6-14-2024

High-Performance Finite Elements with MFEM

Julian Andrej
Lawrence Livermore National Laboratory

Nabil Atallah
Lawrence Livermore National Laboratory

Jan-Phillip Baecker
Dortmund University

Jean-Sylvain Camier
Lawrence Livermore National Laboratory

Dylan Copeland
Lawrence Livermore National Laboratory

See next page for additional authors

Citation Details

Andrej, J., Atallah, N., Bäcker, J.-P., Camier, J.-S., Copeland, D., Dobrev, V., Dudouit, Y., Duswald, T., Keith, B., Kim, D., Kolev, T., Lazarov, B., Mittal, K., Pazner, W., Petrides, S., Shiraiwa, S., Stowell, M., & Tomov, V. (2024). High-performance finite elements with MFEM. The International Journal of High Performance Computing Applications.

Authors

Julian Andrej, Nabil Atallah, Jan-Phillip Baecker, Jean-Sylvain Camier, Dylan Copeland, Veselin Dobrev, Yohann Dudouit, Tobias Duswald, Brendan Keith, Will Pazner, and multiple additional authors

Special Issue Paper

High-performance finite elements with MFEM

Julian Andrej[1], **Nabil Atallah**[1], **Jan-Phillip Bäcker**[2] ,
Jean-Sylvain Camier[1], **Dylan Copeland**[1] , **Veselin Dobrev**[1] ,
Yohann Dudouit[1] , **Tobias Duswald**[3,4] , **Brendan Keith**[5] ,
Dohyun Kim[5] , **Tzanio Kolev**[1] , **Boyan Lazarov**[6],
Ketan Mittal[1] , **Will Pazner**[7] , **Socratis Petrides**[1],
Syun'ichi Shiraiwa[8], **Mark Stowell**[1] and **Vladimir Tomov**[1]

Abstract

The MFEM (Modular Finite Element Methods) library is a high-performance C++ library for finite element discretizations. MFEM supports numerous types of finite element methods and is the discretization engine powering many computational physics and engineering applications across a number of domains. This paper describes some of the recent research and development in MFEM, focusing on performance portability across leadership-class supercomputing facilities, including exascale supercomputers, as well as new capabilities and functionality, enabling a wider range of applications. Much of this work was undertaken as part of the Department of Energy's Exascale Computing Project (ECP) in collaboration with the Center for Efficient Exascale Discretizations (CEED).

1. Introduction

The MFEM (Modular Finite Element Methods) library is a high-performance, scalable, open-source, C++ library for finite element discretizations.[1] MFEM has undergone continual, rapid development since its initial release in July 2010. Its feature set has grown significantly, supporting a large number of applications, discretizations, and target architectures. This paper aims to summarize the recent developments in the library, as well as the related research into the underlying mathematical and numerical methods; it can be seen as a follow-up to the paper of Anderson et al. (2020a) and we recommend reviewing that paper for relevant background material.

Given the importance of graphics processing units (GPUs) to high-performance computing (HPC), MFEM's GPU capabilities have been considerably expanded and improved. For example, MFEM now provides comprehensive support for GPU-accelerated high-order mesh optimization through the target-matrix optimization paradigm. Additionally, MFEM provides end-to-end GPU-accelerated solvers for problems posed in all spaces of the finite element de Rham complex, making use of newly developed low-order-refined preconditioning techniques. These solvers have recently been extended to saddle-point problems, include grad-div problems in H(div) and mixed finite element problems. Novel kernel fusion techniques implemented in MFEM can be used to substantially improve the strong scalability of high-order finite element solvers, resulting in peak performance on problems of sizes 5–10×

[1]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA, USA
[2]Institute of Applied Mathematics (LS III), TU Dortmund University, Dortmund, Germany
[3]CERN, Geneva, Switzerland
[4]Technical University of Munich, Munich, Germany
[5]Division of Applied Mathematics, Brown University, Providence, RI, USA
[6]Computational Engineering Division, Lawrence Livermore National Laboratory, Livermore, CA, USA
[7]Fariborz Maseeh Department of Mathematics and Statistics, Portland State University, Portland, OR, USA
[8]Princeton Plasma Physics Laboratory, Princeton, NJ, USA

Corresponding author:
Will Pazner, Fariborz Maseeh Department of Mathematics and Statistics, Portland State University, 1855 SW Broadway, Portland, OR 97201, USA.
Email: pazner@pdx.edu

smaller than using traditional approaches. GPU-accelerated partial assembly kernels have been expanded to cover a very broad range of operators and discretizations. For example, NURBS-based discretizations (such as isogeometric analysis) now also support partial assembly.

In addition to the performance-focused features and developments described above, MFEM's feature-set has been significantly broadened. Support for specialized discretizations such as the discontinuous Petrov–Galerkin (DPG) method have been added. Automatic differentiation now enables the high-performance automated assembly of Jacobian matrices for applications such as nonlinear elasticity. Multiphysics applications are supported through submesh capabilities, providing users with straightforward mechanisms to use different physics in different parts of the domain. Discretization techniques for stochastic and fractional partial differential equations are enabled by the library, and illustrated in MFEM's included examples and miniapps. In addition to supporting traditional body-fitted meshes and finite element methods, MFEM now also supports level-set based methods; specific examples include distance solvers, shifted boundary methods, and cut integration rules.

Each of these topics is discussed in greater detail in the following sections. We begin with a summary of MFEM's GPU and HPC capabilities, including scalable matrix-free solvers. Then, we describe advances in MFEM's discretization support, such as DPG support, proximal Galerkin methods, immersed discretizations, NURBS/IGA, and automatic differentiation. Subsequently, we give an overview of advances in MFEM's meshing capabilities, with a focus on high-order mesh optimization algorithms. Finally, we illustrate the utility of these features and developments by describing a number of applications that make use of the newly introduced capabilities.

2. GPU acceleration and high-performance computing

One of MFEM's distinguishing features is its support for GPU acceleration throughout the library. GPU support was introduced in MFEM version 4.0, released in May 2019. Since then, an increasing number of features and discretizations support GPU acceleration. For example, MFEM's powerful mesh optimization features based on the target-matrix optimization paradigm now run on GPUs, enabling over 30× speedup compared to CPU evaluation. These advanced meshing capabilities are discussed in greater detail in the meshing section of the paper. Much of MFEM's GPU capabilities are focused on high-order discretizations. The MFEM team continues to perform research and development in generating optimized kernels and in reaching peak performance faster in high-order finite

element simulations. A major area of research and development is that of matrix-free solvers, which are tailored for high-order finite element problems. MFEM's most recent release includes support for full end-to-end GPU acceleration of matrix-free solvers for all spaces in the finite element de Rham complex. These solvers are discussed in the next section.

2.1. Matrix-free solvers for high-order finite elements

High-order finite element problems are typically solved using an iterative method, such as a Krylov subspace method, together with an effective preconditioner. Applying the action of high-order finite element operators in partially assembled form is very well-suited for GPU acceleration; the arithmetic intensity of the algorithm increases with increasing polynomial degree (see Abdelfattah et al., 2021; Kolev et al., 2021). However, the construction and application of performant preconditioners is more challenging. It is typically prohibitively expensive (both in terms of memory usage and computational complexity) to assemble and store the system matrix associated with the discretization, see Figure 8 and Anderson et al. (2020a). As a result, traditional matrix-based methods such as algebraic multigrid cannot be directly used in this context. Moreover, sparse matrix computations have low arithmetic intensity and are memory bandwidth bound; these operations often do not fully utilize the computational resources afforded by GPUs.

To address these issues, MFEM supports a range of *matrix-free* preconditioning methods for a variety of finite element operators and discretizations. Such preconditioners can be constructed and applied without access to the assembled matrix, and their memory usage is asymptotically optimal (linear scaling with the problem size). The two main approaches supported by MFEM are p-multigrid and low-order-refined preconditioning.

Both of these approaches aim to reduce the high-order problem to a related lower-order problem, which can then be assembled, and treated with, for example, algebraic multigrid. In p-multigrid, a hierarchy of finite element spaces with different polynomial degrees is constructed on the same mesh. These spaces are nested: there exists a natural injection from the lower-degree spaces into the higher-degree spaces. This gives rise to restriction and prolongation operators that transfer solutions and residuals between the spaces; the action of these operators can be computed efficiently on the GPU using sum-factorization techniques. At each level, a smoother is required; since the assembled matrix is not available, it is not feasible to use matrix-based relaxation methods such as Gauss–Seidel. Instead, smoothers based only on the diagonal of the matrix are used;

sum-factorization techniques provide algorithms to efficiently compute the diagonal of finite element matrices without assembling the matrix. Chebyshev acceleration can be used to improve the smoother.

Low-order-refined (LOR) preconditioning, also known in the literature as SEM–FEM preconditioning (see e.g. Fischer, 1997), is based on the idea of constructing a *spectrally equivalent* lowest-order discretization on a refined version of the mesh. The high-order discretization on the original mesh and the low-order discretization on the refined mesh have the same number of degrees of freedom. Under some conditions on the construction of the mesh, these two discretization are spectrally equivalent, independent of mesh size and polynomial degree (Canuto, 1994). Consequently, a preconditioner constructed using the low-order matrix can be used to effectively precondition the high-order operator. While this technique is by now considered classical (having originally been proposed in Orszag 1980), recent developments have extended the applicability of LOR approaches to additional discretizations, including all spaces in the finite element de Rham complex (Pazner et al., 2023b), as well as discontinuous Galerkin discretizations with *hp*-refinement (Pazner, 2020; Pazner and Kolev, 2021).

MFEM provides extensive, easy-to-use support for GPU-accelerated low-order-refined discretizations and solvers. Spectrally equivalent low-order discretizations can be created in a single line of code. Any of MFEM's wide array of supported preconditioners can be used for the LOR preconditioner; for example, *hypre*'s high-performance AMG, AMS, and ADS preconditioners, which are readily available in MFEM, provide effective solvers for diffusion problems posed in H^1, $H(\text{curl})$, and $H(\text{div})$.

The entire preconditioning pipeline is GPU-accelerated. The lowest-order matrix is assembled using a batched macro-element algorithm that takes advantage of the semi-structured nature of the refined mesh. High-performance, scalable algebraic multigrid methods can then be created directly on-device and combined with MFEM's highly performant partial assembly kernels for operator evaluation (Pazner et al., 2023a). These solvers are highly performant and scalable up to thousands of GPUs.

Spectrally equivalent LOR discretizations in $H(\text{curl})$ and $H(\text{div})$ use a particular choice of basis constructed using interpolation and histopolation operators; in this basis, the discrete curl and divergence differential operators defined on the high-order and low-order-refined spaces are exactly equal. This remarkable property is the key fact behind the spectral equivalence of the high-order and low-order operators. It can also be used in other contexts to create efficient preconditioners. High-performant preconditioners for saddle-point problems resulting from mixed finite element discretizations (e.g. Darcy problems) can be constructed using this technique (Pazner et al., 2024).

The low-order-refined approach can also be used to couple high-order and low-order discretizations in a multiphysics or mixed discretization framework. MFEM provides conservative and accurate transfer operators that can be used to communicate solutions and residuals between spaces of different orders and refinements (Kolev and Pazner, 2022).

2.2. State-of-the-art performance on exascale platforms and AMD GPUs

In order to achieve high-performance on a range of GPU-based platforms, MFEM supports a spectrum of *operator assembly levels*, ranging from *full assembly* of the system matrix, to completely *matrix-free* (zero storage) operators. The supported assembly levels are summarized below. Each assembly level brings its own computational efficiency and memory usage properties (see also Figure 8 and Anderson et al., 2020a).

2.2.1. Fully matrix-free. All necessary data needed for operator evaluation is computed on the fly, minimizing storage and memory requirements, at the cost of potentially recomputing data.

2.2.2. Partial assembly. Only the essential data at quadrature points is precomputed and stored, decreasing computation costs compared with fully matrix-free, while reducing memory footprint compared with matrix assembly. Partial assembly is especially effective for high-order finite elements.

2.2.3. Element assembly. Local contributions from each element are computed and stored in an element-local dense matrix. While this approach is more computationally and memory-intensive than partial assembly, it also gives access to an algebraic representation of the operator.

2.2.4. Sparse matrix assembly. The traditional assembly method computes and stores the global sparse matrix representing the operator. While this can be memory-intensive for large-scale problems (especially with higher order finite elements), it is compatible with a wide range of existing solvers and preconditioners.

All of the above-described assembly levels in MFEM are designed to be compatible with GPU-based execution, leveraging the computational power of modern GPUs to accelerate simulations. This compatibility with device execution on GPUs allows MFEM to handle complex, large-scale problems more efficiently, making it a versatile tool for scientific computing and engineering simulations.

MFEM has expanded its capabilities to fully support exascale computing platforms, including optimized support for multiple GPU architectures. The integration with AMD

GPUs is particularly noteworthy, as it aligns with the increasing adoption of AMD hardware in exascale and supercomputing platforms. Several bake-off problems (BPs) have been released by the CEED project and are used as important benchmarks: there are the mass and diffusion benchmarks: BP1 and BP3 (Fischer et al., 2020) as well as the more recent preconditioned Poisson solvers BPS3 (Kolev et al., 2021). The MFEM matrix-free kernels have shown good performance portability between NVIDIA and AMD architectures, as illustrated in Figure 1. These results compare the performance of MFEM's default (built-in) backend, and the libCEED backend, which makes use of high-performance kernels from the libCEED library (Brown et al., 2021).

2.3. Kernel fusion

GPU kernel fusion is an important technique for maximizing performance and scalability of GPU-accelerated finite element options. One of the challenges with it is providing a user-friendly programming interface to generate fused computational kernels. In this section we present some of our research in this area performed in MFEM as part of the CEED project.

Coding for kernel fusion often requires a paradigm shift, moving away from explicit GPU kernel launches. One general way to address this challenge is to gain in abstraction, using, for example, a more descriptive mathematical language. For finite element methods, the FEniCS project has formalized the Unified Form Language (UFL, see Logg et al., 2012). UFL is a domain specific language used to declare finite element variational forms, thus providing a high-level abstraction for specifying the mesh, the finite element spaces, the boundary conditions, as well as the linear and bilinear forms. Working at the compiler frontend level allows one to build a modular toolchain that transforms UFL to C++ and raw CUDA. This enables the construction of a source-to-source transformation which uses the graph of all the kernels that are needed at runtime, together with the memory locations that are to be read, written or copied, allowing static analysis and optimizations.
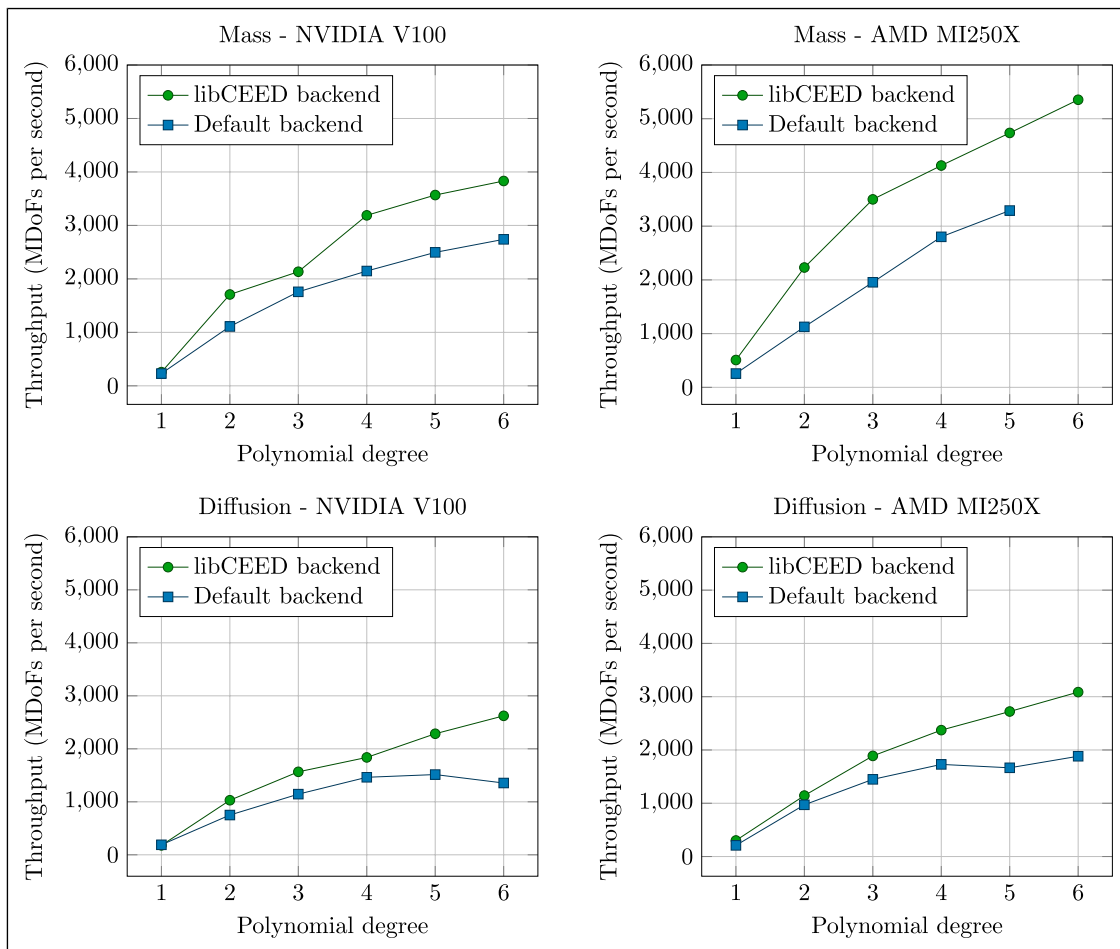

Figure 1. Comparison of the performance on the mass and diffusion benchmark problems (CEED BP1 and BP3) on NVIDIA V100 and AMD MI250X using the libCEED and default MFEM backends.

The implementation of the CEED benchmark problems using kernel fusion requires several building blocks. It is necessary to launch kernels with different topologies (e.g. 1D kernels for vector operations, and 2D and 3D thread blocks kernels for the main partially assembled operators). Specific warp levels instructions are also needed for the dot products. A mechanism to synchronize these different algorithmic parts is also necessary. One natural way to achieve these goals on NVIDIA GPUs is to use the *Cooperative Groups* programming model. We use group partitioning for organization and used the group collectives for synchronizations.

The results presented in the left panel of Figure 2 demonstrate single-GPU performance on the CEED mass matrix benchmark BP1. One main question is whether or not such gain in performance still holds for larger problems or solvers. Moving from a serial operator to the parallel version requires a triple product of the form $P^T A P$: the finite element operator A is prefixed and postfixed by multi-GPU communication operators P and P^T. The prefix communication operator prepares the buffers, sends the required data asynchronously to the neighbors, performs works on local vectors, and finally unpacks the received data to finalize the operation. The transposed operator performs the same operations but in reverse order. These methods have been implemented on NVIDIA GPUs with the *NVSHMEM* library, which proposes a parallel API based on *OpenSHMEM*, providing a global address space for multiple GPUs, allowing fine-grained kernel-initiated communication operations. This approach allows us to move from standard GPU-to-CPU data transfers with the MPI interconnect model to a direct GPU-to-GPU model. The right panel of Figure 2 shows the latest results obtained with a fully fused kernel with MPI communications for the CEED mass matrix benchmark. Note that the number

of degrees of freedom needed to reach 80% of peak performance has been reduced by a factor of five for the higher orders, thereby greatly increasing the strong scalability of these operators. To the best of our knowledge, this is the first time such an improvement has been observed with MPI communications overhead.

2.4. Mixed meshes, nonconforming meshes, and *p*-adaptivity

2.4.1. Matrix-free discontinuous galerkin methods on nonconforming meshes.
MFEM's GPU support has been enhanced with the implementation of matrix-free discontinuous Galerkin method on adaptively refined non-conforming meshes. The non-conforming support builds on the previous implementation of conforming matrix-free discontinuous Galerkin method. A key aspect of this implementation is the effective handling of non-conforming faces, achieved through a transformation of degrees of freedom, which allows the use of standard conforming face kernels on non-conforming meshes. This non-conforming transformation happens at the level of element-wise vectors (E-vectors) only on non-conforming faces, minimizing its computational cost. For non-conforming faces, degrees of freedom are interpolated from the coarse face to the fine face such that the face appears as a conforming face from a computational perspective. Performance results are shown in Figure 3.

2.4.2. Mixed meshes and *p*-adaptivity

The integration of the libCEED library [see Brown et al., 2021] in MFEM adds unique features to matrix-free operators, including support for simplices, SYCL-based
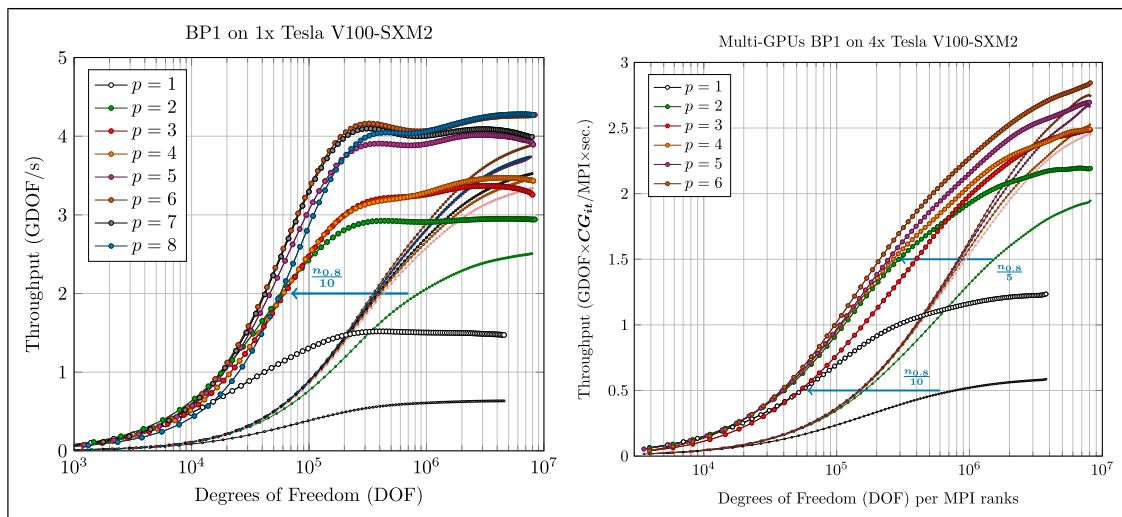

Figure 2. Throughput comparison of fused kernels (larger plot markers) and unfused kernels (smaller plot markers) on the mass matrix benchmark.
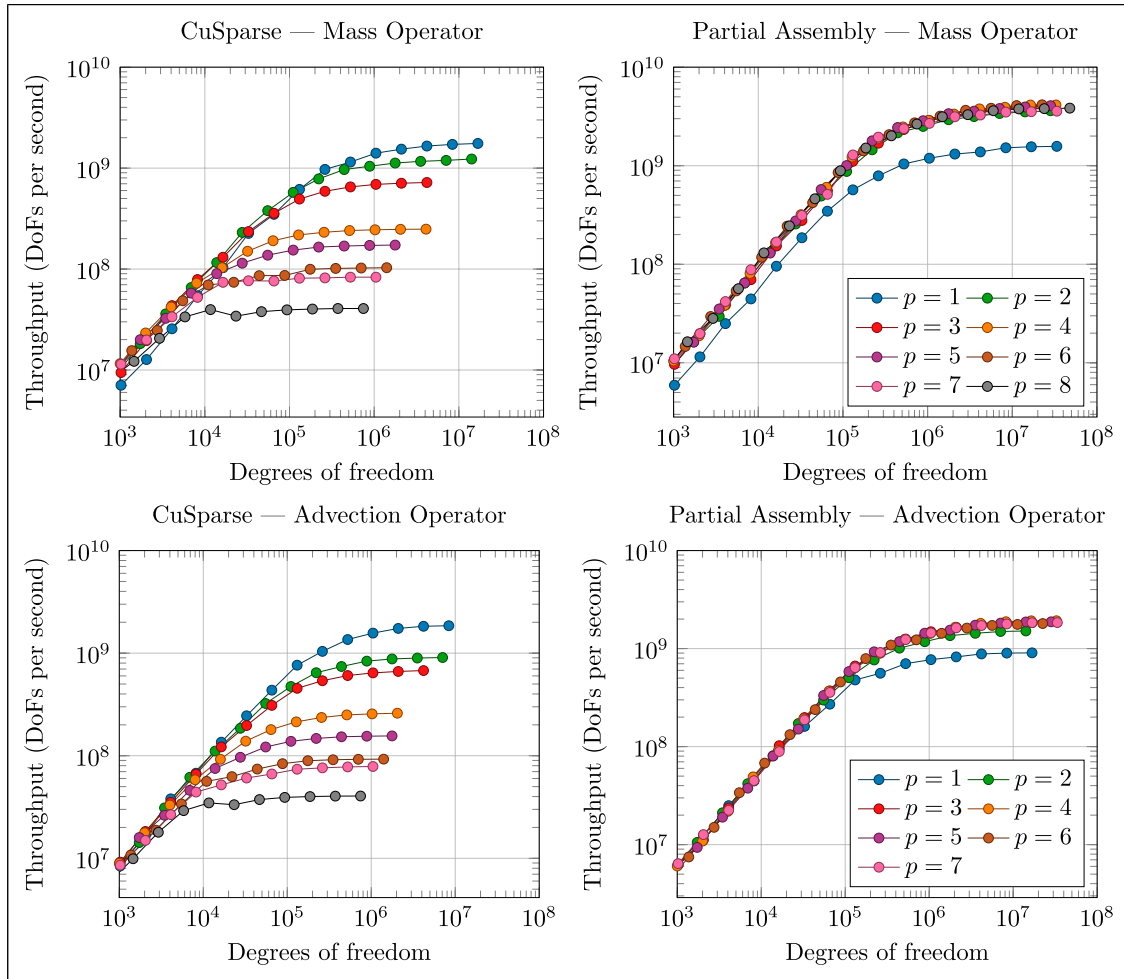
Figure 3. Performance comparison of matrix-free discontinuous Galerkin mass operators (top row) and advection operators (bottom row) (NVIDIA V100).

hardware, and fully matrix-free operators (zero storage operators). The most notable recent addition is the support for mixed meshes with element of different geometric type (including simplices, pyramids, and wedges), and *p*-adaptivity. MFEM users can run simulations using matrix-free operators on mixed meshes and *p*-adaptivity by simply using the libCEED backend. The performance on such meshes is illustrated on Figure 4. We observe similar performance on mixed meshes and on tetrahedral meshes, however pure hexahedral meshes result in significantly higher performance due to the use of sum-factorization algorithms.

3. Discretizations

The MFEM library supports a wide range of finite element methods and discretizations. In addition to traditional conforming finite element methods, MFEM provides ever-increasing support for alternative or non-standard finite

element methods some of which are described below. These methods differ in important ways from standard methods, and can provide advantages such as discrete stability, pointwise bounds preservation, or ability to handle implicitly defined geometries.

3.1. Discontinuous petrov-galerkin (DPG) methods

MFEM now supports a wide range of DPG formulations spanning the whole de Rham complex (Carstensen et al., 2016). DPG is a non-standard minimum residual method that provides high accuracy, unconditional discrete stability and positive definite linear systems (Demkowicz and Gopalakrishnan, 2010). It is well-suited for challenging problems that are prone to loss of discrete stability (e.g. high-frequency time-harmonic acoustics and electromagnetic equations) and for problems that require adaptive mesh refinement (AMR) (e.g. problems with singular solutions; see Demkowicz et al., 2012; Petrides and Demkowicz 2017, 2021).
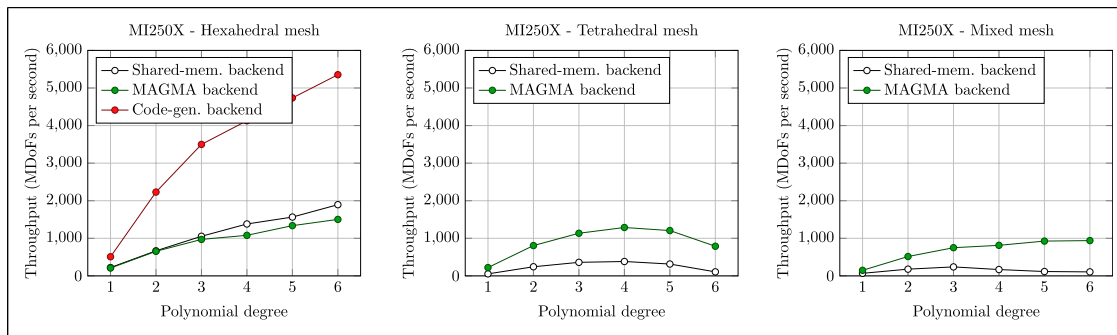
Figure 4. Performance of a matrix-free mass operator (BP1 benchmark problem) for different type of meshes using an AMD MI250X GPU. Comparison of libCEED backend performance.
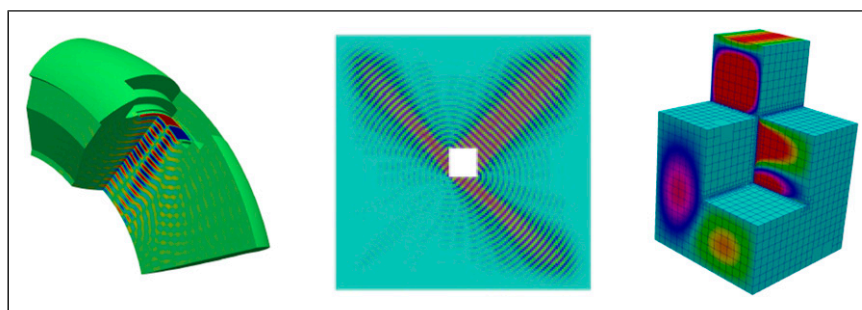

Figure 5. Ultraweak DPG formulations for time-harmonic Maxwell equations for Tokamak simulations(left), 2D scattering of an acoustics beam (center), and for the time-harmonic Maxwell equations with AMR for the simulation of the microwave oven problem (right).

This new feature introduces user-friendly classes that allow users to define multiple trial and test spaces as well as the desired bilinear and linear integrators in a similar fashion to the standard Galerkin formulation. The DPG linear system assembly, which also involves the on-the-fly computation of optimal test functions (see Demkowicz and Gopalakrishnan, 2011; Zitelli et al., 2011), is then performed automatically under the hood. The system is returned to the user in blocked form. Complex-valued systems, static condensation and AMR driven by the DPG built-in error indicator are also supported (Demkowicz et al., 2012). This new development is demonstrated with several examples for diffusion, convection-diffusion, time-harmonic acoustic and electromagnetic wave equations; these features are illustrated in Figure 5.

3.2. Proximal galerkin

The latent variable proximal Galerkin (PG) finite element method (Keith and Surowiec, 2023) is a high-order, non-linear finite element method that preserves the intrinsic geometric structure of pointwise bound constraints in function spaces. MFEM now provides implementations of PG through two new examples: Example 36, which uses PG to solve the classical obstacle problem, is a template for other unilateral-constrained variational inequalities and free boundary problems in Sobolev spaces; and Example 37, which uses a novel information-geometric approach to topology optimization, and provides a template for using PG to solve bilateral-constrained variational inequalities and pointwise bound-constrained optimization problems in Lebesgue spaces.

3.3. Immersed discretizations

Capabilities for finite element calculations over immersed meshes have recently been added to the library. These methods assume the feature of interest (domain boundary or internal interface) is given implicitly by the zero level set of a discrete function. The new features share a common objective: adhering to finite element operations while avoiding purely geometric computations. This principle enables better generality across dimensions, element orders, and types of finite element discretization.

3.3.1. Integration over cut elements. MFEM provides two alternative approaches for constructing integration rules

over discretely prescribed cut surfaces and volumes. Both methods enable the computation of

$$S = \int_{\phi=0} u(x)\, ds, \qquad V = \int_{\phi>0} u(x)\, dx$$

where the level set function ϕ specifies the cut, and u is the integrand. The first approach relies on the external Algoim package (Saye, 2015), which works for quadrilateral and hexahedral elements by performing dimension reduction to construct a combination of 1D quadrature rules. The second approach uses the moment-fitting algorithm introduced in Müller et al. (2013). This method constructs a set of basis functions for each element to define and solve a local under-determined system for the vector of quadrature weights. All surface and volume integrals required to form the system are reduced to 1D integration over intersected segments. The newly developed integration techniques allow for fast and efficient implementation of high-order CutFEM (Burman et al., 2015) discretizations as demonstrated in Figure 6.

3.3.2. Shifted boundary method. MFEM also provides an implementation of the Shifted Boundary Method (SBM) (Atallah et al., 2020), which is a technique that avoids integration in cut elements by using a surrogate computational domain. The method uses a distance function to the true boundary to enforce the required boundary conditions on the (non-aligned) surrogate mesh faces, thereby *shifting* the location where the boundary conditions are imposed, see Figure 7. The enforcement is performed weakly, through face integrals on the surrogate faces. The desired accuracy is obtained by combining the distance function with a Taylor expansion of certain order. These techniques are illustrated in MFEM's *Shifted* miniapp.

3.3.3. Distance and extrapolation solvers. Discrete distance solvers are often needed for computations that involve level set functions. MFEM's *Distance* miniapp demonstrates the capability to compute the shortest path through the computational mesh to a given point source or to the zero level set of a given function (see Figure 7). The implemented methods include the heat method described in Crane et al. (2017), and the *p*-Laplacian method from Belyaev and Fayolle (2015).

Extrapolation is another useful capability in immersed frameworks. The *Extrapolation* miniapp extrapolates a finite element grid function from known values on a set of elements to the rest of the domain. The miniapp supports the PDE-based approaches from Aslam (2004) and Bochkov and Gibou (2020), both of which rely on solving a sequence of advection problems in the direction of the unknown parts of the domain. The extrapolation can be up to third-order in a limited band around the zero level set.

3.4. Partial assembly on NURBS patches

In addition to the existing support for partial assembly on elements, MFEM now supports partial assembly on NURBS patches in the settings of isogeometric analysis. Significant computational savings are possible when assembling patch-wise rather than element-wise, as the tensor-product structure exists on the entire patch of elements, allowing for more benefit in sum factorization. In this sense, a NURBS patch, containing many elements, can be considered as a very high-order single element. This new feature is demonstrated in the *NURBS* miniapp.

Optimization of patch-wise assembly has been done in works such as Hiemstra et al. (2019). At the patch level, 1D basis functions in the sum factorization may span multiple elements, but computations for a pair of test and trial basis functions are limited to the intersection of their spans. We further optimize by computing a reduced quadrature rule for each basis function. Given the smoothness of patch basis
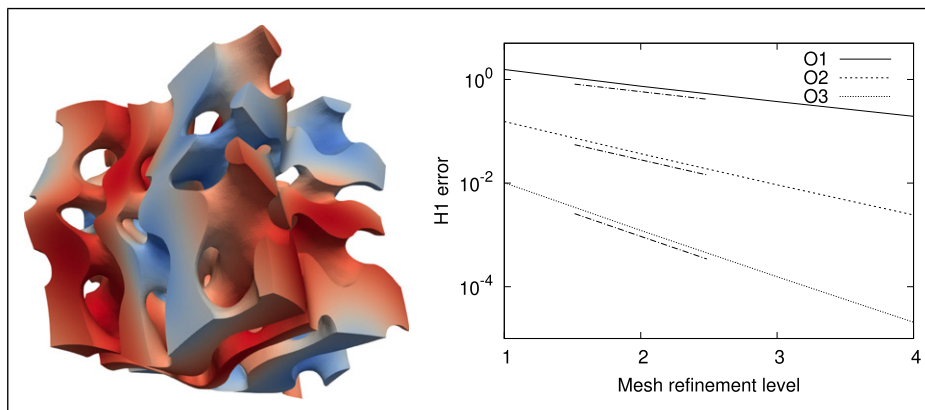

Figure 6. The left figure shows an elastically deformed Gyroid structure with a displacement field obtained through CutFEM solution on a regular grid with a predefined force function, and the right figure shows the H_1 convergence plot of a regularized CutFEM solution for linear, quadratic, and cubic Lagrangian elements.
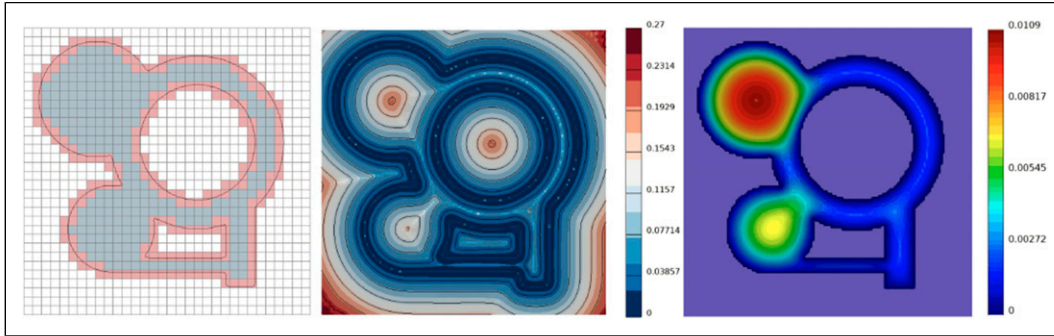
Figure 7. Example shifted boundary calculation. Left: true boundary with background mesh and surrogate domain. Middle: discrete distance field. Right: SBM solution.

functions, quadrature rules chosen for a particular order on one element may have more points than necessary to achieve accuracy, so a reduced rule is possible. We compute reduced rules automatically by the algebraic non-negative least squares (NNLS) method (Lawson and Hanson, 1995), rather than deriving and implementing reduced rules for every possible case. These reduced rules are computed on the fly, for the choice of NURBS space, patch, and bilinear form, and stored for use throughout a system solve or simulation.

The newly introduced NNLS solver is also a generally useful addition, which in particular could be used for computing reduced quadrature rules in other contexts.

3.5. Automatic differentiation (∂FEM)

In recent years, there has been an increased demand for automatic differentiation (AD) of numerical simulations, due in part to the rise of machine learning (ML) frameworks (see e.g. Abadi et al., 2016; Paszke et al., 2017). AD includes a set of well-established techniques for evaluating derivatives of functions written as computer programs with traditional implementation approaches based on operator overloading (Griewank et al., 1996) or source transformation (Hascoët and Pascual, 2013). Some of these conventional approaches require the use of a domain-specific language (Paszke et al., 2017), limiting the portability and the number of possible hardware and software execution platforms. On the other hand, MFEM is used in an HPC context for complex applications composed of libraries in different languages; traditional AD approaches would necessitate significant changes to the data types and code organization, making their application to MFEM more challenging.

To address this, a collaborative effort, the ∂FEM project targeting the integration of the Enzyme tool (Moses et al., 2021) and developing further a native MFEM AD implementation based on operator overloading and templating, emerged from the need to automatically compute residuals,

Jacobians, and Hessians in FEM discretizations and optimal design applications. In contrast to traditional tools, Enzyme performs forward and reverse-mode differentiation on the LLVM compiler intermediate representation (Lattner and Adve, 2004), enabling it to synthesize fast derivatives of programs in any language whose compiler targets LLVM, as well as a wide variety of parallel frameworks and hardware architectures, all in a single tool.

MFEM's infrastructure is built around the finite element operator decomposition, shown in Figure 8, which encapsulates a generic description of an assembly procedure in a finite element library. Instead of following traditional approaches that apply automatic differentiation at a global level and treat the implementation as a black box, the operator decomposition allows MFEM to treat derivatives on the innermost level at the quadrature points (D). The operators transferring data from a global level to subdomain, element, and quadrature levels, i.e., P, G, and B, are linear and topological: they do not depend on the solution, physical coordinates or design parameters. Therefore, they can be excluded from the differentiation loop, saving a significant amount of memory and computations. The decomposition confines the code modifications to the integration point level, allowing full automation of the discretization process for complex non-linear problems.

While the ∂FEM effort is still a work-in-progress, a proof of concept can be found in the *Hooke* miniapp, which implements a finite-strain elasticity discretization with arbitrarily user-defined material models and automatically generated Jacobians (derivatives of the residual equation with respect to the displacement). The derivatives in this example can either be generated by leveraging Enzyme or MFEM's internal dual number type implementation.

4. Meshing

MFEM includes advanced meshing support such as general element types, conforming and non-conforming adaptive mesh refinement, high-order meshing and more, (see
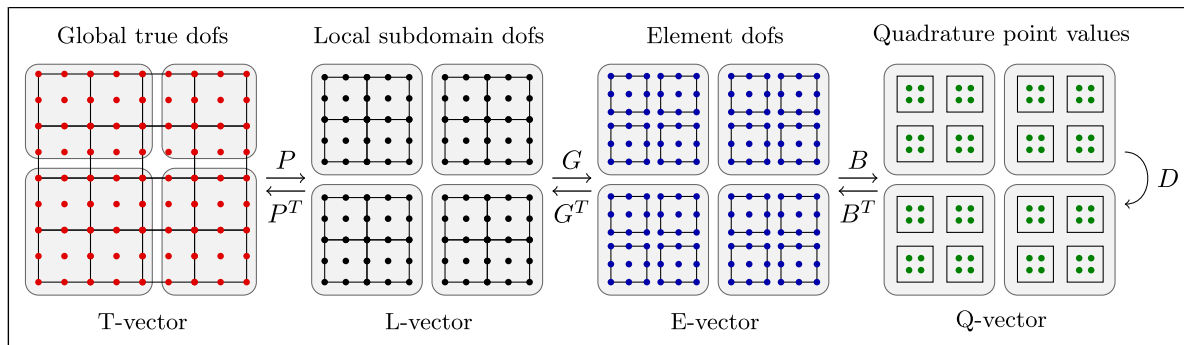
Figure 8. Finite element operators, A_p, have a natural decomposition, $A_p = P^T G^T B^T DBGP$, which exposes multi-level parallelism and allows for AD-friendly, matrix-free, memory-efficient implementations that assemble and store only the innermost, pointwise operator component (partial assembly, c.f. Anderson et al. 2020a).

Anderson et al., 2020a). In this section we review recent progress in MFEM's mesh optimization algorithms and describe the addition of submesh capabilities for multi-domain problems.

4.1. High-order mesh optimization

The high-order mesh optimization framework in MFEM is based on the Target-Matrix Optimization Paradigm (TMOP). This framework enables precise control over local mesh quality, inferred through the local Jacobian $A_{d \times d}$ of the transformation from the reference to physical space coordinates, while still optimizing the mesh globally.

The first step with TMOP is to define a target transformation matrix $W_{d \times d}$ analogous to A. Construction of W is typically driven by the fact that any Jacobian matrix can be composed as a function of four fundamental geometric properties:

$$W_{d \times d} = \underbrace{\zeta}_{[\text{size}]} \underbrace{R_{d \times d}}_{[\text{rotation}]} \underbrace{Q_{d \times d}}_{[\text{skewness}]} \underbrace{D_{d \times d}}_{[\text{aspectratio}]} \quad (1)$$

namely size/volume, rotation, skewness, and aspect-ratio. In general, each of the components of W in (1) can vary spatially based on the desired properties at each point in the mesh. A detailed description of target construction procedure is provided in Knupp (2019).

With the target transformation W defined, a mesh quality metric, $\mu(T)$, $T = AW^{-1}$, is used to measure the difference between the transformations A and W. Mesh quality metrics are categorized based on the geometric parameters they depend on. For example, shape metrics μ_{Sh} depend on the element skewness and aspect-ratio, size metrics μ_{Sz} depend on the element size. There are also composite metrics that depend on some combination of the four parameters (e.g., μ_{ShSz} and μ_{ShSzOr}). In practice, the user defines W and chooses a mesh quality metric based on the geometric properties they wish to optimize.

The quality metric $\mu(T)$ is used to define the TMOP objective function

$$F(\mathbf{x}) = \sum_{E \in \mathcal{M}} \int_{E_t} \mu(T(\mathbf{x})) \, d\mathbf{x}_t \quad (2)$$

where E_t is the target element determined by W, and $d\mathbf{x}_t$ denotes integration over the target element. Optimal node locations are determined using Newton's method by solving $\partial F(\mathbf{x})/\partial \mathbf{x} = 0$. Note that since (2) can be represented solely in terms of finite element operations, it extends to all element types (quadrilaterals, hexahedra, simplices, pyramids) in 2D and 3D, supports h/p-refined meshes, and can leverage advances in finite element assembly technique targeting GPU acceleration.

The effectiveness of TMOP for mesh quality improvement largely depends on the prescribed target and the mesh quality metric, and we have advanced the state-of-the-art in each of these areas. We have developed approaches for automated target construction for different mesh adaptivity goals (Knupp, 2019; Dobrev et al., 2020, 2021). For general mesh quality improvement, the target geometric properties are set based on the ideal element type (e.g., cube or regular tetrahedron). For simulation-driven optimization, the discrete solution or a derived quantity (e.g., error estimate) is used to set the target geometric properties. Figure 9 shows an example of simulation-driven adaptivity where a uniform mesh is morphed to adapt its shape and size with respect to a scalar solution and its size and orientation for a vector solution.

We have also analyzed the theoretical properties of different mesh quality metrics in recent work (Knupp, 2020, 2023) to identify well posed metrics of different types. Compound metrics of the form $\mu_{ShSz} = \mu_{Sh} + \gamma \mu_{Sz}$, $\gamma \in \mathbb{R}^+$ are effective for simulation-driven optimization, but require tuning γ on a case-by-case basis as its subcomponents may scale differently with mesh distortion. We have studied the asymptotic properties of different compound metrics to
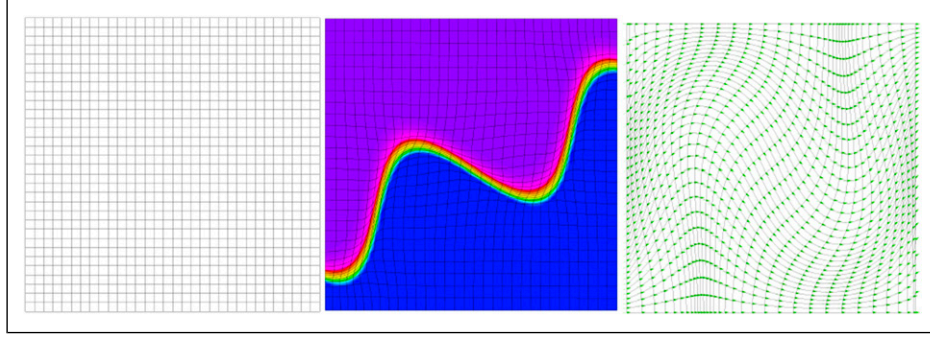
Figure 9. Simulation-driven adaptivity to morph a uniform mesh (left) for shape and size adaptivity using a scalar solution (center) and size and orientation adaptivity using a vector solution (right).

address this issue. We have also developed new asymptotically balanced compound metrics of the form $\mu_{ShSz} = \mu_{Sh}^{\alpha} + \lambda \mu_{Sz}^{\beta}$, $\{\alpha, \beta\} \in \mathbb{R}^{+}$, that are balanced for a prescribed range of λ (Dobrev et al., 2024). These compound metrics have been impactful for shape and size adaptivity in arbitrary Lagrangian-Eulerian (ALE) simulations (cf. Anderson et al., 2018; 2020b), without requiring user intervention; see Figure 10 for an example. Finally, we have also developed new mesh quality metrics that can simultaneously untangle the mesh and improve worst element quality. These metrics are crucial for problems with severe or localized mesh distortion.

In recent work, we have augmented the TMOP-based formulation with a penalization term to weakly enforce alignment of a selected set of mesh nodes with an implicit surface, while simultaneously optimizing the mesh quality (Barrera et al., 2023). This formulation requires that the target surface be prescribed as the zero isocontour of a smooth discrete level-set function, which is often the case for representing interfaces in multimaterial configurations and evolving geometries in shape and topology optimization. The modified objective function is

$$F(\boldsymbol{x}) = \underbrace{\sum_{E \in \mathcal{M}} \int_{E_t} \mu(T(\boldsymbol{x})) d\boldsymbol{x}_t}_{F_\mu} + \underbrace{w_\sigma \sum_{s \in S} \sigma^2(\boldsymbol{x}_s)}_{F_\sigma} \quad (3)$$

Here, F_μ is the mesh quality term, and F_σ is a penalty term that depends on the penalization weight w_σ, the set of mesh nodes $\mathcal{S}$ to be aligned to the level set, and the level set function $\sigma(\boldsymbol{x})$ evaluated at the positions $\boldsymbol{x}_s$ of all nodes $s \in \mathcal{S}$. This formulation has proven to be robust at obtaining body-fitted meshes for complex curvilinear domains, and is helping circumvent the bottleneck of high-order mesh generation; see Figures 11 and 12. We have further augmented this formulation with MFEM's p-adaptivity framework to obtain mixed-order meshes, such that high-order elements are only used adjacent to regions of
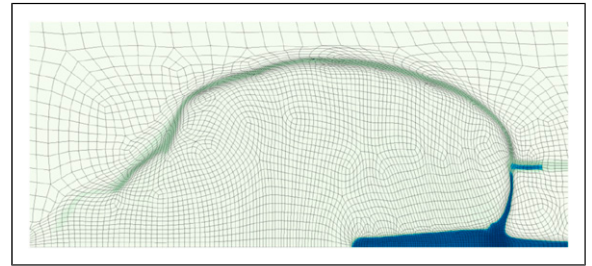

Figure 10. Shape and size optimization for a 2D shaped charge simulation using an asymptotically balanced compound metric, $\mu_{ShSz} = \mu_{Sh} + 3/2\mu_{Sz}$ (Dobrev et al., 2024).

high-curvature in the target surface. This rp-adaptivity approach has proven to be robust at obtaining mixed-order meshes with same geometric accuracy at a lower computational cost in comparison to a constant-order mesh (Mittal et al., 2024).

Finally, to support MFEM-based applications that leverage GPUs, we have implemented TMOP-related operators with sum-factorization and partial assembly (Camier et al., 2023). This includes new GPU kernels for evaluating the TMOP objective function (2) along with its first and second-derivatives, and a Jacobi preconditioner for accelerating the Newton solve. Use of sum-factorization and partial assembly on GPUs has significantly reduced the time for Newton solve in comparison to the traditional fully assembled matrix-based techniques on CPUs. Table 1 and Figure 13 show results for the total mesh optimization time on GPU and CPU from a benchmark problem in Camier et al. (2023).

The use of TMOP algorithms is demonstrated in the *Mesh Optimizer* and *Mesh Fitting* miniapps, which use the core TMOP functions for CPUs GPUs.

4.2. Submesh interface

A recent addition to MFEM is its submesh interface, which allows applications to define a mesh that represents a subset
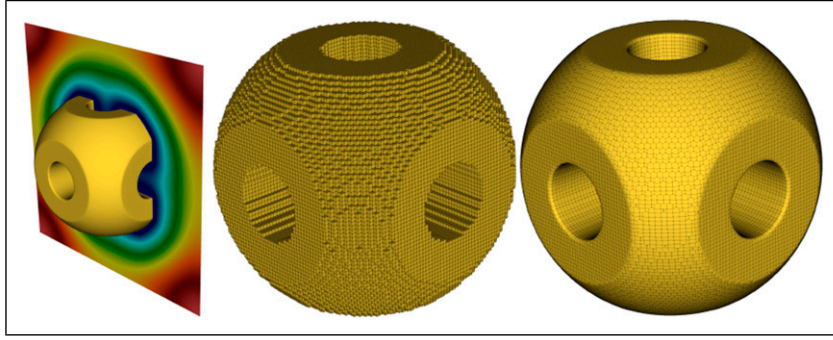
Figure 11. Discrete function along with its zero isosurface representing a 3D curvilinear surface (left). Initial uniform hex mesh (center) morphed to align with the target surface (right).
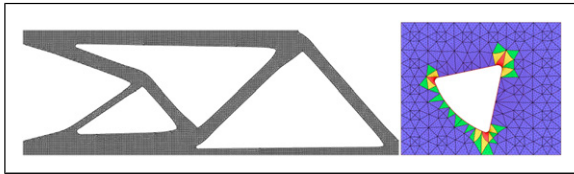

Figure 12. Left panel: topology optimized cantilever beam where the surface fitting formulation is used to obtain a body-fitted mesh at each TO iteration. Right panel: *rp*-refined body-fitted mesh for a 2D version of the Apollo capsule. Different colors represent different polynomial orders.

Table 1. Comparison of total time for mesh optimization for meshes of different orders (*p*).

	Time to mesh optimization (sec.)			
	p = 1	*p* = 2	*p* = 3	*p* = 4
CPU	18.8	43.0	129.6	224.3
GPU	0.4	1.0	3.9	7.5
Speedup	**47×**	**43×**	**33×**	**30×**

of a parent mesh. Subsets can be formed from volume elements or boundary elements, creating domain or surface meshes, respectively. A submesh is a fully functional MFEM mesh, and so all of MFEM's existing machinery can be used on one or multiple submeshes. Furthermore, MFEM provides the ability to transfer grid functions between a parent mesh and its submeshes or between submeshes that share the same parent Mesh. Let S_P^A denote the operator that extracts a submesh grid function u_A from the parent grid function u_p, such that $u_A = S_P^A u_p$. We can use this operator to also extend submesh grid functions to the full mesh and to exchange data between submeshes. See Figure 14 for an illustration. This concept can be used to express multi-physics problems in a straightforward way in MFEM. Another use case for submesh capability is the restriction of operations that compute quantities of interest to specific parts of the domain.

The *Multidomain* miniapp in MFEM is a simple demonstration of how to solve two PDEs, each representing different physics, on the same domain. MFEM's submesh interface is used to compute on and transfer between the spaces of predefined parts of the domain. In this miniapp, the spaces on each domain are using the same order H^1 finite elements; this is not a requirement, and submesh easily handles different finite element spaces on different domains. A 3D domain comprised of an outer box with a cylinder shaped inside is used. A heat equation is described on the outer box domain

$$\frac{\partial T}{\partial t} = \kappa \nabla^2 T \qquad \text{in outer box,}$$
$$T = T_{wall} \qquad \text{on outside wall,}$$
$$\nabla T \cdot \overrightarrow{n} = 0 \qquad \text{on inside (cylinder) wall}$$

with temperature T, outward unit normal $\overrightarrow{n}$ and coefficient κ. A convection-diffusion equation is described inside the cylinder domain

$$\frac{\partial T}{\partial t} = \kappa \nabla^2 T - \alpha \nabla \cdot \left(\overrightarrow{b} \, T \right) \qquad \text{in inner cylinder,}$$
$$T = T_{wall} \qquad \text{on cylinder wall,}$$
$$\nabla T \cdot \overrightarrow{n} = 0 \qquad \text{else}$$

with temperature T, outward unit normal $\overrightarrow{n}$, coefficients κ, α and prescribed velocity profile $\overrightarrow{b}$. To couple the solutions of both equations, a segregated solve with a one way coupling approach is used. The heat equation of the outer box is solved from the time step $T_{box}(t)$ to $T_{box}(t + dt)$. Then, for the convection-diffusion equation, T_{wall} is set to $T_{box}(t + dt)$ and the equation is solved for $T(t + dt)$, resulting in a first-order one-way coupling. A visualized result can be seen in Figure 15.
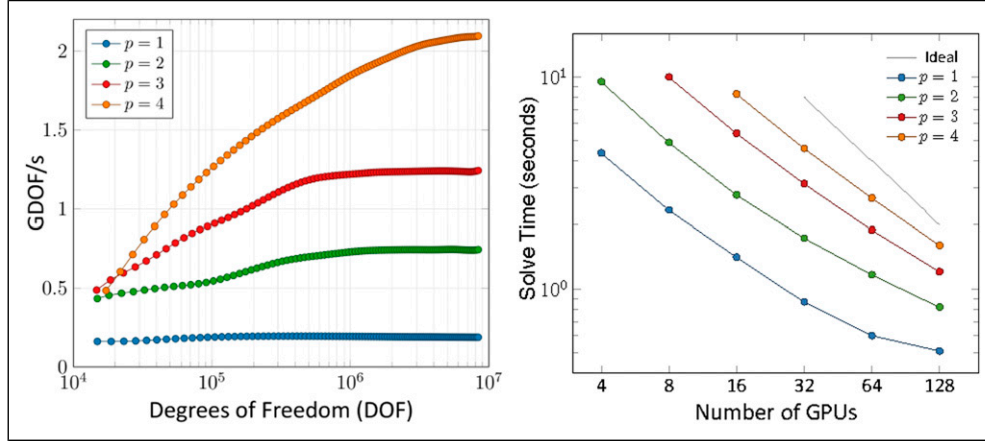
Figure 13. Throughput for action of the TMOP Hessian $\partial^2 F$ (left panel) and strong-scaling of the total time for mesh optimization on GPUs (right panel).
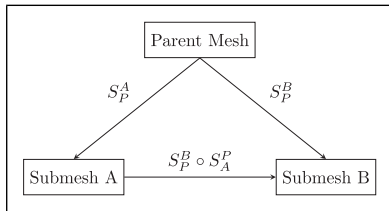

Figure 14. Tree structure showing two submeshes which are derived from the same parent and share a boundary. Operations that transfer grid functions between the different submeshes are also illustrated.
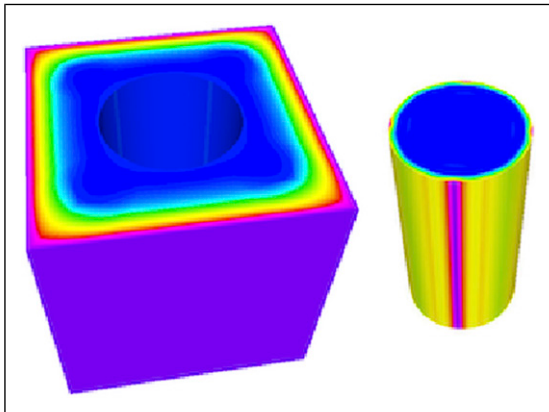

Figure 15. Simulation result of first-order coupling between a heat equation and a convection-diffusion equation on submeshes sharing boundary conditions.

5. Applications

The modular structure of MFEM allows it to powers a wide variety of applications in areas such as compressible and incompressible flow, electromagnetics, magnetic and inertial confinement fusion, additive manufacturing, topology optimization, structural mechanics, subsurface flow, hearth and MRI modeling and more, (see Anderson et al., 2020a). To aid with the development of new applications, a large number of examples and miniapps are included with the library. In this section we review some recent MFEM examples, miniapps and application and describe the PyMFEM interface which allows Python users to take full advantage of MFEM's capabilities.

5.1. Random fields and fractional stochastic PDEs

MFEM's modular and performant implementation enables the design of fast and scalable solvers for fractional and stochastic PDEs. Such equations arise naturally in the context of random fields, which are frequently used to model spatially correlated uncertainties in computational science and engineering applications.

Whittle (1954, 1963) first realized that the solution of the stochastic, fractional PDE

$$(-\Delta + \kappa)^{\alpha/2} u = \mathcal{W} \text{ in } \Omega \qquad (6)$$

is a Gaussian random field of Matérn covariance. Here, $\mathcal{W}$ denotes spatial Gaussian white noise, while the fractional exponent $\alpha > \dim(\Omega)/2$ and the parameter $\kappa \geq 0$ define the regularity and correlation length, respectively. Lindgren et al. (2011, 2022) later popularized using finite elements to solve (6), calling the approach the *SPDE method*.

The *SPDE* miniapp provides a scalable solver for (6). It uses a new implementation of the white noise sampling proposed in Croci et al. (2018) to handle the stochastic load. It further employs the triple-A algorithm (Nakatsukasa et al., 2018) to construct a rational approximation of the fractional PDE by a set of integer-order PDEs (Harizanov and Margenov, 2018; Bolin and Kirchner, 2019). The solver
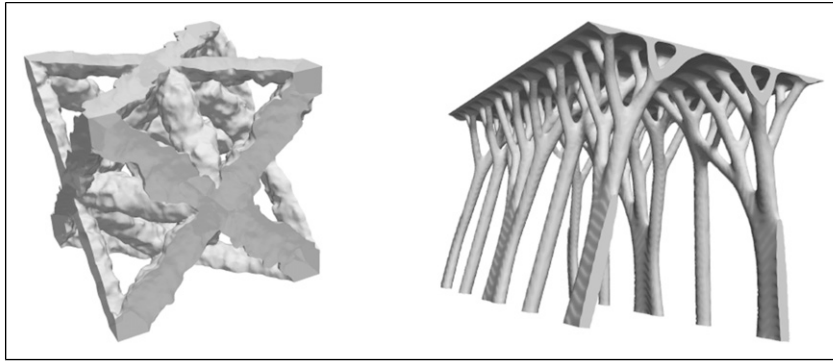
Figure 16. Applications of random fields and the SPDE miniapp. (left) Octet-truss: modeling uncertainties in additive manufacturing. (right) 3D-bridge structure: optimal topology to support an uncertain load (Duswald et al., 2024).

avoids repeated matrix assemblies and solves the associated linear systems with a preconditioned conjugate-gradient algorithm. The full numerical scheme is described in Duswald et al. (2024).

The SPDE-solver is easy to integrate into MFEM-derived projects and may be used to model spatial uncertainties with Matérn-type random fields by adding only two lines of code. The fields subsequently enter into PDE coefficients, the load, or even distort the domain to describe different types of uncertainties (e.g., material, geometric, or environmental uncertainties). Typically, the mathematical model, together with their uncertain SPDE components, is embedded in workflows such as forward uncertainty quantification problems or stochastic inverse problems. Examples ranging from biomechanics to topology optimization under uncertainties are presented in Duswald et al. (2024); Bollapragada et al. (2023). Figure 16 shows a few example applications.

5.2. Hyperbolic conservation laws

A new nonlinear integrator for a general system of first-order hyperbolic conservation laws has been introduced. This integrator implements the element-wise weak divergence and the interface flux,

$$\sum_{E \in \mathcal{M}} \int_E F(\boldsymbol{u}) : \nabla \boldsymbol{v} \ d\boldsymbol{x}, \sum_{e \in \mathcal{E}} \int_e \widehat{F}(\boldsymbol{n}_e; \boldsymbol{u}^{\pm}) \cdot [[\boldsymbol{v}]] \ d\boldsymbol{x}$$

where F is the flux function, and $\widehat{F}$ is the numerical flux function defined on element interfaces. To implement a specific system, users can create a derived class that provides the action of the flux function. The user also specifies the choice of numerical flux function $\widehat{F}$. The Rusanov (local Lax–Friedrichs) flux is provided by MFEM, but other choices of (approximate) Riemann solvers are straightforward to implement by the user. These new capabilities are illustrated for the Euler equations of gas dynamics in an updated example included with MFEM (example 18).

Figure 17 depicts a snapshot of a solution to the shallow water equations computed using the new integrator.

5.3. High-order ALE

Multi-physics Arbitrary Lagrangian-Eulerian (ALE) applications, such as LLNL's MARBL code (Anderson et al., 2020b), are particularly well-suited for using MFEM. One miniapp developed as part of the MARBL/MFEM collaboration is the *Remhos*[2] miniapps. Remhos solves the advection equations that are used to perform monotonic and conservative discontinuous field interpolation (remap) as part of the Eulerian phase in ALE simulations (Anderson et al., 2015, 2017; Hajduk et al., 2020a,b), see left plot of Figure 18. The team also continued its research in Lagrangian compressible hydrodynamics through the Laghos miniapp [Laghos] (Laghos, 2024). Namely, a novel Nitsche-type approach to weakly enforce free-slip wall boundary conditions on curved boundaries was developed (see Atallah et al., 2023b and Figure 18). Furthermore, a weighted Shifted Interface Method (WSIM) was developed, a new immersed method for maintaining exact representation of curved interfaces on high-order Lagrangian grids (Atallah et al., 2023a).

LLNL's MARBL code (Anderson et al., 2020b), a performance-portable multiphysics application built on high-order finite elements, has consistently been on the forefront of leveraging the latest numerical and algorithmic advancements in MFEM. Recognizing the performance advantages offered by matrix-free methods, MARBL has integrated the latest matrix-free algorithms from Laghos, Remhos, and TMOP to enhance its Lagrange, remap, and mesh optimization phases, respectively. Moreover, MARBL has leveraged the work of Pazner et al. (2024) to implement a matrix-free, GPU-accelerated linear solver employing the saddle-point formulation for solving radiation diffusion equations (Stitt et al., 2024). These are some of the essential building blocks that have enabled MARBL

to perform fully matrix-free GPU-enabled practical ALE simulations of multi-material shock and radiation hydro-dynamics, like the ones shown in Figure 19.

5.4. Electromagnetics applications

A range of computational electromagnetics applications have been built using MFEM, benefiting from the library's

support for adaptive mesh refinement, advanced discretizations, and scalable solvers.

One such application is *Petra-M* (Physics Equation Translator for MFEM). Petra-M is an open source GUI platform for the finite element method (Shiraiwa et al., 2017) developed under the Scientific Discovery through Advanced Computing Partnership (SciDAC) program for radio-frequency (RF) wave simulation in fusion plasma (see Figure 20). Petra-M uses MFEM (through the PyMFEM Python interface described in the following section) for its discretization framework and interface to linear solvers. Its weak form interface allows for the construction of complicated linear system for multiphysics and multidomain problems by selecting MFEM's linear and bilinear form integrators. Petra-M's GUI interface allows users to build RF simulation using MFEM, and extending Maxwell's system of equations to develop new implementations of physics models, such as radio-frequency sheath (Shiraiwa et al., 2023), without significant coding efforts. MFEM's efficient high order discretization allows for the inclusion of the complete 3D magnetic fusion plasmas in RF wave simulations (Bertelli et al., 2022).
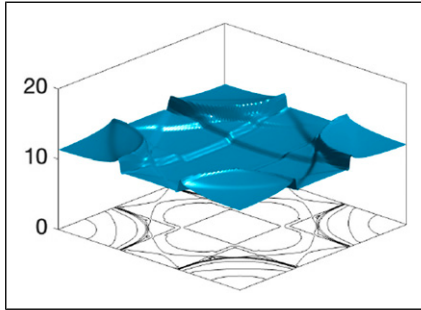
Figure 17. The height of a shallow water wave on a periodic square domain. Initial conditions given by a Gaussian bump.
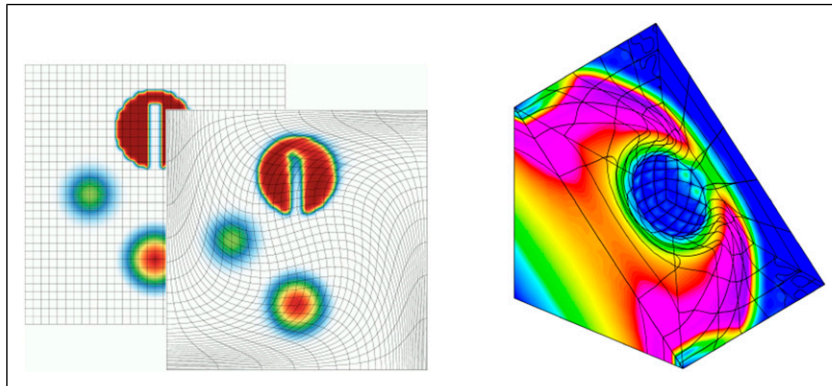
Figure 18. Left: remap computation in Remhos. Right: simulation of explosion inside a cube with a spherical hole in Laghos.
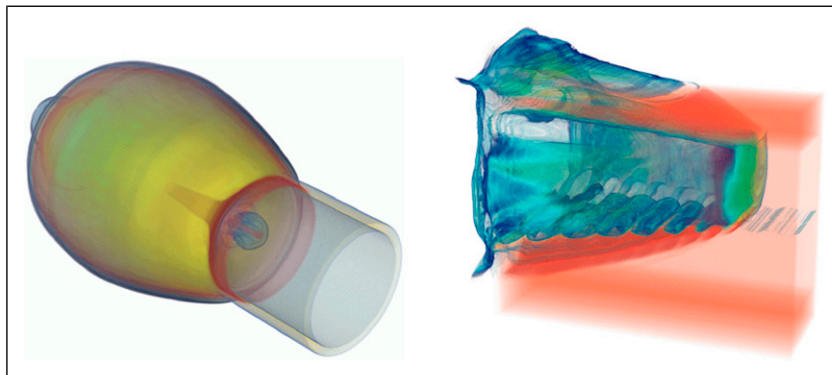
Figure 19. Examples of multi-material ALE simulations in MARBL: BRL81a shaped charge simulation (left) and radiation-driven Kelvin-Helmholtz instability (right).
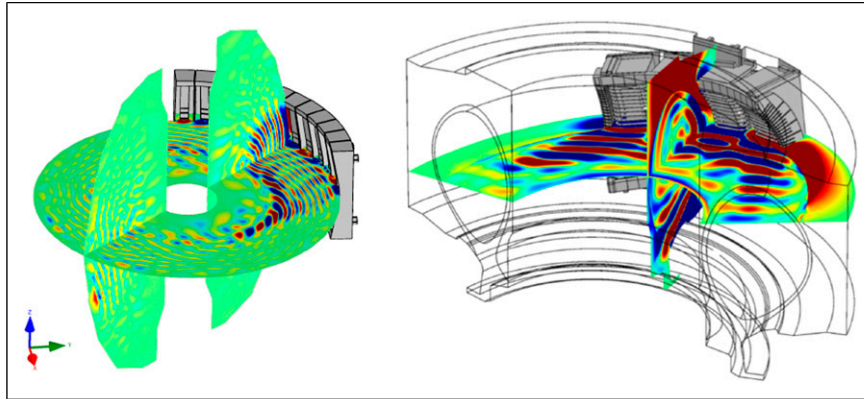
Figure 20. Example of RF wave simulations in plasmas. High harmonic fast wave propagation in the NSTX-U spherical tokamak (left), and ion cyclotron radio-frequency waves excited in the Alcator C-Mod tokamak (right).

A closely related SciDAC effort is in the process of developing an MFEM-based fluid transport application, called *MAPS* (MFEM Anisotropic Plasma Solver), to simulate the neutrals, ions, and electrons which make up a fusion plasma. *MAPS* is designed to be coupled with an RF simulator similar to that explored with *Petra-M*. This coupling will introduce RF heating and pondermotive forces into the fluid equations as well as providing the RF simulator with realistic temperature and density profiles needed to accurately determine the dielectric tensor in Maxwell's equations. Building upon MFEM as a common code base in these two simulation modules will be crucial for efficiently exchanging accurate field information between these coupled physics models.

MFEM is also the discretization framework for the *Palace* (Palace, 2023)[3] finite element code developed by the Design and Simulation group of the AWS Center for Quantum Computing. Palace is used for performing large-scale, full-wave electromagnetic simulations in both frequency and time domain and makes extensive use of MFEM's solvers, discretizations, and adaptive mesh refinement capabilities. Palace also uses the libCEED library for performance-portable linear algebra (Brown et al., 2021); libCEED is one of several runtime-configurable backends that MFEM can target for its partial assembly functionality.

5.5. Python interface (PyMFEM)

MFEM applications can be written using the *PyMFEM* Python interface, in addition to the traditional C++ library API. Python is a high-level, easy-to-use programming language, widely adopted in educational contexts, and among data scientists and domain scientists. The development of PyMFEM was initiated to make the MFEM technology more approachable for this broad range of potential users. In the educational context, PyMFEM allows

for the integration of MFEM and its in situ visualization tool GLVis into interactive Jupyter notebooks.

PyMFEM facilitates the construction and direct access of MFEM data structure and objects from within Python. In the current PyMFEM release, all of the included MFEM example programs are translated into Python. The PyMFEM uses Simple Wrapper Interface Generator (SWIG) for code generation. The SWIG wrapper recipe files are heavily customized so that a user can interact with MFEM in Python native ways.

PyMFEM also supports using Numba just-in-time compiled Python functions for variable coefficients used in MFEM integrators (Lam et al., 2015). This is achieved by decorating a Python function with @mfem.jit. Numba-based coefficients can have dependencies on other coefficients, including MFEM's native coefficients such as GridFunction coefficients, providing a flexible framework for define complex variable coefficients. The following short snippet illustrates how such coefficients can be defined.

```
@mfem.jit.scalar
def product(ptx):
    return ptx[0] * ptx[1]  # return x*y

# Given a GridFunction p
p_coef = mfem.FunctionCoefficient(p)
@mfem.jit.scalar(depencency=(p_coef,))
def scale(ptx, p_coef):
    return ptx[2]*p_coef  # return z*p
```

6. Conclusions

In this paper we provided a brief summary of some of the recent research, development, and advancements in the open-source MFEM finite element library. These include state-of-the-art performance on GPU-based supercomputing architectures, high-performance mesh adaptation and optimization, automatic differentiation, support for

non-standard discretizations, among many others. Such features allow MFEM to power a large number of computational physics and engineering applications in areas such as computational electromagnetics, fractional stochastic PDEs, topology optimization, compressible flow, and more. Further development and improvements to MFEM, including performance optimizations, new discretizations and numerical methods, solvers and preconditioners, miniapps and examples, are continuously underway.

Declaration of conflicting interests

Funding

ORCID iDs

Jan-Phillip Backer https://orcid.org/0000-0003-4521-358X
Dylan Copeland https://orcid.org/0000-0003-3257-8002
Veselin Dobrev https://orcid.org/0000-0003-1793-5622
Yohann Dudouit https://orcid.org/0000-0001-5831-561X
Tobias Duswald https://orcid.org/0009-0000-6937-1676
Brendan Keith https://orcid.org/0000-0002-6969-6857
Dohyun Kim https://orcid.org/0000-0001-9892-0611
Tzanio Kolev https://orcid.org/0000-0002-2810-3090
Ketan Mittal https://orcid.org/0000-0002-2062-852X
Will Pazner https://orcid.org/0000-0003-4885-2934

Notes

1. MFEM (2024) MFEM: Modular finite element methods library. https://mfem.org. doi: 10.11578/DC.20171025.1248.
2. Remhos (2024) Remhos: Remap high-order solver. https://github.com/CEED/Remhos
3. Palace (2023) Palace: Parallel large-scale computational electromagnetics. https://github.com/awslabs/palace.

References

Abadi M, Barham P, Chen J, et al. (2016) TensorFlow: a system for large-scale machine learning. 12th USENIX symposium on operating systems design and implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016, 265–283.

Abdelfattah A, Barra V, Beams N, et al. (2021) GPU algorithms for efficient exascale discretizations. *Parallel Computing* 108: 102841. DOI: 10.1016/j.parco.2021.102841.

Anderson RW, Dobrev VA, Kolev TV, et al. (2015) Monotonicity in high-order curvilinear finite element arbitrary Lagrangian-Eulerian remap. *International Journal for Numerical Methods in Fluids* 77(5): 249–273. DOI: 10.1002/fld.3965.

Anderson RW, Dobrev VA, Kolev TV, et al. (2017) High-order local maximum principle preserving (MPP) discontinuous Galerkin finite element method for the transport equation. *Journal of Computational Physics* 334: 102–124. DOI: 10.1016/j.jcp.2016.12.031.

Anderson RW, Dobrev VA, Kolev TV, et al. (2018) High-order multi-material ALE hydrodynamics. *SIAM Journal on Scientific Computing* 40(1): B32–B58. DOI: 10.1137/17m1116453.

Anderson R, Black A, Busby L, et al. (2020b) The multiphysics on advanced platforms project. Technical Report LLNL-TR-815869, Lawrence Livermore National Lab.(LLNL), Livermore, CA. DOI: 10.2172/1724326

Anderson R, Andrej J, Barker A, et al. (2020a) MFEM: a modular finite element methods library. *Computers & Mathematics with Applications* 81: 42–74. DOI: 10.1016/j.camwa.2020.06.009.

Aslam TD (2004) A partial differential equation approach to multidimensional extrapolation. *Journal of Computational Physics* 193(1): 349–355. DOI: 10.1016/j.jcp.2003.08.001.

Atallah NM, Canuto C and Scovazzi G (2020) The second-generation shifted boundary method and its numerical analysis. *Computer Methods in Applied Mechanics and Engineering* 372: 113341. DOI: 10.1016/j.cma.2020.113341.

Atallah NM, Mittal K, Scovazzi G, et al. (2023a) *A High-Order Shifted Interface Method for Lagrangian Shock Hydrodynamics*. arXiv preprint arXiv: 2309.11821

Atallah NM, Tomov VZ and Scovazzi G (2023b) *Weak Boundary Conditions for Lagrangian Shock Hydrodynamics: A High-Order Finite Element Implementation on Curved Boundaries*. arXiv preprint arXiv: 2309.00785.[

Barrera JL, Kolev T, Mittal K, et al. (2023) High-order mesh morphing for boundary and interface fitting to implicit geometries. *Computer-Aided Design* 158: 103499. DOI: 10.1016/j.cad.2023.103499.

Belyaev AG and Fayolle PA (2015) On variational and PDE-based distance function approximations. *Computer Graphics Forum* 34(8): 104–118. DOI: 10.1111/cgf.12611.

Bertelli N, Shiraiwa S and Ono M (2022) 3D full wave fast wave modeling with realistic HHFW antenna geometry and SOL plasma in NSTX-U. *Nuclear Fusion* 62(12): 126046. DOI: 10.1088/1741-4326/ac9690

Bochkov D and Gibou F (2020) PDE-based multidimensional extrapolation of scalar fields over interfaces with kinks and high curvatures. *SIAM Journal on Scientific Computing* 42(4): A2344–A2359. DOI: 10.1137/19m1307883.

Bolin D and Kirchner K (2019) The rational SPDE approach for Gaussian random fields with general smoothness. *Journal of Computational & Graphical Statistics* 29: 274–285.

Bollapragada R, Karamanli C, Keith B, et al. (2023) *An Adaptive Sampling Augmented Lagrangian Method for Stochastic Optimization With Deterministic Constraints*. arXiv preprint arXiv:2305.01018.

Brown J, Abdelfattah A, Barra V, et al. (2021) libCEED: fast algebra for high-order element-based discretizations. *Journal of Open Source Software* 6(63): 2945. DOI: 10.21105/joss.02945.

Burman E, Claus S, Hansbo P, et al. (2015) Cutfem: discretizing geometry and partial differential equations. *International Journal for Numerical Methods in Engineering* 104(7): 472–501. DOI: 10.1002/nme.4823

Camier JS, Dobrev V, Knupp P, et al. (2023) Accelerating high-order mesh optimization using finite element partial assembly on GPUs. *Journal of Computational Physics* 474: 111808. DOI: 10.1016/j.jcp.2022.111808.

Canuto C (1994) Stabilization of spectral methods by finite element bubble functions. *Computer Methods in Applied Mechanics and Engineering* 116(1-4): 13–26. DOI: 10.1016/s0045-7825(94)80004-9.

Carstensen C, Demkowicz L and Gopalakrishnan J (2016) Breaking spaces and forms for the DPG method and applications including Maxwell equations. *Computers & Mathematics with Applications* 72(3): 494–522.

Crane K, Weischedel C and Wardetzky M (2017) The heat method for distance computation. *Communications of the ACM* 60(11): 90–99. DOI: 10.1145/3131280.

Croci M, Giles MB, Rognes ME, et al. (2018) Efficient white noise sampling and coupling for multilevel Monte Carlo with nonnested meshes. *SIAM/ASA Journal on Uncertainty Quantification* 6: 1630–1655. DOI: 10.1137/18M1175239.

Demkowicz L and Gopalakrishnan J (2010) A class of discontinuous Petrov–Galerkin methods. part I: the transport equation. *Computer Methods in Applied Mechanics and Engineering* 199(23-24): 1558–1572.

Demkowicz L and Gopalakrishnan J (2011) A class of discontinuous Petrov–Galerkin methods. II. Optimal test functions. *Numerical Methods for Partial Differential Equations* 27(1): 70–105.

Demkowicz L, Gopalakrishnan J and Niemi AH (2012) A class of discontinuous Petrov–Galerkin methods. part III: adaptivity. *Applied Numerical Mathematics* 62(4): 396–427.

Dobrev V, Knupp P, Kolev T, et al. (2020) Simulation-driven optimization of high-order meshes in ALE hydrodynamics.

Computers & Fluids 208: 104602. DOI: 10.1016/j.compfluid.2020.104602.

Dobrev V, Knupp P, Kolev T, et al. (2021) *hr*-adaptivity for nonconforming high-order meshes with the target matrix optimization paradigm. *Engineering with Computers* 38(4): 3721–3737. DOI: 10.1007/s00366-021-01407-6.

Dobrev V, Knupp P, Kolev T, et al. (2024) *Asymptotic Analysis of Compound Volume + Shape Metrics for Mesh Optimization*. Preprint submitted to arXiv.

Duswald T, Keith B, Lazarov B, et al. (2024) *Finite Element Analysis for Matérn-type Random Fields: Uncertainties in Computational Mechanics and Design Optimization*. (In preparation).

Fischer P (1997) An overlapping Schwarz method for spectral element solution of the incompressible Navier–Stokes equations. *Journal of Computational Physics* 133(1): 84–101. DOI: 10.1006/jcph.1997.5651.

Fischer P, Min M, Rathnayake T, et al. (2020) Scalability of high-performance PDE solvers. *The International Journal of High Performance Computing Applications* 34(5): 562–586.

Griewank A, Juedes D and Utke J (1996) Algorithm 755: ADOL-C: a package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software* 22(2): 131–167.

Hajduk H, Kuzmin D, Kolev TV, et al. (2020a) Matrix-free subcell residual distribution for Bernstein finite element discretizations of linear advection equations. *Computer Methods in Applied Mechanics and Engineering* 359: 112658. DOI: 10.1016/j.cma.2019.112658.

Hajduk H, Kuzmin D, Kolev TV, et al. (2020b) Matrix-free subcell residual distribution for Bernstein finite elements: monolithic limiting. *Computers & Fluids* 200: 104451. DOI: 10.1016/j.compfluid.2020.104451.

Harizanov S and Margenov S (2018) *Positive Approximations of the Inverse of Fractional Powers of SPD M-Matrices*. Berlin, Germany: Springer, 147–163.

Hascoët L and Pascual V (2013) The Tapenade automatic differentiation tool: principles, model, and specification. *ACM Transactions on Mathematical Software* 39(3): 1–43.

Hiemstra RR, Sangalli G, Tani M, et al. (2019) Fast formation and assembly of finite element matrices with application to isogeometric linear elasticity. *Computer Methods in Applied Mechanics and Engineering* 355: 234–260. DOI: 10.1016/j.cma.2019.06.020.

Keith B and Surowiec TM (2023) Proximal Galerkin: a structure-preserving finite element method for pointwise bound constraints. *arXiv preprint arXiv:2307.12444*.

Knupp P (2019) *Target Formulation and Construction in Mesh Quality Improvement*. Livermore, CA: Lawrence Livermore National Laboratory. Technical Report LLNL-TR-795097.

Knupp P (2020) *Metric Type in the Target-Matrix Mesh Optimization Paradigm*. Livermore, CA (United States): Lawrence Livermore National Lab.(LLNL). Technical Report LLNL-TR-817490.

Knupp P (2023) Seventeen criteria for evaluating Jacobian-based optimization metrics. *Engineering with Computers*. DOI: 10.1007/s00366-023-01869-w

Kolev T and Pazner W (2022) Conservative and accurate solution transfer between high-order and low-order refined finite element spaces. *SIAM Journal on Scientific Computing* 44(1): A1–A27. DOI: 10.1137/21m1403916.

Kolev T, Fischer P, Min M, et al. (2021) Efficient exascale discretizations: high-order finite element methods. *The International Journal of High Performance Computing Applications* 35: 527–552. DOI: 10.1177/10943420211020803.

Kolev T, Fischer P, Beams N, et al. (2021) *ECP Milestone Report CEED-MS37: Port and Optimize the CEED Software Stack to Aurora/Frontier EA*. USA: OSTI

Laghos (2024) *Laghos: High-Order Lagrangian Hydrodynamics Miniapp*. https://github.com/CEED/Laghos.

Lam SK, Pitrou A and Seibert S (2015) Numba: a LLVM-based Python JIT compiler. *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. New York: ACM, SC15.

Lattner C and Adve V (2004) LLVM: a compilation framework for lifelong program analysis & transformation. International symposium on code generation and optimization (CGO), San Jose, California, 20 March 2004, 75–86. DOI: 10.1109/CGO.2004.1281665.

Lawson CL and Hanson RJ (1995) *Solving Least Squares Problems*. Philadelphia: Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9781611971217.

Lindgren F, Rue H and Lindström J (2011) An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society - Series B: Statistical Methodology* 73: 423–498. DOI: 10.1111/j.1467-9868.2011.00777.x.

Lindgren F, Bolin D and Rue H (2022) The SPDE approach for Gaussian and non-Gaussian fields: 10 years and still running. *Spatial Statistics* 50: 100599. DOI: 10.1016/j.spasta.2022.100599.

Logg A, Mardal KA and Wells G (eds) (2012). *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book, Lecture Notes in Computational Science and Engineering*. Berlin; Springer Berlin Heidelberg, Vol. 84. DOI: 10.1007/978-3-642-23099-8.

MFEM (2024) *Mfem: Modular Finite Element Methods Library*. https://mfem.org.10.11578/DC.20171025.1248

Mittal K, Dobrev VA, Knupp P, et al. (2024) *Mixed-Order Meshes through Rp-Adaptivity for Surface Fitting to Implicit Geometries*. arXiv preprint arXiv:2401.16369

Moses WS, Churavy V, Paehler L, et al. (2021) Reverse-mode automatic differentiation and optimization of GPU kernels via Enzyme. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '21*. New York: Association for Computing Machinery, 1–16. DOI: 10.1145/3458817.3476165.

Müller B, Kummer F and Oberlack M (2013) Highly accurate surface and volume integration on implicit domains by means of moment-fitting. *International Journal for Numerical Methods in Engineering* 96(8): 512–528. DOI: 10.1002/nme.4569.

Nakatsukasa Y, Sète O and Trefethen LN (2018) The AAA algorithm for rational approximation. *SIAM Journal on Scientific Computing* 40: A1494–A1522. DOI: 10.1137/16M1106122.

Orszag S (1980) Spectral methods for problems in complex geometries. *Journal of Computational Physics* 37(1): 70–92. DOI: 10.1016/0021-9991(80)90005-4.

Palace (2023) *Palace: Parallel Large-Scale Computational Electromagnetics*. https://github.com/awslabs/palace.

Paszke A, Gross S, Chintala S, et al. (2017) Automatic differentiation in PyTorch. NIPS 2017 Workshop Autodiff, Long Beach, California, USA, 2017

Pazner W (2020) Efficient low-order refined preconditioners for high-order matrix-free continuous and discontinuous Galerkin methods. *SIAM Journal on Scientific Computing* 42(5): A3055–A3083. DOI: 10.1137/19m1282052.

Pazner W and Kolev T (2021) Uniform subspace correction preconditioners for discontinuous Galerkin methods with *hp*-refinement. *Communications on Applied Mathematics and Computation* 4: 697–727. DOI: 10.1007/s42967-021-00136-3.

Pazner W, Kolev T and Camier JS (2023a) End-to-end GPU acceleration of low-order-refined preconditioning for high-order finite element discretizations. *The International Journal of High Performance Computing Applications* 37: 578–599. DOI: 10.1177/10943420231175462.

Pazner W, Kolev T and Dohrmann CR (2023b) Low-order preconditioning for the high-order finite element de Rham complex. *SIAM Journal on Scientific Computing* 45(2): A675–A702. DOI: 10.1137/22M1486534.

Pazner W, Kolev T and Vassilevski PS (2024) *Matrix-free High-Performance Saddle-point Solvers for High-Order Problems in H(div)*. USA: Accepted for publication in the SIAM Journal on Scientific Computing. arXiv eprint 2304.12387.

Petrides S and Demkowicz L (2017) An adaptive DPG method for high frequency time-harmonic wave propagation problems. *Computers & Mathematics with Applications* 74(8): 1999–2017.

Petrides S and Demkowicz L (2021) An adaptive multigrid solver for DPG methods with applications in linear acoustics and electromagnetics. *Computers & Mathematics with Applications* 87: 12–26.

Remhos (2024) *Remhos: Remap High-Order Solver*. https://github.com/CEED/Remhos.

Saye RI (2015) High-order quadrature methods for implicitly defined surfaces and volumes in hyperrectangles. *SIAM Journal on Scientific Computing* 37(2): A993–A1019. DOI: 10.1137/140966290.

Shiraiwa S, Wright JC, Bonoli PT, et al. (2017) RF wave simulation for cold edge plasmas using the MFEM library. *EPJ*

Web of Conferences 157: 03048. DOI: 10.1051/epjconf/201715703048.

Shiraiwa S, Bertelli N, Tierens W, et al. (2023) Magnetic potential based formulation for linear and non-linear 3D RF sheath simulation. *Nuclear Fusion* 63(2): 026024. DOI: 10.1088/1741-4326/aca6f9.

Stitt T, Belcher K, Campos A, et al. (2024) Performance portable graphics processing unit acceleration of a high-order finite element multiphysics application. *Journal of Fluids Engineering* 146: 1–69doi. DOI: 10.1115/1.4064493.

Whittle P (1954) On stationary processes in the plane. *Biometrika* 41: 434. DOI: 10.2307/2332724.

Whittle P (1963) Stochastic processes in several dimensions. *Bulletin of the International Statistical Institute* 40: 974–994.

Zitelli J, Muga I, Demkowicz L, et al. (2011) A class of discontinuous Petrov–Galerkin methods. part IV: the optimal test norm and time-harmonic wave propagation in 1D. *Journal of Computational Physics* 230(7): 2406–2432.

Author biographies

Julian Andrej is a computational mathematician at the Center for Applied Scientific Computing (CASC) in Lawrence Livermore National Laboratory. He is focusing on research including High-Order Computational Fluid Dynamics, Fluid-Structure-Interaction and Automatic Differentiation.

Nabil Atallah is a computational mathematician at the Center for Applied Scientific Computing in Lawrence Livermore National Laboratory. His current research focus is in developing novel finite element methods aimed at reducing overall design/analysis cost in various applications including fluid and solid mechanics, compressible shock and multi-material ALE hydrodynamics.

Jan-Phillip Bäcker is a PhD student at the Institute of Applied Mathematics at TU Dortmund University. His research includes unfitted methods for interface problems and level set approaches for tumor growth models.

John Camier is a research scientist in the numerical analysis and simulations group at the Center for Applied Scientific Computing in Lawrence Livermore National Laboratory. His current research focus is on parallel computer architectures, performance portability, and computing with high-order finite element methods.

Dylan Copeland is a computational mathematician in the numerical analysis and simulations group in the Center for Applied Scientific Computing. His current research interests include finite element methods, adaptive meshing, NURBS meshing, isogeometric analysis, and reduced order models.

Veselin Dobrev is a computational mathematician in the numerical analysis and simulations group in the Center for Applied Scientific Computing. His research interests are in the area of numerical methods for solving PDEs, including finite element and discontinuous Galerkin methods, matrix-free methods, shock hydrodynamics simulations, and iterative and multigrid methods.

Yohann Dudouit is a computational mathematician at the Center for Applied Scientific Computing in Lawrence Livermore National Laboratory. His research currently focuses on high performance computing for finite element methods, with an emphasis on high-dimension physics, matrix-free methods, discontinuous Galerkin methods, and GPU architectures.

Tobias Duswald is a PhD student in the School of Computation, Information, and Technology at the Technical University of Munich. Currently affiliated with CERN's IT department, he specializes in the application of computational models across diverse scientific domains. His research spans mathematical modeling in cancer and neuroscience, Bayesian inference methodologies, PDE-based modeling approaches, and random field generation.

Brendan Keith is an assistant professor of applied mathematics in the Division of Applied Mathematics at Brown University. He works in the area of scientific computing and relies on MFEM for much of his research. Among numerous other efforts, he is deeply invested in developing scalable numerical methods for variational problems with constraints. Brendan's most recent contributions to the MFEM library include software tools for optimal design, random fields, and free boundary problems. Each of the aforementioned contributions are highlighted in this document.

Dohyun Kim is a postdoctoral research associate in Division of Applied Mathematics at Brown University. His research interests include finite element methods for PDE constrained optimization and hyperbolic conservation laws.

Tzanio Kolev is a computational mathematician at the Center for Applied Scientific Computing (CASC) in Lawrence Livermore National Laboratory, where he works on finite element discretizations and solvers for problems in compressible shock hydrodynamics, multi-material arbitrary Lagrangian Eulerian methods, radiation hydrodynamics, and computational electromagnetics. Tzanio is leading the high-order finite element discretization research and development efforts in the MFEM and BLAST projects in CASC and was the director of the Center for Efficient Exascale Discretizations (CEED) in DOE's Exascale Computing Project.

Boyan Lazarov is a research engineer in the Computational Engineering Division at LLNL. His research focuses on the development of methods and tools to model, design, and optimize coupled multi-physical systems with the help of high-performance computing. Application areas include fluid-heat transfer systems, photonics, acoustics, structural

and mechanical designs in the automotive and aerospace industries.

Ketan Mittal is a computational mathematician at the Center for Applied Scientific Computing in Lawrence Livermore National Laboratory. Ketan's research interests include high-order mesh generation and optimization methods, data-driven approaches for adaptive mesh refinement, computational geometry, overlapping grid-based approaches, and time-stepping. Ketan has contributed in the development of open-source frameworks MFEM, Nek5000, and gslib.

Will Pazner is an assistant professor in the Fariborz Maseeh Department of Mathematics and Statistics at Portland State University, where his research focuses on high-order finite elements, efficient solvers and preconditioners, and high-performance computing. Previously, he was the Sidney Fernbach Postdoctoral Fellow at Lawrence Livermore National Laboratory's Center for Applied Scientific Computing.

Socratis Petrides is a computational mathematician at the Center for Applied Scientific Computing in Lawrence Livermore National Laboratory. His work focuses on finite

element methods, computational electromagnetics, multi-level solvers, constrained optimization and scientific machine learning.

Syun'ichi Shiraiwa is a physicist in Princeton Plasma Physics Laboratory. His current focus is computational modeling of radio-frequency waves in hot fusion plasmas. He is a maintainer of the PyMFEM python wrapper for MFEM and is developing the Petra-M FEM framework using MFEM discretization technology.

Mark Stowell is a computational mathematician at the Center for Applied Scientific Computing in Lawrence Livermore National Laboratory. His research interests include finite element methods for computational electromagnetics and high-order discretization.

Vladimir Tomov is a computational mathematician at the Center for Applied Scientific Computing in Lawrence Livermore National Laboratory. His research interests include finite element methods for multi-material ALE and radiation hydrodynamics, immersed discretizations, and high-order mesh optimization.