Bernstein polynomials and finite element algorithms

Robert Kirby[1]

[1]Baylor University

14 July 2014

BAYLOR UNIVERSITY

Problems for high order

Very large element matrices

$$A_{ij} = \int_K w \nabla \phi_i \cdot \nabla \phi_j \, dx$$

	Standard	Tensor product
Basis size:	$\mathcal{O}(n^d)$	
Element matrix size:	$\mathcal{O}(n^{2d})$	
Cost of local matvec:	$\mathcal{O}(n^{2d})$	$\mathcal{O}(n^{d+1})$

BAYLOR UNIVERSITY

But how do we go fast?

Tensor Products

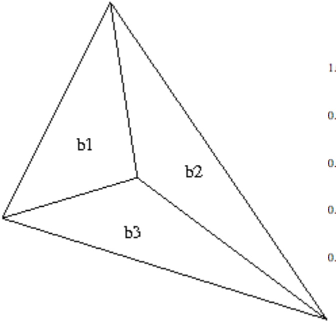
- Sum factorization $\leftrightarrow$ fast matvecs
- Operation count: $\mathcal{O}(n)$ per entry, $\mathcal{O}(n^{d+1})$ total
- Memory usage: $\mathcal{O}(n^d)$

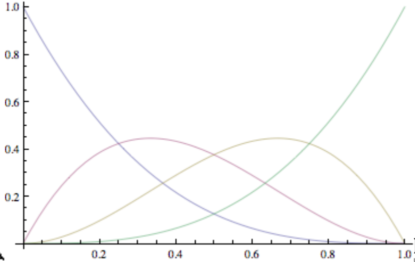
Simplex?

- Collapsed-coordinates: Karniadakis & Sherwin for H^1
- General elements: FIAT (RCK), FEMSTER (White, Castillo)

Bernstein polynomials

$$\left\{ \binom{n}{\alpha} \prod_i b_i^{\alpha_i} \right\}_{|\alpha|=n}$$





Differentiation

It's *sparse* in B-form

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

Differentiation

It's *sparse* in B-form

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

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

Differentiation

It's *sparse* in B-form

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

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

$D \leftrightarrow$ sparse matrix with at most $d+1$ nonzeros per row

BAYLOR UNIVERSITY

Bernstein polynomials

Some history

- Approximation theory: Bernstein, quasi-interpolants, splines
- CAGD: stable and fast algorithms for curves/surfaces
- Finite element analysis?
  - Peterson *et. al.*
  - Schumaker (splines)
  - NURBS - Hughes *et. al.*
  - FEEC (Arnold, Falk, Winther)
  - RCK & Ainsworth

Duffy transforms and tensor products

$[0,1]^d \to d$-simplex

Define inductively:

$$\lambda_0 = t_1$$

$$\lambda_i = t_{i+1}\left(1 - \sum_{j=0}^{i-1} \lambda_j\right)$$

$$\lambda_n = 1 - \sum_{j=0}^{n-1} \lambda_j$$

Tensorialize Bernstein

With

$$\mathbf{x}(\mathbf{t}) = \sum_{i=0}^{n} \mathbf{x}_i \lambda_i(\mathbf{t}),$$

we have

$$B^r_\alpha(\mathbf{x}(\mathbf{t})) = \prod_{i=0}^{n} B_{\alpha_i}^{r-\sum_{j=0}^{i}\alpha_j}(t_i)$$

What operations are fast?

Evaluation

Given $u = \sum_{|\alpha|=n} u_\alpha B^n_\alpha$,

$$\{u_\alpha\}_\alpha \mapsto \{u(\xi_q)\}_q ,$$

when $\{\xi_q\}_q$ are Stroud points. Requires $\mathcal{O}(n^{d+1})$ and *no* pre-tabulated data.

Moment computation

Given $\{f_q = f(\xi_q)\}_q$

$$\{f_q\} \mapsto \left\{ \int_T f B^n_\alpha dx \right\}_\alpha$$

requires $\mathcal{O}(n^{d+1})$ and *no* pre-tabulated data.

Derivatives?

Evaluate/integrate followed by *short* linear combinations!

Optimal-complexity assembly

Constant-order work per entry

Since $B^r_\alpha B^s_\beta = \frac{\binom{\alpha+\beta}{\alpha}}{\binom{r+s}{r}} B^{r+s}_{\alpha+\beta}$, so matrix formation

$$M_{\alpha\beta} = \int_T f B^r_\alpha B^s_\beta$$

just requires (plus arithmetic/bookkeeping) all moments

$$\left\{ \int_T f B^{r+s}_\gamma dx \right\}_\gamma,$$

The de Rham complex

FEEC (Arnold, Falk, Winther)

Basis functions for $P_n^- \Lambda^1$: $B_\alpha^{n-1}\phi_{ij}$
Basis functions for $P_n^- \Lambda^2$: $B_\alpha^{n-1}\phi_{ijk}$, where

$$\phi_{ij} = b_i db_j - b_j db_i$$
$$\phi_{ijk} = b_i db_j \wedge db_k - b_j db_i \wedge db_k + b_k d\lambda_i \wedge db_j$$

Convert to Bernstein form

Short linear combination

$$
\begin{aligned}
B^{n-1}_\alpha \phi_{ij} &= b_i B^{n-1}_\alpha db_j - b_j B^{n-1}_\alpha db_i \\
&= b_i \frac{(n-1)!}{\alpha!} \mathbf{b}^\alpha_d db_j - b_j \frac{(n-1)!}{\alpha!} \mathbf{b}^\alpha_d db_i \\
&= \frac{(n-1)!}{\alpha!} \mathbf{b}^{\alpha+e_i}_d db_j - \frac{(n-1)!}{\alpha!} \mathbf{b}^{\alpha+e_j}_d db_i \\
&= \frac{\alpha_i + 1}{n} B^n_{\alpha+e_i} db_j - \frac{\alpha_j + 1}{n} B^n_{\alpha+e_j} db_i
\end{aligned}
$$

Algorithms

Conversion

- Each k-form basis function requires $k+1$ Bernstein polynomials
- Operator formation/application reuses fast evaluation/integration kernels for Bernstein
- Optimal complexity for $H(\mathrm{div})$ and $H(\mathrm{curl})$.

But I don't like $P_n^- \Lambda^k$!

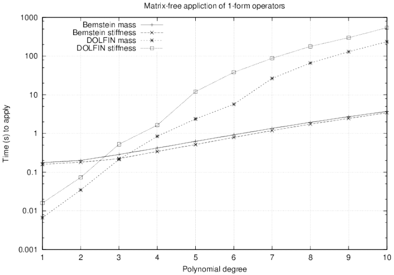
What about $P_n \Lambda^k$?

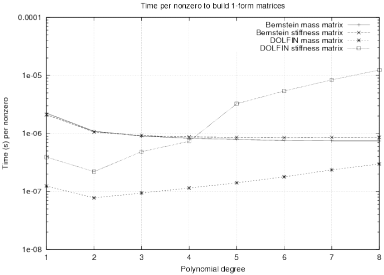
"Second-kind" basis functions look like:

$$B_\alpha^r \psi_\sigma^{\alpha, f, T}$$

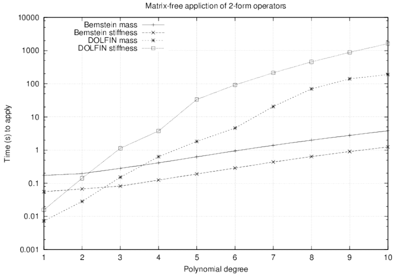
Shorter linear combinations, but more geometric data to load.
Won't discuss more here.

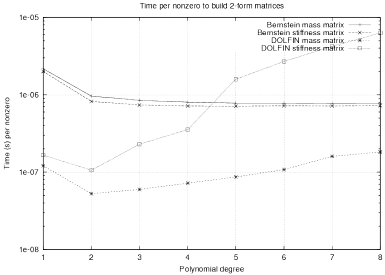
1-form action and per-nonzero build time





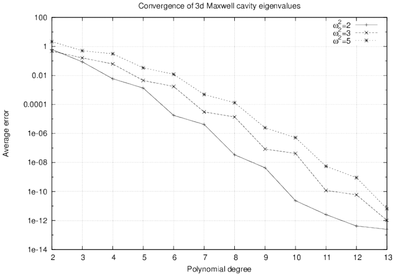
2-form action and per-nonzero build time

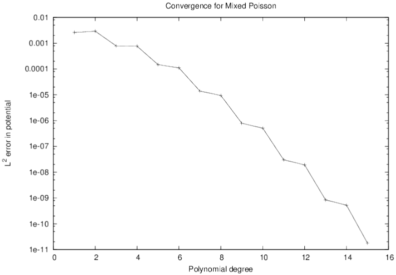




Accuracy?

Maxwell cavity eigenvalue and mixed Poisson error on unit cube meshed into six tetrahedra:





Weak form

Elementwise IBP

$$\sum_e \left[(u_{h,t}, w_h)_e - (F(u_h), \nabla w_h)_e + \langle \widehat{F} \cdot n, w_h \rangle_{\partial e} \right] = 0$$

Can also consider "strong DG" (Hesthaven/Warburton)

What does it cost?

$$\sum_e \left[(u_{h,t}, w_h)_e - (F(u_h), \nabla w_h)_e + \langle \widehat{F} \cdot n, w_h \rangle_{\partial e} \right] = 0$$

Elementwise convection term

- Evaluate u_h at QP: $\mathcal{O}(n^{d+1})$
- Evaluate $F(u_h)$ at QP: $\mathcal{O}(n^d)$
- Moment calculation: $\mathcal{O}(n^{d+1})$

BAYLOR UNIVERSITY

What does it cost?

$$\sum_e \left[(u_{h,t}, w_h)_e - (F(u_h), \nabla w_h)_e + \langle \widehat{F} \cdot n, w_h \rangle_{\partial e} \right] = 0$$

Boundary flux term

- Evaluate u_h at boundary QP: $\mathcal{O}(n^d)$
- Riemann solve at each QP: $\mathcal{O}(n^{d-1})$
- Boundary moment computation: $\mathcal{O}(n^d)$.

What does it cost?

$$\sum_e \left[(u_{h,t}, w_h)_e - (F(u_h), \nabla w_h)_e + \langle \widehat{F} \cdot n, w_h \rangle_{\partial e} \right] = 0$$

Mass inversion

- Cholesky: $\mathcal{O}(n^{3d})$ startup plus $\mathcal{O}(n^{2d})$ per cell
- CG (Fast matvec plus neat theorem): $\mathcal{O}(n^{d+2})$ per cell
- Need a fast algorithm, or this is the bottleneck!

All about the Bernstein mass matrix

Positive operators

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

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

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

2d mass matrices

Positive and *structured*

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

Fast algorithm [RCK'11]

Two facts

- Each block a (scaled) lower-dimensional mass matrix
- Blocks in a column are related by (sparse) degree elevation

Fast algorithm [RCK'11]

Two facts

- Each block a (scaled) lower-dimensional mass matrix
- Blocks in a column are related by (sparse) degree elevation

Algorithmic result

- $x \rightarrow M^{d,n,n}x$ requires $\mathcal{O}(dn^{d+1})$ complexity rather than $\mathcal{O}(n^{2d})$
- Allows fast matrix-free Krylov methods.

BAYLOR UNIVERSITY

Interesting spectrum

Theorem (RCK and Kieu)

The eigenvalues of $M^{d,n,n}$ are

$$\lambda_{i,n,d} = \frac{(n!)^2}{(n+d+i)!\,(n-i)!}, \quad 0 \leq i \leq n$$

The multiplicity of $\lambda_{i,n,d}$ is $\binom{d+i-1}{d-1}$.
For each $\lambda_{i,n,d}$, the eigenspace is spanned by B-form coefficients for $P_i \perp P_{i-1}$

BAYLOR UNIVERSITY

How'd you get that?

The Bernstein-Durrmeyer operator

$$D_n(f) = \sum_{|\alpha|=n} \frac{(f, B_\alpha^n)}{(B_\alpha^n, B_\alpha^n)} B_\alpha^n$$

See [Derriennic85, Farouki/Goodman/Sauer83]: the spectrum of the B-D operator is already known!

Fast inversion (a sketch)

Solve $Mx = y$ in $\mathcal{O}(n^{d+1})$

- Use "blockwise" Gaussian elimination:

$$m_{ij} M^{d-1,n-i,n-j} - \frac{m_{i0} m_{0j}}{m_{00}} M^{d-1,n-i,n} \left(M^{d-1,n,n} \right)^{-1} M^{d-1,n,n-j},$$

becomes, with Bernstein magic

$$m_{ij} M^{d-1,n-i,n-j} - \frac{m_{i0} m_{0j}}{m_{00}} M^{d-1,n-i,n-j} = \left(m_{ij} - \frac{m_{i0} m_{0j}}{m_{00}} \right) M^{d-1,n-i,n-j}$$

- "Auxilliary" matrix is scaled 1d mass matrix.
- Manipulate RHS by elevation + axpy.

What we've seen

No new discretizations

- Bernstein polynomials: new bases for old spaces
- Optimal complexity evaluation/moment/assembly algorithms on simplices
- Gets de Rham complex, (maybe) DG right
- Can we get Hermite, splines, etc? Elliptic DG?

To-do list

Math

- Tool in other discretizations
- *Stable* fast mass inversion
- Preconditioning

Code

- Fine-grained parallelism: GPU/MIC/etc
- FEniCS: polyalgorithmic code generation?