14th World Congress on Computational Mechanics (WCCM)
ECCOMAS Congress 2020)
19–24 July 2020, Paris, France
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

A GENERAL-PURPOSE AMG LINEAR SOLVER FOR HIGH PERFORMANCE COMPUTING

Giovanni Isotton[1], Matteo Frigo[1], Nicolò Spiezia[1], Seid Koric[4], Qiyue Lu[5] and Carlo Janna[1]

[1] M3E S.r.l.,
Via Giambellino 7, Padova, Italy
g.isotton@m3eweb.it, m.frigo@m3eweb.it, n.spiezia@m3eweb.it, c.janna@m3eweb.it
https://www.m3eweb.it

[4] NCSA and MechSE, University of Illinois at Urbana-Champaign
Urbana IL, USA
koric@illinois.edu, https://mechanical.illinois.edu/directory/profile/koric

[5] NCSA, University of Illinois at Urbana-Champaign
Urbana IL, USA
qiyuelu1@illinois.edu, https://experts.illinois.edu/en/persons/qiyue-lu

Key words: Algebraic multigrid, preconditioning, numerical simulations, HPC

Abstract. The numerical simulation of modern engineering problems via finite elements requires the solution of sparse linear systems of millions or even billions of unknowns. The algebraic multigrid (AMG) methods are the most common choice as linear solvers because of their fast convergence even for large-size problems. In this communication, we propose Chronos, a massively parallel implementation of a novel AMG framework, specifically designed to address complex problems by adapting its components, from the smoother, to the coarse grid correction and prolongation to the problem at hand. This work demonstrates not only the numerical performance of the proposed library, but also its robustness and adaptability to very challenging matrices, arising from different fields of application.

1 INTRODUCTION

Numerical simulation is a very common tool in a large number of applications, ranging from structural mechanics, to underground processes, computational fluid-dynamics, electromagnetism and many others. All these kinds of simulations usually require a very high resolution with computational domains easily consisting of several millions or even billions of unknowns [16]. When solving such problems with implicit numerical methods, the associated matrix is very large and sparse. A single solution suffices for linear problems. For nonlinear problems, however, within each increment, the system of non-linear equations is often linearized and solved with a Newton-Raphson iteration scheme, which requires a number of linear solver solutions. In any case, the use of high performance computing is often necessary, and the development of efficient and scalable sparse linear solvers is an important research topic.

Algebraic multigrid (AMG) methods are often the best choice as preconditioners for the iterative solu-

tion of large size 3D problems arising from the discretization of PDEs [14, 15, 17]. The main advantage of AMG methods is represented by their theoretical optimality, as, at least in the ideal case, the number of iterations to converge does not increase with the problem size. AMG optimality is obtained through the complementary action of the smoother, responsible to dump the error components associated to high frequencies, and the coarse grid correction that takes care of the lower part of the spectrum. Unfortunately, in ill-conditioned real-world problems the standard AMG method is far to be optimal with its performance strictly related to a proper selection of the set-up parameters [7, 13].

In this work, we propose Chronos a massively parallel implementation of a novel AMG framework [12, 6] which is able to adapt all of its components to the problem at hand, from the smoother set-up, to the coarse grid hierarchy and prolongation definition. This is achieved by guessing and iteratively improving in a bootstrap fashion the near-null space of the system, which allows for both testing the smoother and the prolongation operator as well as for inferring the connection strengths between system unknowns. In this communication, we restrict our attention to Symmetric Positive Definite (SPD) matrices, and the Chronos performance is assessed through the solution of several engineering problems including structural mechanics, CFD and underground applications, on modern super computers. It is shown that the proposed approach brings together both known and novel algorithms with the aim of improving AMG performance in ill-conditioned systems requiring no other information but the system matrix.

The remainder of the paper is organized as follows. In the next section, we briefly recall the basic concepts of classical AMG describing both its set-up and application to a vector. In the third section, those specif features making Chronos an effective general purpose solver are outlined and assessed in the fourth section on a wide set of test benchmarks. We finally close the paper with some concluding remarks and ideas for further investigations.

2 CLASSICAL ALGEBRAIC MULTIGRID

The effectiveness of any AMG method is based on a good interplay between the following components:

- Smoother, which is a simple preconditioner applied to damp the high frequency error modes;
- Coarsening, where coarse level variables are selectd as principal unknowns for the next level;
- Prolongation, which is the transfer operator between coarse and fine variables.

As already mentioned, the present work is focused on the classical AMG setting, and below we will briefly recall the basic concepts behind this method, referring the interested reader to more detailed and rigorous descriptions in the works [14, 15, 17]. For clearness, the explanation is restricted to a two levels only scheme, as the multilevel version can be easily obtained by recursion.

The first component that is typically set-up in AMG is the smoother, a simpler preconditioner used in a stationary iterative method which is responsible for eliminating the error modes associated with large eigenvalues of A, sometimes also referred to as the high frequency errors. The smoother is generally a rough approximation of $A^{-1} \simeq M^{-1}$ with its operator represented by the equation below:

$$S = I - \omega M^{-1} A, \tag{1}$$

with I the identity matrix and ω a relaxation factor necessary whenever $\rho(M^{-1}A) > 2$, see e.g. [6] for an explanation. Typilly, the smoother is given by a simple pointwise relaxation method such as (block)

Jacobi or Gauss-Seidel, with the second one often preferred even though its efficient parallel implementation is not straightforward. In Chronos, we use the adaptive Factorized Sparse Approximate Inverse (aFSAI) [9], beacuse of its almost perfect strong scalability and its proven robustness in real engineering problems [1, 8].

The second component of AMG is the so-called Coarse-Grid Correction (CGC), that is the A-orthogonal projection operation that should take care of the low-frequency error modes. To build CGC in classical AMG, the unknowns of a given level are divided into a Fine/Coarse (F/C) partition, with coarse variables those becoming the next level unknowns. Coarse variables choice is crucial in the AMG construction, as it determines both the rate at which the problem size is reduced and the convergence of the method. A slow rate of problem size reduction may lead to a too expensive preconditioner, while a too fast rate may result in a large number of iterations.

In this communication, we rely on the concept of Strength of Connection (SoC), that is we associate to each edge of the adjacency graph of A a measure of its relative importance. Using SoC, we rank the graph connections and filter out the smallest ones. A maximum independent set (MIS) is finally constructed on the filtered graph to determine coarse variables.

Algorithm 2.1 AMG Set-up

procedure AMG_SETUP(A_k)
  Define Ω_k as the set of the n_k vertices of the adjacency graph of A_k;
  if n_k is small enough to allow for a direct factorization **then**
    Compute $A_k = L_k L_k^T$;
  else
    Compute M_k such that $M_k^{-1} \simeq A_k^{-1}$;
    Define the smoother as $S_k = \left(I_k - \omega_k M_k^{-1} A_k\right)$;
    Partition Ω_k into the disjoint sets $\mathcal{C}_k$ and $\mathcal{F}_k$ via coarsening;
    Compute the prolongation matrix P_k from $\mathcal{C}_k$ to Ω_k;
    Compute the new coarse level matrix $A_{k+1} = P_k^T A_k P_k$;
    Call AMG_SetUp(A_{k+1});
  end if
end procedure

Only for the sake of explanation, the system matrix is ordered according to this (C/F) partitioning, by numbering first fine variables and second coarse ones:

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{fc}^T & A_{cc} \end{bmatrix} \tag{2}$$

where A_{ff} and A_{cc} are $n_f \times n_f$ and $n_c \times n_c$ matrices, respectively. Using this F/C ordering (2), the prolongation operator P takes the following form:

$$P = \begin{bmatrix} W \\ I \end{bmatrix}, \tag{3}$$

where W is a $n_f \times n_c$ matrix containing the weights for coarse-to-fine variable interpolation. As the system matrix is SPD, we assume a Galerkin approach in defining the restriction operator R as the

transpose of P, with the coarse level matrix A_c simply given by the triple matrix product:

$$A_c = P^T A P \tag{4}$$

In practice, fast convergence and rapid coarsening, i.e. high F/C ratios, are always desired, and the construction of effective prolongation operators is of paramount importance to conciliate these conflicting requirements.

Having defined all the above components, the set-up phase of the two-level multigrid method is completed and the iteration matrix is given by:

$$(S)^{\nu_2} \left(I - P {A_c}^{-1} P^T A\right) (S)^{\nu_1} \tag{5}$$

with ν_1 and ν_2 representing the number smoothing steps performed before and after the coarse-grid correction, respectively.

Algorithms 2.1 and 2.2 briefly report the general AMG set-up phase and application in a V-cycle, respectively, in a multilevel framework, where it is conventionally assumed that $A_0 = A$, $\mathbf{y}_0 = \mathbf{y}$ and $\mathbf{z}_0 = \mathbf{z}$. Some details on the adopted computational kernels sketched in 2.1 will be discussed in the next sections.

Algorithm 2.2 AMG application in a V-cycle

procedure AMG_APPLY(A_k, $\mathbf{y}_k$, $\mathbf{z}_k$)
  if k is the last level **then**
    Solve $A_k \mathbf{z}_k = \mathbf{y}_k$ using L_k, the exact Cholesky factor of A_k;
  else
    Compute $\mathbf{s}_k$ by applying ν_1 smoothing steps to $A_k \mathbf{s}_k = \mathbf{y}_k$ with $\mathbf{s}_0 = \mathbf{0}$;
    Compute the residual $\mathbf{r}_k = \mathbf{y}_k - A_k \mathbf{s}_k$;
    Restrict the residual to the coarse grid $\mathbf{r}_{k+1} = P_k^T \mathbf{r}_k$;
    Call AMG_Apply$(A_{k+1}, \mathbf{r}_{k+1}, \mathbf{d}_{k+1})$;
    Prolongate the correction to the fine grid $\mathbf{d}_k = P_k \mathbf{d}_{k+1}$;
    Update $\mathbf{s}_k \leftarrow \mathbf{s}_k + \mathbf{d}_k$;
    Compute $\mathbf{z}_k$ by applying ν_2 smoothing steps to $A_k \mathbf{z}_k = \mathbf{y}_k$ with $\mathbf{z}_0 = \mathbf{s}_k$;
  end if
end procedure

3 SPECIAL FEATURES AVAILABLE IN CHRONOS TO INCREASE PERFORMANCE

AMG preconditioning is very popular and is available from several open source and commercial packages. Its parallelization does not present particular difficulties as mainly relies on standard sparse linear algebra operations such as sparse matrix by matrix and matrix by vector products or other basic tasks which are nowadays readily handled by the highly optimized Basic Linear Algebra Operations (BLAS) routines [5]. The only two tasks that present some difficulties from the parallelization viewpoint are the smoother set-up and application [2] and the coarsening stage [4, 3], which have been deeply investigated by several authors in recent years.

We choose aFSAI as a smoother which has proven effective in several applications. By distinction to Gauss-Seidel smoother, aFSAI application is perfectly parallel also in the application as, giving an

explicit approximation of the system inverse, it can be applied simply by a matrix-vector product. The price to pay for the use of aFSAI is a not always negligible set-up cost that is usually compensated by a faster covergence, especially in ill-conditioned problems [12, 6], where standard smoother fail in dumping high frequencies.

Another important ingredient included in Chronos is the availability of different prolongation approaches, allowing its use in different context. Chronos provides the classical interpolation formula, which is usually the best compromise between complexity and effectiveness for standard Poisson problems, as well as extended plus i interpolation for more challenging diffusion-like problems. Traditionally, these prolongation methods are not completely satisfactory for 3D elasticity, where aggregation-based AMG is usually preferred. To overcome this issue, Chronos also makes available the so-called BAMG prolongation whose coefficients are computed by fitting a set of test vectors through a least square minimization process. The test vectors must represent, at least approximately, the near null space of the operator and, in the case of elasticity, they can be guessed using the Rigid Body Modes (RBM) of the free body [6].

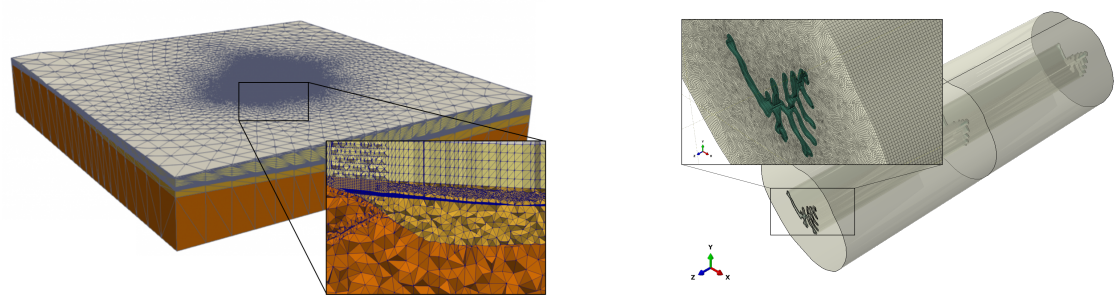
4 NUMERICAL RESULTS

In this section, we assess Chronos robustness and efficiency on a set of benchmarks from various application fields. We test the solver on the Marconi100 supercomputer which is hosted at Cineca, the Italian center for high performance computing. Marconi100, classified within the first ten positions of the TOP500 ranking at the time of writing, is composed by 980 nodes based on the IBM Power9 architecture each equipped with two 16-cores IBM POWER9 AC922 processors at 3.1 GHz. The matrices chosen for the tests are shown in Table 1 along with the number of rows, the number of non-zeroes and the application fields they arise from.

Table 1: Matrices used in the test examples. The table provides the matrix name along with the number of rows, the number of non-zeroes and the application field each matrix arises from

Matrix name	# of rows	# of non-zeroes	Application Field
FINGER4M	4,718,592	23,591,424	3D flow
GUENDA11M	11,452,398	512,484,300	3D geomechanics
AGG14M	14,106,408	633,142,730	3D mesoscale
M20	20,056,050	1,634,926,088	3D mechanical
TRIPOD24M	24,186,993	1,111,751,217	3D mechanical
C4ZZ44M	44,798,517	747,633,815	3D biomechanics
GEO61M	61,813,395	4,966,380,225	3D geomechanics
POI65M	65,939,264	460,595,552	3D flow
PFLOW73M	73,623,733	2,201,828,891	3D flow
C4ZZ134M	134,395,551	10,806,265,323	3D biomechanics

The matrices listed in Table 1 have been generated from real-world problems, arising from different domains such as mechanics, geomechanics, biomechanics, fluid dynamics, porous flow and so on. For example Figure 1a shows the mesh corresponding to the matrix GUENDA11M, while Figure 1b shows the mesh corresponding to matrix C4ZZ44M. The former is a numerical simulation commonly used to asses the geomechanical effects related to subsurface resource exploitation, such as water, oil and gas



(a) GUENDA11M (b) C4ZZ44M

Figure 1: Examples of real-world problems that give rise to two of the matrices presented in this section.

withdraw/injection. The latter represents the complex conformation of the urethral duct, with particular regard to the bulbar region. The relevance of computational modeling of this anatomical region pertains to the investigation of interaction phenomena between artificial sphincter devices and biological tissues [10, 11]. For both these examples, the creation of accurate computational grid is of paramount importance.

For each matrix, we allocate a different number of nodes which is proportional to its number of nonzeroes and we use different preconditioning techniques precisely to highlight how an appropriate choice may greatly reduce the solution time. We use PCG as iterative solver and we consider convergence achieved when the norm of the residual relative to the right-hand-side falls below a tolerance of 10.0^{-8}. Moreover, to ensure that the iterative solution is accurate enough, we validate the final solution against that obtained through a direct solver.

As can be noticed from Table 2, the time reduction may be significant. For some mechanical problems, we also report the performance obtained by using only aFSAI preconditioning instead of AMG. In those cases, it can be noted that the number of iterations is greatly reduced with respect to pure aFSAI at the price of a more expensive preconditioner. Nevertheless, BAMG interpolation always outperforms aFSAI with the exception of GEO61M, which proves a relatively easy problem as also aFSAI needs just few iterations to converge.

In Figure 2, we report the set-up and solve times divided by the number of unknowns assigned to each Marconi100 node and normalized over the average. This quantity is defined to compare our benchmark problems having different sizes and requiring different amount of resources. The purpose of this graph is to show that Chronos, with a proper set-up, is able to solve each problem in a time that depends only on its size and on the allocated computational resources. In the ideal case, we would have obtained an height equal to 1.0 for each column. If we exclude the matrix PFLOW73M which is extremely ill-conditioned, we can observe that the normalized times are very close to 1.0 as is desired.

We finally close this section with strong and weak scalability tests. Figure 3 shows how the total solution time for the matrix C4ZZ134M varies with the allocated resources, from 25 to 200 nodes. The dashed line represents the ideal time, that is the time that would have been obtained with a perfect scaling. It can be observed that Chronos result is very close to the ideal profile. For the weak scalability test, we use a 7 point discretization of the Poisson problem assigning about 7,040,000 unknowns to each node and

Table 2: Performance of Chronos on the test matrices. For each matrix it is provided the number of Marconi100 nodes allocated, the type of preconditioning, the number of iterations, the grid and operator complexities, the set-up time T_p, the solve time T_s and the total time $T_t = T_p + T_s$ in seconds.

Matrix name	# nodes	prec	# iters	gr. comp	op. comp	T_p [s]	T_s [s]	T_t [s]
FINGER4M	1	EXTI	17	1.452	2.555	2.2	0.4	2.6
		CLAS	38	1.467	1.870	2.1	0.9	3.0
		HYBC	20	1.464	2.051	2.2	0.5	2.6
GUENDA11M	2	aFSAI	3831	-	-	58.3	197.8	256.2
		BAMG	726	1.040	1.118	33.1	75.7	109.0
AGG14M	4	aFSAI	647	-	-	36.0	21.5	57.6
		BAMG	133	1.085	1.286	33.5	22.3	55.8
M20	4	aFSAI	3972	-	-	72.2	368.0	440.2
		BAMG	151	1.054	1.677	170.5	70.1	240.7
TRIPOD24M	5	aFSAI	2821	-	-	60.8	127.6	188.4
		BAMG	222	1.041	1.116	24.2	22.2	46.5
C4ZZ44M	12	EXTI	16	1.573	2.580	34.0	3.0	37.0
		CLAS	46	1.611	1.944	23.2	6.5	29.7
		HYBC	35	1.585	2.032	26.6	6.1	32.7
GEO61M	12	aFSAI	609	-	-	27.2	37.2	64.4
		BAMG	224	1.029	1.048	55.4	48.9	104.0
POI65M	8	EXTI	15	1.344	4.475	37.7	1.84	39.5
		CLAS	26	1.381	2.332	30.6	4.12	34.7
		HYBC	16	1.360	2.881	24.1	1.79	25.9
PFLOW73M	8	EXTI	2233	1.234	2.345	46.6	544.0	590.6
		CLAS	1987	1.236	1.391	19.4	472.5	491.9
		HYBC	1991	1.234	1.448	20.8	399.7	420.5
C4ZZ134M	25	aFSAI	4393	-	-	108.9	317.9	426.8
		BAMG	173	1.028	1.189	204.6	59.4	264.0

increasing the number of nodes from 8 to 64. Again, it can be observed that both solution time and the number of iterations remain almost constant for every test, as is theoretically expected.

5 CONCLUSIONS AND FUTURE WORK

This work presents Chronos, an effective general purpose AMG solver. The proposed framework brings together both well known and novel algorithms adapting all of its components to the specific problem at hand regardless to the application it arose from. A set of real world problems, including structural mechanics, CFD and underground applications, are solved on the modern super computer Marconi100 to assess the numerical performance of Chronos. It is shown through an extensive experimentation that the package is robust and effective in addressing a wide range of industrial applications with several solution strategies. Chronos is available along with the source of a sample program from https://www.m3eweb.it/chronos/ for research purpose and benchmarking. The next steps will be the ex-

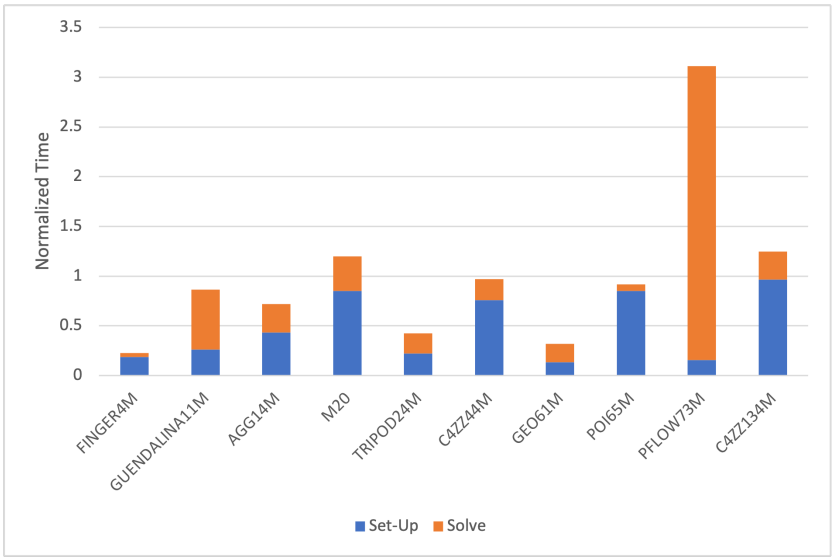


Figure 2: Set-Up and Solve times divided by the number of equations per node.

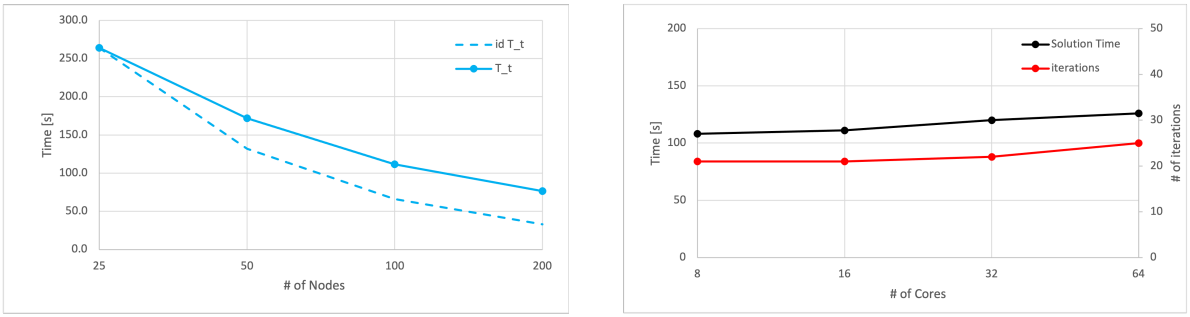


Figure 3: Strong scalability test on `C4ZZ134M` (left) with total solution time, T_t, vs. number of Marconi100 nodes. Solid line represents the true wall time, dashed line ideal time. Weak scalability test on a 3D Poisson problem (right) showing both total solution time and number of iterations to converge. The computational load on each Marconi100 node is kept constant at about 7,040,000 equations.

tensive benchmarking of the GPU-accelerated version and the development of a new implementation for the next generation FPGA systems.

REFERENCES

[1] R. Baggio, A. Franceschini, N. Spiezia, and C. Janna, *Rigid body modes deflation of the preconditioned conjugate gradient in the solution of discretized structural problems*, Computers & Structures, 185 (2017), pp. 15–26, https://doi.org/10.1016/j.compstruc.2017.03.003.

[2] A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang, *Multigrid Smoothers for Ultraparallel Computing*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2864–2887, http://doi.org/10.1137/100798806.

[3] M. BERNASCHI, P. D'AMBRA, AND D. PASQUINI, *AMG based on compatible weighted matching for GPUs*, Parallel Computing, 92 (2019), pp. 1–13, http://doi.org/10.1016/j.parco.2019.102599

[4] H. DE STERCK, U. M. YANG, AND J. J. HEYS, *Reducing Complexity in Parallel Algebraic Multigrid Preconditioners*, SIAM Journal on Matrix Analysis and Applications, 27 (2006), pp. 1019–1039, http://doi.org/10.1137/040615729.

[5] J. J. DONGARRA, J. DUCROZ, I. DUFF, S. HAMMARLING, *A set of level 3 basic linear algebra subprograms*, ACM Transactions on Mathematical Software 16, (1990), pp. 1–17.

[6] A. FRANCESCHINI, V. A. PALUDETTO MAGRI, G. MAZZUCCO, N. SPIEZIA, AND C. JANNA, *A robust adaptive algebraic multigrid linear solver for structural mechanics*, Computer Methods in Applied Mechanics and Engineering, 352 (2019), pp. 389–416. ihttp://doi.org/10.1016/j.cma.2019.04.034

[7] S. KORIC, Q. LU, AND E. GULERYUZ, *Evaluation of massively parallel linear sparse solvers on unstructured finite element meshes*, Computers & Structures, 141(C), (2014), pp. 19–25, http://doi.org/10.1016/j.compstruc.2014.05.009

[8] C. JANNA, M. FERRONATO, AND G. GAMBOLATI, *The use of supernodes in factored sparse approximate inverse preconditioning*, SIAM Journal on Scientific Computing, 37 (2015), pp. C72–C94, https://doi.org/10.1137/140956026.

[9] C. JANNA, M. FERRONATO, F. SARTORETTO, AND G. GAMBOLATI, *FSAIPACK: A software package for high-performance factored sparse approximate inverse preconditioning*, ACM Trans. Math. Softw., 41 (2015), pp. 10:1–10:26, http://doi.acm.org/10.1145/2629475.

[10] A. N. NATALI, E. L. CARNIEL, C. G. FONTANELLA, S. TODROS, G. M. DE BENEDICTIS, M. A.CERRUTO AND W. ARTIBANI, *Urethral lumen occlusion by artificial sphincteric devices: a computational biomechanics approach*, Biomechanics and modeling in mechanobiology, 16 (2017), pp. 1439–1446.

[11] A. N. NATALI, E. L. CARNIEL, C. G. FONTANELLA, A. FRIGO, S. TODROS, A. RUBINI, G. M. DE BENEDICTIS, M. A.CERRUTO AND W. ARTIBANI, *Mechanics of the urethral duct: tissue constitutive formulation and structural modeling for the investigation of lumen occlusion*, Biomechanics and modeling in mechanobiology, 16 (2017), pp. 439–447.

[12] V. A. PALUDETTO MAGRI, A. FRANCESCHINI, AND C. JANNA, *A novel AMG approach based on adaptive smoothing and prolongation for ill-conditioned systems*, SIAM Journal on Scientific Computing, 41 (2019), pp. A190–A219, https://doi.org/10.1137/17M1161178.

[13] F. H. ROUET, C. ASHCRAFT, J. DAWSON, R. GRIMES, E. GULERYUZ, S. KORIC, R. F. LUCAS, J. S. ONG, T. A. SIMONS, AND T. T. ZHU, *Scalability Challenges of an Industrial Implicit Finite Element Code*, 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, pp. 505–514, http://doi.org/10.1109/IPDPS47924.2020.00059.

[14] K. STÜBEN, *A review of algebraic multigrid*, Journal of Computational and Applied Mathematics, 128 (2001), pp. 281 – 309, http://www.sciencedirect.com/science/article/pii/S0377042700005161. Numerical Analysis 2000. Vol. VII: Partial Differential Equations.

[15] U. TROTTENBERG, C. OOSTERLEE, AND A. SCHÜLLER, *Multigrid*, Academic Press, 2001, https://www.elsevier.com/books/multigrid/trottenberg/978-0-08-047956-9.

[16] M. VÁZQUEZ, G. HOUZEAUX, S. KORIC, A. ARTIGUES, J. AGUADO-SIERRA, R. ARÍS, ET AL., *Alya: Multiphysics engineering simulation toward exascale*, Journal of Computational Science, 14 (2016), pp. 15–27, http://doi.org/10.1016/j.jocs.2015.12.007.

[17] J. XU AND L. ZIKATANOV, *Algebraic multigrid methods*, Acta Numerica, 26 (2017), p. 591–721, http://dx.doi.org/10.1017/S0962492917000083.