 32]. We define $z := 1-x-y$ and $\frac{\partial}{\partial z} := \frac{\partial}{\partial y} - \frac{\partial}{\partial x}$.

COROLLARY A.1 (see [19, Corollary 1]). *The following recurrence relations for the partial derivatives hold:*

$$\begin{aligned}
(2k+b+c+1)\tfrac{\partial}{\partial x}P_{n,k}^{(a,b,c)} &= (n+k+a+b+c+2)(k+b+c+1)P_{n-1,k}^{(a+1,b,c+1)} \\
&\quad + (k+b)(n+k+b+c+1)P_{n-1,k-1}^{(a+1,b,c+1)}, \\
\tfrac{\partial}{\partial y}P_{n,k}^{(a,b,c)} &= (k+b+c+1)P_{n-1,k-1}^{(a,b+1,c+1)}, \\
(2k+b+c+1)\tfrac{\partial}{\partial z}P_{n,k}^{(a,b,c)} &= -(n+k+a+b+c+2)(k+b+c+1)P_{n-1,k}^{(a+1,b+1,c)} \\
&\quad + (k+c)(n+k+b+c+1)P_{n-1,k-1}^{(a+1,b+1,c)}.
\end{aligned}$$

COROLLARY A.2 (see [19, Corollary 2]). *The following recurrence relations for the weighted partial derivatives hold:*

$$\begin{aligned}
-(2k+b+c+1)\tfrac{\partial}{\partial x}\big(x^a y^b z^c P_{n,k}^{(a,b,c)}\big) &= x^{a-1}y^b z^{c-1}\Big((k+c)(n-k+1)P_{n+1,k}^{(a-1,b,c-1)} \\
&\quad + (k+1)(n-k+a)P_{n+1,k+1}^{(a-1,b,c-1)}\Big), \\
\tfrac{\partial}{\partial y}\big(x^a y^b z^c P_{n,k}^{(a,b,c)}\big) &= -(k+1)x^a y^{b-1} z^{c-1} P_{n+1,k+1}^{(a,b-1,c-1)}, \\
(2k+b+c+1)\tfrac{\partial}{\partial z}\big(x^a y^b z^c P_{n,k}^{(a,b,c)}\big) &= x^{a-1}y^{b-1} z^{c}\Big((k+b)(n-k+1)P_{n+1,k}^{(a-1,b-1,c)} \\
&\quad - (k+1)(n-k+a)P_{n+1,k+1}^{(a-1,b-1,c)}\Big).
\end{aligned}$$

COROLLARY A.3 (see [19, Corollary 3]). *The following recurrence relations for*

conversions hold:

$$\begin{aligned}
&(2n+a+b+c+2)P_{n,k}^{(a,b,c)}\\
&\qquad = (n+k+a+b+c+2)P_{n,k}^{(a+1,b,c)}\\
&\qquad +(n+k+b+c+1)P_{n-1,k}^{(a+1,b,c)},\\
&(2n+a+b+c+2)(2k+b+c+1)P_{n,k}^{(a,b,c)}\\
&\qquad = (n+k+a+b+c+2)(k+b+c+1)P_{n,k}^{(a,b+1,c)}\\
&\qquad -(n-k+a)(k+b+c+1)P_{n-1,k}^{(a,b+1,c)}\\
&\qquad +(k+c)(n+k+b+c+1)P_{n-1,k-1}^{(a,b+1,c)}\\
&\qquad -(k+c)(n-k+1)P_{n,k-1}^{(a,b+1,c)},\\
&(2n+a+b+c+2)(2k+b+c+1)P_{n,k}^{(a,b,c)}\\
&\qquad = (n+k+a+b+c+2)(k+b+c+1)P_{n,k}^{(a,b,c+1)}\\
&\qquad -(n-k+a)(k+b+c+1)P_{n-1,k}^{(a,b,c+1)}\\
&\qquad -(k+b)(n+k+b+c+1)P_{n-1,k-1}^{(a,b,c+1)}\\
&\qquad +(k+b)(n-k+1)P_{n,k-1}^{(a,b,c+1)}.
\end{aligned}$$

COROLLARY A.4 (see [19, Corollary 4]). *The following recurrence relations for lowering operators hold:*

$$(2n+a+b+c+2)xP_{n,k}^{(a,b,c)} = (n-k+a)P_{n,k}^{(a-1,b,c)} + (n-k+1)P_{n+1,k}^{(a-1,b,c)},$$

$$\begin{aligned}
&(2k+b+c+1)(2n+a+b+c+2)yP_{n,k}^{(a,b,c)}\\
&\qquad = (k+b)(n+k+b+c+1)P_{n,k}^{(a,b-1,c)}\\
&\qquad -(k+1)(n-k+a)P_{n,k+1}^{(a,b-1,c)}\\
&\qquad -(k+b)(n-k+1)P_{n+1,k}^{(a,b-1,c)}\\
&\qquad +(k+1)(n+k+a+b+c+2)P_{n+1,k+1}^{(a,b-1,c)},
\end{aligned}$$

$$\begin{aligned}
&(2k+b+c+1)(2n+a+b+c+2)zP_{n,k}^{(a,b,c)}\\
&\qquad = (k+c)(n+k+b+c+1)P_{n,k}^{(a,b,c-1)}\\
&\qquad +(k+1)(n-k+a)P_{n,k+1}^{(a,b,c-1)}\\
&\qquad -(k+c)(n-k+1)P_{n+1,k}^{(a,b,c-1)}\\
&\qquad -(k+1)(n+k+a+b+c+2)P_{n+1,k+1}^{(a,b,c-1)}.
\end{aligned}$$

Appendix B. Dirichlet basis definitions. Here, we define a basis, denoted by $Q_{n,k}^{(a,b,c)}(x,y)$, that we employ to impose general Dirichlet and Neumann boundary

conditions. We construct $Q_{n,k}^{(a,b,c)}(x,y)$ by augmenting the weighted basis

$$x^a y^b z^c P_{m,k}^{(a,b,c)}(x,y)$$

so that $Q_{0,0}^{(a,b,c)}, Q_{1,0}^{(a,b,c)}, \ldots, Q_{N,0}^{(a,b,c)}, \ldots, Q_{N,N}^{(a,b,c)}$ span all polynomials of degree less than or equal to N, where $a,b,c \in \{0,1\}$. Depending on the choice of a, b, and c, we obtain sparse restriction operators to one, two, or three edges of the triangle. We refer to this basis as the *Dirichlet basis* for its usefulness in solving PDEs with Dirichlet and Neumann boundary conditions.

B.1. One-edge Dirichlet basis.

DEFINITION B.1. *The following polynomials vanish at $x = 0$ apart from when $k = n$:*

$$\begin{aligned} Q_{n,k}^{(1,0,0)}(x,y) &:= x P_{n-1,k}^{(1,0,0)}(x,y) \qquad \text{for } k = 0, \ldots, n-1, \\ Q_{n,n}^{(1,0,0)}(x,y) &:= P_{n,n}(x,y). \end{aligned}$$

The following polynomials vanish at $y = 0$ apart from when $k = 0$:

$$\begin{aligned} Q_{n,0}^{(0,1,0)}(x,y) &:= \tilde{P}_n^{(0,0)}(x), \\ Q_{n,k}^{(0,1,0)}(x,y) &:= y P_{n-1,k-1}^{(0,1,0)}(x,y) \qquad \text{for } k = 1, \ldots, n. \end{aligned}$$

The following polynomials vanish at $z = 0$ (i.e., $y = 1 - x$) apart from when $k = 0$:

$$\begin{aligned} Q_{n,0}^{(0,0,1)}(x,y) &:= \tilde{P}_n^{(0,0)}(x), \\ Q_{n,k}^{(0,0,1)}(x,y) &:= z P_{n-1,k-1}^{(0,0,1)}(x,y) \qquad \text{for } k = 1, \ldots, n. \end{aligned}$$

The ordering is chosen so that the conversion operators derived below are upper triangular. Each basis has a simple restriction formula to the corresponding edge.

PROPOSITION B.2. *Restriction operator to $x = 0$:*

$$\begin{aligned} Q_{n,n}^{(1,0,0)}(0,y) &:= \tilde{P}_n^{(0,0)}(y), \\ Q_{n,k}^{(1,0,0)}(0,y) &:= 0 \qquad \text{for } k \neq n. \end{aligned}$$

Restriction operator to $y = 0$:

$$\begin{aligned} Q_{n,0}^{(0,1,0)}(x,0) &:= \tilde{P}_n^{(0,0)}(x), \\ Q_{n,k}^{(0,1,0)}(x,0) &:= 0 \qquad \text{for } k \neq 0. \end{aligned}$$

Restriction operator to $z = 0$:

$$\begin{aligned} Q_{n,0}^{(0,0,1)}(x,1-x) &:= \tilde{P}_n^{(0,0)}(x), \\ Q_{n,k}^{(0,0,1)}(x,1-x) &:= 0 \qquad \text{for } k \neq 0. \end{aligned}$$

B.2. Two-edge Dirichlet basis.

To handle two edges, consider first $x = 0$ and $y = 0$. As before, we wish to construct a basis that adds in the missing polynomials to $xy P_{n,k}^{(1,1,0)}(x,y)$ in such a way that the restriction operators have the necessary structure. To do this, we select polynomials so that we can construct the conversion operator to expansions in the basis $\boldsymbol{Q}^{(1,0,0)}$ and use the restriction operators we already have (see Corollary C.1).

DEFINITION B.3. *The following polynomials vanish at $x = 0$ and $y = 0$ apart from when $k = 0, n$:*

$$\begin{aligned}
&Q^{(1,1,0)}_{0,0}(x,y) := 1,\\
&Q^{(1,1,0)}_{n,0}(x,y) := x\tilde{P}^{(0,1)}_{n-1}(x),\\
&Q^{(1,1,0)}_{n,k}(x,y) := xyP^{(1,1,0)}_{n-2,k-1}(x,y) \qquad \text{for } k = 1,\dots,n-1,\\
&Q^{(1,1,0)}_{n,n}(x,y) := yP^{(0,1,0)}_{n-1,n-1}(x,y).
\end{aligned}$$

The following polynomials vanish at $x = 0$ and $z = 0$ apart from when $k = 0, n$:

$$\begin{aligned}
&Q^{(1,0,1)}_{0,0}(x,y) := 1,\\
&Q^{(1,0,1)}_{n,0}(x,y) := x\tilde{P}^{(0,1)}_{n-1}(x),\\
&Q^{(1,0,1)}_{n,k}(x,y) := xzP^{(1,0,1)}_{n-2,k-1}(x,y) \qquad \text{for } k = 1,\dots,n-1,\\
&Q^{(1,0,1)}_{n,n}(x,y) := zP^{(0,0,1)}_{n-1,n-1}(x,y).
\end{aligned}$$

The following polynomials vanish at $y = 0$ and $z = 0$ apart from when $k = 0, 1$:

$$\begin{aligned}
&Q^{(0,1,1)}_{0,0}(x,y) := 1,\\
&Q^{(0,1,1)}_{n,0}(x,y) := (1-x)P_{n-1,0}(x,y) = (1-x)\tilde{P}^{(1,0)}_{n-1}(x),\\
&Q^{(0,1,1)}_{n,1}(x,y) := (1-x-2y)P_{n-1,0}(x,y) = (1-x-2y)\tilde{P}^{(1,0)}_{n-1}(x),\\
&Q^{(0,1,1)}_{n,k}(x,y) := yzP^{(0,1,1)}_{n-2,k-2}(x,y) \qquad \text{for } k = 2,\dots,n.
\end{aligned}$$

B.3. Three-edge Dirichlet basis. We finally get to three edges. Again, we want to choose the extra polynomials so that we can easily convert to any two-edge cases. The following does the trick.

DEFINITION B.4. *The following polynomials vanish at $x = 0$, $y = 0$, and $z = 0$ apart from when $k = 0, 1$, and n:*

$$\begin{aligned}
&Q^{(1,1,1)}_{0,0}(x,y) := 1,\\
&Q^{(1,1,1)}_{1,0}(x,y) := 1-2x,\\
&Q^{(1,1,1)}_{1,1}(x,y) := 1-x-2y,\\
&Q^{(1,1,1)}_{n,0}(x,y) := x(1-x)P^{(1,0,0)}_{n-2,0}(x,y) = x(1-x)P^{(1,1)}_{n-2}(x),\\
&Q^{(1,1,1)}_{n,1}(x,y) := x(1-x-2y)P^{(1,0,0)}_{n-2,0}(x,y) = x(1-x-2y)P^{(1,1)}_{n-2}(x),\\
&Q^{(1,1,1)}_{n,k}(x,y) := xyzP^{(1,1,1)}_{n-3,k-2}(x,y) \qquad \text{for } k = 2,\dots,n-1,\\
&Q^{(1,1,1)}_{n,n}(x,y) := yzP^{(0,1,1)}_{n-2,n-2}(x,y).
\end{aligned}$$

Appendix C. Dirichlet basis recurrence relationships. The following allows us to construct sparse conversion operators from the one-edge Dirichlet basis to the standard Jacobi polynomials on the triangle.

COROLLARY C.1. *The following recurrence relationships hold:*

$$\begin{aligned}
Q_{0,0}^{(1,0,0)}(x,y) &= P_{0,0}(x,y),\\
(2n+1)Q_{n,k}^{(1,0,0)}(x,y) &= (n-k)\left[P_{n,k}(x,y)+P_{n-1,k}(x,y)\right],\\
Q_{n,n}^{(1,0,0)}(x,y) &= P_{n,n}(x,y),\\
(2n+1)Q_{n,0}^{(0,1,0)}(x,y) &= (n+1)P_{n,0}(x,y)-nP_{n-1,0}(x,y),\\
(2n+1)Q_{n,k}^{(0,1,0)}(x,y) &= (n+k+1)P_{n,k}(x,y)-(n-k+1)P_{n,k-1}(x,y)\\
&\quad -(n-k)P_{n-1,k}(x,y)+(n+k)P_{n-1,k-1}(x,y),\\
(2n+1)Q_{n,0}^{(0,0,1)}(x,y) &= (n+1)P_{n,0}(x,y)-nP_{n-1,0}(x,y),\\
(2n+1)Q_{n,k}^{(0,0,1)}(x,y) &= -(n+k+1)P_{n,k}(x,y)-(n-k+1)P_{n,k-1}(x,y)\\
&\quad +(n-k)P_{n-1,k}(x,y)+(n+k)P_{n-1,k-1}(x,y).
\end{aligned}$$

Proof. These are either immediate from definitions or are obtained by rearranging recurrence relationships found in Corollary A.4. ∎

The two-edge Dirichlet basis satisfies several sparse recurrence relationships.

COROLLARY C.2. *The following recurrence relationships hold:*

$$\begin{aligned}
Q_{0,0}^{(1,1,0)}(x,y) &= Q_{0,0}^{(1,0,0)}(x,y),\\
2nQ_{n,0}^{(1,1,0)}(x,y) &= (n+1)Q_{n,0}^{(1,0,0)}(x,y)-nQ_{n-1,0}^{(1,0,0)}(x,y),\\
4nQ_{n,k}^{(1,1,0)}(x,y) &= (n+k+1)Q_{n,k}^{(1,0,0)}(x,y)-(n-k)Q_{n,k-1}^{(1,0,0)}(x,y)\\
&\quad +(k-n)Q_{n-1,k}^{(1,0,0)}(x,y)+(n+k-1)Q_{n-1,k-1}^{(1,0,0)}(x,y),\\
2Q_{n,n}^{(1,1,0)}(x,y) &= Q_{n,n}^{(1,0,0)}(x,y)-Q_{n,n-1}^{(1,0,0)}(x,y)+Q_{n-1,n-1}^{(1,0,0)}(x,y).
\end{aligned}$$

$$\begin{aligned}
2Q_{n,0}^{(1,1,0)}(x,y) &= Q_{n,0}^{(0,1,0)}(x,y)+Q_{n-1,0}^{(0,1,0)}(x,y),\\
2nQ_{n,k}^{(1,1,0)}(x,y) &= (n-k)\left[Q_{n,k}^{(0,1,0)}(x,y)+Q_{n-1,k}^{(0,1,0)}(x,y)\right],\\
Q_{n,n}^{(1,1,0)}(x,y) &= Q_{n,n}^{(0,1,0)}(x,y).
\end{aligned}$$

$$\begin{aligned}
Q_{0,0}^{(1,0,1)}(x,y) &= Q_{0,0}^{(1,0,0)}(x,y),\\
2nQ_{n,0}^{(1,0,1)}(x,y) &= (n+1)Q_{n,0}^{(1,0,0)}(x,y)-nQ_{n-1,0}^{(1,0,0)}(x,y),\\
4nQ_{n,k}^{(1,0,1)}(x,y) &= -(n+k+1)Q_{n,k}^{(1,0,0)}(x,y)-(n-k)Q_{n,k-1}^{(1,0,0)}(x,y)\\
&\quad +(n-k)Q_{n-1,k}^{(1,0,0)}(x,y)+(n+k-1)Q_{n-1,k-1}^{(1,0,0)}(x,y),\\
2Q_{n,n}^{(1,0,1)}(x,y) &= -Q_{n,n}^{(1,0,0)}(x,y)-Q_{n,n-1}^{(1,0,0)}(x,y)+Q_{n-1,n-1}^{(1,0,0)}(x,y).
\end{aligned}$$

$$
\begin{aligned}
2Q_{n,0}^{(1,0,1)}(x,y) &= Q_{n,0}^{(0,0,1)}(x,y) + Q_{n-1,0}^{(0,0,1)}(x,y),\\
2nQ_{n,k}^{(1,0,1)}(x,y) &= (n-k)\left[Q_{n,k}^{(0,0,1)}(x,y) + Q_{n-1,k}^{(0,0,1)}(x,y)\right],\\
Q_{n,n}^{(1,0,1)}(x,y) &= Q_{n,n}^{(0,0,1)}(x,y).
\end{aligned}
$$

$$
\begin{aligned}
2Q_{n,0}^{(0,1,1)}(x,y) &= -Q_{n,0}^{(0,1,0)}(x,y) + Q_{n-1,0}^{(0,1,0)}(x,y),\\
2nQ_{n,1}^{(0,1,1)}(x,y) &= -2(n+1)Q_{n,1}^{(0,1,0)}(x,y) - nQ_{n,0}^{(0,1,0)}(x,y)\\
&\quad + 2(n-1)Q_{n-1,1}^{(0,1,0)}(x,y) + nQ_{n-1,0}^{(0,1,0)}(x,y),\\
2n(2k-1)Q_{n,k}^{(0,1,1)}(x,y) &= -(k-1)(n+k)Q_{n,k}^{(0,1,0)}(x,y) - (k-1)(n-k+1)Q_{n,k-1}^{(0,1,0)}(x,y)\\
&\quad +(k-1)(n-k)Q_{n-1,k}^{(0,1,0)}(x,y) + (k-1)(n+k-1)Q_{n-1,k-1}^{(0,1,0)}(x,y).
\end{aligned}
$$

$$
\begin{aligned}
2Q_{n,0}^{(0,1,1)}(x,y) &= -Q_{n,0}^{(0,0,1)}(x,y) + Q_{n-1,0}^{(0,0,1)}(x,y),\\
2nQ_{n,1}^{(0,1,1)}(x,y) &= 2(n+1)Q_{n,1}^{(0,1,0)}(x,y) + nQ_{n,0}^{(0,0,1)}(x,y)\\
&\quad - 2(n-1)Q_{n-1,1}^{(0,1,0)}(x,y) - nQ_{n-1,0}^{(0,0,1)}(x,y),\\
&2n(2k-1)Q_{n,k}^{(0,1,1)}(x,y)\\
&= (k-1)(n+k)Q_{n,k}^{(0,0,1)}(x,y) - (k-1)(n-k+1)Q_{n,k-1}^{(0,0,1)}(x,y)\\
&\quad -(k-1)(n-k)Q_{n-1,k}^{(0,0,1)}(x,y) + (k-1)(n+k-1)Q_{n-1,k-1}^{(0,0,1)}(x,y).
\end{aligned}
$$

Proof. These are either immediate from definitions or are obtained by rearranging recurrence relationships found in Corollary A.4. ∎

The three-edge Dirichlet basis also satisfies several sparse recurrence relationships.

COROLLARY C.3. *The following recurrence relationships hold:*

$$
\begin{aligned}
Q_{0,0}^{(1,1,1)}(x,y) &= Q_{0,0}^{(0,1,1)}(x,y),\\
Q_{1,0}^{(1,1,1)}(x,y) &= 2Q_{1,0}^{(0,1,1)}(x,y) - Q_{0,0}^{(0,1,1)}(x,y),\\
Q_{1,1}^{(1,1,1)}(x,y) &= Q_{1,1}^{(0,1,1)}(x,y),\\
(2n-1)Q_{n,0}^{(1,1,1)}(x,y) &= (n-1)\left[Q_{n,0}^{(0,1,1)}(x,y) + Q_{n-1,0}^{(0,1,1)}(x,y)\right],\\
(2n-1)Q_{n,k}^{(1,1,1)}(x,y) &= (n-k)\left[Q_{n,k}^{(0,1,1)}(x,y) + Q_{n-1,k}^{(0,1,1)}(x,y)\right],\\
Q_{n,n}^{(1,1,1)}(x,y) &= Q_{n,n}^{(0,1,1)}(x,y).
\end{aligned}
$$

$$
\begin{aligned}
Q^{(1,1,1)}_{0,0}(x,y) &= Q^{(1,0,1)}_{0,0}(x,y),\\
Q^{(1,1,1)}_{1,0}(x,y) &= -2Q^{(1,0,1)}_{1,0}(x,y) + Q^{(1,0,1)}_{0,0}(x,y),\\
Q^{(1,1,1)}_{1,1}(x,y) &= 2Q^{(1,0,1)}_{1,1}(x,y) + Q^{(1,0,1)}_{1,0}(x,y) - Q^{(1,0,1)}_{0,0}(x,y),\\
(2n-1)Q^{(1,1,1)}_{n,0}(x,y) &= (n-1)\left[-Q^{(1,0,1)}_{n,0}(x,y) + Q^{(1,0,1)}_{n-1,0}(x,y)\right],\\
(2n-1)Q^{(1,1,1)}_{n,1}(x,y) &= 2(n+1)Q^{(1,0,1)}_{n,1}(x,y) + (n-1)Q^{(1,0,1)}_{n,0}(x,y)\\
&\quad - 2(n-1)Q^{(1,0,1)}_{n-1,1}(x,y) - (n-1)Q^{(1,0,1)}_{n-1,0}(x,y)],\\
(2n-1)(2k-1)&Q^{(1,1,1)}_{n,k}(x,y)\\
&= (n+k)(k-1)Q^{(1,0,1)}_{n,k}(x,y) - (n-k)(k-1)Q^{(1,0,1)}_{n,k-1}(x,y)\\
&\quad -(n-k)(k-1)Q^{(1,0,1)}_{n-1,k}(x,y) + (n+k-2)(k-1)Q^{(1,0,1)}_{n-1,k-1}(x,y),\\
(2n-1)&Q^{(1,1,1)}_{n,n}(x,y)\\
&= (n-1)\left[Q^{(1,0,1)}_{n,n}(x,y) - Q^{(1,0,1)}_{n,n-1}(x,y) + Q^{(1,0,1)}_{n-1,n-1}(x,y)\right].
\end{aligned}
$$

$$
\begin{aligned}
Q^{(1,1,1)}_{0,0}(x,y) &= Q^{(1,1,0)}_{0,0}(x,y),\\
Q^{(1,1,1)}_{1,0}(x,y) &= -2Q^{(1,1,0)}_{1,0}(x,y) + Q^{(1,1,0)}_{0,0}(x,y),\\
Q^{(1,1,1)}_{1,1}(x,y) &= -2Q^{(1,1,0)}_{1,1}(x,y) - Q^{(1,1,0)}_{1,0}(x,y) + Q^{(1,1,0)}_{0,0}(x,y),\\
(2n-1)Q^{(1,1,1)}_{n,0}(x,y) &= (n-1)\left[-Q^{(1,1,0)}_{n,0}(x,y) + Q^{(1,1,0)}_{n-1,0}(x,y)\right],\\
(2n-1)Q^{(1,1,1)}_{n,1}(x,y) &= -2(n+1)Q^{(1,1,0)}_{n,1}(x,y) - (n-1)Q^{(1,1,0)}_{n,0}(x,y)\\
&\quad +2(n-1)Q^{(1,1,0)}_{n-1,1}(x,y) + (n-1)Q^{(1,1,0)}_{n-1,0}(x,y)],\\
(2n-1)(2k-1)&Q^{(1,1,1)}_{n,k}(x,y)\\
&= -(n+k)(k-1)Q^{(1,1,0)}_{n,k}(x,y) - (n-k)(k-1)Q^{(1,1,0)}_{n,k-1}(x,y)\\
&\quad +(n-k)(k-1)Q^{(1,1,0)}_{n-1,k}(x,y) + (n+k-2)(k-1)Q^{(1,1,0)}_{n-1,k-1}(x,y),\\
(2n-1)&Q^{(1,1,1)}_{n,n}(x,y)\\
&= (n-1)\left[-Q^{(1,1,0)}_{n,n}(x,y) - Q^{(1,1,0)}_{n,n-1}(x,y) + Q^{(1,1,0)}_{n-1,n-1}(x,y)\right].
\end{aligned}
$$

Proof. These are either immediate from definitions or are obtained by rearranging recurrence relationships found in Corollary A.4. ∎

C.1. Recurrence relationships for the partial derivatives of the Dirichlet basis. We now turn to recurrence relationships for the partial derivatives of the Dirichlet basis, which are needed when imposing Neumann boundary conditions.

COROLLARY C.4. *The following recurrence relationships hold:*

$$\begin{aligned}
\frac{\partial}{\partial y}Q_{n,0}^{(0,1,1)}(x,y) &= 0,\\
\frac{\partial}{\partial y}Q_{n,1}^{(0,1,1)}(x,y) &= -2P_{n-1,0}(x,y),\\
\frac{\partial}{\partial y}Q_{n,k}^{(0,1,1)}(x,y) &= (1-k)P_{n-1,k-1}(x,y),\\
\frac{\partial}{\partial x}Q_{n,0}^{(1,0,1)}(x,y) &= nP_{n-1,0}(x,y),\\
\frac{\partial}{\partial x}Q_{n,k}^{(1,0,1)}(x,y) &= \frac{k-n}{2}\left[P_{n-1,k-1}(x,y)+P_{n-1,k}(x,y)\right],\\
\frac{\partial}{\partial x}Q_{n,n}^{(1,0,1)}(x,y) &= -nP_{n-1,n-1}(x,y),\\
\frac{\partial}{\partial z}Q_{n,0}^{(1,1,0)}(x,y) &= -nP_{n-1,0}(x,y),\\
\frac{\partial}{\partial z}Q_{n,k}^{(1,1,0)}(x,y) &= \frac{n-k}{2}\left[P_{n-1,k-1}(x,y)-P_{n-1,k}(x,y)\right],\\
\frac{\partial}{\partial z}Q_{n,n}^{(1,1,0)}(x,y) &= nP_{n-1,n-1}(x,y).
\end{aligned}$$

Proof. The first three relations follow from the weighted partial differentiation relationships (see Corollary A.2). The fourth relation requires the additional property that

$$\frac{\mathrm{d}}{\mathrm{d}x}\left[x\tilde{P}_{n-1}^{(0,1)}(x)\right] = n\tilde{P}_{n-1}^{(1,0)}(x),$$

which follows from [15, (15.5.6)]. The fifth relationship also follows from Corollary A.2. For the sixth equation, if we define $t = y/(1-x)$, then the relation reduces to

$$(1-x)^{n-1}\left[((n-1)(1-t)+1)\tilde{P}_{n-1}^{(1,0)}(t) - t(1-t)\frac{\mathrm{d}}{\mathrm{d}t}\tilde{P}_{n-1}^{(1,0)}(t)\right] = n(1-x)^{n-1}\tilde{P}_{n-1}(t),$$

and this expression follows from $\mathcal{L}_2^\dagger$ in [19, Lem. 1]. The last three relations follow from the same manipulation. ∎

Acknowledgments. This work began when the second author visited the first author at The University of Sydney. We are grateful for the travel support provided by The University of Sydney. We also thank Andrew Horning and Nicolas Boulle for carefully reading the draft and improving the text.

REFERENCES

[1] M. AINSWORTH, G. ANDRIAMARO, AND O. DAVYDOV, *Bernstein–Bézier finite elements of arbitrary order and optimal assembly procedures*, SIAM J. Sci. Comput., 33 (2011) pp. 3087–3109, https://doi.org/10.1137/11082539X.

[2] S. BEUCHLER AND J. SCHÖBERL, *New shape functions for triangular p-FEM using integrated Jacobi polynomials*, Numer. Math., 103 (2006), pp. 339–366.

[3] J. P. BOYD, *Chebyshev and Fourier Spectral Methods*, 2nd ed., Dover, Mineola, NY, 2001.

[4] C. W. CLENSHAW, *The numerical solution of linear differential equations in Chebyshev series*, Proc. Cambridge Philos. Soc., 53 (1957), pp. 134–149.

[5] C. W. CLENSHAW, *A note on the summation of Chebyshev series*, Math. Tables Aids Comput., 9 (1955), pp. 118–120.

[6] P. Deift, *Orthogonal Polynomials and Random Matrices: A Riemann–Hilbert Approach*, Courant Lect. Notes Math. 3, AMS, Providence, RI, 1999.
[7] M. Dubiner, *Spectral methods on triangles and other domains*, J. Sci. Comput., 6 (1991), pp. 345–390.
[8] C. F. Dunkl and Y. Xu, *Orthogonal Polynomials of Several Variables*, 2nd ed., Encyclopedia Math. Appl. 155, Cambridge University Press, Cambridge, 2014.
[9] W. Gautschi, *Orthogonal Polynomials: Computation and Approximation*, Oxford University Press, New York, 2004.
[10] K. Julien and M. Watson, *Efficient multi-dimensional solution of PDEs using Chebyshev spectral methods*, J. Comput. Phys., 228 (2009), pp. 1480–1503.
[11] G. Karniadakis and S. Sherwin, *Spectral/hp Element Methods for Computational Fluid Dynamics*, Oxford University Press, New York, 2013.
[12] T. Koornwinder, *Two-variable analogues of the classical orthogonal polynomials*, in Theory and Application of Special Functions (Proc. Advanced Sem., Math. Res. Center, Univ. Wisconsin, Madison, WI, 1975), Academic Press, New York, 1975, pp. 435–495.
[13] D. Kressner, *Bivariate matrix functions*, Oper. Matrices, 8 (2014), pp. 449–466.
[14] H. Li and J. Shen, *Optimal error estimates in Jacobi-weighted Sobolev spaces for polynomial approximations on the triangle*, Math. Comp., 79 (2010) pp. 1621–1646.
[15] F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, and B. V. Saunders, eds., *NIST Digital Library of Mathematical Functions*, http://dlmf.nist.gov/, release 1.0.16 of 2017-09-18.
[16] S. Olver, *BlockBandedMatrices.jl*, v0.4.6, https://www.github.com/JuliaMatrices/BlockBandedMatrices.jl.
[17] S. Olver, *MultivariateOrthogonalPolynomials.jl*, v0.0.1, https://github.com/JuliaApproximation/MultivariateOrthogonalPolynomials.jl.
[18] S. Olver and A. Townsend, *A fast and well-conditioned spectral method*, SIAM Rev., 55 (2013), pp. 462–489, https://doi.org/10.1137/120865458.
[19] S. Olver, A. Townsend, and G. M. Vasil, *Recurrence relations for a family of orthogonal polynomials on a triangle*, in Proceedings of ICOSAHOM 2018, to appear.
[20] E. L. Ortiz, *The tau method*, SIAM J. Numer. Anal., 6 (1969), pp. 480–492, https://doi.org/10.1137/0706044.
[21] J. Shen, T. Tang, and L.-L. Wang, *Spectral Methods: Algorithms, Analysis and Applications*, Springer Ser. Comput. Math. 41, Springer, Heidelberg, 2011.
[22] R. M. Slevinsky, *Conquering the Pre-computation in Two-dimensional Harmonic Polynomial Transforms*, preprint, https://arxiv.org/abs/1711.07866, 2017.
[23] R. M. Slevinsky, *Fast and backward stable transforms between spherical harmonic expansions and bivariate Fourier series*, Appl. Comput. Harmon. Anal., 47 (2019), pp. 585–606.
[24] R. M. Slevinsky, *FastTransforms*, v0.1, https://github.com/MikaelSlevinsky/FastTransforms.
[25] R. M. Slevinsky and S. Olver, *A fast and well-conditioned spectral method for singular integral equations*, J. Comput. Phys., 332 (2017), pp. 290–315.
[26] G. Szegö, *Orthogonal Polynomials*, American Mathematical Society Colloquium Publications 23, American Mathematical Society, New York, 1939.
[27] G. Teschl, *Jacobi Operators and Completely Integrable Nonlinear Lattices*, Math. Surveys Monogr. 72, Amer. Math. Soc., Providence, RI, 2000.
[28] A. Townsend and S. Olver, *The automatic solution of partial differential equations using a global spectral method*, J. Comput. Phys., 299 (2015), pp. 106–123.
[29] L. N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, Philadelphia, 2013.
[30] L. N. Trefethen, *Spectral Methods in MATLAB*, SIAM, Philadelphia, 2000, https://doi.org/10.1137/1.9780898719598.
[31] G. M. Vasil, K. J. Burns, D. Lecoanet, S. Olver, B. P. Brown, and J. S. Oishi, *Tensor calculus in polar coordinates using Jacobi polynomials*, J. Comput. Phys., 325 (2016), pp. 53–73.
[32] Y. Xu, *Approximation and orthogonality in Sobolev spaces on a triangle*, Constr. Approx., 46 (2017), pp. 349–434.