Numerische Mathematik manuscript No.
(will be inserted by the editor)

Fast simplicial finite element algorithms using Bernstein polynomials

Robert C. Kirby[1] ⋆

Texas Tech University

Received: date / Revised version: date

Summary Fast algorithms for applying finite element mass and stiffness operators to the B-form of polynomials over d-dimensional simplices are derived. These rely on special properties of the Bernstein basis and lead to stiffness matrix algorithms with the same asymptotic complexity as tensor-product techniques in rectangular domains. First, special structure leading to fast application of mass matrices is developed. Then, by factoring stiffness matrices into products of sparse derivative matrices with mass matrices, fast algorithms are also obtained for stiffness matrices.

Send offprint requests to:

⋆ *Present address:* Texas Tech University; Department of Mathematics and Statistics; PO Box 1042; Lubbock, TX 79409-1042. This work supported by the National Science Foundation under award number 0830655.

1 Introduction

Spectral element techniques [3,4] accelerate the application of high-order finite element operators while reducing memory usage. At the heart of the methodology is a fast, matrix-free algorithm by which the action of a local element matrix on a vector may be computed. These techniques work by scattering global degrees of freedom to each cell, computing the action of the element stiffness matrix efficiently, then assembling the elementwise results. To develop a fast algorithm, it is sufficient to consider finite element operators acting on one cell in a mesh. For high-degree polynomials in multiple dimensions, the reduction in storage and arithmetic requirements is dramatic, and the benefit of reduced arithmetic is realized at each step of an iterative method. The process of assembling these local contributions in nodal and Bernstein bases is well-documented in the literature [16,9,10], and is not dealt with here.

When the mesh consists of rectangles or hexahedra, polynomial bases are naturally expressed in tensor-product form using Lagrange or other standard polynomials in each coordinate direction. Then, local stiffness matrices have a natural decomposition into a Kronecker product of one-dimensional operators. By applying only one-dimensional operators in each direction, the complexity of applying

the element stiffness matrix is greatly reduced. Moreover, selecting the Lagrange nodes to be the tensor product of Gauss-Lobattto leads to diagonalized mass matrices [16], further improving the efficiency.

Simplicial domains present a greater challenge to the development of efficient algorithms. By using special bases composed of tensor products of Jacobi polynomials combined with special quadrature rules, spectral techniques may be developed. The book of Karniadakis and Sherwin [9] and the literature cited therein contain a thorough presentation of these bases and the efficient evaluation of finite element operators expressed in them. Using nodal bases, low-complexity algorithms are harder to come by. In special cases, mass matrices may be diagonalized by special integration rules [5], but a general technique is not known.

Instead of dealing with special bases for which fast algorithms are apparent, this paper instead deals with a standard polynomial basis, the Bernstein polynomials, and develops special algorithms for it. A polynomial expressed in the Bernstein basis is typically referred to as *B-form*. The Bernstein basis, while not nodal like Lagrange polynomials, has members whose boundary support is restricted to particular facets of a cell, enabling them to be pieced together continuously. Like Lagrange polynomials, they form a partition of unity, but they are

also totally nonnegative. They are also appropriate building blocks for smooth splines [10], and are widely used in computer graphics. Hosever, they are only recently being considered in a finite element context. In [12], Petersen *et al* find Bernstein polynomials for certain acoustics problems to be stable and efficient when systems are solved with preconditioned Krylov methods. The recent work of Arnold *et al* [1] on finite element exterior calculus starts with Bernstein polynomials for simplicial 0-forms and expresses bases for 1-forms and 2-forms in barycentric coordinates similar to B-form. Schumaker [15] has posed finite element methods over triangular splines, which locally are expressed in B-form. The work of Hughes *et al* in isogeometric analysis [8] uses rational splines to represent both the domain geometry and the finite element bases, though typically using rectangular patches. While efficient recursive algorithms in B-form are well-known for evaluation, refinement, and rendering, this work seems to be the first derivation of fast algorithms for finite element operators acting on B-form.

In this paper, fast algorithms for computing the action of finite element mass and stiffness matrices in B-form over a simplex are derived. The mass matrix algorithm requires $\mathcal{O}(dn^{d+1})$ operations, where n is the polynomial degree and d is the spatial dimension.

Stiffness matrices are about d times as expensive. While not as efficient as a diagonalized mass matrix, these algorithms have similar complexity as simplicial sum-factorization techniques in [9]. This complexity and memory usage far improves on the $\mathcal{O}(n^{2d})$ requirements of explicitly constructing the matrices. Moreover, the flexibility of B-form allow it to be used not only in place of Lagrange elements in C^0 finite elements, but in the future can lead to spectrally efficient, high-smoothness splines and also $H(\mathrm{div})$ and $H(\mathrm{curl})$ bases.

While the techniques developed here rely on affine geometry, it is typically possible to resolve more general domains into a large number of affine simplices and used curved elements only near the boundary. Such a method is used in the nodal spectral elements of Hesthaven and Warburton [7], where level 3 BLAS is used to efficiently apply operators on all internal (affine) cells, and a more standard quadrature technique is used to handle curved boundary cells. In the context of NURBS-based finite elements of Sevilla *et al* [13,14], exact geometric representations of boundaries are obtained while using standard finite element bases and low-order geometry in the interior of the domain. More importantly, the techniques here work with constant coefficient problems. In the future, it is hoped that the ideas here

may be extended quadrature-based evaluation algorithms that will both accomodate variable coefficients and curved geometry.

After establishing some notation and reviewing important facts about Bernstein polynomials on the simplex in Section 2, block structure arising in vectors and matrices representing B-form polynomials is explored. This leads to fast algorithms for applying an element mass matrix in Section 3. These techniques are recursive by simplex dimension, relying on a naive algorithm for the one-dimensional case. By factoring element stiffness matrices into products of sparse derivative matrices and mass matrices, fast stiffness algorithms are obtained in Section 4. However, these operators on a line segment can be reduced to $\mathcal{O}(N \log N)$ complexity due to certain Hankel matrix structure that arises. Finally, some timing results for the two-dimensional operators confirming the predicted complexity are presented in Section 6, and some conclusions presented in Section 7.

2 Bernstein polynomials on the *d*-simplex

2.1 Multiindex notation

Multiindex notation will expedite the development of these ideas with some generality with respect to spatial dimension. These shall be denoted by lowercase Greek letters such as α and β. A multiindex of

length d is a d-tuple of nonnegative integers, written

$$\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_d).$$

Let A_d denote the set of all multiindices of length $d+1$. The *order* of a multiindex α, written $|\alpha|$ is given by the sum of its components. That is,

$$|\alpha| = \sum_{i=1}^{d+1} \alpha_i.$$

If $\alpha, \beta \in A_d$. their sum is defined componentwise by

$$\alpha + \beta = (\alpha_1 + \beta_1, \alpha_2 + \beta_2, \cdots, \alpha_{d+1} + \beta_{d+1}).$$

A partial ordering $\leq$ is defined for $\alpha, \beta \in A_d$ by $\alpha \leq \beta$ if $\alpha_i \leq \beta_1$ for each $1 \leq i \leq d+1$. If $\alpha \leq \beta$, then the difference

$$\beta - \alpha = (\beta_1 - \alpha_1, \beta_2 - \alpha_2, \cdots, \beta_{d+1} - \alpha_{d+1})$$

is well-defined.

The factorial of a multiindex is defined as the product of the components' factorials by

$$\alpha! = \Pi_{i=1}^{d+1} \alpha_i!$$

Given a nonnegative integer a and $\alpha \in A_d$, define a new multiindex $a \vdash \alpha \in A_{d+1}$ by

$$a \vdash \alpha = (a, \alpha_1, \alpha_2, \cdots, \alpha_{d+1}).$$

It will also be helpful to define an operation $\cdot'$ from A_d into A_{d-1} that takes the last d components of a multiindex. That is,

$$\alpha' = (\alpha_2, \alpha_3, \cdots \alpha_d).$$

So then, the identity

$$\alpha_1 \vdash \alpha' = \alpha$$

holds.

Let A_d^n denote the subset of A_d whose members have order n. That is,

$$A_d^n = \{\alpha \in A_d : |\alpha| = n\}.$$

The cardinality of A_d^n is simply the dimension of the the space of polynomials of degree n in d variables, that is, $\binom{n+d}{n}$. It is also useful to define $e_d^i \in A_d^1$ to be the multiindex with $(e_d^i)_j = \delta_{ij}$, only nonzero in position i.

2.2 Barycentric coordinates

For an integer $d \geq 1$, let S_d be a nondegenerate simplex in d space dimensions. The vertices of S_d are denoted by $\{x_i\}_{i=1}^{d+1} \subset \mathbb{R}^d$. Let

$$\{b_i\}_{i=1}^{d+1}$$

denote the *barycentric coordinates* of S_d. Each b_i is an affine map from $\mathbb{R}^d$ into $\mathbb{R}$ such that

$$b_i(x_j) = \delta_{ij}$$

for each $1 \leq i, j \leq d+1$, where δ_{ij} is the standard Kronecker delta. Each b_i is nonnegative on S_d, and

$$\sum_{i=1}^{d+1} b_i = 1.$$

2.3 Bernstein polynomials

Let $\mathbf{b}_d = (b_1, b_2, \cdots, b_{d+1})$ denote the tuple of barycentric coordinates on a d-simplex and let $\alpha \in A_d$. Monomials in the barycentric coordinates are compactly written as

$$\mathbf{b}_d^\alpha = \Pi_{i=1}^{d+1} b_i^{\alpha_i}$$

The Bernstein polynomials of degree n on the d-simplex are defined by

$$B_\alpha^n = \frac{n!}{\alpha!}\mathbf{b}_d^\alpha,$$

where $\alpha \in A_d^n$. The complete set of Bernstein polynoimals,

$$\{B_\alpha^n\}_{\alpha \in A_d^n},$$

form a basis of polynomials of complete degree n on S_d. Because the barycentric coordinates are nonnegative on S_d, it immediately

follows that each B^n_α is also nonnegative on S_d. In any dimension and for any degree, the Bernstein polynomials also form a partition of unity, which follows from applying the multinomial theorem to

$$1 = \left(\sum_{i=1}^{d+1} b_i \right)^n .$$

If $n > 0$ and u is a polynomial of degree $n-1$, then

$$u = \sum_{\alpha \in A^{n-1}_d} c_\alpha B^{n-1}_\alpha$$

for some vector of coefficients c_α. However, u is also a polynomial of degree n and so may be expressed as

$$u = \sum_{\alpha \in A^n_d} \tilde{c}_\alpha B^n_\alpha$$

for some vector $\tilde{c}_\alpha$. The process of obtaining $\tilde{c}$ from c is called *degree elevation*.

Degree elevation is naturally represented as a linear transformation $\tilde{c} = E^{d,n} c$ whose rows run over A^n_d and columns over A^{n-1}_d. From basic linear algebra, the column of $E^{d,n}$ corresponding to some $\beta \in A^{n-1}_d$ expresses B^{n-1}_β as a linear combination of $\{B^n_\alpha\}_{\alpha \in A^n_d}$. The structure of this column may be computed by

$$B^{n-1}_\beta = \left(\sum_{i=1}^{d+1} b_i \right) B^{n-1}_\beta = \sum_{i=1}^{d+1} b_i B^{n-1}_\beta, \tag{1}$$

since the barycentric coordinates sum to one. Considering each term of the sum,

$$\begin{aligned} b_i B_\beta^{n-1} &= b_i \frac{(n-1)!}{\beta!} \mathbf{b}_d^\beta \\ &= \frac{(n-1)!}{\beta!} \mathbf{b}_d^{\beta+e_d^i} \\ &= \frac{(n-1)!}{\beta!} \frac{(\beta+e_d^i)!}{n!} \frac{n!}{(\beta+e_d^i)!} \mathbf{b}_d^{\beta+e_d^i} \\ &= \frac{\beta_i+1}{n} B^n_{(\beta+e_d^i)}. \end{aligned}$$

This calculation shows that there are $d+1$ nonzero entries per column of $E^{d,n}$, independent of n, and gives exact row indices and numerical values. This holds independent of the polynomial degree n. The sparsity of $E^{d,n}$ will be important in developing fast algorithms for finite element operators.

Differentiation of polynomials expressed in B-form on a simplex S_d also has a matrix representation. Let $u = \sum_{\alpha \in A_d^n} c_\alpha B_\alpha^n$. Then for any coordinate direction x_i, $\frac{\partial u}{\partial x_i}$ is a polynomial of degree $n-1$ and so has a representation $\frac{\partial u}{\partial x_i} = \sum_{\alpha \in A_d^{n-1}} \hat{c}_\alpha B_\alpha^{n-1}$. Since differentiation is a linear process, there is some matrix D^{S_d,n,x_i} such that $\hat{c} = D^{S_d,n,x_i} c$.

Any directional can be expressed in terms of derivatives with respect to barycentric coordinates by the chain rule, for

$$\frac{\partial}{\partial x} = \sum_{i=1}^{d+1} \frac{\partial b_i}{\partial x} \frac{\partial}{\partial b_i}.$$

In affine geometry, each $\frac{\partial b_i}{\partial x}$ is a constant, and $\sum_{i=1}^{d+1} \frac{\partial b_i}{\partial x} = 0$.

Differentiation with respect to any barycentric coordinate is particularly simple:

$$\frac{d}{db_i} B_\alpha^n = \begin{cases} 0, & \alpha_i = 0 \\ n B_{\alpha - e_i^d}^{n-1}, & \alpha_i \neq 0 \end{cases}$$

Thus, barycentric derivatives are represented by matrices with one entry per row, and so any directional derivative is represented by a matrix with no more than $d+1$ entries per row. Like degree elevation, this is independent of the polynomial degree.

Integration rules for barycentric monomials will play an important role, giving exact formula for element mass matrices. If $\alpha \in A_d$, then

$$\int_{S_d} \mathbf{b}_d^\alpha = \frac{\alpha!}{(|\alpha| + d)!} |S_d| d!, \tag{2}$$

where $|S_d|$ is the hypervolume of S_d.

2.4 Recursive block structure via partial indexing

Let $u = \sum_{\alpha \in A_{d+1}^n} c_\alpha B_\alpha^n$ be some polynomial. The set of coefficients c_α is naturally represented as a vector c. When A_d^n is ordered lexicographically, there is a natural block structure to the vector. For

example, for cubics in two variables, c may be written as

$$c = \begin{pmatrix} c_{300} \\ \hline c_{210} \\ c_{201} \\ \hline c_{120} \\ c_{111} \\ c_{102} \\ \hline c_{030} \\ c_{021} \\ c_{012} \\ c_{003} \end{pmatrix}. \tag{3}$$

The lines are drawn to separate the portions of the vector where the first index is constant. This may also be written as a block vector

$$c = \begin{pmatrix} c_3 \\ \hline c_2 \\ \hline c_1 \\ \hline c_0 \end{pmatrix}. \tag{4}$$

This suggests a kind of *partial indexing* where c_α is interpreted as a block vector. However, the natural structure in the index set makes the blocks of differeing sizes. Here, c_3 is a vector with indices running over A_1^0, so the only index is $(0,0)$ and $(c_3)_{(0,0)} = c_{300}$. c_2 has indices running over $A_1^1 = \{(1,0),(0,1)\}$. More generally, for any $\alpha \in A_d^n$, c_{α_1}

is a vector with indices in $A_{d-1}^{n-\alpha_1}$. Using the $\cdot'$ operator introduced earlier, for any α, the identity

$$c_\alpha = (c_{\alpha_1})_{\alpha'}$$

expresses indexing as a two-stage process. Since c_{α_1} is itself a vector, the process of partial indexing may be repeated.

This blocking of vectors induces a blocking of matrices. Let $b(\cdot,\cdot)$ be a bilinear form over a simplex S_d and let $b : P_d^n \times P_d^m \to \mathbb{R}$. Associated with b is the $|A_d^m| \times |A_d^n|$ matrix

$$T_{\alpha\beta} = b(B_\beta^n, B_\alpha^m). \tag{5}$$

Given $u = \sum_{\alpha\in A_d^n} c_\alpha^u B_\alpha^n$ $v = \sum_{\alpha\in A_d^m} c_\alpha^v B_\alpha^m$, $b(u,v)$ is computed by

$$\sum_{\alpha\in A_d^m}\sum_{\beta\in A_d^n} c_\alpha^v T_{\alpha\beta} c_\beta^u.$$

This may be written in matrix-vector form as $(c^v)^t T c^u$.

Partially indexing the vectors c^u and c^v leads to a block structure of the matrix T. Fix $0 \le \alpha_1 \le m$ and $0 \le \beta_1 \le n$. Then define $T_{\alpha_1\beta_1}$ as a matrix with rows indexed over $T_{d-1}^{m-\alpha_1}$ and columns over $T_{d-1}^{n-\beta_1}$ by

$$(T_{\alpha_1\beta_1})_{\alpha'\beta'} = T_{(\alpha_1\vdash\alpha')(\beta_1\vdash\beta')}. \tag{6}$$

Examples will be given shortly, where blocked representations of degree elevation and differentation are shown.

It is typically the matrix-vector product $u \mapsto Bu$ that is of interest, as this operation is the core of Krylov methods for solving the global linear system. The global operator is a sum of contributions from each cell, each of which has the form of (5). The techniques in the next section rely on the blocking of such matrices developed here, deriving special relationships between the blocks that allow Bu to be computed faster than the standrad matrix-vector product.

2.5 Partitioned elevation and derivative matrices

Here, some specific examples of the degree elevation and derivative matrices are given, with lines drawn in the arrays to present partitioning structure. Note that, because of the ordering on A_d^n, the bottom right block corresponds to $\alpha_1 = \beta_1 = 0$.

Differentiation with respect to the barycentric coordinate b_i, described above, can be expressed as a $|A_d^{n-1}| \times |A_d^n|$ matrix, denoted

$D^{d,n,i}$. For example, the $D^{2,3,i}$ are readily calculated as

$$D^{2,3,0} = \left(\begin{array}{c|cc|ccc|cccc} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \end{array}\right),$$

$$D^{2,3,1} = \left(\begin{array}{c|cc|ccc|cccc} 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \end{array}\right),$$

and

$$D^{2,3,2} = \left(\begin{array}{c|cc|ccc|cccc} 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{array}\right).$$

As described above, any directional derivative may be expressed as a linear combination of barycentric derivatives, leading to a very sparse matrix.

The elevation operators also may be written in blocked matrix form. For example, the cubic one-dimensional elevation operator $E^{1,3}$ is

$$E^{1,3} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{3} & \frac{2}{3} & 0 \\ 0 & \frac{2}{3} & \frac{1}{3} \\ 0 & 0 & 1 \end{pmatrix}$$

The elevation operator from quadratics into cubics on the triangle is, with partitioning drawn in,

$$E^{2,3} = \left(\begin{array}{c|cc|ccc} 1 & 0 & 0 & 0 & 0 & 0 \\ \hline \frac{1}{3} & \frac{2}{3} & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{2}{3} & 0 & 0 & 0 \\ \hline 0 & \frac{2}{3} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{2}{3} & 0 & 0 & \frac{1}{3} \\ \hline 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ 0 & 0 & 0 & 0 & \frac{2}{3} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

3 Mass matrices

For any simplex S_d, the *mass matrix*[1] is defined from the L^2 inner product over S_d

$$m(u,v) = \int_{S_d} uv\,dx. \tag{7}$$

Using the integration rule (2), the entries of the mass matrix are

$$\begin{aligned} M_{\alpha,\beta}^{S_d,m,n} &= m(B_\beta^n, B_\alpha^m) \\ &= \int_{S_d} B_\alpha^m B_\beta^n\,dx \\ &= \int_{S_d} \frac{m!}{\alpha!}\frac{n!}{\beta!}\mathbf{b}_d^{\alpha+\beta}\,dx \\ &= \frac{m!}{\alpha!}\frac{n!}{\beta!}\frac{(\alpha+\beta)!}{(m+n+d)!}|S_d|d!, \end{aligned} \tag{8}$$

where $\alpha \in A_d^m$ and $\beta \in A_d^n$. Since this only depends on S_d through its volume, define

$$M_{\alpha,\beta}^{d,m,n} = \frac{m!}{\alpha!}\frac{n!}{\beta!}\frac{(\alpha+\beta)!}{(m+n+d)!} \tag{9}$$

so that

$$M_{\alpha,\beta}^{S_d,m,n} = M_{\alpha,\beta}^{d,m,n}|S_d|d!.$$

The nonnegativity of the Bernstein polynomials implies that $M^{d,m,n}$ will always be nonnegative. In fact, it is always positive, as can be seen in (9). This property does not hold for typical finite element bases such as the Lagrange polynomials.

[1] This matrix is also frequently called the *Gram matrix*, although this name is less common in the finite element literature

It is interesting to consider some examples of $M^{d,m,n}$. Each may be written with integer entries after suitable scaling. For $d = 1$ and $m = n$, some matrices are

$$6M^{1,1,1} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix},$$

$$30M^{1,2,2} = \begin{pmatrix} 6 & 3 & 1 \\ 3 & 4 & 3 \\ 1 & 3 & 6 \end{pmatrix},$$

and

$$140M^{1,3,3} = \begin{pmatrix} 20 & 10 & 4 & 1 \\ 10 & 12 & 9 & 4 \\ 4 & 9 & 12 & 10 \\ 1 & 4 & 10 & 20 \end{pmatrix}.$$

When $d > 1$, it is helpful to consider partitioning the mass matrices by partial indexing. For $m = n = 2$ and $m = n = 3$, these are

$$180M^{2,2,2} = \left(\begin{array}{c|cc|ccc} 6 & 3 & 3 & 1 & 1 & 1 \\ \hline 3 & 4 & 2 & 3 & 2 & 1 \\ 3 & 2 & 4 & 1 & 2 & 3 \\ \hline 1 & 3 & 1 & 6 & 3 & 1 \\ 1 & 2 & 2 & 3 & 4 & 3 \\ 1 & 1 & 3 & 1 & 3 & 6 \end{array}\right)$$

and

$$1120M^{2,3,3} = \left(\begin{array}{c|cc|ccc|cccc} 20 & 10 & 10 & 4 & 4 & 4 & 1 & 1 & 1 & 1 \\ \hline 10 & 12 & 6 & 9 & 6 & 3 & 4 & 3 & 2 & 1 \\ 10 & 6 & 12 & 3 & 6 & 9 & 1 & 2 & 3 & 4 \\ \hline 4 & 9 & 3 & 12 & 6 & 2 & 10 & 6 & 3 & 1 \\ 4 & 6 & 6 & 6 & 8 & 6 & 4 & 6 & 6 & 4 \\ 4 & 3 & 9 & 2 & 6 & 12 & 1 & 3 & 6 & 10 \\ \hline 1 & 4 & 1 & 10 & 4 & 1 & 20 & 10 & 4 & 1 \\ 1 & 3 & 2 & 6 & 6 & 3 & 10 & 12 & 9 & 4 \\ 1 & 2 & 3 & 3 & 6 & 6 & 4 & 9 & 12 & 10 \\ 1 & 1 & 4 & 1 & 4 & 10 & 1 & 4 & 10 & 20 \end{array}\right)$$

Inspection of of these matrices shows that the one-dimensional square mass matrices appear as the diagonal blocks of the two-dimensional mass matrices, up to a scaling factor. In fact, this is also true of the off-diagonal blocks, for

$$60M^{1,3,2} = \left(\begin{array}{ccc} 10 & 4 & 1 \\ 6 & 6 & 3 \\ 3 & 6 & 6 \\ 1 & 4 & 10 \end{array}\right).$$

In fact, patterns of this form hold for all spatial dimension and polynomial degree, as shall be proven shortly. More subtley, the blocks along a given column are themselves related by degree elevation op-

erators. These two pieces, taken together, form the critical pieces of a fast algorithm for multiplying a mass matrix onto a vector.

Before proceeding with the core results, the following basic fact will be required a few times.

Lemma 1 *Let $n, i \geq 0$. Then*

$$\frac{\binom{n}{i}}{\binom{n}{i+1}} = \frac{i+1}{n-i}. \tag{10}$$

Proof A straightforward calculation:

$$\begin{aligned}\frac{\binom{n}{i}}{\binom{n}{i+1}} &= \frac{\frac{n!}{(n-i)!(i!)}}{\frac{n!}{(n-i-1)!(i+1)!}} \\ &= \frac{(i+1)!}{i!}\frac{(n-i-1)!}{(n-i)!} \\ &= \frac{i+1}{n-i}.\end{aligned}$$

Proposition 1 *Let $d > 1$ and $m, n \geq 0$ be integers. Let $\alpha \in A_d^m$ and $\beta \in A_d^n$ be multiindices. The α_1, β_1 block of $M^{d,m,n}$ is related to a lower-dimensional mass matrix by*

$$M^{d,m,n}_{\alpha_1,\beta_1} = \frac{\binom{m}{\alpha_1}\binom{n}{\beta_1}}{\binom{m+n+d-1}{\alpha_1+\beta_1}(m+n+d)} M^{d-1,m-\alpha_1,n-\beta_1}. \tag{11}$$

Proof Let $\alpha \in A_d^m$ and $\beta \in A_d^n$ be given and decompose $\alpha = \alpha_1 \vdash \alpha'$ and $\beta = \beta_1 \vdash \beta'$.

Equation (9) gives that

$$M^{d-1,m-\alpha_1,n-\beta_1}_{\alpha',\beta'} = \frac{(m-\alpha_1)!(n-\beta_1)!(\alpha'+\beta')!}{(\alpha')!(\beta')!(m+n-\alpha_1-\beta_1+d-1)!}.$$

Partially indexing $M^{d,m,n}$ gives

$$\begin{aligned}
\left(M^{d,m,n}_{\alpha_1,\beta_1}\right)_{\alpha',\beta'} &= M^{d,m,n}_{\alpha,\beta} \\
&= \frac{m!n!(\alpha+\beta)!}{\alpha!\beta!(m+n+d)!} \\
&= \frac{m!n!(\alpha_1+\beta_1)!(\alpha'+\beta')!}{\alpha_1!(\alpha')!\beta_1!(\beta')!(m+n+d)!} \\
&= \frac{(m+n-\alpha_1-\beta_1+d-1)!(\alpha_1+\beta_1)!m!n!}{(m+n+d)!\alpha_1!\beta_1!(m-\alpha_1)!(n-\beta_1)!} \\
&\quad \times \frac{(m-\alpha_1)!(n-\beta_1)!(\alpha'+\beta')!}{(\alpha')!(\beta')!(m+n-\alpha_1-\beta_1+d-1)!},
\end{aligned}$$

where the factorials of multiindices have been split and the quantity has been multiplied and divided by $(m-\alpha_1)!$, $(n-\beta_1)!$, and $(m+n-\alpha_1-\beta_1+d-1)!$. Now, writing $(m+n+d)! = (m+n+d-1)!(m+n+d)$ and recognizing binomial coefficients, the result follows.

This result establishes a recursion down dimension among the blocks of a mass matrix. The next observation to make is that mass matrices over the same dimension but different polynomial degrees are related by degree elevation. In fact, this is generally true for bilinear forms over polynomials.

Lemma 2 *Let $b(\cdot,\cdot)$ be a bilinear form on $P^n_d \times P^m_d$. Let $T^{m,n}_{\alpha,\beta} = b(B^n_\beta, B^m_\alpha)$ be the matrix defined in (5). If $m \geq 0$ and $n \geq 1$, let $T^{m,n-1}_{\alpha,\beta} = b(B^{n-1}_\beta, B^m_\alpha)$ be the matrix obtained by restricting b to $P^{n-1}_d \times P^m_d$. Then*

$$T^{m,n-1} = T^{m,n} E^{d,n}. \tag{12}$$

Similarly, if $m \geq 1$ *and* $n \geq 0$ *and* $T^{m-1,n}_{\alpha,\beta} = b(B^n_\beta, B^{m-1}_\alpha)$, *then*

$$T^{m-1,n} = (E^{d,m})^t T^{m,n}. \tag{13}$$

Proof Let $P^{n-1}_d \ni u = \sum_{\beta \in A^{n-1}_d} c^u_\beta B^{n-1}_\beta$ and $P^m_d \ni v = \sum_{\beta \in A^m_d} c^v_\beta B^m_\beta$

Define $\tilde{c}^u = E^{d,n} c^u$ to be the degree-elevated coefficients of u so that $u = \sum_{\beta \in A^n_d} \tilde{c}^u_\beta B^n_\beta$..

On one hand, $b(u, v)$ is evaluated using $T^{m,n-1}$ by

$$\begin{aligned} b(u,v) &= b\left(\sum_{\beta \in A^{n-1}_d} c^u_\beta B^{n-1}_\beta, \sum_{\alpha \in A^m_d} c^v_\alpha B^m_\alpha \right) \\ &= \sum_{\alpha \in A^m_d} \sum_{\beta \in A^{n-1}_d} c^u_\beta c^v_\alpha b\left(B^{n-1}_\beta, B^m_\alpha \right) \\ &= \sum_{\alpha \in A^m_d} \sum_{\beta \in A^{n-1}_d} c^u_\beta c^v_\alpha T^{m,n-1}_{\alpha,\beta} \\ &= (c^v)^t T^{m,n-1} c^u \end{aligned}$$

On the other hand, using the elevated representation of u and $T^{m,n}$ gives

$$\begin{aligned} b(u,v) &= b\left(\sum_{\beta \in A^n_d} \tilde{c}^u_\beta B^n_\beta, \sum_{\alpha \in A^m_d} c^v_\alpha B^m_\alpha \right) \\ &= \sum_{\alpha \in A^m_d} \sum_{\beta \in A^n_d} \tilde{c}^u_\beta c^v_\alpha b(B^n_\beta, B^m_\alpha) \\ &= \sum_{\alpha \in A^m_d} \sum_{\beta \in A^n_d} \tilde{c}^u_\beta c^v_\alpha T^{m,n}_{\alpha,\beta} \\ &= (c^v)^t T^{m,n} \tilde{c}^u \\ &= (c^v)^t T^{m,n} \left(E^{d,n} c^u \right). \end{aligned}$$

Since $u \in P_d^{n-1}$, $v \in P_d^m$ were arbitrary, this establishes that $T^{m,n-1} = T^{m,n}E^{d,n}$. A similar argument shows (13).

This general result applies to mass matrices.

Proposition 2 *Let $d \geq 1$, $m \geq 0$, and $n \geq 1$. Then the mass matrices satisfy the relation*

$$M^{d,m,n-1} = M^{d,m,n}E^{d,n}. \tag{14}$$

Also, if $m \geq 1$ and $n \geq 0$, then

$$M^{d,m-1,n} = (E^{d,m})^t M^{d,m,n}. \tag{15}$$

These results lead to a recurrence relation between the blocks of a column of $M^{d,m,n}$.

Proposition 3 *For $0 \leq \alpha_1 < m$ and $0 \leq \beta_1 \leq$ and $d \geq 2$, the blocks of the mass matrix $M^{d,m,n}_{\alpha_1,\beta_1}$ satisfy*

$$M^{d,m,n}_{\alpha_1+1,\beta_1} = \frac{(1+\alpha_1+\beta_1)(m-\alpha_1)}{(m+n+d-1-\alpha_1-\beta_1)(1+\alpha_1)} \left(E^{d-1,m-\alpha_1}\right)^t M^{d,m,n}_{\alpha_1,\beta_1}. \tag{16}$$

Proof The proof is by calculation using the previous results. It follows that

$$\begin{aligned} M^{d,m,n}_{\alpha_1+1,\beta_1} &= \frac{\binom{m}{\alpha_1+1}\binom{n}{\beta_1}}{\binom{m+n+d-1}{\alpha_1+1+\beta_1}(m+n+d)} M^{d-1,m-\alpha_1-1,n-\beta_1} \\ &= \frac{\binom{m}{\alpha_1+1}\binom{n}{\beta_1}}{\binom{m+n+d-1}{\alpha_1+1+\beta_1}(m+n+d)} \left(\left(E^{d-1,m-\alpha_1}\right)^t M^{d-1,m-\alpha_1,n-\beta_1}\right). \end{aligned}$$

Now, it remains to multiply and divide the expression by appropriate binomial coefficients and use Proposition 1 again to obtain

$$\begin{aligned} M^{d,m,n}_{\alpha_1+1,\beta_1} &= \frac{\binom{m+n+d-1}{\alpha_1+\beta_1}\binom{m}{\alpha_1+1}}{\binom{m}{\alpha_1}\binom{m+n+d-1}{\alpha_1+1+\beta_1}} \\ &\quad \times \frac{\binom{m}{\alpha_1}\binom{n}{\beta_1}}{\binom{m+n+d-1}{\alpha_1+\beta_1}(m+n+d)} \left(E^{d-1,m-\alpha_1}\right)^t M^{d-1,m-\alpha_1,n-\beta_1} \\ &= \frac{\binom{m+n+d-1}{\alpha_1+\beta_1}\binom{m}{\alpha_1+1}}{\binom{m}{\alpha_1}\binom{m+n+d-1}{\alpha_1+1+\beta_1}} M^{d,m,n}_{\alpha_1,\beta_1}. \end{aligned}$$

The proof is completed by applying the lemma to the leading coefficient to obtain

$$\frac{\binom{m+n+d-1}{\alpha_1+\beta_1}\binom{m}{\alpha_1+1}}{\binom{m}{\alpha_1}\binom{m+n+d-1}{\alpha_1+1+\beta_1}} = \frac{(\alpha_1+1+\beta_1)(m-\alpha_1)}{(m+n+d-1-\alpha_1-\beta_1)(1+\alpha_1)}.$$

So, if a product $M^{d,m,n}_{\alpha_1,\beta_1} x_{\beta_1}$ is already computed, then $M^{d,m,n-1}_{\alpha_1,\beta_1} x_{\beta_1}$ is computed simply by applying the transpose of $E^{d,n}$ to the result and scaling by a constant. Rather than $\mathcal{O}\left(|A_{n-1}|^{d-1} \times |A_m|^{d-1}\right)$ operations for a standard matrix-vector product, this requires only $\mathcal{O}\left(d|A^{d-1}_m|\right)$. This process may be iterated along each colum, giving a columnwise algorithm for a blocked-matrix product, but using the recurrence of Proposition 3 along the columns.

This algorithm possesses a better complexity with respect to the polynomial degree than standard matrix-vector multiplication. To get the order of complexity, let $m = n$ and consider first the case $d = 2$. The standard algorithm will require $\mathcal{O}(n^4)$ operations to multiply a matrix with $\mathcal{O}(n^2)$ rows and columns onto a vector.

Algorithm 1 Improved algorithm for product for $y = M^{d,m,n}x$, where $d \geq 2$.

1: $y \leftarrow 0$

2: **for** $\beta_1 \leftarrow 0,n$ **do**

3: $\quad z^0 \leftarrow \frac{\binom{n}{\beta_1}}{\binom{m+n+d-1}{\beta_1}(m+n+d)} M^{d-1,m,n-\beta_1}_{0,\beta_1} x_{\beta_1}$

4: $\quad y_0 \leftarrow y_0 + z_0$.

5: $\quad$ **for** $\alpha_1 \leftarrow 0,m-1$ **do**

6: $\quad\quad z^{\alpha_1+1} \leftarrow \frac{(\alpha_1+1+\beta_1)(m-\alpha_1)}{(m+n+d-1-\alpha_1-\beta_1)(\alpha_1+1)} \left(E^{d-1,m-\alpha_1+1}\right)^t z^{\alpha_1}$

7: $\quad\quad y_{\alpha_1+1} \leftarrow y_{\alpha_1+1} + z^{\alpha_1+1}$

8: $\quad$ **end for**

9: **end for**

Now consider Algorithm 1. The bulk of the computation occurs in two operations, the multiplication of x_{β_1} by $M^{1,m,n-\beta_1}_{0,\beta_1}$ in line 3 and the degree elevation operation in line 6. Even if a naive matrix-vector algorithm is used, the n iterations of line 3 will cost a total of $n\mathcal{O}(n^2) = \mathcal{O}(n^3)$ operations. Now, each iteration of line 7 will require no more than $2\times|A^1_n| = \mathcal{O}(n)$ operations. Since line 7 will be executed $\mathcal{O}(n^2)$ times, the total cost will be $\mathcal{O}(n^3)$.

By using this algorithm recursively, the complexity for higher d will lead to an even more dramatic reduction in complexity over the standard matrix-vector algorithm, giving $\mathcal{O}(dn^{d+1})$ rather than $\mathcal{O}(n^{2d})$. Arguing inductively, it has just been demonstrated when $d = 2$. Suppose then that the complexity result holds for some $d-1$.

Then, the n iterations of line 3 will cost $n\mathcal{O}((d-1)n^d) = \mathcal{O}(dn^{d+1})$. The $\mathcal{O}(n^2)$ iterations of line 7 will cost $\mathcal{O}(n^2)\mathcal{O}(d|A_n^{d-1}|) = \mathcal{O}(dn^{d+1})$ also. This establishes the following complexity result.

Theorem 1 *Let $m_0 = \max\{m, n\}$ and $N = |A_{m_0}^d|$. Then the mass matrix $M^{d,m,n}$ may be applied to any vector in $\mathcal{O}((dm_0)^{d+1})$ operations.*

As Algorithm 1 stands, it also requires the computation of two binomial coefficients in the construction of z_0 at the beginning of each iteration. These can be incrementally computed as part of the recurrence. Initially, $\beta_1 = 0$, and the coefficient in line 3 reduces to $\frac{1}{m+n+d}$. Denote this coefficient by κ. Then, it is not hard to show that for successive steps of the algorithm,

$$\kappa^{\beta_1+1} = \frac{n-\beta_i}{m+n+d-1-\beta_1}\kappa^{\beta_i}.$$

With this slight refinement, the new algorithm is given in Algorithm 2.

4 Stiffness matrices

Fast application of the Laplace stiffness matrix follows from the sparse representation of differentiation given in Section 2 and the fast mass

Algorithm 2 Algorithm for product for $y = M^{d,m,n}x$, where $d \geq 2$.

Binomial coefficients are removed

$y \leftarrow 0$

$\kappa \leftarrow \frac{1}{m+n+d}$

for $\beta_1 \leftarrow 0,n$ **do**

  $z_0 \leftarrow \kappa M^{d-1,m,n-\beta_1}_{0,\beta_1} x_{\beta_1}$

  $\kappa \leftarrow \frac{n-\beta_1}{m+n+d-1-\beta_1}\kappa$

  $y_0 \leftarrow y_0 + z_0$.

  for $\alpha_1 \leftarrow 0,m-1$ **do**

    $z_{\alpha_1+1} \leftarrow \frac{(\alpha_1+1+\beta_1)(m-\alpha_1)}{(m+n+d-1-\alpha_1-\beta_1)(\alpha_1+1)} \left(E^{d-1,m-\alpha_1+1}\right)^t z_{\alpha_1}$

    $y_{\alpha_1+1} \leftarrow y_{\alpha_1+1} + z_{\alpha_1+1}$

  end for

end for

matrix algorithm just derived. Let S_d be a d-simplex, and define

$$k(u,v) = \int_{S_d} \nabla u \cdot \nabla v \, dx. \tag{17}$$

Considering $k : P_d^n \times P_d^m \to \mathbb{R}$, the stiffness matrix over S_d is then defined to be

$$K_{\alpha\beta}^{S_d,m,n} = k(B_\beta^n, B_\alpha^m).$$

Expressing the integrand of (17) in Cartesian components gives

$$\begin{aligned} k(u,v) &= \int_{S_d} \sum_{i=1}^{d} \frac{\partial u}{\partial x_i}\frac{\partial v}{\partial x_i}\,dx \\ &= \sum_{i=1}^{d} \int_{S_d} \frac{\partial u}{\partial x_i}\frac{\partial v}{\partial x_i}\,dx \\ &= \sum_{i=1}^{d} k_i(u,v), \end{aligned}$$

where

$$k_i(u,v) = \int_{S_d} \frac{\partial u}{\partial x_i}\frac{\partial v}{\partial x_i}\,dx = m\left(\frac{\partial u}{\partial x_i}, \frac{\partial v}{\partial x_i}\right). \tag{18}$$

The stiffness matrix, then, may be written as a sum

$$K^{S_d,m,n} = \sum_{i=1}^{d} K^{S_d,m,n,i},$$

where

$$K^{S_d,m,n,i}_{\alpha,\beta} = k_i(B^n_\beta, B^m_\alpha).$$

Evaluating $k_i(u,v)$ is expressed matrix-vector notation by

$$k_i(u,v) = (c^v)^t\, K^{T,m,n,i} c^u, \tag{19}$$

where $u = \sum_{\alpha\in A^n_d} c^u_\alpha B^n_\alpha$ and $v = \sum_{\beta\in A^m_d} c^v_\beta B^m_\alpha$.

Alternately, $k_i(u,v) = m\left(\frac{\partial u}{\partial x_i}, \frac{\partial v}{\partial x_i}\right)$. Let $\frac{\partial u}{\partial x_i} = \sum_{\alpha\in A^{n-1}_d} \hat{c}^u_\alpha B^{n-1}_\alpha$ and $\frac{\partial v}{\partial x_i} = \sum_{\alpha\in A^{m-1}_d} \hat{c}^v_\alpha B^{m-1}_\alpha$. This gives that

$$k_i(u,v) = (\hat{c}^v)^t\, M^{S_d,m-1,n-1} \hat{c}^u.$$

Finally, since $\hat{c}^u = D^{S_d,n,x_i} c^u$ and $\hat{c}^v = D^{S_d,n,x_i}$, this gives that

$$\begin{aligned} k_i(u,v) &= \left(D^{d,n,x_i}c^v\right)^t M^{S_d,m-1,n-1}\left(D^{d,n,x_i}c^u\right) \\ &= (c^v)^t\left(D^{d,n,x_i}\right)^t M^{S_d,m-1,n-1}D^{d,n,x_i}c^u. \end{aligned} \tag{20}$$

So then,

Proposition 4 *Each term of the stiffness matrix $K^{S_d,m,n,i}$ on a simplex S_d is factored into*

$$K^{S_d,m,n,i} = \left(D^{S_d,n,x_i}\right)^t M^{S_d,m-1,n-1}D^{S_d,n,x_i}. \tag{21}$$

Applying the element stiffness matrix may be accomplished by Algorithm 3.

Algorithm 3 Fast algorithm for computing the product $y = K^{S_d,m,n}x$.

1: $y \leftarrow 0$
2: **for** $i \leftarrow 1,d$ **do**
3: $\quad z_1 \leftarrow D^{S_d,n,i}x$
4: $\quad z_2 \leftarrow (|S_d|d!)\, M^{d,m-1,n-1}z_1$ using Algorithm 2
5: $\quad y \leftarrow y + \left(D^{S_d,m,i}\right)^t z_2$
6: **end for**

The loop in algorithm 3 requires the application of two derivative operators, each of which costs $(d+1)|A_d^{m_0}|) = \mathcal{O}(dm_0)$ operations, where $m_0 = \max(m,n)$. This is less than the cost of applying the mass matrix, even by Algorithm 2 for each of the d. Since the computation

is dominated by d mass matrix applications, the following complexity result holds.

Theorem 2 *Let $m_0 = \max\{m, n\}$ and $N = |A^d_{m_0}|$. Then a stiffness matrix $K^{S_d,m,n}$ may be applied to any vector in $\mathcal{O}((d^2 m_0)^{d+1}) = \mathcal{O}(d^2 N^{1+\frac{1}{d}})$ operations, or about d times the cost of a mass matrix.*

5 One-dimensional operators

While the one-dimensional operators do not admit dimensional recursion as in the previous sections, a different kind of structure can lead to reduced-complexity algorithms. While this low complexity, which depends on the fast Fourier transform, may not actually speed up calculation for practical polynomial degrees, it is still interesting to note the structure. Since stiffness matrices follow naturally from mass matrices as before, it is only necessary to handle the one-dimensional mass matrix.

In one dimension, it is easier to dispense with multiindices and use standard index notation, but with indices beginning with 0 rather than 1. With barycentric coordinates b_1 and b_2, and the Bernstein polynomials are

$$B^n_i = \binom{n}{i} b_1^{n-i} b_2^i.$$

On the interval $[0,1]$, $b_1 = x$ and $b_2 = 1 - x$. Specializing (9) to one dimension gives

$$M_{i,j}^{1,m,n} = \binom{m}{i}\binom{n}{j}\frac{(m+n-i-j)!(i+j)!}{(m+n+1)!}.$$

Define $(n+1) \times (n+1)$ diagonal matrices Δ^n by

$$\Delta_{ij}^n = \binom{n}{i}\delta_{ij},$$

having the binomial coefficients on the diagonal. Next, introduce the matrix

$$\tilde{M}_{ij}^{1,m,n} = \frac{(m+n-i-j)!(i+j)!}{(m+n+1)!}.$$

With these definitions, $M^{1,m,n}$ may be decomposed as

$$M^{1,m,n} = \Delta^m \tilde{M}^{1,m,n} \Delta^n. \tag{22}$$

This decomposition is interesting because of a special property of $\tilde{M}^{1,m,n}$.

Proposition 5 *The matrix $\tilde{M}^{1,m,n}$ has the Hankel property, i.e. is constant along antidiagonals. More precisely, for each $1 \leq i \leq m$ and $0 \leq j \leq n-1$,*

$$\tilde{M}_{i-1,j+1}^{1,m,n} = \tilde{M}_{i,j}^{1,m,n}.$$

Proof The proof is a simple calculation using the definition of $\tilde{M}^{1,m,n}$:

$$\begin{aligned}\tilde{M}^{1,m,n}_{i-1,j+1} &= \frac{(m+n+2-(i-1)-(j+1))!\,((i+1)+(j-1)-2)!}{(m+n)!} \\ &= \frac{(m+n+2-i-j)!\,(i+j-2)!}{(m+n)!} \\ &= \tilde{M}^{1,m,n}_{i,j},\end{aligned}$$

Hankel matrices are just Toeplitz matrices with the rows in reversed order. Just as Toeplitz matrices admit a circulant embedding and hence may be applied with the FFT [6], so may Hankel matrices. Therefore, with $N = \max\{m, n\}$, the matrix $\tilde{M}^{1,m,n}$ may be applied to any vector in $\mathcal{O}(N \log N)$ operations. Hence, $M^{1,m,n}$ may also be applied with the same complexity using the decomposition (22). Since differentiation also may be performed in linear time, this also provides $\mathcal{O}(N \log N)$ algorithms for stiffness matrix application.

While this observation is interesting, the practical value may be limited. The need for fast algorithms in one dimension is not so great as in two and three, and the FFT only will beat explicit matrix multiplication for N larger than commonly used in finite element calculation. Further, it is interesting to note that the fast algorithms in higher dimensions do not rely on reduced complexity for the one-dimensional algorithm.

While considering one-dimensional operators, it is also helpful to derive recurrence relations for the mass matrix entries so that higher-

dimensional codes do not need to explicitly construct and store these operators. From (9), it is clear that

$$M_{0,0}^{1,m,n} = \frac{1}{m+n+1}.$$

Recurring along the column $j = 0$,

$$\begin{aligned}\frac{M_{i+1,0}^{1,m,n}}{M_{i,0}^{1,m,n}} &= \frac{\binom{m}{i+1}\frac{(m+n-i-1)!(i+1)!}{(m+n+1)!}}{\binom{m}{i}\frac{(m+n-i)!i!}{(m+n+1)!}} \\ &= \frac{\binom{m}{i+1}}{\binom{m}{i}}\frac{(m+n-i-1)!}{(m+n-i)!}\frac{(i+1)!}{i!}\end{aligned}$$

so that

$$M_{i+1,0}^{1,m,n} = \left(\frac{m-i}{m+n-i}\right) M_{i,0}^{1,m,n}.$$

Now, recurring along a row with fixed i,

$$\begin{aligned}\frac{M_{i,j+1}^{1,m,n}}{M_{i,j}^{1,m,n}} &= \frac{\binom{m}{i}\binom{n}{j}\frac{(m+n-i-j-1)!(i+j+1)!}{(m+n+1)!}}{\binom{m}{i}\binom{n}{j+1}\frac{(m+n-i-j)!(i+j)!}{(m+n+1)!}} \\ &= \left(\frac{n-j}{j+1}\right)\left(\frac{i+j+1}{m+n-i-j}\right),\end{aligned}$$

giving

$$M_{i,j+1}^{1,m,n} = \left(\frac{n-j}{j+1}\right)\left(\frac{i+j+1}{m+n-i-j}\right) M_{i,j}^{1,m,n}.$$

This allows the application of $M^{1,m,n}$ to be applied to a vector in $\mathcal{O}(mn)$ operations using standard matrix-vector logic, but computing the matrix-entries as the loops unfold.

Algorithm 4 Matrix-free computation of $y = M^{1,m,n}x$. μ contains the first entry of the current row of $M^{1,m,n}$, and ν is updated along the row.

$\mu \leftarrow \frac{1}{m+n+1}$
for $i \leftarrow 0,m$ **do**
  $\nu \leftarrow \mu$
  for $j \leftarrow 0,n$ **do**
    $y_i \leftarrow y_i + \nu x_j$
    $\nu \leftarrow \left(\frac{n-j}{j+1}\right)\left(\frac{i+j+1}{m+n-i-j}\right)\nu$
  end for
  $\mu \leftarrow \left(\frac{m-i}{m+n-i}\right)\mu$
end for

6 Numerical results

The two-dimensional algorithms for mass and stiffness matrices were implemented in C++ using the DOLFIN library [11] for finite element meshes and linear algebra. A 128x128 rectangular mesh of the unit square was divided into right triangles, and Bernstein polynomials of varying degree were used as a local basis on each cell, pieced together in a continuous fashion across elements.

Starting from an array storing global degrees of freedom, the values were scattered to each cell to give local B-form coefficients. On each cell, the local algorithm derived here was applied, and the results assembled back together in a global vector. For each polynomial

degree, the algorithm was run ten times, and the average time was reported. All calculations were performed on a Macintosh laptop with dual-core Intel processor and 2GB of RAM.

To confirm the predicted theoretical scaling, the local dimension of the polynomial space was plotted against the timings for the mass matrix on a log scale, and a linear best fit was performed. The slope of this linear fit was about 1.56. Notice also in Table 1 that the stiffness matrix costs somewhat less than twice the mass matrix. This is due to the cost of scattering the global degrees of freedom and gathering the results back together, which has the exact same cost for mass and stiffness matrices.

7 Conclusions

This paper has begun the study of fast finite element algorithms for polynomial spaces expressed in B-form. The critical step, multiplication by a mass matrix, is handled by a dimensionally recursive algorithm, and opens up fast algorithms for stiffness matrices as well. Obtaining high performance, however, will require considerably more care than for standard dense matrix operations. In particular, the complicated loop structure for degree elevation and differentiation

Table 1. Timings for computing global matrix-vector product for mass and stiffness matrices using Algorithms 2 and 3, respectively. All calculations are performed on a mesh of 32768 triangles. The first column specifies the polynomial degree n, and the second the local dimension of the polynomial space, $\frac{(n+1)(n+2)}{2}$. The third and fourth columns report the average time for the mass and stiffness matrices to be computed.

degree n	$\dim P_2^n$	Mass time	Stiffness time
1	3	1.16E-002	1.39E-002
2	6	3.08E-002	3.64E-002
3	10	6.69E-002	8.74E-002
4	15	1.09E-001	1.68E-001
5	21	2.07E-001	3.06E-001
6	28	3.18E-001	5.41E-001
7	36	5.26E-001	8.51E-001
8	45	7.38E-001	1.22E+000
9	55	1.06E+000	1.62E+000
10	66	1.35E+000	2.34E+000

suggests that straightline code generation for particular polynomial degrees might be effective

Besides providing fast algorithms for a well-known simplicial basis that can be used in standard C^0 finite elements, this work also opens several interesting possibilities. These include application in the context of simplicial splines and the exterior calculus bases in [2]. Also,

to move beyond constant-coefficient forms, adaptation to quadrature-based methods will be required. This will involve considering either some kind of modified sum-factorization with tensor-product quadratures or else finding mutual symmetry between the Bernstein basis and other quadrature rules. In the future, these topics will be studied, as well as matrix structure and precoditioning of operators in B-form.

References

1. Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. Finite element exterior calculus, homological techniques, and applications. *Acta Numer.*, 15:1–155, 2006.
2. Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. Geometric decompositions and local bases for spaces of finite element differential forms. *Comput. Methods Appl. Mech. Engrg.*, 198:1660–1672, 2009.
3. Claudio Canuto, M. Yousuff Hussaini, Alfio Quarteroni, and Thomas A. Zang. *Spectral methods in fluid dynamics*. Springer Series in Computational Physics. Springer-Verlag, New York, 1988.
4. Daniele Funaro. *Spectral elements for transport-dominated equations*, volume 1 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin, 1997.
5. F. X. Giraldo and M. A. Taylor. A diagonal-mass-matrix triangular-spectral-element method based on cubature points. *J. Engrg. Math.*, 56(3):307–322, 2006.

6. Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.
7. J. S. Hesthaven and T. Warburton. Nodal high-order methods on unstructured grids. I. Time-domain solution of Maxwell's equations. *J. Comput. Phys.*, 181(1):186–221, 2002.
8. T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput. Methods Appl. Mech. Engrg.*, 194(39-41):4135–4195, 2005.
9. George Em Karniadakis and Spencer J. Sherwin. *Spectral/hp element methods for computational fluid dynamics*. Numerical Mathematics and Scientific Computation. Oxford University Press, New York, second edition, 2005.
10. Ming-Jun Lai and Larry L. Schumaker. *Spline functions on triangulations*, volume 110 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2007.
11. A. Logg and G. N. Wells. DOLFIN: Automated finite element computing. *Submitted*, 2009.
12. Steffan Petersen, Daniel Dreyer, and Otto von Estorff. Assessment of finite and spectral element shape functions for efficient iterative simulations of interior acoustics. *Comput. Methods Appl. Mech. Engrg.*, 195:6463–6478, 2006.
13. S. Fernndez-Mndez R. Sevilla and A. Huerta. Nurbs-enhanced finite element method (nefem). *International Journal for Numerical Methods in Engineering*, 76(1):56–83, 2008.
14. S. Fernndez-Mndez R. Sevilla and A. Huerta. Nurbs-enhanced finite element method (nefem) for euler equations. *International Journal for Numerical Methods in Fluids*, 57(9):1051–1069, 2008.

15. Larry L. Schumaker. Computing bivariate splines in scattered data fitting and the finite-element method. *Numer. Algorithms*, 48(1-3):237–260, 2008.

16. O. C. Zienkiewicz and R. L. Taylor. *The finite element method. Vol. 3.* Butterworth-Heinemann, Oxford, fifth edition, 2000. Fluid dynamics.