Bivariate Hermite interpolation by a limiting case of the cross approximation algorithm

Yannick Masson, Bert Jüttler

yannick.masson@ricam.oeaw.ac.at, bert.juettler@jku.at

Altenberger Str. 69, 4040 Linz,
Institute of Applied Geometry, Johannes Kepler Universität Linz, Austria

Abstract

Computing low-rank approximations of a given function is a key step for implementing efficiently numerous algorithms in various fields, including the discretisation of non-local integral operators and Isogeometric Analysis. The adaptive cross approximation (ACA) algorithm is an efficient method requiring few computational resources introduced by Bebendorf. We introduce in the present paper the new paradigm of approximating the given function by a piecewise low-rank function with C^1-regularity. The proposed approximation is based on the ACA algorithm and our main contribution is the extension of the interpolation property characterising this algorithm to Hermite interpolation. Therefore, we introduce a new method for low-rank Hermite interpolation using a limiting case of the ACA algorithm. The proposed method has full approximation order. We then propose a piecewise low-rank approximation with adaptive refinement using either the ACA algorithm or our new method to compute each piece. We finally compare the results obtained for the two methods.

Keywords: cross approximation, Hermite interpolation, low-rank approximation

1. Introduction

In the present paper we focus on the approximation of a given bivariate function with a sum of products of univariate functions. This so-called low-rank approximation has received a lot of attention, see [1] and references therein, and [2]. A first approach to perform this approximation uses the singular value decomposition [3, 4]. This method has clear theoretical foundations, but is computationally expensive. A method reducing drastically the resources needed, the adaptive cross approximation (ACA) algorithm, has been introduced by Bebendorf in [2].

To estimate the error made by this approximation, the author focused on the case of integral operators with kernels satisfying a reasonable asymptotic decay. Schneider then showed in [5] that, under a particular choice of the points, the error can be controlled by the infimum of the error in the L^∞-norm over all the functions having the expected rank.

One can find in [6] a review of low-rank approximations of matrices, including the ACA algorithm for the case where the matrix arises from the sampling of a function on a grid.

Such approximations are notably used to implement efficient discretisations of integral operators. Indeed, a specificity of these non-local operators is that the resulting matrices are dense. Nevertheless, under some reasonable asymptotic assumptions on the kernel, a data sparse approximation of these matrices follows from their decomposition into blocks with low-rank [7, 8, 9]. Moreover, the low-rank approximation of the blocks is a direct consequence of the low-rank approximation of the considered kernel.

More recently, in the field of Isogeometric Analysis [10], the authors of [11] have used low-rank approximations to improve significantly the implementation of the discretisation of partial differential equations. Indeed, to operate isogeometric discretisations of PDEs, one mainly uses tensor spline bases mapped to the physical domain by a global parameterisation. The authors of [11] then used the tensor structure to decompose the Galerkin matrix into Kronecker products of matrix factors with small dimensions. Similarly to the case of integral equations, the key step is a low-rank approximation of the kernel (density of the integral).

An algorithm to approximate a bivariate function by low-rank splines using the ACA algorithm has then been proposed in [12]. The decomposition of the matrix obtained in [11] is even crucial when considering higher dimensions to avoid prohibitive computational costs [13, 14]. The rank of the aforementioned kernel depending directly on the rank of the parameterisation, another way to obtain low-rank kernels is to directly construct low-rank parameterisations. To this end, an optimisation approach has successfully been applied in [15] to construct low-rank parameterisations of physical domains delimited by four given curves.

On the other hand, interpolation methods, such as Coons patches [16], can be applied to construct parameterisations. An interpolation algorithm generating low-rank parameterisations has then been proposed in [17]. In a second paper, the authors have generalised this method in order to interpolate multiple curves [18]. They also showed that this method is equivalent to the ACA algorithm, this latter having the advantage of choosing adaptively the points. Indeed, the ACA algorithm performs a low-rank approximation of a given function using the restrictions of this function to some x-lines and y-lines, permitting it to be equivalently used as an interpolation method.

Contrary to the error estimates of the ACA algorithm presented in [2, 5], the authors of [18] focus on local estimations and show that this method has full approximation order. This motivates the approach taken in the present paper: we approximate hierarchically a given function by a piecewise low-rank function. The main bottleneck is to obtain C^1-continuity of the result, leading us to low-rank Hermite interpolation. We note that we could obtain Hermite interpolation using blending functions [19], but the result may have high rank. Therefore, we focus in the first sections of this paper on the development of a new method for low-rank Hermite interpolation.

The proposed method results from a limiting case of the ACA algorithm, renamed as the cross algorithm (CA) in what follows to underline that the points are fixed in our approach. A closed-form solution is first proposed. We then present an efficient implementation of

this closed-form solution using an extension of the CA algorithm. We also prove that this new method has full approximation order. We finally show the efficiency of the piecewise low-rank approximation by numerical examples in the last section.

2. Preliminaries and main results

After recalling the *Cross Approximation* (CA) algorithm and its properties [2, 18], we outline below the main ideas of the present paper, as well as the main results.

The CA algorithm permits to approximate a bivariate function $f \in C^{1,1}(\Omega)$ defined on a tensor domain $\Omega = \Omega_x \times \Omega_y$, with $\Omega_x, \Omega_y \subset \mathbb{R}$ two bounded intervals, by a sum of few products, say $n \geq 1$, of univariate functions. To recall this algorithm, let us first pick the following real values: $x_1, ..., x_n \in \Omega_x$ and $y_1, ..., y_n \in \Omega_y$. We then denote respectively by $L_1, ..., L_n$ and $C_1, ..., C_n$ the x-lines and the y-lines associated with these points, i.e.,

$$L_i = \{(x, y_i), \quad x \in \Omega_x\}, \text{ and } C_i = \{(x_i, y), \quad y \in \Omega_y\},$$

for all $i = 1, ..., n$. The key step of this algorithm is the construction of the following cross interpolation operators:

$$\mathrm{I}_i[f] = \frac{f(\cdot, y_i) f(x_i, \cdot)}{f(x_i, y_i)},$$

for all $i = 1, ..., n$ such that $f(x_i, y_i) \neq 0$. Clearly, the function $\mathrm{I}_i[f]$ interpolates the values of f along the x-line L_i and the y-line C_i crossing at (x_i, y_i), i.e.,

$$\mathrm{I}_i[f](\cdot, y_i) = f(\cdot, y_i) \quad \text{and} \quad \mathrm{I}_i[f](x_i, \cdot) = f(x_i, \cdot). \tag{1}$$

Hence, using the operators $\mathrm{I}_1, ..., \mathrm{I}_n$, the values of f on the x-lines $L_1, ..., L_n$ and on the y-lines $C_1, ..., C_n$ can be interpolated one by one thanks to the following property:

Property 1 (No fill-in)**.** *Let $\ell \in \{2, ..., n\}$ and $i \in \{1, ..., \ell-1\}$, and suppose that $f(x_i, \cdot) = f(\cdot, y_i) = 0$ and $f(x_\ell, y_\ell) \neq 0$. Then, $\mathrm{I}_\ell[f](x_i, \cdot) = \mathrm{I}_\ell[f](\cdot, y_i) = 0$.*

The CA algorithm, which performs this iterative interpolation, is presented above. This algorithm can be applied, up to a permutation of the chosen points, whenever the $n \times n$ matrix S containing the sampling of f, i.e.,

$$\mathrm{S}_{ij} = f(x_j, y_i), \tag{2}$$

for all $i, j = 1, ..., n$, is invertible [18]. Under this condition, it follows from Property 1 that the output Φ_n of the CA algorithm satisfies the following interpolation property.

Property 2 (Value interpolation)**.** *The function Φ_n interpolates the values of f on the x-lines $L_1, ..., L_n$ and on the y-lines $C_1, ..., C_N$, i.e.,*

$$\Phi_n(\cdot, y_i) = f(\cdot, y_i), \quad \Phi_n(x_i, \cdot) = f(x_i, \cdot), \tag{3}$$

for all $i = 1, ..., n$.

Input : $f : \Omega \to \mathbb{R}$, $x_1, ..., x_n \in \Omega_x$, $y_1, ..., y_n \in \Omega_y$.
Output : rank n function $\Phi_n : \Omega \to \mathbb{R}$.
$\Phi_0 \leftarrow 0$;
$\mathrm{R}_0 \leftarrow f$;
for $i \leftarrow 1$ **to** n **do**
  if $\mathrm{R}_{i-1}(x_i, y_i) \neq 0$ **then**
    $\mathrm{I}_i \leftarrow \frac{\mathrm{R}_{i-1}(\cdot, y_i)\,\mathrm{R}_{i-1}(x_i, \cdot)}{\mathrm{R}_{i-1}(x_i, y_i)}$;
    $\Phi_i \leftarrow \Phi_{i-1} + \mathrm{I}_i$;
    $\mathrm{R}_i \leftarrow \mathrm{R}_{i-1} - \mathrm{I}_i$;
  else
    Error during Cross Approximation $\to$ stop;
  end
end

Algorithm CA: Cross Approximation

Some error estimates of the approximation provided by the CA algorithm can be found in [2, 5]. We focus in what follows on the error estimates presented in [18] which we restate in Theorem 3.

Theorem 3 (Dyn, Jüttler, Mokriš, 2017)**.** *Let $\bar{h} > 0$ and $f \in C^{n,n}([0, \bar{h}]^2)$ and suppose that*

$$c_f = \frac{\max\limits_{\substack{\hat{x}_{ij}, \hat{y}_{ij} \in [0,\bar{h}] \\ i,j=0,...,n}} \det\left(\partial_x^i \partial_y^j f(\hat{x}_{ij}, \hat{y}_{ij})\right)_{i,j=0,...,n}}{\min\limits_{\substack{\check{x}_{ij}, \check{y}_{ij} \in [0,\bar{h}] \\ i,j=0,...,n-1}} \det\left(\partial_x^i \partial_y^j f(\check{x}_{ij}, \check{y}_{ij})\right)_{i,j=0,...,n-1}} < \infty.$$

We set moreover $h \in (0, \bar{h})$, $\Omega_x = \Omega_y = [0, h]$, and $\Omega = \Omega_x \times \Omega_y$, and we suppose that $\det S \neq 0$. *Then,*

$$\|f - \Phi_n\|_{L^\infty(\Omega)} \leq c_f \frac{h^{2n}}{(n!)^2}.$$

Remark. Remarkably, the coefficient c_f does not depend on the chosen reals $x_1, ..., x_n$ and $y_1, ..., y_n$. Nevertheless, the properties of the interpolant depend on the choice of these numbers. In practice, one may use the end points of the intervals defining the tensor domain Ω and equally distributed points in the interior of the intervals. This choice is also suitable for the construction of globally smooth spline-type interpolants. Alternatively [2, 5] one may choose these numbers adaptively by always picking them as the coordinates of the point with the largest residual error.

In order to investigate a limiting case of the CA algorithm, we first consider a small real $\varepsilon > 0$ satisfying

$$\varepsilon < \frac{1}{2} \min\left(\min_{i \in \{1,...,n-1\}} |x_{i+1} - x_i|, \min_{j \in \{1,...,n-1\}} |y_{j+1} - y_j| \right). \tag{4}$$

When applying the CA algorithm with the *double grid*, emphasised by II, and defined by the real values

$$\begin{aligned} &x_1, x_1{+}\varepsilon, x_2, x_2{+}\varepsilon, ..., x_{n-1}{+}\varepsilon, x_n{-}\varepsilon, x_n \in \Omega_x, \text{ and,} \\ &y_1, y_1{+}\varepsilon, y_2, y_2{+}\varepsilon, ..., y_{n-1}{+}\varepsilon, y_n{-}\varepsilon, y_n \in \Omega_y, \end{aligned} \tag{5}$$

the coefficient c_f introduced in Theorem 3 is independent of ε. The sampling of f is, for this grid, given by the following matrix:

$$\mathrm{S}^{\mathrm{II}}_{\varepsilon} = \begin{pmatrix} f(x_1,y_1) & f(x_1{+}\varepsilon,y_1) & \cdots & f(x_n{-}\varepsilon,y_1) & f(x_n,y_1) \\ f(x_1,y_1{+}\varepsilon) & f(x_1{+}\varepsilon,y_1{+}\varepsilon) & \cdots & f(x_n{-}\varepsilon,y_1{+}\varepsilon) & f(x_n,y_1{+}\varepsilon) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ f(x_1,y_n{-}\varepsilon) & f(x_1{+}\varepsilon,y_n{-}\varepsilon) & \cdots & f(x_n{-}\varepsilon,y_n{-}\varepsilon) & f(x_n,y_n{-}\varepsilon) \\ f(x_1,y_n) & f(x_1{+}\varepsilon,y_n) & \cdots & f(x_n{-}\varepsilon,y_n) & f(x_n,y_n) \end{pmatrix}.$$

Clearly the determinant of the matrix $\mathrm{S}^{\mathrm{II}}_{\varepsilon}$ converges to zero whenever ε goes to zero. Though, we show in the next section that the output $\Phi^{\mathrm{II}}_{n,\varepsilon}$ of the CA algorithm using the points (5) converges under the same limit.

On the other hand, extending Property 2 of the CA algorithm to Hermite interpolation is an important step to construct C^1-smooth piecewise low-rank functions. With this purpose in mind, we introduce in the next section the *Low-Rank Hermite Interpolation* (LRHI) and we show that $\Phi^{\mathrm{II}}_{n,\varepsilon}$ converges to the unique solution Ψ_n of LRHI whenever ε goes to zero. To be more precise, let us first introduce some samplings of the partial derivatives of f. Using ∂ to highlight the source of the sampling, we denote by $\mathrm{S}^{\partial x}, \mathrm{S}^{\partial y}$, and $\mathrm{S}^{\partial xy}$, the matrices of size $n \times n$ respectively defined by

$$\mathrm{S}^{\partial x}_{ij} = \partial_x f(x_j, y_i), \;\; \mathrm{S}^{\partial y}_{ij} = \partial_y f(x_j, y_i), \text{ and } \mathrm{S}^{\partial xy}_{ij} = \partial_{xy} f(x_j, y_i),$$

for all $i, j = 1, ..., n$. Next, let M be the matrix of size $2n \times 2n$ defined as follows:

$$M = \begin{pmatrix} \mathrm{S} & \mathrm{S}^{\partial x} \\ \mathrm{S}^{\partial y} & \mathrm{S}^{\partial xy} \end{pmatrix}. \tag{6}$$

The convergence result is stated in the following theorem.

Theorem 4 (LRHI as limiting case of the CA algorithm)**.** *Let $\Omega_x, \Omega_y \subset \mathbb{R}$ be two bounded intervals and set $\Omega = \Omega_x \times \Omega_y$. Let $f \in C^{1,1}(\Omega)$ and suppose $\det M \neq 0$. Then, for all $\varepsilon > 0$ small enough, the matrix $\mathrm{S}^{\mathrm{II}}_{\varepsilon}$ is invertible and*

$$\lim_{\varepsilon \to 0} \Phi^{\mathrm{II}}_{n,\varepsilon} = \Psi_n,$$

where Ψ_n is the unique solution of LRHI defined in (11)*.*

A direct consequence of Theorems 3 and 4 is that the order of approximation stated for the CA algorithm is also valid for the solution of LRHI.

Theorem 5 (Order of approximation $4n$)**.** *Let $\bar{h} > 0$ and $f \in C^{2n,2n}([0,\bar{h}]^2)$ and suppose that*

$$c_f = \frac{\max\limits_{\substack{\hat{x}_{ij},\hat{y}_{ij}\in[0,\bar{h}]\\ i,j=0,\dots,2n}} \det\left(\partial_x^i\partial_y^j f(\hat{x}_{ij},\hat{y}_{ij})\right)_{i,j=0,\dots,2n}}{\min\limits_{\substack{\check{x}_{ij},\check{y}_{ij}\in[0,\bar{h}]\\ i,j=0,\dots,2n-1}} \det\left(\partial_x^i\partial_y^j f(\check{x}_{ij},\check{y}_{ij})\right)_{i,j=0,\dots,2n-1}} < \infty.$$

We set moreover $h \in (0,\bar{h})$, $\Omega_x = \Omega_y = [0,h]$, and $\Omega = \Omega_x \times \Omega_y$, and we suppose that $\det M \neq 0$*. Then,*

$$\|f - \Psi_n\|_{L^\infty(\Omega)} \leq c_f \frac{h^{4n}}{\big((2n)!\big)^2},$$

where Ψ_n is the unique solution of LRHI defined in (11)*.*

Remark. The observations made in the previous remark apply to the Hermite setting, too. In order to construct globally C^1 smooth spline-type interpolants, one may use the end points of the intervals defining the tensor domain Ω and equally distributed points in the interior of the intervals.

3. Low-rank Hermite interpolation

In this section we introduce the *Low-Rank Hermite Interpolation* (LRHI): an extension of the closed-form formula satisfied by the output of the CA algorithm permitting Hermite interpolation. We then show that LRHI is a limiting case of the CA algorithm (Theorem 4).

Before this, let us recall the closed-form expression of the output of the CA algorithm [2, Lemma 3]. To clarify the exposition, we define the vector fields $\eta : \Omega_x \to \mathbb{R}^n$ and $\tau : \Omega_y \to \mathbb{R}^n$ as follows:

$$\eta(x) = \begin{pmatrix} f(x,y_1) \\ \vdots \\ f(x,y_n) \end{pmatrix}, \quad \text{and} \quad \tau(y) = \begin{pmatrix} f(x_1,y) \\ \vdots \\ f(x_n,y) \end{pmatrix},$$

for all $(x,y) \in \Omega$.

Proposition 6 (Closed-form expression for the CA algorithm)**.** *Suppose* $\det \mathrm{S} \neq 0$*. Then,*

$$\Phi_n(x,y) = \langle \mathrm{S}^{-1}\,\eta(x), \tau(y)\rangle, \tag{7}$$

for all $(x,y) \in \Omega$, where Φ_n is the output of the CA algorithm.

Proof. We first remark in the CA algorithm that the output Φ_n only depends on the values of f on the x-lines $L_1, ..., L_n$ and on the y-lines $C_1, ..., C_n$. Moreover, this dependence is clearly linear. In other words, there exists a matrix B of size $n \times n$ such that $\Phi_n(x,y) = \langle B\,\eta(x), \tau(y)\rangle$, for all $(x,y) \in \Omega$. Next, omitting the variables x and y, we note that

$\Phi_n = \langle \lambda, \tau \rangle$, with $\lambda = B\,\eta$. Then, denoting by τ_i and η_i the i-th coordinates of τ and η respectively, we obtain using Property 2 that the function Φ_n interpolates η_i, for all $i = 1, ..., n$, and we infer

$$\sum_{k=1}^{n} \lambda_k \tau_k(y_i) = \eta_i \,.$$

Equivalently, using the sampling S, we have $\mathrm{S}\,\lambda = \eta$. Finally, since S is invertible, we conclude that $B\,\eta = \lambda = \mathrm{S}^{-1}\,\eta$, and the result follows. □

In order to formulate LRHI, let us now introduce the interpolated data. We denote by $\mathrm{H} : \Omega_x \to \mathbb{R}^n$ and $\mathrm{T} : \Omega_y \to \mathbb{R}^n$ the derivatives of f in the complementary direction on the x-lines $L_1, ..., L_n$ and the y-lines $C_1, ..., C_n$ respectively, i.e.,

$$\mathrm{H}(x) = \begin{pmatrix} \partial_y f(x, y_1) \\ \vdots \\ \partial_y f(x, y_n) \end{pmatrix}, \quad \text{and} \quad \mathrm{T}(y) = \begin{pmatrix} \partial_x f(x_1, y) \\ \vdots \\ \partial_x f(x_n, y) \end{pmatrix},$$

for all $(x, y) \in \Omega$. We construct in the following proposition a function $\Psi_n \in C^{1,1}(\Omega)$ which solves the Hermite interpolation problem, i.e.,

$$\Psi_n(\cdot, y_i) = f(\cdot, y_i), \qquad \Psi_n(x_i, \cdot) = f(x_i, \cdot), \tag{8a}$$
$$\nabla\Psi_n(\cdot, y_i) = \nabla f(\cdot, y_i), \qquad \nabla\Psi_n(x_i, \cdot) = \nabla f(x_i, \cdot), \tag{8b}$$

or equivalently

$$\Psi_n(\cdot, y_i) = \eta_i, \qquad \partial_y \Psi_n(\cdot, y_i) = \mathrm{H}_i, \tag{9a}$$
$$\Psi_n(x_i, \cdot) = \tau_i, \qquad \partial_x \Psi_n(x_i, \cdot) = \mathrm{T}_i, \tag{9b}$$

for all $i = 1, ..., n$, where η_i, τ_i, H_i, and T_i, are respectively the i-th coordinates of η, τ, H, and T.

Proposition 7 (Low-rank Hermite interpolation)**.** *Suppose* $\det M \neq 0$. *Then, there exists a unique function* $\Psi_n \in C^{1,1}(\Omega)$ *of the form*

$$\Psi_n = \langle \lambda, \tau \rangle + \langle \tilde{\lambda}, \mathrm{T} \rangle, \tag{10}$$

with $\lambda, \tilde{\lambda} \in \big(C^1(\Omega_x)\big)^n$, *satisfying* (8) *for all* $i = 1, ..., n$. *Moreover, this function is given by*

$$\Psi_n(x, y) = \left\langle M^{-1} \begin{pmatrix} \eta(x) \\ \mathrm{H}(x) \end{pmatrix}, \begin{pmatrix} \tau(y) \\ \mathrm{T}(y) \end{pmatrix} \right\rangle, \tag{11}$$

for all $(x, y) \in \Omega$.

Proof. Firstly, Ψ_n satisfies (9a) if and only if it satisfies the following:

$$\begin{aligned}\Psi_n(\cdot, y_i) = &\sum_{k=1}^{n} \lambda_k \tau_k(y_i) + \sum_{k=1}^{n} \tilde{\lambda}_k \, \mathrm{T}_k(y_i) \quad = \eta_i,\\ \partial_y \Psi_n(\cdot, y_i) = &\sum_{k=1}^{n} \lambda_k \frac{d\tau_k}{dy}(y_i) + \sum_{k=1}^{n} \tilde{\lambda}_k \frac{d\,\mathrm{T}_k}{dy}(y_i) = \mathrm{H}_i,\end{aligned}$$

for all $i = 1, ..., n$, with λ_i and $\tilde{\lambda}_i$ the i-th coordinates of λ and $\tilde{\lambda}$ respectively. Using the matrix notation, we obtain

$$\begin{aligned}\big(\Psi_n(\cdot, y_1), \cdots, \Psi_n(\cdot, y_n)\big)^t = &\quad \mathrm{S}\,\lambda + \mathrm{S}^{\partial x}\,\tilde{\lambda} \;= \eta,\\ \big(\partial_y \Psi_n(\cdot, y_1), \cdots, \partial_y \Psi_n(\cdot, y_n)\big)^t = &\; \mathrm{S}^{\partial y}\,\lambda + \mathrm{S}^{\partial xy}\,\tilde{\lambda} = \mathrm{H}\,.\end{aligned}$$

Hence, since M is invertible, there exists a unique couple $(\lambda, \tilde{\lambda})$ such that Ψ_n satisfies (9a), which is given by

$$\begin{pmatrix}\lambda(x)\\ \tilde{\lambda}(x)\end{pmatrix} = \begin{pmatrix}\mathrm{S} & \mathrm{S}^{\partial x}\\ \mathrm{S}^{\partial y} & \mathrm{S}^{\partial xy}\end{pmatrix}^{-1} \begin{pmatrix}\eta(x)\\ \mathrm{H}(x)\end{pmatrix} = M^{-1} \begin{pmatrix}\eta(x)\\ \mathrm{H}(x)\end{pmatrix},$$

for all $x \in \Omega_x$. Now, to check that the uniquely defined function Ψ_n also satisfies (9b), let us note that

$$M \begin{pmatrix}\lambda(x_j)\\ \tilde{\lambda}(x_j)\end{pmatrix} = \begin{pmatrix}\eta(x_j)\\ \mathrm{H}(x_j)\end{pmatrix}, \quad \text{and} \quad M \begin{pmatrix}\lambda'(x_j)\\ \tilde{\lambda}'(x_j)\end{pmatrix} = \begin{pmatrix}\eta'(x_j)\\ \mathrm{H}'(x_j)\end{pmatrix},$$

for all $j = 1, ..., n$. The right hand side of the first and the second equalities are respectively the columns j and $n + j$ of the matrix M. Hence, we have

$$\begin{aligned}&\lambda_i(x_j) = \begin{cases} 1 & \text{if } j = i,\\ 0, & \text{else,}\end{cases} && \tilde{\lambda}_i(x_j) = 0,\\ &\lambda_i'(x_j) = 0, && \tilde{\lambda}_i'(x_j) = \begin{cases} 1, & \text{if } j = i,\\ 0, & \text{else,}\end{cases}\end{aligned}$$

for all $i, j = 1, ..., n$. We finally apply the definition of Ψ_n to conclude that (9b) is satisfied. The result follows. □

We now prove Theorem 4, i.e., that the output $\Phi^{\mathrm{II}}_{n,\varepsilon}$ of the CA algorithm applied to the *double grid* (5) converges to the function Ψ_n, whenever ε goes to 0. To adapt Formula (11), we denote by $\eta^{\mathrm{II}}_\varepsilon$ and $\tau^{\mathrm{II}}_\varepsilon$ the value of f on the x-lines and the y-lines associated with the double grid (5), i.e.,

$$\begin{aligned}\eta^{\mathrm{II}}_\varepsilon(x) &= \big(f(x, y_1), f(x, y_1{+}\varepsilon), \cdots, f(x, y_n{-}\varepsilon), f(x, y_n)\big)^t, \text{ and}\\ \tau^{\mathrm{II}}_\varepsilon(y) &= \big(f(x_1, y), f(x_1{+}\varepsilon, y), \cdots, f(x_n{-}\varepsilon, y), f(x_n, y)\big)^t,\end{aligned}$$

for all $(x,y)\in\Omega$. Supposing $\det \mathrm{S}_\varepsilon^{\mathrm{II}} \neq 0$, we then have by Proposition 7:

$$\Phi_{n,\varepsilon}^{\mathrm{II}}(x,y) = \big\langle (\mathrm{S}_\varepsilon^{\mathrm{II}})^{-1}\eta_\varepsilon^{\mathrm{II}}(x), \tau_\varepsilon^{\mathrm{II}}(y)\big\rangle, \tag{12}$$

for all $(x,y)\in\Omega$. We note that we can not directly compute the limit $\varepsilon\to 0$ in (12) since $\mathrm{S}_\varepsilon^{\mathrm{II}}$ converges to a matrix which is not invertible. Therefore, the first step is to apply some linear combinations on the lines and the columns of the matrix and the vectors involved in (12). The resulting expression, presented in the next lemma, contains the following finite difference operators:

$$\begin{aligned}
\delta_\varepsilon^{x,\pm} f(x,y) &= \pm\frac{f(x\pm\varepsilon,y)-f(x,y)}{\varepsilon}, \text{ for all } (x,y)\in\Omega \text{ s.t. } (x\pm\varepsilon,y)\in\Omega,\\
\delta_\varepsilon^{y,\pm} f(x,y) &= \pm\frac{f(x,y\pm\varepsilon)-f(x,y)}{\varepsilon}, \text{ for all } (x,y)\in\Omega \text{ s.t. } (x,y\pm\varepsilon)\in\Omega.
\end{aligned} \tag{13}$$

Lemma 8 (Equivalence with finite differences)**.** *Let $\varepsilon>0$ satisfy* (4) *and suppose* $\det \mathrm{S}_\varepsilon^{\mathrm{II}} \neq 0$. *Then,*

$$\Phi_{n,\varepsilon}^{\mathrm{II}}(x,y) = \big\langle \tilde{M}_\varepsilon^{-1}\tilde{\eta}_\varepsilon(x), \tilde{\tau}_\varepsilon(y)\big\rangle, \tag{14}$$

for all $(x,y)\in\Omega$, with:

$$\tilde{M}_\varepsilon = \begin{pmatrix}
f(x_1,y_1) & \delta_\varepsilon^{x,+}f(x_1,y_1) & \cdots & f(x_n,y_1) & \delta_\varepsilon^{x,-}f(x_n,y_1)\\
\delta_\varepsilon^{y,+}f(x_1,y_1) & \delta_\varepsilon^{x,+}\delta_\varepsilon^{y,+}f(x_1,y_1) & \cdots & \delta_\varepsilon^{y,+}f(x_n,y_1) & \delta_\varepsilon^{x,-}\delta_\varepsilon^{y,+}f(x_n,y_1)\\
\vdots & \vdots & \ddots & \vdots & \vdots\\
f(x_1,y_n) & \delta_\varepsilon^{x,+}f(x_1,y_n) & \cdots & f(x_n,y_n) & \delta_\varepsilon^{x,-}f(x_n,y_n)\\
\delta_\varepsilon^{y,-}f(x_1,y_n) & \delta_\varepsilon^{x,+}\delta_\varepsilon^{y,-}f(x_1,y_n) & \cdots & \delta_\varepsilon^{y,-}f(x_n,y_n) & \delta_\varepsilon^{x,-}\delta_\varepsilon^{y,-}f(x_n,y_n)
\end{pmatrix},$$

$$\tilde{\eta}_\varepsilon(x) = \begin{pmatrix} f(x,y_1)\\ \delta_\varepsilon^{y,+}f(x,y_1)\\ \vdots\\ f(x,y_n)\\ \delta_\varepsilon^{y,-}f(x,y_n)\end{pmatrix}, \quad \textit{and} \quad \tilde{\tau}_\varepsilon(y) = \begin{pmatrix} f(x_1,y)\\ \delta_\varepsilon^{x,+}f(x_1,y)\\ \vdots\\ f(x_n,y)\\ \delta_\varepsilon^{x,-}f(x_n,y)\end{pmatrix}.$$

Moreover, we have

$$|\det \mathrm{S}_\varepsilon^{\mathrm{II}}| = \varepsilon^{2n}|\det \tilde{M}_\varepsilon|. \tag{15}$$

Proof. The variables x and y will be omitted in this proof. Firstly, we denote by λ the unique solution of the linear system $\mathrm{S}_\varepsilon^{\mathrm{II}}\lambda = \eta$, so that $\Phi_{n,\varepsilon}^{\mathrm{II}} = \langle \lambda, \tau_\varepsilon^{\mathrm{II}}\rangle$. We then apply the following linear combinations to the rows $\mathcal{R}_1,...,\mathcal{R}_n$ of this linear system:

$$(\mathcal{R}'_{2i-1}) \leftarrow (\mathcal{R}_{2i-1}), \quad (\mathcal{R}'_{2i}) \leftarrow \frac{(\mathcal{R}_{2i})-(\mathcal{R}_{2i-1})}{\varepsilon}, \tag{16}$$

for all $i = 1, ..., n$, where $\mathcal{R}'_1, ..., \mathcal{R}'_n$ are the lines of the resulting linear system. We obtain that $\Phi^{\mathrm{II}}_{n,\varepsilon} = \left\langle T^{-1}\tilde{\eta}_\varepsilon, \tau^{\mathrm{II}}_\varepsilon \right\rangle$, with:

$$T = \begin{pmatrix} f(x_1, y_1) & f(x_1{+}\varepsilon, y_1) & \cdots & f(x_n{-}\varepsilon, y_1) & f(x_n, y_1) \\ \delta^{y,+}_\varepsilon f(x_1, y_1) & \delta^{y,+}_\varepsilon f(x_1{+}\varepsilon, y_1) & \cdots & \delta^{y,+}_\varepsilon f(x_n{-}\varepsilon, y_1) & \delta^{y,+}_\varepsilon f(x_n, y_1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ f(x_1, y_n) & f(x_1{+}\varepsilon, y_n) & \cdots & f(x_n{-}\varepsilon, y_n) & f(x_n, y_n) \\ \delta^{y,-}_\varepsilon f(x_1, y_n) & \delta^{y,-}_\varepsilon f(x_1{+}\varepsilon, y_n) & \cdots & \delta^{y,-}_\varepsilon f(x_n{-}\varepsilon, y_n) & \delta^{y,-}_\varepsilon f(x_n, y_n) \end{pmatrix}.$$

Next, using that $\Phi^{\mathrm{II}}_{n,\varepsilon} = \left\langle \tilde{\eta}_\varepsilon, T^{-t}\tau^{\mathrm{II}}_\varepsilon \right\rangle$, we apply the linear combinations (16) to the linear system formed by the matrix T^t and the right-hand side $\tau^{\mathrm{II}}_\varepsilon$. These combinations are then operated on the columns of T and we obtain $\Phi^{\mathrm{II}}_{n,\varepsilon} = \left\langle \tilde{\eta}_\varepsilon, \tilde{M}^{-t}_\varepsilon \tilde{\tau}_\varepsilon \right\rangle$. Formula (14) follows. Formula (15) then directly follows from (16) and the properties of the determinant. □

To emphasise the similarities between the function Ψ_n and the expression of $\Phi^{\mathrm{II}}_{n,\varepsilon}$ given by Lemma 8, similarly to the above proof, we apply some permutations on the lines and the columns of the matrix and the vectors involved in (14). The resulting expression, presented in Lemma 9, is a finite difference counterpart of the expression of Ψ_n. Therefore, we introduce the finite difference version of the matrix M, denoted using the upper index δ. Let $\mathrm{S}^{\delta x}_\varepsilon, \mathrm{S}^{\delta y}_\varepsilon, \mathrm{S}^{\delta xy}_\varepsilon$ be the matrices of size $n \times n$ given by

$$(\mathrm{S}^{\delta x}_\varepsilon)_{ij} = \begin{cases} \delta^{x,+}_\varepsilon f(x_j, y_i), & \text{if } i < n, \\ \delta^{x,-}_\varepsilon f(x_j, y_i), & \text{else,} \end{cases}$$

$$(\mathrm{S}^{\delta y}_\varepsilon)_{ij} = \begin{cases} \delta^{y,+}_\varepsilon f(x_j, y_i), & \text{if } j < n, \\ \delta^{y,-}_\varepsilon f(x_j, y_i), & \text{else,} \end{cases}$$

$$(\mathrm{S}^{\delta xy}_\varepsilon)_{ij} = \begin{cases} \delta^{x,+}_\varepsilon \delta^{y,+}_\varepsilon f(x_j, y_i), & \text{if } j < n \text{ and } i < n, \\ \delta^{x,+}_\varepsilon \delta^{y,-}_\varepsilon f(x_j, y_i), & \text{if } j < n \text{ and } i = n, \\ \delta^{x,-}_\varepsilon \delta^{y,+}_\varepsilon f(x_j, y_i), & \text{if } j = n \text{ and } i < n, \\ \delta^{x,-}_\varepsilon \delta^{y,-}_\varepsilon f(x_j, y_i), & \text{else,} \end{cases}$$

for all $i, j = 1, ..., n$. We then denote by M_ε the matrix of size $2n \times 2n$ given by

$$M_\varepsilon = \begin{pmatrix} \mathrm{S} & \mathrm{S}^{\delta x}_\varepsilon \\ \mathrm{S}^{\delta y}_\varepsilon & \mathrm{S}^{\delta xy}_\varepsilon \end{pmatrix},$$

where S is the sampling of f at the points $x_1, ..., x_n$ and $y_1, ..., y_n$ as defined in (2).

Lemma 9 (Finite difference interpolation)**.** *Let $\varepsilon > 0$ satisfy* (4) *and suppose* $\det \mathrm{S}^{\mathrm{II}}_\varepsilon \neq 0$. *The output $\Phi^{\mathrm{II}}_{n,\varepsilon}$ of the CA algorithm using the double grid* (5) *then satisfies*

$$\Phi^{\mathrm{II}}_{n,\varepsilon}(x, y) = \left\langle M^{-1}_\varepsilon \begin{pmatrix} \eta(x) \\ \mathrm{H}_\varepsilon(x) \end{pmatrix}, \begin{pmatrix} \tau(y) \\ \mathrm{T}_\varepsilon(y) \end{pmatrix} \right\rangle,$$

for all $(x, y) \in \Omega$*, with:*

$$\mathrm{H}_\varepsilon(x) = \begin{pmatrix} \delta_\varepsilon^{y,+} f(x, y_1) \\ \vdots \\ \delta_\varepsilon^{y,-} f(x, y_n) \end{pmatrix}, \qquad \mathrm{T}_\varepsilon(y) = \begin{pmatrix} \delta_\varepsilon^{x,+} f(x_1, y) \\ \vdots \\ \delta_\varepsilon^{x,-} f(x_n, y) \end{pmatrix}.$$

Moreover, we have

$$|\det \mathrm{S}_\varepsilon^{\mathrm{II}}| = \varepsilon^{2n} |\det M_\varepsilon|. \tag{17}$$

Proof. The result is obtained in a similar manner as Lemma 8, using some permutations instead of the linear combinations (16). □

We can now conclude the proof of Theorem 4.

Proof of Theorem 4. First, since $f \in C^{1,1}(\Omega)$, the finite differences $\delta_\varepsilon^{x,\pm} f$, $\delta_\varepsilon^{y,\pm} f$, and $\delta_\varepsilon^{x,\pm} \delta_\varepsilon^{y,\pm} f$, converge uniformly to $\partial_x f$, $\partial_y f$, and $\partial_{xy} f$, respectively. Hence, we obtain that:

$$M_\varepsilon \xrightarrow[\varepsilon \to 0]{} M, \tag{18a}$$

$$\mathrm{H}_\varepsilon(x) \xrightarrow[\varepsilon \to 0]{} \mathrm{H}(x), \text{ for all } x \in \Omega_x, \tag{18b}$$

$$\mathrm{T}_\varepsilon(y) \xrightarrow[\varepsilon \to 0]{} \mathrm{T}(y), \text{ for all } y \in \Omega_y. \tag{18c}$$

Gathering $\det M \neq 0$, (18a), and the continuity of the determinant, we obtain that $|\det M_\varepsilon| \neq 0$, for all $\varepsilon > 0$ small enough. Using moreover (17), we infer that $|\det \mathrm{S}_\varepsilon^{\mathrm{II}}| \neq 0$ for all $\varepsilon > 0$ small enough, so that the CA algorithm can be applied with the double grid.

Next, since M is invertible, we have

$$M_\varepsilon^{-1} \big(\eta(x), \mathrm{H}_\varepsilon(x) \big)^t \xrightarrow[\varepsilon \to 0]{} M^{-1} \big(\eta(x), \mathrm{H}(x) \big)^t,$$

for all $x \in \Omega_x$. Finally, Lemma 9 yields

$$\lim_{\varepsilon \to 0} \Phi_{n,\varepsilon}^{\mathrm{II}}(x, y) = \left\langle M^{-1} \begin{pmatrix} \eta(x) \\ \mathrm{H}(x) \end{pmatrix}, \begin{pmatrix} \tau(y) \\ \mathrm{T}(y) \end{pmatrix} \right\rangle = \Psi_n(x, y),$$

for all $(x, y) \in \Omega$. □

4. Cross approximation approach

In this section we present iterative methods with the aim of implementing efficiently the solution of LRHI. These methods moreover present the advantages of allowing an adaptive choice of the points $x_1, ..., x_n$ and $y_1, ..., y_n$ used in Hermite interpolation, in a similar manner as the ACA algorithm [2]. The choice of the points is nevertheless left for further investigations.

Let us first note that, in addition to the cross interpolation property (1), the operators $\mathrm{I}_1, ..., \mathrm{I}_n$ satisfy:

$$\partial_x \mathrm{I}_i[f](\cdot, y_i) = \partial_x f(\cdot, y_i) \quad \text{and} \quad \partial_y \mathrm{I}_i[f](x_i, \cdot) = \partial_y f(x_i, \cdot),$$

for all $i = 1, ..., n$. Then, we introduce the operators $\mathrm{I}_1^{\mathrm{X}}, ..., \mathrm{I}_n^{\mathrm{X}}$ defined by

$$\mathrm{I}_i^{\mathrm{X}}[f] = \frac{\partial_y f(\cdot, y_i)\partial_x f(x_i, \cdot)}{\partial_{xy} f(x_i, y_i)},$$

for all $i = 1, ..., n$ such that $\partial_{xy} f(x_i, y_i) \neq 0$. One can easily check that the function $\mathrm{I}_i^{\mathrm{X}}[f]$ interpolates the derivatives of f in the opposite direction on the x-line L_i and the y-line C_i intersecting at (x_i, y_i), i.e.,

$$\partial_y \mathrm{I}_i^{\mathrm{X}}[f](\cdot, y_i) = \partial_y f(\cdot, y_i) \quad \text{and} \quad \partial_x \mathrm{I}_i^{\mathrm{X}}[f](x_i, \cdot) = \partial_x f(x_i, \cdot), \tag{19}$$

for all $i = 1, ..., n$. The algorithms introduced later in this section are constructed in a same fashion as the CA algorithm, using the operators $\mathrm{I}_1^{\mathrm{X}}, ..., \mathrm{I}_n^{\mathrm{X}}$ and $\mathrm{I}_1, ..., \mathrm{I}_n$. Hence, before presenting them, we check that no fill-in occurs during the different operations.

Proposition 10 (No fill-in)**.** *Let $\ell \in \{2, ..., n\}$ and $i \in \{1, ..., \ell-1\}$, and suppose $\nabla f(x_i, \cdot) = \nabla f(\cdot, y_i) = 0$. Then,*

- *supposing that $\partial_{xy} f(x_\ell, y_\ell) \neq 0$, we have the following:*
  1. $\mathrm{I}_\ell^{\mathrm{X}}[f](x_i, \cdot) = \mathrm{I}_\ell^{\mathrm{X}}[f](\cdot, y_i) = 0$ *(no fill-in in values by $\mathrm{I}_\ell^{\mathrm{X}}$);*
  2. $\partial_x \mathrm{I}_\ell^{\mathrm{X}}[f](x_i, \cdot) = \partial_y \mathrm{I}_\ell^{\mathrm{X}}[f](\cdot, y_i) = 0$ *(no fill-in in derivatives by $\mathrm{I}_\ell^{\mathrm{X}}$);*
- *supposing that $\partial_x f(x_\ell, \cdot) = \partial_y f(\cdot, y_\ell) = 0$ and $f(x_\ell, y_\ell) \neq 0$, we have that*
  3. $\partial_x \mathrm{I}_\ell[f](x_i, \cdot) = \partial_y \mathrm{I}_\ell[f](\cdot, y_i) = 0$ *(no fill-in in derivatives by I_ℓ).*
  4. $\partial_x \mathrm{I}_\ell[f](x_\ell, \cdot) = \partial_y \mathrm{I}_\ell[f](\cdot, y_\ell) = 0$ *(no fill-in in derivatives by I_ℓ).*

Proof. In each of the four cases, we prove one of the two equalities. The proof that the second term is also null is done in a similar manner.

1. We deduce from $\partial_y f(x_i, \cdot) = 0$ that
$$\mathrm{I}_\ell^{\mathrm{X}}[f](x_i, \cdot) = \frac{\partial_y f(x_i, y_\ell)\partial_x f(x_\ell, \cdot)}{\partial_{xy} f(x_\ell, y_\ell)} = 0.$$
2. Since $\partial_x f(x_i, \cdot) = 0$, we have $\partial_{xy} f(x_i, \cdot) = 0$. We conclude that
$$\partial_x \mathrm{I}_\ell^{\mathrm{X}}[f](x_i, \cdot) = \frac{\partial_{xy} f(x_i, y_\ell)\partial_x f(x_\ell, \cdot)}{\partial_{xy} f(x_\ell, y_\ell)} = 0.$$
3. We infer from $\partial_x f(x_i, \cdot) = 0$ that
$$\partial_x \mathrm{I}_\ell[f](x_i, \cdot) = \frac{\partial_x f(x_i, y_\ell) f(x_\ell, \cdot)}{f(x_\ell, y_\ell)} = 0.$$

4. Using $\partial_x f(x_\ell, \cdot) = 0$, we obtain:

$$\partial_x \mathrm{I}_\ell[f](x_\ell, \cdot) = \frac{\partial_x f(x_\ell, y_\ell) f(x_\ell, \cdot)}{f(x_\ell, y_\ell)} = 0.$$

□

We present below the first implementation of LRHI: the first *Cross Hermite Interpolation* (CHI1) algorithm.

Input : $f : \Omega \to \mathbb{R}$, $x_1, ..., x_n \in \Omega_x$, $y_1, ..., y_n \in \Omega_y$.
Output : rank $2n$ function $\Psi_n : \Omega \to \mathbb{R}$.
$\Psi_0 \leftarrow 0$;
$\mathrm{R}_0 \leftarrow f$;
for $i \leftarrow 1$ **to** n **do**
  if $\partial_{xy}\mathrm{R}_{i-1}(x_i, y_i) \neq 0$ **then**
    $\mathrm{I}_i^{\mathrm{X}} \leftarrow \frac{\partial_y \mathrm{R}_{i-1}(\cdot, y_i)\, \partial_x \mathrm{R}_{i-1}(x_i, \cdot)}{\partial_{xy}\mathrm{R}_{i-1}(x_i, y_i)}$;
    $\Psi_i \leftarrow \Psi_{i-1} + \mathrm{I}_i^{\mathrm{X}}$;
    $\mathrm{R}_i \leftarrow \mathrm{R}_{i-1} - \mathrm{I}_i^{\mathrm{X}}$;
    if $\mathrm{R}_i(x_i, y_i) \neq 0$ **then**
      $\mathrm{I}_i \leftarrow \frac{\mathrm{R}_i(\cdot, y_i)\, \mathrm{R}_i(x_i, \cdot)}{\mathrm{R}_i(x_i, y_i)}$;
      $\Psi_i \leftarrow \Psi_i + \mathrm{I}_i$;
      $\mathrm{R}_i \leftarrow \mathrm{R}_i - \mathrm{I}_i$;
    else
      Error during Cross Hermite Interpolation → stop;
    end
  else
    Error during Cross Hermite Interpolation → stop;
  end
end

Algorithm CHI1: Cross Hermite Interpolation

Remark (Notation of the output)**.** We use the same notation, namely Ψ_n, for the function constructed in the CHI1 algorithm and the unique solution of LRHI. We indeed show in Proposition 12 that these functions are the same.

We leave the discussion about the cases where the algorithm can be applied for later and we now focus on the properties of the output.

Proposition 11 (Hermite interpolation)**.** *Supposing that the CHI1 algorithm processed until the end, the output Ψ_n satisfies Hermite interpolation* (8) *for all $i = 1, ..., n$.*

Proof. We prove this result by induction on $n \geq 0$. First, the proposition is trivially satisfied for $n = 0$ since no equation has to be satisfied. Then, we suppose that the proposition is valid for $n \geq 0$, i.e.,

$$\mathrm{R}_n(x_i, \cdot) = \mathrm{R}_n(\cdot, y_i) = 0, \quad \nabla \mathrm{R}_n(x_i, \cdot) = \nabla \mathrm{R}_n(\cdot, y_i) = 0, \tag{20}$$

for all $i = 1, ..., n$. We denote by $\tilde{\Psi}_{n+1} : \Omega \to \mathbb{R}$ the function obtained after adding the interpolation of the derivatives: $\tilde{\Psi}_{n+1} = \Psi_n + \mathrm{I}^{\mathrm{X}}_{n+1}[\mathrm{R}_n]$. We infer from (20) and Points 1 and 2 of Proposition 10 that

$$\begin{aligned} \mathrm{I}^{\mathrm{X}}_{n+1}[\mathrm{R}_n](x_i, \cdot) = &\quad \mathrm{I}^{\mathrm{X}}_{n+1}[\mathrm{R}_n](\cdot, y_i) = 0, \\ \nabla \mathrm{I}^{\mathrm{X}}_{n+1}[\mathrm{R}_n](x_i, \cdot) = &\nabla \mathrm{I}^{\mathrm{X}}_{n+1}[\mathrm{R}_n](\cdot, y_i) = 0, \end{aligned} \tag{21}$$

for all $i = 1, ..., n$. Moreover, the cross interpolation property of the operator $\mathrm{I}^{\mathrm{X}}_{n+1}$ (Equation (19)) yields the following:

$$\begin{aligned} \partial_x \mathrm{I}^{\mathrm{X}}_{n+1}[\mathrm{R}_n](x_{n+1}, \cdot) = \partial_x \mathrm{R}_n(x_{n+1}, \cdot) = \partial_x f(x_{n+1}, \cdot) - \partial_x \Psi_n(x_{n+1}, \cdot), \\ \partial_y \mathrm{I}^{\mathrm{X}}_{n+1}[\mathrm{R}_n](\cdot, y_{n+1}) = \partial_y \mathrm{R}_n(\cdot, y_{n+1}) = \partial_y f(\cdot, y_{n+1}) - \partial_y \Psi_n(\cdot, y_{n+1}). \end{aligned}$$

We infer that

$$\partial_x \tilde{\Psi}_{n+1}(x_{n+1}, \cdot) = \partial_x f(x_{n+1}, \cdot) \quad \text{and} \quad \partial_y \tilde{\Psi}_{n+1}(\cdot, y_{n+1}) = \partial_y f(\cdot, y_{n+1}). \tag{22}$$

We gather the induction hypothesis for n, and (21) and (22), to conclude that $\tilde{\Psi}_{n+1}$ satisfies the interpolation properties (8), for all $i = 1, ..., n$, and (8b) for $i = n+1$. Next, we denote by $\tilde{\mathrm{R}}_{n+1} : \Omega \to \mathbb{R}$ the following remainder:

$$\tilde{\mathrm{R}}_{n+1} = \mathrm{R}_n - \mathrm{I}^{\mathrm{X}}_{n+1}[\mathrm{R}_n] = f - \tilde{\Psi}_{n+1}.$$

The aforementioned interpolation properties satisfied by $\tilde{\Psi}_{n+1}$ can be rewritten as follows: for all $i = 1, ..., n$,

$$\tilde{\mathrm{R}}_{n+1}(x_i, \cdot) = \quad \tilde{\mathrm{R}}_{n+1}(\cdot, y_i) \quad = 0, \tag{23a}$$

$$\nabla \tilde{\mathrm{R}}_{n+1}(x_i, \cdot) = \nabla \tilde{\mathrm{R}}_{n+1}(\cdot, y_i) \quad = 0, \tag{23b}$$

$$\partial_x \tilde{\mathrm{R}}_{n+1}(x_{n+1}, \cdot) = \partial_y \tilde{\mathrm{R}}_{n+1}(\cdot, y_{n+1}) = 0. \tag{23c}$$

Using moreover Point 3 of Proposition 10 and Property 1, we obtain

$$\begin{aligned} \partial_x \mathrm{I}_{n+1}[\tilde{\mathrm{R}}_{n+1}](x_i, \cdot) = \partial_y \mathrm{I}_{n+1}[\tilde{\mathrm{R}}_{n+1}](\cdot, y_i) = 0, \\ \mathrm{I}_{n+1}[\tilde{\mathrm{R}}_{n+1}](x_i, \cdot) = \mathrm{I}_{n+1}[\tilde{\mathrm{R}}_{n+1}](\cdot, y_i) = 0, \end{aligned} \tag{24}$$

for all $i = 1, ..., n$. Furthermore, we infer from (23c) and Point 4 of Proposition 10 that

$$\partial_x \mathrm{I}_{n+1}[\tilde{\mathrm{R}}_{n+1}](x_{n+1}, \cdot) = \partial_y \mathrm{I}_{n+1}[\tilde{\mathrm{R}}_{n+1}](\cdot, y_{n+1}) = 0. \tag{25}$$

We conclude from (24) and (25) that $\Psi_{n+1} = \tilde{\Psi}_{n+1} + \mathrm{I}_{n+1}[\tilde{\mathrm{R}}_{n+1}]$ satisfies the same interpolation properties as $\tilde{\Psi}_{n+1}$, i.e., (8) for all $i = 1, ..., n$ and (8b) for $i = n+1$. Moreover, we deduce from the cross interpolation property of I_{n+1}, namely (1), that

$$\begin{aligned}\mathrm{I}_{n+1}[\tilde{\mathrm{R}}_{n+1}](x_{n+1}, \cdot) = \tilde{\mathrm{R}}_{n+1}(x_{n+1}, \cdot) = f(x_{n+1}, \cdot) - \tilde{\Psi}_{n+1}(x_{n+1}, \cdot),\\ \mathrm{I}_{n+1}[\tilde{\mathrm{R}}_{n+1}](\cdot, y_{n+1}) = \tilde{\mathrm{R}}_{n+1}(\cdot, y_{n+1}) = f(\cdot, y_{n+1}) - \tilde{\Psi}_{n+1}(\cdot, y_{n+1}).\end{aligned}$$

We infer that Ψ_{n+1} satisfies

$$\Psi_{n+1}(x_{n+1}, \cdot) = f(x_{n+1}, \cdot), \quad \text{and} \quad \Psi_{n+1}(\cdot, y_{n+1}) = f(\cdot, y_{n+1}).$$

Hence, Ψ_{n+1} also satisfies (8a) for $i = n+1$ and the result follows. □

Using this proposition, we now prove that, whenever it exists, the output is the unique solution of LRHI.

Proposition 12 (Closed-form expression for the CHI1 algorithm)**.** *Supposing that the CHI1 algorithm processed until the end, the output is the function Ψ_n defined in* (11)*.*

Proof. The proof is done in a similar manner as the proof of Proposition 6. First, we note that Ψ_n only depends linearly on η, τ, H, and T. As a result, there exists a matrix B such that

$$\Psi_n(x, y) = \left\langle B \begin{pmatrix} \eta(x) \\ \mathrm{H}(x) \end{pmatrix}, \begin{pmatrix} \tau(y) \\ \mathrm{T}(y) \end{pmatrix} \right\rangle,$$

for all $(x, y) \in \Omega$. Next, let $\lambda, \tilde{\lambda} \in \left[C^1(\Omega_x)\right]^n$ be the vector fields defined by $\big(\lambda(x), \tilde{\lambda}(x)\big)^t = B\big(\eta(x), \mathrm{H}(x)\big)^t$, for all $x \in \Omega_x$. The function Ψ_n then satisfies Hermite interpolation (8) and $\Psi_n = \langle \lambda, \tau \rangle + \langle \tilde{\lambda}, \mathrm{T} \rangle$, and the result follows from Proposition 7. □

We though underline that invertibility of the matrix M is not a sufficient condition for the CHI1 algorithm to succeed. We present below an example illustrating this remark.

Example. Consider $n = 2$, the points $x_1 = y_1 = 0$ and $x_2 = y_2 = 1$, and the rank four function $f : [0, 1]^2 \to \mathbb{R}$ given by

$$f(x, y) = \alpha(x) + \alpha(y) + x\alpha(x)\beta(y) + y\alpha(y)\beta(x),$$

for all $x, y \in [0, 1]$, with $\alpha, \beta : [0, 1] \to \mathbb{R}$ defined as follows:

$$\alpha(x) = x(x - 1), \quad \beta(x) = x^2(-3 + 2x),$$

for all $x \in [0, 1]$. Then, $f(x_j, y_i) = \partial_{xy} f(x_j, y_i) = 0$, for all $i, j = 1, 2$. Hence, since f and its cross-derivative cancel on all the vertices of the chosen grid, one can neither apply CA nor the CHI1 algorithm. Though, we have

$$\begin{aligned}\partial_x f(0, 0) = -1, \quad \partial_x f(0, 1) = -1, \quad \partial_x f(1, 0) = 1, \quad \partial_x f(1, 1) = 0,\\ \partial_y f(0, 0) = -1, \quad \partial_y f(0, 1) = 1, \quad \partial_y f(1, 0) = -1, \quad \partial_y f(1, 1) = 0.\end{aligned}$$

And we obtain

$$M = \begin{pmatrix} \mathbf{0} & \mathrm{S}^{\partial x} \\ \mathrm{S}^{\partial y} & \mathbf{0} \end{pmatrix}, \text{ with } \mathrm{S}^{\partial x} = \begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix} \text{ and } \mathrm{S}^{\partial y} = \begin{pmatrix} -1 & -1 \\ 1 & 0 \end{pmatrix}.$$

Hence, the matrix M is invertible and Formula (11) yields:

$$\begin{aligned} \Psi_2(x,y) = \ & \mathrm{H}_2(x)\,\tau_1(y) - \big(\mathrm{H}_1(x) + \mathrm{H}_2(x)\big)\tau_2(y) \\ & - \eta_2(x)\,\mathrm{T}_1(y) + \big(\eta_1(x) - \eta_2(x)\big)\,\mathrm{T}_2(y), \end{aligned}$$

or equivalently, $\Psi_2 = f$.

In order to provide an algorithm working under a predefined condition, we now switch the order in which the operators $\mathrm{I}_1, ..., \mathrm{I}_n$ and $\mathrm{I}_1^{\mathrm{X}}, ..., \mathrm{I}_n^{\mathrm{X}}$ are applied. In what follows, the derivatives are first interpolated and the CA algorithm is then applied to the remainder in order to interpolate the values. To interpolate the derivatives, we use the *Cross Derivative Interpolation* (CDI) algorithm presented below. Before giving the conditions ensuring that

Input : $f : \Omega \to \mathbb{R}$, $x_1, ..., x_n \in \Omega_x$, $y_1, ..., y_n \in \Omega_y$.
Output : rank n function $\Psi_n^{\mathrm{X}} : \Omega \to \mathbb{R}$.
$\Psi_0^{\mathrm{X}} \leftarrow 0$;
$\mathrm{R}_0^{\mathrm{X}} \leftarrow f$;
for $i \leftarrow 1$ **to** n **do**
  if $\partial_{xy}\mathrm{R}_{i-1}^{\mathrm{X}}(x_i, y_i) \neq 0$ **then**
    $\mathrm{I}_i^{\mathrm{X}} \leftarrow \frac{\partial_y \mathrm{R}_{i-1}^{\mathrm{X}}(\cdot, y_i)\,\partial_x \mathrm{R}_{i-1}^{\mathrm{X}}(x_i, \cdot)}{\partial_{xy}\mathrm{R}_{i-1}^{\mathrm{X}}(x_i, y_i)}$;
    $\Psi_i^{\mathrm{X}} \leftarrow \Psi_{i-1}^{\mathrm{X}} + \mathrm{I}_i^{\mathrm{X}}$;
    $\mathrm{R}_i^{\mathrm{X}} \leftarrow \mathrm{R}_{i-1}^{\mathrm{X}} - \mathrm{I}_i^{\mathrm{X}}$;
  else
    Error during Cross Derivative Interpolation → stop;
  end
end

Algorithm CDI: Cross Derivative Interpolation

the CDI algorithm will process until the end, let us show that the derivatives of f in the opposite directions are, as expected, interpolated.

Proposition 13 (Derivatives interpolation)**.** *Supposing that the CDI algorithm processed until the end, the output Ψ_n^{X} of the CDI algorithm satisfies*

$$\partial_y \Psi_n^{\mathrm{X}}(\cdot, y_i) = \mathrm{H}_i, \qquad \partial_x \Psi_n^{\mathrm{X}}(x_i, \cdot) = \mathrm{T}_i, \tag{26}$$

for all $i = 1, ..., n$.

Proof. This result is obtained by induction, using the interpolation property (19) of the operators $\mathrm{I}_1^{\mathrm{X}}, ..., \mathrm{I}_n^{\mathrm{X}}$, and Point 2 of Proposition 10, similarly to the proof of Proposition 11. □

The CDI algorithm can be applied, up to a permutation of the chosen points, under the condition $\det \mathrm{S}^{\partial xy} \neq 0$. This follows from the next lemma adapting [2, Lemma 2] to the CDI algorithm.

Lemma 14 (Formula for the remainder)**.** *The remainders* $\mathrm{R}_1^{\mathrm{X}}, ..., \mathrm{R}_n^{\mathrm{X}}$ *introduced in the CDI algorithm satisfy the following:*

$$\prod_{j=1}^{n} \partial_{xy} \mathrm{R}_{j-1}^{\mathrm{X}}(x_j, y_j) = \det \mathrm{S}^{\partial xy}.$$

Proof. First, we note that, for all $j = 1, ..., n$, there exist $\alpha_1^j, ..., \alpha_j^j : \Omega_x \to \mathbb{R}$ such that

$$\mathrm{R}_j^{\mathrm{X}}(x, y) = f(x, y) - \sum_{k=1}^{j} \alpha_k^j(x) \partial_x f(x_k, y),$$

for all $(x, y) \in \Omega$. Therefore, we have

$$\partial_{xy} \mathrm{R}_j^{\mathrm{X}}(x, y) = \partial_{xy} f(x, y) - \sum_{k=1}^{j} \alpha_k^{j\,\prime}(x) \partial_{xy} f(x_k, y),$$

for all $j = 1, ..., n$ and $(x, y) \in \Omega$. Then,

$$\begin{aligned} \partial_{xy} \mathrm{R}_{j-1}^{\mathrm{X}}(x_j, y_i) &= \partial_{xy} f(x_j, y_i) - \sum_{k=1}^{j-1} \alpha_k^{j-1\,\prime}(x_j) \partial_{xy} f(x_k, y_i) \\ &= \mathrm{S}_{ij}^{\partial xy} - \sum_{k=1}^{j-1} \alpha_k^{j-1\,\prime}(x_j)\, \mathrm{S}_{ik}^{\partial xy}, \end{aligned}$$

for all $i, j = 1, ..., n$. Since the second term on the right hand side is a linear combination of the columns of $\mathrm{S}^{\partial xy}$, we obtain

$$\det \left[\left(\partial_{xy} \mathrm{R}_{j-1}^{\mathrm{X}}(x_j, y_i) \right)_{i,j=1,...,n} \right] = \det \mathrm{S}^{\partial xy}.$$

Moreover, for all $j = 2, ..., n$ and $i = 1, ..., j-1$, we deduce from

$$\partial_{xy} \Psi_{j-1}^{\mathrm{X}}(\cdot, y_i) = \partial_{xy} f(\cdot, y_i)$$

that $\partial_{xy} \mathrm{R}_{j-1}^{\mathrm{X}}(x_j, y_i) = 0$. Hence, the matrix $\left(\partial_{xy} \mathrm{R}_{j-1}^{\mathrm{X}}(x_j, y_i) \right)_{i,j=1,...,n}$ is lower triangular and

$$\det \mathrm{S}^{\partial xy} = \det \left[\left(\partial_{xy} \mathrm{R}_{j-1}^{\mathrm{X}}(x_j, y_i) \right)_{i,j=1,...,n} \right] = \prod_{j=1}^{n} \partial_{xy} \mathrm{R}_{j-1}^{\mathrm{X}}(x_j, y_j).$$

This concludes the proof. □

A consequence of Proposition 13 is the following closed-form expression for the output of the CDI algorithm.

Proposition 15 (Closed-form expression for the CDI algorithm)**.** *Supposing* $\det \mathrm{S}^{\partial xy} \neq 0$*, the output* Ψ_n^{X} *satisfies*

$$\Psi_n^{\mathrm{X}}(x,y) = \left\langle \left(\mathrm{S}^{\partial xy} \right)^{-1} \mathrm{H}(x), \mathrm{T}(y) \right\rangle, \tag{27}$$

for all $(x,y) \in \Omega$*.*

Proof. This result is proven in a very similar manner as Propositions 6 and 12. First, we note that Ψ_n^{X} only depends linearly on H and T. As a result, there exists a matrix B such that

$$\Psi_n^{\mathrm{X}}(x,y) = \langle B\,\mathrm{H}(x), \mathrm{T}(y) \rangle,$$

for all $(x,y) \in \Omega$. Next, omitting the variables x and y, we note that $\Psi_n^{\mathrm{X}} = \langle \tilde{\lambda}, \mathrm{T} \rangle$, with $\tilde{\lambda} = B\,\mathrm{H}$. We then deduce from Proposition 13 that

$$\partial_y \Psi_n^{\mathrm{X}}(\cdot, y_i) = \left\langle \tilde{\lambda}, \frac{d\,\mathrm{T}}{dy}(y_i) \right\rangle = \mathrm{H}_i,$$

for all $i = 1, ..., n$. Using the matrix notation, we obtain $\mathrm{S}^{\partial xy}\,\tilde{\lambda} = \mathrm{H}$, and we conclude that $B = \left(\mathrm{S}^{\partial xy} \right)^{-1}$. □

In order to apply the CA algorithm to the remainder $\mathrm{R}_n^{\mathrm{X}}$ of the CDI algorithm, we present in the following proposition an explicit formula for its associated sampling matrix $\bar{\mathrm{S}}$.

Proposition 16. *Suppose* $\det \mathrm{S}^{\partial xy} \neq 0$ *and denote by* $\bar{S}$ *the matrix containing the sampling of* $\mathrm{R}_n^{\mathrm{X}} = f - \Psi_n^{\mathrm{X}}$*, i.e.,*

$$\bar{S}_{ij} = f(x_j, y_i) - \Psi_n^{\mathrm{X}}(x_j, y_i),$$

for all $i, j \in 1, ..., n$*. Then, the matrix* $\bar{S}$ *is the Schur complement of the block* $\mathrm{S}^{\partial xy}$ *of the matrix* M*, that is,*

$$\bar{S} = S - \mathrm{S}^{\partial x} \left(\mathrm{S}^{\partial xy} \right)^{-1} \mathrm{S}^{\partial y}.$$

Proof. Using Proposition 15, we obtain

$$\begin{aligned}
\Psi_n^{\mathrm{X}}(x_j, y_i) &= \left\langle (\mathrm{S}^{\partial xy})^{-1}\,\mathrm{H}(x_j), \mathrm{T}(y_i) \right\rangle \\
&= \sum_{k=1}^{n} \sum_{l=1}^{n} \left((\mathrm{S}^{\partial xy})^{-1} \right)_{kl} \mathrm{H}_l(x_j)\,\mathrm{T}_k(y_i) \\
&= \sum_{k=1}^{n} \sum_{l=1}^{n} \left((\mathrm{S}^{\partial xy})^{-1} \right)_{kl} \partial_y f(x_j, y_l) \partial_x f(x_k, y_i) \\
&= \sum_{k=1}^{n} \sum_{l=1}^{n} \left((\mathrm{S}^{\partial xy})^{-1} \right)_{kl} \mathrm{S}_{lj}^{\partial y}\,\mathrm{S}_{ik}^{\partial x}.
\end{aligned}$$

Hence, we have

$$\bar{S}_{ij} = S_{ij} - \left(\mathrm{S}^{\partial x} (\mathrm{S}^{\partial xy})^{-1} \, \mathrm{S}^{\partial y} \right)_{ij},$$

and the result follows. □

Using the determinant formula for Schur complements [20], we obtain:

$$\det M = \det \bar{\mathrm{S}} \det \mathrm{S}^{\partial xy}.$$

As a result, under the conditions $\det M \neq 0$ and $\det \mathrm{S}^{\partial xy} \neq 0$, one can apply the CA algorithm to the remainder $\mathrm{R}_n^{\mathrm{X}}$, up to some permutation of the points $x_1, ..., x_n$ and $y_1, ..., y_n$. The combination of these two algorithms, denoted as the second *Cross Hermite Interpolation* (CHI2) algorithm in what follows, is presented below. In a similar manner as earlier

Input : $f : \Omega \to \mathbb{R}$, $x_1, ..., x_n \in \Omega_x$, $y_1, ..., y_n \in \Omega_y$.
Output : rank $2n$ function $\Psi_n : \Omega \to \mathbb{R}$.
$\Psi_n^{\mathrm{X}} \leftarrow \mathrm{CDI}(f)$;
$\mathrm{R}_n^{\mathrm{X}} \leftarrow f - \Psi_n^{\mathrm{X}}$;
$\Psi_n \leftarrow \Psi_n^{\mathrm{X}} + \mathrm{CA}(\mathrm{R}_n^{\mathrm{X}})$;

Algorithm CHI2: Cross Hermite Interpolation

for the CHI1 algorithm, we now prove the interpolation property and conclude that the output is the unique solution of LRHI.

Proposition 17 (Hermite interpolation)**.** *Suppose* $\det M \neq 0$ *and* $\det \mathrm{S}^{\partial xy} \neq 0$*. Then, the output* Ψ_n *of the CHI2 algorithm satisfies Hermite interpolation* (8) *for all* $i = 1, ..., n$*.*

Proof. The proof can be done recursively, similarly to the proof of Proposition 11. Points 3 and 4 of Proposition 10 indeed ensures that (26) remains satisfied by the output Ψ_n. On the other hand, the interpolation of the values of f, i.e., Equalities (3), is proven using the interpolation property (1) of the operators $\mathrm{I}_1, ..., \mathrm{I}_n$ and Property 1. □

Proposition 18 (Closed-form expression for the CHI2 algorithm)**.** *Suppose* $\det M \neq 0$ *and* $\det \mathrm{S}^{\partial xy} \neq 0$*. Then, the output of the CHI2 algorithm is the function* Ψ_n *defined in* (11)*.*

Proof. The result can be proven similarly to the proof of Proposition 12. □

5. Adaptively refined piecewise low-rank approximation

In order to satisfy a given error tolerance, one may increase the rank until the interpolant reaches the desired level accuracy [2, 5]. In the present paper we adopt a different approach, since our aim is to approximate the function f by a piecewise low-rank function. The main idea is to subdivide the domain Ω until the low-rank approximation on each subdomain is close enough to the target function f. We then validate numerically the C^1-regularity of the result whenever using LRHI for the local low-rank approximations.

Before this, let us show the efficiency of LRHI with an example. We apply LRHI and the CA algorithm to a polynomial of bidegree $(10, 10)$ with randomly chosen coefficients on $\Omega = [0, h]^2$, for multiple values of $h > 0$. We compare in Figure 1 the L^∞-error of these two algorithms for ranks 4 and 6. The comparison of the L^∞-error of the gradient in the same configuration with rank 4 is presented in Figure 2. We finally corroborate, using this random polynomial, the convergence announced in Theorem 4. To this extent, we apply the CA algorithm with the *double grid* (5) and we plot the L^∞-distance between the resulting function $\Phi^{\mathrm{II}}_{n,\varepsilon}$ and the output Ψ_n of LRHI for different values of ε in Figure 3.

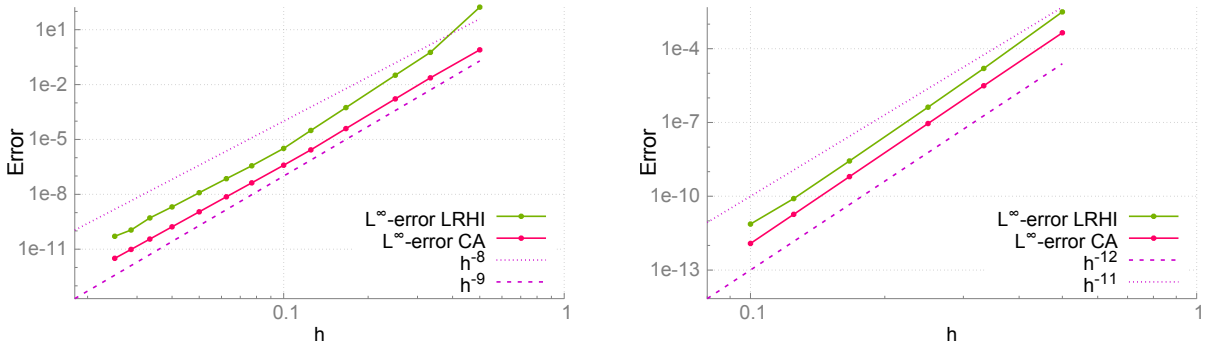

Figure 1: Comparison of the convergence with respect to the L^∞-norm of LRHI and of the CA algorithm with rank 4 (left) and rank 6 (right)

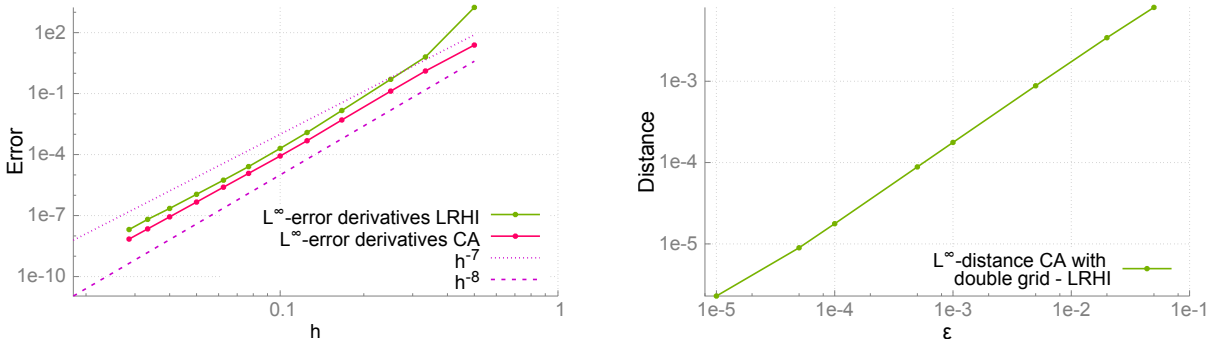

Figure 2: Comparison of the convergence of the gradient with respect to the L^∞-norm for LRHI and the CA algorithm with rank 4

Figure 3: Convergence of $\Phi^{\mathrm{II}}_{n,\varepsilon}$ to Ψ_n with respect to the L^∞-norm when ε goes to 0

To simplify the exposition of the piecewise low-rank approximation, we suppose in what follows that $\Omega_x = \Omega_y = [0, 1]$, and $\Omega = [0, 1]^2$. Also, we restrict ourselves to the case where the domain is split uniformly in two parts, both in the x-direction and in the y-direction. Each subdomain is then associated with two sequences of Booleans. Therefore, set $N \geq 1$ and denote by $\mathbb{B}_N$ the set of Boolean N-tuples. The two sequences $\mathbf{b}_{\mathrm{x}}, \mathbf{b}_{\mathrm{y}} \in \mathbb{B}_N$, with $\mathbf{b}_{\mathrm{x}} = (\mathrm{b}_{x,1}, ..., \mathrm{b}_{x,N})$ and $\mathbf{b}_{\mathrm{y}} = (\mathrm{b}_{y,1}, ..., \mathrm{b}_{y,N})$, encode the side of a given subdomain $\Omega_N^{\mathbf{b}_{\mathrm{x}},\mathbf{b}_{\mathrm{y}}}$ after each of the N splittings, i.e.,

$$\Omega_N^{\mathbf{b}_{\mathrm{x}},\mathbf{b}_{\mathrm{y}}} = \Omega_{x,N}^{\mathbf{b}_{\mathrm{x}}} \times \Omega_{y,N}^{\mathbf{b}_{\mathrm{y}}}, \text{ with } \Omega_{x,N}^{\mathbf{b}_{\mathrm{x}}} = [i_x, i_x + 2^{-N}], \quad \Omega_{y,N}^{\mathbf{b}_{\mathrm{y}}} = [i_y, i_y + 2^{-N}], \tag{28}$$

$i_x = \sum_{j=1}^{N} \mathrm{b}_{x,j} 2^{-j}$, and $i_y = \sum_{j=1}^{N} \mathrm{b}_{y,j} 2^{-j}$. For consistency of the notation, we also set $\Omega_{x,0} = \Omega_x$, $\Omega_{y,0} = \Omega_y$, and $\Omega_0 = \Omega$. These sets are then defined such that:

$$\Omega_{x,N}^{\mathbf{b}_\mathrm{x}} = \Omega_{x,N+1}^{\mathbf{b}_\mathrm{x}\times\{0\}} \cup \Omega_{x,N+1}^{\mathbf{b}_\mathrm{x}\times\{1\}}, \quad \text{and} \quad \Omega_{y,N}^{\mathbf{b}_\mathrm{y}} = \Omega_{y,N+1}^{\mathbf{b}_\mathrm{y}\times\{0\}} \cup \Omega_{y,N+1}^{\mathbf{b}_\mathrm{y}\times\{1\}}.$$

An illustration of the above notation can be found in Figure 4.

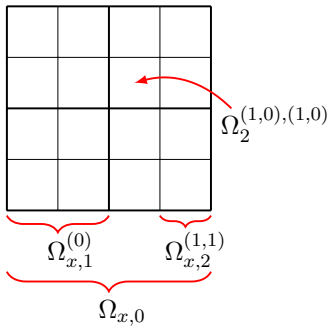

Figure 4: Illustration of the hierarchical subdomains

In order to stay generic, both for the choice between LRHI and the CA algorithm and for the definition of the stopping criterion, we suppose that we are given the following functions:

- lowRankApproximation$(f, [a,b]\times[c,d])$: applies either LRHI or the CA algorithm to $f\big|_{[a,b]\times[c,d]}$ with $x_1 = a$, $x_n = b$, $y_1 = c$, and $y_n = d$;
- stoppingCriterion$(f, \bar{\Omega}, \zeta)$: checks if the approximation of f on $\bar{\Omega}$ by the low-rank function $\zeta : \bar{\Omega} \to \mathbb{R}$ is good enough.

Though it did not occur during our numerical examples, we highlight that we might be in the particular case where the chosen low-rank approximation method can not be applied. This is checked a priori with the condition $\det \mathrm{S} \neq 0$ for the CA algorithm and $\det M \neq 0$ for the LRHI method. If this condition is not satisfied, we simply apply a Coons interpolation of the considered data. Then, the *Adaptively Refined Piecewise Low-Rank Approximation* (ARPLRA) can be outlined as follows:

An illustration of the recursive splitting used in ARPLRA is depicted in Figure 5. ARPLRA returns a list ζ of low-rank functions $\zeta_1 : \Gamma_1 \to \mathbb{R}, ..., \zeta_d : \Gamma_d \to \mathbb{R}$, where $\Gamma_1, ..., \Gamma_d \subset \Omega$ are domains of the form (28). With the purpose in mind to assemble these local functions, we first underline that the domains satisfy:

$$\Omega = \bigcup_{k=1}^{d} \Gamma_k, \text{ and } \Gamma_i \cap \Gamma_j = \partial\Gamma_i \cap \partial\Gamma_j, \tag{29}$$

for all $i, j = 1, ..., d$ such that $i \neq j$. Furthermore, using the interpolation properties of the CA algorithm and LRHI, we infer that the low-rank functions $\zeta_1, ..., \zeta_d$ satisfy

$$\zeta_i\big|_{\partial\Gamma_i} = f\big|_{\partial\Gamma_i}, \tag{30}$$

Input : $f : \Omega \to \mathbb{R}$: function to approximate,
N: number of subdivisions already operated,
$\mathbf{b}_{\mathrm{x}}, \mathbf{b}_{\mathrm{y}} \in \mathbb{B}_N$: position of the considered subdomain in the grid.
Output : ζ: list of low-rank functions.
locInterp$\leftarrow$ lowRankApproximation$(f, \Omega_N^{\mathbf{b}_{\mathrm{x}},\mathbf{b}_{\mathrm{y}}})$;
if stoppingCriterion$(f, \Omega_N^{\mathbf{b}_{\mathrm{x}},\mathbf{b}_{\mathrm{y}}}$, locInterp$)$ **then**
  $\zeta \leftarrow$ {locInterp};
else
  subd1 $\leftarrow$ ARPLRA$(f, N{+}1, \mathbf{b}_{\mathrm{x}}{\times}\{0\}, \mathbf{b}_{\mathrm{y}}{\times}\{0\})$;
  subd2 $\leftarrow$ ARPLRA$(f, N{+}1, \mathbf{b}_{\mathrm{x}}{\times}\{0\}, \mathbf{b}_{\mathrm{y}}{\times}\{1\})$;
  subd3 $\leftarrow$ ARPLRA$(f, N{+}1, \mathbf{b}_{\mathrm{x}}{\times}\{1\}, \mathbf{b}_{\mathrm{y}}{\times}\{0\})$;
  subd4 $\leftarrow$ ARPLRA$(f, N{+}1, \mathbf{b}_{\mathrm{x}}{\times}\{1\}, \mathbf{b}_{\mathrm{y}}{\times}\{1\})$;
  $\zeta \leftarrow$ subd1 $\cup$ subd2 $\cup$ subd3 $\cup$ subd4;
end

Algorithm ARPLRA: Adaptively refined piecewise low-rank approximation

for all $i = 1, ..., d$. Therefore, we conclude from (29) and (30) that the functions $\zeta_1, ..., \zeta_d$ can be assembled to form the continuous function $\mathrm{Z} : \Omega \to \mathbb{R}$ defined as follows:

$$\mathrm{Z}(x, y) = \zeta_i(x, y), \quad \text{where } i \in \{1, ..., d\} \text{ is such that } (x, y) \in \Gamma_i.$$

We finally analyse the efficiency of ARPLRA using a numerical example. To highlight the smoothness of the result whenever using LRHI, we consider in what follows a function with larger variations: a Fourier finite sum with $(7, 7)$ random coefficients. The used stopping criterion for a domain $\tilde{\Omega}$ and a low-rank approximation Ψ is: $\left\| f\big|_{\tilde{\Omega}} - \Psi \right\|_{L^\infty(\tilde{\Omega})} < 10^{-2}$. We compare the number of local approximations level by level using LRHI and the CA algorithm with rank 4 and 6 in the following array.

We remark that the number of splittings is lower when using the CA algorithm. Though, using LRHI ensures that the resulting function Z has C^1-regularity. To visualise this, we plot the output of ARPLRA using LRHI and the CA algorithm in Figure 7. The norm of the gradient of the output using these two methods can be found in Figure 8. One can observe in this figure jumps in the derivatives along the junctions when using the CA algorithm. On the other side, the gradient seems continuous when using LRHI.

To analyse more precisely the regularity of the output, since the functions $\zeta_1, ..., \zeta_d$ are smooth, let us define the gradient jump of Z by the following quantity:

$$\max_{\substack{i,j \in \{1,...,d\} \\ \partial\Gamma_i \cap \partial\Gamma_j \neq \varnothing}} \| \nabla\zeta_i - \nabla\zeta_j \|_{L^\infty(\partial\Gamma_i \cap \partial\Gamma_j)}.$$

We then compare in the following table the gradient jump of Z for the three different methods using rank 6 and tolerances 10^{-2} and 10^{-6} for the stopping criterion.

We observe in this table the smoothness of Z, even for a tolerance 10^{-2}, when using LRHI.

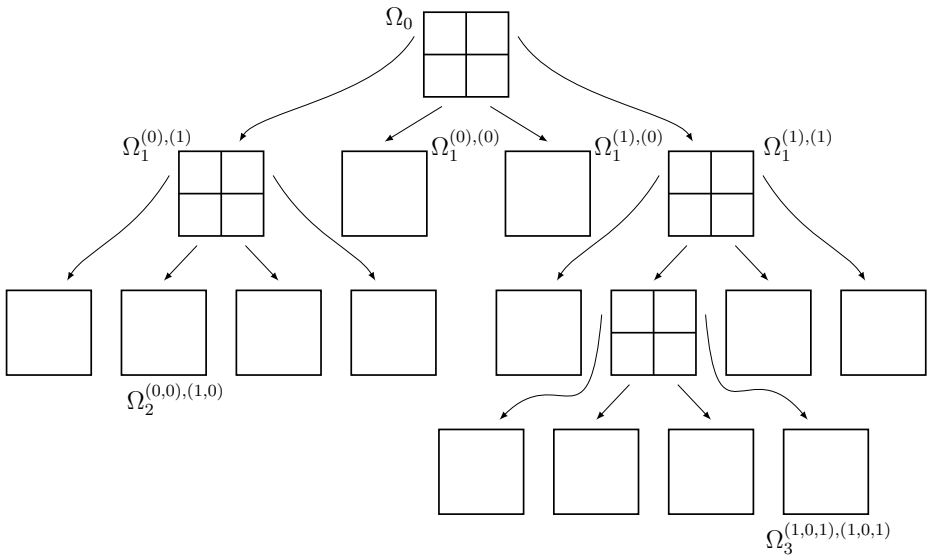

Figure 5: Illustration of the recursive splitting

algorithm	rank	level N=1	level N=2	level N=3	level N=4	level N=5
CA	4	0	1	49	44	0
	6	0	16	0	0	0
LRHI	4	0	0	36	108	16
	6	0	16	0	0	0
CA double grid $\varepsilon = 10^{-1}$	4	0	1	36	93	12
	6	0	16	0	0	0
CA double grid $\varepsilon = 10^{-3}$	4	0	0	36	108	16
	6	0	16	0	0	0

Figure 6: Comparison of the number of local low-rank approximations required for the different approximation algorithms with ranks 4 and 6 and tolerance 10^{-2}

6. Conclusion

We have considered the approximation of a given function by a C^1-regular piecewise low-rank function. The algorithm used for the local approximations is based on the CA algorithm. The main bottleneck is then to obtain C^1-junctions between the local approximations. In order to tackle this problem, we have extended a closed-form formula, which characterises the output of the CA algorithm, to obtain a method providing a Hermite interpolation property. We have then shown that this method is a limiting case of the CA algorithm. A direct consequence of this result is that the full approximation order satisfied by the CA algorithm [18] is also valid for our new method.

We have then proposed an extension of the CA algorithm to implement efficiently the new approximation method. Next, we have presented a second algorithm which is ensured to perform the approximation under some predefined condition. We have then presented a

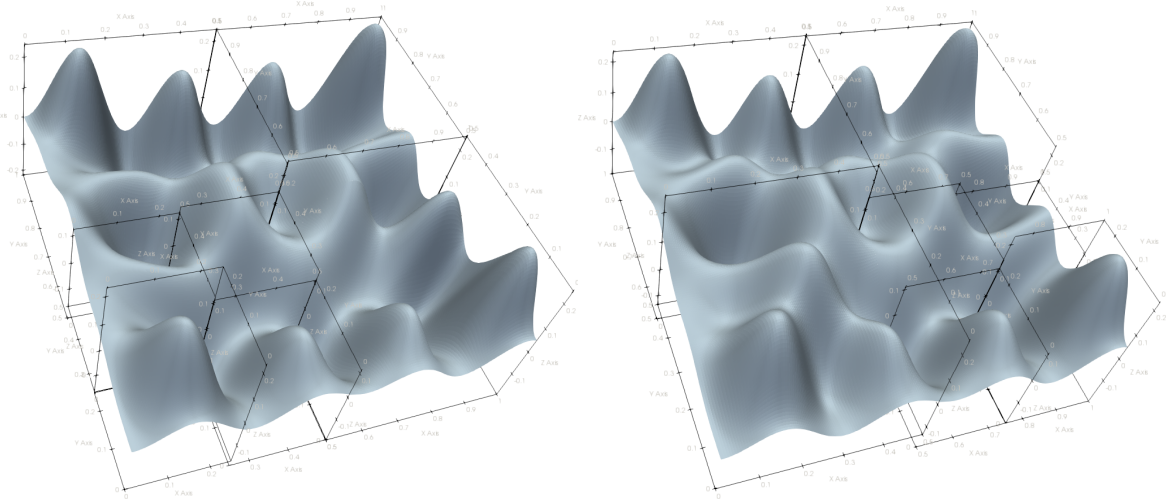

Figure 7: Piecewise low-rank output of ARPLRA using LRHI and the CA algorithm

piecewise low-rank approximation with adaptive refinement using either the CA algorithm or our new method to perform the local approximations. We have finally compared the results obtained using the two methods, confirming that our method ensures C^1-smooth junctions.

For future works, one should investigate if this method can be extended to higher dimensions. This could be done using the extension of the CA algorithm to higher dimensions studied in [21]. Among the challenging problems associated with the extension to the multivariate setting, we mention the fact that the tensor-rank of higher order tensors is much harder to characterize than the matrix rank (in fact, its computation is known to be a NP-hard problem). Also, the result of the CA algorithm is no-longer independent of the order of the successive cross-interpolation steps.

In addition, a characterization of low-rank functions would be very useful. We feel that the error bounds (Theorems 3 and 5 will be useful for that, leading us to conjecture that rank n functions are characterized by $\det\left(\partial_x^i \partial_y^j f(\hat{x}, \hat{y})\right)_{i,j=0,\dots,n} = 0$ on Ω. At this point, however, we are not able to provide further theoretical insights.

Acknowledgments

This work was supported by the Austrian Science Fund (FWF) through project NFN S117 Geometry+Simulation, and by the European Research Council (ERC) through project GA 694515 "Change"

References

References

[1] T. M. Rassias, J. Simsa, Finite sums decompositions in mathematical analysis, Chichester ; New York : Wiley, 1995, includes bibliographical references (p. [161]-163) and

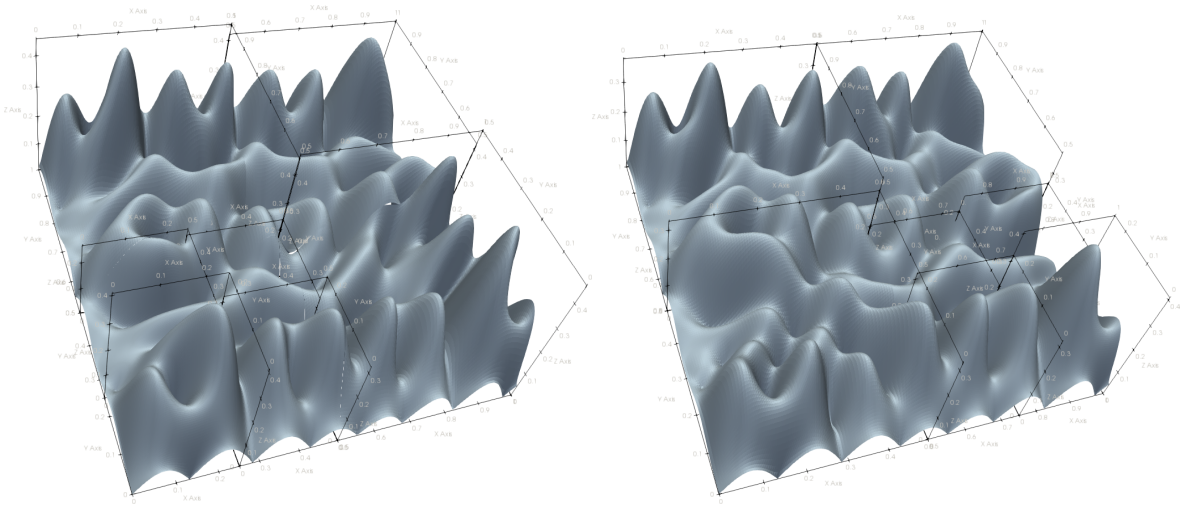

Figure 8: Norm of the gradient of the piecewise low-rank output of ARPLRA using LRHI and the CA algorithm. Note the discontinuity along the junction at the right of the left picture

algorithm	error tolerance $(-\log_{10})$	number of domains	gradient jump
CA	2	13	$4.6\ 10^{-2}$
	6	22	$1.0\ 10^{-5}$
LRHI	2	16	$9.3\ 10^{-8}$
	6	28	$8.4\ 10^{-8}$
CA double grid $\varepsilon = 10^{-1}$	2	13	$1.1\ 10^{-2}$
	6	22	$2.1\ 10^{-6}$
CA double grid $\varepsilon = 10^{-3}$	2	16	$2.6\ 10^{-6}$
	6	28	$9.4\ 10^{-8}$

Figure 9: Comparison of the gradient discontinuity of the output of ARPLRA for the different approximation methods with rank 6

indexes.

[2] M. Bebendorf, Approximation of boundary element matrices, Numerische Mathematik 86 (4) (2000) 565–589.

[3] E. Cheney, The best approximation of multivariate functions by combinations of univariate ones, Approximation Theory IV, Academic Press, New York.

[4] W. Light, E. Cheney, Approximation Theory in Tensor Product Spaces, no. Nr. 1169-1170 in Approximation Theory in Tensor Product Spaces, Springer-Verlag, 1985.

[5] J. Schneider, Error estimates for two-dimensional cross approximation, Journal of Approximation Theory 162 (9) (2010) 1685 – 1700.

[6] N. K. Kumar, J. Schneider, Literature survey on low rank approximation of matrices, Linear and Multilinear Algebra 65 (11) (2017) 2212–2244.

[7] W. Hackbusch, A sparse matrix arithmetic based on H-matrices. part I: Introduction to H-matrices, Computing 62 (2) (1999) 89–108.

[8] W. Hackbusch, B. N. Khoromskij, A sparse H-matrix arithmetic. part II: Application to multi-dimensional problems, Computing 64 (1) (2000) 21–47.

[9] M. Bebendorf, S. Rjasanow, Adaptive low-rank approximation of collocation matrices, Computing 70 (1) (2003) 1–24.

[10] T. Hughes, J. Cottrell, Y. Bazilevs, Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (39) (2005) 4135 – 4195.

[11] A. Mantzaflaris, B. Jüttler, B. N. Khoromskij, U. Langer, Matrix generation in isogeometric analysis by low rank tensor approximation, in: J.-D. Boissonnat, A. Cohen, O. Gibaru, C. Gout, T. Lyche, M.-L. Mazure, L. L. Schumaker (Eds.), Curves and Surfaces, Springer International Publishing, Cham, 2015, pp. 321–340.

[12] I. Georgieva, C. Hofreither, An algorithm for low-rank approximation of bivariate functions using splines, Journal of Computational and Applied Mathematics 310 (2017) 80 – 91, numerical Algorithms for Scientific and Engineering Applications.

[13] A. Mantzaflaris, B. Jüttler, B. N. Khoromskij, U. Langer, Low rank tensor methods in galerkin-based isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 316 (2017) 1062 – 1085, special Issue on Isogeometric Analysis: Progress and Challenges.

[14] F. Scholz, A. Mantzaflaris, B. Jüttler, Partial tensor decomposition for decoupling isogeometric galerkin discretizations, Computer Methods in Applied Mechanics and Engineering 336 (2018) 485 – 506.

[15] M. Pan, F. Chen, W. Tong, Low-rank parameterization of planar domains for isogeometric analysis, Computer Aided Geometric Design 63 (2018) 1 – 16.

[16] S. A. Coons, Surfaces for computer-aided design of space forms, Tech. rep., Massachusetts Institute of Technology, Cambridge, MA, USA (1967).

[17] B. Jüttler, D. Mokriš, Low rank interpolation of boundary spline curves, Computer Aided Geometric Design 55 (2017) 48 – 68.

[18] N. Dyn, B. Jüttler, D. Mokriš, On transfinite interpolation by low-rank functions, Tech. rep., NFN Geometry + Simulation (2017).

[19] W. Gordon, Blending-function methods of bivariate and multivariate interpolation and approximation, SIAM Journal on Numerical Analysis 8 (1) (1971) 158–177.

[20] F. Zhang, The Schur complement and its applications, Vol. 4 of Numerical Methods and Algorithms, Springer, New York, 2005.

[21] M. Bebendorf, Adaptive cross approximation of multivariate functions, Constructive Approximation 34 (2) (2011) 149–179.