# MFEM: A Modular Finite Element Methods Library

Elvin Gültekinoğlu

October 2023

## 1 The Motivation Behind This Paper

This document is prepared to summarize the key points of the article which is named as "MFEM: A Modular Finite Element Methods Library" and published in 2021. Throughout the text, the main points are highlighted and expressed with equations, figures and tables.

## 2 Abstract

In R. Anderson, J. Andrej, A. Barker et al [1] MFEM is a C++ based library which is used for modular finite element methods related to arbitrary high-order finite element meshes and spaces and similar approaches. It becomes prominent with the fact that it is open-source, lightweight, flexible and scalable, which makes it easily accessible, portable and suitable for high-performance computing operations. Considering all these advantages, researchers are able to test their algorithms in settings which are high-order, parallel and GPU-accelerated. This paper is focused on the utilization of this library with theoretical background, explanations and applications.

## 3 Introduction

Finite Element Method (FEM) is used to approximate the solutions of partial differential equations (PDE). It is mainly a discretization technique that utilizes unstructured grids. MFEM provides a support for these techniques and it is widely used due to its certain advantages. Also, being an open-source library, its source code is easily accessible on Spack, OpenHPC, and Github.

MFEM offers certain advantages which makes it distinguisable from other similar packages used for FEM. These advantages come from its usability, accessibility and HPC efficiency. It also enables other researchers or scientists to contribute easily. With its positive features, MFEM is used in different general fields such as manufacturing, hydrodynamics, heat transfer, propulsion and paralellization.

MFEM provides support for different type of meshes, solvers and numerical methods. In addition to these, it is utilizable for Message Passing Interface (MPI)-based paralellism, GPU acceleration with CUDA, OCCA, RAJA and OpenMP.

# 4 Theoretical Background

In this section, general overview about the theory behind FEM and MFEM library is provided with mathematical expressions and related figures.

## 4.1 Finite Element Abstractions

In order to use MFEM to solve related problems, there are certain classes, sub-classes, object types and functions which are used to implement a specific method according to the needs of the problem. To show this process, an example case can be considered, which is the model Poisson problem with homogeneous boundary conditions. The problem is defined as follows:

Find u : $\Omega \rightarrow$ R such that

-$\Delta$u = f in $\Omega$

The approximate solution is obtained by solving the FEM problem with the given mesh. The problem is reduced to the following integral:

$$\int_{\Omega} \nabla u_h \nabla v_h = \int_{\Omega} f v_h \tag{1}$$

This integral is manupilated by defining basis function $\phi$ for the related space $V_h$ and the Equation 1 is rewritten. For the rewritten version, see the Equation 2. By calling necessary methods from revelant classes and methods, the solution is obtained.

$$\Sigma_{j=0} c_j \int_{\Omega} \nabla \phi \nabla \phi = \int_{\Omega} f \phi \tag{2}$$

## 4.2 Meshes

As mentioned in Section 3 MFEM is used for different type of meshes such as triangular, quadrilateral, tetrahedral, hexahedral and prismatic. Considering these types, there are two main classes available for meshes in the software. These are "Mesh" and "ParMesh" classes. The former one is used for a serial mesh and the latter one is used for an MPI-distributed parallel mesh.

### 4.2.1 The Types of Meshes

Serial, unstructured conforming mesh in MFEM is described in terms of two different parts: topological (connectivity) data and geometric (coordinates) data. Topological part is mainly about the type, attribute, vertex indices, edges, faces and connection elements of meshes. Geometric part describes the aspects related to spatial

features of meshes such as coordinates of vertices. To define the precise shape of a mesh element, mapping is performed. This process is presented in the Figure 1. The driving equation behind this mapping operation is given in the Equation 3 .

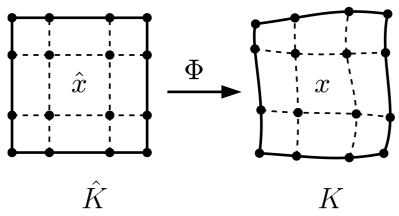


Figure 1: The mapping of an element from the reference element to bi-cubic element in K physical space with high-order nodes.

$$\Omega(x) = \Sigma_{i=1}^{N} x_{K_i} w(x) \tag{3}$$

Other types of meshes are non-conforming meshes which are defined as conforming meshes with additional constraints, NURBS meshes used in geometric modeling mainly and parallel meshes for MPI-based applications.

# 5 High Performance Computing

In this section, high performance computing feature of MFEM is examined.

## 5.1 Parallel Meshes

MFEM is a proper software to be used in applications involving large scale parallelism with Message Passing Interface (MPI). In parallel solving, the problem is analyzed in pieces after it is decomposed into K different parts. As an example case, solution of a Poisson problem is given in the Figure 2.

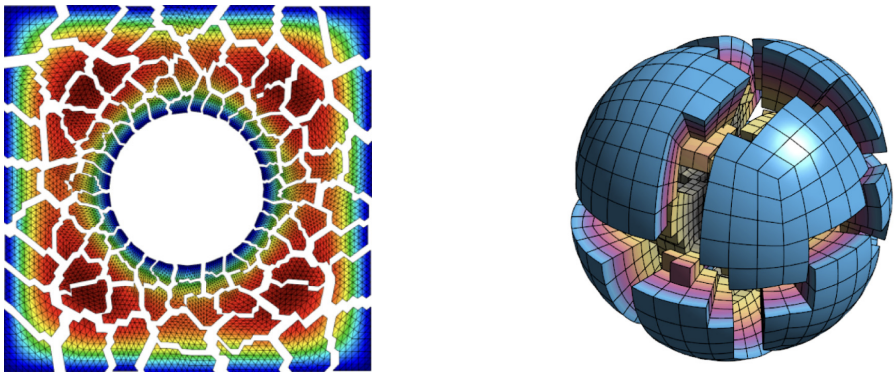

Figure 2: Left: Solving a Poisson problem in parallel on 100 processors. Right: Unstructured parallel decomposition of a fourth order NURBS mesh of the unit ball on 16 processors.

## 5.2 GPU Acceleration

With the release of version of 4.0 MFEM, the compatibility with hardware accelerators is also announced, which amounts to the opportunity of using GPUs and libraries such as CUDA, OCCA, libCEED, RAJA and OpenMP. This feature leads to certain advantages. One main advantage of MFEM about this is being able to select backends at runtime so as to increase performance. MFEM achieves this by its modular design for accelerator support which is given in the Figure 3.

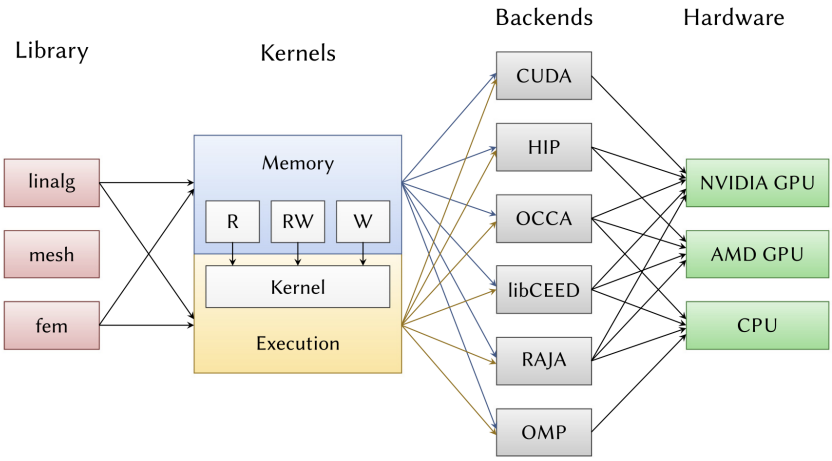


Figure 3: Diagram of MFEM's modular design for accelerator support.

In order to evaluate the performance of aforementioned aspects, performance tests are conducted. These tests are performed on a Linux-based computer with GPU,

multi-core and CPU along with different libraries. As a result, it is observed that GPU acceleration has a positive effect on performance. Test results are provided in the Table 1.

| | | p=1 | p=2 |
|---|---|---|---|
| GPU | OCCA-CUDA | 0.52 | 0.31 |
| | RAJA-CUDA | 0.38 | 0.30 |
| | CUDA | 0.36 | 0.26 |
| | CEED-CUDA | 0.19 | 0.15 |
| Multicore | OCCA-OMP | 3.34 | 2.41 |
| | RAJA-OMP | 3.32 | 2.45 |
| | OMP | 3.30 | 2.46 |
| | MPI | 2.72 | 1.66 |
| CPU | OCCA-CPU | 21.05 | 15.77 |
| | RAJA-CPU | 45.42 | 16.53 |
| | CPU | 25.18 | 16.11 |
| | CEED-AVX | 43.04 | 18.16 |
| | CEED-XSMM | 53.80 | 20.13 |

Table 1: Performance results with MFEM-4.0

# 6 Conclusion

In this paper, MFEM library is introduced with its characteristics, algorithms, advantages and applications as a summary. MFEM is getting more and more popular these days among scientists and researchers because of its easiness and accessibility as well as high performance capability.

# References

[1] Robert Anderson et al. “MFEM: A modular finite element methods library”. In: *Computers & Mathematics with Applications* 81 (2021), pp. 42–74.