libCEED - Lightweight High-Order Finite Elements Library
with Performance Portability and Extensibility

Jeremy Thompson1, Valeria Barra2, Yohann Dudouit3, Oana Marin4 & Jed Brown2

1: Department of Applied Mathematics & 2: Computer Science, University of Colorado Boulder
3: Lawrence Livermore National Laboratory, 4: Argonne National Laboratory

Abstract

High-order numerical methods are widely used in PDE solvers, but software packages that have provided high-performance implementations often have been special-purpose and intrusive. libCEED is a new library that offers a purely algebraic interface for matrix-free operator representation and supports run-time selection of implementations tuned for a variety of computational device types, including CPUs and GPUS. We introduce the libCEED API and demonstrate how it can be used in standalone code or integrated with other packages (e.g., PETSc, MFEM, Nek5000) to solve problems of different scales that often arise in the scientific computing community, ranging from fast solvers via geometric multigrid methods to Computational Fluid Dynamics (CFD) applications.

Performance Benchmarks

The libCEED project uses Benchmark Problems (BP) to test and compare the performance of high order finite element codes. We analyze the performance of libCEED backends on BP3: Poisson problem with homogeneous Dirichlet boundary conditions. We measure performance over 20 iterations of unpreconditioned Conjugate Gradient (CG) on hexahedral 3D elements with 1 more quadrature point than the number of nodes for the shape function in 1D. The plots below show work, measured by DoFs multiplied by CG iterations divided by compute nodes multiplied by seconds, plotted against problem size, measured by points per compute node. A variety of polynomial orders of the 1D shape functions, p, are shown.

Operator Decomposition

Finite element operators are typically defined through weak formulations of PDEs involving integration over a computational mesh. The required integrals are computed by splitting them as a sum over the mesh elements, mapping each element to a single reference element and applying a quadrature rule in the reference space. This is illustrated below for a symmetric linear operator on third order (Q3) scalar continuous (H1) elements, where T-vector, L-vector, E-vector, and Q-vector represent the (true) degrees of freedom on the global mesh, the split local degrees of freedom on the subdomains, the split degrees of freedom on the mesh elements, and the values at quadrature points, respectively.

Extensible Backends

The libCEED API takes an algebraic approach. The user describes the objects G, B, and D; and the library provides backend implementations to apply the local action of the PDE operator AΓ. This purely algebraic description includes all finite element information, so backends can operate on the linear algebra level without explicit finite element code. The separation of the frontend and backends enables applications to easily change backends.

Application - Geometric Multigrid

With high order finite elements, preconditioning is essential to control the condition number and total iteration count for iterative solvers such as CG. The libCEED operator decomposition offers several opportunities for preconditioning. P-multigrid offers mesh independent convergence for unstructured meshes. Restriction, Prolongation, and Smoothing operators can all be implemented in libCEED. We investigate performance of p-multigrid for BP3 on an unstructured mesh with hexahedral 3D elements.

Figure 2: libCEED Backends

libCEED supports several backends and users can develop additional backends. The LIBSSIM backends offer the best performance on the CPU and the CUDA backend with code generation offers the best performance on the GPU.

Figure 6: Solution of the compressible Navier-Stokes equations. In (a) the initial condition; In (b) the evolution at time t = 50 s.

Development Outlook

Status: Ready for collaborators and friendly users.

Further performance tuning for CPU and GPU
Improved non-conforming and mixed mesh support
Preconditioning based on libCEED decomposition
Algorithmic differentiation of Q-functions
HIP, OpenCL, and OpenMP Backends with OCCA

Contact Information:
https://ceed.exascaleproject.org
https://github.com/CEED/libCEED
email: valeria.barra@colorado.edu
email: jeremy.thompson@colorado.edu

email: jeremy.thompson@colorado.edu

libCEED supports several backends and users can develop additional backends. The LIBSSIM backends offer the best performance on the CPU and the CUDA backend with code generation offers the best performance on the GPU.