RUHR-UNIVERSITÄT BOCHUM

RUB

High Order Matrix-Free FEM on GPU

July 3, 2025

Faculty of Mathematics
Numerics
Enes Mustafa Soydan

- Traditional Matrix-based FEM typically involves two primary steps:

$$K = \sum_{e=1}^{N_e} P_e^T K_e P_e \qquad \text{(Assembly)}$$

$$K u_1 = u_2 \qquad \text{(Iterative Solver)}$$

- Requires storage for the stiffness matrix (K).
- Element matrices and right-hand side vectors are transferred from local memory to global memory.

- Iterations are carried out in-place.

$$u^2 = \sum_{e=1}^{N_e} P_e^T K^e (P_e u^1) \qquad \text{(Iteration on the fly)}$$

- Stiffness matrix (K) is not stored.
- Only the local solution (u^2) is transferred from local memory to global memory.

Mass Matrix Operator Example:

$$M^e_{mn} = \sum_{q=1}^{N_q} w_q |J^e| \phi_m(\xi_q, \eta_q, \zeta_q) \phi_n(\xi_q, \eta_q, \zeta_q)$$
$$M^e u^1_e = u^2_e$$

- Local element mass matrix (M_e) can be formed and then multiplied by the local solution (u_1).
- The computational complexity for the forming local stiffness matrix is $O((p+1)^{2d})$. (p: degree, d: dimension).

High Order Matrix-Free FEM on GPU

- Interpolation polynomials are expressed in tensor product of 1-D polynomials.

$$\phi(\xi, \eta, \zeta) = \phi_i(\xi) \otimes \phi_j(\eta) \otimes \phi_k(\zeta)$$

- The evaluation of each shape function at each quadrature point results in a matrix ($I^{1D}_{ia} = \phi_{ia}(\xi_q)$).
- Final form of the mass matrix in tensor notation;

$$M^e = (I^{1D} \otimes I^{1D} \otimes I^{1D})^T J (I^{1D} \otimes I^{1D} \otimes I^{1D})$$

High Order Matrix-Free FEM on GPU

- The mass matrix operator can be expressed in sum factorization form using index notation:

$$u^e_{ijk} = \sum_{a=0}^{q} \phi_{ia}(\xi_q) \sum_{b=0}^{q} \phi_{jb}(\eta_q) \sum_{c=0}^{q} \phi_{kc}(\zeta_q) \sum_{a,b,c=0}^{q} J_{abc} \sum_{i=0}^{p} \phi_{ia}(\xi_q) \sum_{j=0}^{p} \phi_{jb}(\eta_q) \sum_{k=0}^{p} u^e_{ijk} \phi_{kc}(\zeta_q)$$

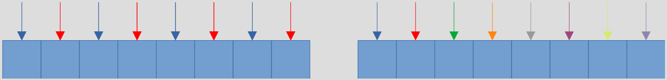
- Performing operations in Mat-vec multiplication form reduces computational complexity to $O(d(p+1)^{d+1})$.

High Order Matrix-Free FEM on GPU

- Our implementations are based on the CEED benchmarks.
- CEED benchmarks can be classified into two main catagories:
  1. Communication-free algorithms (Mass, stiffness matrix operators).
  2. Communication based algorithms (Preconditioned CG).
- As the current algorithms are communication-free, we employ a single GPU.
- Programming models are CUDA and Kokkos.

High Order Matrix-Free FEM on GPU

- Optimization techniques for our implementations can be grouped into two main categories.
  1. Thread granularity (block-centric, warp-centric)
  2. Thread-level Data Mapping (strided fashion, simple map)

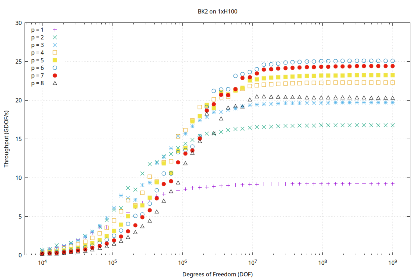


($\rightarrow$: thread ■ : array entry)

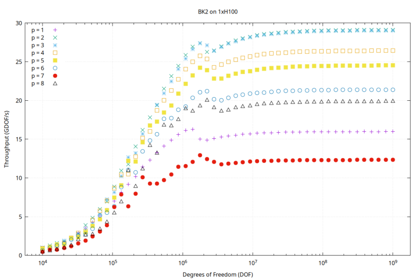
High Order Matrix-Free FEM on GPU

Preliminary Results and Findings

Strided access implementations offer more flexibility in thread block size.



A simple map is approximately 30% faster, but thread block occupancy depends on the polynomial order (i.e p=7, blockDim=729, max thread per SM=2048).



References

T. Kolev, P. Fischer, M. Min, J. Dongarra, J. Brown, V. Dobrev, T. Warburton, S. Tomov, M. S. Shephard, A. Abdelfattah, et al., *Efficient exascale discretizations: High-order finite element methods*, The International Journal of High Performance Computing Applications, 35 (2021), pp. 527–552.

M. Kronbichler and K. Kormann, *A generic interface for parallel cell-based finite element operator application*, Computers & Fluids, 63 (2012), pp. 135–147.

K. Świrydowicz, N. Chalmers, A. Karakus, and T. Warburton, *Acceleration of tensor-product operations for high-order finite element methods*, The International Journal of High Performance Computing Applications, 33 (2019), pp. 735–757.