

MUMPS: Multifrontal Massively Parallel Solver for the direct solution of sparse linear equations

Patrick R. Amestoy, Jean-Yves L'Excellent, Theo Mary, and Chiara Puglisi

Abstract. The scientific library MUMPS (MULTifrontal Massively Parallel Solver) solves large systems of sparse linear equations using a direct method. It is an open-source software library distributed under the [CeCILL-C](#) license which can be downloaded from the [MUMPS solver](#) website.

Direct methods are widely appreciated for their numerical robustness, reliability, and ease of use. They are commonly used either to compute the solution or as a preconditioner within iterative methods. However, they are also known for their high computational complexity.

In this communication, we briefly describe the main features of the MUMPS solver. We then explain how data sparsity with block low-rank (BLR) approximations and mixed precision can be used to reduce the complexity of direct methods. We conclude by showing results on large real-world applications (matrices of order several hundred million) and describing ongoing research work.

1. Introduction

In many areas of numerical simulation (such as geophysics, structural mechanics, economy, chemistry, and health), the solution of sparse linear systems is a critical and expensive part in terms of calculation time and memory footprint. In fact, the initial problem, often non-linear, is commonly approximated by a succession of linear problems which, depending on the formulation and the modeling carried out, can be excessively difficult to solve numerically. Actual problems provided by our users and industrial customers, whether they come from the study of the aging of nuclear power plants, the study of noise inside the passenger compartment of a car, or the study of the deformation and wear of a tire, are numerically challenging. It should be noted

Mathematics Subject Classification 2020: 65F05 (primary); 65F50, 65F55, 65Y05 (secondary).

Keywords: sparse linear algebra, direct methods, low-rank approximation, mixed-precision computation, parallel computation.

that a numerically difficult problem will often lead to longer calculation times, higher memory usage, and significant power consumption.

Today, applications require solving linear systems of increasingly large sizes (several hundred million unknowns), allowing complex problems to be addressed while maintaining performance and numerical stability. In this context, it is necessary to design linear algebra libraries that know how to exploit the computers, regardless of the size of the problem, the architecture, and the power of the computer (from a laptop with a few cores to supercomputers with hundreds of thousands of heterogeneous computing units).

The MUMPS library solves large sparse linear systems of equations, $AX = B$, using direct methods based on the multifrontal method [22, 23]. The multifrontal method [10] organizes the operations that take place during the factorization of sparse matrices in such a way that the entire factorization is performed through partial factorizations of a sequence of dense and small submatrices (*frontal matrices*). It is guided by a tree, the so-called *elimination tree* [32], that represents the dependencies between those partial factorizations.

The matrix A is a sparse square matrix, and X and B are vectors or matrices whose sparsity can also be exploited. Please note that the solution X is, in general, not sparse even if A and B are sparse. The sparsity of X comes from the fact that in many applications, the user is only interested in part of the solution. There are two main classes of methods to solve such problems:

- *direct methods* [21], based on a factorization of A (see [27]), for example, in the form LU , LDL^T , where L and U are (respectively, lower and upper) triangular matrices and D is a diagonal (or block diagonal) matrix when Gaussian elimination is used to factor the matrix, or in the form QR , where Q is an orthogonal matrix and R an upper triangular matrix in case of orthogonal factorization;
- *iterative methods*, which build a series of iterates converging to the solution and often rely on a preconditioner for which a direct method may also play a role.

Direct methods, in general, are more costly (in terms of number of operations and memory usage) than iterative methods but are more numerically robust.

Among direct methods, the multifrontal methods compute a sequence of partial factorizations of smaller dense frontal matrices following the elimination tree. Multifrontal methods, like many direct methods, use an approach in three phases to solve $AX = B$: analysis, factorization, and solve.

- (1) *Analysis*: The undirected graph associated with the matrix A is analyzed. The undirected graph of A is defined on the symmetrized matrix $A + A^T$; it has as many nodes as columns in the matrix and an edge between nodes i and j if and only if a_{ij} is non-zero. A permutation (*ordering*) that reduces the number of operations during the factorization is calculated. The analysis also

predicts the data structures for the factorization. This requires carrying out a symbolic factorization to determine the frontal matrices, to calculate the task dependency tree, and to partially assign tasks to processors.

- (2) Factorization: The permuted matrix is factorized in the form LU or LDL^T , depending on the properties of matrix A . Numerical pivoting based on partial pivoting is commonly used (see [29, Theorem 9.5]) to preserve the numerical stability of LU factorization. Note that due to partial pivoting, the data structures predicted during the analysis can be modified during the factorization phase. The factorization is generally the most computationally intensive part.
- (3) Solve: Triangular systems are solved to compute a solution. Post-processing can also be applied, such as iterative refinement and/or error analysis.

The MUMPS library implements robust and efficient parallel algorithms, taking into account the following objectives and constraints:

- (1) Matrix factorizations are more stable when numerical pivoting is allowed. Efficient implementations are challenging in a parallel distributed environment. One difficulty with sparse matrices is that numerical pivoting results in a modification during the factorization phase of the task dependency graph computed during the analysis phase. Furthermore, the stability of a pivot must be checked against other entries in the matrix, which is not trivial because it often requires costly interprocessor communications.
- (2) Large-scale high-performance computers generally have their memory physically distributed. Therefore, the message-passing paradigm (MPI) is used in combination with the shared-memory paradigm (OpenMP) to address efficiently computing nodes that share memory between a large number of cores. Exploiting GPU accelerators is an ongoing work.
- (3) Architecture evolutions make it very hard or even impossible to perfectly predict the behavior of current computing platforms, with several levels of caches, high-performance redundant networks with a complex topology, some degree of heterogeneity between processors or at the network level, varying load in possibly multi-user environments, etc. Furthermore, the characteristics of the graph of tasks used in MUMPS are not fully predictable when numerical pivoting is allowed (see previous point). Therefore, adaptive approaches are used, with so-called *dynamic scheduling*. Static information is used to reduce the space to be explored by the dynamic schedulers.
- (4) There are often time constraints in numerical simulation. Therefore, it is critical that the algorithms reduce the time for solution (or *makespan*) as much as possible on a given number of processors.

- (5) If the problem is large, special attention must be paid to memory which, if not correctly used, is a very strong limiting factor for parallel direct methods. The order in which the tasks are scheduled has a strong impact not only on the makespan but also on memory, and unfortunately, more parallelism often leads to more global memory used. Therefore, it is crucial to take these two conflicting criteria into account when designing scheduling strategies for the computational tasks.

With these objectives in mind, in the MUMPS solver, a fully asynchronous and distributed scheduling approach [13] to process the graph of tasks during the factorization has been designed, and several types of pivoting strategies have been implemented.

In recent years, the MUMPS group has developed a new low-rank method called block low-rank (BLR) [1, 11] to exploit data sparsity. For a wide range of problems arising from elliptical PDEs, we have proved in [8] that we can reduce the complexity of sparse solvers in terms of the number of floating-point operations and memory usage of the numerical phases and in particular of the factorization phase. In [30], the BLR method was proved to be backward stable. This means that the computed solution is the exact solution of a nearby problem with a small perturbation of the input data (in our case, the matrix A and the right-hand side B). We will describe in Section 2 the low-rank approximation approaches, introducing the block low-rank method.

Moreover, low-rank approximations exhibit remarkable properties that allow for the use of mixed-precision arithmetic without loss of accuracy. For any unitarily invariant norm, the optimal low-rank approximation is the truncated singular value decomposition (SVD), a result known as the Eckart–Young theorem [24]. During BLR compression, an SVD decomposition of the block to be compressed is computed, and we have shown in [2] that the singular vectors associated with small singular values can be safely stored in lower precisions while preserving high accuracy overall. This will be described in Section 3.

In Section 4, we present some promising results on real-world applications combining BLR compression and mixed-precision algorithms. Tracks for future research are mentioned as concluding remarks in Section 5.

2. Low-rank approximations

Motivations. In many applications (such as those coming from the discretization of elliptic partial differential equations), the matrix A has been shown to have a low-rank property: conveniently defined off-diagonal blocks of its Schur complements can be approximated by low-rank products [16]. Several formats have been

proposed to exploit this property, mainly differing on whether they use a strong or weak admissibility condition, whether they have a nested basis property, and whether they use a hierarchy of levels. The most general of the hierarchical formats is the $\mathcal{H}$ -matrix format [16, 17, 28], which is non-nested and strongly admissible. The $\mathcal{H}^2$ -matrix format [17] is its nested counterpart. HODLR matrices [14] are based on the weak admissibility condition, and HSS [19, 37] and the closely related HBS [26] additionally possess nested bases. These low-rank formats can be exploited within dense or sparse solvers to provide a substantial reduction of their complexity, which is quasilinear even for dense solvers.

In the MUMPS solver, we exploit a different low-rank format: the block low-rank (BLR) format [1]. Unlike hierarchical formats, the BLR format is based on a flat, non-hierarchical blocking of the matrix. We have shown in [8] that despite being non-hierarchical, the BLR format can still achieve a substantial complexity reduction. Moreover, when applied to sparse solvers, the complexity reductions from exploiting sparsity of the matrix A and data sparsity thanks to BLR approximations compound, leading to complexities as low as $O(n \log n)$ for storage and $O(n^{4/3})$ for floating-point operations (for 3D problems with constant ranks), which are not so far from the $O(n)$ hierarchical ones. Moreover, the constant prefactor hidden in the $O(\cdot)$ notation is higher for hierarchical formats, and hierarchical formats can also be more complex to implement, especially in parallel, fully featured, general-purpose sparse solvers. For these reasons, the BLR format has been preferred in most sparse direct solvers, including MUMPS and PaStiX [36]; STRUMPACK [25] adopts a composite approach [20] which combines the BLR format with hierarchical ones.

The block low-rank format. A block low-rank (BLR) representation $\tilde{A}$ of a dense matrix A has the $p \times p$ block form

$$\tilde{A} = \begin{bmatrix} A_{11} & \tilde{A}_{12} & \cdots & \tilde{A}_{1p} \\ \tilde{A}_{21} & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ \tilde{A}_{p1} & \cdots & \cdots & A_{pp} \end{bmatrix}, \quad (2.1)$$

where off-diagonal blocks A_{ij} of size $n_i \times n_j$ are approximated by low-rank matrices $\tilde{A}_{ij} = X_{ij} Y_{ij}^T$ of rank k_{ij} , where $X_{ij} \in \mathbb{R}^{n_i \times k_{ij}}$ and $Y_{ij} \in \mathbb{R}^{n_j \times k_{ij}}$. If the ranks k_{ij} are chosen such that $\|\tilde{A}_{ij} - A_{ij}\| \leq \varepsilon \|A\|$, we obtain a BLR approximation satisfying a relative accuracy of order ε , $\|\tilde{A} - A\| \leq c\varepsilon \|A\|$, where c is a constant that depends on the choice of norm (for example, for the Frobenius norm, $c = p$). If the rank k_{ij} is small enough so that $k_{ij}(n_i + n_j) < n_i n_j$, storing the low-rank factors X_{ij} and Y_{ij} requires fewer entries than storing the full block A_{ij} , and A_{ij} is referred to as a low-rank block. Otherwise, it is more efficient to keep A_{ij} as is, and it is referred to as

a full-rank block. The BLR representation is not only useful for reducing storage; it can also be exploited to reduce the cost of computations, in particular that of LU factorization. As explained above and proved in [8], BLR compression allows for an asymptotic complexity reduction, which means that the larger the matrix, the greater the storage gain.

The $\tilde{A}_{ij}$ approximation of each block can be computed in different ways, such as an SVD computation or a truncated QR orthogonal factorization with column pivoting [18]. A truncated QR corresponds to a QR factorization with pivoting which is truncated as soon as the norm of the trailing submatrix falls below the prescribed threshold $\varepsilon\|A\|$.

This ε parameter, called the low-rank threshold, defines the accuracy of the low-rank approximations. An important question is how the use of BLR compression affects the stability of LU factorization. Indeed, while the threshold ε straightforwardly controls the blockwise truncation errors, the question of how these errors propagate within the LU factorization is far from obvious. It has been proved in [30] that the ε parameter can be used to control the backward error of the final solution. Hence the BLR LU factorization is backward stable with respect to ε .

Because the multifrontal method relies on dense factorizations, the BLR approximations can be easily incorporated into the multifrontal factorization by representing the frontal matrices as BLR matrices as defined by equation (2.1).

High-performance implementation. While the reduced complexity of BLR LU factorization should in principle lead to a significantly faster execution time, translating these theoretical gains into actual speedups is quite challenging, especially in a parallel computing environment. One of the main reasons is that BLR computations tend to be memory (or communication) bound. Indeed, while BLR approximations reduce both flops and memory costs, the former are reduced by a much larger factor than the latter. Moreover, BLR approximations not only require partitioning the matrix into smaller blocks but also replace these blocks by low-rank products of even smaller dimensions. The granularity, i.e., the size of a task in terms of number of floating-point operations, of BLR computational tasks is therefore radically smaller than that of standard dense linear algebra.

In order to achieve efficiency, it is therefore necessary to rethink standard algorithms to reduce the amount of data movement or communications. MUMPS is based on such *communication-avoiding* implementations of the BLR LU factorization described in [11]. They are based on using a specific order of computations (left-looking factorization) that minimizes accesses to uncompressed data and on grouping low-rank blocks in order to increase their granularity and hence the compute intensity of the factorization. Our performance experiments with MUMPS presented in [11] confirm that these optimizations lead to significant speedups. In a more recent work [3],

we have also extended these ideas to accelerate the BLR triangular solution step with many right-hand sides by minimizing accesses to the uncompressed right-hand sides.

3. Mixed-precision computations

The increasing support for lower-precision arithmetics in hardware provides new opportunities for high-performance scientific computing. However, even though low-precision arithmetics can provide significant speed, communication, and energy benefits, its use in scientific computing poses the challenge of preserving the accuracy and stability of the computation.

To address this issue, a variety of mixed-precision algorithms that combine low and high precisions have emerged.

Adaptive-precision BLR. Block low-rank approximations allow for the use of mixed-precision arithmetic without loss of accuracy. During BLR compression, an SVD decomposition of the block to be compressed is computed. We have shown in [2] that the singular vectors associated with small singular values can be safely stored in lower precisions while preserving high accuracy overall. Given p available precision formats with unit roundoffs $u_1 \leq \varepsilon < u_2 < \dots < u_p$, for any off-diagonal low-rank block $\tilde{A}_{ij} = U \Sigma V^T$, this leads to an adaptive-precision representation of the form

$$\hat{A}_{ij} = [\hat{U}_1 \cdots \hat{U}_p] \begin{bmatrix} \hat{\Sigma}_1 & & \\ & \ddots & \\ & & \hat{\Sigma}_p \end{bmatrix} [\hat{V}_1 \cdots \hat{V}_p]^T,$$

where $\hat{U}_k$, $\hat{\Sigma}_k$, and $\hat{V}_k$ are stored in precision u_k . According to the theory developed in [2], the singular values should be partitioned such that $\|\Sigma_k\| \leq \varepsilon \|A\|/u_k$. With this criterion, we have $\|\hat{A}_{ij} - \tilde{A}_{ij}\| \leq (2p - 1)\varepsilon \|A\|$, and so a preserved accuracy of order ε is guaranteed.

For typical BLR matrices, the singular values of the low-rank blocks decay rapidly. As a result, a large part of the coefficients and operations can be switched to lower-precision arithmetics. This leads to significant storage savings as we will illustrate in the next section.

Iterative refinement and MUMPS as a preconditioner. Another possible usage of MUMPS in mixed precision is to compute the LU factorization in a lower precision, use this lower-precision LU factorization to obtain an approximate initial solution, and use an iterative solver in higher precision to refine the accuracy of the solution. In particular, we have shown in [6] that, as long as the problem is not too ill-conditioned, LU-based iterative refinement can deliver full double-precision accuracy even with

a single-precision factorization, reducing the time and memory costs of the solver by a factor of $2\times$. Moreover, this iterative refinement can also be combined with a BLR factorization (with potentially very coarse accuracy ε) for even greater time and memory savings.

In order to handle (potentially very) ill-conditioned problems, LU-based refinement can be replaced by more robust variants based on Krylov iterative solvers such as GMRES [7]. The low-precision LU factors computed by MUMPS can then be used as a robust preconditioner.

4. Numerical results

We have been able to successfully use the mixed-precision block low-rank feature of MUMPS [4, 33] in the context of collaborations with the [WIND project](#) on full-waveform inversion applications (see [33]) and with TotalEnergies-Inria [Makutu](#) on hybridizable discontinuous Galerkin discretization for 3D wave propagation (see [4, 5]).

We focus here on results obtained with full-waveform inversion of the Gorgon model with the WIND project (see [33, 34]).

High-resolution imaging has many applications in civil engineering, oil exploration, earthquake seismology, medical imaging, non-destructive control testing, etc. The so-called full-waveform inversion (FWI) has become the baseline imaging method in many application domains with the development of new acquisition technologies and the continuous advances in high-performance computing. FWI relies on simple principles: FWI aims at estimating the constitutive parameters of a medium contained in the coefficients of the wave equation by minimizing the distance between waves recorded by an array of sensors and the numerically simulated counterparts. As such, it is a partial differential equation (PDE)-constrained optimization problem, which is solved with iterative gradient-based methods. At each iteration, building the gradient of the misfit function with the adjoint-state method requires computing the incident wavefields triggered by each source of the experiment and the so-called adjoint wavefields by propagating the data residuals backward in time from the sensors. The wave equation is classically solved with numerical methods such as finite difference or finite element schemes to exploit the full information content in the data. Moreover, the wave equation can be solved either in the time-space or in the frequency-space domain. In the former case, solving the wave equation is an initial condition evolution problem, which can be solved with explicit (matrix-free) schemes. In the latter case, solving the time-harmonic wave equation for each source of the experiment is a boundary value problem requiring the solution of a large and sparse complex-valued linear system with multiple sparse right-hand sides. Operto et al. [35] have developed a frequency-domain full-waveform inversion (FWI) for

LU size (TBytes)			Flops		Time (seconds)		Scaled Residual
FR	BLR	+Mixed	FR	BLR+Mixed	Analysis +Factorization	Solve	BLR+Mixed
73	34	26	2.6×10^{18}	0.5×10^{18}	5946	27	7×10^{-4}

Table 1. Results for the Gorgon model, reservoir 23 km × 11 km × 6.5 km, grid size 15 m, Helmholtz equation, 25 Hz: FR (full-rank) with single-precision fp32; BLR (block low-rank) with $\varepsilon_{\text{BLR}} = 10^{-5}$; mixed-precision BLR: 3 precisions (32 bits, 24 bits, 16 bits) for storage using 48,000 cores (500 MPI × 96 threads/MPI). Solve is time to compute the solution with one right-hand side.

geophysical applications to image a geological medium containing several tens to hundreds of millions of unknowns after discretization, while the seismic experiment involves hundreds to thousands of seismic sources (right-hand sides). They implement FWI in the frequency domain, where the Helmholtz equation is solved with a finite difference method and the MUMPS solver.

In [33], we show how the memory and computational time of the LU factorization can be mitigated with adaptive-precision BLR approximations. In Table 1, the performance of MUMPS is illustrated with an industrial seabed case study from the Northwest Shelf of Western Australia: the Gorgon gas field (see [34]). The inversion is designed as a hierarchical multiscale approach proceeding across increasing frequencies between 1.7 Hz and 25 Hz. For each frequency, the discretization of the computational domain is matched to the frequency to optimize the size of the linear system to solve. The largest matrix to be processed at 25 Hz contains 531 million degrees of freedom. The number of floating-point operations (flops) for a full-rank (FR) LU factorization is 2.6×10^{18} , and the estimated memory necessary for the LU factors is equal to 73 TBytes. We remark that the FR factorization could not be carried out due to the large memory necessary for the computations. The results show that the use of BLR approximations reduces the size of the LU factors by a factor of about 2×, and using mixed precision further increases this factor to 3×. These results have been achieved on the supercomputer Adastra at CINES (Montpellier, France) on 48,000 cores (500 MPI × 96 threads/MPI). It is interesting to mention that since hundreds to thousands of right-hand sides need to be processed, the FWI simulation will be dominated by the time for solution which becomes more critical than the time for factorization.

5. Concluding remarks

A notable feature of the adaptive-precision BLR representation is that it can exploit an arbitrary number of precision formats. In the current version of MUMPS, we use seven

precision formats – including custom formats with truncated mantissa – that maximize the storage reduction. Additionally, blocks in BLR fronts that are not admissible to BLR compression may be stored in their full-rank format in low precision if their norm is small. In order to maintain efficiency with these custom formats which can be critical for the solution phase, we have developed a memory accessor approach [9] that implements an efficient, lightweight conversion from custom to standard formats. The efficiency of the approach relies on cache-aware mechanisms. When a computation is executed using (non-standard) low-precision operands, the operation is performed if such an operand and a working-precision copy of it can coexist in the cache. If this is not possible, the operation is split up into smaller blocks.

In the last few years, nodes of parallel computers have become very heterogeneous, leading to extremely powerful nodes equipped with multicore CPUs together with accelerators such as GPUs. Using accelerators efficiently, in the context of a general-purpose sparse direct solver, is very challenging. Our approach is based on offloading BLAS-3 kernels using XKBlas [31] from Inria. In collaboration with Inria, we are studying computational nodes with unified memory platforms. The unified memory architectures significantly reduce memory transfer overheads between the CPU and the GPU, enabling efficient offloading of smaller BLAS-3 kernels to the GPU and offering great opportunities to cover a wider range of real-world applications.

As mentioned, the asymptotic complexity of the flat BLR format, although already quite low, is however not optimal. Some more compact alternative formats, achieving intermediate complexities between BLR and fully hierarchical formats, have been theoretically proposed: the BLR² [15] and the multilevel BLR (MBLR) [12] formats. In collaboration with EDF (Electricité de France) company, we are currently assessing their practical potential for further improving the complexity of MUMPS.

Acknowledgments. The MUMPS project is a joint effort by several researchers and institutions (CERFACS, CNRS, ENS Lyon, Toulouse INP, Inria, Sorbonne University, and the University of Bordeaux) spanning more than 25 years of research and development. Today, the MUMPS solver is developed and maintained by Mumps Technologies, an enterprise founded in 2019. We are grateful to Stéphane Operto for his careful proofreading of the section on full-waveform inversion.

Funding. This work was carried out in the context of two PhD theses supported by EDF and is supported by a consortium of industrial and academic users.

References

- [1] P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker, [Improving multifrontal methods by means of block low-rank representations](#). *SIAM J. Sci. Comput.* **37** (2015), no. 3, A1451–A1474 Zbl [1314.05111](#) MR [3355774](#)
- [2] P. Amestoy, O. Boiteau, A. Buttari, M. Gerest, F. Jézéquel, J.-Y. L'Excellent, and T. Mary, [Mixed precision low-rank approximations and their application to block low-rank LU factorization](#). *IMA J. Numer. Anal.* **43** (2023), no. 4, 2198–2227 Zbl [1525.65038](#) MR [4621842](#)
- [3] P. Amestoy, O. Boiteau, A. Buttari, M. Gerest, F. Jézéquel, J.-Y. L'Excellent, and T. Mary, [Communication avoiding block low-rank parallel multifrontal triangular solve with many right-hand sides](#). *SIAM J. Matrix Anal. Appl.* **45** (2024), no. 1, 148–166 Zbl [1535.65052](#) MR [4688196](#)
- [4] P. Amestoy, A. Buttari, F. Faucher, M. Gerest, J.-Y. L'Excellent, and T. Mary, Mixed precision sparse direct solver applied to 3D wave propagation. In *Proceedings of the 8th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2022), Oslo, Norway, 2022*
- [5] P. Amestoy, A. Buttari, F. Faucher, J.-Y. L'Excellent, and T. Mary, Application of the sparse direct solver MUMPS to Hybridizable Discontinuous Galerkin (HDG) discretization for wave modeling. In *ECCOMAS Congress 2024, Lisbon, Portugal, 2024*
- [6] P. Amestoy, A. Buttari, N. J. Higham, J.-Y. L'Excellent, T. Mary, and B. Vieublé, [Combining sparse approximate factorizations with mixed-precision iterative refinement](#). *ACM Trans. Math. Software* **49** (2023), no. 1, article no. 4, 29 pp. Zbl [07908566](#) MR [4567890](#)
- [7] P. Amestoy, A. Buttari, N. J. Higham, J.-Y. L'Excellent, T. Mary, and B. Vieublé, [Five-precision GMRES-based iterative refinement](#). *SIAM J. Matrix Anal. Appl.* **45** (2024), no. 1, 529–552 Zbl [1545.65124](#) MR [4704629](#)
- [8] P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary, [On the complexity of the block low-rank multifrontal factorization](#). *SIAM J. Sci. Comput.* **39** (2017), no. 4, A1710–A1740 Zbl [1372.65089](#) MR [3691219](#)
- [9] P. Amestoy, A. Jegou, J.-Y. L'Excellent, T. Mary, and G. Pichon, [BLAS-based mixed precision block memory accessor with applications to sparse direct solvers](#). *ACM Trans. Math. Softw.* **52** (2026), no. 2, article no. 10, 29 pp.
- [10] P. Amestoy, A. Buttari, I. Duff, A. Guermouche, J.-Y. L'Excellent, and B. Uçar, [Multifrontal method](#). In D. Padua (ed.), *Encyclopedia of parallel computing*, pp. 1209–1216, Springer, 2011
- [11] P. R. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary, [Performance and scalability of the block low-rank multifrontal factorization on multicore architectures](#). *ACM Trans. Math. Software* **45** (2019), no. 1, article no. 2, 26 pp. Zbl [1471.65025](#) MR [3939092](#)
- [12] P. R. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. A. Mary, [Bridging the gap between flat and hierarchical low-rank matrix formats: the multilevel block low-rank format](#). *SIAM J. Sci. Comput.* **41** (2019), no. 3, A1414–A1442 Zbl [1434.65051](#) MR [3945811](#)

- [13] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster, [A fully asynchronous multifrontal solver using distributed dynamic scheduling](#). *SIAM J. Matrix Anal. Appl.* **23** (2001), no. 1, 15–41 Zbl 0992.65018 MR 1856597
- [14] A. Aminfar, S. Ambikasaran, and E. Darve, [A fast block low-rank dense solver with applications to finite-element matrices](#). *J. Comput. Phys.* **304** (2016), 170–188 Zbl 1349.65595 MR 3422408
- [15] C. Ashcraft, A. Buttari, and T. Mary, [Block low-rank matrices with shared bases: potential and limitations of the BLR² format](#). *SIAM J. Matrix Anal. Appl.* **42** (2021), no. 2, 990–1010 Zbl 07365307 MR 4275498
- [16] M. Bebendorf, [Hierarchical matrices. A means to efficiently solve elliptic boundary value problems](#). Lect. Notes Comput. Sci. Eng. 63, Springer, Berlin, 2008, 290 pp. Zbl 1151.65090 MR 2451321
- [17] S. Börm, L. Grasedyck, and W. Hackbusch, [Introduction to hierarchical matrices with applications](#). *Eng. Anal. Bound. Elem.* **27** (2003), no. 5, 405–422 Zbl 1035.65042
- [18] P. Businger and G. H. Golub, [Linear least squares solutions by householder transformations](#). *Numer. Math.* **7** (1965), no. 3, 269–276 Zbl 0142.11503 MR 0176590
- [19] S. Chandrasekaran, M. Gu, and T. Pals, [A fast ULV decomposition solver for hierarchically semiseparable representations](#). *SIAM J. Matrix Anal. Appl.* **28** (2006), no. 3, 603–622 Zbl 1120.65031 MR 2262971
- [20] L. Claus, P. Ghysels, Y. Liu, T. A. Nhan, R. Thirumalaisamy, A. P. S. Bhalla, and S. Li, [Sparse approximate multifrontal factorization with composite compression methods](#). *ACM Trans. Math. Software* **49** (2023), no. 3, article no. 24, 28 pp. Zbl 07910039 MR 4658761
- [21] I. S. Duff, A. M. Erisman, and J. K. Reid, [Direct methods for sparse matrices](#). Monographs on Numerical Analysis, The Clarendon Press, Oxford University Press, New York, 1986, 341 pp. Zbl 0604.65011 MR 0892734
- [22] I. S. Duff and J. K. Reid, [The multifrontal solution of indefinite sparse symmetric linear equations](#). *ACM Trans. Math. Software* **9** (1983), no. 3, 302–325 Zbl 0515.65022 MR 0791968
- [23] I. S. Duff and J. K. Reid, [The multifrontal solution of unsymmetric sets of linear equations](#). *SIAM J. Sci. Statist. Comput.* **5** (1984), no. 3, 633–641 Zbl 0557.65017 MR 0754490
- [24] C. Eckard and G. Young, [The approximation of one matrix by another of lower rank](#). *Psychometrika* **1** (1936), no. 3, 211–218 JFM 62.1075.02
- [25] P. Ghysels, X. S. Li, F.-H. Rouet, S. Williams, and A. Napov, [An efficient multicore implementation of a novel HSS-structured multifrontal solver using randomized sampling](#). *SIAM J. Sci. Comput.* **38** (2016), no. 5, S358–S384 Zbl 1352.65092 MR 3565567
- [26] A. Gillman, P. M. Young, and P.-G. Martinsson, [A direct solver with \$O\(N\)\$ complexity for integral equations on one-dimensional domains](#). *Front. Math. China* **7** (2012), no. 2, 217–247 Zbl 1262.65198 MR 2897703

- [27] G. H. Golub and C. F. Van Loan, *Matrix computations*. 4th edn., Johns Hopkins Stud. Hist. Math., Johns Hopkins University Press, Baltimore, MD, 2013, 756 pp.
Zbl [1268.65037](#) MR [3024913](#)
- [28] W. Hackbusch, [A sparse matrix arithmetic based on \$\mathcal{H}\$ -matrices. Part I: Introduction to \$\mathcal{H}\$ -matrices](#). *Computing* **62** (1999), no. 2, 89–108 Zbl [0927.65063](#) MR [1694265](#)
- [29] N. J. Higham, *Accuracy and stability of numerical algorithms*. 2nd edn., Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002, 680 pp.
Zbl [1011.65010](#) MR [1927606](#)
- [30] N. J. Higham and T. Mary, [Solving block low-rank linear systems by LU factorization is numerically stable](#). *IMA J. Numer. Anal.* **42** (2022), no. 2, 951–980 Zbl [1511.65024](#) MR [4410733](#)
- [31] J. V. F. Lima, T. Gautier, V. Danjean, B. Raffin, and N. Maillard, [Design and analysis of scheduling strategies for multi-CPU and multi-GPU architectures](#). *Parallel Comput.* **44** (2015), 37–52 MR [3326470](#)
- [32] J. W. H. Liu, [The role of elimination trees in sparse factorization](#). *SIAM J. Matrix Anal. Appl.* **11** (1990), no. 1, 134–172 Zbl [0697.65013](#) MR [1032223](#)
- [33] S. Operto, P. Amestoy, H. Aghamiry, S. Beller, A. Buttari, L. Combe, V. Dolean, M. Gerest, G. Guo, P. Jolivet, J.-Y. L'Excellent, F. Mamfoumbi, T. Mary, C. Puglisi, A. Ribodetti, and P.-H. Tournier, [Is 3D frequency-domain FWI of full-azimuth/long-offset OBN data feasible? The Gorgon data FWI case study](#). *The Leading Edge* **42** (2023), no. 3, 173–183
- [34] S. Operto, P. Amestoy, S. Beller, A. Buttari, L. Combe, F. Faucher, P. Jolivet, J.-Y. L'Excellent, T. Mary, and S. Migeon, [Pushing the limits of 3D frequency-domain FWI with the 2015/2016 OBN Gorgon dataset](#). In *85th EAGE Annual Conference & Exhibition, Oslo*, 10–13 June 2024
- [35] S. Operto, A. Miniussi, R. Brossier, L. Combe, L. Métivier, V. Monteiller, A. Ribodetti, and J. Virieux, [Efficient 3-D frequency-domain mono-parameter full-waveform inversion of ocean-bottom cable data: application to Valhall in the visco-acoustic vertical transverse isotropic approximation](#). *Geophys. Journal Int.* **202** (2015), no. 2, 1362–1391
- [36] G. Pichon, E. Darve, M. Faverge, P. Ramet, and J. Roman, [Sparse supernodal solver using block low-rank compression: design, performance and analysis](#). *J. Comput. Sci.* **27** (2018), 255–270 MR [3834258](#)
- [37] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, [Fast algorithms for hierarchically semiseparable matrices](#). *Numer. Linear Algebra Appl.* **17** (2010), no. 6, 953–976
Zbl [1240.65087](#) MR [2759603](#)

Patrick R. Amestoy

Mumps Technologies SAS, 193 Rue Marcel Mérieux, 69007 Lyon, France;
patrick.amestoy@mumps-tech.com

Jean-Yves L'Excellent

Mumps Technologies SAS, 193 Rue Marcel Mérieux, 69007 Lyon, France;
jean-yves.l.excellent@mumps-tech.com

Theo Mary

Sorbonne Université, CNRS, LIP6, 4 place Jussieu, 75005 Paris, France; theo.mary@lip6.fr

Chiara Puglisi

Mumps Technologies SAS, 193 Rue Marcel Mérieux, 69007 Lyon, France;
chiara.puglisi@mumps-tech.com