An Interface from M++ to Ginkgo for Accelerated Linear Algebra

January 27, 2026

People:	Niklas Baumgarten, Marcel Koch, Simon Wülker, Tobias Merkel, Tim Schrader
Project Title:	An Interface from M++ to Ginkgo for Accelerated Linear Algebra
Duration:	$\leq$ 6 months, possible extension
Type of Work:	Feature development, Parallelization, GPU support
Scientific Field:	Applied Mathematics
Subfield:	Numerical Linear Algebra and Uncertainty Quantification
Existing Codebase:	Open Source
License:	GPL
Link to repository:	`https://gitlab.kit.edu/kit/mpp/mpp/` (latest release: 3.4.1)
Programming Languages:	C++, Python (only for tooling)
Involved Technologies:	MPI, OpenMP, Cuda
Target Cluster:	Horeka (NHR Project UQHPC: 120k GPUh, 6.2m CPUh)

Problem Statement

Solving linear systems is a critical part of the finite element (FE) library M++ [2]. Until now, these linear solvers are developed by the M++ team themselves, which led to an increased maintenance burden on the developer team, as well as less functionality than specialized libraries. Thus, the task of solving linear systems shall be handed off to a specialized library.

Proposed Solution

The numerical linear algebra library Ginkgo [1] is introduced as a backend for solving linear systems and generic linear algebra operations. Only a small interface between M++ and Ginkgo will be introduced, which delegates all most all operations to Ginkgo. The new backend will most likely not cover the full functionality of the existing backend. Thus, the new Ginkgo backend will co-exist with the existing backend. Only after the Ginkgo backend fulfills all needs, the homegrown backend will be removed. *Note: This might not be achievable in the given timeframe.*

Detailed Project Description

This project aims to develop an interface between M++[2], a finite element (FE) library, and Ginkgo [1], a high-performance solver package for sparse linear systems supporting GPU acceleration. Ginkgo has been successfully integrated with other projects, e.g. OpenFOAM [7], whereas M++ offers unique applications and algorithms in uncertainty quantification [3], optimal control, inverse problems [4], cardio-vascular simulations [6] and wave propagation [5].

The goal is to enable M++ to solve large-scale linear systems arising, for example, from sample-batched FE simulations or space-time discretizations, using Ginkgo's linear solvers. This will speed up existing algorithms in M++, but also enable new applications. The project involves designing, implementing and testing the new interface, while ensuring it supports existing applications in M++ and solvers in Ginkgo. The integration should be as tight as possible and as intrusive as necessary, potentially even replacing existing linear algebra components in M++ with Ginkgo's components. We suspect that by doing so, we can achieve significant performance improvements, increased robustness of the overall system and enable new applications on novel hardware architectures.

Short summary of the project goals:

- Ginkgo solvers can be selected in addition to the existing linear system solvers.
- Minimal overhead from running Ginkgo solvers with M++ data structures.
- Demonstrate benefit of new solvers.

Objectives and Outline of the Project

To realize the interface between M++ and Ginkgo, we have identified the following goals and challenges. This description is not exhaustive and is refined during the project as new insights are gained and unforeseen issues arise. The current status is (Prelimianary

work by Suryansh Chaturvedi in his Master's thesis): Both tools compile together where M++ includes Ginkgo as a submodule. M++ can call Ginkgo functions, but the data structures are not yet compatible. First small test cases are defined but do not run yet.

Milestones

We define the following milestones for the project, together with the estimated time of completion. Each milestone is described in more detail below.

1. Integration of Ginkgo into the M++ build system and CI. **Month 1**
2. An additional interface to Ginkgo's solvers, which take the existing M++ sequential vectors and matrices as input. **Month 1-2**
3. An additional interface to Ginkgo's sequential vectors and matrices. The Ginkgo solvers can now use those as input. **Month 4**
4. Performance evaluation on specified benchmarks. **Month 4-5**
5. Extension to distributed data structures. **Stretch milestone**

Milestone 1: The pre-existing work already contains a simple test combining M++ and Ginkgo data structures. This uses Ginkgo as a standalone, external dependency, and doesn't use any specialized interface between both libraries. The test is extended to be run as part of the CI. Additionally, the Git-submodule based inclusion of Ginkgo is replaced by using Ginkgo as an external package through CMake. This also requires adapting the CI setup of M++ to provide containers with Ginkgo preinstalled.

Milestone 2: A first interface between M++ and Ginkgo is implemented, restricted to the Ginkgo solvers. The solvers will take M++ data structures as input. This implies that data transfers between M++ and Ginkgo are required when using these solvers. Specifically, the sparse matrix format of M++ is used and copied into the CSR format of Ginkgo. It should be noted that the CSR format is not tested in M++ and the conversion is an open question. As a first step to finding the right format and design, the existing matrix formats and operators in both, M++ and Ginkgo, have to be understood to get an idea how they can be mapped. Based on the results from MS 1, the interface is included into the testing infrastructure, where the first tests will consist of a Poisson problem on a simple geometry, discretized with standard linear FE.

Milestone 3: After collecting initial experiences with both tools in MS 1 & 2, design choices on the interface are made. The goal is to develop a tight integration, where sequential linear algebra objects from Ginkgo, such as vectors and matrices, are wrapped in M++ objects to provide most of the existing linear algebra interface in M++. With this integration, the wrapped solvers from MS 3 can be applied to Ginkgo objects directly. Most of the previously introduced overhead should be reduced. To achieve this milestone, the linear algebra module of M++ can (probably must) undergo major changes. However, compatibility with existing projects must be maintained and verified through the existing multi-project CI/CD pipeline.

Milestone 4: Within the final phase of the project, the interface is finalized and tested within M++ applications on the HoreKa system. Benchmarking and performance tests are executed and documented in preparation for publication. The focus is on the single-node performance of the added interface. New objectives are defined for potential follow-up projects.

Stretch Milestone: After MS 4 and during MS 5, the extension of the interface to distributed data structures is evaluated. If those findings indicate that the extension can be realized in ~ 1 month time, then this MS will be included. Otherwise, initiating a new project will be suggested.

Deliverables

At the end of the project, we deliver to the M++ developer team:

- Sequential M++ applications can Ginkgo solvers. The full functionality of (sequential) Ginkgo is available, including selecting different solvers and running on GPUs.
- The Ginkgo support in M++ is thoroughly tested as part of M++'s CI.

Outside of Scope:

While integrating Ginkgo as a linear algebra backend will help to use modern hardware more effectively, it is only one part of the full picture. To achieve full efficiency, other parts of M++ have to be adapted as well. The most important aspect is assembling the system matrixes and right-hand-sides on the GPU, which would remove most of the currently necessary host-device communication. Since this is a topic where other finite element libraries, such as deal.II, DUNE, or MFEM, also struggle, or just don't support it, it is unrealistic to support this in addition to the tasks defined above.

The existing, homegrown backend might contain functionality that doesn't have a directly corresponding Ginkgo equivalent. An example could be the handling of constraints. As a consequence, the Ginkgo backend might not be able to cover 100% of the existing use-cases for the linear algebra backend. It will be necessary to keep the homegrown and Ginkgo backend around at the same time.
Summary:

- Finite element assembly on the GPU.
- Deletion of the existing LA backend.
- Supporting unusual finite element types.

Challenges and Questions

- How can existing MPI parallelization and distributed data-structures be extended to Ginkgo? *Addressed in stretch MS.*
- How can the data, organized via hash maps where the hash of each nodal point is mapped to a coefficient in the finite element vector, be mapped to the Ginkgo data structures? *Addressed in MS 3.*
- How can the data directly be assembled in Ginkgo data-structures using M++ routines? *Addressed in MS 3.*

- How should boundary conditions be handled? *Addressed in MS 3.*
- How are the domain boundaries of MPI ranks handled? *Addressed in stretch MS.*
- How do central objects in M++ (e.g. `MatrixGraph`, `Operator`, `Matrix`, `Vector`) be changed to address the questions above? *Addressed in MS 3.*

Expectations on the M++ Team

The bwRSE4HPC team will tackle the milestones and objectives presented in **??** . To ensure smooth operation, the M++ team should also contribute some minor assistance. In particular, we define the following tasks, which the M++ team is responsible for:

- Granting access to the code repository. *Done*
- Writing unit and integration tests. The M++ team has student assistants available who can cover this task.
- Attending scheduled progress updates.
- Granting access to the computing project on Horeka.
- Publishing a software release with the deliverables in **??** , and providing a persistent object identifier (e.g. through Zenodo).

Background on M++

M++ is a C++ library for FE methods running on distributed memory systems using MPI. Similar tools are deal.II or DUNE. Over the past 20 years, it has been developed at the Institute for Applied and Numerical Mathematics (IANM) in the research group of Prof. C. Wieners at the Karlsruhe Institute of Technology (KIT) in collaboration with external partners. Recent efforts on the software include a major refactoring, support of new file formats, reduction of technical debt and incorporating modern C++ into the codebase. The project uses CMake, version control, continuous integration and automated benchmarking practices.

The software provides a wide range of modules for modern FE, including discontinuous Galerkin, space-time methods, multiscale techniques, and UQ methods. Each application is designed to integrate into distributed memory systems to address large-scale problems on high-performance computing (HPC) resources.

The scaling of the memory footprint depends on the dimensionality of the problem and its resolution. The software utilizes an MPI parallelization for all finite elements, linear algebra, and UQ methods. The code is designed to be executed on CPU clusters, and we have successfully run it on the HoreKa system with up to 128 nodes. Most of the numerical experiments are conducted via an CI/CD-pipeline to be able to reproduce the results at any time and to benchmark recent developments.

Further Collaborators

- **Christian Wieners (KIT):** Initial developer and copyright holder of M++

- **Daniele Corallo (KIT):** Maintainer of M++, experienced in using the HoreKa system with M++, deep system knowledge
- **Mathias Reichardt (KIT):** Technical support, maintainer of Gitlab runner and CI/CD pipelines on HoreKa system
- **Suryansh Chaturvedi (Uni-HD):** Master student, has previously worked on the interface

Outlook and Future Project Proposal

Quantifying uncertainties in computational models is a key challenge in scientific computing. In many applications, the input data is not known exactly, and the used models themselves are only approximations of real-world systems. Uncertainty quantification (UQ) aims to quantify the impact of these uncertainties on the model output. This information is crucial for decision-making, risk assessment, and reliability analysis. In a followup project to the Ginkgo-M++ interface, we could focus on the development and application of high-performance sampling methods for efficient and scalable uncertainty quantification. These methods include Monte Carlo methods, stochastic gradient descent methods, and particle filters—all already implemented on a multilevel data structure in M++. However, all these UQ methods heavily rely on the efficient solution of large-scale linear systems. The Ginkgo integration ensures that UQ methods can be applied efficiently with minimal overhead and technology stack. Particularly, budgeting techniques as in [3] in combination with GPU-accelerated solvers promise to achieve computational results of high accuracy and low variance.

References

[1] H. Anzt, T. Cojean, G. Flegar, F. Göbel, T. Grützmacher, P. Nayak, T. Ribizel, Y. M. Tsai, and E. S. Quintana-Ortí. Ginkgo: a modern linear operator algebra framework for high performance computing. *ACM Trans. Math. Software*, 48(1):Art. 2, 33, 2022. ISSN 0098-3500. doi: 10.1145/3480935. URL `https://doi.org/10.1145/3480935`.

[2] N. Baumgarten and C. Wieners. The parallel finite element system M++ with integrated multilevel preconditioning and multilevel Monte Carlo methods. *Comput. Math. Appl.*, 81:391–406, 2021. ISSN 0898-1221. doi: 10.1016/j.camwa.2020.03.004. URL `https://doi.org/10.1016/j.camwa.2020.03.004`.

[3] N. Baumgarten, S. Krumscheid, and C. Wieners. A fully parallelized and budgeted multilevel monte carlo method and the application to acoustic waves. *SIAM/ASA Journal on Uncertainty Quantification*, 12(3):901–931, 2024.

[4] T. Bohlen, M. R. Fernandez, J. Ernesti, C. Rheinbay, A. Rieder, and C. Wieners. Visco-acoustic full waveform inversion: from a DG forward solver to a Newton-CG inverse solver, 2021. ISSN 0898-1221. URL `https://doi.org/10.1016/j.camwa.2021.09.001`.

[5] D. Corallo, W. Dörfler, and C. Wieners. Space-time discontinuous galerkin methods for weak solutions of hyperbolic linear symmetric friedrichs systems. *Journal of Scientific Computing*, 94(1):27, 2023.

[6] J. Fröhlich. *A segregated finite element method for cardiac elastodynamics in a fully coupled human heart model*. PhD thesis, Karlsruher Institut für Technologie (KIT), 2022.

[7] G. Olenik, M. Koch, Z. Boutanios, and H. Anzt. Towards a platform-portable linear algebra backend for openfoam. *Meccanica*, pages 1–14, 2024.