Software Impacts 6 (2020) 100041

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

BootCMatchG: An adaptive Algebraic MultiGrid linear solver for GPUs☆

Massimo Bernaschi [a], Pasqua D'Ambra [b,*], Dario Pasquini [a,c]

[a] Institute for Applied Computing (IAC) - CNR, Via dei Taurini 19, 00185 - Rome, Italy
[b] Institute for Applied Computing (IAC) - CNR, Naples branch, Via P. Castellino, 111, 80131, Naples, Italy
[c] Department of Computer Science, "Sapienza" University, Via Salaria, 113, 00198, Rome, Italy

ARTICLE INFO

MSC:
65F10
65N55
65-04

Keywords:
Adaptive AMG
GPU

ABSTRACT

Sparse solvers are one of the building blocks of any technology for reliable and high-performance scientific and engineering computing. In this paper we present a software package which implements an efficient multigrid sparse solver running on Graphics Processing Units. The package is a branch of a wider initiative of software development for sparse Linear Algebra computations on emergent HPC architectures involving a large research group working in many application projects over the last ten years.

Code metadata

Current code version	v0.1
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2020-54
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/4631894/tree/v1
Legal Code License	MIT
Code versioning system used	git
Software code languages, tools, and services used	C++ & CUDA C. & CUB & modified nsparse
Compilation requirements, operating environments & dependencies	NVCCV ≥ 9.1.85, G++ ≥ 5.5.0, A NVIDIA GPU with compute capability ≥ 7.0.
If available Link to developer documentation/manual	https://github.com/bootcmatch/BootCMatchG
Support email for questions	pasquini@di.uniroma1.it

1. Introduction

BootCMatchG is the software implementation of an adaptive Algebraic MultiGrid (α-AMG) method, specifically tuned for exploiting the fine-grained parallelism of Graphics Processing Unit (GPU) devices in scientific and engineering computing applications.

GPU computing is, by now, a usual practice in general-purpose computations, since GPU accelerators are installed on devices ranging from laptops to supercomputers. Moreover, high-level programming environments, such as NVIDIA CUDA or OpenCL, enable developers to write codes for GPUs in a reasonable simple form. As a matter of fact, many of the most powerful supercomputers are equipped with heterogeneous nodes using GPU accelerators to achieve performance in the order of PetaFLOPS (10^{15}) [1]. However, efficient use of GPUs generally requires a re-factoring of algorithms, data structures and software development paradigms for taking full advantage from their computing power.

With *BootCMatchG* we make available a recently proposed adaptive AMG method for preconditioning and solving algebraic linear systems $Ax = b$, where $A \in \mathbb{R}^{n \times n}$ is a symmetric positive definite (s.p.d.), large and sparse matrix [2,3]. All the computational kernels for setup and application of the AMG method were designed and tuned for accessing GPU global memory according to best practices of CUDA programming and for using the available computing resources in an effective way.

☆ This work has been partially supported by the EC under the Horizon 2020 Project *Energy Oriented Center of Excellence (EoCoE II): Toward Exascale for Energy*, Project ID: 824158.

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail address: pasqua.dambra@cnr.it (P. D'Ambra).

https://doi.org/10.1016/j.simpa.2020.100041
Received 5 November 2020; Received in revised form 7 November 2020; Accepted 9 November 2020

AMG methods are widely used as preconditioners for Krylov-type iterative methods in solving general sparse linear systems, when no information are available on system's origin [4,5]. Many variants of AMG methods [6–9], which differ from each other in the setup of the method, and related software [10–12] have been proposed. However, the above methods lose their robustness and efficiency when applied to classes of s.p.d. matrices more general than that stemming from 2nd order scalar elliptic partial differential equations (PDEs). To overcome those limits, adaptive (α-)AMG methods have been proposed [2,13–15], which exploit information obtained by testing a current method to modify it, so as to improve convergence. To the best of our knowledge, only the α–AMG method in [2] is currently available as public-domain software package (see [3]). Here we present the GPU version of the method which has been described in [16] and successfully compared with the NVIDIA AmgX library, which implements just standard AMG methods [17].

2. Software functionalities and use

BootCMatchG has a modular structure including different components for setup of the AMG method and for its application as preconditioner in an optimized version of the Conjugate Gradient method (see [16] for details). Many algorithmic parameters are available for AMG setup, so that the package assures flexibility with respect to problem features and convergence and accuracy requirements. Default values are available for a large part of the parameters, however they could be changed by simple interfaces depending on the user's level of expertise and needs. The AMG method implemented in *BootCMatchG* is based on a bootstrap strategy which builds different AMG hierarchies and composes them in additive or multiplicative way in order to obtain the desired convergence rate [2,3]; a maximum number of bootstrap steps is set, so that the procedure can also built a single-component AMG, as in usual multigrid methods. Each AMG hierarchy is built by a pairwise aggregation scheme and multiple steps of the basic aggregation can be applied to obtain larger aggregates than pairs and then smaller size coarse matrices. The aggregation scheme can be either applied as plain aggregation, when piecewise constant inter-grid transfer operators are built, or according to a smoothed approach, where more regular transfer operators are built by employing one iteration of a Jacobi smoother to the original un-smoothed operator. Number of AMG hierarchy levels can be also set by the user, while maximum size of the coarsest matrix is set by default to the value $40n^{1/3}$, where n is the system's matrix dimension. During the setup of the bootstrap AMG, in order to estimate the convergence rate of the composite AMG, the method is tested on the homogeneous system associated to the original one, so we also consider as setup parameter the way the single AMG hierarchy is applied, i.e. the AMG cycle. At this time, the smoothers available in the package are weighted versions of the highly parallel Jacobi smoother, specifically tuned for GPU exploitation, and also applied as coarsest solvers. The user can set the number of pre- and post-smoothing steps at each intermediate level of a cycle as well as the number of iterations of the weighted Jacobi method to be applied at the coarsest level. In Table 1 we summarize main parameters to be set in a configuration file for bootstrap AMG setup.

Further parameters related to the final linear solver application includes the maximum number of iterations and a tolerance specifying the user's accuracy requirement. The software package has been tested with respect to a number of cases arising from scalar and vector anisotropic PDEs. An extended analysis of parallel performance is reported in [16].

Table 1
Main setup parameters.

Parameter	Values
Bootstrap parameters	
solver_type (integer)	0/1/2 (mult/sym-mult/add)
max_hrc (integer)	arbitrary
ρ (double)	$0 < \text{value} < 1$
Hierarchy parameters	
aggr_sweeps (integer)	arbitrary
aggr_type (integer)	0/1 (un-smoothed/smoothed)
max_levels (integer)	arbitrary
cycle_type (integer)	0/1/2/3 (V/H/W/K)
coarsest_solver_type	0/4 (standard weighted Jacobi/ℓ_1-Jacobi)
relax_type	0/4 (standard weighted Jacobi/ℓ_1-Jacobi)
prerelax_sweeps (integer)	arbitrary
postrelax_sweeps (integer)	arbitrary
coarserelax_sweeps (integer)	arbitrary

3. Impact

Solution of sparse linear systems is an ubiquitous task in Computational and Data Science. As a consequence, efficient sparse solvers are essential tools which play a major role in the reduction of the time-to-solution [18]. The CPU version of our Adaptive Algebraic Multigrid solver [3] has been already included in some projects aimed at designing and developing software technology for HPC. Moreover, extensions to that software package are in progress in a project whose goal is to provide efficient methods for spectral analysis of complex networks [19]. Since GPUs are, by now, widely used to accelerate many computing applications, we argue that it can be useful to make available BootCMatchG, the CUDA based implementation of the same solver. The efficiency of this variant has been fully demonstrated in [16] and a number of activities are in progress for extending this code and integrating it in a more general software framework for sparse matrix computations [20–28] within the context of EoCoE-II, a EU-funded project focused on the transition toward exascale of simulation codes for renewable and low-carbon energy applications. In particular, we are working on a hybrid MPI-CUDA version of BootCMatchG for large clusters composed of nodes equipped with one or multiple GPUs and connected with state-of-art network technologies.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Top 500 lists, 2020, https://www.top500.org/lists/.
[2] P. D'Ambra, P.S. Vassilevski, Adaptive AMG with coarsening based on compatible weighted matching, Comput. Vis. Sci. 16 (2) (2013) 59–76, http://dx.doi.org/10.1007/s00791-014-0224-9.
[3] P. D'Ambra, S. Filippone, P.S. Vassilevski, BootCMatch: a software package for bootstrap AMG based on graph weighted matching, ACM Trans. Math. Software 44 (4) (2018) 39:1–39:25, http://dx.doi.org/10.1145/3190647.
[4] K. Stüben, Algebraic multigrid (AMG): an introduction with applications, in: U. Trottenberg, C. Oosterlee, A. Schüller (Eds.), Multigrid, in: Frontiers in Applied Mathematics, Academic Press, 2001, pp. Appendix A, 413–532.
[5] P.S. Vassilevski, Multilevel Block Factorization Preconditioners: Matrix-Based Analysis and Algorithms for Solving Finite Element Equations, Springer, New York, USA, 2008.
[6] J.W. Ruge, K. Stüben, Algebraic multigrid (AMG), in: S.F. McCormick (Ed.), Multigrid Methods, in: Frontiers in Applied Mathematics, SIAM, Philadelphia, USA, 1987, pp. 73–130.
[7] P. Vaněk, J. Mandel, M. Brezina, Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems, Computing 56 (3) (1996) 179–196.
[8] V.E. Henson, U.M. Yang, BoomerAMG:a parallel algebraic multigrid solver and preconditioner, Appl. Numer. Math. 41 (2000) 155–177, http://dx.doi.org/10.1016/S0168-9274(01)00115-5.
[9] Y. Notay, An aggregation-based algebraic multigrid method, Electron. Trans. Numer. Anal. 37 (2010) 123–146.
[10] R.D. Falgout, J.E. Jones, U.M. Yang, The design and implementation of hypre, a library of parallel high-performance preconditioners, in: A.M. Bruaset, A. Tveito (Eds.), Numerical Solutions of Partial Differential Equations on Parallel Computers, in: Lecture Notes in Computational Science and Engineering, vol. 15, Springer-Verlag, Berlin, Germany, 2006, pp. 267–294.

[11] M.W. Gee, C.M. Siefert, J.J. Hu, R.S. Tuminaro, M.G. Sala, ML 5.0 Smoothed Aggregation 's Guide, Tech. Rep. SAND2006-2649, Sandia National Laboratories, 2006.
[12] Y. Notay, User's Guide to AGMG, Tech. Rep., Université Libre de Bruxelles, 2018.
[13] M. Brezina, R.D. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, J. Ruge, Adaptive smoothed aggregation αSA multigrid, SIAM Rev. 47 (2005) 317–346.
[14] M. Brezina, R.D. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, J. Ruge, Adaptive algebraic multigrid, SIAM J. Sci. Comput. 27 (2006) 1261–1286, http://dx.doi.org/10.1137/040614402.
[15] A. Brandt, J. Brannick, K. Kahl, I. Livshitz, Bootstrap AMG, SIAM J. Sci. Comput. 33 (2011) 612–632.
[16] M. Bernaschi, P. D'Ambra, D. Pasquini, AMG based on compatible weighted matching for GPUs, Parallel Comput. 92 (102599) (2020) 1–13, http://dx.doi.org/10.1016/j.parco.2019.102599.
[17] M. Naumov, M. Arsaev, P. Castonguay, J. Cohen, J. Demouth, J. Eaton, S. Layton, N. Markovskiy, I. Reguly, N. Sakharnykh, V. Sellappan, R. Strzodka, AmgX: a library for GPU accelerated algebraic multigrid and preconditioned iterative methods, SIAM J. Sci. Comput. 37 (2015) S602–S626, http://dx.doi.org/10.1137/140980260.
[18] H. Anzt, E. Boman, R. Falgout, P. Ghysels, M. Heroux, X. Li, L. Curfman McInnes, R. Tran Mills, S. Rajamanickam, K. Rupp, B. Smith, I. Yamazaki, U.M. Yang, Preparing sparse solvers for exascale computing, Philos. Trans. R. Soc. A 378 (2166) (2020) 1–17, http://dx.doi.org/10.1098/rsta.2019.0053.
[19] P. D'Ambra, L. Cutillo, P.S. Vassilevski, Bootstrap AMG for spectral clustering, Comput. Math. Methods 1 (e1020) (2019) 1–17, http://dx.doi.org/10.1002/cmm4.1020.
[20] A. Buttari, P. D'Ambra, D. di Serafino, S. Filippone, 2LEV-D2P4: a package of high-performance preconditioners for scientific and engineering applications, Appl. Algebra Engrg. Comm. Comput. 18 (3) (2007) 223–239, http://dx.doi.org/10.1007/s00200-007-0035-z.
[21] P. D'Ambra, D. di Serafino, S. Filippone, MLD2P4: a package of parallel algebraic multilevel domain decomposition preconditioners in Fortran 95, ACM Trans. Math. Softw. 37 (3) (2010) Art. 30, 23. http://dx.doi.org/10.1145/1824801.1824808.
[22] A. Aprovitola, P. D'Ambra, F. Denaro, D. di Serafino, S. Filippone, Scalable algebraic multilevel preconditioners with application to CFD, in: D. Tromeur-Dervout, B. Gunther, D.R. Emerson, J. Erhel (Eds.), Parallel Computational Fluid Dynamics 2008: Parallel Numerical Methods, Software Development and Applications, in: Lecture Notes in Computational Science and Engineering, vol. 74, Springer, Berlin, Heidelberg, D, 2011, pp. 15–27, Invited paper.
[23] P. D'Ambra, D. Di Serafino, S. Filippone, Performance analysis of parallel Schwarz preconditioners in the LES of turbulent channel flows, Comput. Math. Appl. 65 (3) (2013) 352–361.
[24] A. Aprovitola, P. D'Ambra, F.M. Denaro, D. di Serafino, S. Filippone, SParC-LES: Enabling large eddy simulations with parallel sparse matrix computation tools, Comput. Math. Appl. 70 (11) (2015) 2688–2700, http://dx.doi.org/10.1016/j.camwa.2015.06.028.
[25] P. D'Ambra, S. Filippone, A parallel generalized relaxation method for high-performance image segmentation on GPUs, J. Comput. Appl. Math. 293 (C) (2016) 35–44, http://dx.doi.org/10.1016/j.cam.2015.04.035.
[26] A. Abdullahi Hassan, V. Cardellini, P. D'Ambra, D. di Serafino, S. Filippone, Efficient algebraic multigrid preconditioners on clusters of GPUs, Parallel Process. Lett. 29 (1) (2019) 1950001–1–1950001–15. http://dx.doi.org/10.1142/S0129626419500014.
[27] P. D'Ambra, P.S. Vassilevski, Improving solve time of aggregation-based adaptive AMG, Numer. Linear Algebra Appl. 26 (E2269) (2019) 1–14, http://dx.doi.org/10.1002/nla.2269.
[28] P. D'Ambra, F. Durastante, S. Filippone, AMG preconditioners for linear solvers towards extreme scale, 2020, arXiv:2006.16147v2.