Accelerated High-order finite element Assembly (A-HA!)

Chris Eldred, Manel Tayachi, Thomas Dubos, Evaggelos Kritsikis and Fabrice Voitus

August 25th, 2016

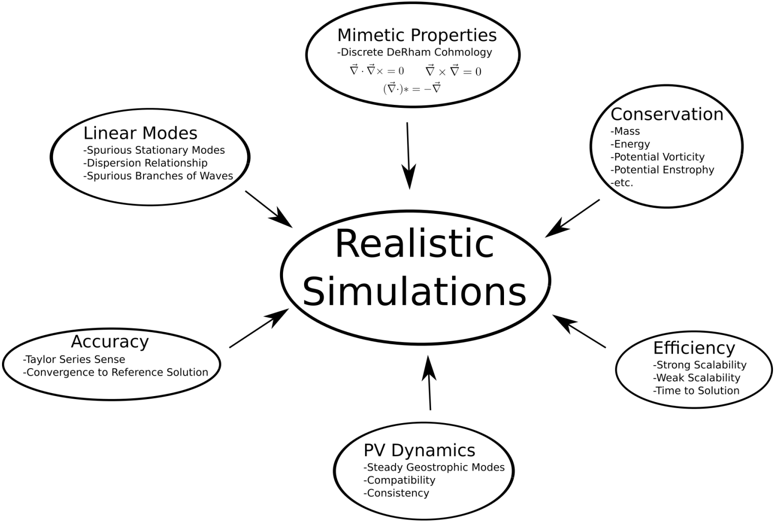
LMD

UNIVERSITÉ PARIS 13

METEO FRANCE

Introduction

(Incomplete) List of Desirable Model Properties



Non-Canonical Hamiltonian Dynamics

Evolution of an arbitrary functional $\mathcal{F} = \mathcal{F}[\vec{x}]$ is governed by:

$$\frac{d\mathcal{F}}{dt} = \{\frac{\delta\mathcal{F}}{\delta\vec{x}}, \frac{\delta\mathcal{H}}{\delta\vec{x}}\} \tag{1}$$

with Poisson bracket $\{,\}$ antisymmetric (also satisfies Jacobi):

$$\{\frac{\delta\mathcal{F}}{\delta\vec{x}}, \frac{\delta\mathcal{G}}{\delta\vec{x}}\} = -\{\frac{\delta\mathcal{G}}{\delta\vec{x}}, \frac{\delta\mathcal{F}}{\delta\vec{x}}\} \tag{2}$$

Also have Casimirs $\mathcal{C}$ that satisfy:

$$\{\frac{\delta\mathcal{F}}{\delta\vec{x}}, \frac{\delta\mathcal{C}}{\delta\vec{x}}\} = 0 \quad \forall\mathcal{F} \tag{3}$$

Neatly encapsulates conservation properties ($\mathcal{H}$ and $\mathcal{C}$).

General Formulation for Mimetic Discretizations: Primal deRham Complex

$$\begin{tikzcd}
\mathbb{W}_0 \arrow[r, "\overset{d}{\vec{\nabla}}"] & \mathbb{W}_1 \arrow[r, "\overset{d}{\vec{\nabla}\times}"] \arrow[l, dashed, bend left, "\underset{\delta}{\vec{\nabla}\cdot}"] & \mathbb{W}_2 \arrow[r, "\overset{d}{\vec{\nabla}\cdot}"] \arrow[l, dashed, bend left, "\underset{\delta}{\vec{\nabla}\times}"] & \mathbb{W}_3 \arrow[l, dashed, bend left, "\underset{\delta}{\vec{\nabla}}"]
\end{tikzcd}$$

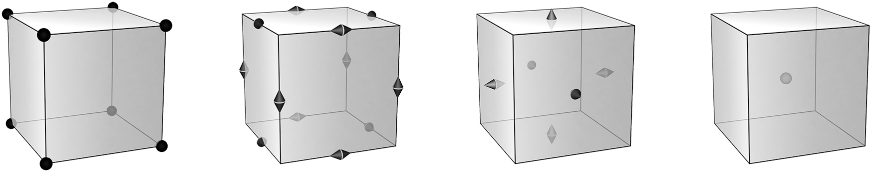
$$(da^k, b^{k+1}) = (a^k, \delta b^{k+1})$$

$$\delta = *d*$$

$$\nabla^2 = d\delta + \delta d$$

$$\vec{\nabla} \cdot \vec{\nabla} \times = 0 = \vec{\nabla} \times \vec{\nabla}$$

$$dd = 0 = \delta\delta$$



$\mathbb{W}_0$ $\mathbb{W}_1$ $\mathbb{W}_2$ $\mathbb{W}_3$

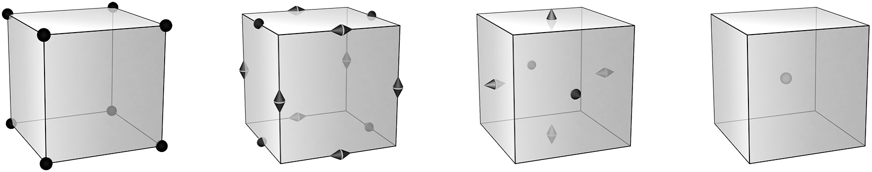
General Approach to Mimetic Galerkin Spaces

Mimetic Spaces

$$\text{Select 1D Spaces } \mathcal{A} \text{ and } \mathcal{B} \text{ such that } : \mathcal{A} \xrightarrow{\frac{d}{dx}} \mathcal{B} \tag{4}$$

- Use tensor products to extend to n-dimensions
- Works for ANY set of spaces $\mathcal{A}$ and $\mathcal{B}$ that satisfy this property (mimetic finite elements use P_n and $P_{DG,n-1}$)
- Mimetic spectral element, Mimetic isogeometric methods (B-splines) all fall under this framework
- We are also exploring (not shown) alternative choices of $\mathcal{A}$ and $\mathcal{B}$ which are guided by linear mode properties and coupling to physics/tracer transport
- See Hiemstra et. al 2014 (and references therein)

Overview of 3D Spaces



$\mathbb{W}_0$ $\mathbb{W}_1$ $\mathbb{W}_2$ $\mathbb{W}_3$

$$\mathbb{W}_0 \xrightarrow{\vec{\nabla}} \mathbb{W}_1 \xrightarrow{\vec{\nabla}\times} \mathbb{W}_2 \xrightarrow{\vec{\nabla}\cdot} \mathbb{W}_3$$

$\mathbb{W}_0 = \mathcal{A} \otimes \mathcal{A} \otimes \mathcal{A} = H_1 =$ Continuous Galerkin
$\mathbb{W}_1 = (\mathcal{B} \otimes \mathcal{A} \otimes \mathcal{A})\hat{i} + \ldots = H(curl) =$ Nedelec
$\mathbb{W}_2 = (\mathcal{A} \otimes \mathcal{B} \otimes \mathcal{B})\hat{i} + \ldots = H(div) =$ Raviart-Thomas
$\mathbb{W}_3 = \mathcal{B} \otimes \mathcal{B} \otimes \mathcal{B} = L_2 =$ Discontinuous Galerkin

Assembly and Operator Action Algorithms and Results

Standard Assembly Algorithm

Consider mass matrix using H^1 elements:

$$\int_\Omega u(x,y,z)\; v(x,y,z) d\Omega$$

On each element:

- Loop over u
- Loop over v
- Loop over quadrature points
- Compute $\hat{u}_{n,q} \hat{v}_{m,q} w_q |J|_q$

With $u = v = P^n$, in 3D, and Gaussian quadrature, costs $\mathbf{O(n^9)}$ (n is the order of the finite element space)

Can we do better?

Tensor Product Assembly Algorithm

Recognize that

$$u(x,y,z) = u^x(x)u^y(y)u^z(z)$$

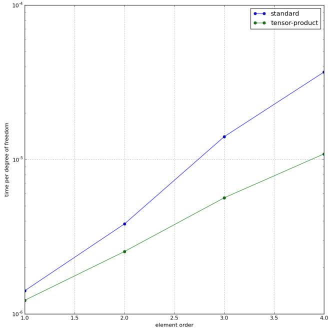
(and similarly for v). Therefore the integral from before can be factored as

$$\int_x u^x(x)v^x(x) \int_y u^y(y)v^y(y) \int_z u^z(z)v^z(z)d\Omega$$

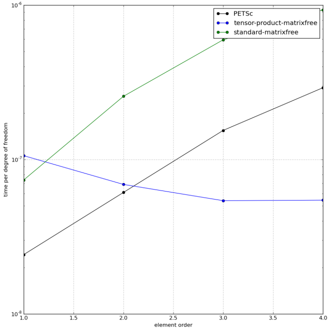
This is **sum-factorization**

With $u = v = P^n$, in 3D, and Gaussian quadrature, costs $\mathbf{O(n^7)}$

Results (Weighted Mass Matrix for H^1)



Assembly

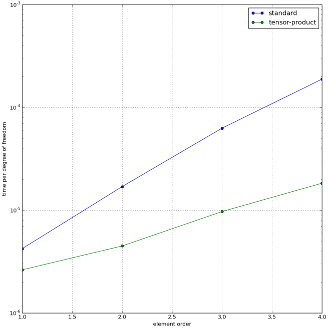


Operator Action

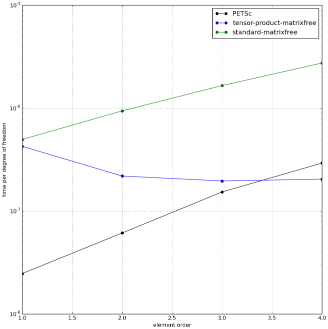
$$\int \phi uv \text{ where } u, v \in H^1 \text{ and } \phi \in L_2$$

13824 dofs, single process, on laptop

Results (Weighted Laplacian for H^1)



Assembly



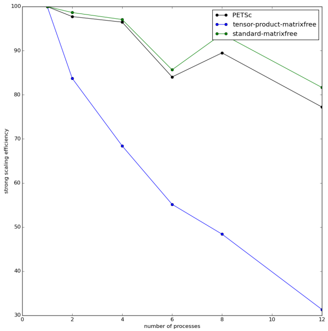
Operator Action

$$\int \phi \vec{\nabla} u \cdot \vec{\nabla} v \text{ where } u, v, \phi \in H^1$$

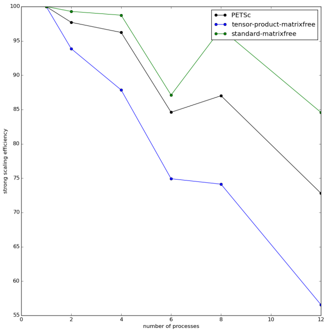
13824 dofs, single process, on laptop

Strong Scaling Results (Operator Action)

H^1 Weighted Mass

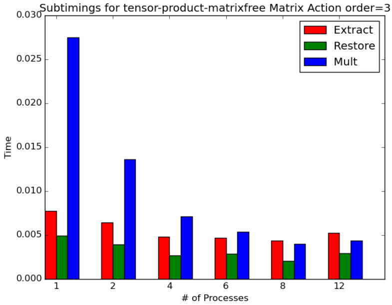


H^1 Weighted Laplacian

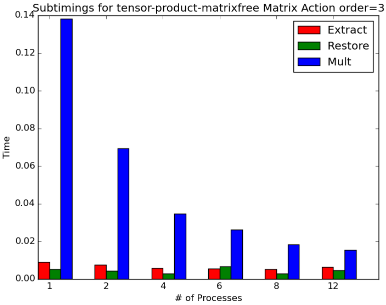


Ranges from 110592 dofs per process to 9216 dofs per process
Tests done on a SINGLE node with 2 6-core Westmere processors

Timing Breakdowns for Operator Action (Tensor Product Only)



H^1 Weighted Mass



H^1 Weighted Laplacian

Ranges from 110592 dofs per process to 9216 dofs per process
Tests done on a SINGLE node with 2 6-core Westmere processors

Conclusions

Summary

Conclusions

- Structure preserving numerical schemes can be derived from the combination of a **Mimetic Discretization Method** and a **Hamiltonian Formulation**
- Exploiting tensor product structure (sum factorization) is key to good performance
- For finite elements, the superior choice appears to be operator action rather than assembly, ASSUMING effective preconditioners can be found

Next Steps

- Look at ways to improve strong scaling (reduced communication, overlapping computation and communication)
- "Matrix-free" preconditioners- geometric multigrid (h + p), low order matrices

Extra Slides

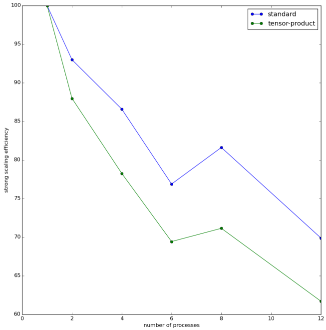
Future Work

Further Possible Performance Enhancements

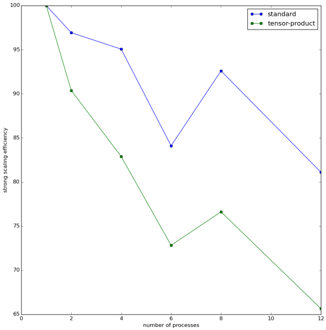
- Overlap of communication and computation
- PyOP2 style redundant computation (no off-process matrix entry creation, similarly for operator action)
- Vectorization across elements
- Optimized tensor contraction routines
- Specialized matrix data structures + matrix-free products for structured grid tensor product finite elements (reduced data movement, increased vectorization potential)
- Specialized vector data structure and insertion/extraction routines (reduced data movement, increased vectorization potential)
- Shared memory features through MPI-3 (windows, neighborhood collectives)

Strong Scaling Results (Assembly)

H^1 Weighted Mass



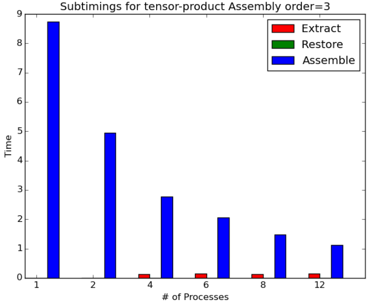
H^1 Weighted Laplacian



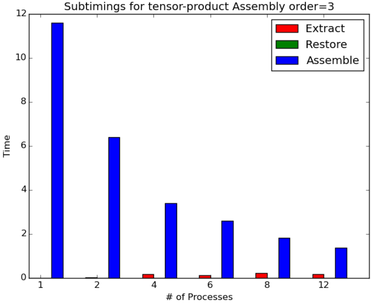
Ranges from 110592 dofs per process to 9216 dofs per process

Tests done on a SINGLE node with 2 6-core Westmere processors

Timing Breakdowns for Assembly (Tensor Product Only)



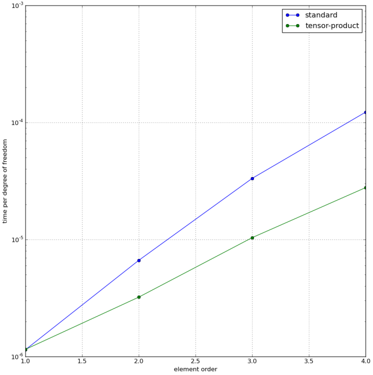
H^1 Weighted Mass



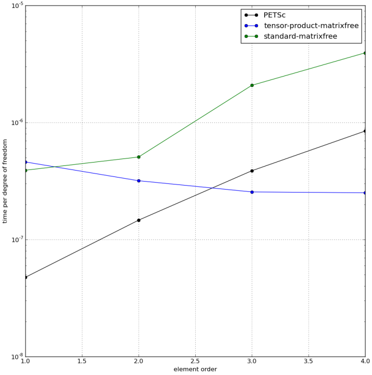
H^1 Weighted Laplacian

Ranges from 110592 dofs per process to 9216 dofs per process
Tests done on a SINGLE node with 2 6-core Westmere processors

Results (Mass Matrix for $H(div)$)



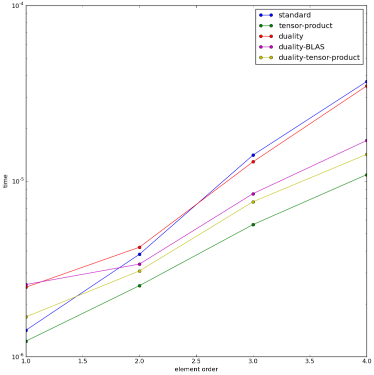
Assembly



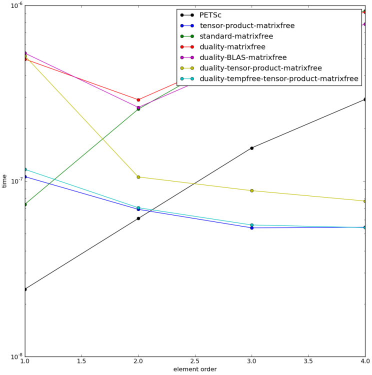
Operator Action

$$\int \vec{u} \cdot \vec{v} \text{ where } u, v \in H(div)$$

Duality Results (algorithm from Kirby 2014)



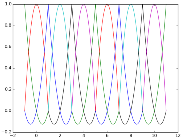
Assembly



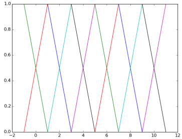
Operator Action

$\int \phi uv$ where $u, v \in H^1$ and $\phi \in L_2$

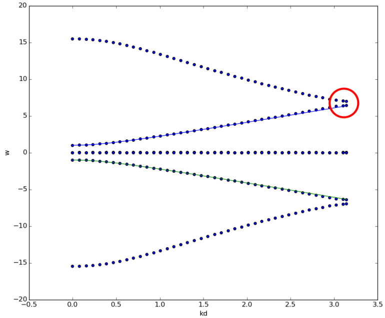
$P_2 - P_{1,DG}$ Dispersion Relationship



$\mathcal{A} = H_1$ Space (1D)



$\mathcal{B} = L_2$ Space (1D)

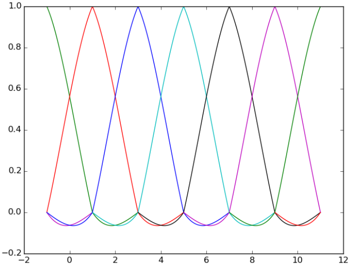


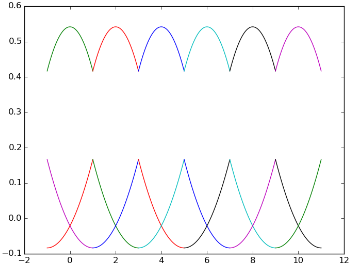
Inertia-Gravity Wave Dispersion Relationship (1D)

Multiple dofs per element $\rightarrow$ breaks translational invariance $\rightarrow$ spectral gaps

We have developed an alternative: mimetic Galerkin differences

Mimetic Galerkin Differences: Basis





$\mathcal{A} = H_1$ Space (1D)

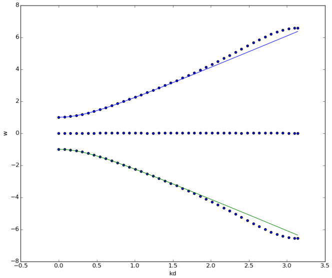
$\mathcal{B} = L_2$ Space (1D)

Single degree of freedom per geometric entity (physics coupling)

Higher order by larger stencils (less local)

3rd Order Elements

Mimetic Galerkin Differences- Dispersion



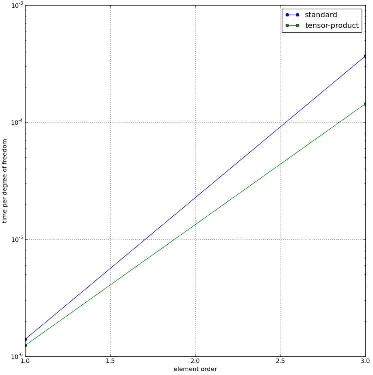
Inertia-Gravity Wave Dispersion Relationship (1D) for 3rd Order Elements

Spectral gap is gone

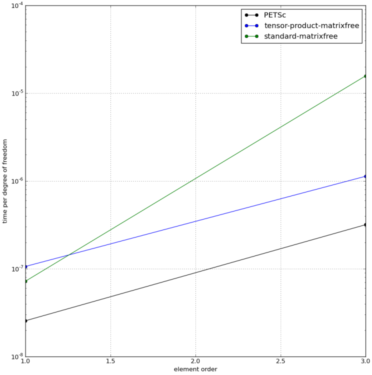
Can show that dispersion relation is $O(2n)$ where n is the order

More details in a forthcoming paper

MGD Results (Weighted Mass Matrix for H^1)



Assembly



Operator Action

$$\int \phi uv \text{ where } u, v \in H^1 \text{ and } \phi \in L_2$$

MGD Enhancements

How do we speed up MGD?

- MGD is structurally identical to IGA in terms of basis function support: Use ideas from that literature!
- Lookup tables- requires isoparametric geometry and coefficients/fields, tables get very large at high order
- Reduced/optimal quadrature rules: finite element based assembly uses too many quadrature points, reduce them (this is the source of p^d slowdown)
- Weighted row based assembly and operator action: based on reduced quadrature, assembly/action becomes truly independent per row so simpler parallelization