Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

ANALYSIS OF THE SVD SCALING ON LARGE SPARSE MATRICES

María de Castro-Sánchez[1], José A. Moríñigo[2], Filippo Terragni[3], and Rafael Mayo-García[2]

[1]School of Aerospace Engineering, Technical University of Madrid (UPM), SPAIN
[2]Dept. of Technology, CIEMAT, Madrid, SPAIN
[3]Dept. of Mathematics, Universidad Carlos III de Madrid, Leganés, SPAIN

ABSTRACT

There has been great interest in the Singular Value Decomposition (SVD) algorithm over the last years because of its wide applicability in multiple fields of science and engineering, both standalone and as part of other computing methods. The advent of the exascale era with massively parallel computers brings incredible possibilities to deal with very large amounts of data, often stored in a matrix. These advances set the focus on developing better scaling parallel algorithms: e.g., an improved SVD to efficiently factorize a matrix. This study assesses the strong scaling of four SVDs of the SLEPc library, plugged into the PETSc framework to extend its capabilities, via a performance analysis on a population of sparse matrices with up to 10^9 degrees of freedom. Among them, there is a randomized SVD with promising performance at scale, a key aspect in solvers for exascale simulations since communication must be minimized for scalability success.

1 INTRODUCTION

Computational performance of the SVD has been into focus over the last decades (Dongarra et al. 2018) due to its instrumental role in many fields of science and engineering (e.g., computational chemistry, astronomy, finance, plasma and fluid physics, etc.). Presently, with the advent of the big data and exascale computing revolutions, the availability of an efficient, scalable SVD implementation turns out to be an issue of crucial concern (even when sometimes the aim is 'simply' to fit a huge amount of data into the distributed memory of the supercomputer, and then apply a parallel SVD). Typically, the SVD acts on a matrix which may be sparse or dense, tall-skinny or fat-shaped, well-conditioned or very ill-conditioned, to mention some frequent scenarios that are commonly linked to the specific fields above mentioned.

As an example connected to our group interest, recent promising ideas from the research in numerical linear algebra point out to use an SVD as part of new preconditioner formulations, to be specific taking advantage of randomized SVD schemes (Buluc et al. 2021; Martinsson 2018; Halko et al. 2009) because of their computing economy, which suggests a better convergence of iterative solvers applied to extreme-scale systems of linear equations. In addition to speeding up time-to-solution, other benefits arise: improved scalability and resilience (Moríñigo et al. 2022). These metrics are key aspects in the design of state-of-the-art solvers, as it is the case of communication-avoiding Krylov solvers.

This study delves into an authors' previous investigation (Ferrero-Roza et al. 2023) on the performance of several CPU-based SVD versions available in the SLEPc (Scalable Library for Eigenvalue Problem Computations) library (Román et al. 2022), which extends the capabilities of PETSc (Portable, Extensible Toolkit for Scientific Computation) framework (Mills et al. 2021). In particular, much larger matrix dimensions $n \times n$ are taken into consideration, up to $n \approx 10^9$, which implies 100x the largest n taken in (Ferrero-Roza et al. 2023). The chosen population of matrices spans different scenarios of coupling among the distributed parallel processes over the cluster, then a varying impact on the cluster communications.

The structure of this article is as follows. The next section gives an overview of the PETSc-SLEPc framework and introduces the generalities of the deterministic and randomized SVD algorithms of SLEPc

applied to the matrices. Section 3 presents the computing framework setup as well as the characteristics of the sparse matrices used in the benchmarking. Then, it follows the Results section where the strong scaling tests carried out with two computing clusters at CIEMAT are discussed. Section 5 states the recent research related to this investigation. To end some conclusions are given.

2 SVD SOLVERS

2.1 PETSc Framework and SLEPc

PETSc is a suite of libraries designed to ease the implementation and customization of large-scale parallel application codes. A variety of parallel linear and nonlinear solvers (mostly iterative), besides time integrators, are included as building blocks to simplify the coding. In addition, it provides the functionality needed within application codes by means of advanced matrix and vector assembly routines that store, distribute and operate specifically on sparse and dense entities to speed up the computations. PETSc uses the Message Passing Interface (MPI) standard for all the communications in distributed computing and recent versions have started to include several GPU-based algorithms (Mills et al. 2021). Optimally, PETSc interfaces a variety of numerical algebra libraries: among them, SLEPc provides a bunch of parallel SVD solvers (PETSc's SVD algorithm is serial, with no parallel counterpart included).

SLEPc (Román et al. 2022) is a library for the solution of large sparse eigenproblems and other problems such as the partial SVD of both sparse and dense matrices on parallel computers. It also provides solvers for the computation of the action of a matrix function on a vector by a Krylov method. SLEPc is built on top of PETSc and works as a seamless extension (it shares the same programming paradigm as PETSc). The implemented SVDs are well-suited for sparse matrices derived from the discretization of partial differential equations. Besides, it interfaces other external software libraries (i.e., ScaLAPACK (Blackford et al. 1997) and more) to efficiently carry out parallel SVDs on large dense matrices.

2.2 Overview of the SVD

A very brief mathematical description of the SVD is given in what follows. Exhaustive information can be found elsewhere besides the user's manual of SLEPc (Trefethen and Bau 1997).

The SVD of an $m \times n$ matrix A corresponds to $A = U\Sigma V^*$, where $U = [u_1, u_2, ...u_m]$ is an $m \times m$ unitary matrix ($U^*U = I$), whose columns are the left singular vectors; $V = [v_1, v_2, ...v_n]$ is an $n \times n$ unitary matrix ($V^*V = I$), whose columns are the right singular vectors; and Σ is an $m \times n$ diagonal matrix with real entries (non-negative, sorted in decreasing order) called singular values of A, that is $\Sigma_{ii} = \sigma_i$ for $i = 1, 2, ...min(m, n)$ (illustrated in Figure 1a). If A is real, then U and V result to be real and orthogonal ($U^* = U^T, V^* = V^T$). In the general case of a non-square matrix A with $m \gg n$, the resulting factorization (sometimes called *thin* SVD) exploits the slenderness of the matrix, then U collapses into an $m \times n$ matrix (depicted in Figure 1b), and the factorization reads $A = U_n \Sigma_n V_n^*$. The n singular triplets correspond to (σ_i, u_i, v_i) for $i = 1, 2, ...n$.

2.3 Deterministic SVDs in SLEPc

The computation of the SVD of a matrix can be done with an equivalent eigenvalue problem setup in SLEPc, recasting the matrix A conveniently to extract its singular triplets:

1. The *cross* product matrix method builds the matrix A^*A or AA^*.
2. The *cyclic* matrix method builds the matrix $H(A) = \begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix}$.

The default SVD solver in SLEPc is the one using the cross product matrix (*cross*). This has several implications regarding the number of singular values that can be computed because of the size of Σ. Then, a drawback of this method is that there is a loss of accuracy in computing the smallest singular values. On

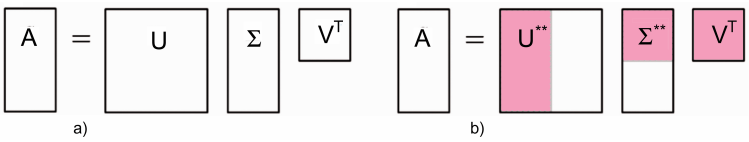

Figure 1: SVD factorization of matrix A: standard SVD (left); and *thin* SVD (right), done for $m \gg n$.

the other hand, the eigendecomposition of $H(A)$ of the *cyclic* matrix method outputs singular values that are not squared, then the smallest computed values will be more accurate. The expense of this approach is more RAM memory and extra computing cost compared to *cross*. Hence, though the *cross* product matrix method tends to be faster and the most memory-efficient approach for the standard SVD, it is only appropriate when the leading singular values are searched for.

Other specific, robust SVD formulations available in SLEPc are two *Lanczos*-type solvers: the so-called *Lanczos* and the *thick-restart (TR) Lanczos*. Both are two-stage algorithms. Taking $A = PBQ^*$ (being P and Q unitary matrices, and B an $m \times n$ upper bidiagonal matrix), the built tridiagonal matrix B^*B is unitarily similar to A^*A. Hence, expressing the SVD of B as $X\Sigma Y^*$ immediately leads to state $U = PX$ and $V = QY$. Its first stage is iterative and builds the bidiagonalization of A. The *TR Lanczos* is a variant that exploits the restarting mechanism and conducts one-sided or two-sided reorthogonalization via iterated Classical Gram-Schmidt method in the bidiagonalization process. In particular, the *TR* concept is easier to implement than the restarting used in the pure *Lanczos*-based SVD (Hernández et al. 2008; Alvarruíz et al. 2022).

2.4 Randomized SVD in SLEPc

The *rSVD* uses a low-rank representation of A and exploits randomized linear algebra techniques (Martinsson 2018; Halko et al. 2009). Due to accuracy concerns, the size of the vector basis *ncv* is set (at least) double the number of requested singular values *nsv* ($ncv \geq 2 \cdot nsv$). The implemented *rSVD* is iterative and stops when either the prescribed tolerance condition or the maximum number of iterations is reached. Then, the tolerance should be set with care to balance computing cost and desired accuracy because the iterative block is a time consuming operation driven by matrix-matrix and matrix-vector multiplications.

3 BENCHMARKING SETUP AND METHODOLOGY

3.1 Computing Environment

The software framework comprises the library PETSc v3.19.5 and SLEPc v3.19.2. Besides, MPIch v4.1.1 is installed as an external package of PETSc. The installation of SLEPc and MPIch is straightforward once the PETSc environment has been set since their respective `configure` scripts inherit the PETSc setup parameters for optimal performance. Compilation has been done with the GNU compilers (v11.3.1 in ACME and v8.5.0 in XULA clusters), using aggressive `-O3` optimization to enhance the acceleration of executions according to prior performance tests (Ferrero-Roza et al. 2023). All libraries and codes have been compiled under 64-bit arithmetic, enforced at the configuration stage and required by the size of the sparse matrices considered in this study. With smaller matrices (Davis and Hu 2011) as the ones analyzed by the authors in (Ferrero-Roza et al. 2023), the 32-bit arithmetic (default in PETSc) is enough.

Strong scaling tests have been executed in two clusters located at CIEMAT: ACME, which is a research facility (Rodríguez-Pascual et al. 2019) intended for software development and benchmarking; and XULA, which is a production facility shared by research groups. Both have the Slurm workload manager installed.

In ACME a homogeneous partition of 10 nodes has been used. Each node comprises 2 Intel Xeon Gold 6138 'Skylake' processors (20 cores/CPU) @2.0 GHz, with 192 GB RDIMM memory. Node connection is done with a 56 Gb/s FDR InfiniBand network. The setup permits MPI-based parallel executions up to 400 ranks (one rank per core; hyperthreading disabled).

In XULA a homogeneous partition of 64 nodes (denoted Xula3) has been used. Each node comprises 2 Intel Xeon Gold 6342 processors (24 cores/CPU) @2.8 GHz, with 256 GB DDR4 memory. Node connection is done with a 100 Gb/s Enhanced Data Rate (4X EDR) InfiniBand network. The setup permits MPI-based parallel executions up to 3,072 ranks (one rank per core; hyperthreading disabled).

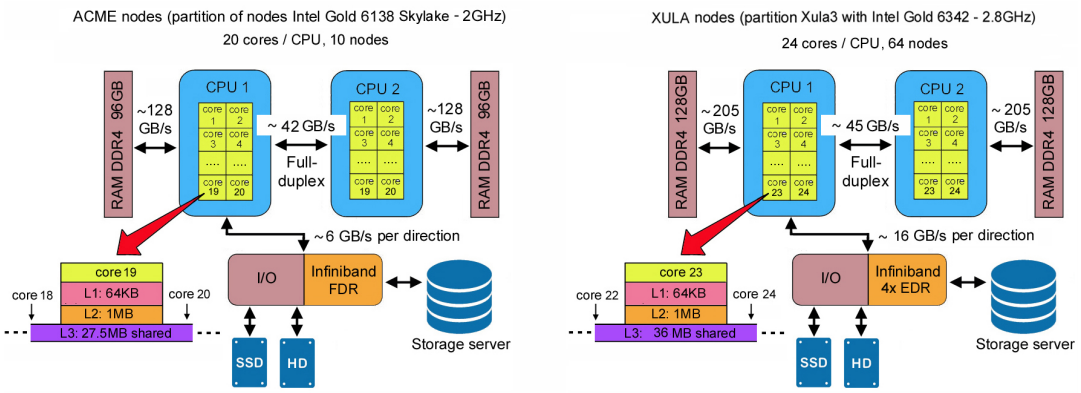

Figure 2: Computing nodes architecture of ACME (left) and XULA (right) clusters. Bandwidth corresponds to RAM read/write operations, intra-node (CPUs) bus, and inter-node communication is indicated.

All tests have been performed with dedicated access to the partition nodes, enforced with the `exclusive` command of Slurm. Thus, it is ensured that no interference with other users occurs during MPI executions. It should be mentioned that benchmarking tests in XULA were completed at its commissioning phase (before entering production), so network traffic caused by other users was reduced to a minimum.

3.2 Matrices

Two types of square, real, sparse matrices have been built with size n up to order 10^9 (equivalent to the degrees of freedom, DoF). Four matrices are derived from solving the Poisson equation in 1D, 2D and 3D domains using 2^{nd}-order stencils to discretize the Laplace operator; in addition a 4^{th}-order stencil has been implemented for the 3D case. Poisson-like equations appear in many fields of Physics as a member of the system of governing equations: gyrokinetics of plasmas, materials physics, shallow water dynamics, to cite some. Besides, their numerical solution may take a major portion of the total computing cost (e.g., in problems involving incompressible fluid dynamics). The second type corresponds to four banded matrices (11, 25, 39 and 71 diagonals) filled with Gaussian numbers in [-1,1]. This type of non-symmetric matrices is representative of many-body complex quantum systems with local interactions (similarly, the Anderson model matrix provided by the ScaMaC library (Alappat et al. 2020) describes the motion of a quantum-mechanical particle in a disordered solid). Therefore, in total 8 matrices of up to 10^9 DoF are considered (some have been downscaled to a smaller size to explore the sensitivity to having less computing load per core, see next section). It is noticed that all the analyzed matrices have structured patterns of non-zero numbers (nnz), see Figure 3, hence there is an almost linear relationship between n and nnz.

3.3 Execution Parameters

Generation of matrices for PETSc SVD-solvers has been accomplished on-the-fly, which facilitates the I/O operations during tests setup. Relative tolerance is bounded by 10^{-3} and the leading ten singular values are requested (no singular vectors) for output. These are stored in the output file for accuracy comparison.

Computing time corresponds to the SVD-solvers object execution in a strict sense, then neither accounting for the time invested in matrix building nor other C structures setups or memory de-/allocations. It should be noticed that an a priori study was carried out with a population of matrices of 10^7 to 10^9 DoF in both clusters to quantify the expected walltime fluctuations in identical executions. This has led to bound the

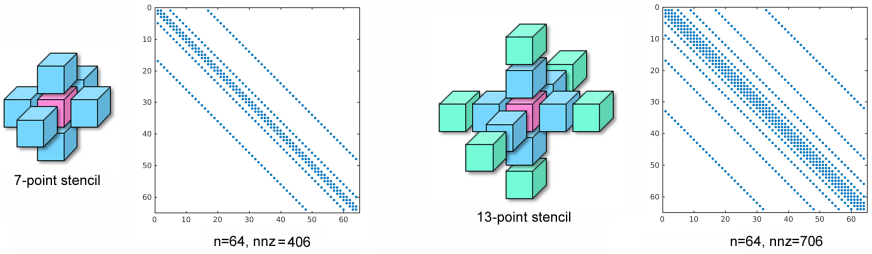

Figure 3: Sketch of discretized 3D Laplacian operator of the Poisson equation, corresponding to a 7-point (left) and 13-point (right) stencils of second and fourth-order accuracy, respectively. Differences in pattern of non-zero matrix entries (*nnz*) is shown for a matrix size set to $n = 64$ (chosen for visualization purposes).

variability margin within $\pm 1.5\%$, as well as to accept the mean of three consecutive executions as adequate. Distribution of MPI ranks over the cluster has been enforced with the `cyclic` Slurm command. Besides, parameters to control the SVD versions execution have been set following (Román et al. 2022) (specifically, the singular vectors subspace dimension is triple the number of requested singular values).

4 RESULTS

The numerical experiments correspond to two types of studies: sensitivity and strong scaling tests. The first one has been completed in cluster ACME because of the much more computing time available. The second study has been carried out in XULA besides ACME. The computing time available using 64 nodes of the entire partition of XULA (denoted Xula3 in Figure 2) for benchmarking has been used to test the strong scaling of three SVD versions: *cyclic*, *cross* and *rSVD*. The *TR Lanczos* is not included among these (only tested in ACME) due to the above mentioned usage restriction of partition Xula3.

4.1 Sensitivity Tests

The interplay between SVD scaling and matrix size n has been investigated by building matrices with increasing size (equivalent to increasing *DoF*). Strong scaling corresponding to n of order $O(10^7)$, $O(10^8)$ and $O(10^9)$ for the 1D Poisson problem matrix (Figure 4), 2D Poisson problem matrix (Figure 5) and banded matrix with 11 diagonals (Figure 6) have been analyzed in ACME. The scenario $n \sim O(10^{10})$, which leads to an *nnz* 10x larger than its counterpart for $n \sim O(10^9)$, results in being unaffordable with the cluster ACME because of its total RAM constraints. Consequently, it can be stated that $n \sim O(10^9)$ is the largest problem that fits into ACME according to its present day configuration. The plots clearly show that *DoF* of $O(10^7)$ implies a too low computing load per core, as the quicker decay of the speedup and parallel efficiency indicates. On the contrary, the scaling behaviour improves as *DoF* tends to $O(10^9)$ because *nnz* is proportional to n, thus the computing load per core increases. It is clearly seen that the larger the matrix size, the closer the speedup curve to its ideal counterpart results to be, as well as the flat portion of the parallel efficiency enlarges over a wider range of MPI ranks. Interestingly, all matrices show that the speedup and parallel efficiency curves get more and more similar to each other, then the scaling behaviour becomes more insensitive to the SVD algorithm itself. However, the considered SVD versions significantly in the execution time (see Figure 7), which is a fundamental criterion for taking decisions about the suitability of a specific SVD method to apply.

In addition, the effect of the *nnz* pattern on the scaling properties has been analyzed using a sequence of Poisson equation discretizations in 1D, 2D and 3D computational domains and $n \sim O(10^9)$ (see Figure 8). Typically the increase of the domain dimensionality implies a progressive spreading of non-zero numbers away from the main diagonal of the matrix. By default PETSc partitions matrices into bands of rather similar numbers of rows, then these are distributed over the cores. In particular, for the 1D Poisson problem, this means that all *nnz* are placed into the diagonal blocks of their respective bands. A priori this concentration

of non-zeroes makes the communication among the MPI ranks less intensive. The opposite situation arises for the 3D Poisson problem matrix, whose non-zeroes are located in both diagonal and off-diagonal blocks for each band. Specifically, when higher-order stencils are involved, the spreading of the *nnz* may be quite strong (the matrices given in Figure 3 show the larger spreading of the *nnz* as a consequence of the stencil complexity). Precisely the progressive scaling degradation observed for the Poisson problem matrices as the domain dimensionality increases from 1D to 3D (see Figure 8) can be explained by the much stronger computation coupling involved. That is, the spreading of non-zeroes across the off-diagonal matrix blocks, as dimensionality increases, implies more data to communicate among the MPI ranks over the cluster.

4.2 Strong Scaling in XULA

Benchmarking using the partition Xula3 has demonstrated a very good strong scaling, showing a quite remarkable speedup and parallel efficiency achieved for both 3D Poisson problem (discretized with 7- and 13-point stencils) and the three banded matrices "BD" (25, 39 and 71 diagonals) up to the maximum number of MPI ranks tested (2,560 cores). This behaviour is summarized in Figures 9 and 10, respectively.

The 3D Poisson problem matrix discretized with a 7-point stencil has been benchmarked in both clusters ACME (right column in Figure 8) and XULA (upper row in Figure 9), thus the comparison of the scaling behaviour deserves attention as qualitative differences are visible. First, it should be noted that fewer nodes are available in ACME, which favours a quicker concentration of MPI ranks per node over the cluster compared to XULA (which has 32 nodes set for this case and 64 for the BD matrices). Said that, it is reasonable to expect heavier memory contention issues to appear earlier in ACME than in the second cluster, hence a worse SVD scaling is expected. In addition, ACME's InfiniBand has half XULA's bandwidth, so internode communication is slower. Lastly, all tests in ACME have been designed to fill 100% of the CPUs' cores (400 MPI ranks in total). On the contrary, XULA's filling is 67% of the CPUs' cores (1,024 ranks of 1,536 in total, with the same ratio in those tests carried out with the BD matrices). This much lower filling ratio seems beneficial to reduce the memory contention issues. Their effects are visible in Figure 11 by comparing the attained speedup with the BD25 and BD71 matrices using 32 and 64 nodes and the same cluster architecture (Xula3). It can be seen that at 1024 MPI ranks the corresponding speedup with 64 nodes is better because of the lower CPUs filling (33% vs. 66% with 32 nodes). Therefore, cluster XULA offers a better scenario to attain higher performance than ACME for SVD benchmarking. Precisely this point is confirmed by the much better scaling obtained with the entire population of matrices.

The comparison of the SVD versions (*cyclic*, *cross* and *rSVD*) in XULA shows that *rSVD* provides very competitive execution time (see the central column in Figures 9 and 10): while for BD matrices it shows quite similar execution time compared to the *cross* version, the *rSVD* applied to the 3D Poisson mroblems matrices clearly outperforms the quickest *cross*. An explanation for this behaviour is that randomized algebra typically implies asymptotic savings in the cluster communications. Then, it turns out to be more efficient at dealing with the *nnz* pattern of the 3D Poisson discretizations, which yield many non-zeroes spread across several off-diagonal matrix blocks. Consequently, it is more communication-intensive. This finding is a rather promising property at scale which demands further benchmarking with a wider set of matrices (in terms of larger n and *nnz* pattern variation).

5 RELATED WORK

A major concern regarding the parallel SVD algorithm is driven by the difficulty in overcoming its memory contention and communication constraints at scale. At present, systematic benchmarking of available SVD implementations is scarce and summarized in few works. In (Dongarra et al. 2018) a review of state-of-the-art SVD implementations for dense matrix computations with CPUs and GPUs is provided. Systematic tests on a multicore machine (in-node performance) and a distributed computing platform have been carried out with the SVD of ScaLAPACK in a distributed-memory computer. Albeit they quantify the attained relative speedup with a reference SVD in distributed computing for increasing matrix size, no

analysis of the absolute speedup is provided, nor do they discuss techniques for solving SVD problems with sparse linear systems. Interestingly, both square and very-tall dense matrices scenarios are compared and they show that the tallest matrices exhibit an improved speedup attributed to *caché* issues.

The SVD scaling has been analyzed in (Blanchard et al. 2019), where recent formulations tuned for the full computing capabilities of some novel architectures are investigated. In particular, the one-stage SVD polar decomposition has shown its improved scaling compared with well established numerical libraries as ScaLAPACK and others on pure CPU distributed-memory computers and hybrid CPU-GPU platforms.

In (Schmidt 2020) three MPI-based SVD solvers corresponding to the normal-equations, QR-based and randomized-SVD formulations are analyzed and executed on the Summit supercomputer (USA). These implementations are tested with a group of tall/skinny matrices and their performance is compared.

In (Hernández et al. 2008) the authors conduct some strong scaling tests using the PETSc-SLEPc framework on a reduced set of sparse matrices, hence to test the scaling properties of the SVD solver based on the restart Lanczos bidiagonalization. Their results are extended in various presentations delivered by the same authors, focused on strong and weak scaling attained with some SVD versions of SLEPc. However, a systematic benchmarking is missing.

Another fundamental aspect has to do with the matrix generation and its input to the SVD solver. Benchmarking of SVD libraries requires sparse and dense matrices of increasing dimension. There exist well known sparse matrices repositories such as the Suite Sparse Matrix Collection (Davis and Hu 2011), which provides a variety of matrices of size $n \leq 10^8$. When larger sparse matrices are needed, one option is the usage of the ScaMaC library (Alappat et al. 2020), a scalable matrix generation framework for a broad set of physics problems, which permits acting on the size of the problem matrix, as well as controlling its sparsity pattern by modifying a few involved parameters, then designing both strong and weak scaling tests. It comes with an example template for PETSc to help with the implementation of the on-the-fly matrix building and partitioning. In addition, the generation of large dense matrices with prescribed singular values and condition numbers can be efficiently done according to (Fasi and Higham 2021). This work describes a methodology suitable for scaling tests with extreme-scale matrices in petascale computers and beyond.

The present investigation extends the strong scaling results provided in (Ferrero-Roza et al. 2023) with systematic benchmarking of sparse matrices up to 10^9 *DoF* and using the most promising parallel SVD solvers coded in the PETSc-SLEPc framework. Their analysis at scale leads to a better knowledge of their potential, both standalone and as a building block of a solver which exploits an SVD factorization.

6 CONCLUSIONS

The strong scaling of four SVD versions (three deterministic plus one randomized) implemented in SLEPc has been analyzed using two classes of sparse matrices: Poisson-derived and banded, both up to size $n \sim 10^9$. The tests performed in cluster XULA, which has a more upgraded architecture and quicker InfiniBand network with respect to ACME, show that all matrices with 10^9 *DoF* exhibit a very good speedup within the tested range of MPI ranks (1,024 ranks for Poisson matrices; 2,560 ranks for BD matrices).

Interestingly, the execution time of the randomized SVD shows promising behaviour compared to the quickest deterministic SVD version (*cross*) at scale. This is specially remarkable for the 3D Poisson matrices (where the *nnz* pattern leads to a much stronger coupling among the MPI ranks than for the BD matrices) since the *rSVD* clearly outperforms *cross*. This result suggests the benefit of the randomization concept in the SVD implementation as it seems to drive communication reduction over the cluster nodes, which means a better strong scaling. This issue results to be of key importance in solvers for exascale simulations, since communication must be minimized for scalability success. Finally, it is worth mentioning that our group plans to conduct further research with parallel executions of at least $O(10^4)$ MPI ranks with a broader range of sparsity patterns to deepen into this behaviour in the next future, also by including additional metrics to better quantify the involved communication overhead and memory contention.

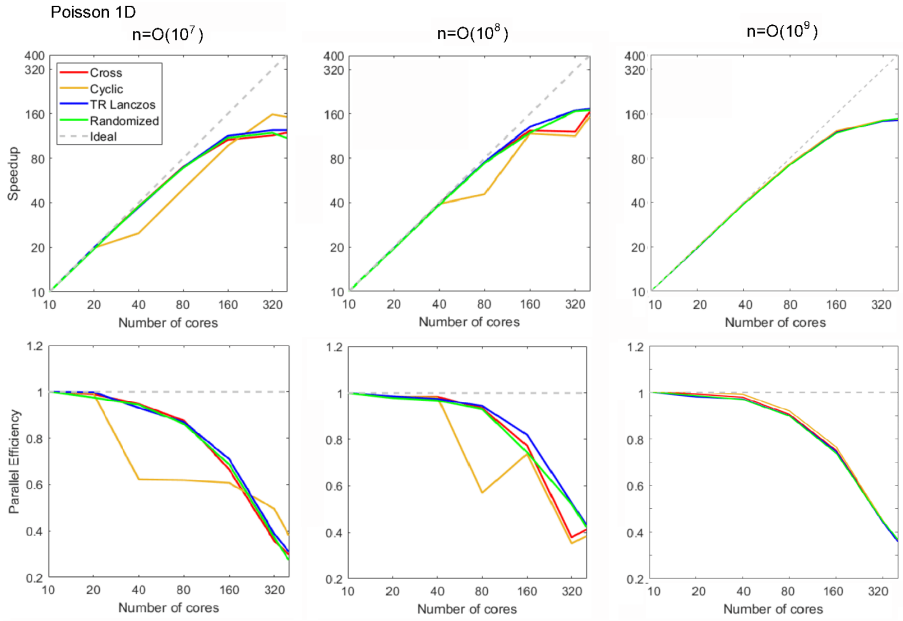

Figure 4: Speedup and parallel efficiency attained with four SVD algorithms applied to the 1D Poisson problem matrix with a 3-point stencil discretization, solved in cluster ACME. Columns correspond to matrix DoFs equal to order 10^7 (left), 10^8 (center) and 10^9 (right).

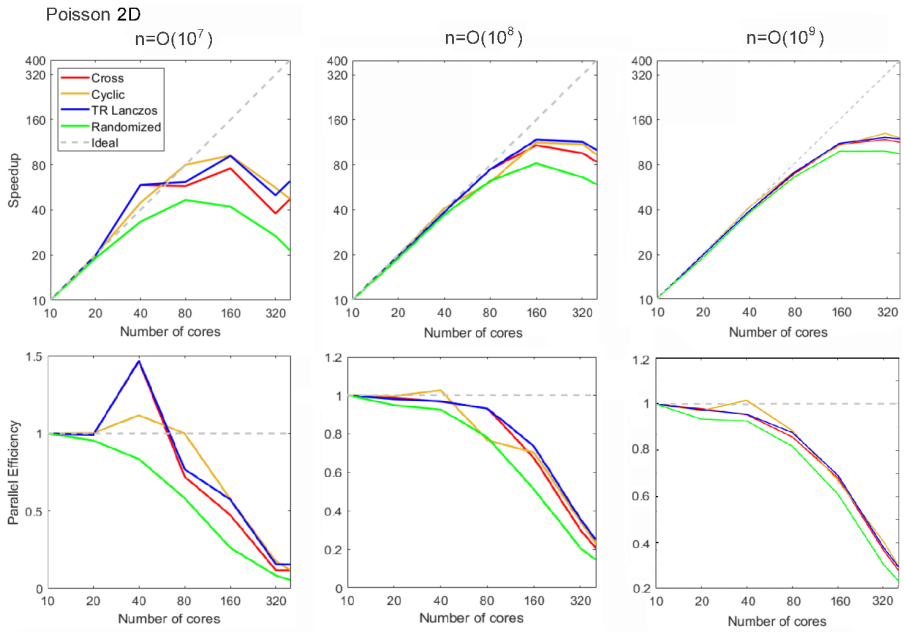

Figure 5: Speedup and parallel efficiency attained with four SVD algorithms applied to the 2D Poisson problem matrix with a 5-point stencil discretization, solved in cluster ACME. Columns correspond to matrix DoFs equal to order 10^7 (left), 10^8 (center) and 10^9 (right).

ACKNOWLEDGMENTS

CIEMAT contribution was partially funded by EuroHPC project EoCoE3 (Grant ID: 101144014) and EC H2020 project EU LAC ResInfra Plus (Grant ID: 101131703), and led within the SciTrack group tasks

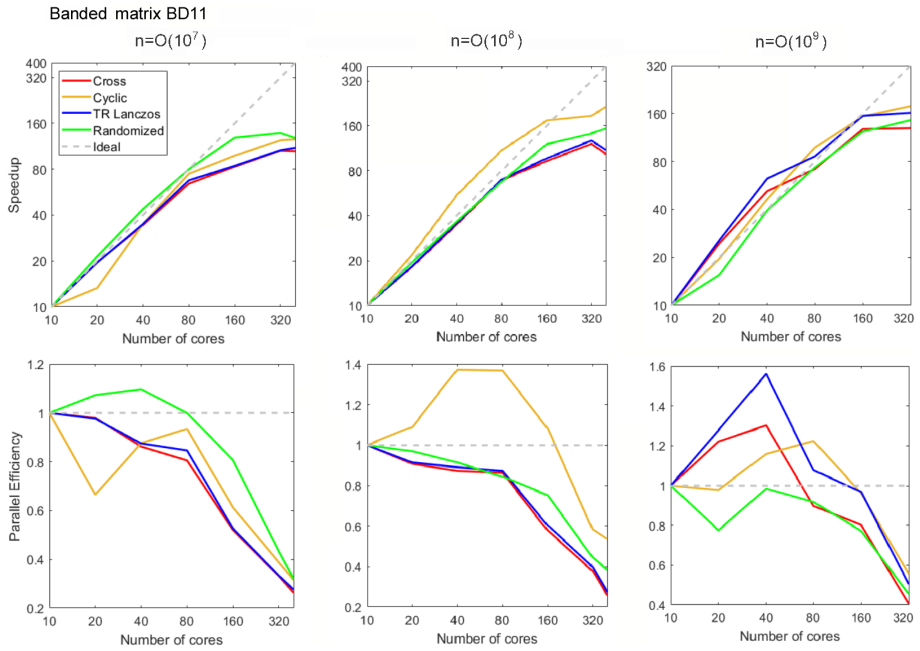

Figure 6: Speedup and parallel efficiency attained with four SVD algorithms applied to banded matrix BD11 of 11 diagonals (main plus 5 diagonals on either side filled with Gaussian numbers), solved in cluster ACME. Columns correspond to matrix DoFs of order 10^7 (left), 10^8 (center) and 10^9 (right).

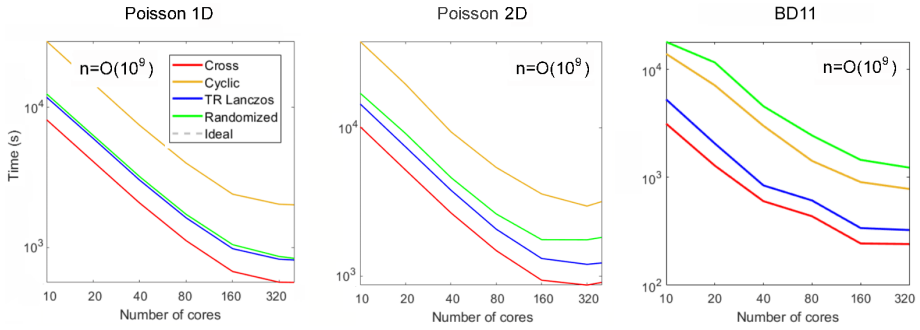

Figure 7: Execution time with four SVD algorithms applied to Poisson 1D (left), 2D Poisson (center) and BD11 (right) matrices, solved in cluster ACME. All cases correspond to matrix DoFs of order 10^9.

towards exascale computing. F. Terragni (UC3M) was supported by the ERDF Spanish Ministry of Science, Innovation, and Universities under grant PID2020-112796RB-C22. The authors acknowledge access to cluster ACME and thank XULA's administrator A. Alberto-Morrillas for her kind support.

REFERENCES

Alappat, C. L., A. Alvermann, A. Basermann, H. Fehske, Y. Futamura, M. Galgon, , , , , , , , , , , *et al.* 2020. "ESSEX: Equipping Sparse Solvers For Exascale". In *Software for Exascale Computing - SPPEXA 2016-2019*, edited by H.-J. Bungartz, S. Reiz, B. Uekermann, P. Neumann, and W. E. Nagel, 143–187. Cham: Springer International Publishing.

Alvarruíz, F., C. Campos, and J. E. Román. 2022. "Thick-restarted Joint Lanczos Bidiagonalization for the GSVD". *ArXiv*:1–25.

Blackford, L. S., J. Choi, A. Cleary, E. D'Azeuedo, J. Demmel, I. Dhillon, , , , , , *et al.* 1997. *ScaLAPACK User's Guide*. USA: Society for Industrial and Applied Mathematics.

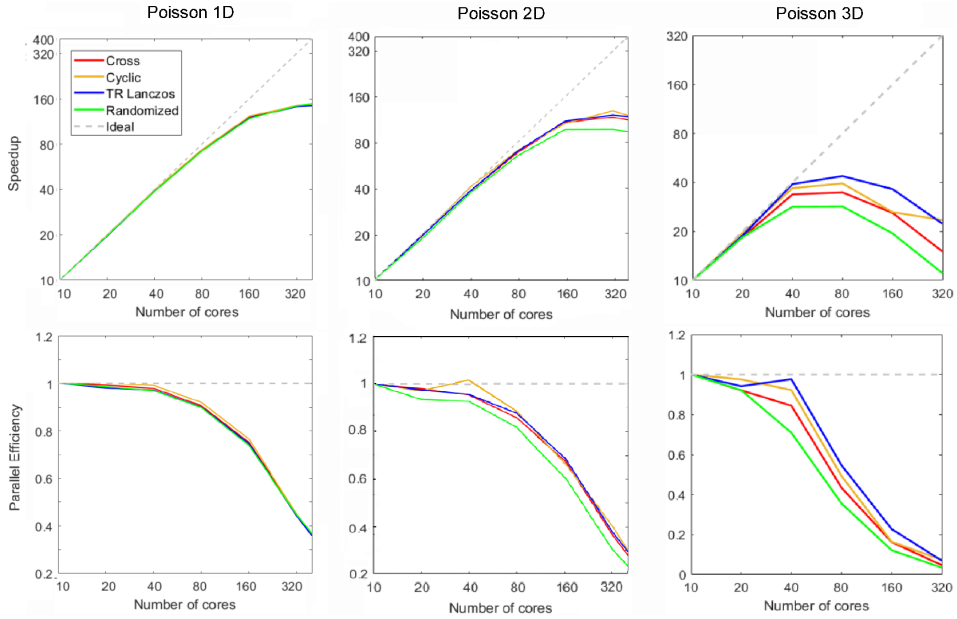

Figure 8: Speedup and parallel efficiency attained with four SVD algorithms applied to the 1D, 2D and 3D discretization of the Poisson equation. Columns correspond to Poisson 1D with 3-point stencil (left); Poisson 2D with 5-point stencil (center); and Poisson 3D with 7-point stencil (right), solved in cluster ACME. Matrix size is for 1D: n=1,000,000,000; 2D: 1,000,014,129; 3D: 1,073,741,824.

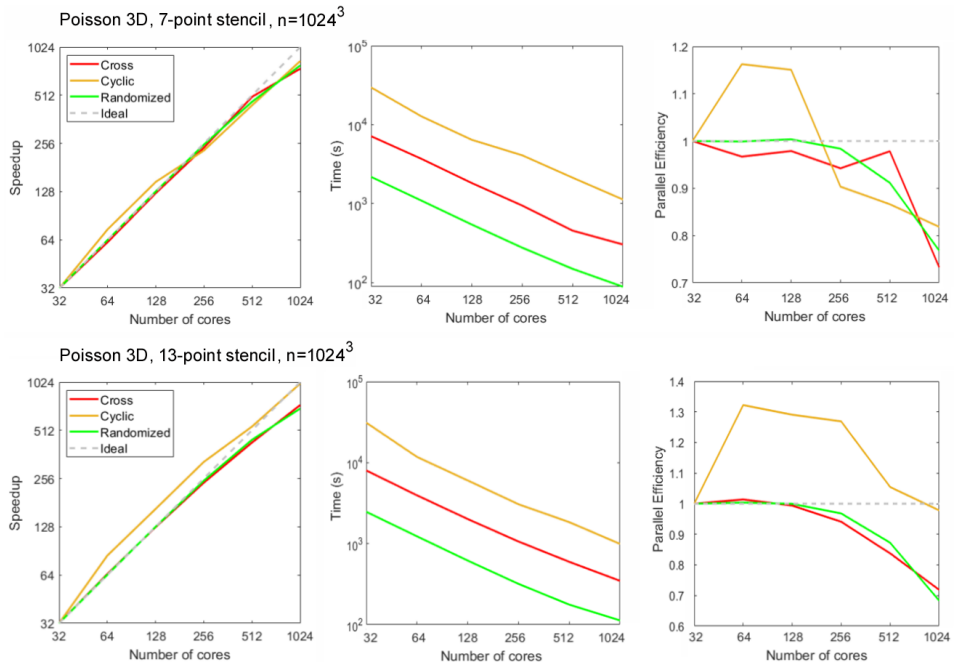

Figure 9: Speedup, execution time and parallel efficiency attained with three SVD algorithms applied to the 3D Poisson matrix computed in partition Xula3 with 32 dedicated nodes. Matrix size is n=1,073,741,824 and discretization has been done with a 7-point (upper row) and 13-point (lower row) stencil.

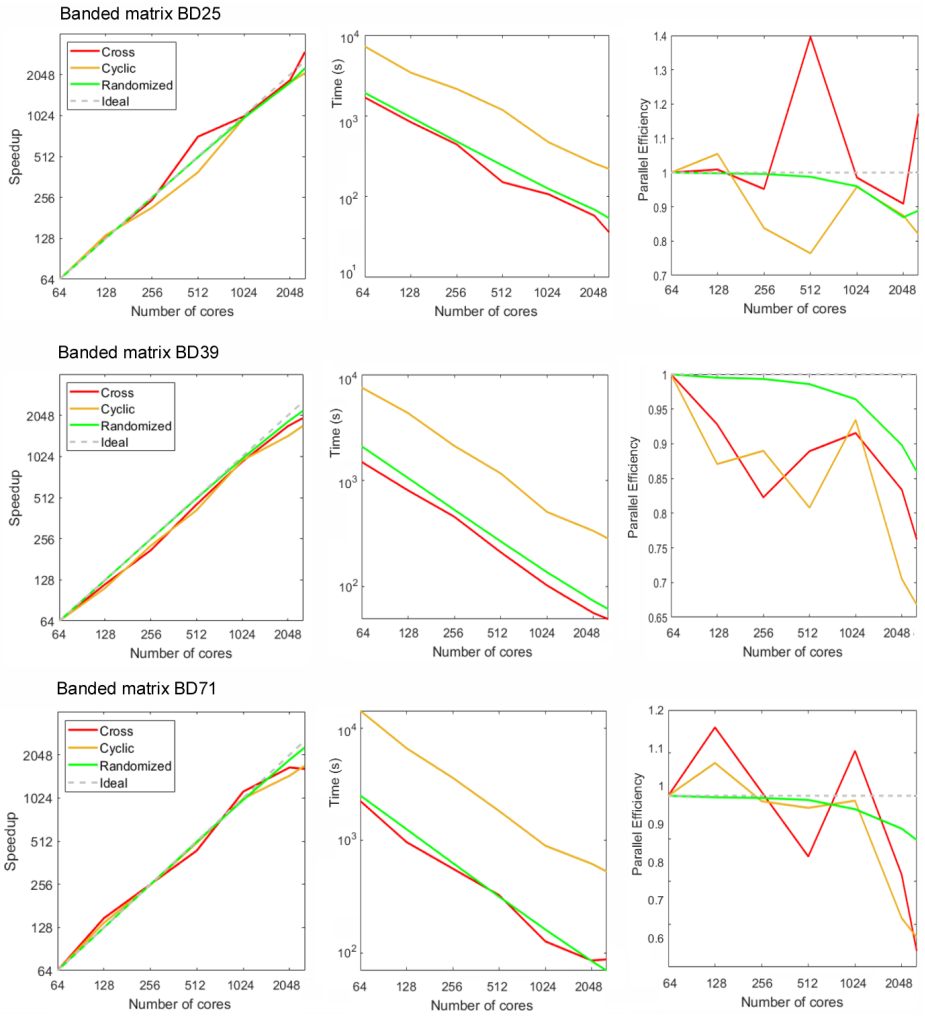

Figure 10: Speedup, execution time and parallel efficiency attained with three SVD algorithms applied to the banded matrices of 25, 39 and 71 diagonals filled with Gaussian numbers (BD25, BD39 and BD71, respectively, $n = 1{,}000{,}000{,}000$), computed in partition Xula3 with 64 dedicated nodes.

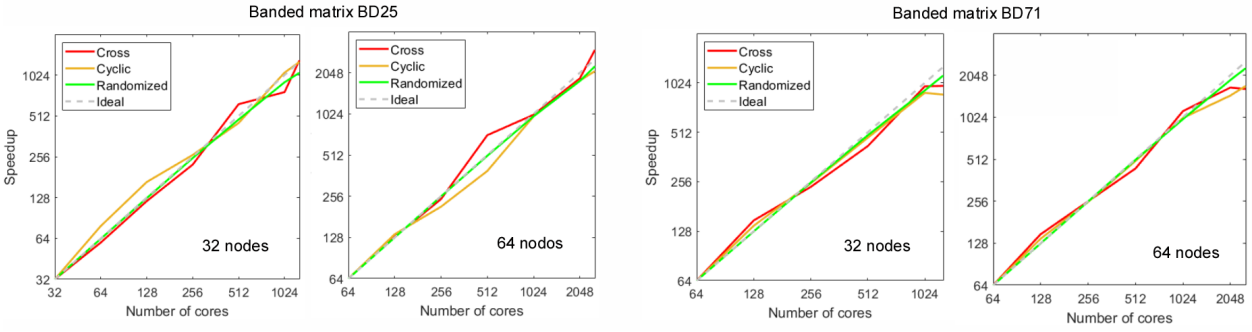

Figure 11: Speedup variation when distributing the MPI ranks over 32 and 64 nodes of Xula3, shown for the banded matrices BD25 (left) and BD71 (right).

Blanchard, P., M. Zounon, J. Dongarra, and N. Higham. 2019. "Novel SVD Algorithm - Deliverable D2.9". Technical Report H2020-FETHPC-2014:GA671633.

Buluc, A., T. Kolda, S. Wild, M. Anitescu, A. Degennaro, J. Jakeman, , , , , , , , , , , , *et al*. 2021, July. *Randomized Algorithms for Scientific Computing (RASC)* https://doi.org/10.2172/1807223.

Davis, T. A. and Y. Hu. 2011. "The University of Florida Sparse Matrix Collection". *ACM Transactions on Mathematical Software* 38(1):1–25 https://doi.org/10.1145/2049662.2049663.

Dongarra, J., M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov *et al*. 2018. "The Singular Value Decomposition: Anatomy of Optimizing an Algorithm for Extreme Scale". *SIAM Review* 60(4):808–865 https://doi.org/10.1137/17M1117732.

Fasi, M. and N. J. Higham. 2021. "Generating Extreme-Scale Matrices With Specified Singular Values or Condition Number". *SIAM Journal on Scientific Computing* 43(1):A663–A684 https://doi.org/10.1137/20M1327938.

Ferrero-Roza, P., J. A. Moríñigo, and F. Terragni. 2023. "Strong Scaling of the SVD Algorithm for HPC Science: A PETSc-Based Approach". In *Proceedings of the Winter Simulation Conference*, WSC '23, 2872–2883: IEEE Press.

Halko, N., P.-G. Martinsson, and J. A. Tropp. 2009. "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions". *ArXiv* https://doi.org/10.48550/ARXIV.0909.4061.

Hernández, V., J. E. Román, and A. Tomás. 2008. "A Robust and Efficient Parallel SVD Solver Based on Restarted Lanczos Bidiagonalization". *ETNA. Electronic Transactions on Numerical Analysis [electronic only]* 31:68–85.

Martinsson, P. 2018. *Randomized Methods for Matrix Computations*, 187–230. American Mathematical Society.

Mills, R. T., M. F. Adams, S. Balay, J. Brown, A. Dener, M. Knepley, , , , , , , *et al*. 2021. "Toward Performance-portable PETSc for GPU-based Exascale Systems". *Parallel Computing* 108:102831 https://doi.org/https://doi.org/10.1016/j.parco.2021.102831.

Moríñigo, J. A., A. Bustos, and R. Mayo-García. 2022. "Error Resilience of Three GMRES Implementations under Fault Injection". *J. Supercomputing* 78(5):7158–7185 https://doi.org/10.1007/s11227-021-04148-x.

Rodríguez-Pascual, M., J. A. Moríñigo, and R. Mayo-García. 2019. "Effect of MPI Tasks Location on Cluster Throughput Using NAS". *Cluster Computing* 22(4):1187–1198 https://doi.org/10.1007/s10586-018-02898-7.

Román, J. E., C. Campos, L. Dalcin, E. Romero and A. Tomas. 2022. "SLEPc Users Manual". Technical Report DSIC-II/24/02.

Schmidt, D. 2020. "A Survey of Singular Value Decomposition Methods for Distributed Tall/Skinny Data". In *2020 IEEE/ACM 11th Workshop ScalA*, 27–34: IEEE Computer Society https://doi.org/10.1109/ScalA51936.2020.00009.

Trefethen, L. N. and D. Bau. 1997. *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics.

AUTHOR BIOGRAPHIES

MARÍA DE CASTRO-SÁNCHEZ earned her MSc in Industrial Mathematics at the Technical University of Madrid (Spain) and holds a degree in Aerospace Engineering. Her interests are linked to Mathematics and its application to problem solving in industry. Recently she conducted applied research in high performance computing in the context of her MSc thesis, carried out in the Department of Technology of CIEMAT. Her email address is mariadecastro2000@gmail.com.

JOSÉ A. MORÍÑIGO is a senior researcher at the Department of Technology of the Centre for Energy, Environmental and Technological Research (CIEMAT). He earned his PhD in Aeronautical Engineering from Universidad Politécnica de Madrid (2004). His research interests include the development of scalable, resilient solvers of PDEs and numerical algebra for supercomputers and their application to fluid dynamics. He has a long experience in simulation of turbulent compressible flow, mostly applied to aerospace propulsion. He is a lecturer of space propulsion at the School of Industrial Engineering of Bilbao. His e-mail is josea.morinigo@ciemat.es and his research group website is http://rdgroups.ciemat.es/web/sci-track.

FILIPPO TERRAGNI is an associate professor in Applied Mathematics at the Mathematics Department of the Universidad Carlos III de Madrid. His research interests include reduced order models based on modal expansions and data-processing for problems involving fluid dynamics, pattern-forming systems, and transport phenomena. On the other hand, he is also working on modeling and numerical simulation of biological processes, like the angiogenesis. He is a lecturer in the Interuniversity Master in Industrial Mathematics (Spain). His email is fterragn@ing.uc3m.es and his research group website is https://scala.uc3m.es/.

RAFAEL MAYO-GARCÍA is a senior researcher at CIEMAT, Harvard University Fellow, and coordinator of the European EERA Joint Programme 'Digitalisation for Energy'. He earned his PhD in Physics from Universidad Complutense de Madrid (2004). He has been involved in many experiments in the US, Bulgaria, Sweden, and Ireland (funded, among others, by the European Commission with a Marie Curie Action). He has also obtained a postdoctoral fellowship in the Spanish Juan de la Cierva Programme. He authored more than 160 scientific articles. He has participated in 64 projects (being PI in 9 out of them) and has been involved in several European and National initiatives working on HPC & BD scientific developments. He also has served several institutions as an evaluator for their competitive Calls, European Commission included and has supervised 8 theses. His email is rafael.mayo@ciemat.es.