General Algorithms for Multiscale Approximation

Gilcélia Regiâne de Souza[1,a)] and Jorge Stolfi[2,b)]

[1]*UFSJ - Rodóvia MG 443 - km 7, Ouro Branco/MG CEP: 36420-00.*
[2]*UNICAMP - Instituto de Computação - Av. Albert Einstein, 1251 Cidade Universitria, Campinas/SP - Brasil CEP 13083-852*

[a)]gilcelia@ufsj.edu.br
[b)]stolfi@ic.unicamp.br

Abstract. 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. Our algorithm constructs the approximation from the top down, using a special least squares fitting of the residual at each level, followed by a basis reduction procedure to discard elements that contribute very little to the approximation. An important feature of these algorithms is their generality, since they are independent of domain dimension and shape, mesh, and approximation basis. Another important feature is that they do not need to generate the full basis at each level.

INTRODUCTION

In this paper we describe efficient algorithms for adaptive multiscale approximation of functions that are *heterogeneous* — 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.

We assume that the function to be approximated (the *target function*) is nominally defined on some multi-dimensional domain $\mathbb{D} \subseteq \mathbb{R}^d$, but is known only at a finite number of *sampling points* $p = (p_1, p_2, \ldots, p_n)$ 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.

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 is a subset of a full one, excluding elements that 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.

Contributions: The main contributions of this paper are three algorithms:

(a) *Analyze*: a heuristic procedure to find a subset of the domain where the target function admits a good approximation with a given basis, and said approximation;
(b) *Reduce*: a procedure exclude basis elements from an approximation without increasing the approximation error beyond a given tolerance;
(c) *HApp*: a procedure that efficiently constructs an adaptive multiscale basis, with prescribed error tolerance, in top-down fashion.

An important feature of our algorithms is that they are very general, independent of domain shape and dimension, mesh, and approximation basis. They require only very loose conditions on the mesh and base hierarchy. Another important feature is their efficiency, both in terms of computation time and final basis size, for functions that are

amenable to adaptive approximation. This efficiency is achieved by original basis reduction heuristics and by a top-down construction that uses only a subset of the full basis at each level.

The procedures *Analyze* and *Reduce* are omitted from this extended abstract for lack of space. The basic ideas of *Analyze* and a detailed description of *Reduce* are available as a technical report [1]. That report also presents some empirical tests of *HApp* with various target functions and a specific hierarchical mesh.

Related work: A general introduction to approximation theory is provided by the book of Hammerlin and Hoffmann [2]. The main algorithms (*HApp*, *Reduce*, and *Analyze*) proposed in this paper are improved versions of the algorithms developed in the Ph. D. Thesis of the first author [3]. The *HApp* procedure was inspired by the algorithms of Armin Iske and others for adaptive multiscale scattered data approximation [4, 5, 6, 7, 8], but tries to be as independent as possible from the specifics of mesh topology, basis functions, and sampling grid. It fits in the broader area of multiscale adaptive approximation [9, 10, 11, 12, 13, 14, 15, 16].

OVERVIEW OF THE METHOD

We assume that a function (or basis element) is represented as vectors in $\mathbb{R}^N$, where element i is the value of the function on sampling point p_i. The *approximation space* is a linear subspace $\mathcal{S}$ or $\mathbb{R}^N$. An *instance* of the approximation problem is then *target* vector, f in $\mathbb{R}^N$, representing the function to be approximated. A *solution* for that instance is a vector s from $\mathcal{S}$, such that the pair (f, s) satisfies some specified approximation criterion. The *residual* is the difference $e = f - s \in \mathbb{R}^N$.

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 $s = S\alpha$ where $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ is a (column) vector of n real coefficients.

We use the notations $S_{i:}$ and $S_{:j}$ for row i and column j of a matrix S. We also denote by S_{IJ} the submatrix of S consisting of the lines whith indices in the set $I \subseteq \mathbb{N}$ and the columns with indices in the set $J \subseteq \mathbb{N}$.

Multiscale approximation: Our multiscale adaptive approximation procedure *HApp* (Algorithm 1) uses a *hierarchical basis*, a sequence $S^{(*)} = (S^{(\ell_{\min})}, S^{(\ell_{\min}+1)}, \ldots, S^{(\ell_{\max})})$ of approximation bases, the *levels*, with increasing spatial resolution at each level. Each basis $S^{(\ell)}$ defines an approximation space $\mathcal{S}^{(\ell)}$.

In many multiscale approximation schemes [13], the spaces $\mathcal{S}^{(\ell)}$ and $\mathcal{S}^{(k)}$ of different levels are required to be linearly independent, or even orthogonal. We make the opposite that assumption, namely that

$$\mathcal{S}^{(\ell_{\min})} \subseteq \mathcal{S}^{(\ell_{\min}+1)} \subseteq \ldots, \subseteq \mathcal{S}^{(\ell_{\max})}). \tag{1}$$

The main advantage of using nesting (or near-nesting) spaces $\mathcal{S}^{(\ell)}$ is that accidental omission of an element from the pre-basis $\widetilde{s}^{(\ell)}$ of each level can be corrected at the subsequent levels.

Element cells and the hierarchical mesh: We assume that each element $S^{(\ell)}_{:j}$ of each basis level $S^{(\ell)}$ has an associated *cell* $K^{(\ell)}_j$, a subset of the point indices $\{1, 2, \ldots, N\}$ (that is, of the row indices of S) where that element dominates in some sense. We will denote by $\mathbf{K}^{(\ell)}$ the *mesh* of level ℓ, defined as the set of all cells of level ℓ. The list $\mathbf{K}^{(*)} = (\mathbf{K}^{(\ell_{\min})}, K^{(\ell_{\min}+1)}, \ldots, \mathbf{K}^{(\ell_{\max})})$ is the *hierarchical mesh*.

The *HApp* 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 algorithm also assumes that the cells of successive levels are properly nested; that is, for all r, j, and ℓ, cell $K^{(\ell+1)}_r$ is either disjoint of cell $K^{(\ell)}_j$, or is contained in it. The *children* of a cell $K^{(\ell)}_j$ are all the cells $K^{(\ell+1)}_r$ of the next level that are contained in it.

The *support* of a basis element $S^{(\ell)}_{:j}$ is the set $\text{supp}(S^{(\ell)}_{:j})$ of cells of level ℓ where element $S^{(\ell)}_{:j}$ is non-zero. We assume that the cell $K^{(\ell)}_j$, in particular, belongs to $\text{supp}(S^{(\ell)}_{:j})$.

THE MULTISCALE APPROXIMATION ALGORITHM

Our hierarchical approximation algorithm is described more precisely by the procedure *HApp* below.

In typical situations, constructing the full hierarchical basis $S^{(*)}$ would be probabively expensive. Therefore, *HApp* performs a sequence of single-scale approximations, one per level, starting at each level with a *pre-basis* $\widetilde{S}^{(\ell)}$, a

subset of the full basis $S^{(\ell)}$, that includes only those elements that cover those parts of the domain where the uniform norm of that residual $f^{(\ell)}$ is still too large.

At each level, *HApp* finds a *small* basis $\widehat{S}^{(\ell)} \subseteq \widetilde{S}^{(\ell)}$ that provides a sufficiently good approximation $\widehat{s}^{(\ell)}$ to the level's target function $f^{(\ell)}$, at least over the part of the domain in where such a good approximation exists. The residual $f^{(\ell)} - \widehat{s}^{(\ell)}$ of that approximation becomes the target $f^{(\ell+1)}$ at the next finer scale. Thus, the approximations $\widehat{s}^{(\ell)}$ obtained at all levels must be added together to obtain the final approximation $\widehat{s}$.

Ideally, $\widehat{S}^{(\ell)}$ should be the smallest subset of $\widetilde{S}^{(\ell)}$ that provides the desired accuracy. That goal would be too expensive to achieve, however; so *HApp* uses a heuristic method that is not overly expensive and has been shown to produce reasonably small bases, even if not the minimum ones. Namely, it first computes an initial approximation $\widetilde{s}^{(\ell)}$ in the space of the pre-basis $\widetilde{S}^{(\ell)}$, by the auxiliary procedure *Analyze*, a robust non-linear variant of the least-squares method that ignores outlier values and regions of the domain where the function cannot be well approximated. It then uses the auxiliary procedure *Reduce* to remove from the basis $\widetilde{S}^{(\ell)}$ all elements that make a negligible contribution to $\widetilde{s}^{(\ell)}$, obtaining a *reduced basis* matrix $\widehat{S}^{(\ell)}$ and the corresponding *reduced coefficient vector* $\widehat{\alpha}^{(\ell)}$.

Besides the sampled target function f (a vector of $\mathbb{R}^N$), the indices $\ell_{\min}, \ell_{\max}$ of the first and last basis levels to use, and the error tolerance $\epsilon_{\max} > 0$, the *HApp* algorithm requires the following procedural parameters:

- *GetAllCells*(ℓ) that returns the indices of all cells (that is, elements of the full basis) of a given level ℓ;
- *GetChildrenCells*(ℓ, j) that returns a list of the indices r all children cells $K_r^{(\ell+1)}$ of a given cell $K_j^{(\ell)}$;
- *GetElement*(ℓ, j) that computes the element $S_{:j}^{(\ell)}$ of the discrete basis $S^{(\ell)}$;
- *GetSupportCells*(ℓ, j) that returns the support of element $S_{:j}^{(\ell)}$, that is, a list of all indices r of the cells $K_r^{(\ell)}$ of level ℓ that contain at least one sampling point where element $S_{:j}^{(\ell)}$ is nonzero; and
- *GetRelevantElements*(ℓ, j) that returns the list of all indices r of elements $S_{:r}^{(\ell)}$ of level ℓ whose support includes the cell $K_j^{(\ell)}$.

The procedures *GetAllCells*, *GetChildrenCells*, *GetElement*, *GetSupportCells*, and *GetRelevantElements* depend on the application; specifically on the mesh hierarchy and on the hierarchical basis. If the cells in each level are a regular tessellation of $\mathbb{D}$, the arguments *GetSupportCells* and *GetRelevantElements* are usually the same procedure.

The outputs of *HApp* are the side-by-side concatenation $\widehat{S}^{(*)} = \widehat{S}^{(\ell_{\min})} \,\text{⨪}\, \widehat{S}^{(\ell_{\min}+1)} \,\text{⨪}\, \widehat{S}^{(\ell_{\max})}$ of the reduced bases $\widehat{S}^{(\ell)}$; the-top-to bottom concatenation $\widehat{\alpha}^{(*)} = \widehat{\alpha}^{(\ell_{\min})} \,\text{⨧}\, \widehat{\alpha}^{(\ell_{\min}+1)} \,\text{⨧}\, \cdots \,\text{⨧}\, \widehat{\alpha}^{(\ell_{\max})}$ of the corresponding coefficient vectors; and the computed approximation $\widehat{s}^{(*)} = \widehat{S}^{(*)}\widehat{\alpha}^{(*)}$.

In steps 7 and 8 of each iteration, this algorithm identifies the set $\mathbf{C}^{(\ell)}$ of *critical* cells, where the current residual $f^{(\ell)}$ exceeds the tolerance $\epsilon_{\max}$. Those cells must be children of a certain set $\mathbf{M}^{(\ell-1)}$ of *modifiable* cells identified in the previous iteration. In step 10 the algorithm builds the set $\mathbf{B}^{(\ell)}$ of all *basis cells* whose elements have supports that intersect cells of $\mathbf{C}^{(\ell)}$. In step 14, the algorithm builds a pre-basis $\widetilde{S}^{(\ell)}$ with the elements of the full basis $S^{(\ell)}$ associated to each basis cell. Note that the support of each element may overlap other cells, but is contained in $\bigcup \mathbf{M}^{(\ell)}$.

In steps 15– 18, procedures *Analyze* and *Reduce* are used to find a reduced approximation of the current target funtion $f^{(\ell)}$, which is added to the partial approximation $\widehat{s}^{(\ell)}$. The new residual $\widehat{e}^{(\ell)} = f^{(\ell)} - \widehat{s}^{(\ell)}$ will be the target for the next iteration.

Observe that $\mathbf{C}^{(\ell)} \subseteq \mathbf{B}^{(\ell)} \subseteq \mathbf{M}^{(\ell)} \subseteq \mathbf{U}^{(\ell)}$, for each level $\ell \geq \ell_{\min}$. It can be proved by induction on ℓ that, for every sampling point p_i, (1) if $p_i \notin \bigcup \mathbf{M}^{(\ell-1)}$, then $\left|e_i^{(\ell)}\right| \leq \epsilon_{\max}$; (2) if $p_i \notin \bigcup \mathbf{M}^{(\ell)}$, then $\widehat{s}_i^{(\ell)} = 0$; (3) $\bigcup \mathbf{U}^{(\ell)} \subseteq \bigcup \mathbf{U}^{(\ell-1)}$. (The sets $\mathbf{U}^{(\ell)}$ are defined only for the purpose of analysis and correcness, and may be omitted in actual implementations.)

The algorithm may end before reaching level $\ell_{\max}$, if the set of critical cells $\mathbf{C}^{(\ell)}$ becomes empty (step 9); in that case, we will have $\left\|f^{(\ell)}\right\|_\infty \leq \epsilon_{\max}$, as desired. If the full space $S^{(\ell_{\max})}$ at the last level can interpolate any target function, then $\left\|f^{(\ell)}\right\|_\infty \leq \epsilon_{\max}$ will hold in any case.

Algorithm 1 ***Procedure*** *HApp*$(f, \ell_{\min}, \ell_{\max}, \epsilon_{\max},$ *GetAllCells*, *GetChildrenCells*, *GetElement*, *GetSupportCells*, *GetRelevantElements*)

1. $\mathbf{M}^{(\ell_{\min}-1)} \leftarrow$ *GetAllCells*$(\ell_{\min} - 1)$;
2. $\mathbf{U}^{(\ell_{\min}-1)} \leftarrow \mathbf{M}^{(\ell_{\min}-1)}$;
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. $\mathbf{C}^{(\ell)} \leftarrow \bigcup_{j \in \mathbf{M}^{(\ell_{\min}-1)}}$ *GetChildrenCells*$(\ell - 1, j)$;
   8. Remove from $\mathbf{C}^{(\ell)}$ every cell X such that $\left|f_i^{(\ell)}\right| \le \epsilon_{\max}$ for all $i \in X$.
   9. If $\mathbf{C}^{(\ell)} = \{\}$, go to step 19.
   10. $\mathbf{B}^{(\ell)} = \bigcup_{j \in \mathbf{C}^{(\ell)}}$ *GetRelevantElements*(ℓ, j)
   11. $\mathbf{M}^{(\ell)} \leftarrow \bigcup_{j \in \mathbf{B}^{(\ell)}}$ *GetSupportCells*(ℓ, j)
   12. $\mathbf{U}^{(\ell)} \leftarrow \bigcup_{j \in \mathbf{M}^{(\ell)}}$ *GetRelevantElements*(ℓ, j);
   13. $\mathbf{U}^{(\ell)} \leftarrow \bigcup_{j \in \mathbf{U}^{(\ell)}}$ *GetSupportCells*(ℓ, j);
   14. For each $j \in \mathbf{C}^{(\ell)}$, set $\widetilde{S}^{(\ell)}_{:j} \leftarrow$ *GetElement*(ℓ, j).
   15. $(\widetilde{s}^{(\ell)}, \widetilde{\alpha}^{(\ell)}) \leftarrow$ *Analyze*$(f^{(\ell)}, \widetilde{S}^{(\ell)}, \epsilon_{\max})$
   16. $(\widehat{S}^{(\ell)}, \widehat{\alpha}^{(\ell)}) \leftarrow$ *Reduce*$(\widetilde{S}^{(\ell)}, \widetilde{\alpha}^{(\ell)}, \widetilde{e}^{(\ell)}, \epsilon_{\max})$;
   17. $\widehat{s}^{(\ell)} \leftarrow \sum_k \widehat{\alpha}^{(\ell)}_k \widehat{S}^{(\ell)}_{:k}$; $\widehat{e}^{(\ell)} \leftarrow f^{(\ell)} - \widehat{s}^{(\ell)}$.
   18. $\widehat{S}^{(*)} \leftarrow \widehat{S}^{(*)} \mathbin{⫲} \widehat{S}^{(\ell)}$; $\widehat{\alpha}^{(*)} \leftarrow \widehat{\alpha}^{(*)} \mathbin{⫡} \widehat{\alpha}^{(\ell)}$; $\widehat{s}^{(*)} \leftarrow \widehat{s}^{(*)} + \widehat{s}^{(\ell)}$.
19. Return $\widehat{S}^{(*)}$, $\widehat{\alpha}^{(*)}$, $\widehat{s}^{(*)}$.

COMPUTATIONAL COMPLEXITY

Note that only a subset $\widetilde{S}^{(\ell)}$ of the full basis $S^{(\ell)}$ is computed in step 14 of *HApp*; namely, only the columns j whose cells are in $\mathbf{B}^{(\ell)}$, and only for those points p_i such that $i \in \bigcup \mathbf{M}^{(\ell-1)}$). These algorithms are most efficient (both in the running time and in the size of the final adaptive basis) if the basis elements have small support.

Significant savings in computing time are possible, 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. Then only one column of the basis matrix needs to be computed, since the other columns will be just permutations of it. Ditto for the matrix M of the least squares method. [3].

The total cost of *HApp* will be dominated by the cost of *Analyze* in each level, which is at most proportional to $\widetilde{t}^{(\ell)} = (\widetilde{n}^{(\ell)})^2 \log(\widetilde{n}^{(ell)})(\widetilde{q}^{(\ell)} + \widetilde{n}^{(\ell)})$, where $\widetilde{q}^{(\ell)}$ is the average number of sampling points in the support of each element of the pre basis $\widetilde{S}^{(\ell)}$. The total time then is at most proportional to $\widetilde{t}^{(*)} = \sum_{\ell=\ell_{\min}}^{\ell_{\max}} \widetilde{t}^{(\ell)}$.

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 $S^{(\ell_{\max})}$. For target functions that are amenable to adaptive multiscale approximation, this cost can be thousands of times the cost of *HApp*.

ACKNOWLEDGMENTS

This research was partially supported by CNPq (grant 310706/2015-7).

REFERENCES

[1] G. R. de Souza and J. Stolfi, "Adaptive multiscale function approximation I: General discrete bases," Tech. Rep. IC-15-08 (Institute of Computing, UNICAMP, 2016).

[2] G. Hammerlin and K. H. Hoffmann, *Numerical Mathematics* (Springer, 1991).

[3] G. R. de Souza, "Aproximação de funções irregularmente amostradas com bases hierárquicas adaptativas de elementos tensoriais compactos," Ph.D. thesis, Institute of Mathematics, Statistics and Scientific Computing (IMECC), State University of Campinas (UNICAMP), BrazilOctober 2013, advisor: Jorge Stolfi.

[4] A. Iske, *Multiresolution Methods in Scattered Data Modelling* (Springer, 2000).

[5] A. Iske and J. Levesley, Numerical Algorithms **39**, 187–198 (2005).

[6] M. S. Floater and A. Iske, Journal of Computational and Applied Mathematics **73**, 65–78 (1996).

[7] A. Iske, "Hierarchical scattered data filtering for multilevel interpolation schemes," in *Proc. of the 2000 Conference on Mathematical Methods for Curves and Surfaces*, edited by T. Lyche and L. L. Schumaker (2000), pp. 211–220.

[8] M. S. Floater and A. Iske, "Multistep scattered data interpolation using compactly supported radial basis functions," in *Proc. of the 2000 Conference on Mathematical Methods for Curves and Surfaces*, edited by T. Lyche and L. L. Schumaker (2000), pp. 211–220.

[9] F. J. Narcowich, R. Schaback, and J. D. Ward, Approximations Theory **7**, 1–9 (1995).

[10] G. E. Fasshauer, Interdisciplinary Mathematical Sciences **6** (2007).

[11] M. K. Kaibara, *Análise de Multiresolução para Leis de Conservação em Malhas Adaptativas*, Master's thesis, Institute of Mathematics, Statistics, and Scientific Computing (IMECC), State University of Campinas (UNICAMP), BrazilMarch (2000).

[12] S. Lee, G. Wolberg, and S. Y. Shin, IEEE Transactions on Visualization and Computer Graphics **3**, 228–244 (1997).

[13] S. Müller, *Adaptive Multiscale Schemes for Conservation Laws*, Lectures Notes in Computational Science and Engineering, Vol. 27 (Springer, 2003).

[14] D. A. Castro, *Multilevel Approximation Schemes and Applications*, Master's thesis, Institute of Mathematics, Statistics, and Scientific Computing (IMECC), State University of Campinas (UNICAMP), BrazilDecember (2011).

[15] C. G. S. Cardoso, M. C. Cunha, A. Gomide, D. J. Schiozer, and J. Stolfi, Mathematics and Computers in Simulation **73**, 87–104 (2006).

[16] M. O. Domingues, S. M. Gomes, O. Roussel, and K. Schneider, ESAIM Proceedings **34**, 1–98 (2011).