Mathematics and Computers in Simulation 229 (2025) 525–538

Contents lists available at ScienceDirect

Mathematics and Computers in Simulation

journal homepage: www.elsevier.com/locate/matcom

Original articles

Analysis and comparison of high-performance computing solvers for minimisation problems in signal processing

Simone Cammarasana [*], Giuseppe Patané

CNR IMATI, Via de Marini 6, Genova, 16149, Italy

ARTICLE INFO

Keywords:
High-performance computing
Minimisation
PRAXIS
Signal processing
Signal approximation and denoising

ABSTRACT

Several physics and engineering applications involve the solution of a minimisation problem to compute an approximation of the input signal. Modern hardware and software use high-performance computing to solve problems and considerably reduce execution time. In this paper, different optimisation methods are compared and analysed for the solution of two classes of non-linear minimisation problems for signal approximation and denoising with different constraints and involving computationally expensive operations, i.e., (i) the global optimisers *divide rectangle-local* and the *improved stochastic ranking evolution strategy*, and (ii) the local optimisers *principal axis*, the *Limited-memory Broyden, Fletcher, Goldfarb, Shanno*, and the *constrained optimisation by linear approximations*. The proposed approximation and denoising minimisation problems are attractive due to their numerical and analytical properties, and their analysis is general enough to be extended to most signal-processing problems. As the main contribution and novelty, our analysis combines an efficient implementation of signal approximation and denoising on arbitrary domains, a comparison of the main optimisation methods and their high-performance computing implementations, and a scalability analysis of the main algebraic operations involved in the solution of the problem, such as the solution of linear systems and singular value decomposition. Our analysis is also general regarding the signal processing problem, variables, constraints (e.g., bounded, non-linear), domains (e.g., structured and unstructured grids, dimensionality), high-performance computing hardware (e.g., cloud computing, homogeneous vs. heterogeneous). Experimental tests are performed on the CINECA Marconi100 cluster at the 26th position in the "*top500*" list and consider several parameters, such as functional computation, convergence, execution time, and scalability. Our experimental tests are discussed on real-case applications, such as the reconstruction of the solution of the fluid flow field equation on an unstructured grid and the denoising of a satellite image affected by speckle noise. The experimental results show that *principal axis* is the best optimiser in terms of minima computation: the efficiency of the approximation is 38% with 256 processes, while the denoising has 46% with 32 processes.

1. Introduction

Motivation. Minimisation methods are widespread for solving various physics and engineering problems. Minimisation problems are also widely applied to signal processing for image analysis [1], 2D videos [2], and graph signal processing [3], with applications in biomedicine [4], astronomy [5], and computer vision [6]. Signal processing requires a significant computational effort and the

[*] Corresponding author.
E-mail address: simone.cammarasana@ge.imati.cnr.it (S. Cammarasana).

https://doi.org/10.1016/j.matcom.2024.10.003
Received 11 April 2024; Received in revised form 5 August 2024; Accepted 4 October 2024
Available online 17 October 2024

real-time constraint [7] requires reducing the execution time. In several application fields (e.g. mechanical, biomedical, digital images), simulation problems often involve the solution of non-linear optimisation problems on complex domains with constrained variables. Solving these optimisation problems requires specialised algorithms and can be computationally and memory intensive. *High-performance computing* (HPC) is relevant in simulations and optimisation problems allowing the users to handle large-scale systems and big data, improve speed and efficiency for real-time processing, and implement advanced algorithms that increase the accuracy and precision of the solution. Through cloud-based HPC, users access modern computing resources on-demand without a significant capital investment in infrastructures.

Problem definition and importance. Given an input signal $f : \Omega \to \mathbb{R}$ defined on a connected and compact domain Ω in $\mathbb{R}^d$, several problems in signal processing are associated with the computation of an approximating function as the solution to the minimisation problem.

$$\begin{cases} \min_{\mu} \|f - \hat{f}(\mu)\|_2^2 + \alpha \mathcal{P}(\hat{f}(\mu)), \\ \qquad\qquad \text{s.t.} \quad g(\mu), \end{cases} \tag{1}$$

where the approximating function $\hat{f}$ depends on a set of variables μ to be optimised; $\mathcal{P}$ is a penalisation operator that may be applied to regularise the approximating function; α is a scalar coefficient; and $g(\cdot)$ is a set of constraints. Starting from the general formulation in Eq. (1), our analysis focuses on two minimisation problems in signal processing that require computationally expensive algebraic operations: (i) an approximation problem, where the functional is non-convex, non-linear, and the derivatives in the analytic form are not available; (ii) a denoising problem, where analytic derivatives are known, with bound constraints. Both these problems are computationally attractive: the availability of the analytical derivatives and the presence of non-linear constraints affect the selection of the optimiser. Furthermore, the properties of each functional (e.g., convexity, algebraic operations) induce a large number of iterations of the minimisation method and, consequently, evaluations of the functional. Our discussion focuses on minimisation problems, as maximisation problems can be addressed similarly.

Recent advances. State-of-the-art optimisers account for several properties, such as global vs local search and easy pre-computation of analytical derivatives. The *constrained optimisation by linear approximations* (COBYLA) [8] is applied for the simulation and prediction of structural and acoustic properties of a geometrical model [9], image segmentation and classification [10], and design of pressure vessel in terms of geometrical properties to minimise material and fabrication cost [11]. The *Limited-memory Broyden, Fletcher, Goldfarb, Shanno method* (L-BFGS) [12] is used to large-scale prediction problems on biomedical images co-registration [13], protein structure [14], and earth surface reconstruction from seismic waveform tomography [15]. The local optimiser *principal axis* (PRAXIS) [16] is applied for the evaluation of image compression [17], finite element modelling in the biomedical ultrasound industry for the design of piezoelectric transducers [18], and estimation of accuracy of dichotomous tests in the psychometric class [19]. The *improved stochastic ranking evolution strategy* (ISRES) [20] is applied for adjusting high energy resolution X-ray beamlines [21] and modelling parameters for determining fatigue crack growth in novel materials [22]. The global optimiser *divide rectangle-local* DIRECT-L [23] is used for modelling metal-fill parasitic capacitance to reduce manufacturing defects of on-chip transmission lines [24] and design analogue integrated circuits [25]. Modern architectures use HPC to solve complex minimisation problems on high-resolution data sets through access to large memory and high computational power. It becomes relevant to analyse the characteristics and performance of HPC hardware (e.g., heterogeneous clusters) and software (e.g., PETSc [26], SLEPc [27]) for solving different minimisation problems in signal processing applications.

Research gaps and contribution. The literature lacks a comprehensive analysis that combines an efficient implementation of signal approximation and filtering problems on arbitrary domains, a comparison of the main optimisation methods and their implementations using HPC techniques, and a scalability analysis of the main algebraic operations involved in the solution of the problem, such as the solution of linear systems and singular value decomposition. In this context, the efficient computation of the main operations drastically reduces the execution time of the minimisation problem. As the main contribution, our work aims to:

- introduce an efficient implementation of the proposed approximation and denoising problems with HPC techniques;
- compare and discuss the properties of convergence, execution time, and scalability of five minimisation methods, i.e., the global optimisers *divide rectangle-local* DIRECT-L [23] and the *improved stochastic ranking evolution strategy* (ISRES) [20], the local optimisers *principal axis* (PRAXIS) [16], the *Limited-memory Broyden, Fletcher, Goldfarb, Shanno* (L-BFGS) [12], and the *constrained optimisation by linear approximations* (COBYLA) [8] for the computation of the optimal solution of the proposed problems;
- present the main results regarding the accuracy and HPC scalability of the approximation and denoising problems. These problems apply the main algebraic operations common to most minimisation problems in signal processing and our analysis is generally applicable to other classes of problems.

Our analysis is general with respect to different signal processing problems and variables, constraints, domains, and HPC hardware configurations. Our comparison supports the users to identify bottlenecks and performance issues in HPC environments for optimisation methods in signal processing and to select the best algorithm and scientific library for their specific needs.

Paper structure. The paper is structured as follows: after discussing the related work in minimisation methods and HPC applications (Section 2), our implementation and analysis of the approximation (Section 3) and denoising (Section 4) problems is presented. Then, the numerical results and the main limitations (Section 5) are presented, with a review of conclusion and future work (Section 6).

2. Related work

The main methods for solving minimisation problems, their computational cost, and the available scientific libraries and hardware are discussed.

Minimisation solvers. DIRECT [28] is a global, derivative-free, and deterministic search algorithm that systematically divides the search domain into smaller hyperrectangles. Rescaling the bound constraints to a hypercube gives equal weight to all dimensions in the search procedure. DIRECT derives from Lipschitzian global optimisation, i.e., a branch-and-bound model, where bounds are computed through the knowledge of a Lipschitz constant for the objective function. DIRECT introduces modifications to the Lipschitzian approach to improve the results in higher dimensions by eliminating the need to know the Lipschitz constant. The global optimiser DIRECT-L [23] is the locally-biased form that enhances the efficiency of functions without too many local minima. As a global method, DIRECT-L spans all the possible solutions without considering local minima as the optimal solution. Furthermore, analytic or numeric derivatives are unnecessary to compute the optimal solution. Finally, DIRECT-L supports unconstrained and linearly-constrained problems.

The *improved stochastic ranking evolution strategy* (ISRES) [20] balances between objective and penalty functions stochastically, i.e., stochastic ranking, by proposing an improved evolutionary algorithm that accounts for evolution strategies and differential variation. The evolution strategy combines a mutation rule (with a log-normal step-size update and exponential smoothing) and differential variation (a Nelder–Mead-like update rule). The fitness ranking is simplified through the objective function for problems without non-linear constraints; if non-linear constraints are included, then the stochastic ranking is employed. ISRES supports unconstrained and constrained problems.

The local optimiser *principal axis* (PRAXIS) [16] is a gradient-free local optimiser that minimises a multivariate function through the *principal-axis* method. PRAXIS is a modification of Powell's direction-set method [29]; given n variables, the set of search directions n is repeatedly updated until a set of conjugate directions with respect to a quadratic form is reached after n iterations. To ensure the correctness of the minimum, the matrix of the search directions is replaced by its principal axes so that the direction set spans the entire parameter space. PRAXIS is designed for unconstrained problems; bound constraints can be applied by considering a penalisation when constraints are violated.

In the family of quasi-Newton methods, the *Limited-memory Broyden, Fletcher, Goldfarb, Shanno* (L-BFGS) [12] is an optimisation algorithm that approximates the *Broyden–Fletcher–Goldfarb–Shanno algorithm* (BFGS) using limited computer memory. Analogously to BFGS, the L-BFGS solver estimates the inverse Hessian matrix for the minimum search in the variable space; however, the L-BFGS method represents the approximation through a few vectors, thus involving a limited memory requirement. At each iteration, a short history of the past updates of the position and the gradient of the energy functional is used to identify the direction of the steepest descent and to implicitly perform operations requiring vector products with the inverse Hessian matrix. Since L-BFGS needs the derivatives of the functional, they are computed through numerical methods (e.g., finite difference method), where they are not available in analytic form. L-BFGS supports both unconstrained and constrained problems.

The *constrained optimisation by linear approximations* (COBYLA) [8] generates successive linear approximations of the objective function and constraints through a simplex of $n + 1$ points in n dimensions and optimises these approximations in a trust region at each step. Each iteration defines linear approximations of the objective and constraint functions by interpolating at the vertices of the simplex, and a trust region bound restricts each change to the variables. A new vector of variables is computed, which may replace one of the current vertices, either to improve the shape of the simplex or because it provides the best results according to a merit function that accounts for the constraint violation. For a deeper review of optimisation methods, the reader may refer to the survey in [30].

Memory storage and computational cost. Given a set of n variables, the L-BFGS method requires a memory storage of $\mathcal{O}(n^2)$ and the computational cost is $\mathcal{O}(nv)$ at each iteration, where v is the number of steps stored in memory. For the PRAXIS method, the computational cost is $\mathcal{O}(n^2)$. For the DIRECT-L method, the worst case computational cost is $\mathcal{O}(2^n)$, even if novel variants of this method propose improved performances. COBYLA has $\mathcal{O}(n^2)$ computational cost. The number of variables for ISRES is $20 \cdot (n + 1)$. Finally, global and local optimisers are applied consecutively: the global optimiser searches for the solution to the minimisation problem in the global parameter space. In contrast, the local optimiser refines the accuracy of the solution.

High-performance computing analysis: related work. A large parallel optimisation solver [31] applies Python to a generic model that can be easily extended with new algorithms and implements NumPy to efficiently account for the computational resources and MPI4py for communication in HPC environments. In [32], an analysis of optimisation software libraries, their parallel implementation, and the application for solving non-linear optimisation problems is presented. In contrast, our work discusses different optimisation problems (i.e., denoising), analyses different optimisers (e.g., PRAXIS), and discusses the performance of the main algebraic operations (e.g., singular value decomposition, solution of a linear system). The interaction between the computational fluid dynamics solver and the reinforcement learning [33] is implemented efficiently on HPC hardware. In contrast, our work focuses on a more generic processing problem, with a scalability analysis on algebraic operations. In [34], the parallelisation of a multi-frontal solver for finite-element meshes is ported to a cloud infrastructure to analyse the communication cost and the efficiency. An efficient implementation of the ant colony optimisation problem [35] applies a hybrid message passing interface (MPI) and OpenMP parallel implementation. Conversely to this combinatorial problem, our method discusses a deterministic optimisation problem with different libraries and minimisation solvers.

Algorithm 1 Constrained least-squares approximation.

```
1:  $\mathbf{f}$ = Input discrete signal
2:  procedure $\hat{\mathbf{f}}$ = APPROXIMATE($\mathbf{f}$)
3:      Mat $\mathbf{\Phi}(\mu)$
4:      Mat $\mathbf{\Phi}_T(\mu) = \mathbf{\Phi}^\top(\mu)$
5:      Mat $\overline{\mathbf{\Phi}}(\mu) = \mathbf{\Phi}_T(\mu)\mathbf{\Phi}(\mu)$
6:      Mat $\mathbf{A}(\mu) = \overline{\mathbf{\Phi}}(\mu) + \lambda\mathbf{I}$
7:      Vec $\mathbf{b}(\mu) = \mathbf{\Phi}_T(\mu)\mathbf{f}$
8:      Vec $\beta(\mu) = \mathbf{A}(\mu) \setminus \mathbf{b}(\mu)$
9:      Vec $\hat{\mathbf{f}}(\mu) := \mathbf{\Phi}(\mu)\beta(\mu)$
10:     Real $\epsilon = \|\mathbf{f} - \hat{\mathbf{f}}(\mu)\|_2$
11: end procedure
12: Apply constraints $g(\mu)$: $\mu_j \in \Omega$, $j = 1, \dots, n$.
```

Numerical libraries and hardware. Among scientific libraries, our tests include Eigen [36] for the data structures management, BLAS [37] and sparse BLAS [38] for the dense and sparse matrices operations, PETSc [26] and SLEPc [27] for the parallel interface to BLAS routines, IBM Spectrum MPI [39] for the interface to distributed computing environments. Tests and analyses are performed on CINECA cluster Marconi100. The CINECA Marconi100 cluster occupies the 26th position in the "*top500*" list [40]. The cluster uses 980 nodes, each with IBM Power9 AC922 at 3.1 GHz 32 cores and 4 NVIDIA Volta V100 GPUs per node, with the GPU interconnection NVlink 2.0 at 16 GB and 256 GB of RAM each node.

3. Constrained least-squares approximation

The approximation of an input signal is widespread in image processing, e.g., computer graphics for compression [41] and restoration [42], biomedicine [43] and physics [44] applications, and graph processing, e.g., time-varying graphs signal reconstruction [45] and compression [46]. Given the problem in Eq. (1), our analysis compares the selected optimisation methods and proposes an efficient implementation for the solution of the minimisation problem for the signal approximation with different constraints (Section 3.1). Finally, the experimental results (Section 3.2) are discussed in terms of computational efficiency.

3.1. Constrained least-squares approximation

Given an input signal $f : \Omega \to \mathbb{R}$ and a set $\mathcal{B} = \{\varphi_j\}_{j=1}^n$ of basis functions, we consider the approximating function $\hat{f} := \sum_{j=1}^n \beta_j \varphi_j$ with $\beta := (\beta_j)_{j=1}^n$. In our setting, we assume that $\varphi_j(\mathbf{q}) := \varphi(\|\mathbf{q} - \mu_j\|_2)$ is a radial basis function (RBF) generated by the 1D Gaussian function $\varphi(s) := \exp(-s)$) where μ_j is its centre. To compute the approximating function, the constrained minimisation problem in Eq. (1) is solved, where the variables are the centres $\mu := (\mu_j)_{j=1}^n$ of the RBFs, $\alpha := 0$, and the constraints $g(\mu)$ represent the membership of the centres μ_j to Ω, as

$$\begin{cases} \min_\mu \|f - \hat{f}(\mu)\|_2^2, \\ \text{s.t. } \mu_j \in \Omega, \quad j = 1, \dots, n. \end{cases} \tag{2}$$

In the discrete setting, f and $\hat{f}$ are sampled at a set $\mathcal{Q} := \{\mathbf{q}_i\}_{i=1}^m \subseteq \Omega$ of points, i.e., $\mathbf{f} := (f(\mathbf{q}_i))_{i=1}^m$ and $\hat{\mathbf{f}}(\mu) := (\hat{f}(\mathbf{q}_i))_{i=1}^m$. The approximating signal is defined as

$$\hat{f}(\mathbf{q}_i) := \sum_{j=1}^n \beta_j \varphi(\|\mathbf{q}_i - \mu_j\|_2), \quad i = 1, \dots, m, \tag{3}$$

i.e., $\hat{\mathbf{f}}(\mu) = \mathbf{\Phi}(\mu)\beta(\mu)$, where $\mathbf{\Phi}(\mu) := (\varphi(\mathbf{q}_i, \mu_j))_{i=1\dots m}^{j=1\dots n}$ is the $m \times n$ Gram matrix. Since $n \ll m$, the solution to $\mathbf{\Phi}^\top(\mu)\mathbf{\Phi}(\mu)\beta(\mu) = \mathbf{\Phi}^\top(\mu)\mathbf{f}$ in Eq. (2) is generally ill-conditioned; indeed, the regularised linear system is solved as

$$(\mathbf{\Phi}^\top(\mu)\mathbf{\Phi}(\mu) + \lambda\mathbf{I})\beta(\mu) = \mathbf{\Phi}^\top(\mu)\mathbf{f}, \tag{4}$$

with $\lambda = 1e-12$ and constraints $\mu_j \in \Omega$, $j = 1, \dots, n$. Two variants of this problem are discussed: the first one is the bound-constraint version, which is typical of image processing as the signal is defined on a regular grid [47]; the second one is the non-linear geometric constraint version, which is typical of signals on graphs/meshes [48]. These two variants share the same objective function; however, the non-linear geometric constraints affect the selection and analysis of the minimisation method. Furthermore, signals defined on an irregular grid may not allow a straightforward domain decomposition; thus, the direct parallelisation of the computational kernels across the whole data set becomes relevant.

Table 1
Comparison among minimisation methods of the approximation problem on 2D images. $t = 1200$ s with 256 processes. The best results are in bold.

Metric	Functional value	Functional count	Time [s]
PRAXIS	**4.96**	8K	t
DIRECT-L	26.5	2M	$40t$
L-BFGS	13.94	0.5K	$2t$
COBYLA	9.48	8K	$4t$
ISRES	>30	**195**	$<t$
DIRECT-L + PRAXIS	25.94	2M + 7K	$41t$
DIRECT-L + L-BFGS	26.46	2M + 3K	$42t$

Algorithm and parallelisation. The objective function in Eq. (2) is computed through the Algorithm 1.

- Line 1 is performed out of the computation of the functional. One MPI process reads the input signal, scatters the signal values across the MPI processes, and broadcasts the input points m.
- Line 3 (k*−nn search* and *matrix definition*) computes the matrix $\mathbf{\Phi}$. The sparsity of the matrix depends on the parameters of the k−d search (e.g., k index). The query is parallelised on the k−d tree, where each MPI process is assigned a subset of centres. For each centre (i.e., the variable μ_j), the definition of the matrix $\mathbf{\Phi}$ requires the computation of the closest points among the point set Q. For this reason, a k−d tree is built on the input point set. Then, the parallelisation of the query is performed by scattering the subset of variables to each MPI process and broadcasting the k−d tree to every MPI process. Finally, after each MPI process has searched the k nearest points (i.e., 200 in our tests) for each variable assigned, the global matrix $\mathbf{\Phi}$ is computed by gathering the results from each MPI process. It is emphasised that each MPI acts on a subset of the rows of the matrix $\mathbf{\Phi}$.
- Line 4 (*Matrix transpose*) computes the transpose of the matrix $\mathbf{\Phi}$ that computationally corresponds to a sparse matrix copy (BLAS *omatcopy*).
- Line 5 (*Matrix-matrix multiplication*) computes the sparse matrix-sparse matrix multiplication through a BLAS *usmm* routine, with $\mathcal{O}(k \cdot k \cdot m)$ operations, where k is the number of non-zero elements per row of the sparse matrix, and m is the input point set.
- Line 6 (*Matrix shift*) computes a matrix shift, which computationally corresponds to a BLAS *axpy*, linear cost with m.
- Line 7 (*Matrix–vector multiplication*) computes a sparse matrix–vector multiplication through a BLAS *usmv* routine, with $\mathcal{O}(2\,\mathrm{km})$ operations.
- Line 8 (*Solve system*) solves the sparse linear system. The computational cost depends on the selected algorithm.
- Line 9 (*Matrix–vector multiplication*) computes a sparse matrix–vector multiplication through a BLAS *usmv* routine, with $\mathcal{O}(2\,\mathrm{km})$ operations.
- Line 10 (*Vex AXPY* and *Vec Norm*) computes the error norm through a BLAS vector sum (*axpy*) and a BLAS vector norm (*nrm2*), both with a computational cost that is linear with the number of input points.
- Line 12 computes the non-linear constraints for the variables μ.

All the BLAS operations are parallelised by distributing the matrices and vectors by rows among the MPI processes.

Solution of the linear system. The solution of the sparse linear system is the main operation of our problem in computational terms. According to [49], the *iterative biconjugate gradient stabilised* (BICGSTAB) [50] method is selected as solver, with the Block-Jacobi preconditioner. The BICGSTAB is a transpose-free version of the *biconjugate gradient* (BICG) method [51] that improves convergence and smoothness. BICGSTAB is composed of two matrix–vector multiplications, two scalar products, one vector norm, and two vector sum operations (i.e., *waxpy* and *axpbypcz*) that are linear with the elements of the vectors. The computational cost is $\mathcal{O}(kmt)$ with t iterations, k non-zero elements of the sparse matrix, and m rows. BICGSTAB guarantees the stability of the solution and good scalability properties.

3.2. Experimental results

Bound-constraints experimental set-up. The input signal is discretised on a 2D regular grid of $m \times m \approx 65\,\mathrm{K}$ points, with $n = 12\,\mathrm{K}$ variables related to the 2D spatial coordinates of $6K$ centres of the RBFs of the approximating function $\hat{\mathbf{f}}$ in Eq. (3). The matrix $\mathbf{\Phi}$ is computed with a k−nearest neighbourhood of 200 points. The coefficient matrix of the linear system in Eq. (4) is a sparse matrix of approximately $65\,\mathrm{K} \times 65\,\mathrm{K}$ elements with $40\,\mathrm{K}$ non-zero elements. The bound constraints force each variable to belong to the image domain, i.e., to fall between the lower and upper bounds for each dimension.

Comparison between minimisation methods. Table 1 shows the comparison among minimisation methods for the solution of Eq. (2) with bound constraints. PRAXIS has the best performance both in terms of computation time and functional value. DIRECT-L has a very high computation time due to the global search for the optimal solution. L-BFGS does not converge to the optimal solution. Furthermore, the initialisation of the solution through DIRECT-L does not improve the minimisation of PRAXIS and L-BFGS. Finally, COBYLA and ISRES perform worse than PRAXIS: ISRES does not converge to an optimal solution, while COBYLA has worse results regarding functional value and execution time.

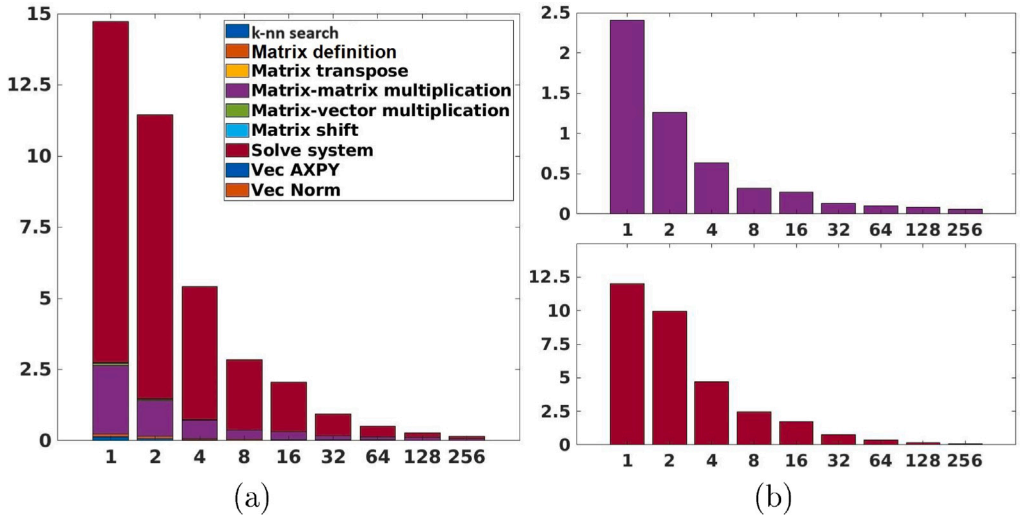

Fig. 1. (a): Execution time (y-axis, in seconds) with respect to the processes (x-axis), approximation problem. (b): Magnification of matrix-matrix multiplication (top) and solve system (bottom). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

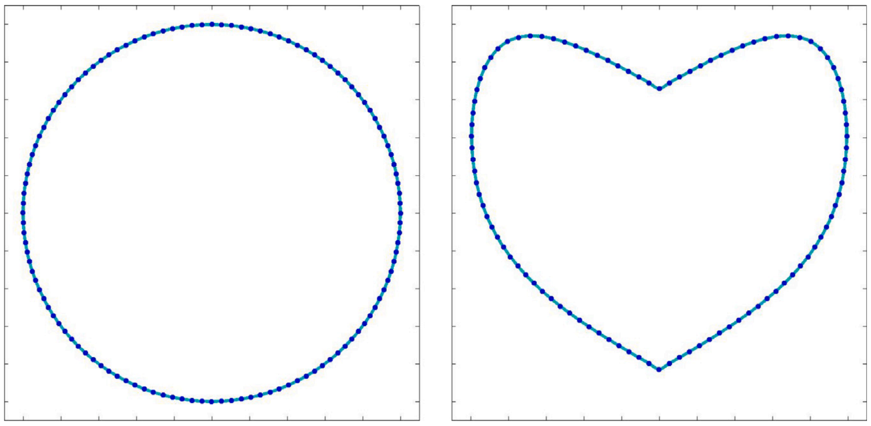

Fig. 2. Examples of input points sampled on a curve (blue dots) and the respective interpolating curve (cyan).

Scalability of functional computation. Fig. 1 and Table 2 show the scalability of the Algorithm 1 without constraints. It is mentioned that the $k-$nearest neighbours is a perfectly parallel operation. Matrix definition strictly depends on the type of the generating functions, k-index value, and other parameters related to the type of application. Matrix-matrix multiplication and linear system solver are the most expensive operations and show good scalability when increasing the number of processes. In particular, the matrix-matrix multiplication passes from 2.4 s with 1 process to 0.06 s with 256 processes; the linear system solve passes from 12 s with 1 process to 0.07 s with 256 processes. Matrix-matrix multiplication and linear system solver cover the 97.7% of the overall time with 1 process and the 87.3% with 256 processes. Even if these two operations are the most expensive, the contribution of the rest of the operations on a large number of iterations is not negligible. The total time varies from 14.7 s with 1 process to 0.95 s with 32 processes and 0.15 s with 256 processes. The efficiency is 48% with 32 processes and 38% with 256 processes. It is recalled that each node of the Marconi100 cluster is composed of 32 processes, and the efficiency further reduces when inter-node communications are required.

Non-linear geometric constraints. In curved domains (e.g., Fig. 2), a non-linear constraint is included to account for the geometry of the domain for each variable μ_i of the minimisation problem in Eq. (2). In particular, given the set of m input points representing a discrete curve, it is computed the interpolating curve and defined the non-linear constraints as an equality constraint of the distance between each variable and the curve. Fig. 2 shows examples of input points sampled on a curve and the respective interpolating curve. In our tests, it is selected a signal defined on a curve discretised with 2K input points and 125 variables μ_i are optimised.

Comparison among minimisation methods. For the computation and scalability of the functional, we refer to the Algorithm 1. In this case, the non-linear constraint affects the selection of the minimisation solver. In Table 3, it is presented the convergence of the

Table 2
Scalability analysis of each operation (Op.) in milliseconds of the approximation problem.

Op.	k−nn search	Mat def.	Mat transp.	Mat-Mat mult.	Mat-Vec mult.	Mat shift	Solve system	Vec-Vec add.	Vec norm	Total
1	122	111	5	2407	73	27	11 978	<0.1	<0.1	14 725
2	65	81	5	1266	39	17	9966	<0.1	<0.1	11 444
4	32	35	5	639	23	8	4681	<0.1	<0.1	5425
8	16	29	5	321	8	4	2463	<0.1	<0.1	2850
16	8	19	5	275	4	5	1748	<0.1	<0.1	2070
32	4	19	9	132	4	2	766	<0.1	<0.1	952
64	2	10	11	105	2	1	376	<0.1	<0.1	518
128	1	6	13	88	1	0.4	171	<0.1	<0.1	290
256	0.2	2	3	60	0.8	0.7	71	<0.1	<0.1	150

Table 3
Comparison among minimisation methods for the approximation problem with constraints. The best results are in bold.

Metric	Functional value	Functional count	Time [s]
ISRES	**4.05**	500K	>3K
COBYLA	4.51	44K	120
L-BFGS	4.84	**1500**	5

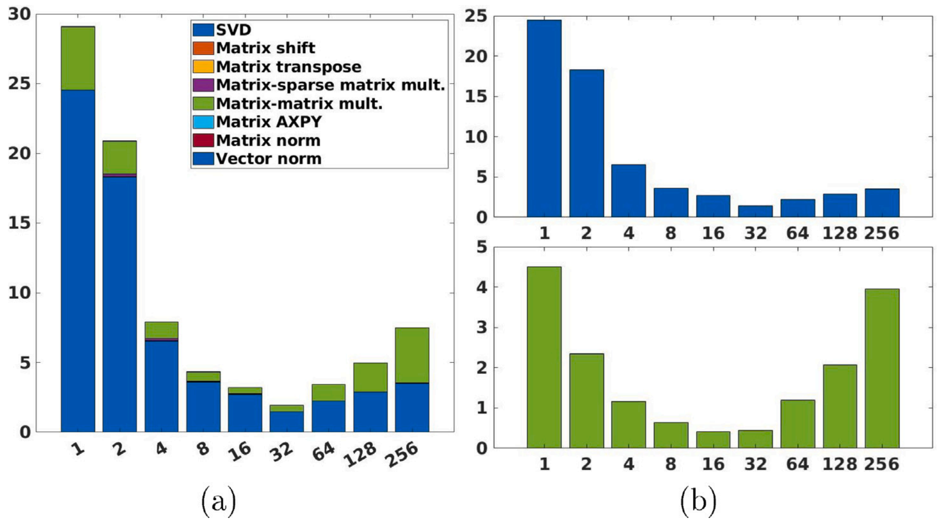

Fig. 3. (a): Execution time (y-axis, in seconds) with respect to the processes (x-axis), denoising problem. (b): Magnification of SVD (top) and matrix-matrix multiplication (bottom). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

minimisation methods for the solution of Eq. (2): L-BFGS does not converge to the optimal solution, and the global optimiser ISRES has better accuracy with respect to the local optimiser COBYLA (4.05 vs 4.51), at the cost of a larger number of iterations and execution time. Finally, PRAXIS and DIRECT-L do not manage non-linear constraints and cannot be applied.

4. Constrained singular value decomposition for signal denoising

The denoising of signals is widespread in many applications; images acquired by digital sensors are generally affected by different types of noise, such as speckle [52] and exponential [53] noise on biomedical images, salt-and-pepper [54], Gaussian [55], and Poisson [56] noise on images acquired through camera sensors. Given the problem in Eq. (1), it is analysed the minimisation for the denoising problem (Section 4.1) with the experimental results (Section 4.2).

4.1. Constrained SVD

Given an input 2D image $\mathbf{f}$ represented on a squared $m \times m$ grid, it is computed the singular value decomposition $\mathbf{f} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$ where $\mathbf{U}$ and $\mathbf{V}$ are $m \times m$ dense matrices and $\mathbf{S}$ is a diagonal $m \times m$ matrix. This factorisation is well-known for separating high-frequency by low-frequency components of the image and allowing us to reduce the noise components while preserving the features/properties of the input image. The approximating function is defined as $\hat{\mathbf{f}} = \mathbf{U}\hat{\mathbf{S}}\mathbf{V}^\top$, where $\hat{\mathbf{S}}$ is computed through a threshold

Algorithm 2 Constrained singular value decomposition for signal denoising.

```
1:  $\mathbf{f}$ = Input discrete signal
2:  procedure $\hat{\mathbf{f}}$ = DENOISE($\mathbf{f}$)
3:      Mat $\mathbf{USV}$ = SVD($\mathbf{f}$)
4:      Mat $\hat{\mathbf{S}} = \mathbf{S} - \mu$
5:      Mat $\overline{\mathbf{U}} = \mathbf{U}\hat{\mathbf{S}}$
6:      Mat $\mathbf{V}_T = \mathbf{V}^\top$
7:      Mat $\hat{\mathbf{f}} = \overline{\mathbf{U}}\mathbf{V}_T$
8:      Real $\epsilon_1 = \|\mathbf{f} - \hat{\mathbf{f}}\|_F$
9:      Real $\epsilon_2 = \|\hat{\mathbf{S}}\|_1$
10:     Real $\epsilon = \epsilon_1 + \alpha\epsilon_2$
11: end procedure
12: Apply constraints $g(\mu)$: $\mathbf{S}_{ii} - \mu_i > 0$, $i = 1, \dots, m$
```

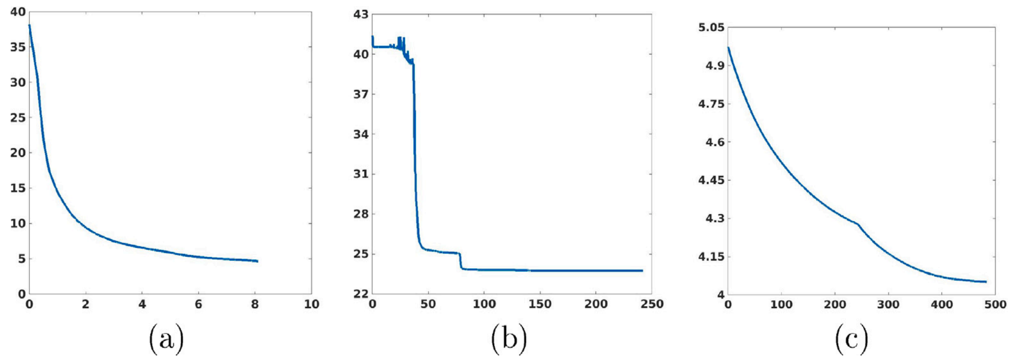

Fig. 4. Minimisation of PRAXIS with functional value (y-axis) with respect to the number of evaluations of the functional (x-axis, $\times 10^3$): (a) approximation problem, (b) denoising problem. Minimisation of ISRES, approximation problem with non-linear constraints (c).

operation on $\mathbf{S}$. The penalisation term is defined as the nuclear norm of the approximated signal $\mathcal{P} = \|\hat{\mathbf{f}}\|_*$; the nuclear norm is linked with the $\mathbf{S}$ matrix of the SVD and regularises high-frequency components [57].

The variables of our minimisation problem $\mu = (\mu_i)_{i=1}^m$ are defined as the threshold values to be applied to the diagonal of $\mathbf{S}$ and the singular values are computed by applying the threshold values as $\hat{\mathbf{S}} = \mathbf{S} - \mu$, where it is assumed $\mathbf{S} - \mu$ as $\mathbf{S}_{ii} - \mu_i$, with $\mathbf{S}_{ii}$ as the (i,i) entry of $\mathbf{S}$. The bound constraint is added to the minimisation problem to ensure the non-negativity of the threshold singular values. The minimisation problem is defined as

$$\begin{cases} \min_\mu \|\mathbf{f} - \mathbf{U}(\mathbf{S} - \mu)\mathbf{V}^\top\|_F^2 + \alpha \sum (\mathbf{S} - \mu), \\ \qquad\qquad \text{s.t.} \quad \mathbf{S} - \mu > \mathbf{0}. \end{cases} \tag{5}$$

This problem can be assumed as convex, and the derivatives in the analytic form are available. The computation of the objective function requires the application of the main algebraic operations, including BLAS 1 (e.g., *axpy*), BLAS 3 (e.g., *gemm*) and sparse BLAS 3 (e.g., *usmm*). Furthermore, it includes the computation of the singular values of a dense/sparse matrix, depending on the selected input signal. The analysis of this problem is general and can be extended to other image-processing applications. For the efficient computation of the SVD, the tridiagonalisation of the cross-product matrix without forming it explicitly is achieved through the bidiagonalisation $\mathbf{f} = \mathbf{PBQ}^*$ where $\mathbf{P}$ and $\mathbf{Q}$ are unitary matrices and $\mathbf{B}$ is an upper bidiagonal matrix. Then, the SVD of $\mathbf{B}$ is applied to recover the SVD of $\mathbf{f}$. The bidiagonalisation is achieved through the Lanczos method [58]. According to our experimental tests (Section 4.2), the tridiagonalisation of the cross-product matrix is selected as the SVD method.

Algorithm and parallelisation. The objective function in Eq. (5) is computed through the Algorithm 2.

- Line 1 is performed out of the computation of the functional. One MPI process reads the input 2D signal, scatters the signal values across the MPI processes, and broadcasts the input points m.
- Line 3 (*SVD*) computes the SVD decomposition of the image and saves two full matrices ($\mathbf{U}$, $\mathbf{V}$) and a sparse matrix ($\mathbf{S}$).
- Line 4 (*Matrix shift*) computes a matrix shift, which computationally corresponds to a BLAS *axpy*, linear cost with m.
- Line 5 (*Matrix-sparse matrix multiplication*) computes a sparse matrix-matrix multiplication through sparse BLAS *usmm* routine with $\mathcal{O}(\mathrm{km}^2)$ operations, where k is the number of non-zero elements per row of the sparse matrix, and m is the input point set.
- Line 6 (*Matrix transpose*) computes the transpose of the right eigenvectors matrix that computationally corresponds to a matrix copy (BLAS *omatcopy*).

Table 4
Comparison among minimisation methods for the denoising problem on the 2D image (5616 × 3744 matrix). t = 800 s with 32 processes. The best results are in bold.

Metric	Functional value	Functional count	Time [s]
L-BFGS	31.25	**3**	t
DIRECT-L	25.45	1M	2200t
PRAXIS	**24.47**	250K	600t
COBYLA	39.88	250K	1100t
ISRES	>50	1M	2000t
DIRECT-L + PRAXIS	25.42	1M + 1K	2210t
DIRECT-L + L-BFGS	25.45	1M	2200t

Table 5
Comparison among SVD methods on dense 5616 × 3744 matrix with first 300 singular values and sparse 16 368 × 16 384 matrix. Execution time is expressed in milliseconds. N.C. means the method does not converge.

Matrix	Dense			Sparse		
Processes	**1**	**32**	**128**	**1**	**32**	**128**
Cross	24 548	1460	2880	19 360	876	242
Randomised	N.C.			90 950	12 543	9124
Cyclic	>100K	>100K	4217	>100K	12 362	6855
Lanczos	N.C.			28 702	1160	306
Trlanczos	6073	3415	3879	28 599	1175	302

- Line 7 (*Matrix-matrix multiplication*) computes a matrix-matrix product through BLAS *gemm*, with a maximum computational cost of $\mathcal{O}(\mathrm{m}^3)$.
- Line 8 (*Matrix AXPY* and *Matrix norm*) computes both a matrix-matrix addition and a matrix Frobenius norm in linear cost with the number of input points.
- Line 9 (*Vector norm*) computes the $\mathcal{L}^1$ norm of the sparse matrix $\hat{\mathbf{S}}$, linear cost with the number of variables.
- Line 10 computes the sum of two scalars.
- Line 12 computes the set of non-linear constraints for the variables μ, as $\mu_i < \mathbf{S}_{ii}$ for each variable i with respect to the related diagonal entry of the singular values matrix $\mathbf{S}$.

All the BLAS operations are parallelised by distributing the matrices and vectors by rows among the MPI processes.

4.2. Experimental results: bound constraints

Experimental set-up. It is selected an input signal as a high-resolution image or the weighted Laplacian matrix of a large grid. As a dense rectangular matrix, a 21-megapixel camera sensor acquires a typical image resolution of 5616 × 3744. Given a regular grid of 256 × 256, the Laplacian matrix is a 16 368 × 16 384 matrix and is typically banded sparse.

Comparison between minimisation methods. Table 4 compares the minimisation methods for the solution of Eq. (5). L-BFGS converges to the global minimum and has the best results in terms of execution time/functional evaluation number; the knowledge of the analytic derivatives allows the method to compute the optimal minima with few evaluations of the functional. PRAXIS and DIRECT-L have a large number of evaluations of the functional. However, they both perform better than L-BFGS even without knowing the derivatives. In particular, PRAXIS has better results than DIRECT-L. Finally, the initialisation of the solution through DIRECT-L does not improve the minimisation of PRAXIS and L-BFGS. COBYLA and ISRES have worse results than PRAXIS regarding the functional value and computation time.

Comparison between singular value decomposition methods. Five numerical methods for the computation of the SVD [27] are compared on two different matrices in terms of execution time and scalability, by searching for the complete set of singular values of a dense matrix and a subset of 500 singular values of a sparse matrix. All the SVD methods have a convergence tolerance of 10^{-6}. Table 5 shows that *Cross* has the best results on sparse matrix, while *Lanczos* and *thick-restart Lanczos* have slightly worse results. On dense matrix, *thick-restart Lanczos* has better results than the other methods but worse scalability properties. *Cross* has the best results on 32 processes. After this preliminary test, it is selected *Cross* as the SVD solver. It is mentioned that a complete comparison among SVD solvers should consider additional metrics and parameters, e.g., the accuracy when computing the first singular value or the complete set of singular values.

Scalability of functional computation. Fig. 3 and Table 6 shows the scalability of the Algorithm 2 on a dense 5616 × 3744 matrix with 300 singular values; the eigenvectors matrices are dense 5610 × 300 and 3744 × 300, and the singular values matrix is diagonal sparse 300 × 300. The SVD passes from 24.5 s with 1 process to 1.4 s with 32 processes; the matrix-matrix multiplication passes from 4.5 s with 1 process to 0.4 s with 32 processes. The total time varies from 29 s with 1 process to less than 2 s with 32 processes; after this number of processes, both SVD and matrix-matrix multiplication increases the execution time. The efficiency with 32 processes is 46%.

Table 6
Scalability analysis of each operation (Op.) in milliseconds of the denoising problem.

Op.	SVD	Mat shift	Mat transp.	Mat-Mat mult	Mat-Mat mult	Mat-Mat add.	Mat norm	Vec norm	Total
1	24 548	<0.1	1	2	4503	21	33	<0.1	29 111
2	18 293	<0.1	56	168	2340	13	17	<0.1	20 888
4	6546	<0.1	40	139	1168	6	8	<0.1	7910
8	3580	<0.1	23	73	648	4	4	<0.1	4334
16	2716	<0.1	14	41	419	3	2	<0.1	3197
32	1460	<0.1	7	27	449	1	1	<0.1	1948
64	2223	<0.1	4	18	1189	<0.1	1	<0.1	3436
128	2880	<0.1	4	11	2069	<0.1	<0.1	<0.1	4965
256	3517	<0.1	4	11	3945	<0.1	<0.1	<0.1	7487

5. Discussion

Managerial insights. In many applications (e.g., design, structural, chemical, medicine, manufacturing), simulation problems are defined as non-linear optimisation problems on a complex domain with constrained variables. For example, given a mechanical component, it is relevant to minimise the stress on different sub-parts depending on the parametrisation of physical variables (e.g., the position of the joints). In a medical signal (e.g., a magnetic resonance), it is relevant to filter the image to improve the post-processing of the data (e.g., segmentation) or the visual analysis of the physician. Fig. 8 shows two real-case applications: the reconstruction of the solution of the fluid flow-field equation on an unstructured grid (top row) and the denoising of a satellite image affected by speckle noise (second row).

Solving optimisation problems on complex domains requires the application of dedicated minimisation algorithms, and the solution may require high computational cost and memory use (see Fig. 4). Previous work still lacks an analysis that jointly proposes an efficient implementation for the approximation and filtering of signals on arbitrary domains, a comparison of optimisation methods and their implementations with HPC techniques for solving the mentioned problems, including a scalability analysis of the main algebraic operations (e.g., solution of a linear system, singular value decomposition). Our analysis is general regarding the signal processing problem, variables, constraints (e.g., bounded, non-linear), domain (e.g., structured and unstructured grid, dimensionality), HPC hardware (e.g., cloud computing, homogeneous vs. heterogeneous).

Our comparison of the minimisation methods and the analysis of the main algebraic operations (i.e., matrix-matrix multiplication, singular value decomposition, solution of a linear system) in terms of scalability and efficiency allows the user to identify the bottlenecks and performance of the optimisers in an HPC environment for the solution of processing problems. Furthermore, our comparison of the optimisation methods in terms of applicability, domain, and supported constraints can help the users to select the algorithm and scientific library that best fit to their problem.

Limitations. As the main limitation, our comparison selects two types of problems (approximation and denoising) with different declinations depending on the chosen solution approach. This implies that, even if the algebraic operations included are the same as our analysis, their balance in terms of computation time could be different. Therefore, our analysis does not propose an optimal solving method for these types of problems, but rather shows an efficient implementation by offering a comparison of the different minimisation methods and using high-performance hardware.

6. Conclusions and future work

Our paper proposes an analysis of the solution of two minimisation problems of signal processing with HPC tools: approximation and denoising. For each problem, the characteristics and results of the minimisation methods are analysed in terms of convergence and execution time. PRAXIS has shown the best results on bound-constrained problems, while ISRES has shown the best results on constrained problems. Also, the computation of the functional and the scalability properties of the algebraic operations are discussed, including the solution of a linear system and the decomposition of the singular values. The two problems apply the main algebraic operations common to most signal minimisation problems; our general analysis can be extended to other signal processing problems. As examples of the applications, the approximation of a signal on a regular grid (Fig. 5), on a curve (Fig. 6), and the image denoising (Fig. 7) are presented. Also, real-case applications to computational fluid dynamic reconstruction and satellite images denoising are shown. In future work, it is proposed to extend the analysis to other classes of problems in signal processing (e.g., clustering) and perform the experimental tests on the novel Leonardo cluster of CINECA. Furthermore, it is planned to account for software libraries that exploit hybrid computing (e.g., Magma) and heterogeneous architectures to improve the workload balance. To further improve the generalisation of our analysis, it is planned to extend our comparison to finite-element simulations in 2D and 3D domains.

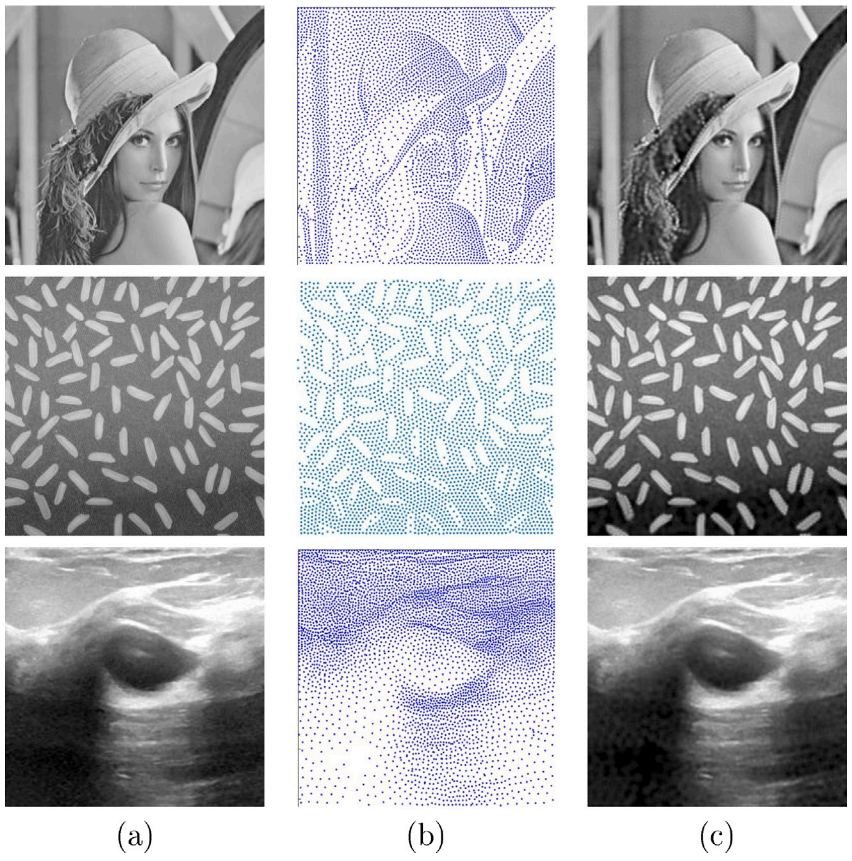

Fig. 5. Input image (a), variables (i.e., RBF centres) (b), and reconstructed image (c). Third row: ultrasound image from breast district.

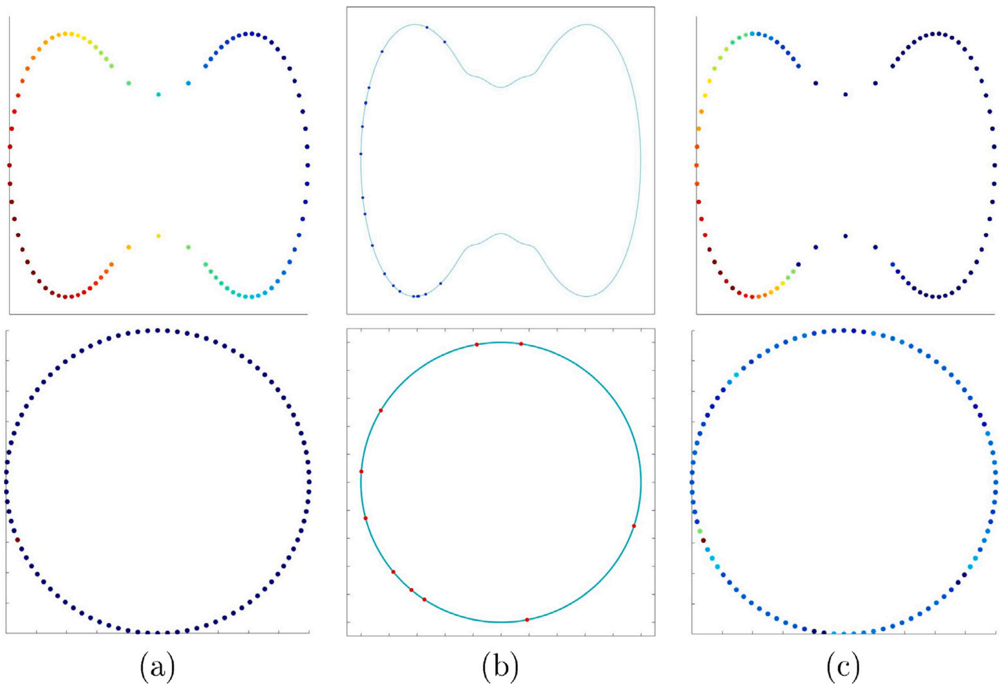

Fig. 6. Input signal on a curve (a), variables (i.e., RBF centres, dots) with respect to the interpolating curve (cyan) (b), and reconstructed signal (c). Second row: Dirac signal.

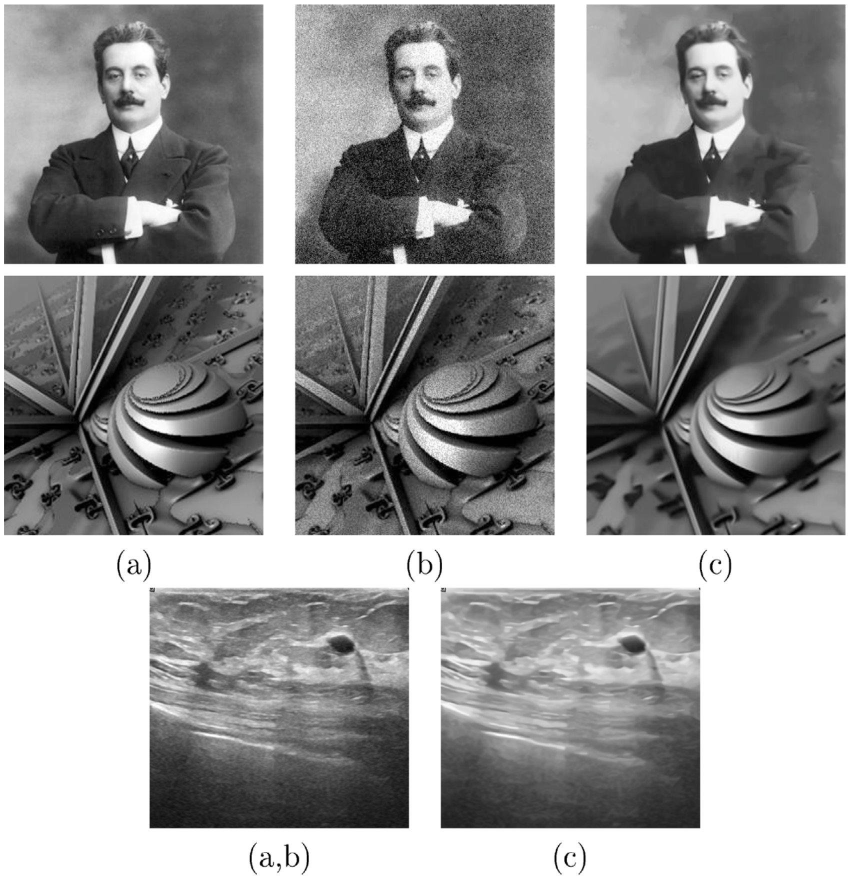

Fig. 7. Input (a), noisy (b), and denoised image (c). First row: Gaussian noise; second row: speckle noise; third row: breast ultrasound image.

CRediT authorship contribution statement

Simone Cammarasana: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Formal analysis, Conceptualization. **Giuseppe Patané:** Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Formal analysis, Conceptualization.

Ethical approval

The proposed work does not include any ethically approved human or animal experiments.

Funding

SC and GP are part of RAISE Innovation Ecosystem, funded by the European Union - NextGenerationEU and by the Ministry of University and Research (MUR), National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.5, project “RAISE - Robotics and AI for Socio-economic Empowerment” (ECS00000035). Tests on the CINECA Cluster are supported by the ISCRA-C project US-SAMP, HP10CXLQ1S.

Declaration of competing interest

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

Availability of data and materials

Data sharing does not apply to this article as no datasets were generated or analysed during the current study.

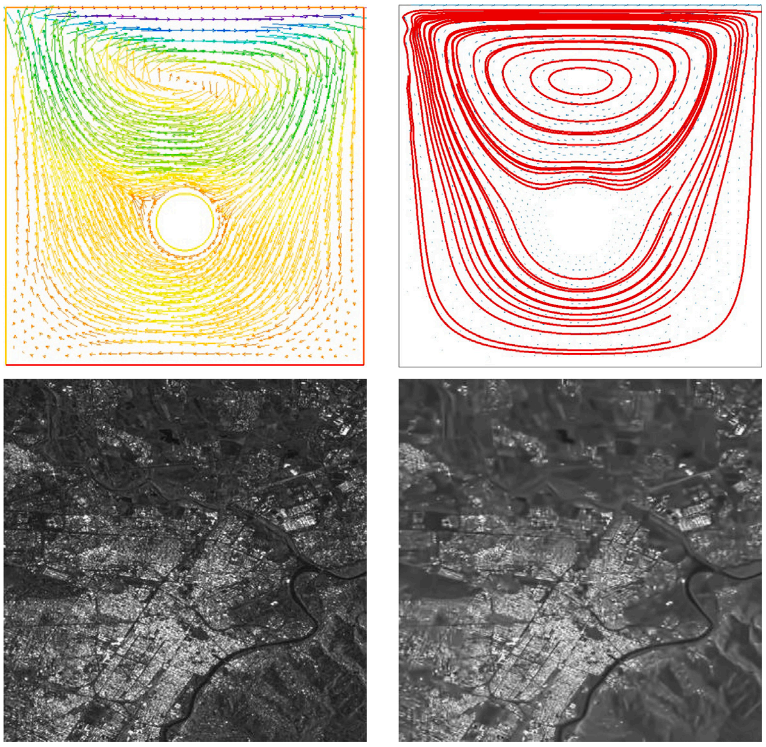

Fig. 8. First row: input (left) and reconstructed (right) flow field. Second row: noisy (left) and denoised (right) satellite image.

References

[1] Y. Huo, J. Blaber, S.M. Damon, B.D. Boyd, S. Bao, P. Parvathaneni, C.B. Noguera, S. Chaganti, V. Nath, J.M. Greer, et al., Towards portable large-scale image processing with high-performance computing, J. Digit. Imaging 31 (2018) 304–314.

[2] S. Cammarasana, P. Nicolardi, G. Patané, Fast learning framework for denoising of ultrasound 2D videos and 3D images, in: Image Analysis and Processing. ICIAP 2022 Workshops: ICIAP International Workshops, Lecce, Italy, May 23–27, 2022, Revised Selected Papers, Part I, Springer, 2022, pp. 475–486.

[3] B. Hendrickson, J.W. Berry, Graph analysis with high-performance computing, Comput. Sci. Eng. 10 (2) (2008) 14–19.

[4] C.A. Gulo, A.C. Sementille, J.M.R. Tavares, Techniques of medical image processing and analysis accelerated by high-performance computing: A systematic literature review, J. Real-Time Image Process. 16 (2019) 1891–1908.

[5] J. Kocz, L. Greenhill, B. Barsdell, D. Price, G. Bernardi, S. Bourke, M. Clark, J. Craig, M. Dexter, J. Dowell, et al., Digital signal processing using stream high performance computing: A 512-input broadband correlator for radio astronomy, J. Astron. Instrum. 4 (01n02) (2015) 1550003.

[6] J.A. Webb, High performance computing in image processing and computer vision, in: Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 2-Conference B: Computer Vision & Image Processing.(Cat. No. 94CH3440-5), IEEE, 1994, pp. 218–222.

[7] S. Cammarasana, P. Nicolardi, G. Patanè, Real-time denoising of ultrasound images based on deep learning, Med. Biol. Eng. Comput. 60 (8) (2022) 2229–2244.

[8] M.J.D. Powell, A direct search optimization method that models the objective and constraint functions by linear interpolation, in: S. Gomez, J.-P. Hennart (Eds.), Advances in Optimization and Numerical Analysis, Springer Netherlands, Dordrecht, ISBN: 978-94-015-8330-5, 1994, pp. 51–67.

[9] J. Bös, R. Nordmann, Numerical acoustical optimization with respect to various objective functions, Acta Acust. United Acust. 88 (2) (2002) 278–285.

[10] S. Pramanik, M.G. Chandra, C.V. Sridhar, A. Kulkarni, P. Sahoo, C.D.V. Vishwa, H. Sharma, V. Navelkar, S. Poojary, P. Shah, M. Nambiar, A quantum-classical hybrid method for image classification and segmentation, in: 2022 IEEE/ACM 7th Symposium on Edge Computing, SEC, 2022, pp. 450–455.

[11] D. Woldemichael, A. Woldeyohannes, Optimization of pressure vessel design using pyopt, ARPN J. Eng. Appl. Sci. 11 (24) (2016) 14264–14268.

[12] C. Zhu, R.H. Byrd, P. Lu, J. Nocedal, Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization, ACM Trans. Math. Softw. 23 (4) (1997) 550–560.

[13] F. Yang, M. Ding, X. Zhang, W. Hou, C. Zhong, Non-rigid multi-modal medical image registration by combining L-BFGS-B with cat swarm optimization, Inf. Sci. 316 (2015) 440–456.

[14] L. Jiang, R.H. Byrd, E. Eskow, R.B. Schnabel, A Preconditioned L-BFGS Algorithm with Application to Molecular Energy Minimization, Tech. Rep., Colorado Univ. at Boulder dept. of Computer Science, 2004.

[15] Y. Rao, Y. Wang, Seismic waveform tomography with shot-encoding using a restarted L-BFGS algorithm, Sci. Rep. 7 (1) (2017) 1–9.

[16] R.P. Brent, Algorithms for Minimization Without Derivatives, Courier Corporation, 2013.

[17] C. Charrier, H. Cherifi, Quantifying perceptual image quality by difference scaling, in: Proceedings of the Sixth International Symposium on Signal Processing and Its Applications (Cat.No.01EX467), vol. 2, 2001, pp. 422–425.

[18] P. Reynolds, V. Pereyra, Application of optimisation techniques to finite element analysis of piezocomposite devices, in: IEEE Ultrasonics Symposium, 2004, vol. 1, IEEE, 2004, pp. 634–637.

[19] A. Ünlü, Estimation of careless error and lucky guess probabilities for dichotomous test items: A psychometric application of a biometric latent class model with random effects, J. Math. Psych. 50 (3) (2006) 309–328.

[20] T.P. Runarsson, X. Yao, Stochastic ranking for constrained evolutionary optimization, IEEE Trans. Evol. Comput. 4 (3) (2000) 284–294.
[21] F. Zhan, Development of X-ray spectrometer automatic adjustment system based on global optimization algorithm, 2021, arXiv preprint arXiv:2105.02779.
[22] A. Iliopoulos, J.G. Michopoulos, R. Jones, A.J. Kinloch, D. Peng, A framework for automating the parameter determination of crack growth models, Int. J. Fatigue 169 (2023) 107490.
[23] J.M. Gablonsky, C.T. Kelley, Tech. Rep., North Carolina State University. Center for Research in Scientific Computation, 2000.
[24] V.S. Shilimkar, A. Weisshaar, Modeling of metal-fill parasitic capacitance and application to on-chip slow-wave structures, IEEE Trans. Microw. Theory Tech. 65 (5) (2017) 1456–1464.
[25] G. Nicosia, S. Rinaudo, E. Sciacca, An evolutionary algorithm-based approach to robust analog circuit design using constrained multi-objective optimization, in: Research and Development in Intelligent Systems XXIV: Proceedings of AI-2007, the Twenty-Seventh SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Springer, 2008, pp. 7–20.
[26] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, et al., Argonne National Laboratory, 2019.
[27] V. Hernandez, J.E. Roman, V. Vidal, SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems, ACM Trans. Math. Softw. 31 (3) (2005) 351–362.
[28] D.R. Jones, C.D. Perttunen, B.E. Stuckman, Lipschitzian optimization without the Lipschitz constant, J. Optim. Theory Appl. 79 (1993) 157–181.
[29] M.J. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives, Comput. J. 7 (2) (1964) 155–162.
[30] L.M. Rios, N.V. Sahinidis, Derivative-free optimization: A review of algorithms and comparison of software implementations, J. Global Optim. 56 (2013) 1247–1293.
[31] A. Gómez-Iglesias, Solving large numerical optimization problems in HPC with python, in: Proceedings of the 5th Workshop on Python for High-Performance and Scientific Computing, 2015, pp. 1–8.
[32] F. Liu, A. Fredriksson, S. Markidis, A survey of HPC algorithms and frameworks for large-scale gradient-based nonlinear optimization, J. Supercomput. 78 (16) (2022) 17513–17542.
[33] M. Kurz, P. Offenhäuser, D. Viola, O. Shcherbakov, M. Resch, A. Beck, Deep reinforcement learning for computational fluid dynamics on hpc systems, J. Comput. Sci. 65 (2022) 101884.
[34] B. Balis, K. Figiela, K. Jopek, M. Malawski, M. Pawlik, Porting HPC applications to the cloud: A multi-frontal solver case study, J. Comput. Sci. 18 (2017) 106–116.
[35] P. González, R.R. Osorio, X.C. Pardo, J.R. Banga, R. Doallo, An efficient ant colony optimization framework for HPC environments, Appl. Soft Comput. 114 (2022) 108058.
[36] G. Guennebaud, B. Jacob, et al., Eigen v3, 2010, http://eigen.tuxfamily.org.
[37] L.S. Blackford, A. Petitet, R. Pozo, K. Remington, R.C. Whaley, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, et al., An updated set of basic linear algebra subprograms (BLAS).
[38] K. Remington, R. Pozo, NIST Sparse BLAS User's Guide, - 6744, National Institute of Standards and Technology, Gaithersburg, MD, 2001.
[39] IBM Spectrum MPI, 2023, https://www.ibm.com/products/spectrum-mpi. (Accessed 4 May 2023).
[40] CINECA MARCONI100, 2023, https://www.top500.org/system/179845/. (Accessed 1 August 2023).
[41] S. Dhawan, A review of image compression and comparison of its algorithms, Int. J. Electron. Commun. Technol. 2 (1) (2011) 22–26.
[42] M.R. Banham, A.K. Katsaggelos, Digital image restoration, Signal Process. Mag. 14 (2) (1997) 24–41.
[43] M.T. McCann, M. Unser, et al., Biomedical image reconstruction: From the foundations to deep neural networks, Found. Trends® Signal Process. 13 (3) (2019) 283–359.
[44] É. Thiébaut, J.-F. Giovannelli, Image reconstruction in optical interferometry, Signal Process. Mag. 27 (1) (2009) 97–109.
[45] K. Qiu, X. Mao, X. Shen, X. Wang, T. Li, Y. Gu, Time-varying graph signal reconstruction, IEEE J. Sel. Top. Sign. Proces. 11 (6) (2017) 870–883.
[46] M. Besta, S. Weber, L. Gianinazzi, R. Gerstenberger, A. Ivanov, Y. Oltchik, T. Hoefler, Slim graph: Practical lossy graph compression for approximate graph processing, storage, and analytics, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2019, pp. 1–25.
[47] S. Cammarasana, G. Patane, Adaptive membership functions and F-transform, Trans. Fuzzy Syst. (2024).
[48] S. Cammarasana, G. Patanè, Kernel-based sampling of arbitrary signals, Comput. Aided Des. 141 (2021) 103103.
[49] A. Galizia, S. Cammarasana, A. Clematis, G. Patane, Evaluating accuracy and efficiency of HPC solvers for sparse linear systems with applications to PDEs, 2022, arXiv preprint arXiv:2201.05413.
[50] H.A. Van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of bi-CG for the solution of nonsymmetric linear systems, J. Sci. Stat. Comput. 13 (2) (1992) 631–644.
[51] R. Fletcher, Conjugate gradient methods for indefinite systems, in: Numerical Analysis: Proceedings of the Dundee Conference on Numerical Analysis, 1975, Springer, 2006, pp. 73–89.
[52] C.B. Burckhardt, Speckle in ultrasound B-mode scans, Trans. Sonics Ultrason. 25 (1) (1978) 1–6.
[53] A. Shrivastava, M. Shinde, S. Gornale, P. Lawande, An approach-effect of an exponential distribution on different medical images, Int. J. Comput. Sci. Netw. Secur. 7 (9) (2007) 235.
[54] J. Azzeh, B. Zahran, Z. Alqadi, Salt and pepper noise: Effects and removal, JOIV: Int. J. Inform. Vis. 2 (4) (2018) 252–256.
[55] F. Russo, A method for estimation and filtering of Gaussian noise in images, Trans. Instrum. Meas. 52 (4) (2003) 1148–1154.
[56] H.J. Trussell, R. Zhang, The dominance of poisson noise in color digital cameras, in: 2012 19th IEEE Intern. Conf. on Image Processing, IEEE, 2012, pp. 329–332.
[57] S. Cammarasana, G. Patane, Learning-based low-rank denoising, Signal, Image Video Process. 17 (2) (2023) 535–541.
[58] G. Golub, W. Kahan, Calculating the singular values and pseudo-inverse of a matrix, J. Soc. Ind. Appl. Math. Ser. B: Numer. Anal. 2 (2) (1965) 205–224.