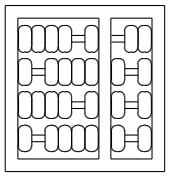


INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Adaptive Multiscale Function Approximation I: General Discrete Bases

Gilcélia Regiâne de Souza *Jorge Stolfi*

Technical Report - IC-15-08 - Relatório Técnico

December - 2015 - Dezembro

Adaptive Multiscale Function Approximation I: General Discrete Bases

Gilcélia Regiâne de Souza[1]
Jorge Stolfi[2]

1 Introduction

In this paper we describe efficient algorithms for adaptive multiscale approximation of functions that are sampled with uneven density and/or have important small-scale detail limited to small portions of their domain. Such functions and datasets are very common in natural sciences and engineering, where different physical causes often give rise to effects with very different scales. Our algorithms are very general, independent of domain shape, mesh, and approximation basis.

We assume that the function to be approximated (the *target function*) is nominally defined on some multi-dimensional domain, but is known only at a finite number of *sampling points* in that domain. Every element of our approximation bases is assumed to be some continuous function with relatively small support, sampled at the same points of the domain as the target function itself. For the purpose of approximating the given samples function values, only the sampled values of the elements are relevant. However, the analytic expression of the elements is relevant if the approximation is to be interpolated at arbitrary points of the domain.

A *single-scale* basis has linearly independent elements with supports of similar size, that cover the domain. A full *multiscale* basis is a hierarchy of single-scale bases, whose elements have progressively smaller supports and are arranged more densely over the domain.

An *adaptive* multiscale basis comprises a subset of a full one, that excludes elements that are found to contribute very little to the approximation. The resulting basis has higher flexibility in those parts of the domain where the target function has more small-scale detail and/or is sampled more densely. In many problems, an adequate adaptive basis may be orders of magnitude smaller than a full multiscale basis with the same accuracy.

In section 2 we define the basic approximation problem in a discrete setting, and related concepts. In section 3 and 4 we describe our general algorithms for single-scale and multiscale adaptive approximation.

An earlier version of this *HApp* algorithm was described in the Ph. D. Thesis of the first author [1].

[1]UFSJ, Ouro Branco, Brazil gilcelia@ufsj.edu.br
[2]UNICAMP, Campinas, SP, Brazil stolfi@ic.unicamp.br

2 The discrete approximation problem

In principle, the target function and the approximations are real-valued functions defined on some domain $\mathbb{D} \subseteq \mathbb{R}^d$, for some dimension d. However, in our approximation method we only evaluate those functions at a fixed list of sampling points $p = (p_1, p_2, \cdots, p_N)$ in $\mathbb{D}$. Therefore, we can regard any real-valued function F on $\mathbb{D}$ as a vector f of $\mathbb{R}^N$, whose component f_i is the value of the function at point p_i.

With this premise, we define a *discrete approximation problem* on those sampling points as consisting of:

- the *target function space* $\mathcal{F}$, which is a linear subspace of $\mathbb{R}^N$;
- the *approximation function space* $\mathcal{S}$, another linear subspace of $\mathbb{R}^N$;
- the *approximation criterion*, a predicate on $\mathcal{F} \times \mathcal{S}$.

An *instance* of the problem is a vector f in $\mathcal{F}$, a *target vector*, representing the target function to be approximated. A *solution* for that instance is a vector s from $\mathcal{S}$, such that the pair (f, s) satisfies the approximation criterion. An *approximation method* for such a problem is a numerical algorithm that yields a solution for any given instance.

Usually, the target space $\mathcal{F}$ is relevant only for analysis and empirical evaluations; it is not used in the approximation algorithms themselves.

For a given target function $f \in \mathcal{F}$ and a candidate approximation $s \in \mathcal{S}$, we define the *residual* as the difference $e = f - s$, which is an element of the *space of residuals* $\mathcal{E} = \mathcal{F} + \mathcal{S} \subseteq \mathbb{R}^N$.

We will assume that the space $\mathcal{S}$ is defined by a *basis matrix* S, with $N \times n$ entries, such that S_{ij} is the value of the basis element with index j on the sampling point p_i, for $i \in \{1, \cdots, N\}$ and $j \in \{1, \cdots, n\}$. Therefore, the approximation s can be written as

$$s_i = \sum_{j=1}^{n} \alpha_j S_{ij} \tag{1}$$

for all i in $1, 2, \cdots, N$, where $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ is a (column) vector of n real coefficients; that is,

$$s = \sum_{j=1}^{n} \alpha_j \sigma_j \tag{2}$$

where σ_j is column j of S; or simply as $s = S\alpha$. Note that the same approximation space $\mathcal{S}$ has infinitely many basis matrices S.

The cost of computing the matrix S from the definition is proportional to nN, assuming that each element can be evaluated in bounded time independent of n and N. However, if each element j is know to be zero outside of some small region $R_j \subseteq \mathbb{D}$, it needs to be evaluated only at the points p_i that lie inside R_j. Further savings in computation time may be achievable by exploiting symmetries of the sampling point grid and of the basis elements.

Typically, the dimension n of $\mathcal{S}$ is much smaller than the dimension m of $\mathcal{F}$ and the number N of sampling points.

The discrete approximation problem could be generalized by allowing the target function values to be elements of some vector space $\mathcal{V}$ with finite dimension k, rather than real numbers. However, approximating such a function f is often equivalent to solving k separate approximation problems for k real-valued functions, all with the same space $\mathcal{S}$ of real-valued approximation functions.

2.1 Norms

A norm $\|\cdot\|_{\mathcal{V}}$ of some vector space $\mathcal{V}$ is said to be *Hilbertian* if it is derived from some inner product $\langle\cdot|\cdot\rangle_{\mathcal{V}}$ on $\mathcal{V}$ by the formula $\|f\| = \sqrt{\langle f|f\rangle}$. Every inner product $\langle\cdot|\cdot\rangle$ on $\mathbb{R}^N$ (or a subspace thereof) can be described by a symmetric matrix $E \in \mathbb{R}^{N\times N}$ such that $\langle f|g\rangle_{\mathcal{E}} = f^\top E g$ for all $f, g \in \mathbb{R}^N$. In this work we will often use the *Euclidean norm* $\|\cdot\|_2$ defined by

$$\|f\|_2 = \sqrt{f^\top f} = \sqrt{\sum_{i=1}^{N} f_i^2}. \tag{3}$$

associated to the ordinary inner product $\langle f|g\rangle_2 = \sum_{i=1}^N f_i g_i$, whose matrix E is the $N \times N$ identity matrix I_N. We will also use the *root-mean-square norm* (or *RMS norm*) $\|\cdot\|_{\overline{2}}$ defined by $\|g\|_{\overline{2}} = \|g\|_2 / \sqrt{N}$. This norm is associated to the inner product $\langle f|g\rangle_{\overline{2}} = \langle f|g\rangle_2/N$, whose E matrix is $(1/N)I_N$. Compared to the Euclidean norm, the RMS norm is less sensitive to the number N of sampling points.

2.2 Approximation and evaluation criteria

Approximation criteria usually involve some norm $\|\cdot\|_{\mathcal{E}}$ for the space of residuals $\mathcal{E}$. We will define the *error* of an approximation $s \in \mathcal{S}$ for a target $f \in \mathcal{F}$ as being the norm of the residual, $\|f - s\|_{\mathcal{E}}$. A typical approximation criterion requires that the error be minimum among all approximations $s \in \mathcal{S}$. Alternatively, one may require only that the error $\|f - s\|_{\mathcal{E}}$ be smaller than a given *tolerance* $\epsilon_{\max} > 0$.

Another common approximation criterion (that does not directly involve the norm $\|\cdot\|_{\mathcal{E}}$) is to require that s *interpolates* f at a certain subset $Q = \{p_{i_1}, p_{i_2}, \ldots, p_{i_n}\}$ of the sampling points, where n is the dimension of $\mathcal{S}$ and $\{i_1, i_2, \cdots, i_n\}$ is a subset of $\{1, 2, \cdots, N\}$ with n elements. This criterion can be used only if the restriction of $\mathcal{S}$ to those sampling points has the same dimension n as the full space $\mathcal{S}$. It is actually a special case of the minimum-norm criterion with the Hilbertian norm

$$\|f\|_2^Q = \sqrt{\sum_{k=1}^{n} (f_{i_k})^2} \tag{4}$$

where $f_{i_k} = f(p_{i_k})$.

2.3 Approximation operators

For a given choice of approximation space $\mathcal{S}$, and for a given approximation criterion that determines a *unique* approximation s in $\mathcal{S}$ for every target f in $\mathcal{F}$, we can define the *ap-*

proximation operator $\mathcal{O}$, that maps f to s. We can also define the *residual operator* $\mathcal{R}$ that maps f to the residual $f - s$; that is, $\mathcal{R} = \mathcal{I} - \mathcal{O}$, where $\mathcal{I}$ is the identity operator on $\mathcal{F}$.

Finally, we define the *analysis operator* $\mathcal{C}$ that, given f and a basis S of $\mathcal{S}$, returns the coefficients of $s = \mathcal{O}(f)$ in that basis; namely, the vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ of formula (1). Note that the operators CO and $\mathcal{R}$, depend only on the space $\mathcal{S}$ and on the approximation criterion; whereas the operator $\mathcal{C}$ depends also on the basis S chosen for $\mathcal{S}$.

The minimum error criterion with an Hilbertian norm (and, in particular, the interpolation criterion) results in an approximation operator $\mathcal{O}$ that is *linear*; that is, $\mathcal{O}(af + bg) = a\mathcal{O}(f) + b\mathcal{O}(g)$ for any $a, b \in \mathbb{R}$ and $f, g \in \mathcal{F}$. This is not true for non-Hilbertian norms like $\|\cdot\|_\infty$.

If $\mathcal{O}$ is linear, then $\mathcal{R}$ and $\mathcal{C}$ are linear operators, too. In that case, the three operators can be represented by matrices O, R, and C with sizes $N \times N$, $N \times N$, and $N \times n$, respectively. In this case, we can write

$$s_i = \sum_{k=1}^{N} O_{ik} f_k \quad \alpha_j = \sum_{k=1}^{N} C_{kj} f_k \tag{5}$$

That is,

$$s = \sum_{k=1}^{N} f_k \omega_k \quad \alpha = \sum_{k=1}^{N} f_k \gamma_k \tag{6}$$

where ω_k is column k of matrix O and γ_k is column of C. Note that ω_k and γ_k describe the influence of the function sample $f_k = f(p_k)$ on the approximation s and on the coefficient vector α, respectively.

We extend $\mathcal{O}$ and $\mathcal{R}$ to the residual space $\mathcal{E} = \mathcal{F} + \mathcal{S}$ by stating that, for any $t \in \mathcal{S}$, $\mathcal{O}(f + t) = \mathcal{O}(f) + t$ and $\mathcal{R}(f + t) = \mathcal{R}(f)$. With this convention, the operators $\mathcal{O}$ and $\mathcal{R}$ are *projections* (idempotent operators) of $\mathcal{E}$, that is, $\mathcal{O}(\mathcal{O}(f)) = \mathcal{O}(f)$ and $\mathcal{R}(\mathcal{R}(f)) = \mathcal{R}(f)$ for any $f \in \mathcal{E}$. Furthermore, $\mathcal{O}(\mathcal{R}(f)) = \mathcal{R}(\mathcal{O}(f)) = 0_N$ (the null vector of $\mathbb{R}^N$). Therefore, $\mathcal{O}$ and $\mathcal{R}$ determine a decomposition of $\mathcal{E}$ into two disjoint orthogonal subspaces, namely the space $\mathcal{S}$ and the complement of $\mathcal{S}$ in $\mathcal{E}$ that is orthogonal to $\mathcal{S}$ in the inner product associated to the error norm that is to be minimized. Note also that $\mathcal{C}(\mathcal{O}(f), S) = \mathcal{C}(f, S)$ for any f in $\mathcal{E}$, and $\mathcal{C}(\mathcal{R}(f), S) = 0_n$.

2.4 Least squares approximation

When the approximation criterion is to minimize some Hilbertian norm $\|\cdot\|_\mathcal{E}$, the optimum coefficient vector α can be found by the *least squares* (LS) method [2], namely by solving the linear system $M\alpha = b$, where the $n \times n$ matrix M and the n-element vector b are defined by

$$M = S^\top E S \qquad b = S^\top E f \tag{7}$$

The matrix of the analysis operator $\mathcal{C}$ is therefore $C = M^{-1} S^\top E = (S^\top E S)^{-1} S^\top E$, and the matrices of the operators $\mathcal{O}$ and $\mathcal{R}$ are

$$O = S(S^\top E S)^{-1} S^\top E \qquad R = I_N - O \tag{8}$$

For the Euclidean norm $\|\cdot\|_2$, where $E = I_N$, these formulas simplify to

$$M = S^\top S \qquad b = S^\top f \qquad C = (S^\top S)^{-1} S^\top \qquad O = S(S^\top S)^{-1} S^\top \tag{9}$$

For the RMS norm $\|\cdot\|_{\overline{2}}$, which has $E = (1/N)I_N$, we have

$$M = \frac{1}{N}S^\top S \qquad b = \frac{1}{N}S^\top f \tag{10}$$

and (as expected) the operators $\mathcal{O}$, $\mathcal{R}$, and $\mathcal{C}$ are the same as those of the Euclidean norm.

When using the interpolation criterion, on the subset $Q = \{p_1, p_2, \cdots, p_{i_n}\}$, the least squares method can be simplified to solving $M\alpha = b$ where

$$M_{kj} = S_{i_k j} \qquad b_k = f_{i_k} \tag{11}$$

for all k and j in $\{1, \ldots, n\}$.

If the matrix S is dense, the cost of computing α given S and f is proportional to n^2N for the computation of M and n^3 for solving the system $M\alpha = b$ or $n^2(N + n)$ total.

On the other hand, if the basis element have small support specifically, if each column of S has only $q \ll N$ nonzero elements, on average, the cost of computing M is only n^2q but the cost of solving the system will still be to close to n^3; then the cost of the LS method will be proportional to $n^2(N + n)$.

3 Basis reduction

As we observed in section 1, it may be possible to remove from the approximation formula (2) any terms that make a negligible contribution to it. That is, we replace a generic *prebasis matrix* S with columns $\sigma_1, \sigma_2, \cdots, \sigma_n$, independent of the target function f, by a *reduced basis matrix* $\widehat{S}$ with columns $\widehat{\sigma}_1, \widehat{\sigma}_2, \ldots, \widehat{\sigma}_{\widehat{n}}$, specific to f; so that the approximation becomes

$$\widehat{s} = \sum_{k=1}^{\widehat{n}} \widehat{\alpha}_k \widehat{\sigma}_k, \tag{12}$$

where $\widehat{\alpha} = (\widehat{\alpha}_1, \widehat{\alpha}_2, \ldots, \widehat{\alpha}_{\widehat{n}})$ is the *reduced coefficient vector*. The computational problem then includes the choice of the reduced basis $\widehat{\sigma}$, as well as the corresponding coefficient vector $\widehat{\alpha}$.

As we shall see in section 4, basis reduction is an essential step in the construction of multiscale adaptive bases. In general, basis reduction is worth the cost only if both the target function f and the basis elements have small support.

3.1 Reduction criterion

For basis reduction, the criterion of mimimum approximation error (as used in the LS method) is no longer adequate, since in general it would be satisfied only when $\widehat{S} = S$. Typically one seeks a tradeoff between minimizing some error norm $\|\cdot\|_*$ of the residual $f - \widehat{s}$ and minimizing the size n of the reduced basis (the number of columns of $\widehat{S}$). For example, one may require that n be as small as possible as long as $\|f - s\|_*$ does not exceed a given tolerance $\epsilon_{\max}$.

When the norm $\|\cdot\|_*$ is Hilbertian, and the prebasis elements σ_j are orthonormal under the associated inner product $\langle\cdot|\cdot\rangle_*$ (that is, $S^\top ES = I_n$, then every element $\widehat{\sigma}_j$ of the optimal

approximation with any sub-basis $\widehat{S}$ of S has the same coefficient as the corresponding element σ_k in the optimal approximation with the full prebasis S. In that case, the smallest reduced basis $\widehat{S}$ that satisfies $\|f - s\|_{\mathcal{E}} \leq \epsilon_{\max}$ can be obtained by eliminating columns σ_j of S in increasing order of $|\alpha_j|$, while the Euclidean norm of the discarded coefficients α_j is less than $\epsilon_{\max}$.

Unfortunately, many prebases that are useful in practice (including the ones used in this article) are not orthogonal, so we cannot use the simple reduction algorithm above. This algorithm also fails if the error norm $\|\cdot\|_*$ is not Hilbertian. Without those conditions, finding the smallest reduced basis $\widehat{\sigma}$ that satisfies $\|f - s\|_* \leq \epsilon_{\max}$ is a difficult problem, for which no efficient algorithm is currently known.

It should be noted that any basis reduction method is inherently non-linear. Therefore, the operations $\mathcal{O}$, $\mathcal{R}$ and $\mathcal{C}$ are not linear in general adaptive approximation methods.

3.2 The sequential truncation heuristic

Since we cannot find the optimal solution, we use a heuristic method that is not overly expensive and has been shown to produce reasonably small bases. We first compute an approximation $s = \sum_{j=1}^{n} \alpha_j \sigma_j$ by the least squares method, that is, one that minimizes the Euclidean norm of the residual. Then we the apply the *sequential truncation heuristic*, formalized by the procedure *Reduce* below, that discards terms of equation (1) while the uniform norm $\|\cdot\|_\infty$ of the residual does not increase too much. The inputs of *reduce* are:

- the $N \times n$ prebasis matrix S with columns $\sigma_1, \sigma_2, \cdots, \sigma_n$,
- the corresponding coefficient vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$,
- the residual $e = f - s = f - S\alpha$, also in discrete form (an N-vector).
- a tolerance $\epsilon_{\max} > 0$.

The *Reduce* algorithm returns a reduced basis matrix $\widehat{S}$ (whose columns $\widehat{\sigma}_1, \widehat{\sigma}_2, \ldots, \widehat{\sigma}_n$ are a subset of the columns of S); and a vector $\widehat{\alpha} = (\widehat{\alpha}_1, \widehat{\alpha}_2, \ldots, \widehat{\alpha}_{\widehat{n}})$ of the corresponding coefficients (which are a subset of the elements of α). The algorithm guarantees that the residual $\widehat{e} = f - \widehat{s}$ of the reduced approximation $\widehat{s} = f - \sum_k \widehat{\alpha}_k \widehat{\sigma}_k$ satisfies the following condition, for every sampling point p_i:

- if $|e_i| \leq \epsilon_{\max}$, then $|\widehat{e}_i| \leq \epsilon_{\max}$;
- if $|e_i| > \epsilon_{\max}$, then $|\widehat{e}_i| \leq |e_i|$.

In other words, *Reduce* may increase the error $|e_i|$ at each sampling point, as long as it does not exceed $\epsilon_{\max}$; but if some $|e_i|$ already exceeds $\epsilon_{\max}$, then that value will not increase.

Algorithm 1. *Reduce*$(S, \alpha, e, \epsilon_{\max})$

1. Compute the values $\delta_j = |\alpha_j| \, \|\sigma_j\|_\infty$ for $j = 1, \ldots, n$.
2. For each index j in $1, \ldots, n$, in order of increasing δ_j, do:
   3. If $\delta_j > 2\epsilon_{\max}$, go to step 6.
   4. If, for all i, either $|e_i| \leq \epsilon_{\max}$ and $|e_i + \alpha_j \sigma_{ji}| \leq \epsilon_{\max}$, or $|e_i| > \epsilon_{\max}$ and $|e_i + \alpha_j \sigma_{ji}| \leq |e_i|$, then do:
      5. $e \leftarrow e + \alpha_j \sigma_j$; $\alpha_j \leftarrow 0$.
6. Return the basis matrix $\widehat{S}$ consisting of those $\widehat{n}$ columns σ_j with $\alpha_j \neq 0$, and the corresponding vector of coefficients $\widehat{\alpha}$.

It is easy to check that the output of the *Reduce* algorithm satisfies the criterion above. The test $\delta_j > 2\epsilon_{\max}$ in step 3 stops the algorithm when it is very unlikely that any of the remaining elements will be eliminated.

The cost of *Reduce* is dominated by the computation of the values δ_j in step 1, by the test of step 4, and by the update of the residual e in step 5. The cost of step 1 is proportional to nN. Note that the sorting of $\delta_1, \delta_2, \ldots, \delta_n$ in step 2 has cost proportional to $n \log n$, which is usually negligible. Each iteration of steps 4 and 5 has cost proportional to N; since these steps are executed at most n times, its total cost — and therefore the cost of *Reduce* — is bounded by a constant times nN.

This heuristic may not find the optimal solution (the smallest possible basis), because the elimination of an element σ_j may prevent the algoritm from eliminating two elements later on. Moreover, the coefficients $\widehat{\alpha}$ returned may not be optimal even by the LS criterion, applied to the basis $\widehat{S}$.

4 Adaptive multiscale approximation

Our adaptive multiscale approximation method consists of the application of the *Reduce* heuristic at multiple scales of detail, from coarsest to finest. At each scale, the target function is the residual of the approximation at the previous level. At each scale, we use a prebasis that is only large enough to cover the part of the domain where the current residual is non negligible.

4.1 Hierarchical prebasis

We define a *hierarchical prebasis* as being a sequence of *levels* $S^{(\ell_{\min})}, S^{(\ell_{\min}+1)}, \ldots S^{(\ell_{\max})}$. Each level $S^{(\ell)}$ is a discrete approximation basis, defined over the domain of interest $\mathbb{D}$. The basis $S^{(\ell)}$ is such that the generated space $\mathcal{S}^{(\ell)}$ allows the modeling of smaller details than the space $\mathcal{S}^{(\ell-1)}$.

In many multiscale approximation schemes [3], the spaces $\mathcal{S}^{(\ell)}$ and $\mathcal{S}^{(k)}$ of different levels are required to be linearly independent, or even orthogonal. We do not make that assumption, so that we have more freedom in the choice of the bases $S^{(\ell)}$ and in the approximation operators.

4.2 Element cells and hierarchical meshes

For the algorithm, we assume that each element $\sigma_j^{(\ell)}$ of each basis level $S^{(\ell)}$ has an associated *cell* $\mathbb{K}_j^{(\ell)}$, a subset of the domain $\mathbb{D}$ where that element dominates in some sense. We will denote by $\mathfrak{K}^{(\ell)}$ the set of all cells of level ℓ. The algorithm does not impose any restriction on these cells of any single level ℓ, except that every sampling point must be contained in exactly one cell. The integer j is the cell's *index*. We define the *support* of an element $\sigma_j^{(\ell)}$ as being the set $\mathrm{supp}(\sigma_j^{(\ell)})$ of cells of a given level ℓ where element $\sigma_j^{(\ell)}$ is non-zero. We assume that the cell $\mathbb{K}_j^{(\ell)}$, in particular, belongs to $\mathrm{supp}(\sigma_j^{(\ell)})$.

In the discrete version, each cell $\mathbb{K}_j^{(\ell)}$ is a set of row indices of the full pre-basis matrix $S^{(\ell)}$, corresponding to the sampling points p_i where column σ_j is assumed to predominate; and all the sets $\mathbb{K}_j^{(\ell)}$ with same ℓ form a partition of the set $\{1, 2, \ldots, N\}$.

The algorithm assumes that the cells of successive levels are properly nested; that is, for all r, j, and ℓ, cell $\mathbb{K}_r^{(\ell+1)}$ is either disjoint of cell $\mathbb{K}_j^{(\ell)}$, or is contained in it. The *children* of a cell $\mathbb{K}_j^{(\ell)}$ are all the cells $\mathbb{K}_r^{(\ell+1)}$ of the next level that are contained in it. For any set $\mathcal{X}$ of indices of cells of some level ℓ, we will denote by $\mathbb{K}_{\mathcal{X}}^{(\ell)}$ the union of the cells of level ℓ with those indices, that is, $\cup_{j\in\mathcal{X}}\mathbb{K}_j^{(\ell)}$.

4.3 Adaptive multiscale approximation

If all elements at all levels were linearly independent (that is, if $\mathcal{S}^{(\ell)} \cap \mathcal{S}^{(k)} = \{0_N\}$ whenever $k \neq \ell$), one could consider creating a single prebasis matrix

$$S^{(*)} = S^{(\ell_{\min})} \upharpoonright S^{(\ell_{\min}+1)} \upharpoonright \cdots \upharpoonright S^{(\ell_{\max})},$$

where $\vdash$ denotes side-by-side concatenation of matrices; then compute a single coefficient vector $\alpha^{(*)}$ by the least squares method, and then apply the *Reduce* heuristic to that approximation. However, recall that we are not assuming that the bases $S^{(\ell)}$ of different levels are linearly independent. Moreover, the size of that full multi-level prebasis matrix $S^{(*)}$ would be prohibitive for most applications.

For these reasons, our algorithm builds the approximation level by level, from coarsest to finest; where the problem at each level is to find a parsimonious approximation $s^{(\ell)}$ to the residual $f^{(\ell)} = e^{(\ell-1)} = f^{(l-1)} - s^{(\ell-1)}$, the part of f that is left over from the approximations at the coarser levels. Thus, the approximations $s^{(\ell)}$ obtained at all levels must be added together to obtain the final approximation s.

Moreover, at each level, our algorithm uses as prebasis only a submatrix $\tilde{S}^{(\ell)}$ of the full basis matrix $S^{(\ell)}$. The submatrix $\tilde{S}^{(\ell)}$ includes only the elements of $S^{(\ell)}$ that cover those parts of the domain where the uniform norm of that residual $f^{(\ell)}$ is still too large. Indeed, it is very important for the efficiency of the algorithm that the full basis matrix $S^{(\ell)}$ is never computed.

4.4 The hierarchical approximation algorithm

Our hierarchical approximation algorithm is described more precisely by the procedure *HApp* below. Its inputs are:

- the sampled target function f (a vector of $\mathbb{R}^N$),
- the indices $\ell_{\min}, \ell_{\max}$ of the first and last levels, with $1 \leq \ell_{\min} \leq \ell_{\max}$;
- a procedure *GetAllCells*(ℓ) that returns the indices of all cells (that is, elements of the full basis) of a given level ℓ;
- a procedure *GetChildrenCells*(ℓ, j) that returns a list of the indices r all children cells $\mathbb{K}_r^{(\ell+1)}$ of a given cell $\mathbb{K}_j^{(\ell)}$;
- a procedure *GetElement*(ℓ, j) that computes the element $\sigma_j^{(\ell)}$ of the discrete basis $S^{(\ell)}$;
- a procedure *GetSupportCells*(ℓ, j) that returns the support of element $\sigma_j^{(\ell)}$, that is, a list of all indices r of all cells $\mathbb{K}_r^{(\ell)}$ of level ℓ that contain at least one sampling point where element $\sigma_j^{(\ell)}$ is nonzero; and
- a tolerance $\epsilon_{\max} > 0$.

The outputs are

- the side-by-side concatenation $\widehat{S}^{(*)} = \widehat{S}^{(\ell_{\min})} \vdash \widehat{S}^{(\ell_{\min}+1)} \vdash \widehat{S}^{(\ell_{\max})}$ of the reduced bases $\widehat{S}^{(\ell)}$;
- the top to bottom concatenation $\widehat{\alpha}^{(*)} = \widehat{\alpha}^{(\ell_{\min})} \amalg \widehat{\alpha}^{(\ell_{\min}+1)} \amalg \cdots \amalg \widehat{\alpha}^{(\ell_{\max})}$ of the corresponding coefficient vector; and
- the computed approximation $\widehat{s}^{(*)} = \sum_k \widehat{\alpha}_k^{(*)} \widehat{\sigma}_k^{(*)}$, where $\widehat{\sigma}_k^{(*)}$ is the column k of $\widehat{S}^{(*)}$.

Algorithm 2. ***Procedure*** *HApp*($\ell_{\min}, \ell_{\max}, f$, *GetAllCells*, *GetChildrenCells*, *GetElement*, *GetSupportCells*, $\epsilon_{\max}$)

1. $\mathfrak{R}^{(\ell_{\min}-1)} \leftarrow$ *GetAllCells*$(\ell_{\min} - 1)$;
2. $\mathfrak{U}^{(\ell_{\min}-1)} \leftarrow \mathfrak{R}^{(\ell_{\min}-1)}$;
3. $\widehat{S}^{(*)} i \leftarrow ()$; $\widehat{\alpha}^{(*)} \leftarrow ()$; $\widehat{s}^{(*)} \leftarrow ()$;
4. $\widehat{e}^{(\ell_{\min}-1)} \leftarrow f$;
5. For $\ell = \{\ell_{\min}, \ell_{\min} + 1, \ldots, \ell_{\max}\}$, do:
   6. $f^{(\ell)} \leftarrow \widehat{e}^{(\ell-1)}$;
   7. $\mathfrak{C}^{(\ell)} \leftarrow \bigcup_{j \in \mathfrak{R}^{(\ell_{\min}-1)}}$ *GetChildrenCells*$(\ell - 1, j)$;
   8. Eliminate from $\mathfrak{C}^{(\ell)}$ the indices of all cells except those that have any points p_i where $\left|f_i^{(\ell)}\right| > \epsilon_{\max}$.
   9. If $\mathfrak{C}^{(\ell)} = \{\}$, go to step 18.
   10. $\mathfrak{R}^{(\ell)} \leftarrow \bigcup_{j \in \mathfrak{C}^{(\ell)}}$ *GetSupportCells*(ℓ, j)
   11. $\mathfrak{U}^{(\ell)} \leftarrow \bigcup_{j \in \mathfrak{R}^{(\ell)}}$ *GetSupportCells*(ℓ, j).
   12. Build the prebasis matrix $\tilde{S}^{(\ell)}$, by calling *GetElement*(ℓ, j) for every index $j \in \mathfrak{C}^{(\ell)}$.
   13. $\tilde{\alpha}^{(\ell)} \leftarrow \mathcal{C}(f^{(\ell)}, \tilde{S}^{(\ell)})$; $\tilde{s}^{(\ell)} \leftarrow \sum_j \tilde{\alpha}_j \tilde{\sigma}_j$.
   14. $\tilde{e}^{(\ell)} \leftarrow f^{(\ell)} - \tilde{s}^{(\ell)}$.
   15. $(\widehat{S}^{(\ell)}, \widehat{\alpha}^{(\ell)}) \leftarrow$ *Reduce*$(\tilde{S}^{(\ell)}, \tilde{\alpha}^{(\ell)}, \tilde{e}^{(\ell)}, \epsilon_{\max})$;
   16. $\widehat{s}^{(\ell)} \leftarrow \sum_k \widehat{\alpha}_k \widehat{\sigma}_k$; $\widehat{e}^{(\ell)} \leftarrow f^{(\ell)} - \widehat{s}^{(\ell)}$.
   17. $\widehat{S}^{(*)} \leftarrow \widehat{S}^{(*)} \shortmid \widehat{S}^{(\ell)}$; $\widehat{\alpha}^{(*)} \leftarrow \widehat{\alpha}^{(*)} \perp\!\!\!\perp \widehat{\alpha}^{(\ell)}$; $\widehat{s}^{(*)} \leftarrow \widehat{s}^{(*)} + \widehat{s}^{(\ell)}$.
18. Return $\widehat{S}^{(*)}$, $\widehat{\alpha}^{(*)}$, $\widehat{s}^{(*)}$.

In steps 7 and 8 of each iteration, this algorithm identifies the set $\mathfrak{C}^{(\ell)}$ of *critical* cells, where the current residual $f^{(\ell)}$ exceeds the tolerance $\epsilon_{\max}$. As we shall see, those cells must be children of a certain set $\mathfrak{R}^{(\ell-1)}$ of *relevant* cells identified in the previous iteration. In step 12, the algorithm builds a prebasis $\tilde{S}^{(\ell)}$ with the elements of the full basis $S^{(\ell)}$ associated to each critical cell. Note that the support of each element may overlap other cells, but is contained in $\mathbb{K}_{\mathfrak{R}^{(\ell)}}$. In step 13, the least squares operator $\mathcal{C}$ is used to obtain the coefficients $\tilde{\alpha}^{(\ell)}$ of an approximation $\tilde{s}^{(\ell)}$ to the current residual $f^{(\ell)}$, in the prebasis $\tilde{S}^{(\ell)}$. The *Reduce* heuristic is then used in step 15 to discard columns from $\tilde{S}^{(\ell)}$ with the constraints stated in section 3, resulting in a submatrix $\widehat{S}^{(\ell)}$ of $\tilde{S}^{(\ell)}$ and the corresponding coefficient vector $\widehat{\alpha}^{(\ell)}$. In step 16, these are combined into the partial approximation $\widehat{s}^{(\ell)}$ and the new residual $\widehat{e}^{(\ell)}$, which will be the target for the next iteration.

4.5 Correctness

Since it is based on the LS method and the *Reduce* heuristic, the procedure *HApp* is not guaranteed to find an approximation with $\|f - s\|_\infty$ less than $\epsilon_{\max}$; unless the prebasis, $\tilde{S}^{(\ell)}$ of the last level has enough elements to exactly match the final residual. If the sampling points

are evenly distributed over the domain, this condition will be satisfied when $|\mathfrak{C}^{(\ell_{\max})}| \geq N$. In many problems of practical interest, however, the tolerance may be satisfied well before that level.

We can also state that the algorithm is correct in the following sense: at each level $\ell \geq \ell_{\min}$, for every sampling point p_i,

- if $p_i \notin \mathbb{K}^{(\ell-1)}_{\mathfrak{R}^{(\ell-1)}}$, then $\left|e_i^{(\ell)}\right| \leq \epsilon_{\max}$;
- $\mathfrak{C}^{(\ell)} \subseteq \mathfrak{R}^{(\ell)}$
- $\mathbb{K}^{(\ell)}_{\mathfrak{U}^{(\ell)}} \subseteq \mathbb{K}^{(\ell-1)}_{\mathfrak{U}^{(\ell-1)}}$;
- $\mathfrak{R}^{(\ell)} \subseteq \mathfrak{U}^{(\ell)}$.

One can also verify that, if p_i is not in any cells $\mathbb{K}^{(\ell)}_r$ for $r \in \mathfrak{R}^{(\ell)}$, (that is $p_i \notin \bigcup \mathfrak{R}^{(\ell)}$), then $\widehat{s}_i^{(\ell)} = 0$.

These statements can be proved by induction on ℓ. Note that, while the critical region $\mathbb{K}^{(\ell)}_{\mathfrak{C}^{(\ell)}}$ and relevant region $\mathbb{K}^{(\ell)}_{\mathfrak{R}^{(\ell)}}$ may expand from one level to the next, the region $\mathbb{K}^{(\ell)}_{\mathfrak{U}^{(\ell)}}$ that contains both may only shrink at each iteration. Moreover, $\mathbb{K}^{(\ell-1)}_{\mathfrak{R}^{(\ell-1)}}$ encloses all points where the residual from the preceding levels still exceeds $\epsilon_{\max}$. (Actually, the sets $\mathfrak{U}^{(\ell)}$ are not used in the algorithm. They are defined only for the purpose of analysis of the algorithm, and may be omitted from an actual implementation.)

Therefore, when looking for points where the error exceeds $\epsilon_{\max}$, the algorithm needs only look inside the cells of level ℓ that are contained in the cells of $\mathfrak{R}^{(\ell-1)}$. Moreover, if the algorithm ends prematurely because the set of critical cells $\mathfrak{C}^{(\ell)}$ becomes empty (step 9), we will have $\left\|f^{(\ell)}\right\|_\infty \leq \epsilon_{\max}$, as desired.

Note also that the entries $\tilde{S}^{(\ell)}_{ij}$ of the basis matrix for level ℓ must be computed in step 12 only for the elements a subset of the indices j, and only for those points p_i that are inside the cells with indices $\mathbb{K}^{(\ell-1)}_{\mathfrak{R}^{(\ell-1)}}$

4.6 Computational Complexity

Note that the size of the support of the basis elements has a significant impact on the efficiency of the adaptive approximations algorithm. As discussed in section 2.4 applying the LS method to a full single scale basis, if the supports of the basis elements cover a large fraction of the domain, the arrays S and M will be dense.

Significant savings in computing time are possible, however, if the sampling points form a regular rectangular grid, and the basis elements of each level ℓ are copies of the same "mother element" translated by multiples of the grid spacing. We will detail this special case in a separate paper.

The total cost of happen will be dominated by the cost of the LS method in each level, that is, to $\tilde{t}^{(*)} = \sum_{\ell=\ell_{\min}}^{\ell_{\max}} \tilde{t}^{(\ell)}$, where $\tilde{t}^{(*)} = (\tilde{n}^{(\ell)})^2(\tilde{q}^{(\ell)} + \tilde{n}^{(\ell)})$ and $\tilde{q}^{(\ell)}$ is the average number of sampling points in the support of each element of the pre basis $\tilde{S}^{(\ell)}$.

In contrast, if we used the single-level basis of level $S^{(\ell_{\max})}$ the cost would be $t^{(\ell_{\max})}$ where $t^{(\ell)} = (n^{(\ell)})^2(q^{(\ell)} + n^{(\ell)})$ and $q^{(\ell)}$ is the average number of sampling points in the support of an element of $S^{(\ell_{\max})}$.

5 Tests

To illustrate and evaluate our method, we used it with the data sets — mesh, sampling points, target vectors, basis elements, and other parameters — described in Sections 5.1 – 5.4. The results of these tests are summarized in Section 5.5.

5.1 Hierarchical mesh

In all our tests, the domain $\mathbb{D}$ is the a regular hexagon $\mathbb{H}$ of circumradius $R_{\mathbb{H}} = 3.92$ centered at the point (4, 4). To define the full pre-basis $S^{(\ell)}$ of each level ℓ, we considered a triangular mesh covering the hexagon $\mathbb{H}$. The hexagon was divided into 6 equilateral triangles, and each triangle was further divided into 4^ℓ equilateral triangles with sides $R_{\mathbb{H}}/2^\ell$. Therefore each cell of level ℓ is divided into 4 cells of level $\ell + 1$, and has $n^{(\ell)} = 6 \times 4^\ell$ cells in total. See Figure 1 .

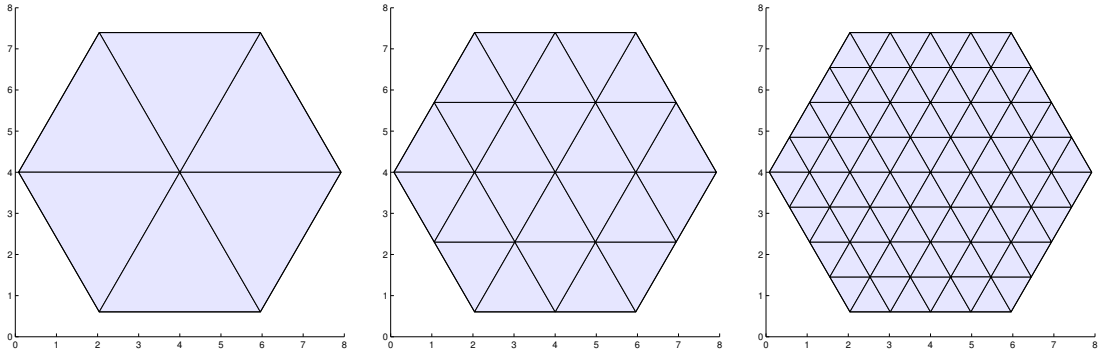
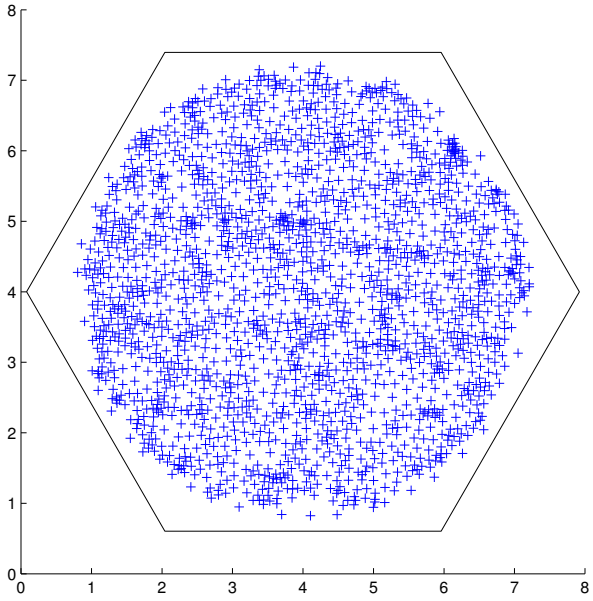


Figure 1: Level 0, 1, and 2 of the hierarchical mesh.

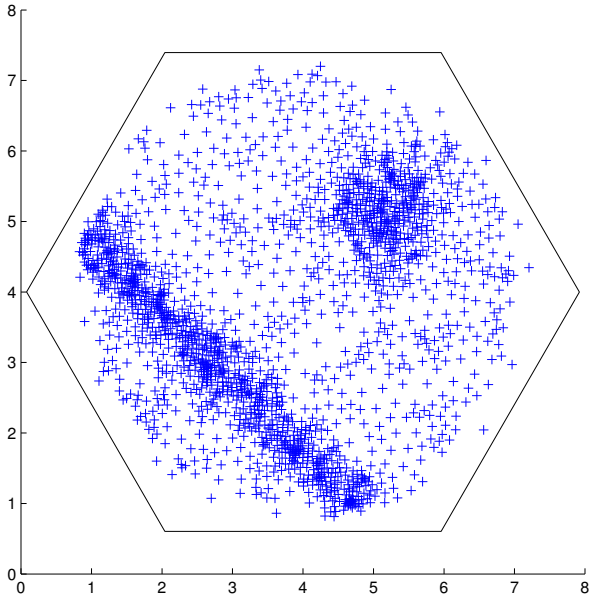
5.2 Sampling points

(that encloses the circle $\mathbb{B}$ of Section 5.2)

In all our tests, the list p of sampling points consists of $N = 20\ 000$ points randomly chosen inside the circle $\mathbb{B}$ with center $(4, 4)$ and radius $R_{\mathbb{B}} = 0.95R_{\mathbb{I}} \approx 3.2250786$, where $R_{\mathbb{I}}$ is the inradius of the hexagon $\mathbb{H}$, $R_{\mathbb{I}} = R_{\mathbb{H}}\sqrt{3}/2$. We used two probability distributions, uniform $\mathscr{D}_{\mathrm{U}}$ and non-uniform $\mathscr{D}_{\mathrm{N}}$. The latter is a uniform distribution mixed with two Gaussian distributions, one concentrated and one spread out along a line. In each case, the N points were initially drawn independently from the corresponding distribution, and then iteratively adjusted to obtain a slightly more uniformly spacing. Namely, we built the Delaunay triangulation of the points, and then replaced each internal vertex by the average of its Delaunay neighbors. This smoothing was repeated 5 times. See figure 2.



(a) Uniform



(b) Non Uniform

Figure 2: A set of $N = 2\,000$ sample points p_i drawn from (a) the uniform distribution $\mathscr{D}_U$ and (b) the non uniform distribution $\mathscr{D}_N$, both after 5 smoothing iterations. The hexagon is the outline of the mesh (see section 5.1).

5.3 Test functions

For our tests we used target vectors f obtained by sampling two continuous target functions from $\mathbb{D}$ to $\mathbb{R}$ (F_{O} and F_{M}), described in Appendix 7. See figure 3. In each test, the discretized target vector of $\mathbb{R}^N$ (f_{O} or f_{M}), was obtained by evaluating the corresponding continuous function at the sampling points.

For visualization purposes, each discretized target function f (as well as the approximations $s^{(\ell)}$ and, residuals $e^{(\ell)}$ were plotted as triangular terrain meshes. Namely, we constructed the Delaunay triangulation of the sampling points $p_1, \ldots, p_N$, and then assigned to each vertex p_i a z coordinate equal to the corresponding function value f_i. See Figure 4.

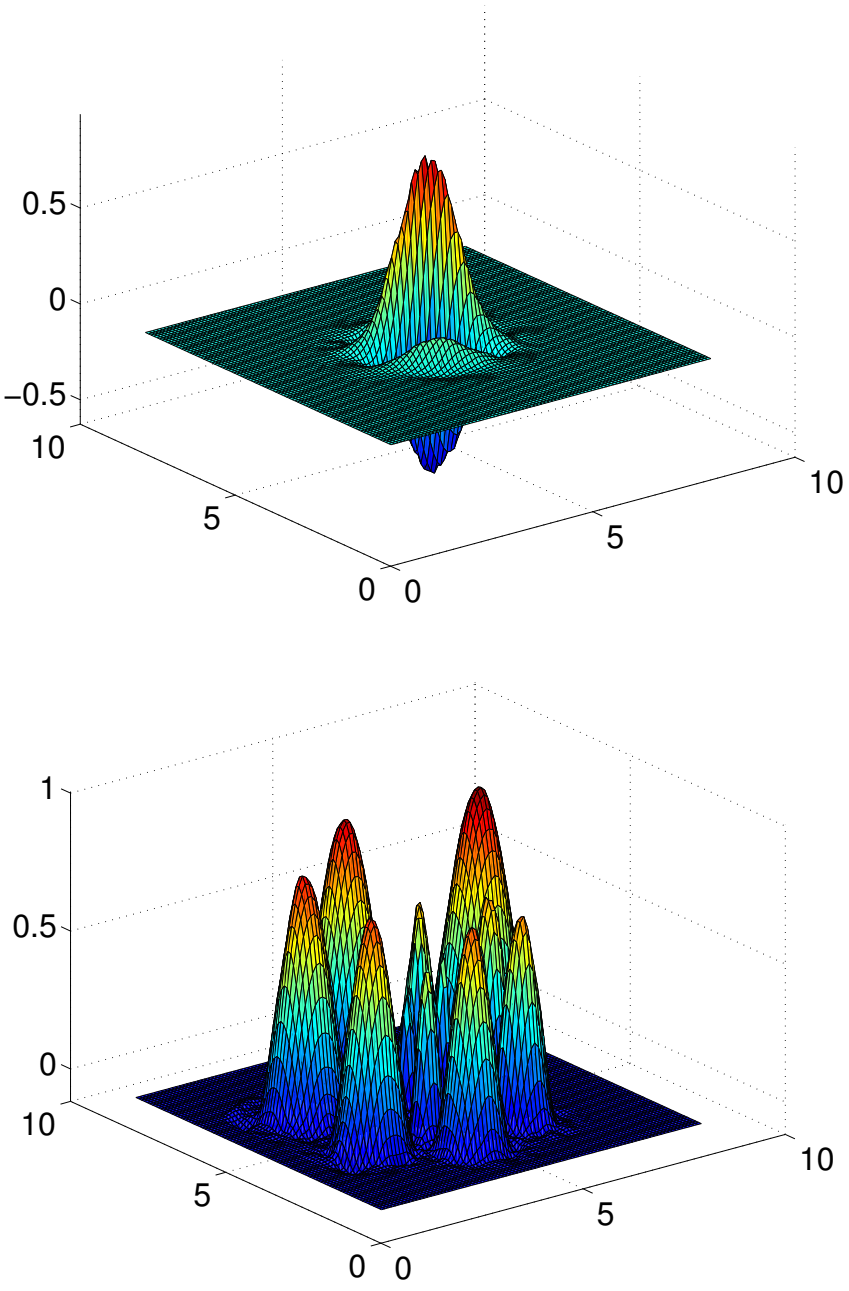


Figure 3: The target functions F_{O} (left) and F_{M} (right) used in our tests, before sampling, plotted on a regular square grid.

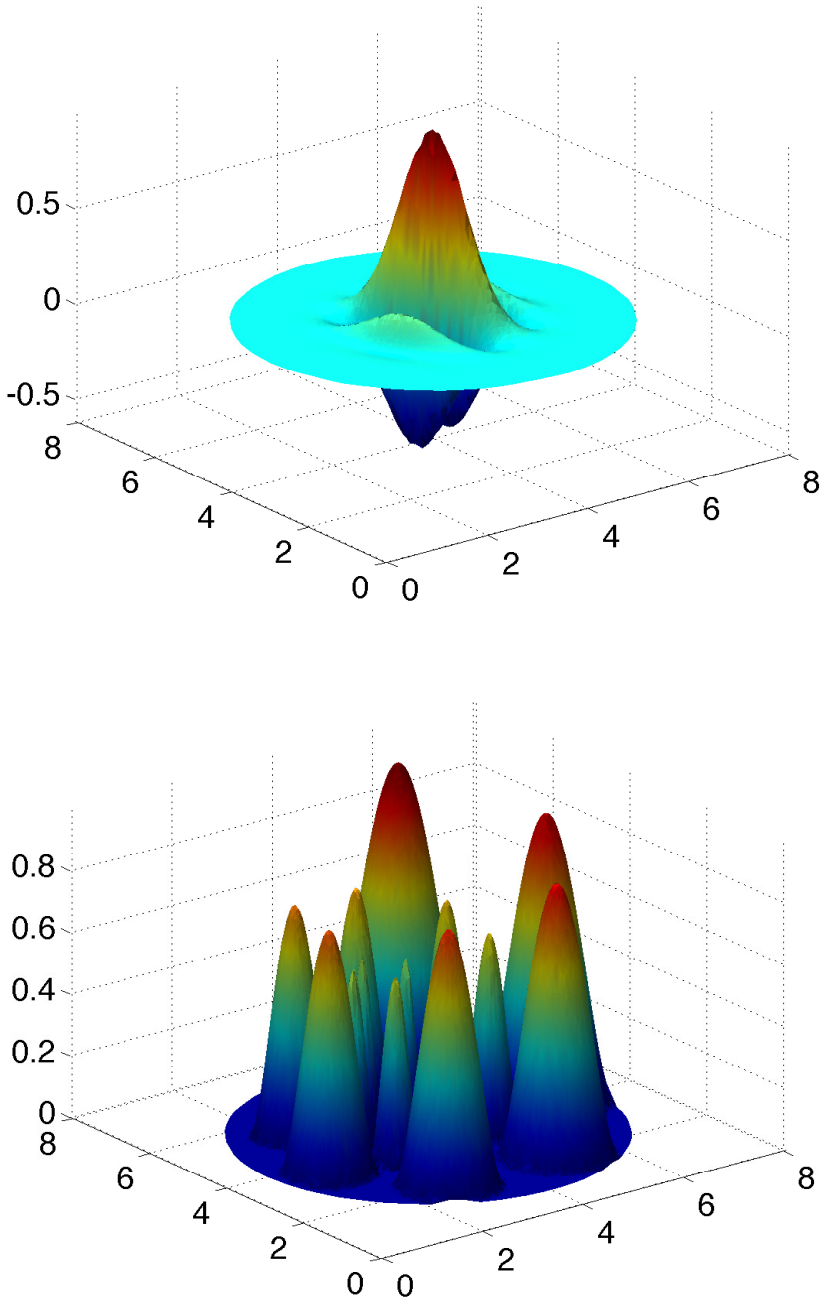


Figure 4: The target functions f_{O} (left) and f_{M} (right), used in our tests, sampled at $N = 20000$ points with distribution $\mathscr{D}_{\mathrm{U}}$ of Figure 2, and visualized as terrains with the Delaunay triangulations of the sampling points.

5.4 Basis elements

Each basis element $\sigma_j^{(\ell)}$ was a C^1 finite spline element consisting of 13 triangular Bézier patches of degree 3, defined on the central cell and the surrounding 12 cells. See Figure 5.

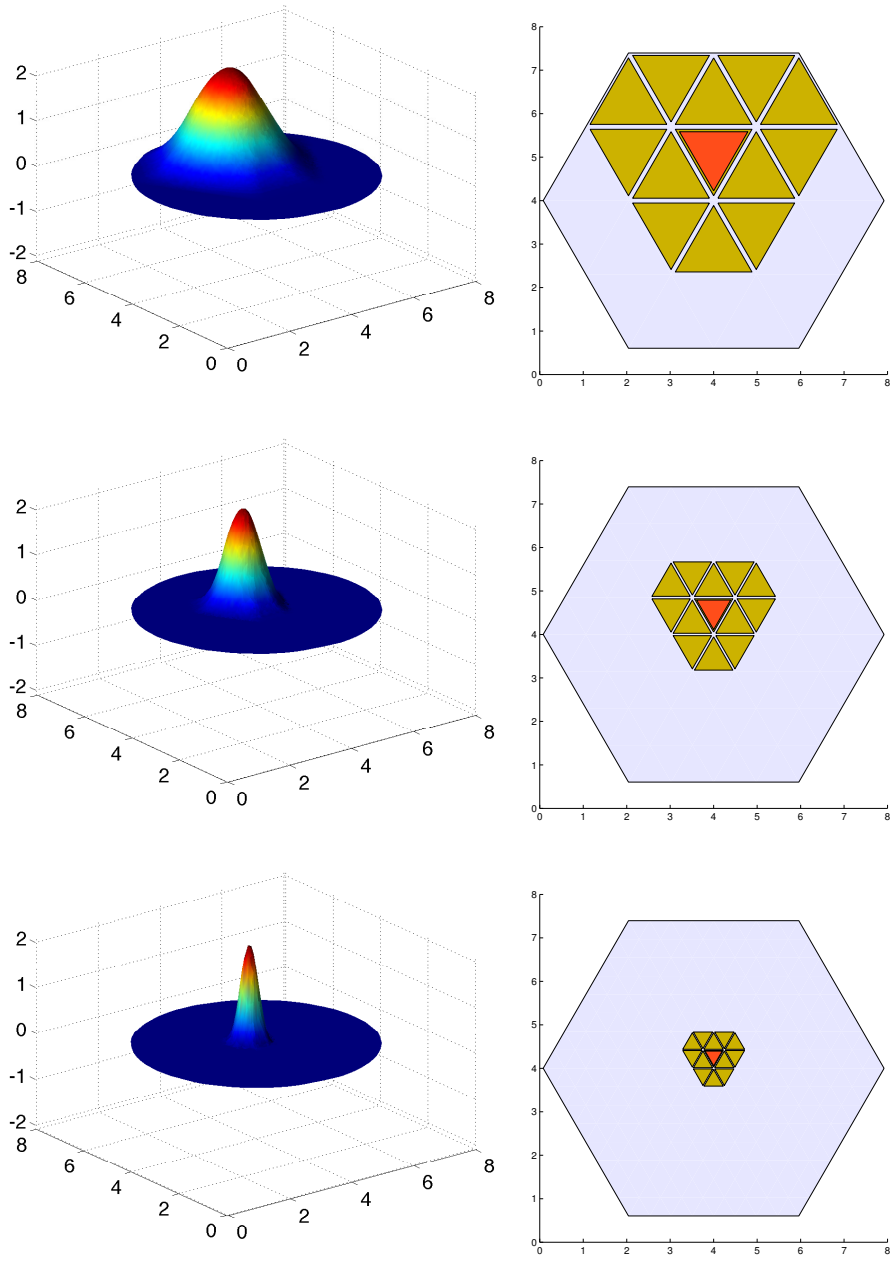


Figure 5: At left, typical basis elements with mother function Φ_2^{S} at levels 1, 2, and 3 (from top to bottom), sampled at $N = 20\,000$ points with distribution $\mathscr{D}_{\mathrm{U}}$. At right, the support of each element (that is, the cells where the element is nonzero).

5.5 Results

For each target function f we computed a hierarchical adaptive approximation $\widehat{s}^{(*)}$ with the *HApp* algorithm with the parameters $\ell_{\min} = 3$, $\ell_{\max} = 8$, and $\epsilon_{\max} = 1.0 \times 10^{-4}$.

The results of the tests are summarized in table 1. The table shows the number $\widehat{n}^{(k)}$ elements on the reduced basis for each level $\ell \in \{3, \ldots, 8\}$, and the total number $\widehat{n}^{(*)}$ of elements in the final adaptive basis $\widehat{S}^{(*)}$. Plots of the results of selected tests are shown in sections 5.5.1.

For each test, we measured the efficiency of the final approximation by the ratio τ between $\widehat{n}^{(*)}$ and the number $n_{\max} = n^{(\ell_{\max})}$ of elements in the last level $S^{(\ell_{\max})}$ of the full hierarchical basis. The last column of table 1 shows the ratio λ between the total estimated running time $\tilde{t}^{(*)}$ of the LS method applied to all reduced bases $\tilde{S}^{(\ell)}$, and the estimated time $t^{(\ell_{\max})}$ of the LS method with the full basis $S^{(\ell_{\max})}$.

Table 1: Summary of test results.

f	$\mathscr{D}$	Φ	$\widehat{n}^{(3)}$	$\widehat{n}^{(4)}$	$\widehat{n}^{(5)}$	$\widehat{n}^{(6)}$	$\widehat{n}^{(7)}$	$\widehat{n}^{(8)}$	$\widehat{n}^{(*)}$	$n_{\max}$	τ	λ
f_{O}	$\mathscr{D}_{\mathrm{U}}$	Φ_2^{S}	203	867	1539	1037	202	3	3851	393216	0.0098	1.3×10^{-7}
f_{O}	$\mathscr{D}_{\mathrm{N}}$	Φ_2^{S}	204	898	1375	638	108	3	3226	393216	0.0082	8.8×10^{-8}
f_{M}	$\mathscr{D}_{\mathrm{U}}$	Φ_2^{S}	276	1273	4934	8136	3367	440	18441	1572864	0.0117	2.1×10^{-7}
f_{M}	$\mathscr{D}_{\mathrm{N}}$	Φ_2^{S}	271	1255	4525	6369	2388	370	15192	1572864	0.0097	1.1×10^{-7}

5.5.1 Test function f_{O} with mother function Φ_2^{S} and smoothed uniform point distribution

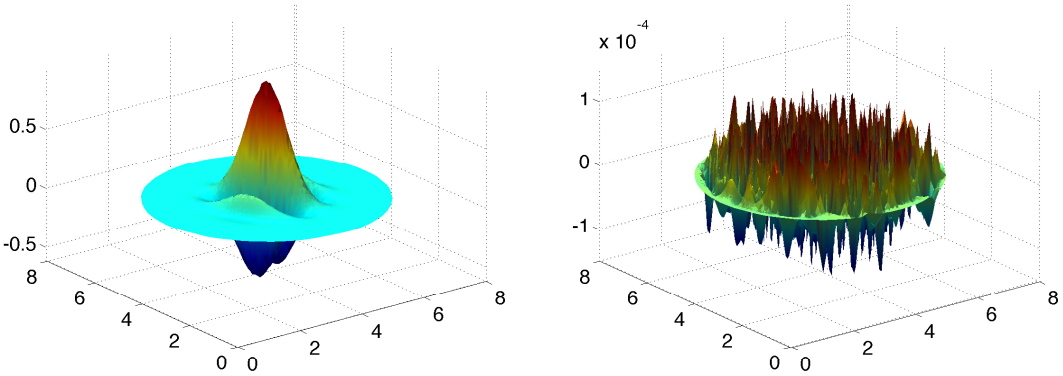


Figure 6: The final approximation s computed by our algorithm for target function $f = f_{\mathrm{O}}$, $N = 20000$ points, mother function $\Phi = \Phi_2^{\mathrm{S}}$ with $\epsilon_{\max} = 1.0 \times 10^{-4}$, $\ell_{\min} = 3$, $\ell_{\max} = 9$ (left) and the residual $e = f - s$ (right) .

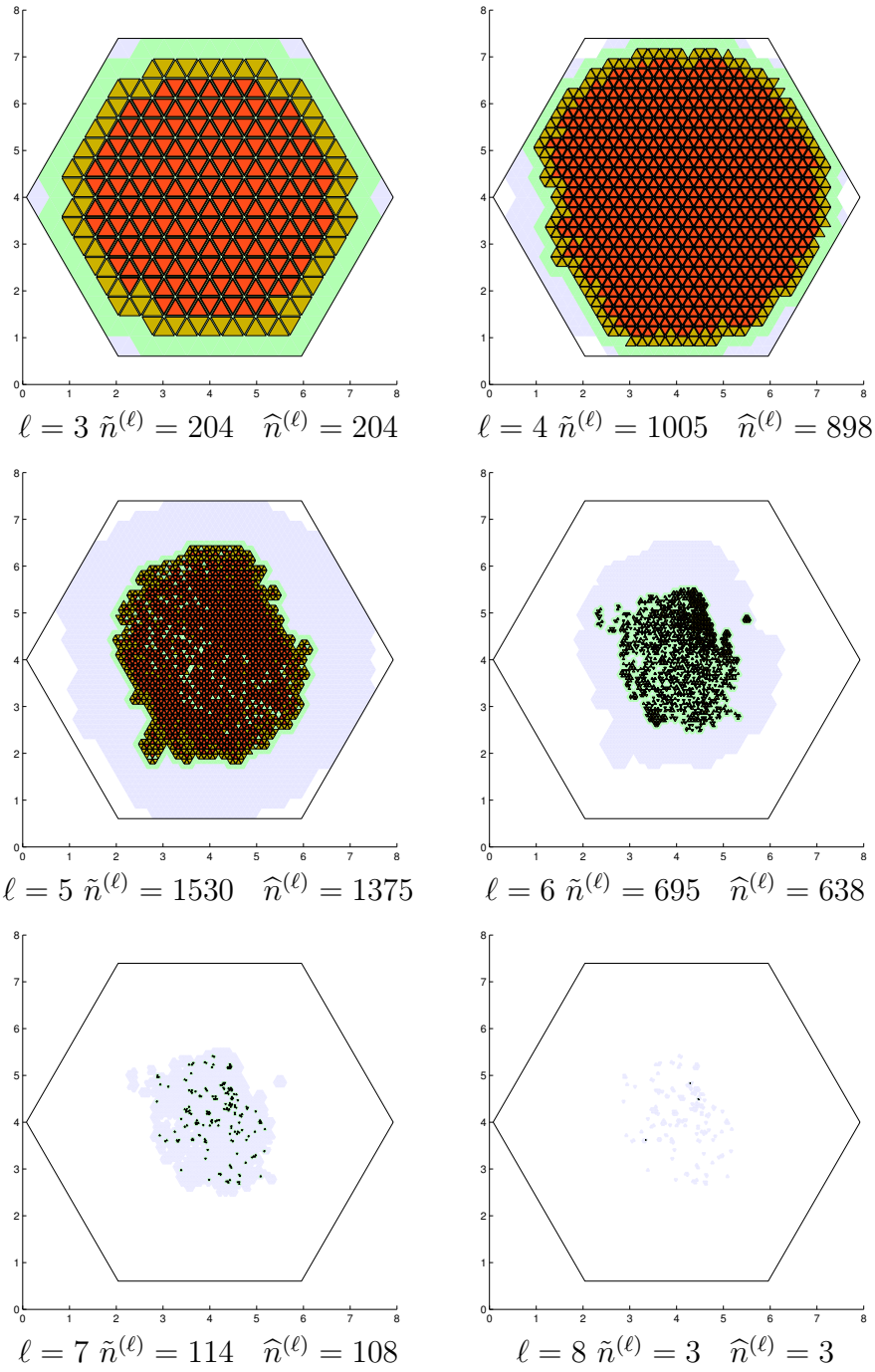


Figure 7: The cell sets $\mathfrak{C}^{(\ell)}$ (orange), $\mathfrak{R}^{(\ell)}$ (yellow), and $\mathfrak{U}^{(\ell)}$ (green) for each level ℓ in $\{\ell_{\min} .. \ell_{\max}\}$, with the same parameters as in Figure 6. The final reduced basis has $\widehat{n}^{(*)} = 3226$ elements.

6 Conclusions

We described a very general algorithm for adaptive multiscale approximation of discrete data that can be used with arbitrary domains, sampling points, meshes, and basis elements. Tests indicate that the algorithm can be very efficient in computational time and in the size of the resulting basis.

In Table 1, we observe that, for the target function f_{O}, the number $\widehat{n}^{(*)}$ of elements in the final base is much smaller than the number of sampling points ($N = 20\ 000$), and also much smaller than the number $n_{\max}$of elements of the maximum level $S^{(\ell)}$ of the full basis ($6 \times 4^{\ell_{\max}}$). Note that it would impractical to achieve this scale of resolution with a single-scale basis $S^{(\ell_{\max})}$ because that would imply solving a linear system of size $n_{\max} \times n_{\max}$. The reduction factor in basis size is the parameter τ in that table. The estimated savings in computing time are expressed by the λ factor in the table.

7 Appendix A: Definition of the test functions

In each case, the function is defined on the *natural domain* $\mathbb{U} = [-1/2, 1/2] \times [-1/2, 1/2]$ which is then mapped to the domain $\mathbb{D} = [0, L]^2$ by the formulas $X = x/L - 1/2$ and $Y = y/L - 1/2$, where (X, Y) are the natural arguments of the objective function and (x, y) a point of $\mathbb{D}$.

7.1 Test function F_{O}

Function F_{O} is a variant of Gabor's element [**?**], that is, a two-dimensional sinusoidal wave modulated by a Gaussian bell

$$F_{\mathrm{O}}(X, Y) = \exp\left(-\frac{1}{2R^2}(X^2 + Y^2) \right) \cos\Big((\alpha_x X + \alpha_y Y)\pi \Big), \tag{13}$$

where $R = 0.087$, $\alpha_x = 10$, and $\alpha_y = 5$.

7.2 Test function F_{M}

Function F_{M} is the sum of 15 bell-shaped humps with varying widths and amplitudes arranged in spiral around point (0.005, 0).

$$F_{\mathrm{M}}(X, Y) = c_i \Psi\left(\frac{\|(X, Y) - (X_i, Y_i)\|}{w_i} \right) \tag{14}$$

where $c_i = \frac{1}{2 - i/15}$, $w_i = 0.40\, r_i$ and

$$\Psi(z) = \begin{cases} 0 & \text{if} \quad z \geq 1 \\ (1 - z^2)^2 & \text{if} \quad z \leq 1 \end{cases}$$

with $r_i = 0.086 \times 2^{s_i}$, $X_i = r_i \cos(2\pi s_i) - 0.05$, and $Y_i = r_i \sin(2\pi s_i)$ where $s_i = (i - 1)/6.5$.

References

[1] Gilcélia Regiane de Souza. *Aproximação de Funções Irregularmente Amostradas com Bases Hierárquicas Adaptativas de Elementos Tensoriais Compactos.* PhD thesis, Institute of Mathematics, Statistics and Scientific Computing, UNICAMP, October 2013. Advisor: Jorge Stolfi.

[2] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 1996.

[3] S. Müller. *Adaptive Multiscale Schemes for Conservation Laws*, volume 27 of *Lectures Notes in Computational Science and Engineering.* Springer, 2003.