The Compadre Toolkit for Native Degrees-of-Freedom

Paul Kuberry, Mauro Perego, Nat Trask, Pavel Bochev
Center for Computing Research, Sandia National Laboratories

Native DoFs

Different codes may employ different discretizations of the same PDE due to different designs, e.g., stabilized vs. compatible.

The same field may be represented differently in a coupled multi-physics simulation, e.g.,
- Raviart-Thomas (H(div)) velocity vs. nodal (H1) velocity in Darcy-Stokes coupling

The field may be represented by the same type of discretization but on a different cell shape:
- Raviart-Thomas on tets, Raviart-Thomas on hexes and mimetic difference on polyhedrons

Research Problem

Provide a field reconstruction capability that:
- Accepts any reasonable native DoF type
- Reconstructs accurately a field from a type without having to know the internal reconstruction procedure for each code utilizing this DoF type

Generalized Moving Least Squares

Given a set of sampling functionals \( \mathcal{J}_i \) and a vector space \( V \), we seek the element of \( V \) that best matches a set of scattered degrees of freedom, and use that to approximate a target functional \( \tau \) over \( V \)

1. (GMLS Approximate) Constrained Optimization formulation:

   \[
   \min_{\mathbf{x} \in V} \left\{ \sum_i \mathcal{J}_i(x) \right\}
   \]

   \[
   \mathcal{J}_i(x) = \sum_{j} \mathcal{J}(x_j) \quad \forall \mathbf{x}_i \in V
   \]

2. (Practical Recipe) Least Square formulation:

   \[
   \mathbf{u} = \mathbf{u}(x) = \mathbf{u}(x) = \sum_{j} \mathcal{J}(x_j) \mathbf{u}(x_j)
   \]

Example: Point evaluation reconstruction of derivatives from nodal data in 1D.

Sampling Functional Choices

GMLS allows for the reconstruction of a vector field from Raviart-Thomas H(div) conforming scalar representations or from Nedelec H(curl) conforming representations by choosing \( \mathcal{J}_i \) (point evaluation) as \( \tau \), and \( \lambda_j \) as:

\[
\mathbf{u}(x_i) \cdot \mathbf{v}(x_i) \text{ or } \int_{x_i} \mathbf{u}(x_i) \cdot \mathbf{v}(x_i) \, d\mathbf{f}_i
\]

and

\[
\mathbf{u}(x_i) \cdot \mathbf{v}(x_i) \text{ or } \int_{x_i} \mathbf{u}(x_i) \cdot \mathbf{v}(x_i) \, d\mathbf{e}_i
\]

respectively.

Coupled Clamping Codes

The remap capability supports many different discretizations, easily allowing users to couple codes using discretizations including finite elements, finite volumes, finite differences, spectral elements, as well as fields generated by other meshless methods.

Disparate mesh discretizations can be coupled as is required by the Canga SciDAC-BER to demonstrate remap between cell-averaged finite volumes and spectral elements.

Caption: Coupling Approaches for Next Generation Architectures (CANGA)

Additionally, the Compadre Toolkit comes with an optional Python module generation that allows users to call a limited subset of the GMLS functionality, using their GPU (if selected). The Python package is now available through the PyPi repository and can be installed by running:

\[
\text{pip install compadre}
\]

Compadre Toolkit

The Compadre Toolkit provides a massively parallel solution framework for setting up and solving the quadratic programs (QP) defined for meshless discretizations such as Generalized Moving Least Squares (GMLS).

The coefficients generated by the toolkit are amenable to meshless remap and PDE solution, allowing users to harness meshless discretizations such as GMLS while executing these parallel communication-sparse, computationally-dense kernels on modern architectures.

The only third party library (TPL) needed by the toolkit is Kokkos, a performance portability library produced by Sandia, which allows code to be written once that will target multiple architectures and thread-parallel frameworks such as OpenMP, Pthreads, and CUDA.

Kokkos source code is included and automatically built by the toolkit, effectively reducing the requirements of Compadre to zero TPLs.

Caption: Target heterogeneous computational architectures via Kokkos.

Results

Raviart-Thomas (Face) elements

Low order Raviart-Thomas, data is contained at midpoints of faces \( \mathbf{f}_i \) and represents either \( u(x_i) \cdot \mathbf{v}(x_i) \) or \( \int_{x_i} u(x_i) \cdot \mathbf{v}(x_i) \, d\mathbf{f}_i \).

Nedelec (Edge) elements

Low order Nedelec, data is contained at midpoints of edges \( e_j \) and represents either \( u(x_j) \cdot \mathbf{v}(x_j) \) or \( \int_{x_j} u(x_j) \cdot \mathbf{v}(x_j) \, d\mathbf{e}_j \).

Visual Representation of RT Data

Face normal (Raviart-Thomas) field data

Caption: Reconstruct of a vector field represented as Raviart-Thomas and Nedelec DoFs

Remap of H(div) and H(curl) DoFs

Exact solution:

\[
\psi(x) = \begin{cases} \sin(x) & \sin(y) \\ \sin(x) & \sin(y) \end{cases}
\]

Degree of basis for reconstruction: 4

Case 1 Edges from a quasiform uniform mesh

Caption: Reconstructed edges from dilated mesh

Case 2 Point cloud of edge data, randomly rotated and dilated

Caption: Reconstructed edges from dilated mesh

Performance for Standard Point Data Remap

Scaling study of a single CPU vs. threaded CPU (Intel Pentium) vs. GPU (NVIDIA Tesla P100)

Caption: Standard Point Data Remap

Caption: Randomly Remapped Edges from Dilation Mesh

Caption: Standard Point Data Remap