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

Jeremy Thompson*
jeremy.thompson@colorado.edu
University of Colorado
Boulder, Colorado

Valeria Barra*
valeria.barra@colorado.edu
University of Colorado
Boulder, Colorado

Yohann Dudouit†
dudouit1@llnl.gov
Lawrence Livermore National Laboratory
Livermore, California

Oana Marin‡
oanam@mcs.anl.gov
Argonne National Laboratory
Lemont, Illinois

Jed Brown*
jed@jedbrown.org
University of Colorado
Boulder, Colorado

Figure 1: libCEED QR-code

CCS CONCEPTS

• **Mathematics of computing → Mathematical software performance**; *Solvers*.

KEYWORDS

high-order finite elements, spectral elements, exascale computing

ABSTRACT

High-order numerical methods are widely used in PDE solvers, but software packages that have provided high-performance implementations have often 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 examples of problems that often arise in the scientific computing community, ranging from fast solvers via geometric multigrid methods to Computational Fluid Dynamics (CFD) applications.

1 INTRODUCTION

In finite element formulations, the weak form of a PDE is evaluated on a subdomain (element) and the local results are composed into a larger system of equations that models the entire problem. In particular, when high-order finite elements or spectral elements are used, the resulting sparse matrix representation of the global operator is computationally expensive, with respect to both the memory transfer and floating point operations needed for its evaluation. libCEED provides an interface for matrix-free operator description that enables efficient evaluation on a variety of computational device types (selectable at run time). Moreover, libCEED's purely algebraic interface can unobtrusively be integrated in new and legacy software to provide performance portable interfaces.

2 LIBCEED API

The libCEED API provides the local action of the linear or nonlinear operator without assembling its sparse representation. Let us define the global operator as

$$A = P^T G^T B^T D B G P\,, \tag{1}$$

where P is the parallel process decomposition operator (external to libCEED) in which the degrees of freedom (DOFs) are scattered to and gathered from the different compute devices. The operator given by $A_L = G^T B^T D B G$ gives the local action on a compute node or process, where G is a local element restriction operation that localizes DOFs based on the elements, B defines the action of the basis functions (or their gradients) on the nodes, and D is the user-defined pointwise function describing the physics of the problem at the quadrature points, also called the Q-function.

To achieve high performance, libCEED can take advantage of a tensor-product structure finite-element basis and quadrature rule to apply the action of the basis operator B or efficiently operate on bases that are defined on arbitrary-topology elements. Furthermore, the algebraic decomposition described in Eq. 1 can represent either

*All authors contributed equally to this research.

CEED Backend	Description
/cpu/ self /opt/*	Serial/Blocked optimized C backend
/cpu/ self /avx/*	Serial/Blocked AVX backend
/cpu/ self /xsmm/*	Serial/Blocked LIBXSMM backend
/*/ occa	OCCA CUDA, OCL, OMP, and HIP kernels
/gpu/cuda/ref	Refrence CUDA kernels
/gpu/cuda/gen	Optimized CUDA JiT operator kernels

Table 1: libCEED Backends

linear/nonlinear or symmetric/asymmetric operators and exposes opportunities for device-specific optimization.

Table 1 summarizes a subset of the backend implementations available in libCEED. GPU implementations are available via pure CUDA as well as the OCCA and MAGMA libraries. CPU implementations are available via pure C and AVX intrinsics as well as the LIBXSMM library. Backends can be selected at run-time, and each process or thread can instantiate an arbitrary number of backends.

3 PERFORMANCE BENCHMARKS

The CEED project uses Benchmark Problems to test and compare the performance of high order finite element codes. We analyze the performance of libCEED backends on Poisson problem with homogeneous Dirichlet boundary conditions. We measure performance over 20 iterations of unpreconditioned Conjugate Gradient (CG) on hexahedral 3D elements and plot work, measured by DoFs multiplied by CG iterations divided by compute nodes multiplied by seconds, against problem size, points per compute node. A variety of polynomial orders of the 1D shape functions, p, are shown in Fig 2.

4 GEOMETRIC MULTIGRID EXAMPLE

This example investigates p-multigrid for the Poisson problem, Eq. 2, using an unstructured high-order finite element discretization. All of the operators associated with the geometric multigrid are implemented in libCEED.

$$-\nabla \cdot (\kappa(x)\,\nabla x) = g(x) \tag{2}$$

The Poisson operator can be specified with the decomposition given by Eq. 1, and the restriction and prolongation operators given by interpolation basis operations, B, and B^T, respectively, act on the different grid levels with corresponding element restrictions, G. These three operations can be exploited by existing matrix-free multigrid software and smoothers.

We demonstrate performance for p-multigrid examples on an unstructured mesh. Preconditioning based on the libCEED finite element operator decomposition is an ongoing area of research.

5 COMPUTATIONAL FLUID DYNAMICS EXAMPLES

This example represents work being done towards the construction of a set of libCEED miniapps that describe problems typically arising in CFD, fully exploit the libCEED capability, and highlight the ease of library reuse for solver composition. This example solves the time-dependent Navier-Stokes equations of compressible gas dynamics in a static Eulerian three-dimensional frame using unstructured high-order spectral element spatial discretization and explicit high-order time-stepping (external to libCEED). The compressible Navier-Stokes equations are solved in conservative form:

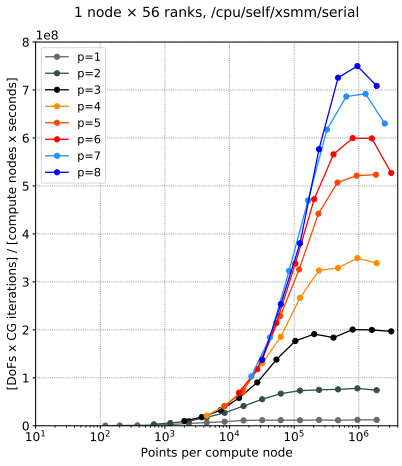

Figure 2: libCEED LIBXSMM Performance

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{U} = 0\,, \tag{3a}$$

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \left(\frac{\mathbf{U} \otimes \mathbf{U}}{\rho} + P\mathbf{I}_3 \right) + \rho g \hat{\mathbf{k}} = \nabla \cdot \sigma\,, \tag{3b}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot \left(\frac{(E+P)\mathbf{U}}{\rho} \right) = \nabla \cdot (\mathbf{u} \cdot \sigma + \mathbf{k}\nabla T)\,, \tag{3c}$$

In equations (3), ρ represents the volume mass density, U the momentum density (defined as $\mathbf{U} = \rho\mathbf{u}$, where $\mathbf{u}$ is the vector velocity field), E the total energy density (defined as $E = \rho e$, where e is the total energy), g the gravitational acceleration constant, $\hat{\mathbf{k}}$ the unit vector in the z direction, k the thermal conductivity constant, T represents the temperature, and P is the pressure.

6 CONCLUSION

libCEED is a new library that offers a purely algebraic interface for a matrix-free operator representation and supports run-time selection of implementations tuned for a variety of computational device types, including CPUs and GPUs. We demonstrate preconditioning of a Poisson operator with the geometric multigrid example and demonstrate support for nonlinear operator composition and library reuse with examples that arise in the CFD community, such as the compressible Navier-Stokes solver or the shallow-water equations solver.