Scalable, High-Order Continuity Across Block Boundaries of Functional Approximations Computed in Parallel

1st Iulian Grindeanu
Argonne National Laboratory
Lemont, IL, USA
iulian@mcs.anl.gov

2nd Tom Peterka
Argonne National Laboratory
Lemont, IL, USA
tpeterka@mcs.anl.gov

3rd Vijay S. Mahadevan
Argonne National Laboratory
Lemont, IL, USA
mahadevan@anl.gov

4th Youssef S. G. Nashed
Argonne National Laboratory
Lemont, IL, USA
ynashed@anl.gov

***Abstract*—We investigate the representation of discrete scientific data with a C^k functional model, where C^k denotes k-th order continuity, in a distributed-memory parallel setting. The Multivariate Functional Approximation (MFA) model is a piecewise-continuous functional approximation based on multivariate high-dimensional B-splines. When computing an MFA approximation in parallel over multiple blocks in a spatial domain decomposition, the interior of each block will be C^k, k being the B-spline polynomial degree, but discontinuities exist across neighboring block boundaries. We present an efficient and scalable solution that involves blending neighboring approximations to ensure C^k continuity across block boundaries. We show that after decomposing the domain in structured, overlapping blocks and approximating blocks independently to high degrees of accuracy, we can extend the local solution, in a postprocessing step, to the global domain by using compact, multidimensional smoothstep functions. We prove that this approach, which can be viewed as an extended partition of unity approximation method, is scalable on high-performance computing architectures.**

***Index Terms*—Multivariate functional approximation, B-splines, NURBS modeling, blending, smoothstep**

I. Introduction

This paper builds on the ongoing investigation in redefining discrete scientific data by an alternative representation, the *Multivariate Functional Approximation (MFA)*. By approximating raw discrete data with a hypervolume of piecewise-continuous functions, the MFA model can represent scientific data from simulations, experiments, analyses, and visualizations. Compared with existing discrete data models, the MFA model is designed to enable most spatiotemporal analyses without reverting back to the original discrete form, usually while occupying less storage space.

In [1], the authors showed an effective way of scaling and parallelizing the MFA model, using block decomposition of the global domain, but they left open the problem of continuity along block boundaries. In this follow-up paper, we offer a solution to this problem by extending local block approximations to the global domain, while maintaining desired error levels and degree of continuity. We also demonstrate that our solution scales well on existing high-performance computing architectures.

Our method is applicable to very large datasets for which a global continuous and differentiable approximation is needed. In our divide-and-conquer approach, locally computed parallel approximations are combined in a systematic way to produce a global continuous approximation.

Section II summarizes related work in functional data analysis, low-rank approximations, and applications of functional data representations. Section III reviews the approach used in [1] with respect to NURBS-based approximations and block decompositions, needed for scaling and parallelization. Section IV explains the novel methodology used to ensure C^k continuity across blocks, which is the main contribution of this paper. Section V evaluates our model on synthetic and actual test data. Section VI summarizes our work and outlines the next research topics we intend to pursue.

II. Related Work

Multivariate functional approximation borrows ideas from several fields: compression, statistics, modeling, visualization, and analysis. Piecewise functional approximations replace discrete data points with linear combinations of basis functions and a small number of reference points called *control points*. Statisticians call this method *functional data analysis* [2], [3], with low-dimensional serial implementations available in popular statistics packages [4].

A key decision in functional data analysis is the choice of the basis function family. Fourier [5], wavelet [6], [7], geometric (i.e., spline) [8], and radial basis functions [9] are commonly used. We use geometric (NURBS) bases for the MFA because they mirror the space-time properties in the original data and retain these geometric properties in analytics and visualization. NURBS are commonly used in computer-aided design, manufacturing, and engineering and are part of numerous industry-wide standards. Computing geometric bases requires minimal memory overhead, and the parallel decomposition of the original space-time domain is identical in the geometric functional domain, an important consideration for minimizing data movement at scale.

Partitions of unity are useful because they allow local constructions to extend to the entire domain of interest. Partition

U.S. Government work not protected by U.S. copyright

of unity methods [10] are used in [11] and [12] to compute interpolants on scattered data lying in convex domains, through a block-based searching technique or through a kd-tree space partitioning structure, by blending radial basis functions as local approximants.

Sederberg et al. [13] introduced T-splines to ensure continuity along B-spline patches, by merging two surfaces with different knot vectors into a single T-spline. Compared with NURBS, modeling surfaces with T-splines can reduce the number of control points and make pieces easier to merge, but T-splines increase the bookkeeping effort to keep track of the irregular connectivity. In order to smoothly join points where more than three surface patches meet, T-splines have been combined with geometrically continuous constructions of degree 3 by 3 (bicubic) and degree 4 by 4 (biquartic).

III. Multivariate Functional Approximation

NURBS, in addition to being able to model complex data with a small number of control points, are piecewise-continuous, are differentiable to degree k, have local support, and are invariant to affine transformations. To compute the MFA in d dimensions, we use a tensor(d) product of NURBS bases in 1-d. We approximate field geometry (space-time vertex positions) and science variables (pressure, density, temperature, etc.) in separate MFA models, which can have different polynomial degrees, knots, and control points. The knots are scalars in the range $[0.0, 1.0]$ that map partitions of the data to the control points and basis functions.

As in [1], we assume the input data points lie in a tensor product of dimension d. The tensor product is topologically a high-dimensional rectangular grid. However, the physical mesh does not have to be strictly regular as long as there exists a one-to-one mapping of physical coordinates $(x, y, z, ...)$ to $(u, v, w, ...)$ parameters. The experiments for this paper assume that we use regular datasets, assuming the entire domain to be one input patch.

We first decompose the input data in structured blocks, aligned with the coordinate axes. Blocks are extended with halo (ghost) regions, so that they are overlapping in the computational domain by uniform amounts. We approximate within each extended block independently; the result is a structured set of block approximations that will be blended at evaluation time.

Within each block, because the input points in our experiments align with the principal axes, we assign input point parameters according to the spacing of input points in each of the principal domain dimensions.

Block parallelism is used to decompose the domain into blocks and execute each block in distributed-memory compute nodes of a supercomputer or computing cluster. We use DIY [14] to accomplish block parallelism. We use DIY's framework for communication between blocks, for exchanging shared points and ghost regions for input data, for exchanging the evaluated points of the overlapping blocks, and for computing global error measures.

As in [1], each block of the parallel data model is a tensor product defined by n control points and $n+k+1$ knots in each dimension. Control points are in the same coordinate system as original data points, meaning that the same spatiotemporal domain decomposition used for input data can be reused for the MFA. The basis functions are not stored and are recomputed as needed.

The output MFA is stored in a binary file in DIY format, which is read and written in parallel using MPI-I/O. A serial utility converts the file to VTK format so that the results are compatible with visualization and analysis tools derived from VTK such as ParaView [15] and VisIt [16].

The MFA is designed to be used for subsequent data analysis and visualization, distinguishing it from other basis representations such as wavelets, cosine transformations, and compression algorithms that require the inverse transform to first be applied. Once the model is computed, one can evaluate the model at any point in the interior of any block, not just at the original input points, and differentiate the model up to the k-th derivative in any combination of partial derivatives in the domain dimensions [17].

IV. Continuity Across Blocks

To ensure continuity across blocks, we devise a novel method in which we combine neighboring block MFA solutions, using multidimensional smoothstep functions. After we decompose the global domain in structured blocks, we enlarge each block with a ghost region that covers a fraction of neighboring blocks, and we compute an MFA approximation on each extended block. In the overlapping regions, we blend overlapping local solutions with smoothstep functions of desired continuity that are guaranteed to not increase local approximation error. The blending is done during the evaluation of points from the MFA. The new blended global solution is guaranteed to have the same error as the independently computed local approximations.

A. Continuity in 1-D

We begin by explaining how to achieve continuity in 1-d; we will extend it to higher dimensions later. We need a function of continuity C^k, which transitions from 0.0 to 1.0 over a closed interval [0, 1]. After scaling and translating, this function will allow for a smooth transition between blocks (intervals in 1-d). Many C^k functions have the desired properties; simple polynomials can be determined by imposing null derivatives at 0 and 1, up to the kth derivative.

$$\mathrm{S}_k(x) = x^{k+1} \sum_{n=0}^{k} \binom{n+k}{n} \binom{2k+1}{k-n} (-x)^n \ , \ x \in [0, 1] \tag{1}$$

Equation (1) defines a family of smoothstep functions, and Figure 1 shows smoothstep functions of continuity C^k and their first derivatives, where k is 0, 1, 2, or 3. These functions are simple to evaluate, since they involve only polynomials of degree $2k + 1$, and are used in computer graphics and

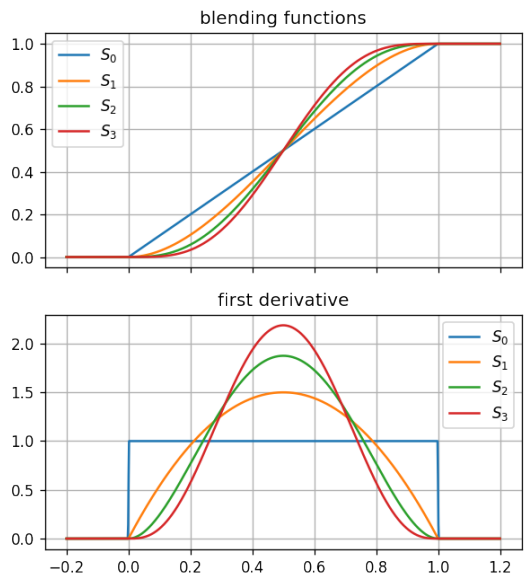

Fig. 1. Smoothstep functions S_0, S_1, S_2, S_3

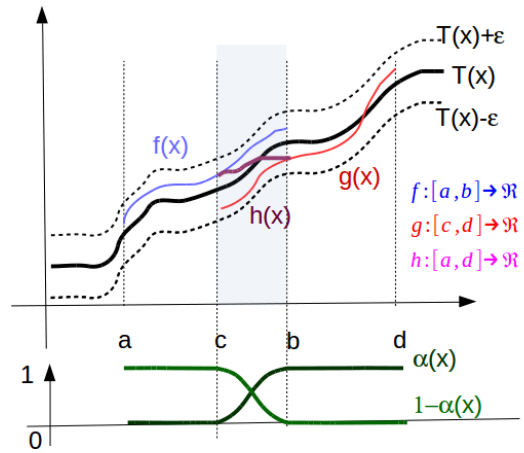

Fig. 2. Overlapping local approximations by blending

visualizations [18]. We note that the range of first derivatives of these functions increases with increasing k. In other words, as k increases, the magnitude of the derivatives increases, too.

Theorem 1 Let $f(x)$ and $g(x)$ be two approximations for a given function $T(x)$ on overlapping intervals, as $f : [a, b] \to \mathbb{R}$, $g : [c, d] \to \mathbb{R}$, with $a < c < b < d$.

Define a blending combination with

$$h(x) = (1 - \alpha(x))f(x) + \alpha(x)g(x) \tag{2}$$

where $\alpha(x)$ is the smoothstep function on the overlapping interval $[c, b]$, such as

$$\alpha(x) = \begin{cases} 0 & \text{if } x < c \\ S_k(\frac{x-c}{b-c}) & \text{if } c \le x \le b \\ 1 & \text{if } b < x \end{cases} \tag{3}$$

where S_k is given by Equation (1). Then $h(x)$ is defined everywhere over the interval $[a, d]$, is C^k continuous, and the approximation given by $h(x)$ is as good as the local approximations $f(x)$ and $g(x)$.

Proof: Let ϵ be the approximation error for f and g, with respect to input function T. In other words, $|f(x)-T(x)| < \epsilon$ for $a \le x \le b$, and $|g(x) - T(x)| < \epsilon$ for $c \le x \le d$. If both $f(x)$ and $g(x)$ are within ϵ from the test function $T(x)$, we prove that $h(x)$ will be within ϵ from $T(x)$.

Let $\alpha(x)$ be the smoothstep function over $[c, b]$, so $\alpha(x) = 0$ over $[a, c]$, while $1 - \alpha(x) = 0$ over $[b, d]$. The linear combination $h(x)$ is correctly defined everywhere in the interval $[a, d]$ (see Figure 2).

On the overlapping interval $[c, b]$, we can bound the error for the blended function $h(x)$ as follows.

$$\begin{aligned} h(x) - T(x) =& (1 - \alpha(x))f(x) + \alpha(x)g(x) - T(x) \\ =& (1 - \alpha(x))(f(x) - T(x)) + \alpha(x)(g(x) - T(x)) \end{aligned}$$

Using the triangle inequality, we have the following.

$$\begin{aligned} |h(x) - T(x)| \le& |(1 - \alpha(x))(f(x) - T(x))| \\ &+ |\alpha(x)(g(x) - T(x))| \end{aligned}$$

Next, $(1 - \alpha(x)) \ge 0$, and $\alpha(x) \ge 0$ implies that

$$\begin{aligned} |(1 - \alpha(x)) \cdot (f(x) - T(x))| =& |1 - \alpha(x)| \cdot |(f(x) - T(x))| \\ \le& |1 - \alpha(x)| \cdot \epsilon \end{aligned}$$

and

$$\begin{aligned} |\alpha(x)(g(x) - T(x))| =& |\alpha(x)| \cdot |g(x) - T(x)| \\ \le& |\alpha(x)|\epsilon. \end{aligned}$$

It follows that

$$|h(x) - T(x)| \le |1 - \alpha(x)| \cdot \epsilon + |\alpha(x)| \cdot \epsilon = \epsilon.$$

We must also prove that if the approximations f, g and the smoothstep function are C^k continuous, then the blending approximation $h(x)$ is C^k continuous. We apply the chain rule of differentiation on Equation (2) and the fact that the derivatives of degree 1 to k at the endpoints of the interval $[c, b]$ are 0. For example, consider the first derivative of $h(x)$ at the interval point c. The first derivative of $h(x)$ from the left at c is $\lim_{x\to c^-} h'(x) = \lim_{x\to c^-} f'(x) = f'(c)$, while the derivative from the right is

$$\begin{aligned} \lim_{x\to c^+} h'(x) =& \lim_{x\to c^+} (f'(x)(1 - \alpha(x)) + f(x)(-\alpha'(x))) \\ &+ \lim_{x\to c^+} (g'(x)\alpha(x) + g(x)\alpha'(x)) \\ =& f'(c). \end{aligned}$$

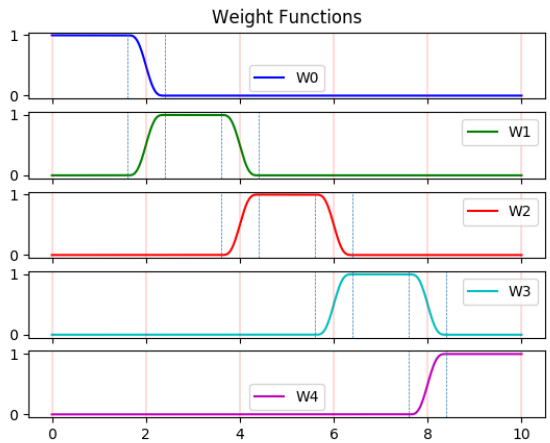

Fig. 3. Example of weight functions in a partition of unity

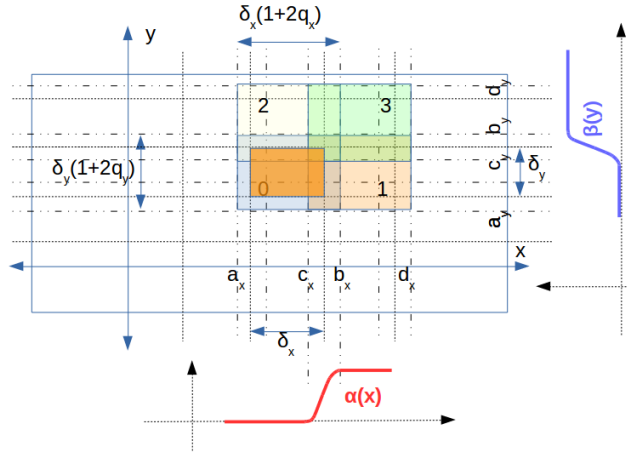

Fig. 4. Domain decomposition, overlappingi, and blending in 2d

As $\lim_{x\to c^+}(1-\alpha(x)) = 1$, $\lim_{x\to c^+}(-\alpha'(x)) = 0$, $\lim_{x\to c^+}\alpha(x) = 0$ and $\lim_{x\to c^+}\alpha'(x) = 0$, it follows that $\lim_{x\to c^+} h'(x) = \lim_{x\to c^+} f'(x) = f'(c)$. Hence $h'(x)$ is continuous at c. Similarly, first derivative continuity can be proved at the other end of the interval, b. Higher-degree derivative continuity is proved in the same way, since the smoothstep function derivatives up to order k are 0 at the interval ends c and b. ■

These properties are important in defining the approximations globally from the local approximations. We first divide the 1-d domain into equal intervals, corresponding to 1-d blocks. Then, we extend by ghosting each interval, so that each block will cover a fraction of neighboring blocks. To make bookkeeping easier, we use the same fraction of overlap between blocks.

The approximation for each extended block can be computed independently in parallel. Then local solutions are blended according to Equation (2), ensuring continuity for the global solution across blocks. Local approximations in extended blocks are blended by using a partition of unity type of formula

$$H(x) = \sum_{i=0}^{n} W_i(x) f_i(x), \tag{4}$$

in which $W_i(x)$ are weights with compact support that satisfy partition of unity constraint $\sum_{i=0}^{n} W_i(x) = 1$, $f_i(x)$ are local approximations, and $H(x)$ is a global approximation. Figure 3 shows weight functions corresponding to a decomposition of the interval $[0, 10]$ into five subintervals, with an overlap of 0.3 per side. Note that the weight functions are constant (0 or 1) except for the overlapped subintervals between blocks.

B. Continuity in Higher Dimensions

Next, we show the construction in 2-d, but the same ideas can be applied inductively for higher dimensions.

Figure 4 shows how to blend in 2-d. Basically, blending occurs independently per direction. Different blending functions and different overlap ratios can be used in each direction. At various points in the domain, the resulting global approximation is a combination of 1, 2, or 4 local approximations, if the blocks are congruent, and their overlap ratios are less than 0.5. The only requirement is that the extended blocks completely cover the domain.

Consider the four overlapping blocks in Figure 4. $F_i(x, y)$ are their local approximations, with $i = 0, ..., 3$. Each of the blocks has initial size $\delta_x\delta_y$ and an extended size $(\delta_x + 2q_x)(\delta_y + 2q_y)$. We define blending smoothstep functions in the x and y directions, $\alpha(x)$ and $\beta(y)$, with blending intervals of size $2q_x$ and $2q_y$. The blending functions $\alpha(x)$ and $\beta(y)$ vary from 0 to 1, and they are C^k continuous. For example, extended block 0 has limits defined by $[a_x, b_x] \times [a_y, b_y]$, while extended block 1 has limits $[c_x, d_x] \times [a_y, b_y]$, where $a_x < c_x < b_x < d_x$, and $a_y < c_y < b_y < d_y$. We define the blending expression in 2-d as follows: First, in the x direction, blend blocks 0 and 1 and also blocks 2 and 3:

$$H_{01}(x, y) = F_0(x, y)(1 - \alpha(x)) + F_1(x, y)\alpha(x) \tag{5}$$

$$H_{23}(x, y) = F_2(x, y)(1 - \alpha(x)) + F_3(x, y)\alpha(x). \tag{6}$$

Next, blend in the y direction:

$$H(x, y) = H_{01}(x, y)(1 - \beta(y)) + H_{23}(x, y)\beta(y). \tag{7}$$

The end result is a function defined in the domain covered by the four blocks.

$$\begin{aligned} H(x, y) = & F_0(x, y)(1 - \alpha(x))(1 - \beta(y)) + \\ & F_1(x, y)\alpha(x)(1 - \beta(y)) + \\ & F_2(x, y)(1 - \alpha(x))\beta(y) + \\ & F_3(x, y)\alpha(x)\beta(y) \end{aligned} \tag{8}$$

We note that Equation (8) can be interpreted as a partition of unity on the domain covered by four blocks, with the weight factors $(1-\alpha(x))(1-\beta(y))$, $\alpha(x)(1-\beta(y))$, $(1-\alpha(x))\beta(y)$, and $\alpha(x)\beta(y)$. A 2-d weight function corresponding to an extended 2-d block is a product of 1-d weight functions. An example is shown in Figure 5.

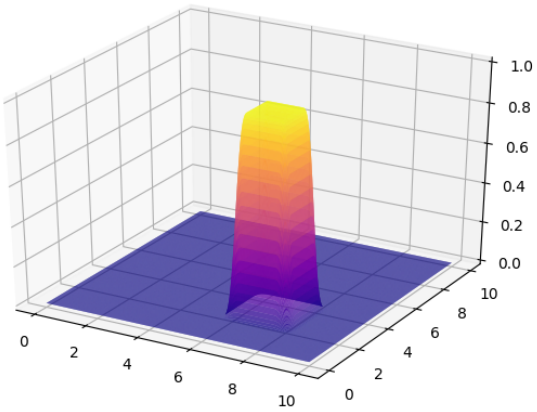

Fig. 5. Example of a 2-d weight function

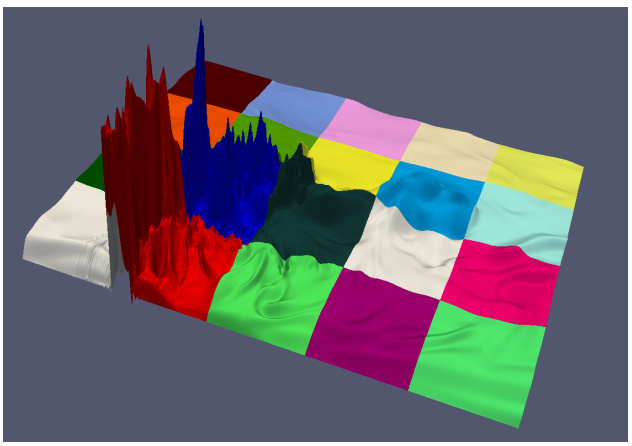

Fig. 6. Block independent approximations

Continuity and differentiability can be proved in higher-dimensional cases. One may interpret Equations (5) and (6) as an application of Theorem 1, when y is kept fixed. In other words, the theorem can be applied for $H_{01}(x, y)$ and $H_{23}(x, y)$ as functions of x only. So, $H_{01}(x, y)$ and $H_{23}(x, y)$ are C^k with respect to x. Because $F_0(x, y)$ and $F_1(x, y)$ are C^k with respect to y, we can conclude that $H_{01}(x, y)$ and $H_{23}(x, y)$ are C^k-continuous with respect to both variables. Applying once again Theorem 1 for Equation (7), with y as primary variable, we conclude that $H(x, y)$ is C^k in y. So $H(x, y)$ is C^k in both variables.

For example, Figure 6 shows a 2-d rectangular domain decomposition in 20 structured blocks. This is a 2-d slice of a 3-d dataset that will be used for experimental results in Section V. Local approximations are C^k, and they overlap across the boundary lines, as in Figure 7. Each approximation is then blended with its neighbors, resulting in a smooth C^k global approximation, as in Figure 8, which shows a zoomed-in detail of a transition area between neighboring blocks.

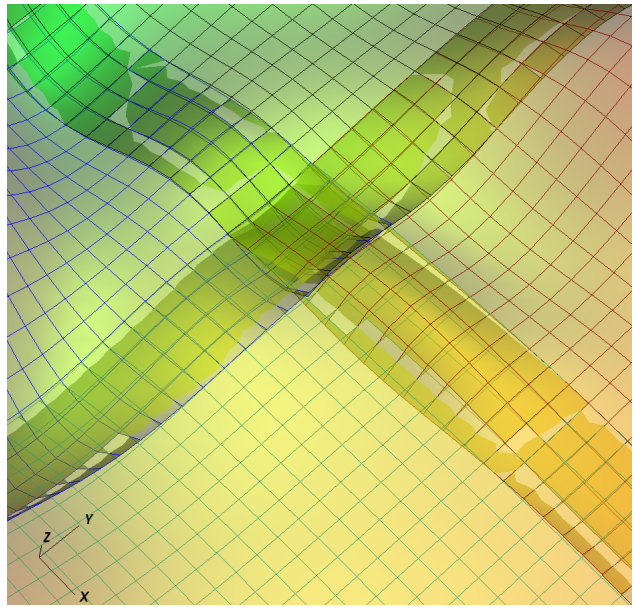

Fig. 7. Detail of overlapping approximations over 4 blocks

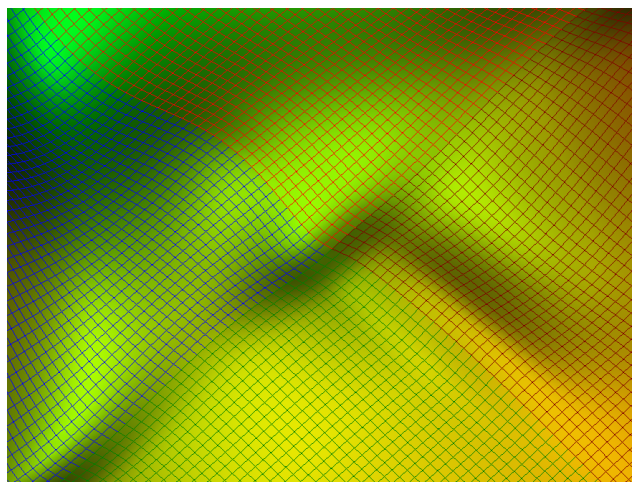

Fig. 8. Blended detail over 4 blocks

C. Blending Block Approximations

Algorithm 1 shows the main steps for blending. Throughout, we are using DIY [14] for block decomposition, computation of the overlap domains, ghost layer, and for communication between blocks. Global domain $\mathbf{D}$ is decomposed into blocks B_i, which are then extended by a fraction to overlap the neighbors. Approximations are then made locally, independently, for each extended block, using MFA [1]. There are communication steps for sending the requests from overlapped regions to neighboring blocks and for receiving the evaluations from neighboring blocks. Approximations and evaluations are local operations per block, and blending is also a local operation after the values are received from neighboring blocks. For structured data, the communication pattern is predictable and localized to exchanging information only between neighboring

blocks.

Algorithm 1 Global Blending Method

```
function GLOBALBLEND
    B_i ← decompose(D), i = 1, N
    B_i^ext ← extend(B_i)
    F_i ← localFit(B_i^ext)
    for x ∈ B_i do
        f(x) ← F_i(x)
        if x ∈ overlap(B_i) then
            receive evaluations from neighbors
            f_j(x) ← F_j(x) for j ∈ {neighbors(i)}
            blend(x)
    return
```

Algorithm 2 explains the blending methodology for the multidimensional case, by using the direction of the neighboring affected blocks, relative distance to the shared boundary, and size of the overlap per direction. For a given position x in the overlapped regions, each extended block that covers it computes a different approximation. The blended result is a weighted average, with weights dependent just on position x, similar to Equation (8). Each weight is a product of one-dimensional weights, corresponding to each dimension. Each one-dimensional weight is of the form $\alpha(x)$ or $1 - \alpha(x)$, depending on whether the neighboring block is in the positive or negative direction.

Algorithm 2 Multidimensional Blending

```
function BLEND(x)
    val ← local_MFA(x)
    nbor ← affected_neighbors(x)
    dir ← quadrant(x)
    blending[D] ← 1
    for all d ∈ domain dimensions D do
        if dir(d) < 0 then
            blending[d] ← 1 − α(x_d)
        if dir(d) > 0 then
            blending[d] ← α(x_d)
        val ← val · blending[d]
    for all neighbor ∈ nbor do
        val_remote ← remote_MFA(x)
        factor ← 1
        for all d ∈ domain dimensions D do
            if dir(d) ≠ 0 then
                factor ← 1 − blending[d]
            else
                factor ← blending[d]
        val ← val + factor * val_remote
    return val
```

V. Experimental Results

We performed two sets of experiments, one with synthetic data and the other with simulation data, on the Bebop cluster at the Argonne Laboratory Computing Resource Center. Bebop has 1,024 public nodes, with Intel Broadwell or Intel Knights Landing (KNL) processors, with 128 GB of memory per Broadwell node, and 104 GB per KNL node. It has an Omni-Path fabric interconnect. Broadwell processors have 36 cores, while KNL processors have 64 cores. We used Broadwell processors for these tests.

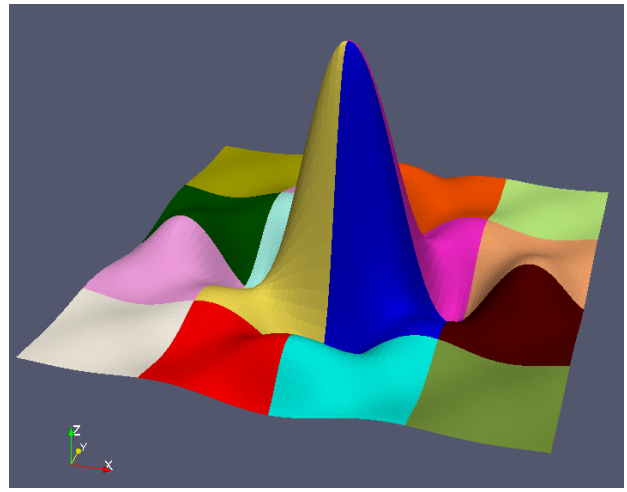

Fig. 9. Example of sinc approximation on 16 blocks.

A. Synthetic Data

Sinc is a synthetic dataset of the cardinal sine function, $y = sin(x)/x$, that we can generate in any dimensionality and resolution. To increase the range and slope of the data, we scaled the sinc function by a factor of 10. The 1-d sinc function is $f(x) = 10sinc(x)$. In two dimensions, $f(x, y) = 10sinc(x)sinc(y)$, and so forth for higher dimensions. Figure 9 shows a 2-d sinc function solved on 16 blocks. The sinc function was chosen for its smoothness; because of its high degree of continuity, the MFA is able to model such data efficiently using high degree (we used $k = 4$).

In this experiment, we selected a fixed number of 2 DIY blocks per MPI process; we used a fixed number of input points per block per direction (50), a fixed overlap fraction between blocks (0.1, or 10%), a fixed number of control points per block and per direction (6), and a fixed spatial dimension (3). We wanted to estimate how much more expensive it is to use blending, compared with just evaluating points independently on each block. A uniform grid of points was evaluated for each block, with 21 points per direction. In each direction, points that are within the 10% overlap are blended with neighboring blocks.

In the weak-scaling study, we varied the number of MPI processes from 1 to 2048, while keeping the amount of work constant per process. The number of processes is at most 32 per node, and tests with no more than 32 MPI processes are run on one node. The results in Figure 10 show good scalability, since the times for evaluating and blending remain almost constant with increasing number of processes. Blending is slightly more expensive than just point evaluation, but Figure 10 shows that the blending cost remains within a constant of the evaluating-only cost, and the overhead for blending is approximately 150%.

B. Scientific Data

In Figure 11 we show strong scaling on an actual scientific dataset. *S3D* is a turbulent combustion dataset generated by an

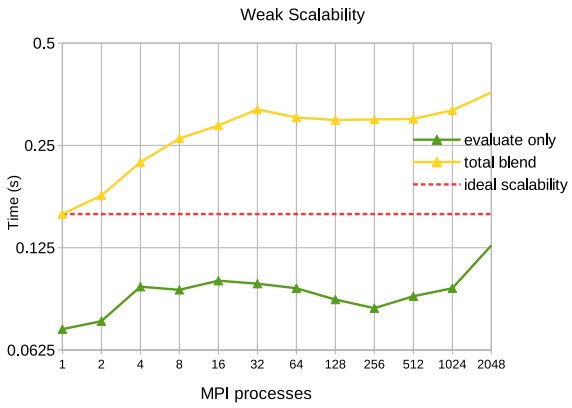

Fig. 10. Weak scaling for sinc 3-d blending

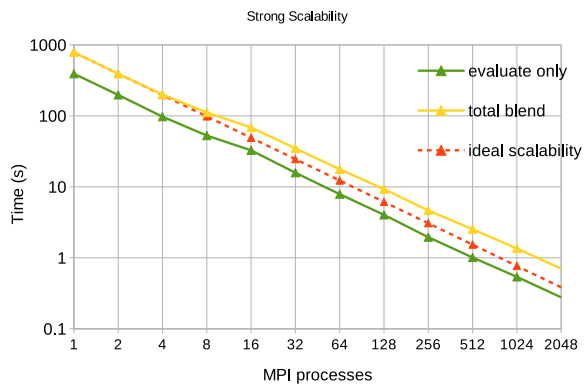

Fig. 11. Strong scaling for S3D dataset

S3D simulation [19] of fuel jet combustion in the presence of an external cross-flow [20]–[23]. The domain is 3-d (x, y, z) $(704 \times 540 \times 550)$, and the range variable $f(x, y, z)$ is the magnitude of the 3-d velocity. This dataset can be sliced to produce 1-d, and 2-d cross-sections, but in this experiment it is used in its original 3-d form. A 2-d slice is shown in Figure 6. We used $k = 3$ for modeling the S3D dataset. The number of control points per direction is about 5 times smaller than the number of input points. We used fixed encoding as in [1], and we doubled the number of blocks alternatively per direction. We used a fixed overlap (ghosting) of 2 input points per direction and a constant loading factor of 2 DIY blocks per MPI process. We started with a grid of $2 \times 1 \times 1$ blocks on 1 process and ended with a grid of $16 \times 16 \times 16$ blocks on 2,048 processes. We used 32 processes per node, so the experiments on at most 32 tasks are run on one node only. Blending added a factor of about 2.2 to the cost of evaluating, and the strong-scaling efficiency is 52% at 2,048 MPI processes.

C. Overlap Cost

Figure 12 shows the total cost of blending, with respect to the number of ghost points per direction. This experiment was run on the same S3D combustion dataset, on 16 compute nodes, with 32 MPI processes per node and 2 DIY blocks per

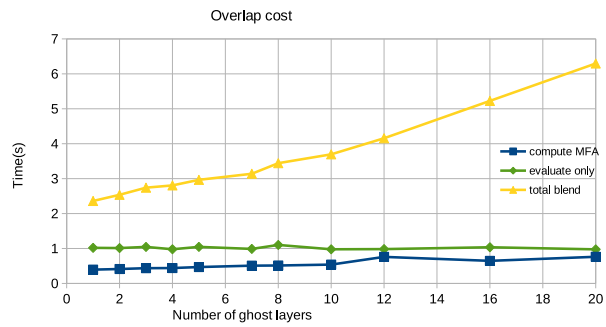

Fig. 12. Computational cost of increasing overlap size

MPI process. Blocks are arranged in a structure $8 \times 8 \times 16$, they are not all the same size. The average size of the blocks is $71 \times 70 \times 47$. The number of ghost layers varies from 2 to 20. Although blending is more expensive, the cost increases almost linearly with the size of the ghosting layer. This is expected, because the number of points to be blended for one block depends on the boundary area of the block and the size of the ghosting layer. The extra cost involves communication between blocks (as shown in Algorithm 1) and the blending computation.

Figures 13 and 14 show a 1-d experiment in which we study the effect of the size of the overlap. We approximate input data with a global MFA with a fixed number of control points (16) and degree 3. We split the problem into two overlapping intervals. We approximate each interval with a local MFA with 8 control points and degree 3, and we blend according to Equation (2) with a blending function of degree 3. We vary the size of the overlap, and we study the effect on the resulting blending approximation and its first derivative.

The top panel in each figure shows the initial input points, global approximation, local approximations for left and right interval, and blended approximation. The middle panel shows the first derivatives of the global approximation, of each local approximation, and of the blended approximation. The bottom panel shows the difference between the first derivative of the global approximation and first derivative of the blended approximation. The vertical dashed lines show the overlap region, where blending occurs.

From Figures 13 and 14 (where Figure 14 is a zoomed-in version over the blending interval of the Figure 13) one may conclude that the amount of overlap was insufficient to approximate the first derivative, because of the spike in the difference in the first derivatives in the blended region.

Figure 15 shows the effect of varying the overlap size on the first derivative difference. The number of input points in the overlap area varies from 3 to 9, or 1 to 4 extra points per side. A notable spike occurs in the first derivative of the difference when the number of input points is lower. This is due to the fact that the blending function first derivative magnitude increases with reducing of the overlap interval, everything else being constant. For overlap size of about 4 input points on each

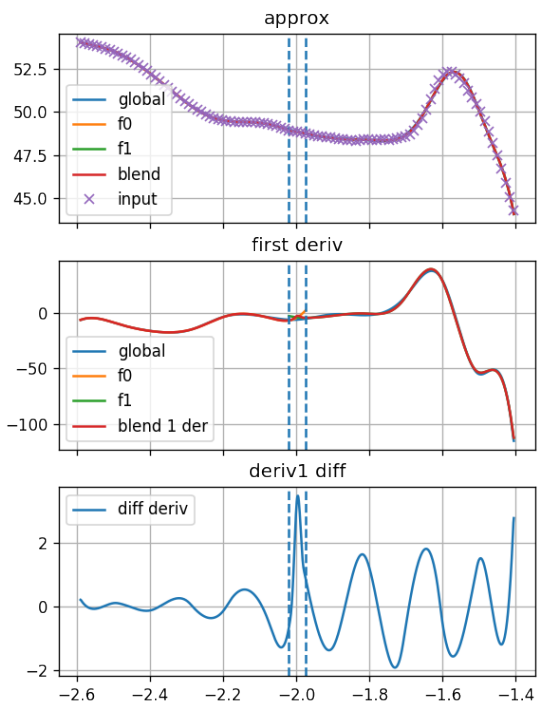

Fig. 13. 1-D overlap experiment

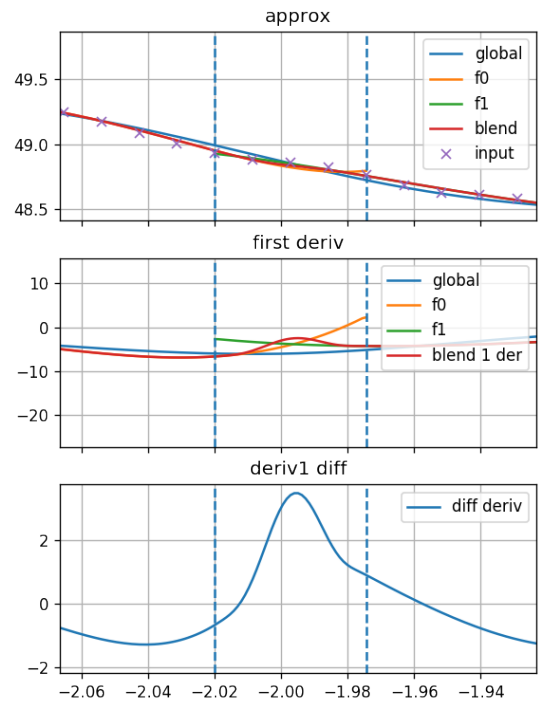

Fig. 14. 1-D overlap experiment zoom-in detail

size (9 in total), the difference in the first derivative in the blending region is about the same magnitude as in the global interval. At the same time, differences in the first derivatives increase in the rest of the interval because more input points are now approximated for the same number of control points per extended interval. We can try to correlate the degree of the blending function with the size of the overlap, and we suggest at least k input points overlap on each side, when k is the continuity degree of the blending function.

VI. Conclusion

In this paper, we showed a simple way to ensure continuity across block boundaries for MFA approximations, and we showed that the method is scalable and its overhead is manageable. We extended blocks with neighboring block data, and we used a multidimensional blending strategy to compute a global approximation that has a desired degree of continuity, as a postprocessing step, during the evaluation of the approximation solution at any point in the domain. In our experiments, we kept a fitting process within each block simple, with a fixed number of control points per direction and per block, since the primary goal of this research is to prove that the blending method used is scalable.

We plan to combine this method with local adaptivity approaches as in [1] and [24], which guarantee a local approximation error that is user-defined for each local block. Each

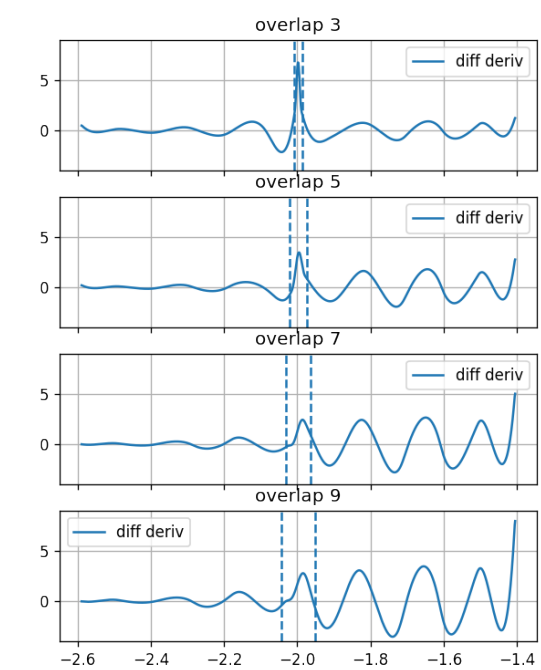

Fig. 15. Effect of the overlap size in first derivative difference

block will adapt its solution to the local input data. Postprocessing blending will just ensure that the global solution is continuous and differentiable over the global domain, keeping the global error under control.

In contrast to enforcing the solution continuity as a postprocessing step through blending functions, a domain decomposition [25] approach to computing the MFA in a concurrent fashion can be employed with constrained minimizers that enforce C^k continuity in the solution across subdomain interfaces. Such an approach would guarantee that the MFA represents the underlying input data without approximations, and would remove the need for corrections during the evaluation stage. Note that this procedure can be applied hierarchically for arbitrary-dimensional problems as well. However, the optimization-based solvers such as L-BFGS or the truncated-Newton family [26] will impose a higher overall encoding computational cost as opposed to the overhead introduced through blending with overlaps. As part of our future MFA studies, we will explore a domain-decomposed additive-Schwartz method [27], with adaptivity, to optimally converge to a global MFA representation that satisfies user-specified error tolerances; we will compare both the approximations and the parallel performance with the blending scheme introduced in this paper.

Acknowledgments

This work is supported by the Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357, program manager Laura Biven. We gratefully acknowledge the computing resources provided on Bebop, a high-performance computing cluster operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

References

[1] T. Peterka, Y. Nashed, I. Grindeanu, V. Mahadevan, R. Yeh, and X. Trixoche, "Foundations of Multivariate Functional Approximation for Scientific Data," in *Proceedings of 2018 IEEE Symposium on Large Data Analysis and Visualization*, 2018.

[2] J. O. Ramsay and B. W. Silverman, *Applied Functional Data Analysis: Methods and Case Studies*, vol. 77. Citeseer, 2002.

[3] F. Ferraty and P. Vieu, *Nonparametric Functional Data Analysis: Theory and Practice*. Springer Science & Business Media, 2006.

[4] J. O. Ramsay, G. Hooker, and S. Graves, *Functional Data Analysis with R and MATLAB*. Springer Science & Business Media, 2009.

[5] B. Boashash, *Time-frequency Signal Analysis and Processing: A Comprehensive Reference*. Academic Press, 2015.

[6] C. E. Heil and D. F. Walnut, "Continuous and Discrete Wavelet Transforms," *SIAM Review*, vol. 31, no. 4, pp. 628–666, 1989.

[7] M. H. Jansen and P. J. Oonincx, *Second Generation Wavelets and Applications*. Springer Science & Business Media, 2005.

[8] C. De Boor, *A Practical Guide to Splines (revised ed.)*. Springer, 2001.

[9] Z. Majdisova and V. Skala, "Radial Basis Function Approximations: Comparison and Applications," *Applied Mathematical Modelling*, vol. 51, pp. 728–743, 2017.

[10] I. Babuska and J. M. Melenk, "The Partition of Unity Method," *International Journal for Numerical Methods in Engineering*, vol. 40, no. 4, pp. 727–758, 1997.

[11] R. Cavoretto, A. D. Rossi, and E. Perracchione, "Partition of Unity Interpolation on Multivariate Convex Domains," *International Journal of Modeling, Simulation, and Scientific Computing*, vol. 06, no. 04, 2015.

[12] R. Cavoretto, A. D. Rossi, and E. Perracchione, "Efficient Computation of Partition of Unity Interpolants through a Block-Based Searching Technique," *Computers & Mathematics with Applications*, vol. 71, no. 12, pp. 2568 – 2584, 2016.

[13] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri, "T-splines and T-NURCCs," *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, pp. 477–484, 2003.

[14] D. Morozov and T. Peterka, "DIY2: Data-Parallel Out-of-Core Library," in *Proceedings of the 2016 IEEE Large Data Analysis and Visualization Symposium LDAV'16*, (Baltimore, MD), 2016.

[15] U. Ayachit, *The ParaView Guide: A Parallel Visualization Application*. USA: Kitware, Inc., 2015.

[16] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, C. Harrison, G. H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E. W. Bethel, D. Camp, O. Rübel, M. Durant, J. M. Favre, and P. Navrátil, "VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data," in *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, pp. 357–372, Oct 2012.

[17] L. Piegl and W. Tiller, *The NURBS Book*. 1997.

[18] Ebert, Musgrave, Peachey, Perlin, and Worley, *Texturing and Modeling, Third Edition: A Procedural Approach*. Elsevier, 2002.

[19] J. H. Chen, A. Choudhary, B. de Supinski, M. DeVries, E. R. Hawkes, S. Klasky, W. K. Liao, K. L. Ma, J. Mellor-Crummey, N. Podhorszki, R. Sankaran, S. Shende, and C. S. Yoo, "Terascale Direct Numerical Simulations of Turbulent Combustion Using S3D," *Comput. Sci. Disc.*, vol. 2, p. 015001, 2009.

[20] R. W. Grout, A. Gruber, C. Yoo, and J. Chen, "Direct Numerical Simulation of Flame Stabilization Downstream of a Transverse Fuel Jet in Cross-flow," *Proceedings of the Combustion Institute*, vol. 33, pp. 1629–1637, 2011.

[21] T. F. Fric and A. Roshko, "Vortical Structure in the Wake of a Transverse Jet," *Journal of Fluid Mechanics*, vol. 279, pp. 1–47, 1994.

[22] T. Peterka, R. Ross, B. Nouanesengsy, T.-Y. Lee, H.-W. Shen, W. Kendall, and J. Huang, "A Study of Parallel Particle Tracing for Steady-state and Time-varying Flow Fields," in *Proceedings of IPDPS 11*, (Anchorage AK), 2011.

[23] T. Peterka, R. Ross, W. Kendall, A. Gyulassy, V. Pascucci, H.-W. Shen, T.-Y. Lee, and A. Chaudhuri, "Scalable Parallel Building Blocks for Custom Data Analysis," in *Proceedings of the 2011 IEEE Large Data Analysis and Visualization Symposium LDAV'11*, (Providence, RI), 2011.

[24] Y. Nashed, T. Peterka, V. Mahadevan, and I. Grindeanu, "Rational Approximation of Scientific Data," in *Proceedings of 2019 International Conference on Computational Science*, to appear.

[25] B. Smith, P. Bjorstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 2004.

[26] C. T. Kelley, *Iterative Methods for Optimization*. SIAM, 1999.

[27] T. F. Chan, B. F. Smith, and J. Zou, "Overlapping Schwarz Methods on Unstructured Meshes Using Non-matching Coarse Grids," *Numerische Mathematik*, vol. 73, no. 2, pp. 149–167, 1996.