# GPU Acceleration of Algebraic Multigrid Preconditioners for Discrete Elliptic Field Problems

Christian Richter[1], Sebastian Schöps[2], and Markus Clemens[1]

[1]Bergische Universität Wuppertal, Chair of Electromagnetic Theory, Wuppertal 42119, Germany
[2]Technische Universität Darmstadt, Graduate School of Computational Engineering and Institut für Theorie Elektromagnetischer Felder, Darmstadt 64285, Germany

**The simulation of coupled electromagnetic/thermal problems with high resolution requires efficient numerical schemes. High-performance computing languages like CUDA help in unlocking the massively parallel capabilities of graphic processor units (GPUs) to accelerate those calculations. This reduces the time needed to solve real-world problems. In this paper, the speedup is discussed, which is obtained using NVIDIA's recently presented Kepler architecture as well as by GPU-accelerated algebraic multigrid preconditioners. In particular, extended memory allows for the solving of larger problems with more degrees of freedom without swapping. We discuss a new host-based multigrid setup for GPU-accelerated iterative solvers.**

***Index Terms*— Algebraic multigrid method (AMG), bioheat, conjugate gradients, CUDA, finite differences (FD), finite elements (FE), graphics processor unit (GPU), multiphysics.**

## I. Introduction

IN LOW-FREQUENCY electromagnetic field simulations and many related coupled problems, such as heat evolution, elliptic boundary value problems of Poisson type are solved on a computational domain $\Omega$

$$-\nabla \cdot (\xi(\vec{r})\nabla u(\vec{r})) = f \tag{1}$$

where $\xi$ is a material property, $f$ is a given field source, and $u$ is the field solution, which is supplied with adequate boundary conditions, e.g., in the simplest case, a homogeneous Dirichlet constraint $u(\vec{r}) = 0$ for $\vec{r} \in \partial\Omega$.

Accurate volume discretization turns the problem above into a very large linear system of equations with a positive definite system matrix

$$\tilde{\mathbf{S}}\mathbf{M}_\xi\tilde{\mathbf{S}}^\top\mathbf{u} = \mathbf{f} \tag{2}$$

factorized into matrices for the divergence $\tilde{\mathbf{S}}$ and gradient operators $\mathbf{G} = -\tilde{\mathbf{S}}^\top$ and the material $\mathbf{M}_\xi$ when using finite differences (FDs), finite integration technique [1] or lowest order Whitney finite elements (FE) [2]. In this paper, we discuss the acceleration of the solution procedure of the resulting linear system by advanced preconditioners, e.g., multigrid on graphic processing units (GPUs) exploiting the new capabilities of NVIDIA's newest programming language CUDA 5. The various discretization methods share many commonalities. Nonetheless, the sparsity patterns of the system matrices are very different and thus each case is discussed separately.

The FD scheme is commonly used for high-frequency-domain problems including electromagnetic field dosimetry using voxel body phantoms. It is also convenient for coupled electromagnetic/thermal simulations where the effects of electromagnetic radiation on a body's thermal field are evaluated [3]. A GPU acceleration was presented in [4]. In this paper, we discuss the usage of advanced preconditioners on GPUs for the thermal part, which is of type (2), where $\mathbf{u}$ is the discrete temperature and $\mathbf{f}$ is the sum of all heat sources.

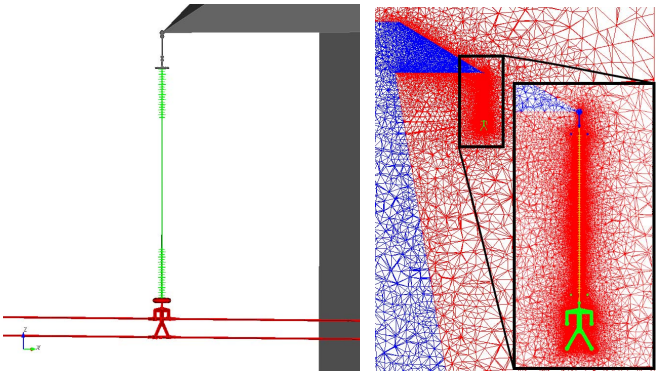

Fig. 1. Model geometry and mesh grid of the high-voltage isolator as simulated in this paper. The model is taken from [6].

FE discretization is in particular popular in static and low-frequency field simulations. We employ the in-house code MEQSICO, which is capable of the calculation of static and quasi-static electromagnetic problems and coupled multiphysical problems [5]. We discuss the electrostatic example shown in Fig. 1, which is described by (2), where $\mathbf{u}$ is the discrete electrical potential and $\mathbf{f}$ is given by discrete electrical charges or a boundary condition.

This paper is organized as follows: Following this introduction (Section I), we discuss the algebraic multigrid method (AMG) in Section II. In Section III, accelerated computations (i.e., the AMG preconditioner) for FD and FE are presented and we propose modifications to existing software libraries to speed up the computations and allow for larger problems. In Section IV, the performance for two examples (FD and FE) is measured and compared with other common solver libraries. Finally, this paper closes with a conclusion.

## II. Algebraic Multigrid Method

In the recent years, AMG [7] has become a common approach for preconditioning large linear systems for Krylov subspace methods as preconditioned conjugate gradients (PCG) [8]. Multigrid employs a grid hierarchy (levels) with

Manuscript received June 29, 2013; revised August 7, 2013; accepted September 18, 2013. Date of current version February 21, 2014. Corresponding author: C. Richter (e-mail: christian.richter@uni-wuppertal.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMAG.2013.2283099

increasing coarseness. In contrast to the geometric multigrid method, AMG constructs the levels without exploiting knowledge of the geometric mesh. Therefore, its application to both FD with structured hexahedral grids and FE with unstructured tetrahedral grids is straightforward.

When AMG is used as a preconditioner, the overall process can be divided into two parts: in the setup phase, wherein the coarser levels are assembled according to the strength of interaction between the degrees of freedom. Afterward, the prolongation and restriction operators are constructed, which connect two adjudicating levels. The setup of the system matrix of the coarser level is executed by performing a triple matrix product, the so-called Galerkin product.

Within the solve phase of PCG, the constructed multigrid preconditioner is used in every iteration step. After smoothing the given linear system, the residual is computed and restricted to the next coarse level. Within this next level, the described function is called again recursively. After returning from the next coarse level, the result is corrected by a calculated error and smoothed again. On the coarsest level, no further recursive calls are performed. The coarsest system is solved by direct or iterative solvers. The details of the so-called V-cycle are given in [9] and [10].

The complex calculations of the setup phase are performed only once for a given system matrix. The multigrid V-cycle is less time intense. It is only a couple of sparse matrix–vector multiplications. It performs very effectively on GPUs but it is executed in every iteration step of PCG.

## III. Accelerating Discrete Elliptic Field Problems With a GPU-Based AMG Preconditioner

In [4], we have discussed the benefits of GPUs acceleration using pointwise Jacobi preconditioners. They have been shown to be effective in FD simulations. They reduce the number of iteration steps by eight times and only negligibly increased the calculation time per iteration step. This is because the sparse-matrix–vector products in the diagonal storage format (DIA), where all nonzero diagonals are stored, are only an additional pointwise vector–vector product. More complex preconditioners will increase the computational costs per step significantly but are known to decrease the total calculation time on CPU systems, see [8]. We will show that also GPUs reduce this effort in practical applications. With CUDA 5.0, a new version of CuSparse was introduced, offering possibilities to use incomplete LU and incomplete Cholesky (IC) preconditioners [11]. Unfortunately, they support only the compressed row sparse (CRS) storage format, which was already shown to be disadvantageous for banded FD matrices in [4].

CUSP is another efficient library with GPU acceleration offering a wide range of preconditioners including IC and approximate inverse (AInv) [12], [13]. In [14], an AMG implementation for GPUs was presented. Its advantage is a high speedup in the solve phase, where AMG is used as a preconditioner. Its disadvantage is the high temporary need of global memory in the setup phase for the Galerkin products. Hence, if the setup is performed on the GPU, this reduces the maximal size of the problem drastically. Therefore, a new mixed approach is proposed in which the setup of the preconditioner is performed on the host. The result is uploaded to the GPU where the solve phase will be performed. In the mean time, this feature has been pushed to the official CUSP library [12]. Finally, on the device, the sparse matrix–vector multiplications are performed with a matrix stored in DIA format if possible, e.g., for FD.

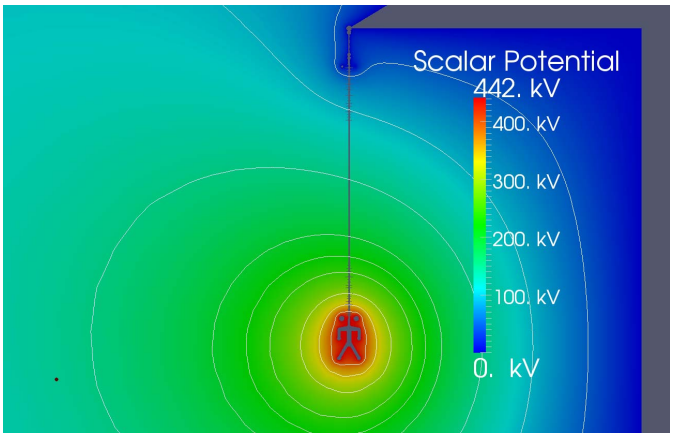


Fig. 2. Electrostatic potential distribution for the model shown in Fig. 1.

Sophisticated preconditioners as, e.g., AMG, are in particular beneficial in FE-based simulations. As the corresponding linear systems typically have a higher complexity (and condition) than those in FD, a higher efficiency gain can be expected. The discretization and assembly of the test example (Fig. 1) are based on the in-house code MEQSICO and the linear systems are passed over to AMG as discussed previously. FE matrices do not necessarily feature banded structure, thus the storage format remains in CRS.

To allow for a realistic discussion of the performance, the problem is solved using GPUs and a host-only system that employs up to all available CPU cores via OpenMP. For further investigation, we compare our solvers with the state of art: we use PETSc's multigrid implementation as preconditioner as well as the abilities of Trilinos ML [15], [16]. For using a varying number of cores, the message passing interface (MPI) is used in this case.

## IV. Numerical Experiments

In this section, we discuss the GPU acceleration featuring our proposed modifications. All calculations were performed on the same machine: the host is equipped with an Intel i7-3820 eight-core CPU with 16 GB RAM and a NVIDIA Tesla K20 GPU accelerator. To ensure data consistency error-correcting code is turned on. Therefore, the effective bandwidth of the Tesla K20 is reduced by 20 %. The GPU codes are compiled using OpenMP on the host, and the current architecture model 3.5 on the device. We use CUDA 5.0, Thrust 1.8.0, CUSP 0.3.0, and PETSc in Version 3.4. PETSc's performance is optimized by following the compiling hints as suggested in [15].

### A. GPU-Accelerated Structured Grid FD-Simulations

The computational example for a temperature thermostatic FD simulation is the head and upper torso of the model Duke [17]. For reasons of comparability the same problem data is used as in [4]. The human model is exposed

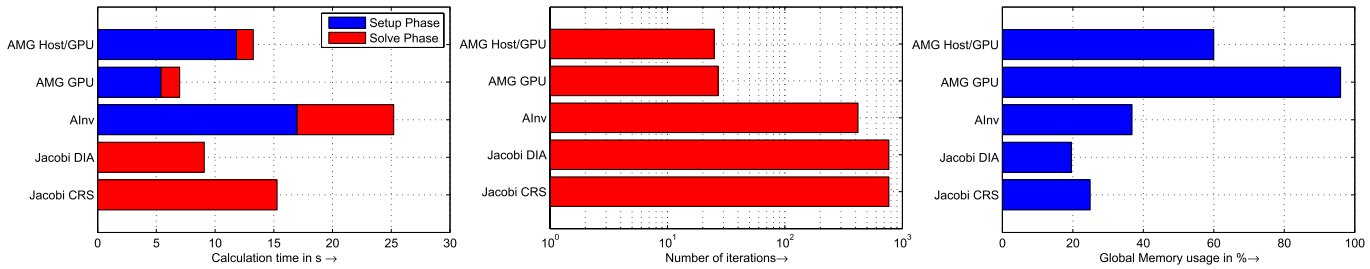


Fig. 3. Time for setup and solve phase, number of iterations, and memory allocated on the GPU.

to a homogeneous electromagnetic wave at 900 MHz with an intensity of 42 V/m. The head is resolved into voxels of 2 $mm^3$. Including the surrounding air this leads to a problem consisting of 6.8 million degrees of freedom. The electromagnetic material parameters are taken from the Italian National Research Council. The thermal parameters are extracted from [18] and have been solved using double precision with a stopping criterion of a relative residual of below $10^{-12}$.

For the FD problem five preconditioners are compared.

1) Jacobi CRS is a Jacobi preconditioner using the commonly used CRS storage format.
2) Jacobi DIA is a Jacobi preconditioner using the DIA storage format, that is well suited to GPUs [4].
3) AInv is an approximate inverse preconditioner with the scaled Brinson-preconditioner-option of CUSP [12].
4) AMG GPU is CUSPs multigrid implementation, involving a four-level V-Cycle with the coarsest problem solved on the host using a direct LU solver.
5) AMG Host/GPU is the modified CUSP Multigrid implementation where the setup phase is performed on the host and the solve phase is performed only on the GPU. The host-to-GPU upload to the device is included in the setup costs.

Fig. 3 shows the calculation times of the various preconditioners in total and divided into setup and solve time. The measurements of the Jacobi preconditioner correspond to previously published results but now using the Tesla K20. It is known that the calculation speed is limited by the available bandwidth. This can be seen when comparing the Jacobi-preconditioned solvers in Fig. 3. By using a more bandwidth-optimal storage format, the calculation time can be reduced significantly [4].

The AInv preconditioner of CUSP does not perform well. The small speedup during the solve phase compared with the DIA Jacobi implementation comes with a long setup phase and results in the worst total computational time. Further variants and optimizations of this approach are not considered in this paper.

On the other hand, the AMG preconditioner provides better results than the Jacobi preconditioner in terms of the computational time. The time is divided between the setup and solve phases 3.38 to 1, which means that for this problem more time is spent on setting up the preconditioner than solving the problem.

In Fig. 3 the memory usage is depicted. The main disadvantage of the AMG GPU implementation becomes obvious: it needs almost three times more storage as AInv and more than a factor four compared with PCG with Jacobi. This is due to the additional matrices that have to be stored in global memory and the more complex storage formats that need to be used for effectively implementing the AMG on the GPU. Furthermore, the temporary allocated memory is increased while calculating the Galerkin product. This rises linearly with the number of columns of the system matrix, [14], i.e., the number of unknowns.

To reduce the amount of storage allocated on the GPU device, the AMG setup is performed on the host as described in Section III. Fig. 3 shows that this increases the setup time and thus decreases the overall performance as the overall time is dominated by the setup. On the other hand, this drawback vanishes if systems with multiple right-hand-sides are solved. For very large systems the setup requires more memory than the GPU can supply (5 GB for the Tesla K20), thus there is no alternative to a adopting a domain decomposition approach or the host setup. The host-based AMG setup reduces the allocated storage of the Tesla K20 by over 30 %–60 % of the overall global memory.

### *B. Using GPU-Accelerated AMG for FEM Simulations*

The FE example is an electrostatic simulation of the high-voltage isolator presented in [6]. The model as shown in Fig. 1 consists of 1.5 million degrees of freedom leading to a algebraic system matrix containing 21 million nonzero entries. It is numerically solved using the following iterative solvers.

1) CUSP AMG: the GPU-accelerated AMG-PCG solver performed on the GPU as well as only on the host using a varying number of CPU cores.
2) Trilinos ML: PETSc's CG solver using Trilinos ML as a preconditioner.
3) PETSc AMG: PETSc's CG solver using its own AMG preconditioner.
4) Jacobi-PCG: the Jacobi preconditioned CG performed on the GPU as well as only on the host.

Fig. 4 shows the time for solving and setup phase when using the GPU-accelerated code compared with the other preconditioners. The comparison considers the number of CPU cores. Especially PETSc that uses MPI for parallelization is accelerated by using multiple cores. The calculation speed of the single cores is limited by the CPUs' bandwidth [15], and it is slowed down when using too many cores due to the communication overhead. The best results for the setup phase on this specific machine were achieved by using four cores. With OpenMP, the CUSP host implementation is less affected by this effect. Nonetheless, also here four cores lead to the best result. However, the best results on the host (CPU) are still only half as fast as the setup performed on the GPU. This is nearly the same result as those previously shown in Fig. 3

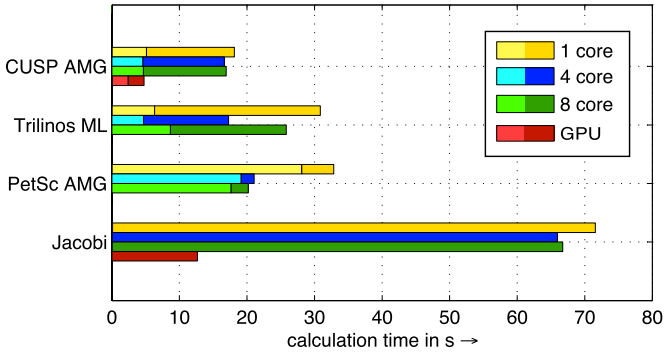


Fig. 4. Computational time for the GPU accelerated code compared with the given solvers for varying number of CPU cores. Light colors: computational time for setup phase. Dark colors: computational time for solve phase.

when discussing the AMG setup of the FD code. The setup of AMG GPU is about twice as fast as AMG Host/GPU where the setup is performed on the host.

In Fig. 4, the computational time of the solve-phase is shown as well. When performing the solve phase of AMG-GPU on the GPU, it is accelerated by a factor of 5.2 compared with the same code executed on the CPU. This factor is still valid when using the mixed approach, as shown for the FD code. On the host, in case of the FE-problem, the solve phase takes much longer than the setup phase. Therefore, more promising accelerations can be achieved for this kind of problems due to the worse condition of the system matrix, e.g., because of the unstructured grid. The relation between setup-times and solve phase-times changes from 1 to 2.62 for the fastest host-based CUSP-AMG-implementation using four cores to 1 to 0.97 when using the GPU.

The overall speedup is determined by the relation of setup to solve phase. For well-conditioned problems like the temperature field of the model Duke discussed in Section I, it is dominated by the setup phase and therefore a lower speedup can be expected. For bad conditioned systems with large material jumps (like in the given FE electrostatic example) or when multiple cycling-phases are performed with an unchanged system-matrix, higher speedups can be expected because of the dominance of the time spent on the solve phase.

## V. Conclusion

Implementations for elliptic problems using GPU-accelerated AMG preconditioners for accelerating large sparse problems have been presented. For the more complex unstructured grid, the GPU-CUSP-AMG-library-implementation is compared with state-of the-art solvers like Trilinos ML and PETSc, and was shown to outperform these.

As numerical problems arising from numerical test problems, structured grid FD, and unstructured grid FEM discretizations have been presented, showing significant speedup to established AMG libraries on CPU multicore systems. Memory limitations of the GPU in the setup phase have been solved by including calculations on the CPU host, which allowed larger problems to be solved. The results indicate that the speedup of the AMG solve phase will be especially beneficial in transient quasi-static simulations.

## Acknowledgment

The work of S. Schöps was supported in part by the Excellence Initiative of the German Federal, in part by the State Governments, and in part by the Graduate School of Computational Engineering with the Technische Universität Darmstadt, Darmstadt, Germany. The authors would like to thank NVIDIA for supporting this research by donating a Tesla K20 to the CUDA Research Center at the Bergische Universität Wuppertal. They would also like to thank C. Schmitt from the Graduate School CE for his support in administrating and configuring software and hardware.

## References

[1] T. Weiland, "A discretization model for the solution of Maxwell's equations for six-component fields," *AEU*, vol. 31, pp. 116–120, Mar. 1977.

[2] P. Monk, *Finite Element Methods for Maxwell's Equations*. Oxford, U.K.: Oxford Univ. Press, 2003.

[3] A. Bitz, Y. Zhou, A. E. Ouardi, and J. Streckert, "Occupational exposure at mobile communication base station antenna sites," *Frequenz*, vol. 63, nos. 7–8, pp. 123–128, Aug. 2009.

[4] C. Richter, S. Schöps, and M. Clemens, "GPU acceleration of finite differences in coupled electromagnetic/thermal simulations," *IEEE Trans. Magn.*, vol. 49, no. 5, pp. 1649–1652, May 2013.

[5] T. Steinmetz, M. Helias, G. Wimmer, L. O. Fichte, and M. Clemens, "Electro-quasistatic field simulations based on a discrete electromagnetism formulation," *IEEE Trans. Magn.*, vol. 42, no. 4, pp. 755–758, Apr. 2006.

[6] H. Ye, M. Clemens, and J. Seifert, "Electro-quasistatic field simulation for the layout optimization of outdoor insulation using microvaristor material," *IEEE Trans. Magn.*, vol. 49, no. 5, pp. 1709–1712, May 2013.

[7] K. Stüben, "Algebraic multigrid (AMG): An introduction with applications," GMD, New York, NY, USA, Rep. 53, Mar. 1999.

[8] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Boston, MA, USA: SIAM, 2003.

[9] Y. Shapira, *Matrix-Based Multigrid: Theory and Applications* (Numerical Methods and Algorithms). New York, NY, USA: Springer-Verlag, 2008.

[10] U. Trottenberg and A. Schuller, *Multigrid*. Orlando, FL, USA: Academic, 2001.

[11] J. Meijerink and H. A. van der Vorst, "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix," *Math. Comput.*, vol. 31, no. 137, pp. 148–162, Jan. 1977.

[12] N. Bell and M. Garland. (2012). *Cusp: Generic Parallel Algorithms for Sparse Matrix and Graph Computations* [Online]. Available: http://cusp-library.googlecode.com

[13] M. M. Dehnavi, D. M. Fernandez, J.-L. Gaudiot, and D. D. Giannacopoulos, "Parallel sparse approximate inverse preconditioning on graphic processing units," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1852–1862, Sep. 2013.

[14] N. Bell, S. Dalton, and L. N. Olson, "Exposing fine-grained parallelism in algebraic multigrid methods," *SIAM J. Sci. Comput.*, vol. 34, no. 4, pp. C123–C152, Apr. 2012.

[15] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, *et al.*, "PETSc users manual," Argonne Nat. Lab., Lemont, IL, USA, Tech. Rep. ANL-95/11, 2013.

[16] M. Gee, C. Siefert, J. Hu, R. Tuminaro, and M. Sala, "ML 5.0 smoothed aggregation user's guide," Sandia Nat. Lab., Albuquerque, NM, USA, Tech. Rep. SAND2006-2649, 2006.

[17] A. Christ, W. Kainz, E. G. Hahn, K. Honegger, M. Zefferer, E. Neufeld, *et al.*, "The virtual family—Development of surface-based anatomical models of two adults and two children for dosimetric simulations," *Phys. Med. Biol.*, vol. 55, no. 2, pp. N23–N38, Jan. 2010.

[18] P. Bernardi, M. Cavagnaro, S. Pisa, and E. Piuzzi, "Specific absorption rate and temperature elevation in a subject exposed in the far-field of radio-frequency sources operating in the 10-900-MHz range," *IEEE Trans. Biomed. Eng.*, vol. 50, no. 3, pp. 295–304, Mar. 2003.