CROQuant: Complex Rank-One Quantization Algorithm, with Application to Butterfly Factorizations

MAËL CHAUMETTE and RÉMI GRIBONVAL, Inria, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, France

ELISA RICCIETTI, ENS de Lyon, CNRS, Inria, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, France

This paper presents a quantization algorithm for complex-valued rank-one matrices that exploits rescaling-invariances of the problem to reach improved accuracy with the same number of mantissa bits compared to the usual round-to-nearest strategy. This work extends results previously established in the real-valued setting to the complex domain, and provides an algorithm with a complete theoretical convergence analysis. The proposed approach replaces the naive expression of the optimal quantization problem as the optimization over $m + n$ (quantized) complex entries by the optimization of a function of a single complex-valued scaling parameter. The resulting function is piecewise constant, and we show that, despite the infinite number of piecewise constant regions induced by quantization, the associated optimization problem admits a minimizer under mild non-degeneracy assumptions. This algorithm is also used as a building block for a heuristic strategy to quantize complex-valued butterfly-structured sparse matrices appearing for example in the fast Fourier transform. For butterfly matrices, the number of bits is reduced by 30% for a given precision by our algorithm, compared to element-wise round-to-nearest quantization.

CCS Concepts: • **Mathematics of computing → Discrete optimization**; **Computations on matrices**; • **Theory of computation → Design and analysis of algorithms**.

Additional Key Words and Phrases: Low-precision Quantization, Rank-One matrices, Butterfly matrices, Fast Fourier transform

ACM Reference Format:
Maël Chaumette, Rémi Gribonval, and Elisa Riccietti. 2026. CROQuant: Complex Rank-One Quantization Algorithm, with Application to Butterfly Factorizations. *ACM Trans. Math. Softw.* 1, 1 (February 2026), 30 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Quantization is a fundamental tool in numerical analysis, computer science and machine learning. With the ever-increasing size of models and data, the need to reduce memory by lowering numerical precision while maintaining acceptable performance is becoming crucial [10, 24]. In linear algebra for instance, the propagation of rounding errors for fundamental operations is well-understood and low precision floating-point arithmetic has been exploited to reduce the cost of many computational tasks, often while preserving a high accuracy thanks to mixed precision algorithms [1, 9, 23]; see [14] for a recent survey.

This project was supported by the SHARP project of the PEPR-IA (ANR-23-PEIA-0008, granted by France 2030), and the project ANR MEPHISTO ANR-24-CE23-7039. The authors would like to thank the Centre Blaise Pascal's platform at ENS de Lyon (Lyon, France) for computing facilities. The platform uses the SIDUS solution [22] developed by Emmanuel Quemener.

Authors' addresses: Maël Chaumette, mael.chaumette@inria.fr; Rémi Gribonval, remi.gribonval@inria.fr, Inria, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, Lyon cedex 07, France; Elisa Riccietti, elisa.riccietti@ens-lyon.fr, ENS de Lyon, CNRS, Inria, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, Lyon cedex 07, France.

Manuscript submitted to ACM

Rank-one matrices in particular play a central role in many compression problems in machine learning and signal processing. They appear for instance as a building block of the singular value decomposition [15] and in quasi-Newton optimization methods [3]. Most notably, they have a key role in butterfly factorization algorithms. Butterfly matrices are structured sparse matrices and their product is often used to factorize dense matrices, such as those appearing in the Hadamard or discrete Fourier transforms [6, 16], to speed up matrix-vector products and reduce storage [7, 18]. They also have been used to accelerate Gaussian elimination by removing pivoting [19, 21]. If sparse matrix factorization already serves as a compression technique, quantization of such factorizations can further amplify the computational gains. The problem of quantizing a product of butterfly matrices can be heuristically decomposed into a succession of rank-one quantization problems, thanks to the particular structure of such matrices. The quantization of rank-one matrices thus plays a key role in the quantization of butterfly factorizations.

Recently, an optimal quantization approach for *real-valued* rank-one matrices, which exploits rescaling-invariances to minimize the quantization error, was introduced in [12]. In this work, we extend these results to the case of *complex-valued* rank-one matrices. This extension is fundamental to treat several applications in signal processing, such as the Fast Fourier Transform (FFT). Indeed, the FFT involves matrices with the butterfly structure which are complex-valued, hence the approach in [12] cannot be directly applied. The floating-point accuracy of the FFT has been carefully studied [2], providing a solid basis against which quantization errors can be assessed.

As in the real-valued case in [12], the naive *round-to-nearest* (RTN) approach that quantizes a rank-one matrix $\boldsymbol{x}\boldsymbol{y}^H$ (with $\boldsymbol{x}, \boldsymbol{y}$ two vectors) by mapping each element of $\boldsymbol{x}$ and $\boldsymbol{y}$ to its nearest floating-point neighbor is generally not optimal. Thus, inspired by [12], we propose a method that exploits the rescaling invariances of the problem, i.e., the fact that $\forall \lambda \in \mathbb{C}^*, \boldsymbol{x}\boldsymbol{y}^H = (\lambda \boldsymbol{x})(\frac{1}{\lambda}\boldsymbol{y})^H$. While this is true in exact arithmetic, in finite precision it does not hold anymore, and values of the scaling parameter different from one (which corresponds to the classical round-to-nearest strategy), as we will show, allow for more accurate quantization. Based on this property, we show that the complex quantization problem we are interested in, which involves $m + n$ complex-valued scalar variables, where m and n are the dimensions of the input vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, can actually be reduced to the much simpler problem of finding the optimal complex-valued rescaling parameter λ minimizing a certain real-valued scalar function $f(\lambda)$.

We show that the minimization of f can be addressed by exploiting its geometric properties. We show that, just as in the real-valued case, f possesses important symmetries that allow us to restrict the search to a compact domain. Moreover, its piecewise constant behaviour enables us to search for the optimal value by sequentially testing the value of the function on the pieces.

We uncover however a fundamental difference with the real-valued case, which prevents us from easily finding the global minimum of f. Specifically, we characterize the constant regions of f as 2D regions in the complex plane delimited by certain *breaklines*. Such breaklines show a curious accumulation phenomenon, so that the number of regions on which f is constant is infinite, diverging from the real-valued case in which the algorithm could find the optimal value by testing a finite number of parameters.

By studying the behaviour of f near the points where this accumulation phenomenon occurs, we can however demonstrate, under mild assumptions, that the function is continuous at such points, and that its minimum exists.

Based on this analysis, we propose *CROQuant*, a Complex Rank-One Quantization algorithm adapted to this complex setting. Differently from the real-valued case and because of this accumulation phenomenon, it is not possible to simply evaluate the function f on all its constant regions. To solve this problem, we introduce a parameter, δ, which controls the number of regions visited. The resulting algorithm has a time complexity of $O\left(\min(m,n)^2 \max(m,n)\delta^2 2^{2t}\right)$, which is polynomial in the dimension m, n of the rank-one matrix and exponential in t, the number of significand bits. The

exponential dependence on t is tractable in this context because we are interested in small values of t, typically $t = 2, 3$ or 7 for applications in modern float formats typically used in machine learning and scientific computing. These values respectively correspond to the quarter precision Floating Point 8-bit also called `float8 E5M2`, the quarter precision `float8 E4M3` and the Brain Floating Point 16-bit also known as `bfloat16`.

We numerically test our algorithm on random rank-one matrices and on butterfly factors arising in the FFT. We demonstrate empirically that it is more accurate than the baseline RTN on random rank-one matrices. We also study the impact of the dimension of the matrices and of the parameter δ on the accuracy and on the computational time of our method. With increasing size of matrices the cost of the algorithm increases and the gains with respect to the baseline decrease. The use of the proposed strategy is thus beneficial especially with small matrices. This is exactly the case for butterfly matrices, for which the algorithm is applied to small (typically 2×2) blocks. However, even in the case of large matrices, evaluating the function f only on a limited number of regions of the complex plane (by setting typically $\delta = 2$) achieves a relatively fast and accurate solution.

We thus test the behaviour of CROQuant on the quantization of butterfly factorizations, by exploiting the same heuristic proposed in [12] to reduce the problem to a sequence of rank-one problems. We specifically focus on the FFT. We show that our algorithm provides quantization that is also more accurate than the naive rounding approach. To reach a given accuracy, our algorithm needs 30% fewer mantissa bits than the naive rounding method.

This work extends the preliminary work presented in [4]. The main new features concern the theoretical results given in Section 3, such as the continuity of the function f at some points and the proof that a minimizer of the problem exists under mild assumptions on $\boldsymbol{x}, \boldsymbol{y}$ and t. The quantization algorithm is also more efficient because it better exploits the geometrical properties of the function f to optimize it. Finally, we provide numerical experiments on the choice of the parameter δ based on the trade-off between the computation time and the quantization error.

The rest of the paper is organized as follows. In Section 2, we present the quantization problem for complex-valued rank-one matrices, we show how it can be reduced to the minimization of a scalar function through the rescaling invariances of the problem and we study the geometrical properties of this function. Then, in Section 3, we theoretically study the scalar function to demonstrate that it is continuous and its minimum exists. We use, in Section 4, the characterization of the quantization problem to build CROQuant, an algorithm finding an approximation of the optimal solution. In Section 5, we present numerical experiments on random rank-one matrices to compare our method and the baseline round-to-nearest. We apply our algorithm to the quantization of butterfly matrices (with a focus on the FFT) in Section 6. Finally, we summarize the contributions and we outline details for future works in Section 7.

Notations. $\mathbb{F}_t$ is a set of floating-point numbers with t-bit of significand, as in [12]. We denote vectors in lowercase boldface ($\boldsymbol{x}$) and matrices in uppercase boldface letters ($\boldsymbol{X}$). $\|\cdot\|$ is the Frobenius norm. Re, Im denote the real and imaginary parts of a complex number, which we quantize separately. We therefore define $\mathbb{CF}_t := \mathbb{F}_t + i\mathbb{F}_t$. We finally define the function $\mathrm{round}_t(\cdot)$ that maps the real and the imaginary parts of a complex number to their (set of) nearest neighbor(s) in $\mathbb{F}_t$. In most cases, there is a unique nearest neighbor, except in the case of tie. For this technical reason, we consider that round_t is a set-valued function. For a vector, the round_t function is applied element-wise.

2 PROBLEM STATEMENT AND SCALAR REDUCTION

Given a set of floating-point numbers with t-bit significand $\mathbb{F}_t$, the set $\mathbb{CF}_t := \mathbb{F}_t + i\mathbb{F}_t$, and two vectors[1] $\boldsymbol{x} \in \mathbb{C}^m$, $\boldsymbol{y} \in \mathbb{C}^n$, the quantization problem consists in finding vectors $\hat{\boldsymbol{x}} \in \mathbb{CF}_t^m, \hat{\boldsymbol{y}} \in \mathbb{CF}_t^n$ leading to a small approximation error,

[1] In practice these often come in some higher-precision floating point format $\mathbb{CF}_p$, $p > t$, i.e., $\boldsymbol{x} \in \mathbb{CF}_p^m$, $\boldsymbol{y} \in \mathbb{CF}_p^n$.

as measured by

$$C_{\boldsymbol{x},\boldsymbol{y}}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}) := \|\boldsymbol{x}\boldsymbol{y}^H - \hat{\boldsymbol{x}}\hat{\boldsymbol{y}}^H\|^2 . \tag{1}$$

When seeking minimizers of $C_{\boldsymbol{x},\boldsymbol{y}}$, one faces two potential difficulties: a) the function may not admit a minimizer; b) its minimization may be a too complex optimization problem, as it is expressed over $m+n$ elements of $\mathbb{CF}_t$, or (taking real and imaginary parts) $2(m+n)$ elements of $\mathbb{F}_t$.

In this section we focus on the second difficulty, showing that the minimization of $C_{\boldsymbol{x},\boldsymbol{y}}$ is equivalent to the optimization over a *single* complex rescaling parameter $\lambda \in \mathbb{C}$ of a particular real-valued function $f_{\boldsymbol{x},\boldsymbol{y},t}$, and that if λ^* minimizes $f_{\boldsymbol{x},\boldsymbol{y},t}(\lambda)$ then $\boldsymbol{x}^* = \mathrm{round}(\lambda^*\boldsymbol{x})$ and $\boldsymbol{y}^* = \mathrm{round}(\mu^*\boldsymbol{y})$ minimize $C_{\boldsymbol{x},\boldsymbol{y}}$, where μ^* has a simple expression in terms of $\boldsymbol{x}$ and λ^*. This reduction will be used in Section 3 to solve the first difficulty: by proving the existence of a minimizer λ^* (under mild assumptions on $\boldsymbol{x}$ and $\boldsymbol{y}$) we will deduce the existence of an optimal quantization $\boldsymbol{x}^*, \boldsymbol{y}^*$. The reduction will also lead in Section 4 to a quantization algorithm.

2.1 Reducing to a one-dimensional problem

We exploit the rescaling invariance properties of the problem to cast the minimization of $C_{\boldsymbol{x},\boldsymbol{y}}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$ into the minimization of a function of a single complex rescaling parameter λ.

Lemma 2.1. *Given $\boldsymbol{x} \in \mathbb{C}^m$, $\boldsymbol{y} \in \mathbb{C}^n$ and $t \geq 1$, it holds*

$$\inf_{\hat{\boldsymbol{x}}\in\mathbb{CF}_t^m, \hat{\boldsymbol{y}}\in\mathbb{CF}_t^n} C_{\boldsymbol{x},\boldsymbol{y}}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}) = \inf_{\lambda\in\mathbb{C}} f_{\boldsymbol{x},\boldsymbol{y},t}(\lambda) , \tag{2}$$

$$\text{with } f_{\boldsymbol{x},\boldsymbol{y},t}(\lambda) := \max_{\hat{\boldsymbol{x}}\in \mathrm{round}_t(\lambda\boldsymbol{x})} \|\boldsymbol{x}\boldsymbol{y}^H - \hat{\boldsymbol{x}}\,\mathrm{round}_t(\mu(\hat{\boldsymbol{x}})\boldsymbol{y})^H\|^2 , \tag{3}$$

$$\text{where } \mu(\hat{\boldsymbol{x}}) := \begin{cases} \frac{\langle \hat{\boldsymbol{x}}, \boldsymbol{x}\rangle}{\|\hat{\boldsymbol{x}}\|^2} & \text{if } \hat{\boldsymbol{x}} \neq 0 \\ 0 & \text{otherwise} . \end{cases} \tag{4}$$

Furthermore $\lambda^* \in \mathbb{C}$ is a minimizer[2] of $f_{\boldsymbol{x},\boldsymbol{y},t}$ if and only if $\hat{\boldsymbol{x}}^* \in \mathrm{round}(\lambda^*\boldsymbol{x})$ and $\hat{\boldsymbol{y}}^* \in \mathrm{round}(\mu(\hat{\boldsymbol{x}}^*)\boldsymbol{y})$ are minimizers of $C_{\boldsymbol{x},\boldsymbol{y}}$.

The proof in Appendix A mimics the one established for the real-valued case, cf. [12, Lemma 4.2]. For brevity, we will use C, f and round instead of $C_{\boldsymbol{x},\boldsymbol{y}}$, $f_{\boldsymbol{x},\boldsymbol{y},t}$ and round_t when $\boldsymbol{x}$, $\boldsymbol{y}$ and t are clear from context.

2.2 Geometric properties of the function f

We now focus on the study of the function f, and list several geometric properties useful to optimize it, namely: the function f is piecewise constant, with polygonal pieces delimited by *breaklines*, and it is invariant by simple transformations, enabling to restrict the search for a minimizer over $\lambda \in \mathbb{C}$ to a search over a simple compact trapezoidal domain $\Omega \subseteq \mathbb{C}$.

2.2.1 Invariances. The function f is invariant by multiplication by 2 and by i, as stated in the following lemma and as shown in Figure 1, left.

Lemma 2.2. *The function f defined in* (3) *satisfies $f(\cdot) = f(2\cdot) = f(i\cdot)$.*

[2]The *existence* of such a minimizer is equivalent to the existence of minimizers $\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{y}}^*$ of $C_{\boldsymbol{x},\boldsymbol{y}}$

The invariance by multiplication by two also appears in the real-valued case [12, Lemma 4.3]. Its proof, as well as the proof of the invariance by multiplication by i, is analogous to the proof for the real-valued case and is reported in Appendix A.

A direct consequence of these invariances is that the properties of f on $\mathbb{C}$ can be deduced from a study of f *restricted to any compact domain* $\Omega \subseteq \mathbb{C}$ *that generates a tiling of the complex plane* via dilations and quarter turn rotation (i.e., such that $\mathbb{C} = \cup_{j\in\mathbb{Z}} \cup_{\ell=0}^{3} i^\ell 2^j \Omega$). We will call such a domain a *tiling domain*. Figure 1 (left) presents two such potential tiling domains delimited in red (solid lines with stripes) and blue (dashed lines with dots). The most intuitive one is in blue: it is a quarter ring with inner and outer radii equal to 1 and 2, respectively. In contrast, the tiling domain in red[3] takes into account some geometric properties of the function f, which will be explained in the next section together with the computational advantages of choosing such a domain.

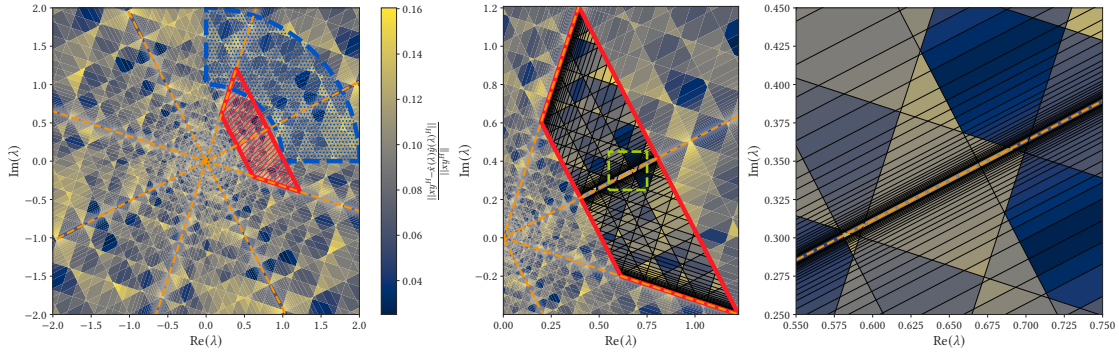

Fig. 1. Typical shape of the function $\lambda \in \mathbb{C} \mapsto f(\lambda)$ where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{C}^2$ were drawn with real and imaginary parts following a uniform distribution in $[0, 1]$ and $t = 3$. *Left*: Two potential tiling domains Ω are diplayed in red (solid lines with stripes) and blue (dashed lines with dots). Accumulation lines are displayed in orange with dashed lines. *Center*: Zoom on the red tiling domain $\Omega_t(\boldsymbol{x})$. Breaklines of the function $\lambda \in \Omega_t(\boldsymbol{x}) \mapsto \text{round}(\lambda \boldsymbol{x})$ are displayed in black with solid lines. Each breakline is associated with an integer $e \in \mathbb{Z}$, corresponding to the exponent in the offset $\beta = \left(k + \frac{1}{2}\right) 2^{e-t} \in \mathbb{M}_t^*$ (here, $e \geq -5$). *Right*: Zoom in the green square (with dashed lines) of the middle figure with more breaklines to observe the accumulation phenomenon, i.e., $e \geq -8$.

2.2.2 Piecewise constant nature of f. Since $\lambda \in \mathbb{C} \mapsto \text{round}(\lambda \boldsymbol{x}) \in \mathbb{C}$ is piecewise constant, f is also piecewise constant, so finding a minimizer on $\mathbb{C}$ (or on any compact tiling domain Ω discussed in the previous section) amounts to find on which of the corresponding piece(s) f attains the smallest value. The simplest way to individuate these piece(s) is to study the *breakpoints* of the function $\lambda \in \mathbb{C} \mapsto \text{round}(\lambda \boldsymbol{x})$ where $\boldsymbol{x} \in \mathbb{C}^m$, which are organized into *breaklines*.

Definition 2.3 (Breakpoints). Consider $\boldsymbol{x} \in \mathbb{C}^m$. A scalar $\lambda \in \mathbb{C}$ is a breakpoint of the function $\lambda \mapsto \text{round}(\lambda \boldsymbol{x})$ if there *does not* exist a neighborhood of λ in which $\text{round}(\lambda \boldsymbol{x})$ is constant.

As characterized next, breakpoints correspond to λ values for which there is at least one entry $\boldsymbol{x}_j$ such that $\text{round}(\lambda \boldsymbol{x}_j)$ corresponds to a *tie* in the choice of the nearest neighbor for the real or the imaginary part.

Lemma 2.4. *Given $\boldsymbol{x} \in \mathbb{C}^m$ and $t \geq 1$, we denote $\mathbb{M}_t := \mathbb{M}_t^* \cup \{0\}$ the set of* midpoints *of $\mathbb{F}_t$, where*

$$\mathbb{M}_t^* := \left\{ s\left(k + \frac{1}{2}\right) 2^{e-t} : s \in \{-1, 1\}, k \in [\![2^{t-1}, 2^t - 1]\!], e \in \mathbb{Z} \right\}, \tag{5}$$

[3] we denote this tiling domain $\Omega_t(\boldsymbol{x})$. Its construction will be addressed in Section 4.1.

and we denote $\mathbb{C}_{\boldsymbol{x}}$ *the following set of* directions*:*

$$\mathbb{C}_{\boldsymbol{x}} := \{i^l \boldsymbol{x}_j : l \in \{0,1\}, j \in [\![1,m]\!], \boldsymbol{x}_j \neq 0\} \subseteq \mathbb{C}\,. \tag{6}$$

A parameter $\lambda \in \mathbb{C}$ *is a breakpoint of the function* $\lambda \mapsto \mathrm{round}(\lambda \boldsymbol{x})$ *if and only if there exist an offset* $\beta \in \mathbb{M}_t$ *and a direction* $z \in \mathbb{C}_{\boldsymbol{x}}$ *such that* λ *belongs to the line* $\mathbb{D}(z,\beta)$ *(called a* breakline*), defined as*

$$\mathbb{D}(z,\beta) := \{\mu \in \mathbb{C} : \mathrm{Re}(\mu z) = \beta\}\,. \tag{7}$$

Proof. First, it is easy to check that $\lambda \in \mathbb{C}$ is a breakpoint for $\mu \mapsto \mathrm{round}(\mu \boldsymbol{x})$ if, and only if, there exists at least one *nonzero* coordinate $\boldsymbol{x}_j$ such that it is a breakpoint of $\mu \mapsto \mathrm{round}(\mu \boldsymbol{x}_j)$. Second, the latter holds if, and only if, it is either a breakpoint of $\mu \mapsto \mathrm{round}(\mathrm{Re}(\mu \boldsymbol{x}_j))$ or of $\mu \mapsto \mathrm{round}(\mathrm{Im}(\mu \boldsymbol{x}_j)) = \mathrm{round}(\mathrm{Re}(\mu i \boldsymbol{x}_j))$ (or both). By the definition of $\mathbb{C}_{\boldsymbol{x}}$ this holds if and only if there is $z \in \mathbb{C}_{\boldsymbol{x}}$ such that λ is a breakpoint of $\mu \mapsto \mathrm{round}(\mathrm{Re}(\mu z))$, i.e., such that $\mathrm{Re}(\lambda z)$ is a breakpoint of $u \in \mathbb{R} \mapsto \mathrm{round}(u)$. By the definition of $\mathbb{M}_t$, this exactly means that $\beta := \mathrm{Re}(\lambda z) \in \mathbb{M}_t$. □

In words, Lemma 2.4 shows that breakpoints are structured as a union of breaklines in the complex plane. These breaklines are directed by elements of $\mathbb{C}_{\boldsymbol{x}}$, and with offset defined by the midpoints of $\mathbb{F}_t$. Each breakline $\mathbb{D}(z,\beta)$ is indeed a line, corresponding to a rotation of $-\arg z$ of the line defined via the equation $\mathrm{Re}(\lambda) = \frac{\beta}{|z|}$. Breaklines (intersected with the chosen tiling domain $\Omega_t(\boldsymbol{x})$) are shown in black with solid lines in Figure 1 (center). Since the function f is constant on each piece delimited by such breaklines, evaluating it in the centroids will suffice.

2.2.3 Accumulation phenomenon. Since zero is an accumulation point of $\mathbb{M}_t$ (see (5)) the lines $\mathbb{D}(z,0)$, $z \in \mathbb{C}_{\boldsymbol{x}}$, which all cross the origin (and are distinct for directions $z \in \mathbb{C}_{\boldsymbol{x}}$ with distinct argument), play a particular role: they are "accumulation lines" of the infinitely many parallel lines $\mathbb{D}(z,\beta)$, $\beta \in \mathbb{M}_t^*$. This contrasts with all other lines $\mathbb{D}(z,\beta)$, $\beta \in \mathbb{M}_t^*$ which are somehow isolated. Accumulation lines are depicted in orange with dashed lines in Figure 1.

Definition 2.5 (Accumulation lines). Given $\boldsymbol{x} \in \mathbb{C}^m$, each breakline $\mathbb{D}(z,0)$, $z \in \mathbb{C}_{\boldsymbol{x}}$ is called an *accumulation line*. Their union is denoted

$$\mathbb{A} := \bigcup_{z \in \mathbb{C}_{\boldsymbol{x}}} \mathbb{D}(z,0)\,. \tag{8}$$

For practical purposes, we denote $\mathbb{A}_z := \mathbb{D}(z,0)$ the accumulation line associated to the direction $z \in \mathbb{C}_{\boldsymbol{x}}$. By the very definition of $\mathbb{C}_{\boldsymbol{x}}$, it also comes with a "twin" accumulation line, associated to iz, which is orthogonal to it. Figure 1 (right) also illustrates the accumulation phenomenon around the accumulation lines displayed in orange with dashed lines. The collection of *ordinary* breaklines (that are *not* accumulation lines) is denoted

$$\mathcal{D}^* := \{\mathbb{D}(z,\beta) : z \in \mathbb{C}_{\boldsymbol{x}}, \beta \in \mathbb{M}_t^*\}\,. \tag{9}$$

An important consequence of this accumulation phenomenon is that, in contrast to the real-valued case, the number of pieces of any tiling domain on which f is constant is infinite. This significantly complicates the proof of the existence of a global minimizer of f, conducted in Section 3, and will also have a strong impact on the quantization algorithm that we will propose in Section 4.

3 EXISTENCE OF A MINIMIZER DESPITE INFINITELY MANY PIECES

In this section, we show that (under technical but mild non-degeneracy conditions on $\boldsymbol{x}, \boldsymbol{y}, t$, described in Section 3.1) despite its infinite number of constant pieces on every tiling domain Ω, the functions f *admits a minimizer*. To do so,

we prove in Section 3.2 that f is continuous at every point lying on an accumulation line but not on any ordinary breakline, i.e., on the set $\mathbb{A} \setminus \mathbb{B}$, with

$$\mathbb{B} := \{\lambda \in \mathbb{A} : \exists \mathbb{D} \in \mathcal{D}^*, \lambda \in \mathbb{D}\}\,. \tag{10}$$

This set is the finite union $\cup_{z\in\mathbb{C}_{\boldsymbol{x}}} \mathbb{A}_z \setminus \mathbb{B}_z$, with $\mathbb{B}_z := \{\lambda \in \mathbb{A}_z : \exists \mathbb{D} \in \mathcal{D}, \lambda \in \mathbb{D}\}$. Moreover, $\mathbb{A}_z \setminus \mathbb{B}_z$ itself is a disjoint union of *elementary (line) segments*

$$(\lambda_-, \lambda_+) := \{t\lambda_- + (1-t)\lambda_+ : 0 < t < 1\}, \quad \text{with } \lambda_-, \lambda_+ \in \mathbb{B}_z \text{ s.t. } (\lambda_-, \lambda_+) \cap \mathbb{B}_z = \emptyset\,, \tag{11}$$

that correspond to portions of accumulation lines between two consecutive intersections $\lambda_-, \lambda_+ \in \mathbb{B}_z$ with a breakline (see Figure 2). As a consequence of continuity we will show in Section 3.3 that the minimum of f indeed exists.

3.1 Non-degeneracy assumption and its consequences

To prove that f is continuous on the set $\mathbb{A} \setminus \mathbb{B}$, we need to control the behaviour of the rounded vectors

$$\hat{\boldsymbol{x}}(\lambda) \in \mathrm{round}(\lambda \boldsymbol{x}) \quad \text{and} \quad \hat{\boldsymbol{y}}(\lambda) \in \mathrm{round}(\mu(\hat{\boldsymbol{x}}(\lambda))\boldsymbol{y})\,,$$

for λ close to $\mathbb{A} \setminus \mathbb{B}$. For this, we introduce *non-degeneracy assumptions*:

Definition 3.1 (Non-degenerate vectors). Consider $\boldsymbol{x} \in \mathbb{C}^m$, $\boldsymbol{y} \in \mathbb{C}^n$, and $t \geq 1$. The triplet $\boldsymbol{x}, \boldsymbol{y}, t$ is non-degenerate if both of the two following properties hold:[4]

(1) at most one ordinary breakline intersects an accumulation line at any point:

$$\forall \lambda \in \mathbb{B}, \quad |\{\mathbb{D} \in \mathcal{D}^* : \lambda \in \mathbb{D}\}| = 1\,;$$

(2) for each $\lambda \in \mathbb{A} \setminus \mathbb{B}$, $\mathrm{round}(\lambda \boldsymbol{x}) = \{\hat{\boldsymbol{x}}(\lambda)\}$ is a singleton, and the (set valued) function $\mu \mapsto \mathrm{round}(\mu \boldsymbol{y})$ is locally constant around the scalar $\mu(\hat{\boldsymbol{x}}(\lambda))$ defined in (4) (hence this scalar is *not* a breakpoint of this function).

We check numerically (see the beginning of Section 5) that this non-degeneracy assumption is valid with probability one when the real/imaginary parts of $\boldsymbol{x}, \boldsymbol{y}$ are drawn i.i.d. according to a continuous distribution.

This non-degeneracy assumption allows to introduce a family of *trapezoids* defined by the breaklines (see Figure 2 left for a visual representation of such a domain) that will be essential for the continuity analysis.

Definition 3.2. Given non-degenerate $\boldsymbol{x}, \boldsymbol{y}, t$, consider $z \in \mathbb{C}_{\boldsymbol{x}}$ and (λ_-, λ_+) a corresponding elementary segment, cf. (11). For $\varepsilon > 0$, denote $\mathcal{T}(\lambda_-, \lambda_+, \varepsilon)$ the *open* trapezoid with parallel sides defined as the lines parallel to the line $\mathbb{A}_z$ generated by the segment and at distance ε from this line; its other sides being the unique[5] ordinary breaklines $\mathbb{D}_\pm$ such that $\lambda_\pm \in \mathbb{D}_\pm$. Given $\lambda \in \mathbb{A} \setminus \mathbb{B}$ we also denote $\mathcal{T}(\lambda, \varepsilon) := \mathcal{T}(\lambda_-, \lambda_+, \varepsilon)$ with (λ_-, λ_+) the unique elementary segment such that $\lambda \in (\lambda_-, \lambda_+)$.

We will use the following key facts.

Lemma 3.3. *Consider non-degenerate $\boldsymbol{x}, \boldsymbol{y}, t$ and (λ_-, λ_+), $z \in \mathbb{C}_{\boldsymbol{x}}$ as in Definition 3.2. Denote I the set of indices k such that $\boldsymbol{x}_k/z \in \mathbb{R} \cup i\mathbb{R}$. There is $\varepsilon \in (0, \infty]$ such that, for every $\lambda \in (\lambda_-, \lambda_+)$ and $\lambda' \in \mathcal{T}(\lambda_-, \lambda_+, \varepsilon)$:*

- *for each $\hat{\boldsymbol{x}}(\lambda) \in \mathrm{round}(\lambda \boldsymbol{x})$ and $\hat{\boldsymbol{x}}(\lambda') \in \mathrm{round}(\lambda' \boldsymbol{x})$*

$$\|\hat{\boldsymbol{x}}(\lambda') - \hat{\boldsymbol{x}}(\lambda)\| \leq (1 + v_t) \cdot \|\boldsymbol{x}_I\| \cdot |\,\mathrm{Re}(\lambda' z/|z|)|\,, \tag{12}$$

[4]Recall that, by definition, the sets $\mathcal{D}$, $\mathbb{A}$, and $\mathbb{B}$ are all dependent on $\boldsymbol{x}$ and t. For the sake of brevity this is not explicited in the corresponding notations.
[5]thanks to the non-degeneracy of $\boldsymbol{x}, \boldsymbol{y}, t$

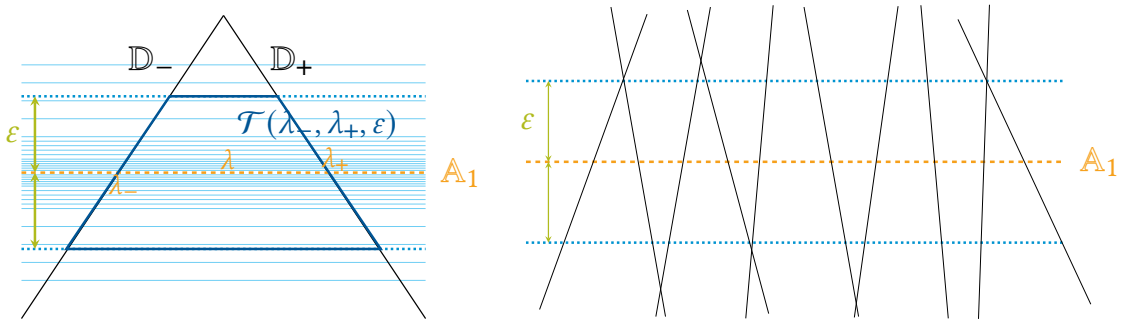

Fig. 2. *Left*: Visual representation of a trapezoid $\mathcal{T}(\lambda_-, \lambda_+, \varepsilon)$ where the *light* solid blue lines are the breaklines parallel to $\mathbb{A}_1$. *Right*: Visual representation of the neighbourhood of an accumulation line for a non-degenerate vector (without displaying the breaklines parallel to $\mathbb{A}_1$).

where $v_t := u_t/(1+u_t)$ *and* $u_t := 2^{-t}$ *is the unit roundoff of* $\mathbb{F}_t$ *[13, Thm. 2.2];*

- $\hat{\boldsymbol{y}}(\lambda')$ *is constant on* $\mathcal{T}(\lambda_-, \lambda_+, \varepsilon)$ *and we denote its value as* $\hat{\boldsymbol{y}}(\lambda_-, \lambda_+)$.

We denote $\varepsilon(\lambda_-, \lambda_+) \in (0, \infty]$ *the largest such* ε.

Proof. It is easy to show that for small enough $\varepsilon_1 > 0$ the open trapezoid $\mathcal{T}_1 := \mathcal{T}(\lambda_-, \lambda_+, \varepsilon_1)$ does not intersect any breakline $\mathbb{D}$ except those parallel to $\mathbb{A}_z$. Besides, we observe that if $i^\ell \boldsymbol{x}_k \in \mathbb{C}$ (with $\ell \in \{0, 1\}$) is *not* aligned with z, then $\lambda' \in \mathcal{T}_1 \mapsto \mathrm{round}(\mathrm{Re}(\lambda' i^\ell \boldsymbol{x}_k))$ is constant (and is a singleton). We now consider the possible values of $\hat{\boldsymbol{x}}_k(\lambda') - \hat{\boldsymbol{x}}_k(\lambda)$ depending on k:

- if $\boldsymbol{x}_k = 0$: trivially $\hat{\boldsymbol{x}}_k(\lambda') = \hat{\boldsymbol{x}}_k(\lambda) = 0$;
- if $\boldsymbol{x}_k/z \notin \mathbb{R}$ *and* $\boldsymbol{x}_k/z \notin i\mathbb{R}$: treating real and imaginary parts separately yields $\hat{\boldsymbol{x}}_k(\lambda') = \hat{\boldsymbol{x}}_k(\lambda)$ since $\lambda, \lambda' \in \mathcal{T}_1$;
- if $\boldsymbol{x}_k \neq 0$ and *either* $\boldsymbol{x}_k/z \in \mathbb{R}$ *or* $\boldsymbol{x}_k/z \in i\mathbb{R}$: then either $\mathrm{Im}(\hat{\boldsymbol{x}}_k(\lambda')) = \mathrm{Im}(\hat{\boldsymbol{x}}_k(\lambda))$ or the same holds with $\mathrm{Im}(\cdot)$ replaced by $\mathrm{Re}(\cdot)$; without loss of generality consider the first case, which occurs if $0 \neq \rho_k := \boldsymbol{x}_k/z \in \mathbb{R}$. Since $\lambda \in (\lambda_-, \lambda_+) \subset \mathbb{A}_z$ we have $\mathrm{Re}(\lambda z) = 0$ hence $\mathrm{Re}(\hat{\boldsymbol{x}}_k(\lambda)) = \mathrm{round}(\mathrm{Re}(\lambda \boldsymbol{x}_k)) = \mathrm{round}(\rho_k \cdot \mathrm{Re}(\lambda z)) = 0$. Since $\boldsymbol{x}_k/z \notin i\mathbb{R}$, the same reasoning as above yields $\mathrm{Im}(\hat{\boldsymbol{x}}_k(\lambda')) = \mathrm{round}(\mathrm{Re}(\lambda' i\boldsymbol{x}_k)) = \mathrm{round}(\mathrm{Re}(\lambda i \boldsymbol{x}_k)) = \mathrm{Im}(\hat{\boldsymbol{x}}_k(\lambda))$, so that

$$\begin{aligned} |\hat{\boldsymbol{x}}_k(\lambda') - \hat{\boldsymbol{x}}_k(\lambda)| &= |\mathrm{Re}(\hat{\boldsymbol{x}}_k(\lambda'))| = |\mathrm{round}(\mathrm{Re}(\lambda' \boldsymbol{x}_k))| \\ &\leq (1+v_t)|\mathrm{Re}(\lambda' \boldsymbol{x}_k)| = (1+v_t) \cdot |\boldsymbol{x}_k| \cdot |\mathrm{Re}(\lambda' z/|z|)|\,, \end{aligned}$$

where we used that $|\mathrm{round}(u)| \leq (1+v_t)|u|$ with $v_t := u_t/(1+u_t)$ and $u_t := 2^{-t}$ the unit roundoff of $\mathbb{F}_t$ [13, Thm. 2.2].

This concludes the proof of (12).

Furthermore, since $\boldsymbol{x}, \boldsymbol{y}, t$ is non-degenerate, $\mu \mapsto \mathrm{round}(\mu \boldsymbol{y})$ is locally constant around $\mu(\hat{\boldsymbol{x}}(\lambda))$ for each $\lambda \in (\lambda_-, \lambda_+)$. By (12) and the continuity of $\mu(\hat{x})$ away from the origin (see (4)) we deduce the existence of $\varepsilon_2 > 0$ such that if $\lambda' \in \mathcal{T}_1$ satisfies $|\mathrm{Re}(\lambda' z)| \leq \varepsilon_2$ then $\hat{\boldsymbol{y}}(\lambda') = \mathrm{round}(\mu(\hat{\boldsymbol{x}}(\lambda')\boldsymbol{y}) = \mathrm{round}(\mu(\hat{\boldsymbol{x}}(\lambda)\boldsymbol{y}) = \hat{\boldsymbol{y}}(\lambda)$. Since $\mathcal{T}(\lambda_-, \lambda_+, \varepsilon) = \mathcal{T}_1 \cap \{\lambda' : |\mathrm{Re}(\lambda' z)| \leq \varepsilon_2\}$, we conclude by setting $\varepsilon := \min(\varepsilon_1, \varepsilon_2)$. □

3.2 Continuity of f on $\mathbb{A} \setminus \mathbb{B}$

We can now state our continuity result.

Proposition 3.4. *Assume that $\boldsymbol{x} \in \mathbb{C}^m$, $\boldsymbol{y} \in \mathbb{C}^n$ and $t \geq 1$ are non-degenerate. Then the function f defined in* (3) *is continuous on $\mathbb{A} \setminus \mathbb{B}$ where we recall that $\mathbb{A}$ and $\mathbb{B}$ are respectively defined in* (8) *and* (10).

It follows straightforwardly from a stronger result which proof is slightly postponed.

Lemma 3.5. *Given non-degenerate $\boldsymbol{x}, \boldsymbol{y}, t$ and (λ_-, λ_+), $z \in \mathbb{C}_{\boldsymbol{x}}$ as in Definition 3.2, and $\varepsilon := \varepsilon(\lambda_-, \lambda_+)$ from Lemma 3.3, the function f defined in* (3) *satisfies*

$$|f(\lambda) - f(\lambda')| \leq L \cdot |\operatorname{Re}(\lambda' z/|z|)| \quad \forall \lambda \in (\lambda_-, \lambda_+), \quad \forall \lambda' \in \mathcal{T}(\lambda_-, \lambda_+, \varepsilon)\,, \tag{13}$$

for some $L < \infty$.

Observe that $\operatorname{Re}(\lambda' z/|z|) = 0$ if, and only if $\lambda' \in \mathbb{A}_z$, and in fact $|\operatorname{Re}(\lambda' z/|z|)|$ is the Euclidean distance from λ' to $\mathbb{A}_z$.

Corollary 3.6. *Consider f defined with non-degenerate $\boldsymbol{x}, \boldsymbol{y}, t$. If $\Omega \cap \mathbb{A}$ is bounded and bounded away from zero then there are $\varepsilon_{\boldsymbol{x},\boldsymbol{y},t} > 0$ and $0 < L_{\boldsymbol{x},\boldsymbol{y},t} < +\infty$ such that*

$$|f(\lambda) - f(\lambda')| \leq L_{\boldsymbol{x},\boldsymbol{y},t} \cdot |\operatorname{Re}(\lambda' z/|z|)| \quad \forall \lambda \in (\mathbb{A} \setminus \mathbb{B}) \cap \Omega, \quad \forall \lambda' \in \mathcal{T}(\lambda, \varepsilon_{\boldsymbol{x},\boldsymbol{y},t})\,.$$

Proof. The assumptions on Ω imply that $\Omega \cap \mathbb{A}$ is covered by a finite union of elementary segments. The result thus holds with $\varepsilon_{\boldsymbol{x},\boldsymbol{y},t}$ the minimum of $\varepsilon(\lambda_-, \lambda_+)$ over such segments, and $L_{\boldsymbol{x},\boldsymbol{y},t}$ the maximum of the corresponding Lipschitz-like constants. □

Proof of Lemma 3.5. Denote $\hat{\boldsymbol{x}}(\lambda)$ any element of $\operatorname{round}(\lambda \boldsymbol{x})$, and $\hat{\boldsymbol{y}}(\lambda)$ any element of $\operatorname{round}(\mu(\hat{\boldsymbol{x}}(\lambda)\boldsymbol{y})$, and similarly with λ'. Since $f(\cdot)$ is the max over such pairs of $\|\Delta(\cdot)\|$ where $\Delta(\cdot) := \boldsymbol{x}\boldsymbol{y}^H - \hat{\boldsymbol{x}}(\cdot)\hat{\boldsymbol{y}}(\cdot)^H$, it is enough to bound

$$\begin{aligned}
\left|\|\Delta(\lambda')\|^2 - \|\Delta(\lambda)\|^2\right| &= \left|\|\Delta(\lambda') - \Delta(\lambda) + \Delta(\lambda)\|^2 - \|\Delta(\lambda)\|^2\right| \\
&= \left|\|\Delta(\lambda' - \Delta(\lambda)\|^2 + 2\operatorname{Re}\langle \Delta(\lambda), \Delta(\lambda') - \Delta(\lambda)\rangle_F\right| \\
&\leq \|\Delta(\lambda') - \Delta(\lambda)\|^2 + 2|\operatorname{Re}\langle \Delta(\lambda), \Delta(\lambda') - \Delta(\lambda)\rangle_F| \\
&\leq \|\Delta(\lambda') - \Delta(\lambda)\| \left(2\|\Delta(\lambda)\| + \|\Delta(\lambda') - \Delta(\lambda)\|\right)\,.
\end{aligned} \tag{14}$$

The first inequality above is obtained with the triangular inequality and the second one with the fact that Re is 1-Lipschitz and the Cauchy-Schwarz inequality. As we are only considering rank-one matrices, we further get

$$\begin{aligned}
\|\Delta(\lambda') - \Delta(\lambda)\| &= \|\hat{\boldsymbol{x}}(\lambda')\hat{\boldsymbol{y}}(\lambda')^H - \hat{\boldsymbol{x}}(\lambda)\hat{\boldsymbol{y}}(\lambda)^H\| \\
&\leq \|\hat{\boldsymbol{x}}(\lambda')\| \cdot \|\hat{\boldsymbol{y}}(\lambda') - \hat{\boldsymbol{y}}(\lambda)\| + \|\hat{\boldsymbol{y}}(\lambda)\| \cdot \|\hat{\boldsymbol{x}}(\lambda') - \hat{\boldsymbol{x}}(\lambda)\|\,.
\end{aligned} \tag{15}$$

Since $\lambda \in (\lambda_-, \lambda_+)$ and $\lambda' \in \mathcal{T}(\lambda_-, \lambda_+, \varepsilon)$, by Lemma 3.3 we have

$$\|\hat{\boldsymbol{x}}(\lambda') - \hat{\boldsymbol{x}}(\lambda)\| \leq \|\boldsymbol{x}_I\| \cdot (1 + v_t) \cdot |\operatorname{Re}(\lambda' z/|z|)|\,,$$

and $\hat{\boldsymbol{y}}(\lambda') = \hat{\boldsymbol{y}}(\lambda) = \hat{\boldsymbol{y}}(\lambda_-, \lambda_+)$ irrespective of the particular λ, λ', thus (15) becomes

$$\|\Delta(\lambda') - \Delta(\lambda)\| \leq \underbrace{\|\boldsymbol{x}_I\| \cdot (1 + v_t) \cdot \|\hat{\boldsymbol{y}}(\lambda)\|}_{=:L_1} \cdot |\operatorname{Re}(\lambda' z/|z|)|\,.$$

Since $|\mathrm{Re}(\lambda' z/|z|)|$ is the Euclidean distance from λ' to $\mathbb{A}_z$, by Definition 3.2 it is bounded by ε for every $\lambda' \in \mathcal{T}(\lambda_-, \lambda_+, \varepsilon)$. Moreover, since $\lambda \in (\lambda_-, \lambda_+) \subseteq \mathcal{T}(\lambda_-, \lambda_+, \varepsilon)$, $\hat{\boldsymbol{x}}(\lambda)$ and $\hat{\boldsymbol{y}}(\lambda)$ are singletons, so $\Delta(\lambda)$ is constant and we denote its value by $\Delta(\lambda_-, \lambda_+)$.

$$2\|\Delta(\lambda)\| + \|\Delta(\lambda') - \Delta(\lambda)\| \leq 2\|\Delta(\lambda_-, \lambda_+)\| + L_1 \cdot \varepsilon =: L_2 \,.$$

Combining these inequalities with (14), taking the needed supremum over $\hat{\boldsymbol{x}}(\cdot)$, and noting that $\hat{\boldsymbol{y}}(\cdot)$ is constant over the considered trapezoid, we obtain

$$|f(\lambda') - f(\lambda)| \leq L_1 \cdot L_2 \cdot |\mathrm{Re}(\lambda' z/|z|)| \,.$$

□

3.3 Existence of a minimizer of the function f

From the continuity of f on $\mathbb{A} \setminus \mathbb{B}$ and its invariances, we deduce that its admits a global minimum.

Proposition 3.7. *If $\boldsymbol{x}, \boldsymbol{y}, t$ is non-degenerate, then f admits a global minimum.*

Proof. By the invariances of f, it is enough to prove that it has a global minimum on a tiling domain Ω. Without loss of generality consider $\Omega = \Omega_t(\boldsymbol{x})$ and denote Ω_k^{far}, $k \geq 1$, the set of points in Ω that are at distance at least $\frac{1}{k}$ from any accumulation line. Decomposing Ω into two pieces $\Omega = \left(\bigcup_{k\in\mathbb{N}} \Omega_k^{\mathrm{far}}\right) \cup (\mathbb{A} \cap \Omega)$ yields

$$\inf_{\lambda\in\Omega} f(\lambda) = \min\Big(\inf_{k\geq 1} \underbrace{\inf_{\lambda\in\Omega_k^{\mathrm{far}}} f(\lambda)}_{=:m_k}, \underbrace{\inf_{\lambda\in\mathbb{A}\cap\Omega} f(\lambda)}_{=:m_{\mathbb{A}}} \Big) \,. \tag{16}$$

Since $\Omega = \Omega_t(\boldsymbol{x})$ is bounded away from the origin, we deduce from the structure of $\mathbb{A}$ (see notably Lemma 2.4) that $\mathbb{A} \cap \Omega$ is included in a finite union of line segments (λ_-, λ_+). As f is constant on each segment, there is $\lambda_{\mathbb{A}} \in \Omega \cap \mathbb{A}$ such that $m_{\mathbb{A}} = f(\lambda_{\mathbb{A}})$.

Similarly, since f is piecewise constant on Ω_k^{far} with a finite number of discontinuities, each m_k, which is defined as an infimum, is actually a minimum reached at some $\lambda_k \in \Omega_k^{\mathrm{far}}$. Moreover, since by construction $\Omega_k^{\mathrm{far}} \subset \Omega_{k+1}^{\mathrm{far}}$ for all $k \geq 1$, the sequence $(m_k)_{k\geq 1}$ is non-increasing and bounded from below by 0 (since f is non-negative). By the monotone convergence theorem, it converges to its infimum, m_∞.

We distinguish two cases: if $m_{\mathbb{A}} \leq m_\infty$, then $\lambda_{\mathbb{A}}$ is a global minimizer of f. Otherwise $m_{\mathbb{A}} > m_\infty$, and we prove below that the sequence m_k must stabilize after a certain rank K. Thus $m_\infty = m_K = f(\lambda_K)$ is the global minimum of f.

Proof that m_k stabilizes after a certain rank when $m_{\mathbb{A}} > m_\infty$. We proceed by contradiction: assuming this is not the case we can define a sequence of sets $(\Lambda_k)_{k\geq 1} \subseteq \Omega$ such that for any $k \geq 1$, $\forall \lambda \in \Lambda_k$, $f(\lambda) = m_k$. We will prove below that there must then exist $\lambda_\infty \in \mathbb{A} \setminus \mathbb{B}$ that is a limit point of elements of $(\Lambda_k)_{k\geq 1}$. By the continuity of f on $\mathbb{A} \setminus \mathbb{B}$ (Proposition 3.4) and the fact that $(m_k)_{k\geq 1}$ a converges, we get

$$m_\infty = \lim_{k\to+\infty} m_k = \lim_{k\to+\infty} f(\lambda_k) = f(\lambda_\infty) \overset{\lambda_\infty\in\mathbb{A}}{\geq} m_{\mathbb{A}} \,.$$

which contradicts the assumption that $m_{\mathbb{A}} > m_\infty$.

To build λ_∞, observe that since $(m_k)_{k\geq 1}$ does not stabilize, it strictly decreases an infinite number of times, at indexes k_j. For all $j \geq 1$, $m_{k_j} > m_{k_j+1}$, so by definition of Λ_k and m_{k_j} we have $\Lambda_{k_j+1} \subseteq (\Omega_{k_j}^{\mathrm{far}})^c$. By definition of the sets Ω_k^{far} this implies $d(\Lambda_{k_{j+1}}, \mathbb{A}) < \frac{1}{k_j}$ hence $\lim_{j\to\infty} d(\Lambda_{k_j}, \mathbb{A}) = 0$. Since each set Λ_{k_j} is a subset of $\Omega = \Omega_t(\boldsymbol{x})$, which is

compact, we can extract a subsequence which converges towards some limit denoted by λ_∞. For the sake of simplicity we still index this subsequence with k_j. Since $\mathbb{A}$ is a finite union of accumulation lines, it is a closed set, so $\lambda_\infty \in \mathbb{A}$. If $\lambda_\infty \in \mathbb{A} \setminus \mathbb{B}$ there is nothing left to prove, so we now treat the case where $\lambda_\infty \in \mathbb{B}$. In this case there is $z \in \mathbb{C}_{\boldsymbol{x}}$ such that $\lambda_\infty \in \mathbb{B}_z$, and λ_∞ is thus the endpoint of two adjacent (open) elementary segments $(\lambda_-, \lambda_\infty)$ and $(\lambda_\infty, \lambda_+)$, both included in $\mathbb{A}_z$. We will show that it is possible to build another subsequence that converges either to λ'_∞ that is the mid-point of one of these line segments, and therefore belongs to $\mathbb{A} \setminus \mathbb{B}$. For this, consider $\varepsilon := \min(\varepsilon(\lambda_-, \lambda_\infty), \varepsilon(\lambda_\infty, \lambda_+))$ and $\mathcal{T} := \mathcal{T}(\lambda_-, \lambda_\infty, \varepsilon) \cup \mathcal{T}(\lambda_-, \lambda_\infty, \varepsilon)$. From some index onwards, Λ_{k_j} must intersect $\mathcal{T}$, which means that it must intersect one of its two constituent trapezoids. At least one of the two trapezoids (without loss of generality assume this is $\mathcal{T}(\lambda_-, \lambda_\infty, \varepsilon)$) is thus intersected infinitely many times, and once again this leads to the extraction of a subsequence still denoted k_j. Due to the structure of the constant pieces of f on $\mathcal{T}(\lambda_-, \lambda_\infty, \varepsilon)$, its intersection with Λ_{k_j} contains a whole "band" (see Figure 2) parallel to the line segment $\mathcal{T}(\lambda_-, \lambda_\infty, \varepsilon)$. For large enough j, the midpoint of this band is arbitrarily close to the midpoint of $\mathcal{T}(\lambda_-, \lambda_\infty, \varepsilon)$, hence the conclusion. □

4 CONSTRUCTION OF A PRACTICAL ALGORITHM

In this section, we use the characterization of the optimal solution given in Lemma 2.1 in order to design an algorithm that finds a (possibility approximate) minimizer of the function f.

As in the real-valued case [12], since f is piecewise constant, minimizing it amounts to find on which of the *polygonal pieces* (delimited by its breaklines) it attains the smallest value. In the complex-valued case, one also needs to consider the (constant) values of f on the finitely many *line segments* in $\mathbb{A} \setminus \mathbb{B}$, i.e., lying between the intersections of the accumulation lines and the breaklines. Compared to the real-valued case, a difficulty stems from the fact that, even in a compact tiling domain Ω, the number of polygonal pieces is infinite. Indeed, the infinite number of possible offsets $\beta \in \mathbb{M}_t^*$ generates an infinite number of breaklines. However, we can write $\mathbb{M}_t^*$ as a union of finite sets:

$$\mathbb{M}_t^* = \bigcup_{e \in \mathbb{Z}} \mathbb{M}_t^e \quad \text{where } \mathbb{M}_t^e := \left\{ s\left(k + \frac{1}{2}\right) 2^{e-t} : s \in \{-1, 1\}, k \in [\![2^{t-1}, 2^t - 1]\!] \right\}.$$

This expression invites us to consider, for each $e \in \mathbb{Z}$, the collection $\mathcal{D}^e$ of breaklines "of degree e" (with offsets in $\mathbb{M}_t^e$) that intersect $\Omega_t(\boldsymbol{x})$. We first define $e_{\max} \in \mathbb{Z}$, the largest integer such that $\mathcal{D}^{e_{\max}} \neq \emptyset$. For example, in Figure 4, we have $e_{\max} := 2$. For the practical algorithm, we also define for each $e \leq e_{\max}$:

- the finite set $\mathcal{D}^{\geq e} := \cup_{e_{\max} \geq e' \geq e} \mathcal{D}^{e'}$ of all breaklines "of degree at least e", with offsets in $\mathbb{M}_t^{\geq e} := \cup_{e_{\max} \geq e' \geq e} \mathbb{M}_t^{e'}$, *that intersect* $\Omega_t(\boldsymbol{x})$;
- the finite set $\mathcal{R}_e$ of polygonal regions of $\Omega_t(\boldsymbol{x})$ defined by the breaklines in $\mathcal{D}^{\geq e}$;
- the finite subset $\hat{\mathcal{R}}_e \subseteq \mathcal{R}_e$ of *stable* regions: polygonal regions that would not be subdivided if breaklines of lower degree (from $\mathcal{D}^{e'}$ for some $e' < e$) were added.

This leads to Algorithm 1 which starts by calling Algorithm 2 to build the compact tiling domain Ω. Then, the algorithm calls Algorithm 3 that builds the set of midpoints of the line segments on $(\mathbb{A} \setminus \mathbb{B}) \cap \Omega$, denoted $\Lambda_{\mathbb{A}}$. Algorithm 1 takes as input the parameter $\delta \in \mathbb{N}$, which controls the smallest degree for the considered breaklines (details below), and as a consequence controls the number of tested regions and therefore the complexity of the method. Algorithm 4 uses this parameter to build Λ_δ, the set of centroids associated with the stable regions. Finally, Algorithm 1 evaluates the function f on every point of $\Lambda_{\mathbb{A}} \cup \Lambda_\delta$ and keeps the one achieving the minimum value.

Algorithm 1 Complex rank-one quantization algorithm to minimize (1)

Input: $\boldsymbol{x} \in \mathbb{C}^m, \boldsymbol{y} \in \mathbb{C}^n, t \geq 1, \delta \in \mathbb{N}$.
Output: $\hat{\boldsymbol{x}} \in \mathbb{CF}_t^m, \hat{\boldsymbol{y}} \in \mathbb{CF}_t^n, \hat{\lambda} \in \mathbb{C}$.

Initialize $\hat{\boldsymbol{x}} \leftarrow 0, \hat{\boldsymbol{y}} \leftarrow 0, \hat{\lambda} \leftarrow 0, \hat{\mu} \leftarrow 0$.
if $\boldsymbol{x} \neq 0$ and $\boldsymbol{y} \neq 0$ **then**
  Build $\mathbb{C}_{\boldsymbol{x}}$ from (6) and its *normalized* version $\bar{\mathbb{C}}_{\boldsymbol{x}}$ from (17).
  $\Omega \leftarrow$ BuildDomain$(\bar{\mathbb{C}}_{\boldsymbol{x}}, t)$ from Algorithm 2.
  $\Lambda_{\mathbb{A}} \leftarrow$ BuildSegmentCentroids$(\bar{\mathbb{C}}_{\boldsymbol{x}}, t, \Omega)$ from Algorithm 3.
  $\Lambda_{\delta} \leftarrow$ BuildPolygonalCentroids$(\bar{\mathbb{C}}_{\boldsymbol{x}}, t, \delta, \Omega)$ from Algorithm 4.
  for $\lambda \in \Lambda_{\mathbb{A}} \cup \Lambda_{\delta}$ **do**
    $\tilde{\boldsymbol{x}} \leftarrow \text{round}(\lambda \boldsymbol{x})$
    $\mu \leftarrow \frac{\langle \tilde{\boldsymbol{x}}, \boldsymbol{x} \rangle}{\|\tilde{\boldsymbol{x}}\|^2}$
    $\tilde{\boldsymbol{y}} \leftarrow \text{round}(\mu \boldsymbol{y})$
    if $C(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) < C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ **then**
      $\hat{\boldsymbol{x}} \leftarrow \tilde{\boldsymbol{x}}, \hat{\boldsymbol{y}} \leftarrow \tilde{\boldsymbol{y}}, \hat{\lambda} \leftarrow \lambda, \hat{\mu} \leftarrow \mu$.
    end if
  end for
end if
return $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\lambda}, \hat{\mu}$.

4.1 Choice of a tiling domain Ω_t

In the previous sections, to establish theoretical properties, we only required $\Omega_t(\boldsymbol{x})$ to be a bounded tiling domain, bounded away from zero, without other shape or scale constraints. From a computational perspective, the adequation of its shape to the "natural" polygonal pieces of f is however desirable to somehow minimize the number of explored centroids, which suggests to select a polygonal shape. We begin by noting that for every $q \in \mathbb{Z}$, $s \in \{-1, +1\}$

$$\mathbb{D}(z, \beta) = \mathbb{D}(sz2^{-q}, s\beta 2^{-q}), \quad \forall z \in \mathbb{C}_{\boldsymbol{x}}, \forall \beta \in \mathbb{M}_t^* .$$

This means that we can *normalize* the directions such that their modulus is between 1 and 2. Each normalized direction comes as a pair with opposite signs, z and $-z$, and we arbitrarily choose one of them to obtain a *normalized set of directions* denoted

$$\bar{\mathbb{C}}_{\boldsymbol{x}} := \{\tilde{z} := 2^p z : p \in \mathbb{Z} \text{ such that } |\tilde{z}| \in [1, 2) \text{ and } \arg(\tilde{z}) \in [0, \pi), z \in \mathbb{C}_{\boldsymbol{x}}\} . \tag{17}$$

We also notice that we have the choice for the offset of the breaklines that we will use to build our compact tiling domain, so, we arbitrarily define $\beta := (2^t + 1)2^{-t} \in \mathbb{M}_t^* \cap [1, 2)$. We describe here the principles of the construction corresponding to Algorithm 2.

First, we treat the case where all[6] nonzero entries $\boldsymbol{x}_j \in \mathbb{C}$ of the vector $\boldsymbol{x}$ have the same argument modulo $\pi/2$. There are then exactly two accumulation lines, one associated to $\mathbb{A}_z$ for some $z \in \bar{\mathbb{C}}_{\boldsymbol{x}}$, and the accumulation line orthogonal to it, $\mathbb{A}_{iz}$. All other breaklines are parallel to one of these two accumulation lines, as illustrated on Figure 3 (left), where we display the chosen L-shaped domain, with two sides supported by the accumulation lines $\mathbb{A}_z = \mathbb{D}(z, 0)$, $\mathbb{A}_{iz} = \mathbb{D}(iz, 0)$, two sides being supported by ordinary breaklines $\mathbb{D}(z, \beta)$, $\mathbb{D}(iz, \beta)$ with some common offset β, and the two sides of the "inside corner" similarly supported by the breaklines $\mathbb{D}(z, \beta/2)$, $\mathbb{D}(iz, \beta/2)$.

[6]When $\boldsymbol{x} = 0$ the optimal quantization is trivially zero, hence we focus on nonzero $\boldsymbol{x}$.

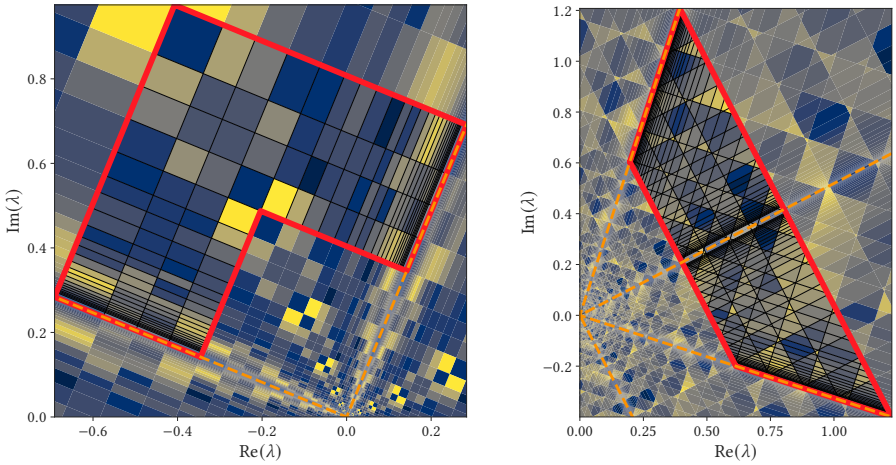

Fig. 3. Visualization of the two types of tiling domains built in Algorithm 2. *Left*: special case with two accumulation lines (line 8 in Algorithm 2). *Right*: general case with at least four distinct accumulation lines (line 5 in Algorithm 2)

We now treat the case where at least two components $\boldsymbol{x}_j$ have arguments that are not equal modulo $\pi/2$. This corresponds to the presence of at least four accumulation lines (two distinct pairs of perpendicular accumulation lines). The domain constructed in this case is illustrated on Figure 3 (right), which is a trapezoid with sides selected from the breaklines of f. To choose these breaklines, we proceed as follows: we first arbitrarily select an accumulation line $\mathbb{A}_z$ and consider the accumulation line orthogonal to it, $\mathbb{A}_{iz}$. The two non-parallel sides of the trapezoid will lie on these two accumulation lines. The two parallel sides are selected of the form $\mathbb{D}(z', \beta), \mathbb{D}(z', \beta/2)$ where $z' \in \mathbb{C}_{\boldsymbol{x}}$ is not colinear to z or to iz (such a z' exists for the considered case). The longer parallel side of the trapezoid is supported by $\mathbb{D}(z', \beta)$, while the shorter one is supported by $\mathbb{D}(z', \beta/2)$. The overall procedure is described in Algorithm 2.

Algorithm 2 Build the compact polygonal tiling domain $\Omega_t(\boldsymbol{x})$

Input: $\bar{\mathbb{C}}_{\boldsymbol{x}} \subseteq \mathbb{C}$, $t \geq 1$.
Output: compact polygonal tiling $\Omega_t(\boldsymbol{x})$.
1: Select an arbitrary $z \in \bar{\mathbb{C}}_{\boldsymbol{x}}$.
2: Describe $\mathbb{A}_z$ and $\mathbb{A}_{iz}$ from (8) with `shapely`.
3: Remove from $\bar{\mathbb{C}}_{\boldsymbol{x}}$ every z' with argument equal to the one of z modulo $\pi/2$.
4: $\beta \leftarrow (2^t + 1)2^{-t} \in \mathbb{M}_t^*$.
5: **if** $\bar{\mathbb{C}}_{\boldsymbol{x}} \neq \emptyset$ **then**
6: Select an arbitrary $z' \in \bar{\mathbb{C}}_{\boldsymbol{x}}$.
7: Describe Ω with `shapely`, based on $\mathbb{A}_z$, $\mathbb{A}_{iz}$, $\mathbb{D}(iz', \beta)$ and $\mathbb{D}(iz', \beta/2)$.
8: **else**
9: Describe Ω with `shapely`, based on $\mathbb{A}_z$, $\mathbb{A}_{iz}$, $\mathbb{D}(z, \beta)$, $\mathbb{D}(iz, \beta')$, $\mathbb{D}(z, \beta/2)$ and $\mathbb{D}(iz, \beta'/2)$.
10: **end if**

4.2 Building the set of centroids Λ_δ

The construction of Λ_δ in Algorithm 4 starts by finding the smallest exponent parameter e in the definition of $\mathbb{M}_t^*$ such that the set of stable regions is empty – we denote it $e_{\min}$. This algorithm is defined in Appendix B and a visual

Algorithm 3 Build $\Lambda_{\mathbb{A}}$ (midpoints of the line segments on $(\mathbb{A} \setminus \mathbb{B}) \cap \Omega$)

Input: $\bar{\mathbb{C}}_{\boldsymbol{x}} \subseteq \mathbb{C}$, $t \geq 1$, closed polygonal domain $\Omega \subseteq \mathbb{C}$.
Output: set $\Lambda_{\mathbb{A}}$ of midpoints of the line segments on $(\mathbb{A} \setminus \mathbb{B}) \cap \Omega$.

$\Lambda_{\mathbb{A}} \leftarrow \emptyset$.
for $z \in \bar{\mathbb{C}}_{\boldsymbol{x}}$ such that $\mathbb{A}_z \cap \Omega \neq \emptyset$ **do**
  $\mathbb{B}_z \leftarrow \emptyset$
  for $z' \in \bar{\mathbb{C}}_{\boldsymbol{x}} \setminus \{z\}$ **do**
    for $\beta \in \mathbb{M}_t^*$ such that $\mathbb{A}_z \cap \mathbb{D}(z', \beta) \cap \Omega \neq \emptyset$ **do**
      $\mathbb{B}_z \leftarrow \mathbb{B}_z \cup \mathbb{A}_z \cap \mathbb{D}(z', \beta) \cap \Omega$
    end for
  end for
  Sort $\mathbb{B}_z$ by increasing modulus.
  Add to $\Lambda_{\mathbb{A}}$ the midpoints between two consecutive points of $\mathbb{B}_z$.
end for

representation is given in Figure 4: it shows the evolution of the stable regions when decreasing e. In this example, stable regions (depicted in blue) appear for $e = 0$, therefore, we set $e_{\min} := 1$. Then, the directions $\mathbb{C}_{\boldsymbol{x}}$ and the midpoints of degree at least $e := e_{\min} - \delta$ are used to build, thanks to the python library `shapely`[7], a first list of polygonal regions $\mathcal{R}_e$. Among these, Algorithm 4 keeps only the stable ones in $\hat{\mathcal{R}}_e$. Finally, `shapely` is used to compute a centroid for each region in $\hat{\mathcal{R}}_e$. Setting $\delta = 0$ amounts to restricting the search to $\Lambda_{\mathbb{A}}$, since with the above definition we have $\Lambda_0 = \emptyset$ by definition of $e_{\min}$.

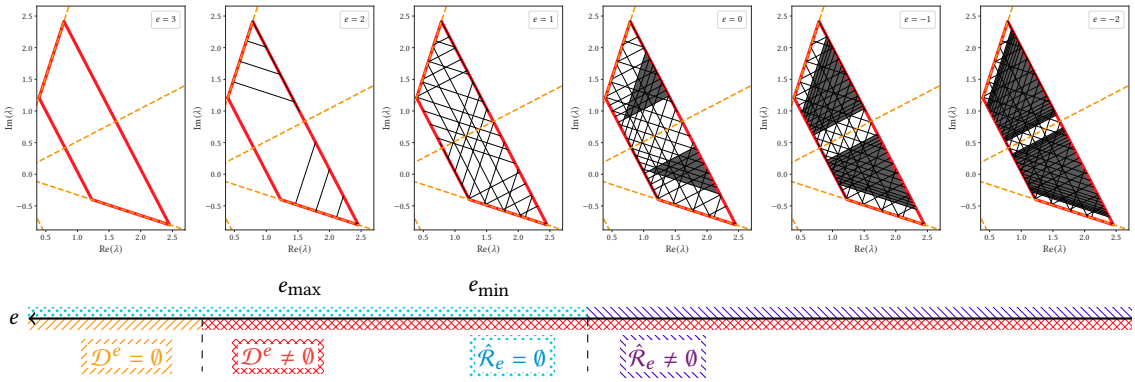

Fig. 4. Visual representation of the evolution of the breaklines (drawn in black with solid lines) intersecting $\Omega_t(\boldsymbol{x})$ (depicted in red) and the stable regions (drawn in gray) depending on e. The larger e (i.e., towards the left), the further away the breaklines and the stable regions are from the accumulation lines (drawn in orange with dashed lines). When $e > e_{\max}$ (here $e_{\max} = 2$), there is no breaklines inside $\Omega_t(\boldsymbol{x})$ anymore. When $e \geq e_{\min}$ (here $e_{\min} = 1$), there is no stable region anymore.

To build the set of stable regions, $\hat{\mathcal{R}}_e$, Algorithm 4 defines $\beta_e := (2^t + 1)2^{e-1-t}$ the smallest offset from $\mathbb{M}_t^{\geq e}$. Then, with `shapely`, Algorithm 4 removes from $\mathcal{R}_e$ all the regions that are between $\mathbb{D}(z, \beta_e)$ and $\mathbb{D}(z, -\beta_e)$ for some $z \in \mathbb{C}_{\boldsymbol{x}}$. These removed regions are indeed non-stable because, with a higher e, more breaklines would appear between $\mathbb{D}(z, \beta_e)$ and $\mathbb{D}(z, -\beta_e)$. They are also the only non-stable regions.

[7] https://shapely.readthedocs.io/en/stable/

Algorithm 4 Build Λ_δ (centroids associated to breaklines of degree at least $e_{\min} - \delta$)

Input: $\bar{\mathbb{C}}_{\boldsymbol{x}} \subseteq \mathbb{C}$, $t \geq 1$, $\delta \in \mathbb{N}$, closed polygonal domain $\Omega \subseteq \mathbb{C}$.
Output: set $\Lambda_\delta(\boldsymbol{x})$ of centroids.
$e_{\max} \leftarrow$ FindEmax$(\bar{\mathbb{C}}_{\boldsymbol{x}}, t, \Omega)$ from Algorithm 6.
$e_{\min} \leftarrow$ FindEmin$(\bar{\mathbb{C}}_{\boldsymbol{x}}, t, \Omega)$ from Algorithm 7.
$e \leftarrow e_{\min} - \delta$.
Describe with `shapely` the finite set $\mathcal{D}^{\geq e}$ (breaklines of degree at least e that intersect Ω).
Compute $\mathcal{R}_e$ from $\mathcal{D}^{\geq e}$ using `shapely`.
$\beta \leftarrow (2^t + 1)2^{e-1-t}$.
for $z \in \bar{\mathbb{C}}_{\boldsymbol{x}}$ such that $\mathbb{A}_z \cap \Omega \neq \emptyset$ **do**
    Describe with `shapely` the two breaklines $\mathbb{D}(z, \beta)$ and $\mathbb{D}(z, -\beta)$.
    With `shapely`, remove from $\mathcal{R}_e$ all regions lying between $\mathbb{D}(z, \beta)$ and $\mathbb{D}(z, -\beta)$.
end for
$\Lambda_\delta \leftarrow \emptyset$.
for $R \in \mathcal{R}_e$ **do**
    $\Lambda_\delta \leftarrow \Lambda_\delta \cup \{c\}$ with c the centroid of the region R computed with `shapely`.
end for

4.3 Complexity

The complexity of the algorithm depends on the numbers of centroids that are tested. For every $e \leq e_{\min} - 1$, $|\mathbb{M}_t^e| = 2^t$ and for every $\boldsymbol{x} \in \mathbb{C}^m$, $|\bar{\mathbb{C}}_{\boldsymbol{x}}| \leq |\mathbb{C}_{\boldsymbol{x}}| \leq 4m$, hence $|\mathcal{D}^e| \leq 4m2^t$. Therefore, for every $\delta \geq 1$, $|\mathcal{D}^{e_{\max} \geq e_{\min} - \delta}| \leq 4m\delta 2^t$. According to [11, sec. 1.2], the maximum number of regions defined by k straight lines is $\frac{k(k+1)}{2} + 1$. In our case, we can conclude that $|\Lambda_\delta| = |\hat{\mathcal{R}}_\delta| \leq |\mathcal{R}_\delta| = O(m^2\delta^2 2^{2t})$. Furthermore, $|\Lambda_{\mathbb{A}}|$ is negligible compared to Λ_δ. Indeed, if we consider an accumulation line $\mathbb{A}_z$, then by considering the breaklines parallel to $\mathbb{A}_{z'}$, only 2^t of these breaklines can intersect $\mathbb{A}_z$ inside $\Omega_t(\boldsymbol{x})$. Thus, we have $|\Lambda_{\mathbb{A}}| \leq 4m \cdot 4(m-1) \cdot 2^t = O(m^2 2^t)$. For each centroid, we need to calculate $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$, which has a cost of $O(m+n)$, by exploiting the properties of the Frobenius norm, as in the real-valued case [12, eq. 5.2]. The total cost of the algorithm is therefore $O\left((m+n)m^2\delta^2 2^{2t}\right)$.

Swapping of $\boldsymbol{x}, \boldsymbol{y}$ to reach a minimal complexity. Since $\boldsymbol{x}$ and $\boldsymbol{y}$ play symmetrical roles, we can swap their roles, which will result in a time complexity of $O\left((m+n)n^2\delta^2 2^{2t}\right)$. Thus, using the best of the two possible swaps as input of Algorithm 1 yields our main algorithm, CROQUant (Complex Rank-One Quantization algorithm) described in Algorithm 5, which has a time complexity

$$O\left(\min(m,n)^2 \cdot \max(m,n) \cdot \delta^2 \cdot 2^{2t}\right). \tag{18}$$

Algorithm 5 CROQuant (Complex Rank-One Quantization algorithm) to minimize (1) with minimal time complexity

Input: $\boldsymbol{x} \in \mathbb{C}^m$, $\boldsymbol{y} \in \mathbb{C}^n$, $t \geq 1$, $\delta \in \mathbb{N}$.
Output: $\hat{\boldsymbol{x}} \in \mathbb{CF}_t^m$, $\hat{\boldsymbol{y}} \in \mathbb{CF}_t^n$, $\lambda \in \mathbb{C}$.
if $m \leq n$ **then**
    $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \lambda, \mu \leftarrow$ ComplexRankOneQuantization$(\boldsymbol{x}, \boldsymbol{y}, t, \delta)$ from Algorithm 1.
else
    $\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}, \mu, \lambda \leftarrow$ ComplexRankOneQuantization$(\boldsymbol{y}, \boldsymbol{x}, t, \delta)$ from Algorithm 1.
end if
return $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \lambda$.

5 EXPERIMENTS

In this section we numerically compare the proposed quantization method to the classical round to nearest (RTN) strategy. To compare the two methods, we define the generic comparison metric

$$\rho_{\boldsymbol{x},\boldsymbol{y}}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}) \coloneqq \frac{\|\boldsymbol{x}\boldsymbol{y}^H - \hat{\boldsymbol{x}}\hat{\boldsymbol{y}}^H\|}{\|\boldsymbol{x}\boldsymbol{y}^H\|}\,.$$

For our algorithm we denote $\rho_\delta \coloneqq \rho_{\boldsymbol{x},\boldsymbol{y}}(\hat{\boldsymbol{x}}_\delta,\hat{\boldsymbol{y}}_\delta)$ the metric obtained with $\hat{\boldsymbol{x}}_\delta$ and $\hat{\boldsymbol{y}}_\delta$, the vectors returned by Algorithm 5 with the parameter δ. For the baseline RTN, the metric is denoted by $\rho_{\text{rtn}} \coloneqq \rho_{\boldsymbol{x},\boldsymbol{y}}(\text{round}(\boldsymbol{x}),\text{round}(\boldsymbol{y}))$.

In the following sections, we will give some insight on the role of the parameter δ in terms of time/accuracy tradeoff and on the dependence of $\rho_\delta/\rho_{\text{rtn}}$ on the vector dimensions m, n.

5.1 Experimental setting

We consider both problems of various sizes m, n, some with equal sizes ($m = n$) and other with unbalanced sizes. The case where $m = 2$ and n is larger (or vice-versa) is particularly relevant for the application to butterfly factors described in the next section. Regarding the entries entries of $\boldsymbol{x}$ and $\boldsymbol{y}$, they are draw i.i.d. according to one of the following distributions:

- *uniform*: the real and the imaginary part of $\boldsymbol{x}$ and $\boldsymbol{y}$ are drawn i.i.d. according to a uniform distribution on $[0, 1]$.
- *normal*: the real and the imaginary part of $\boldsymbol{x}$ and $\boldsymbol{y}$ are drawn i.i.d. according to the standard normal distribution.
- *finite alphabet*: the components are drawn uniformly from a finite set ("alphabet") of 32 complex-values $\{\alpha_1,\dots,\alpha_{32}\}$. Each $\alpha_i \in \mathbb{C}$ is drawn with i.i.d. real and imaginary values according to the uniform distribution on $[0, 1]$.
- *roots of unity*: the components of $\boldsymbol{x}$ and $\boldsymbol{y}$ are drawn uniformly at random among the roots of unity of order 32. This is a special case of finite alphabet that matches the entries of natural factors in butterfly factorizations of the discrete Fourier transform in certain dimensions.

As we will see, while the non-degeneracy assumption is always satisfied with uniformly or normally drawn entries, it can fail with non-negligible probability with entries from a finite alphabet, whether random or built algebraically with roots of unity. In the latter case, the failure probability further increases.

Experiments are run on an Intel(R) Xeon (R) Gold 5218 CPU @ 2.30GHz. Our implementation is available in open source [5].

5.2 On the degeneracy assumption

We begin by generating 1000 couples $(\boldsymbol{x},\boldsymbol{y}) \in \mathbb{C}^m \times \mathbb{C}^n$, where $t = 4$, to obtain the proportion of couples that do not satisfy the non-degeneracy assumption in Definition 3.1. The results are given in Table 1 and the method to obtain them is described in Appendix C. We observe that if the distribution of the real and the imaginary part is continuous according to the Lebesgue measure, then the assumption is always true, whereas when $\boldsymbol{x}$ and $\boldsymbol{y}$ come from a finite alphabet, the non-degeneracy hypothesis is not plausible anymore. This means that in the latter case there is no guarantee on the existence of an optimal solution, but we will see in the following that, despite this, our algorithm is still able to find a better rescaling parameter than the baseline $\lambda = 1$.

Table 1. Proportion of 1000 couples $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{C}^m \times \mathbb{C}^n$ that do not satisfy the non-degeneracy assumption with $t = 4$.

	$m = n = 2$	$m = 2, n = 32$	$m = n = 32$
Uniform	0.0%	0.0%	0.0%
Normal	0.0%	0.0%	0.0%
Finite alphabet	0.5%	2.1%	4.2%
Roots of unity	12.4%	23.5%	100%

5.3 Choice of the parameter δ

In this section we study the impact of the parameter δ on CROQuant, in terms of accuracy and computational time. For the next experiments, we consider only two distributions: *uniform* and *roots of unity* as representing a continuous distribution and a distribution on a finite alphabet.

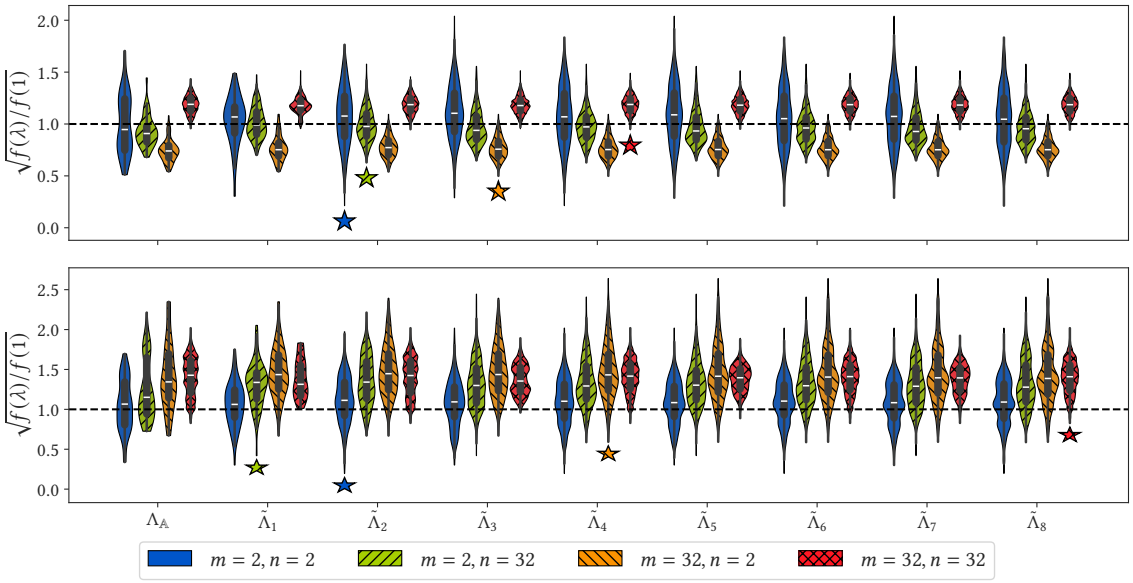

Fig. 5. Distribution of $\sqrt{f(\lambda)/f(1)}$ where $\lambda \in \Lambda_{\mathbb{A}}$ (resp. $\lambda \in \tilde{\Lambda}_\delta := \Lambda_\delta \setminus (\Lambda_{\delta-1} \cup \Lambda_{\mathbb{A}})$ for $1 \le \delta \le 8$) for a single pair $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{C}^m \times \mathbb{C}^n$ with different values of m and n ($t = 4$). *Upper part*: each component of $\boldsymbol{x}$ and $\boldsymbol{y}$ are random roots of unity of order 32. *Bottom part*: the real and the imaginary part of $\boldsymbol{x}$ and $\boldsymbol{y}$ follow a uniform distribution on $[0, 1]$.

We generate a couple $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{C}^m \times \mathbb{C}^n$ for different values of m and n. Instead of simply considering the output of Algorithm 5, we use the fact that internally it computes collections $\Lambda_{\mathbb{A}}$ and $\tilde{\Lambda}_\delta := \Lambda_\delta \setminus (\Lambda_{\delta-1} \cup \Lambda_{\mathbb{A}})$ of centroids for $1 \le \delta \le 8$, and report in Figure 5 a violin plot of $\sqrt{f(\lambda)/f(1)}$ where $\lambda \in \Lambda_{\mathbb{A}}$ (resp. $\lambda \in \tilde{\Lambda}_\delta$, $1 \le \delta \le 8$). Each color corresponds to a choice of m, n. Notice that the special case $\delta = 0$ corresponds to searching for the optimum midpoint only on the accumulation lines. Restricting the search at this initial exploration would be a cheap option, since it requires to explore just a small number of parameters.

We observe that, for each choice of m, n:

- both in $\Lambda_{\mathbb{A}}$ and in $\tilde{\Lambda}_\delta$, there are (sometimes many) choices of λ improving over naive RTN, that is to say such that $f(\lambda)/f(1) < 1$ and the minimum (marked by a star) over $\lambda \in \Lambda_{\mathbb{A}} \cap \Lambda_8$ can be much better than naive RTN;

- for the considered pair $(\boldsymbol{x}, \boldsymbol{y})$ the minimum *never* corresponds to $\lambda \in \Lambda_{\mathbb{A}}$. It thus seems worthwhile in terms of accuracy to continue the search beyond the accumulation lines, especially when m, n are both small.

Let us now focus also on the computational time. We consider 100 randomly generated pairs $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{C}^m \times \mathbb{C}^n$ with our two considered distributions.

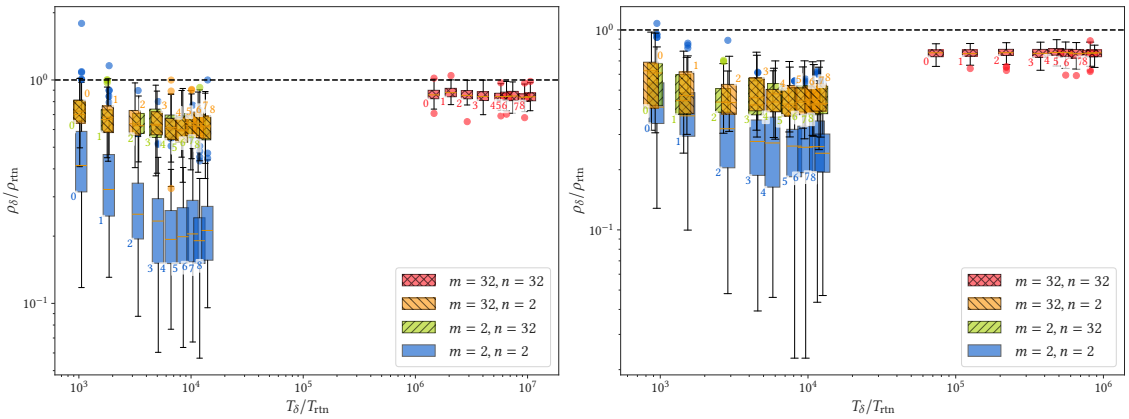

Fig. 6. Evolution of $\rho_\delta/\rho_{\text{rtn}}$ in terms of T_δ/T_{rtn} for 100 pairs $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{C}^m \times \mathbb{C}^n$ with different values of m and n ($t = 4$). The numbers next to the boxplots corresponds to the value δ used to generate the boxplot. *Left*: the real and the imaginary part of $\boldsymbol{x}$ and $\boldsymbol{y}$ follow a uniform distribution on $[0, 1]$. *Right*: each component of $\boldsymbol{x}$ and $\boldsymbol{y}$ are random roots of unity of order 32. One can observe that: a) on almost all the pairs, there is an improvement over the baseline RTN; b) as m and n increase, increasing δ is less worthwhile.

Figure 6 shows the evolution of $\rho_\delta/\rho_{\text{rtn}}$ in terms of T_δ/T_{rtn} for our two considered distributions, where T_δ and T_{rtn} are the computation time in seconds of our algorithm with the given parameter δ and of the RTN approach, respectively. First of all, we observe that these tests confirm that the result shown in Figure 5 is a generic trend, and that the proposed algorithm provides an improved accuracy compared to the RTN method when $\delta \geq 2$. They also highlight that the achieved quantization error increases with m, n.

Furthermore, as $\min(m, n)$ increases, the computation time increases quadratically, while the gain in quantization error becomes smaller and smaller. This suggests that for small values of $\min(m, n)$, it is interesting to use a high δ, while for large values of m and n, only using the centroids on the accumulation lines (i.e., $\delta = 0$) seems to be the most interesting choice.

In some rare cases, when $\delta < 2$, our algorithm achieves a higher error than the baseline RTN. This implies that the stable regions that we are considering for the algorithm do not contain $\lambda = 1$. Otherwise, our algorithm would evaluate the function f at 1, which is exactly the scaling factor of RTN, hence the quantization error of CROQuant described in Algorithm 5 would be at most equal to that of the baseline.

For the experiments of the next section with butterfly matrices, where $\min(m, n) = 2$, we will use $\delta = 2$, as we find it to be a good compromise between the gain in terms of quantization error and the computation time compared to the baseline RTN.

6 APPLICATION TO BUTTERFLY QUANTIZATION

We now exploit our algorithm to quantize butterfly factors, which appear for instance in the FFT. The FFT accelerates the discrete Fourier transform (DFT) using the Cooley-Tukey algorithm [6]. The main idea is to use the divide-and-conquer

strategy, which recursively factorizes the DFT of size $n \times n$ into $\log_2(n)$ sparse and structured matrices (for simplicity we consider here a dimension n that is a power of two, but more flexible butterfly factorizations can be defined [17]). Denoting $\boldsymbol{F} \in \mathbb{C}^{n \times n}$ the DFT matrix, its factorization associated to the Cooley-Tukey algorithm is

$$\boldsymbol{F} = \boldsymbol{B}_1 \cdots \boldsymbol{B}_L \boldsymbol{P},$$

where $L := \log_2(n)$, $\boldsymbol{P}$ is a so-called *bit-reversal permutation* matrix, and the $\boldsymbol{B}_i$ are structured sparse matrices with the so-called *Kronecker-sparse structure* [17].

To quantize the factors $\boldsymbol{B}_i$, we use the heuristic procedure proposed in [12], and replace the real-valued quantization algorithm with the proposed Algorithm 1, in order to work on a product of complex-valued butterfly factors, as it is common in real-life FFTs. For completeness of the presentation, we outline the heuristic from [12] in the following subsection.

6.1 Quantization algorithm for butterfly matrices

Let us start with the case of two factors. We want to quantize the product of *two* (possibly sparse) matrices $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{C}^{n \times r}$. The quantization problem then reads

$$\boldsymbol{X}^*, \boldsymbol{Y}^* \in \arg \min_{\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}} \in C\mathcal{F}_t} \|\boldsymbol{X}\boldsymbol{Y}^H - \hat{\boldsymbol{X}}\hat{\boldsymbol{Y}}^H\|^2, \tag{19}$$

where $C\mathcal{F}_t$ is the set of matrices with coefficient in $\mathbb{CF}_t$ and the same support[8] as $\boldsymbol{X}, \boldsymbol{Y}$.

To the best of our knowledge, there is no method for finding the optimal quantization for this problem in the most general setting. However, if the rank-one matrices, $\boldsymbol{x}_j \boldsymbol{y}_j^H$, where $\boldsymbol{x}_j$ and $\boldsymbol{y}_j$ are the corresponding columns of $\boldsymbol{X}$ and $\boldsymbol{Y}$, $1 \leq j \leq r$, have disjoint supports, then the problem (19) can be decomposed into r independent complex rank-one matrix quantization problems:

$$\|\boldsymbol{X}\boldsymbol{Y}^H - \hat{\boldsymbol{X}}\hat{\boldsymbol{Y}}^H\|^2 = \sum_{j=1}^{r} \|\boldsymbol{x}_j \boldsymbol{y}_j^H - \hat{\boldsymbol{x}}_j \hat{\boldsymbol{y}}_j^H\|^2,$$

and each sub-problem can be addressed using CROQuant defined in Algorithm 5.

To address the generic problem with L factors, we can leverage the two-factor setting. When considering certain so-called *chainable* Kronecker-sparse factors [17], for any subset of consecutive factors, the product between $\boldsymbol{X} = \boldsymbol{B}_{l_0} \cdots \boldsymbol{B}_{l_1} \in \mathbb{C}^{n \times n}$ and $\boldsymbol{Y}^H = \boldsymbol{B}_{l_1+1} \cdots \boldsymbol{B}_{l_2} \in \mathbb{C}^{n \times n}$, with $1 \leq l_0 \leq l_1 \leq l_2 \leq L$, can be written as a sum of n rank-one matrices with disjoint support [16]. The factors $\boldsymbol{B}_i$ arising in the FFT are a particular case of such Kronecker-sparse factors. Thus, this fact can be used to heuristically decompose the product of L butterfly matrices into several products of two matrices and apply the optimal quantization of the problem (19). We consider the Left-to-Right (LTR) heuristic of [12]: writing $\boldsymbol{B}_1 (\boldsymbol{B}_2(\cdots(\boldsymbol{B}_{L-1}\boldsymbol{B}_L)))$, quantization is iteratively performed from left to right (details in [12]). For comparison, we also consider stochastic rounding [8] strategy in our tests, which works like RTN, i.e., quantization is applied element-wise without any rescaling, however, the quantized output is randomly chosen between the two nearest neighbors.

6.2 Evaluation of our method

We conclude this section with some numerical experiments on random butterfly matrices and on butterfly factors arising in the discrete Fourier transform. We compare the LTR heuristic method and the two baseline approaches (RTN

[8]The support of a matrix $\boldsymbol{A}$ is the set of coordinates (i, j) where $\boldsymbol{A}_{i,j} \neq 0$.

and stochastic rounding). Experiments are run on an Intel(R) Xeon (R) Gold 5218 CPU @ 2.30GHz. Our open source implementation is available in [5].

6.2.1 Evaluation on random butterfly matrices. To compare these methods, we generate 10 random collections of Kronecker-sparse factors $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_L$ with square dyadic sparsity pattern [17] and quantize them with the methods described before to obtain $\hat{\boldsymbol{B}}_1 \cdots \hat{\boldsymbol{B}}_L$. Figure 7 shows the evolution of $\rho := \frac{\|\boldsymbol{B}_1 \cdots \boldsymbol{B}_L - \hat{\boldsymbol{B}}_1 \cdots \hat{\boldsymbol{B}}_L\|}{\|\boldsymbol{B}_1 \cdots \boldsymbol{B}_L\|}$, the relative quantization error, as a function of t (left) and as a function of n (right) with $\delta = 2$. Despite the matrix product parenthesis heuristic, LTR is more accurate than RTN and stochastic rounding. Indeed, we see that LTR needs about $t = 3$ bits of mantissa (which corresponds to the quarter precision `float8 E4M3` format) to achieve an error of approximately $3 \cdot 10^{-2}$, while RTN and stochastic rounding require more than 5 bits of mantissa to reach a comparable error level. More generally, Table 2 gives qualitative correspondences between the number of mantissa bits for LTR and RTN to achieve comparable quantization errors. Finally, the exponential fit (shown in the legend of Figure 7) shows that, in general, LTR needs $1 - \frac{1}{1.4} \approx 30\%$ fewer bits than RTN and stochastic rounding to achieve the same quantization error, as it is the case also in the real-valued case, cf. [12].

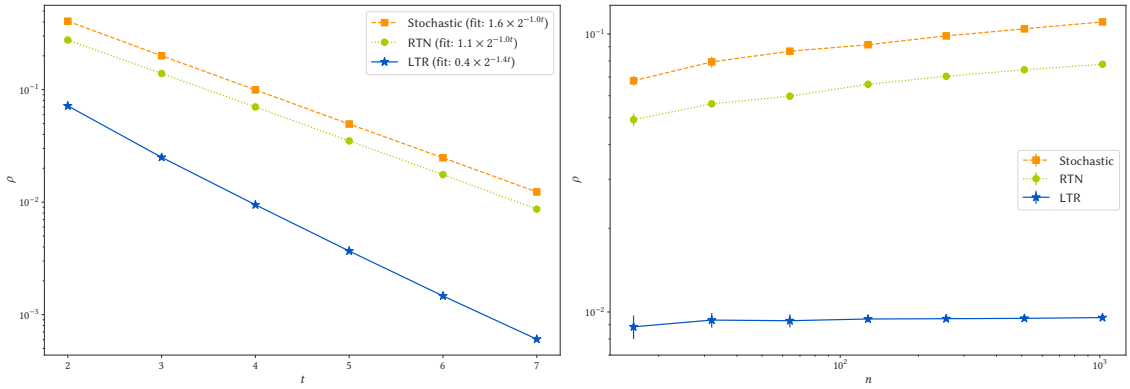

Fig. 7. Average of the quantization error on the butterfly decomposition of 10 random Kronecker-sparse matrices with the square-dyadic sparsity pattern, as a function of t (*left*) with $n = 256$ and as a function of n (*right*) with $t = 4$ ($\delta = 2$). The standard deviation (over 10 draws of random Kronecker-sparse matrices) is too small ($\approx 10^{-4}$) to be visible on the left.

Table 2. Number of mantissa bits required by LTR and RTN to achieve comparable quantization errors, ρ_{target}, for random Kronecker-sparse matrices and FFT. The real error of the two methods is written right next to it in parentheses.

Setting	ρ_{target}	t_{RTN}	(ρ_{RTN})	t_{LTR}	(ρ_{LTR})	Δt
Random butterfly	$7 \cdot 10^{-2}$	4	$(7.02 \cdot 10^{-2})$	2	$(7.18 \cdot 10^{-2})$	-2 bits
	$3 \cdot 10^{-2}$	5	$(3.50 \cdot 10^{-2})$	3	$(2.50 \cdot 10^{-2})$	-2 bits
	$9 \cdot 10^{-3}$	7	$(8.67 \cdot 10^{-3})$	4	$(9.47 \cdot 10^{-3})$	-3 bits
FFT	$2 \cdot 10^{-2}$	5	$(2.39 \cdot 10^{-2})$	3	$(1.54 \cdot 10^{-2})$	-2 bits
	$5 \cdot 10^{-3}$	7	$(4.77 \cdot 10^{-3})$	4	$(5.67 \cdot 10^{-3})$	-3 bits

6.2.2 Evaluation on the FFT. We also apply the three quantization algorithms to the (exact) Cooley-Tukey factorization $\boldsymbol{F} = \boldsymbol{B}_1 \cdots \boldsymbol{B}_L \boldsymbol{P} \in \mathbb{C}^{n \times n}$ where $\boldsymbol{F}$ is the DFT. In this case we consider not only the quantization error ρ on the factors as in the previous case, but also the error on the action of the operator, i.e., what happens when we apply the quantized

version of $\boldsymbol{F}$ to a vector $\boldsymbol{x}$ instead of the original high-resolution DFT. More precisely, let $\boldsymbol{x} \in \mathbb{R}^n$ be a signal and $\boldsymbol{y} := \boldsymbol{F}\boldsymbol{x} \in \mathbb{C}^n$ its Fourier transform. We define $\hat{\boldsymbol{y}} := \hat{\boldsymbol{F}}\boldsymbol{x} = \hat{\boldsymbol{B}}_1 \cdots \hat{\boldsymbol{B}}_L \boldsymbol{P}\boldsymbol{x}$, the result of the application of the quantized Fourier transform and consider the relative error $\rho_{\text{fft}} := \frac{\|\boldsymbol{y}-\hat{\boldsymbol{y}}\|}{\|\boldsymbol{y}\|}$. The average of ρ_{fft} for 10 standard Gaussian signals, with the four considered algorithms is displayed in Figure 8 for varying t (left) and n (right) with $\delta = 2$. Like for the butterfly quantization error, the LTR method is more accurate than RTN and the stochastic quantization. Table 2 provides qualitative correspondences between the number of mantissa bits for LTR and RTN to achieve comparable errors. For example, LTR needs $t = 2$ bits (which corresponds to the quarter precision `float8 E5M2` format), while RTN needs 4 bits to reach a comparable error level. Finally, LTR needs, in general, $1 - \frac{1}{1.5} \approx 33\%$ fewer bits than RTN and stochastic rounding to achieve the same quantization error.

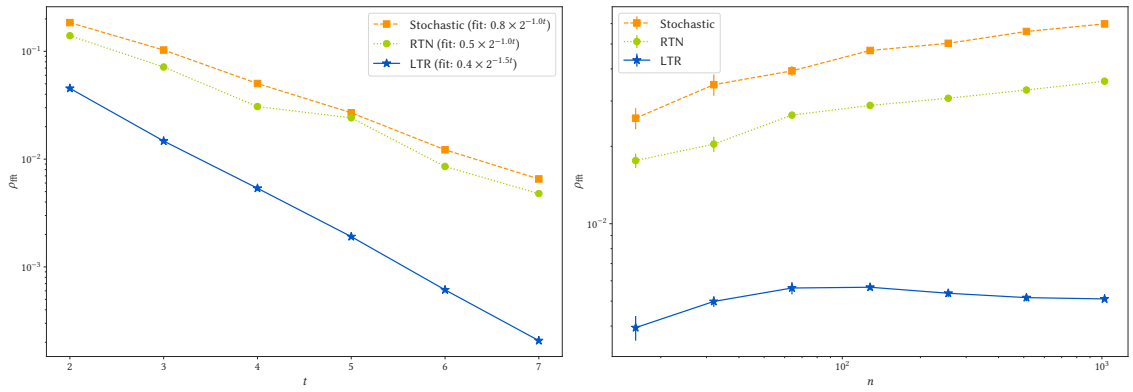

Fig. 8. Average of ρ_{fft} for 10 standard Gaussian signals in terms of t (*left*) with $n = 256$ and in terms of n (*right*) with $t = 4$ for different quantization strategies ($\delta = 2$). The standard deviation (over 10 draws of standard Gaussian signals) is too small ($\approx 10^{-5}$) to be visible on the left.

7 CONCLUSION

In this paper, we studied the problem of quantizing a complex-valued rank-one matrix. We showed, like in the real-valued case, that the problem can be characterized as a scalar function minimization problem, see Lemma 2.1. We proved that this minimization problem, despite exhibiting infinitely many piecewise constant regions due to an accumulation phenomenon absent in the real-valued case, admits a minimizer under mild assumptions. Then, we provided CROQuant (Complex Rank-One Quantization algorithm) described in Algorithm 5 that finds an approximated solution to the quantization problem. We showed empirically that this approach enables more accurate quantization than a naive element-wise rounding while guaranteeing a time complexity in $O\left(\min(m,n)^2 \max(m,n)\delta^2 2^{2t}\right)$ which is both tractable and better than naive rounding for low target precision t and reasonable parameters δ. Applying this algorithm to butterfly matrices, which play a key role in many fast transforms such as the Fast Fourier Transform, we demonstrated the possibility to reduce quantization error for a given number of mantissa bits, or alternatively to reduce by 30% the required number of mantissa bits for a given precision.

Thanks to the quantization algorithm that we have proposed in this paper we demonstrated that, as in the real-valued case, quantization gains can be achieved compared to naive round to nearest in this setting. Yet, the proposed algorithm achieves tradeoffs between accuracy and computational cost (controlled by a parameter δ) that may remain somewhat suboptimal. A natural objective for future work is thus to achieve quantization accuracies closer to the optimal one

with reduced computation time. For this, one can notably envision to leverage quantitative versions of the "continuity results" from Section 3 in order to prune out some regions from the current brute force search, in the spirit of branch and bound approaches. Another avenue woulds be to define stochastic sampling strategies on the complex plane to avoid brute force search. This would essentially replace the dependency on the parameter δ by a dependency on the number of stochastic draws.

Besides improvements to the algorithm, since our complex quantization framework enables to move from quantizing real-valued rank-one matrices to complex-valued rank-one matrices, one can also envision to leverage it to mimic a number of potential applications of the real-valued framework to the complex setting. For example, natural heuristic extensions from rank-one to rank-r matrices via greedy approaches, which were only accessible for the real-valued case, are now directly applicable to the complex-valued case. They also raise challenging questions regarding the feasibility of tractable optimal rank-r quantization.

Rescaling-invariance is the key property exploited both in the real-valued and complex-valued rank-one quantization problems. The shift from the real-valued to the complex-valued setting has shown that this property remains an opportunity, as it leads to useful degrees of freedom, but also revealed novel geometric features of the resulting optimization problem on $f(\lambda)$. There are many other problems which display variants of this property, notably when considering rank-one tensors, or ReLU neurons for which rescaling heuristics have already be exploited [20]. It is thus particularly appealing to investigate hte possibility of near optimal quantization beyond such heuristics.

REFERENCES

[1] Patrick Amestoy, Alfredo Buttari, Nicholas J. Higham, Jean-Yves L'Excellent, Theo Mary, and Bastien Vieublé. 2023. Combining Sparse Approximate Factorizations with Mixed-precision Iterative Refinement. *ACM Trans. Math. Softw.* 49, 1, Article 4 (March 2023), 29 pages. https://doi.org/10.1145/3582493

[2] Nicolas Brisebarre, Mioara Joldeş, Jean-Michel Muller, Ana-Maria Naneş, and Joris Picot. 2020. Error Analysis of Some Operations Involved in the Cooley-Tukey Fast Fourier Transform. *ACM Trans. Math. Softw.* 46, 2, Article 11 (May 2020), 27 pages. https://doi.org/10.1145/3368619

[3] Charles G. Broyden. 1970. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *Ima Journal of Applied Mathematics* 6 (1970), 76–90. https://api.semanticscholar.org/CorpusID:53868113

[4] Maël Chaumette, Rémi Gribonval, and Elisa Riccietti. 2025. CROQuant: Complex Rank-One Quantization Algorithm. In *GRETSI 2025 – XXXème Colloque Francophone de Traitement du Signal et des Images*. Strasbourg, France, 4. https://hal.science/hal-05140371

[5] Maël Chaumette, Rémi Gribonval, and Elisa Riccietti. 2026. *Code for Reproducible research - CROQuant: Complex Rank-One Quantization Algorithm, with Application to Butterfly Factorizations.* https://hal.science/view/index/docid/5505640

[6] James W. Cooley and John W. Tukey. 1965. An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. Comp.* 19, 90 (1965), 297–301. http://www.jstor.org/stable/2003354

[7] Tri Dao, Albert Gu, Matthew Eichhorn, Atri Rudra, and Christopher Ré. 2019. Learning fast algorithms for linear transforms using butterfly factorizations. In *International conference on machine learning*. PMLR, San Diego, CA, 1517–1527. https://proceedings.mlr.press/v97/dao19a/dao19a.pdf

[8] El-Mehdi El Arar, Devan Sohier, Pablo de Oliveira Castro, and Eric Petit. 2023. Stochastic Rounding Variance and Probabilistic Bounds: A New Approach. *SIAM Journal on Scientific Computing* 45, 5 (2023), C255–C275. https://doi.org/10.1137/22M1510819

[9] Weiguo Gao, Yuxin Ma, and Meiyue Shao. 2025. A Mixed Precision Jacobi SVD Algorithm. *ACM Trans. Math. Softw.* 51, 1, Article 5 (April 2025), 33 pages. https://doi.org/10.1145/3721124

[10] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. A Survey of Quantization Methods for Efficient Neural Network Inference. arXiv:2103.13630 [cs.CV]

[11] Ronald L. Graham. 1994. *Concrete mathematics: a foundation for computer science.* Addison-Wesley, Boston, MA.

[12] Rémi Gribonval, Théo Mary, and Elisa Riccietti. 2023. Optimal quantization of rank-one matrices in floating-point arithmetic—with applications to butterfly factorizations. https://inria.hal.science/hal-04125381 working paper or preprint on HAL.

[13] Nicholas J. Higham. 2002. *Accuracy and Stability of Numerical Algorithms* (second ed.). Society for Industrial and Applied Mathematics, Philadelphia, PA. https://doi.org/10.1137/1.9780898718027

[14] Nicholas J. Higham and Théo Mary. 2022. Mixed precision algorithms in numerical linear algebra. *Acta Numerica* 31 (2022), 347–414. https://doi.org/10.1017/S0962492922000022

[15] Quoc-Tung Le, Elisa Riccietti, and Remi Gribonval. 2023. Spurious Valleys, NP-Hardness, and Tractability of Sparse Matrix Factorization with Fixed Support. *SIAM J. Matrix Anal. Appl.* 44, 2 (2023), 503–529. https://doi.org/10.1137/22M1496657

[16] Quoc-Tung Le, Léon Zheng, Elisa Riccietti, and Rémi Gribonval. 2022. Fast Learning of Fast Transforms, with Guarantees. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Singapore, 3348–3352. https://doi.org/10.1109/ICASSP43922.2022.9747791

[17] Quoc-Tung Le, Léon Zheng, Elisa Riccietti, and Rémi Gribonval. 2025. Butterfly Factorization with Error Guarantees. *SIAM J. Matrix Anal. Appl.* 46, 4 (Oct. 2025), 2253–2309. https://doi.org/10.1137/24M1708796

[18] Luc Le Magoarou and Rémi Gribonval. 2016. Flexible Multilayer Sparse Approximations of Matrices and Applications. *IEEE Journal of Selected Topics in Signal Processing* 10, 4 (2016), 688–700. https://doi.org/10.1109/JSTSP.2016.2543461

[19] Neil Lindquist, Piotr Luszczek, and Jack Dongarra. 2024. Generalizing Random Butterfly Transforms to Arbitrary Matrix Sizes. *ACM Trans. Math. Softw.* 50, 4, Article 26 (Dec. 2024), 23 pages. https://doi.org/10.1145/3699714

[20] Markus Nagel, Mart Van Baalen, Tijmen Blankevoort, and Max Welling. 2019. Data-Free Quantization Through Weight Equalization and Bias Correction. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Seoul, Korea (South), 1325–1334. https://doi.org/10.1109/ICCV.2019.00141

[21] Douglass S. Parker. 1995. *Random Butterfly Transformations with Applications in Computational Linear Algebra.* UCLA Computer Science Department, Los Angeles, CA. https://books.google.fr/books?id=HnghHAAACAAJ

[22] Emmanuel Quemener and Marianne Corvellec. 2013. SIDUS—the solution for extreme deduplication of an operating system. *Linux Journal* 2013, 235 (2013), 3.

[23] Jennifer Scott and Miroslav Tůma. 2024. Avoiding Breakdown in Incomplete Factorizations in Low Precision Arithmetic. *ACM Trans. Math. Softw.* 50, 2, Article 9 (June 2024), 25 pages. https://doi.org/10.1145/3651155

[24] Yanshu Wang, Tong Yang, Xiyan Liang, Guoan Wang, Hanning Lu, Xu Zhe, Yaoming Li, and Li Weitao. 2024. Art and Science of Quantizing Large-Scale Models: A Comprehensive Overview. arXiv:2409.11650 [cs.LG]

A PROOFS OF SECTION 2

All the proofs in this section follow the same pattern as in [12, sec. 4]. We start by providing the following preliminary result that shows that, for every $\hat{\boldsymbol{y}} \in \mathbb{CF}_t^n$, the cost function $C(\cdot, \hat{\boldsymbol{y}})$ decreases for a particular choice of $\hat{\boldsymbol{x}} \in \mathbb{CF}_t^m$, which can be identified simply from an optimal scaling parameter λ.

Lemma A.1. *Let $\boldsymbol{x} \in \mathbb{C}^m$, $\boldsymbol{y} \in \mathbb{C}^n$ and $t \geq 1$. For all $\hat{\boldsymbol{x}} \in \mathbb{CF}_t^m$, $\hat{\boldsymbol{y}} \in \mathbb{CF}_t^n$ it holds*

$$C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \geq C(\hat{\boldsymbol{x}}', \hat{\boldsymbol{y}}) \quad \forall \hat{\boldsymbol{x}}' \in \operatorname{round}(\lambda \boldsymbol{x}),$$

where

$$\lambda = \lambda_{\boldsymbol{y}}(\hat{\boldsymbol{y}}) := \begin{cases} \frac{\langle \boldsymbol{y}, \hat{\boldsymbol{y}} \rangle}{\|\hat{\boldsymbol{y}}\|^2} & \text{if } \hat{\boldsymbol{y}} \neq 0 \\ 0 & \text{otherwise}. \end{cases} \tag{20}$$

Proof of Lemma 2.1. First, we prove (2), (3) and (4). Consider $\hat{\boldsymbol{x}} \in \mathbb{CF}_t^m$, $\hat{\boldsymbol{y}} \in \mathbb{CF}_t^n$ and $\lambda := \lambda_{\boldsymbol{y}}(\hat{\boldsymbol{y}})$ defined in (20). Let $\hat{\boldsymbol{x}}' \in \operatorname{round}(\lambda \boldsymbol{x})$. By Lemma A.1, we have

$$C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \geq C(\hat{\boldsymbol{x}}', \hat{\boldsymbol{y}}). \tag{21}$$

Since $\hat{\boldsymbol{x}}' \in \mathbb{CF}_t^m$, we can use an analog of Lemma A.1 where the role of rows/columns is exchanged, to state that

$$C(\hat{\boldsymbol{x}}', \hat{\boldsymbol{y}}) \geq C(\hat{\boldsymbol{x}}', \hat{\boldsymbol{y}}') \quad \forall \hat{\boldsymbol{y}}' \in \operatorname{round}(\mu(\hat{\boldsymbol{x}}')\boldsymbol{y}'), \tag{22}$$

with $\mu(\hat{\boldsymbol{x}}') := \overline{\lambda_{\boldsymbol{x}}(\hat{\boldsymbol{x}}')} = \frac{\overline{\langle \boldsymbol{x}, \hat{\boldsymbol{x}}' \rangle}}{\|\hat{\boldsymbol{x}}'\|^2}$.

With (21) and (22), we get $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \geq C(\hat{\boldsymbol{x}}', \hat{\boldsymbol{y}}')$ for every $\hat{\boldsymbol{x}}' \in \operatorname{round}(\lambda \boldsymbol{x})$ and $\hat{\boldsymbol{y}}' \in \operatorname{round}(\mu(\hat{\boldsymbol{x}})\boldsymbol{y})$. Hence, $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \geq f(\lambda)$. This holds for each $\hat{\boldsymbol{x}} \in \mathbb{CF}_t^m$ and $\hat{\boldsymbol{y}} \in \mathbb{CF}_t^n$, then we get

$$\inf_{\hat{\boldsymbol{x}} \in \mathbb{CF}_t^m, \hat{\boldsymbol{y}} \in \mathbb{CF}_t^n} C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \geq \inf_{\lambda \in \mathbb{C}} f(\lambda). \tag{23}$$

For any $\lambda \in \mathbb{C}$, since $\operatorname{round}(\lambda \boldsymbol{x}) \subset \mathbb{CF}_t^m$ and $\operatorname{round}(\mu(\hat{\boldsymbol{x}})\boldsymbol{y}) \subset \mathbb{CF}_t^n$, $f(\lambda)$ is lower-bounded by the left-hand side of (23). Therefore, the inequality is an equality.

We now prove that λ^* is a minimizer of f, iff, $\hat{\boldsymbol{x}}^* \in \operatorname{round}(\lambda^* \boldsymbol{x})$ and $\hat{\boldsymbol{y}}^* \in \operatorname{round}(\mu(\hat{\boldsymbol{x}}^*)\boldsymbol{y})$ are minimizers of C. For this, consider λ^* a minimizer (assuming it exists) of f. According to (2), we have

$$\min_{\hat{\boldsymbol{x}} \in \mathbb{CF}_t^m, \hat{\boldsymbol{y}} \in \mathbb{CF}_t^n} C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = \min_{\lambda \in \mathbb{C}} f(\lambda) = f(\lambda^*) = \max_{\hat{\boldsymbol{x}} \in \operatorname{round}(\lambda^* \boldsymbol{x})} C(\hat{\boldsymbol{x}}, \operatorname{round}(\mu(\hat{\boldsymbol{x}})\boldsymbol{y})).$$

We define $\hat{\boldsymbol{x}}^* \in \operatorname{round}(\lambda^* \boldsymbol{x})$ the quantized vector that achieves the maximum in the previous equation. By considering $\hat{\boldsymbol{y}}^* \in \operatorname{round}(\mu(\hat{\boldsymbol{x}}^*)\boldsymbol{y})$, we obtain

$$\min_{\hat{\boldsymbol{x}} \in \mathbb{CF}_t^m, \hat{\boldsymbol{y}} \in \mathbb{CF}_t^n} C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = C(\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{y}}^*).$$

Reciprocally, we consider $\hat{\boldsymbol{x}}^* \in \operatorname{round}(\lambda^* \boldsymbol{x})$ and $\hat{\boldsymbol{y}}^* \in \operatorname{round}(\mu(\hat{\boldsymbol{x}}^*)\boldsymbol{y})$ the minimizers of C. By following the same steps as before, we obtain

$$C(\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{y}}^*) \geq f(\lambda^*) \geq \min_{\lambda \in \mathbb{C}} f(\lambda) \overset{(2)}{=} \min_{\hat{\boldsymbol{x}} \in \mathbb{CF}_t^m, \hat{\boldsymbol{y}} \in \mathbb{CF}_t^n} C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = C(\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{y}}^*).$$

This means that $f(\lambda^*) = \min_{\lambda \in \mathbb{C}} f$. □

Remark A.2. Actually, Lemma A.1 and Lemma 2.1 remain true if we replace the set $\mathbb{CF}_t$ by any closed countable set $\mathbb{S} \subseteq \mathbb{C}$ such that the function $\operatorname{round}_{\mathbb{S}}(u) := \arg\min_{z \in \mathbb{S}} |z - u|$ is non-empty for every $u \in \mathbb{C}$.

Proof of Lemma A.1. For $\boldsymbol{x} \in \mathbb{C}^m$ and $\boldsymbol{y} \in \mathbb{C}^n$, it holds

$$\|\boldsymbol{x}\boldsymbol{y}^H - \hat{\boldsymbol{x}}\hat{\boldsymbol{y}}^H\|^2 = \sum_{j=1}^{m} \|\boldsymbol{x}_j\boldsymbol{y} - \hat{\boldsymbol{x}}_j\hat{\boldsymbol{y}}\|^2 .$$

Let us consider each term separately. First of all notice that

$$\begin{aligned}
\|\boldsymbol{x}_j\boldsymbol{y} - \hat{\boldsymbol{x}}_j\hat{\boldsymbol{y}}\|^2 &\geq \inf_{w \in \mathbb{CF}_t} \|\boldsymbol{x}_j\boldsymbol{y} - w\hat{\boldsymbol{y}}\|^2 \\
&= \inf_{w \in \mathbb{CF}_t} \left\{ \|\boldsymbol{x}_j\boldsymbol{y}\|^2 + \|w\hat{\boldsymbol{y}}\|^2 - 2\,\mathrm{Re}\langle \boldsymbol{x}_j\boldsymbol{y}, w\hat{\boldsymbol{y}}\rangle \right\} \\
&= \|\boldsymbol{x}_j\boldsymbol{y}\|^2 + \|\hat{\boldsymbol{y}}\|^2 \inf_{w \in \mathbb{CF}_t} \left\{ |w|^2 - 2\,\mathrm{Re}(\lambda \boldsymbol{x}_j \bar{w}) \right\} ,
\end{aligned}$$

where we define $\lambda := \frac{\langle \boldsymbol{y}, \hat{\boldsymbol{y}}\rangle}{\|\hat{\boldsymbol{y}}\|^2}$ so that $\langle \boldsymbol{x}_j\boldsymbol{y}, w\hat{\boldsymbol{y}}\rangle = \boldsymbol{x}_j\bar{w}\langle \boldsymbol{y}, \hat{\boldsymbol{y}}\rangle = \lambda \boldsymbol{x}_j\bar{w}\|\hat{\boldsymbol{y}}\|^2$. Now observe that

$$\inf_{w \in \mathbb{CF}_t} |\lambda \boldsymbol{x}_j - w|^2 = |\lambda \boldsymbol{x}_j|^2 + \inf_{w \in \mathbb{CF}_t} \left\{ |w|^2 - 2\,\mathrm{Re}(\lambda \boldsymbol{x}_j \bar{w}) \right\} ,$$

which allows us to write

$$\|\boldsymbol{x}_j\boldsymbol{y} - \hat{\boldsymbol{x}}_j\hat{\boldsymbol{y}}\|^2 \geq \|\boldsymbol{x}_j\boldsymbol{y}\|^2 - |\lambda \boldsymbol{x}_j|^2\|\hat{\boldsymbol{y}}\|^2 + \|\hat{\boldsymbol{y}}\|^2 \inf_{w \in \mathbb{CF}_t} |\lambda \boldsymbol{x}_j - w|^2 .$$

The infimum is the distance between $\lambda \boldsymbol{x}_j$ and the set $\mathbb{CF}_t$. Since $\mathbb{CF}_t$ is closed, the infimum is attained for all $w \in \mathrm{round}(\lambda \boldsymbol{x}_j)$, so for $\hat{\boldsymbol{x}}' \in \mathrm{round}(\lambda \boldsymbol{x}) \subset \mathbb{CF}_t^m$, we have

$$\|\boldsymbol{x}_j\boldsymbol{y} - \hat{\boldsymbol{x}}_j\hat{\boldsymbol{y}}\|^2 \geq \|\boldsymbol{x}_j\boldsymbol{y} - \hat{\boldsymbol{x}}'_j\hat{\boldsymbol{y}}\|^2 .$$

Summing over all $j = 1, \ldots, m$, we obtain

$$\|\boldsymbol{x}\boldsymbol{y}^H - \hat{\boldsymbol{x}}\hat{\boldsymbol{y}}^H\|^2 = \sum_{j=1}^{m} \|\boldsymbol{x}_j\boldsymbol{y} - \hat{\boldsymbol{x}}_j\hat{\boldsymbol{y}}\|^2 \geq \sum_{j=1}^{m} \|\boldsymbol{x}_i\boldsymbol{y} - \hat{\boldsymbol{x}}'_j\hat{\boldsymbol{y}}\|^2 = \|\boldsymbol{x}\boldsymbol{y}^H - \hat{\boldsymbol{x}}'\hat{\boldsymbol{y}}^H\|^2 .$$

□

Proof of Lemma 2.2. For every $\boldsymbol{w} \in \mathbb{CF}_t^m$, we have

$$\mu(2\boldsymbol{w}) = \frac{\langle 2\boldsymbol{w}, \boldsymbol{x}\rangle}{\|2\boldsymbol{w}\|^2} = \frac{1}{2}\frac{\langle \boldsymbol{w}, \boldsymbol{x}\rangle}{\|\boldsymbol{w}\|^2} = \frac{1}{2}\mu(\boldsymbol{w}) \text{ and } \mu(i\boldsymbol{w}) = \frac{\langle i\boldsymbol{w}, \boldsymbol{x}\rangle}{\|i\boldsymbol{w}\|^2} = i\frac{\langle \boldsymbol{w}, \boldsymbol{x}\rangle}{\|\boldsymbol{w}\|^2} = i\mu(\boldsymbol{w}) .$$

We also have $\mathrm{round}(2\boldsymbol{w}) = 2\,\mathrm{round}(\boldsymbol{w})$ and $\mathrm{round}(i\boldsymbol{w}) = i\,\mathrm{round}(\boldsymbol{w})$. We define

$$g(\boldsymbol{w}) = C(\boldsymbol{w}, \mathrm{round}(\mu(\boldsymbol{w})\boldsymbol{y})) .$$

Then, we have

$$g(2\boldsymbol{w}) = \|\boldsymbol{x}\boldsymbol{y}^H - 2\boldsymbol{w}\,\mathrm{round}(\mu(2\boldsymbol{w})\boldsymbol{y})^H\|^2 = \|\boldsymbol{x}\boldsymbol{y}^H - \boldsymbol{w}\,\mathrm{round}(2\frac{1}{2}\mu(\boldsymbol{w})\boldsymbol{y})^H\|^2 = g(w) ,$$

and

$$\begin{aligned}
g(i\boldsymbol{w}) &= \|\boldsymbol{x}\boldsymbol{y}^H - i\boldsymbol{w}\,\mathrm{round}(\mu(i\boldsymbol{w})\boldsymbol{y})^H\|^2 = \|\boldsymbol{x}\boldsymbol{y}^H - i\boldsymbol{w}\,\mathrm{round}(i\mu(\boldsymbol{w})\boldsymbol{y})^H\|^2 \\
&= \|\boldsymbol{x}\boldsymbol{y}^H - i\bar{i}\boldsymbol{w}\,\mathrm{round}(\mu(\boldsymbol{w})\boldsymbol{y})^H\|^2 = g(w) .
\end{aligned}$$

Therefore

$$f(2\lambda) = \max_{\hat{\boldsymbol{x}} \in \mathrm{round}(2\lambda \boldsymbol{x})} g(\hat{\boldsymbol{x}}) = \max_{\hat{\boldsymbol{x}}' \in \mathrm{round}(\lambda \boldsymbol{x})} g(2\hat{\boldsymbol{x}}') = \max_{\hat{\boldsymbol{x}}' \in \mathrm{round}(\lambda \boldsymbol{x})} g(\hat{\boldsymbol{x}}') = f(\lambda) ,$$

and

$$f(i\lambda) = \max_{\hat{\boldsymbol{x}} \in \text{round}(i\lambda \boldsymbol{x})} g(\hat{\boldsymbol{x}}) = \max_{\hat{\boldsymbol{x}}' \in \text{round}(\lambda \boldsymbol{x})} g(i\hat{\boldsymbol{x}}') = \max_{\hat{\boldsymbol{x}}' \in \text{round}(\lambda \boldsymbol{x})} g(\hat{\boldsymbol{x}}') = f(\lambda)\,.$$

□

B ALGORITHMS SEARCHING FOR $e_{\max}$ AND $e_{\min}$

Algorithm 6 provides the whole procedure to find the largest integer such that the set of breaklines intersecting $\Omega_t(\boldsymbol{x})$ is non-empty. The procedure consists in increasing or decreasing an integer $e_{\max}$ until $\mathcal{D}^{e_{\max}}$ becomes empty (if we increase $e_{\max}$) or becomes non-empty (if we decrease $e_{\max}$). The variable "direction" describes if, with our initial choice of $e_{\max} = 0$, we have to increase or decrease $e_{\max}$: according to Figure 4, if $\mathcal{D}^{e_{\max}}$ is empty, then we have to increase $e_{\max}$ so that $\mathcal{D}^{e_{\max}}$ becomes empty and vice-versa.

The approach and Algorithm 7 are similar to the previous one with $\hat{\mathcal{R}}_{e_{\min}}$ the set of stable regions instead of $\mathcal{D}^{e_{\max}}$.

Algorithm 6 Find $e_{\max}$ (largest integer such that the set of breaklines intersecting $\Omega_t(\boldsymbol{x})$ is non-empty)

Input: $\bar{\mathbb{C}}_{\boldsymbol{x}} \subseteq \mathbb{C}$, $t \geq 1$, polygonal domain $\Omega \subseteq \mathbb{C}$.
Output: $e_{\max} \in \mathbb{Z}$.

function HasBreaklines(e)
    Describe with `shapely` the finite set $\mathcal{D}^e$ (breaklines of degree e that intersect Ω).
    return $|\mathcal{R}_e| > 0$
end function
$e_{\max} \leftarrow 0$.
if HasBreaklines($e_{\max}$) **then**
    direction $\leftarrow 1$.
else
    direction $\leftarrow -1$.
end if
while true **do**
    if HasBreaklines($e_{\max}$) **then**
        if direction $= -1$ **then**
            return $e_{\max}$
        end if
        $e_{\max} \leftarrow e_{\max} + 1$.
    else
        if direction $= 1$ **then**
            return $e_{\max}$
        end if
        $e_{\max} \leftarrow e_{\max} - 1$.
    end if
end while

C ON THE NON-DEGENERACY ASSUMPTION

In this section, we deal with how can we describe a couple $(\boldsymbol{x}, \boldsymbol{y})$ that does not satisfy the non-degeneracy assumption. To do so, we will focus on each condition separately.

First, we start by rewriting the first condition 1 of Definition 3.1:

$$\forall \lambda \in \mathbb{B}, \quad |\{\mathbb{D} \in \mathcal{D}^* : \lambda \in \mathbb{D}\}| = 1\,.$$

Algorithm 7 Find $e_{\min}$ (smallest integer such that the set of stable regions is empty)

Input: $\bar{\mathbb{C}}_{\boldsymbol{x}} \subseteq \mathbb{C}$, $t \geq 1$, polygonal domain $\Omega \subseteq \mathbb{C}$.
Output: $e_{\min} \in \mathbb{Z}$.

function HasStableRegions(e)
  Describe with `shapely` the finite set $\mathcal{D}^e$ (breaklines of degree e that intersect Ω).
  Compute $\mathcal{R}_e$ from $\mathcal{D}^e$ using `shapely`.
  $\beta \leftarrow (2^t + 1)2^{e-1-t}$.
  for $z \in \bar{\mathbb{C}}_{\boldsymbol{x}}$ such that $\mathbb{A}_z \cap \Omega \neq \emptyset$ **do**
    Describe with `shapely` the two breaklines $\mathbb{D}(z, \beta)$ and $\mathbb{D}(z, -\beta)$.
    With `shapely`, remove from $\mathcal{R}_e$ all regions lying between $\mathbb{D}(z, \beta)$ and $\mathbb{D}(z, -\beta)$.
  end for
  return $|\mathcal{R}_e| > 0$
end function
$e_{\min} \leftarrow 0$.
if HasStableRegions($e_{\min}$) **then**
  direction $\leftarrow 1$.
else
  direction $\leftarrow -1$.
end if
while true **do**
  if HasStableRegions($e_{\min}$) **then**
    if direction $= -1$ **then**
      return $e_{\min} + 1$
    end if
    $e_{\min} \leftarrow e_{\min} + 1$.
  else
    if direction $= 1$ **then**
      return $e_{\min}$
    end if
    $e_{\min} \leftarrow e_{\min} - 1$.
  end if
end while

We want a characterization of this condition to derive an algorithm to easily check if it holds or not. The following lemma gives a first simple characterization that will allows us to derive another characterization.

Lemma C.1. *Consider $\boldsymbol{x} \in \mathbb{C}^m$, $\boldsymbol{y} \in \mathbb{C}^n$ and $t \geq 1$. The couple $(\boldsymbol{x}, \boldsymbol{y})$ fails the condition 1 of the non-degeneracy assumption given in Definition 3.1 if and only if there exists $((z, z_1, z_2), (\beta_1, \beta_2)) \in \bar{\mathbb{C}}^3_{\boldsymbol{x}} \times (\mathbb{M}^*_t)^2$ such that $z \neq z_1 \neq z_2 \neq z$ and $\mathbb{A}_z \cap \mathbb{D}(z_1, \beta_1) \cap \mathbb{D}(z_2, \beta_2) \neq \emptyset$.*

Proof. The proof is trivial because the first condition of the non-degeneracy assumption fails if and only if two ordinary breaklines intersect on an accumulation line. The first constraint on the normalized directions z, z_1, z_2 is there to prevent the three lines from being parallel because in $\bar{\mathbb{C}}_{\boldsymbol{x}}$ no two normalized directions are colinear. □

This simple characterization allows us to derive a more sophisticated characterization given in the next lemma.

Lemma C.2. *Consider $\boldsymbol{x} \in \mathbb{C}^m$, $\boldsymbol{y} \in \mathbb{C}^n$ and $t \geq 1$. The couple $(\boldsymbol{x}, \boldsymbol{y})$ fails the condition 1 of the non-degeneracy assumption given in Definition 3.1 if and only if there exists $((z, z_1, z_2), (\beta_1, \beta_2)) \in \bar{\mathbb{C}}^3_{\boldsymbol{x}} \times (\mathbb{M}^*_t)^2$ such that $z \neq z_1 \neq z_2 \neq z$*

and

$$\beta_1 \operatorname{Im}(\bar{z}z_2) = \beta_2 \operatorname{Im}(\bar{z}z_1) \neq 0\,. \tag{24}$$

We postpone the proof to directly provide an algorithm to check if this condition holds or not. According to Lemma C.2, to check if a couple $(\boldsymbol{x}, \boldsymbol{y})$ satisfies the non-degeneracy assumption, then we can loop over $\bar{\mathbb{C}}^3_{\boldsymbol{x}} \times (\mathbb{M}^*_t)^2$ to see if (24) holds or not. However, this is not possible because $\mathbb{M}^*_t$ has infinitely many elements. Luckily, any element from $\mathbb{M}^*_t$ can be written as $s(2k+1)2^{e-1-t}$, where $s \in \{-1, 1\}$, $k \in [\![2^{t-1}, 2^t - 1]\!]$ and $e \in \mathbb{Z}$, therefore, by taking the absolute value, given z, z_1, z_2, the existence of $\beta_1, \beta_2 \in \mathbb{M}^*_t$ satisfying (24) is equivalent to the existence of integers $k_1, k_2 \in [\![2^{t-1}, 2^t - 1]\!]$, $e_1, e_2 \in \mathbb{Z}$ such that

$$(2k_1+1)2^{e_1-1-t}|\operatorname{Im}(\bar{z}z_2)| = (2k_2+1)2^{e_2-1-t}|\operatorname{Im}(\bar{z}z_1)|\,,$$

or equivalently such that

$$\left|\frac{\operatorname{Im}(\bar{z}z_2)}{\operatorname{Im}(\bar{z}z_1)}\right| = \frac{2k_2+1}{2k_1+1}2^{e_2-e_1}\,. \tag{25}$$

Since all numbers z, z_1, z_2 are stored in finite arithmetic, $\left|\frac{\operatorname{Im}(\bar{z}z_2)}{\operatorname{Im}(\bar{z}z_1)}\right|$ is a rational number of the form $\frac{p}{q}2^a$, where p and q are odd coprime integers and a is the 2-adic valuation of either p or q. Therefore, the existence of integers $k_1, k_2 \in [\![2^{t-1}, 2^t - 1]\!]$, $e_1, e_2 \in \mathbb{Z}$ satisfying (25) is equivalent to the existence of integers $k_1, k_2 \in [\![2^{t-1}, 2^t - 1]\!]$ such that

$$\frac{p}{q} = \frac{2k_2+1}{2k_1+1}\,.$$

When such integers k_1, k_2 exist, since p and q are both coprime and odd, with the Gauss lemma, there exists $k \in \mathbb{N}$ such that $2k_1 + 1 = (2k+1)p$ and $2k_2 + 1 = (2k+1)q$, and vice-versa. To summarize, to check if a couple $(\boldsymbol{x}, \boldsymbol{y})$ fails the first condition of Definition 3.1, we have to iterate over $(z, z_1, z_2) \in \bar{\mathbb{C}}^3_{\boldsymbol{x}}$ such that $z \neq z_1 \neq z_2 \neq z$ and test if there exists $k \in \mathbb{N}$ such that $(2k+1)p, (2k+1)q \in [\![2^t + 1, 2^{t+1} - 1]\!]$. The whole procedure is described in Algorithm 9. Its time complexity is $O\left(m^3\right)$.

Proof of Lemma C.2. According to Lemma C.1, a couple $(\boldsymbol{x}, \boldsymbol{y})$ fails the first condition of the non-degeneracy assumption if and only if there exists $((z, z_1, z_2), (\beta_1, \beta_2)) \in \bar{\mathbb{C}}^3_{\boldsymbol{x}} \times (\mathbb{M}^*_t)^2$ such that $z \neq z_1 \neq z_2 \neq z$ and $\mathbb{A}_z \cap \mathbb{D}(z_1, \beta_1) \cap \mathbb{D}(z_2, \beta_2) \neq \emptyset$. We consider such a couple $((z, z_1, z_2), (\beta_1, \beta_2))$ and also an associated $\lambda \in \mathbb{A}_z \cap \mathbb{D}(z_1, \beta_1) \cap \mathbb{D}(z_2, \beta_2)$.

- $\lambda \in \mathbb{A}_z$ implies that there exists $\alpha \in \mathbb{R}^*$ such that $\lambda = \alpha i\bar{z}$ because $\operatorname{Re}(\alpha i\bar{z}z) = 0$.
- $\lambda \in \mathbb{D}(z_1, \beta_1)$ implies that $\operatorname{Re}(\alpha i\bar{z}z_1) = \beta_1$, therefore

  $$\alpha = -\frac{\beta_1}{\operatorname{Im}(\bar{z}z_1)}\,. \tag{26}$$

  $\operatorname{Im}(\bar{z}z_1) \neq 0$ otherwise $\beta_1 = 0$ which is not possible since β_1 belongs to $\mathbb{M}^*_t$, a set which does not include zero.
- $\lambda \in \mathbb{D}(z_2, \beta_2)$ implies, like in the previous case, that

  $$\alpha = -\frac{\beta_2}{\operatorname{Im}(\bar{z}z_2)}\,, \tag{27}$$

  and $\operatorname{Im}(\bar{z}z_2) \neq 0$.

With (26) and (27), we deduce that indeed

$$\beta_1 \operatorname{Im}(\bar{z}z_2) = \beta_2 \operatorname{Im}(\bar{z}z_1) \neq 0\,.$$

Reciprocally, if there exists $((z, z_1, z_2), (\beta_1, \beta_2)) \in \bar{\mathbb{C}}^3_{\boldsymbol{x}} \times (\mathbb{M}_t^*)^2$ such that $z \neq z_1 \neq z_2 \neq z$ and (24) holds, then we can define $\lambda := -\frac{\beta_1}{\mathrm{Im}(\bar{z}z_1)} i\bar{z} = -\frac{\beta_2}{\mathrm{Im}(\bar{z}z_2)} i\bar{z}$ and check

- $\mathrm{Re}(\lambda z) = -\frac{\beta_1}{\mathrm{Im}(\bar{z}z_1)} \mathrm{Re}(i\bar{z}z) = 0$, so $\lambda \in \mathbb{A}_z$;
- $\mathrm{Re}(\lambda z_1) = -\frac{\beta_1}{\mathrm{Im}(\bar{z}z_1)} \mathrm{Re}(i\bar{z}z_1) = \beta_1$, so $\lambda \in \mathbb{D}(z_1, \beta_1)$;
- $\mathrm{Re}(\lambda z_2) = -\frac{\beta_2}{\mathrm{Im}(\bar{z}z_2)} \mathrm{Re}(i\bar{z}z_2) = \beta_2$, so $\lambda \in \mathbb{D}(z_2, \beta_2)$.

Therefore, $\lambda \in \mathbb{A}_z \cap \mathbb{D}(z_1, \beta_1) \cap \mathbb{D}(z_2, \beta_2)$, hence this set is non-empty. □

Now, we want to deal with the second condition 2 of Definition 3.1:

$$\forall \lambda \in \mathbb{A} \setminus \mathbb{B}, \forall z \in \mathbb{C}_{\boldsymbol{y}}, \quad \mathrm{Re}(\mu(\hat{\boldsymbol{x}}(\lambda))z) \notin \mathbb{M}_t \,,$$

where $\mu(\cdot)$ is defined in (4). Characterizing this condition is easier than the first one. Indeed, as the function $\lambda \in \mathbb{A} \setminus \mathbb{B} \mapsto \mu(\hat{\boldsymbol{x}}(\lambda))$, is piecewise constant, we can evaluate it only on $\Lambda_{\mathbb{A}}$, the midpoints of the line segments of $\mathbb{A} \setminus \mathbb{B}$, therefore, the couple $(\boldsymbol{x}, \boldsymbol{y})$ fails the second condition of the non-degeneracy assumption if and only if there exist $\lambda \in \Lambda_{\mathbb{A}}$ and $z \in \mathbb{C}_{\boldsymbol{y}}$ such that $\mathrm{Re}(\mu(\lambda)z) \in \mathbb{M}_t$. To check if a number is in $\mathbb{M}_t$, then we write it as $s(2k+1)2^{e-1-t}$ and we perform the same derivation as for the first condition: we write $|\mathrm{Re}(\mu(\lambda)z)|$ as $\frac{p}{q}2^a$, where p and q are two odd coprime integers and a is the 2-adic valuation of either p or q. Hence, the Gauss lemma states that we have to iterate on $(\lambda, z) \in \Lambda_{\mathbb{A}} \times \mathbb{C}_{\boldsymbol{y}}$ and check if $q = 1$ and if $p \in [\![2^t + 1, 2^{t+1} - 1]\!]$. This procedure is described in Algorithm 10. Its time complexity is $O\left(m^2 n 2^t\right)$.

As Algorithm 5 swaps the role of $\boldsymbol{x}$ and $\boldsymbol{y}$ depending on their dimension, Algorithm 8 performs the same swap and has a time complexity

$$O\left(\min(m,n)^2 \cdot \max(m,n) \cdot 2^t\right) .$$

Algorithm 8 Check if a couple $(\boldsymbol{x}, \boldsymbol{y})$ satisfies the non-degeneracy assumption from Definition 3.1

Input: $\boldsymbol{x} \in \mathbb{C}^m, \boldsymbol{y} \in \mathbb{C}^n, t \geq 1$.
Output: `true` if the input couple does not satisfy the non-degeneracy assumption and `false` otherwise.

1: **if** $m \leq n$ **then**
2: Build $\mathbb{C}_{\boldsymbol{x}}$ from (6) and its *normalized* version $\bar{\mathbb{C}}_{\boldsymbol{x}}$ from (17).
3: **return** CheckCondition1$(\bar{\mathbb{C}}_{\boldsymbol{x}}, t)$ ∨ CheckCondition2$(\bar{\mathbb{C}}_{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{y}, t)$.
4: **else**
5: Build $\mathbb{C}_{\boldsymbol{y}}$ from (6) and its *normalized* version $\bar{\mathbb{C}}_{\boldsymbol{y}}$ from (17).
6: **return** CheckCondition1$(\bar{\mathbb{C}}_{\boldsymbol{y}}, t)$ ∨ CheckCondition2$(\bar{\mathbb{C}}_{\boldsymbol{y}}, \boldsymbol{y}, \boldsymbol{x}, t)$.
7: **end if**

Algorithm 9 Check if the first condition of Definition 3.1 is satisfied

Input: $\bar{\mathbb{C}}_{\boldsymbol{x}} \subseteq \mathbb{C}$, $t \geq 1$.
Output: `true` if the first condition is not satisfied and `false` otherwise.

for $z \in \bar{\mathbb{C}}_{\boldsymbol{x}}$ **do**
  for $z_1 \in \bar{\mathbb{C}}_{\boldsymbol{x}}$ such that $z_1 \neq z$ **do**
    for $z_2 \in \bar{\mathbb{C}}_{\boldsymbol{x}}$ such that $z_2 \neq z$ and $z_2 \neq z_1$ **do**
      Find $(p, q, a) \in \mathbb{N} \times \mathbb{N}^* \times \mathbb{N}$ such that $\frac{p}{q} 2^a = \left| \frac{\mathrm{Im}(\bar{z} z_2)}{\mathrm{Im}(\bar{z} z_1)} \right|$ and $\gcd(p, q) = 1$.
      if $\left[\!\left[\lceil \frac{2^t+1}{p} \rceil, \lfloor \frac{2^{t+1}}{p} \rfloor \right]\!\right] \cap \left[\!\left[\lceil \frac{2^t+1}{q} \rceil, \lfloor \frac{2^{t+1}-1}{q} \rfloor \right]\!\right] \cap (2\mathbb{N}+1) \neq \emptyset$ **then**
        return `true`.
      end if
    end for
  end for
end for
return `false`.

Algorithm 10 Check if the second condition of Definition 3.1 is satisfied

Input: $\bar{\mathbb{C}}_{\boldsymbol{x}} \subseteq \mathbb{C}$, $\boldsymbol{x} \in \mathbb{C}^m$, $\boldsymbol{y} \in \mathbb{C}^n$, $t \geq 1$.
Output: `true` if the second condition is not satisfied and `false` otherwise.

Build $\mathbb{C}_{\boldsymbol{y}}$ from (6).
$\Omega \leftarrow$ BuildDomain$(\bar{\mathbb{C}}_{\boldsymbol{x}}, t)$ from Algorithm 2.
$\Lambda_{\mathbb{A}} \leftarrow$ BuildSegmentCentroids$(\bar{\mathbb{C}}_{\boldsymbol{x}}, t, \Omega)$ from Algorithm 3.
for $\lambda \in \Lambda_{\mathbb{A}}$ **do**
  $\hat{\boldsymbol{x}} \leftarrow \mathrm{round}(\lambda \boldsymbol{x})$.
  $\mu \leftarrow \frac{\langle \hat{\boldsymbol{x}}, \boldsymbol{x} \rangle}{\|\hat{\boldsymbol{x}}\|^2}$.
  for $z \in \mathbb{C}_{\boldsymbol{y}}$ **do**
    if $\mathrm{Re}(\mu z) = 0$ **then**
      return `true`.
    end if
    Find $(p, q, a) \in \mathbb{N} \times \mathbb{N}^* \times \mathbb{N}$ such that $\frac{p}{q} 2^a = |\mathrm{Re}(\mu z)|$ and $\gcd(p, q) = 1$.
    if $(q = 1) \wedge (p \in [\![2^t + 1, 2^{t+1} - 1]\!])$ **then**
      return `true`.
    end if
  end for
end for
return `false`.