Efficient L^2 projection on simplices using Bernstein polynomials

Jesse Chan

Contents

Abstract

Alternative formula for mass inversion. GPU-accelerated versions of Ainsworth and Kirby Duffy transforms. Additional polynomial multiplication-based approach.

1 Introduction

[1]

2 Bernstein-Bezier bases

3 Fast L^2 projections with Bernstein polynomials

Involves three steps - evaluation in an enriched representation (either at quadrature points or in a higher degree polynomial basis), scaling by the weight, and projection down to polynomials of degree N.

3.1 Collapsed-coordinate quadratures

Fast evaluation at Duffy points [2, 3, 4].

Component-wise scaling by weight evaluated at collapsed-coordinate quadrature points.

3.2 Polynomial multiplication

For non-constant coefficients, DG requires being able to deal with polynomial multiplication and projection onto lower-dimensional subspaces. Multiplying polynomials together may be done using a discrete convolution and polynomial multiplication (J. Sanchez-Reyes 2003). The projection operator may be derived by noting that degree elevation operators are diagonal when transformed to a modal basis.

Rescaling by binomial coefficients results in the unscaled Bernstein basis. Polynomial multiplication is then equivalent to discrete convolution of the scaled binomial coefficients.

Quadrature-free strategy for nonlinear volume terms: polynomial multiplication + projection.

1. Polynomial multiplication of two BB basis functions representable as coefficient scaling, N_p scalar multiplications and storage of N_p coeffs, and another coefficient scaling.
2. To reduce local memory costs, process coeffs for fg over one or more $(d-1)$ dimensional layers.
3. Store ids and load a triangular number of loads.

3.3 Inversion of modally diagonal matrices

Any modally diagonal matrix $\boldsymbol{M}$ can be represented in the form

$$\boldsymbol{M}_N^{-1}\boldsymbol{M}_{N,M} = \sum_{j=0}^{N} c_j E^N_{N-j}\left(E^M_{N-j}\right)^T.$$

This property was shown for the polynomial projection matrix by Waldron in [5, 6]. We give an alternative proof of this below where $\boldsymbol{M}$ is any modally diagonal matrix.

The constants c_j may be computed through the solution of an $(N+1)\times(N+1)$ matrix system, using the fact that upon transformation to a modal basis, E^N_{N-j} is a diagonal matrix of ones and zeros, while E^M_{N-j} is a diagonal matrix with entries

$$\frac{\lambda_i^{N-j}}{\lambda_i^M}, \qquad i = 0,\ldots,N.$$

This may be factored into an application of E^M_N, then an application of

$$\begin{aligned}\sum_{j=0}^{N} c_j E^N_{N-j}\left(E^N_{N-j}\right)^T &= c_0\boldsymbol{I} + c_1 E^N_{N-1}\left(E^N_{N-1}\right)^T + c_2 E^N_{N-1}E^{N-1}_{N-2}\left(E^{N-1}_{N-2}\right)^T\left(E^N_{N-1}\right)^T + \ldots\\ &= c_0\boldsymbol{I} + c_1 E^N_{N-1}\left(I + \frac{c_2}{c_1}E^{N-1}_{N-2}\left(I+\ldots\right)\left(E^{N-1}_{N-2}\right)^T\right)\left(E^N_{N-1}\right)^T.\end{aligned}$$

This may be applied in two sweeps of length N, using in-place updates to memory. Unfortunately, for shared-memory parallelization, this will require synchronizations between each application of each matrix.

The cost of applying $\left(E^M_N\right)^T$ is the application of $(M-N)$ sparse degree elevation operations, each of which is $O(M^d)$ cost. Assuming $M \approx N$ (it is reasonable to match the order of the data with the order of approximation), this gives $O(N^{d+1})$ cost.

When applying the projection operator, since each operation is $O(N^3)$ and we apply $O(N)$ total operations, we have an $O(N^{d+1})$ overall cost.

This can also be used to apply the inverse mass matrix since it is diagonal under the transformation T.

4 Numerical experiments

4.1 Expansion kernels

4.1.1 Polynomial multiplication

4.1.2 Collapsed-coordinate quadrature

4.2 Mass matrix inversion kernel

4.3 Monolithic kernel

5 Application to Weight-adjusted Discontinuous Galerkin (WADG) methods

Numerical experiment: time to solution for order 7 and 8 using both NDG-WADG and BB-WADG.
[7, 8]

References

[1] Jesse Chan and T Warburton. GPU-accelerated Bernstein-Bezier discontinuous Galerkin methods for wave problems. *arXiv preprint arXiv:1512.06025*, 2015.

[2] Mark Ainsworth, Miangaly Gaelle Andriamaro, and Oleg Davydov. Bernstein-Bézier finite elements of arbitrary order and optimal assembly procedures. *SIAM Journal on Scientific Computing*, 33(6):3087–3109, 2011.

[3] Robert C Kirby. Fast simplicial finite element algorithms using Bernstein polynomials. *Numerische Mathematik*, 117(4):631–652, 2011.

[4] Robert C Kirby and Kieu Tri Thinh. Fast simplicial quadrature-based finite element operators using Bernstein polynomials. *Numerische Mathematik*, 121(2):261–279, 2012.

[5] Shayne Waldron. On the Bernstein–Bézier form of Jacobi polynomials on a simplex. *Journal of Approximation Theory*, 140(1):86–99, 2006.

[6] Shayne Waldron. Computing orthogonal polynomials on a triangle by degree raising. *Numerical Algorithms*, 42(2):171–179, 2006.

[7] Jesse Chan, Russell J Hewett, and T Warburton. Weight-adjusted discontinuous Galerkin methods: wave propagation in heterogeneous media. *arXiv preprint arXiv:1608.01944*, 2016.

[8] Jesse Chan, Russell J Hewett, and T Warburton. Weight-adjusted discontinuous Galerkin methods: curvilinear meshes. *arXiv preprint arXiv:1608.03836*, 2016.