MULTIPLE SCALE METHODS FOR OPTIMIZATION OF DISCRETIZED CONTINUOUS FUNCTIONS[*]

NICHOLAS J. E. RICHARDSON[†], NOAH MARUSENKO[‡], AND MICHAEL P. FRIEDLANDER[*†]

Abstract. A multiscale optimization framework for problems over a space of Lipschitz continuous functions is developed. The method solves a coarse-grid discretization followed by linear interpolation to warm-start project gradient descent on progressively finer grids. Greedy and lazy variants are analyzed and convergence guarantees are derived that show the multiscale approach achieves provably tighter error bounds at lower computational cost than single-scale optimization. The analysis extends to any base algorithm with iterate convergence at a fixed rate. Constraint modification techniques preserve feasibility across scales. Numerical experiments on probability density estimation problems, including geological data, demonstrate speedups of an order of magnitude or better.

Key words. multiresolution analysis, continuous optimization, approximation theory

MSC codes. 65B99, 65D15, 90C59

1. Introduction.

Optimization over function spaces arises in applications ranging from density estimation to signal reconstruction. We consider the general problem of optimizing an objective $\mathcal{L} : \mathcal{C} \to \mathbb{R}$ over continuous functions $f : \mathcal{D} \to \mathbb{R}$ defined on an N-dimensional domain $\mathcal{D} \subseteq \mathbb{R}^N$:

$$\min_{f} \{ \mathcal{L}(f) \mid f \in \mathcal{C} \}, \tag{P}$$

where $\mathcal{C} \subseteq \mathcal{F} := \{f : \mathcal{D} \to \mathbb{R} \mid f \text{ is continuous}\}$ is a constraint set.

The space of all continuous functions $\mathcal{F}$ is too large to optimize directly. We therefore restrict $\mathcal{C} \subseteq \mathcal{F}$ to reduce the uncountably infinite dimensional problem in (P) to a countably infinite or finite problem amenable to computation. We focus on Lipschitz continuous solutions. This mild regularity condition holds automatically for continuous functions on compact domains and is needed to prove precise error bounds for our multiscale approach.

One common approach discretizes the problem over a finite grid:

$$\min_{X} \{ \tilde{\mathcal{L}}(X) \mid X \in \tilde{\mathcal{C}} \}, \tag{1.1}$$

where the tensor $X \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ represents the function f evaluated on a set of grid points. The tensor has N modes, matching the dimension of $\mathcal{D}$, with I_n grid points along the nth dimension. The discretized objective $\tilde{\mathcal{L}}$ and constraint set $\tilde{\mathcal{C}}$ correspond to their continuous counterparts.

Any discretization raises the question, How fine should be the discretization? How large should be the grid dimensions $I_1, \ldots, I_N$? When we have discretization choices to make, we favor finer grids for accuracy. When data arrives pre-discretized, we prefer to incorporate all available information. In either case, fine discretization (large I_n) is desirable. Fine discretizations, however, carry computational costs: most operations are slower and consume more memory as problem size grows, often superlinearly, such

[*]This work was funded by the Natural Sciences and Engineering Research Council of Canada. Manuscript Date: December 11, 2025.

[†]Department of Mathematics, University of British Columbia, Vancouver, BC, Canada

[‡]Department of Computer Science, University of British Columbia, Vancouver, BC, Canada/

as with matrix-matrix multiplication [23]. We propose a multiresolution approach that achieves faster convergence on fine-scale problems. Our method is inspired by wavelets [3] and multigrid [25] methods, and optimizes f over progressively finer discretizations.

1.1. Related Work. Many approaches exist for optimizing over function spaces beyond simple discretizations. One class of methods restricts $\mathcal{C}$ to a Hilbert space with basis functions $\{f_k\}$ and solves for optimal weights $\{a_k\}$ in the representation $f(v) = \sum_k a_k f_k(v)$. Examples include Fourier features, wavelets [3], and Legendre polynomials [24]. The calculus of variations provides an analytical framework for optimizing functionals, typically integrals involving the input function [9].

Finite parametrization offers another approach, exemplified by finding the mean μ and variance σ^2 in the family of normal distributions $f(v) = (2\pi\sigma)^{-1/2}\exp(-(v-\mu)^2/(2\sigma^2))$ [26], or optimizing over weights $\{W_k\}$ and biases $\{b_l\}$ in fully connected neural networks [11]. Non-parametric methods such as kernel density estimation [5] and Gaussian mixture models [28] provide yet another alternative. Finally, convex duality techniques transform infinite-dimensional problems into finite-dimensional counterparts [6].

We focus on uniformly spaced discretizations: the approach is simple, used across a variety of settings, easy to implement and analyze, and sufficient in practice and theory for Lipschitz functions. More sophisticated discretization schemes have been well studied in differential equations [22], probability density estimation [4], machine learning [19], and approximation theory [24]. These schemes concentrate the grid where f exhibits larger variations (for differentiable f, where the gradient magnitude is larger) or near domain endpoints. Many results in this paper extend to such alternative discretization schemes.

Our method synthesizes ideas from wavelets [3], multigrid methods for partial differential equations [25], and multiresolution matrix factorization [13]. At each scale s, we solve a distinct optimization problem $(\mathrm{P_s})$ that discretizes the continuous problem (P). Both the tensor dimension $X_s \in \mathbb{R}^{I_s^1 \times \cdots \times I_s^N}$ and the objective function $\tilde{\mathcal{L}}_s$ adapt to the scale: a coarse-scale objective involves fewer summands, different grid points, and modified constraints. This differs from approaches that merely subsample variables while keeping the objective structure fixed. By re-discretizing both the domain and the objective at each scale, our framework ensures that each subproblem P_s properly approximates the continuous problem, not just its reduced fine-scale version.

Each resolution level provides a warm start for the next finer scale, typically with a smaller error than random initialization. Warm-start strategies appear throughout numerical optimization, from stochastic gradient descent [15] to first-order methods [1].

Our coarse-scale optimization updates only a subset of coordinates (the free variables present at the current resolution) while finer-scale variables remain implicit. This connects to coordinate-wise optimization methods, including greedy descent [8], coordinate descent [17], and block coordinate decent [27].

1.2. Contributions. We develop multiscale methods for optimizing over spaces of Lipschitz functions and establish the first theoretical framework for multiscale optimization in this setting. We propose two variants: a *greedy* approach that re-optimizes all variables at each scale, and a *lazy* approach that freezes coarse-scale values and optimizes only newly interpolated points. Both variants achieve faster convergence to tighter error bounds than gradient descent on a single fine discretization. The analysis extends to any optimization algorithm with iterate convergence at a fixed rate. We further develop constraint modification techniques that preserve feasibility across scales. Numerical tests illustrate the effectiveness of the multiscale approach on synthetic and realworld data for a distribution de-mixing problem.

1.3. Reproducible research. The data files and scripts used to generate the numerical results presented in this paper are available from the GitHub repository [21].

2. The Multiscale Optimization Method.

We introduce the multiscale method through a concrete example that demonstrates how it can exploit problem structure across discretization scales to reduce overall computation cost.

2.1. Motivating Example. Consider recovering a smooth probability density function from polynomial measurements. We formulate this as an inverse least-squares problem over differentiable densities p on $\mathcal{D} = [-1, 1]$:

$$\min_p \left\{ \tfrac{1}{2} \left\| \mathcal{A}(p) - y \right\|_2^2 + \tfrac{1}{2}\lambda \left\| p' \right\|_2^2 \;\middle|\; \left\| p \right\|_1 = 1 \text{ and } p \geq 0 \right\},$$

where the constraints ensure p is a valid probability density. The measurement operator $\mathcal{A}$ maps p to an M-vector with elements

$$\mathcal{A}(p)[m] = \langle a_m, p \rangle \quad \text{for} \quad m = 1, \ldots, M,$$

where the normalized Legendre polynomials are

$$a_m(t) = \sqrt{\frac{2m+1}{2}} \sum_{k=0}^{m} \binom{m}{k} \binom{m+k}{k} \left(\frac{t-1}{2} \right)^k.$$

The regularization term $\|p'\|_2^2 = \int_{-1}^{1} (p'(t))^2 \, dt$ encourages smoothness in the recovered density. We discretize this continuous problem on a uniform grid to obtain

$$\min_{x_1} \left\{ \tfrac{1}{2} \left\| A_1 x_1 - y \right\|_2^2 + \tfrac{1}{2}\lambda x_1^\top G_1 x_1 \;\middle|\; \left\| x_1 \right\|_1 = 1 \text{ and } x_1 \geq 0 \right\}, \tag{2.1}$$

where the vector $x_1 = (x_1[1], x_1[2], \ldots, x_1[I_1])$ represents the uniform discretization of p on $[-1, 1]$ at the finest scale with grid spacing $\Delta t_1 = 2/(I_1 - 1)$:

$$x_1[i] = p(t_1[i]) \Delta t_1, \qquad \text{with} \qquad t_1[i] = -1 + 2\frac{i-1}{I_1 - 1}.$$

The discretized measurement operator A_1 and graph Laplacian G_1, which approximates the second derivative operator, are

$$A_1[m, i] = a_m(t_1[i]) \quad \text{and} \quad G_1 = \frac{1}{\Delta t_1^3} \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix}.$$

We return to this example in subsection 2.2, after introducing the general multiscale framework, to demonstrate how the method applies to this specific problem.

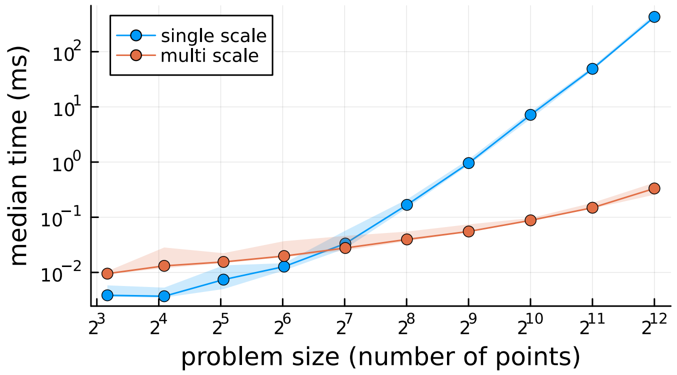


Fig. 1: Comparison of multiscale vs single-scale approach for the motivating example from subsection 2.1. Dots represent the median total time in milliseconds over 100 trials; shaded regions represent the 5th and 95th percentile times. The multiscale approach improves algorithm speed as problem size grows. Details are provided in subsection 2.3.

Algorithm 2.1 High-level multiscale method

```
Require: Discretization chain $\mathcal{D}_S \subset \cdots \subset \mathcal{D}_1$ from (2.3); initial point $X_S^0 \in \tilde{\mathcal{C}}_S$;
    algorithm to approximately solve ($\mathrm{P_s}$); interpolation method.
1: Solve $P_S$ with initial point $X_S^0$ to obtain $X_S$.
2: for scales $s = S-1, S-2, \dots, 1$ do
3:     Compute $\underline{X}_{s+1}$ by interpolating $X_{s+1}$.
4:     Solve $P_s$ with initial point $X_s^0 = \underline{X}_{s+1}$ to obtain $X_s$.
5: end for
6: Construct piecewise linear approximation $\hat{f}_{X_1}$ from $X_1$.
7: return $\hat{f}_{X_1}$, an approximate solution to $P$.
```

2.2. Method Overview. We solve the continuous optimization problem (P) via progressive grid refinement. Starting from a coarse discretization, we solve the discretized problem at each scale, interpolate the solution to the next finer grid, and use it to warm-start the optimization. Algorithm 2.1 formalizes this refinement process.

We index discretization levels by a *scale* parameter s, where $s = 1$ denotes the finest discretization and increasing s corresponds to progressively coarser grids. At scale s, the discretized problem becomes

$$\min_{X_s} \left\{ \tilde{\mathcal{L}}_s(X_s) \;\middle|\; X_s \in \tilde{\mathcal{C}}_s \right\}, \tag{$\mathrm{P_s}$}$$

where X_s is a real-valued N-mode tensor with dimension $(I_s^1 \times \cdots \times I_s^N)$, and $\tilde{\mathcal{L}}_s$ and $\tilde{\mathcal{C}}_s$ denote the objective and constraint set discretized at scale s.

We return to the motivating example from subsection 2.1 to demonstrate how the general framework applies. Recall that we recover a probability density from Legendre polynomial measurements, discretized at the finest scale as (2.1). At scale s, the discretized problem becomes

$$\min_{x_s} \left\{ \tfrac{1}{2}\, \|A_s x_s - y\|_2^2 + \tfrac{1}{2}\lambda x_s^\top G_s x_s \;\middle|\; \|x_s\|_1 = 2^{s-1} \text{ and } x_s \geq 0 \right\} \tag{2.2}$$

where x_s is an $I_s = 2^{s-1} + 1$ vector that discretizes p at scale s with grid spacing $\Delta t_s = 2^{s-1}\Delta t_1$. The constraint $\|x_s\|_1 = 2^{s-1}$ scales with the grid spacing to preserve the continuous constraint $\int p(t)\, dt = 1$ across all scales (see section 3). The scaled measurement operator A_s and graph Laplacian G_s are defined as

$$A_s = 2^{s-1} A_1[:, 1 : 2^{s-1} : I_1] \quad \text{and} \quad G_s = 2^{1-s} G_1[1 : 2^{s-1} : I_1, 1 : 2^{s-1} : I_1],$$

where the slicing notation $1 : 2^{s-1} : I_1$ selects every (2^{s-1})-th point from the finest grid. The prefactors 2^{s-1} and 2^{1-s} ensure the two main quantities in the loss remain balanced at each scale:

$$A_s x_s \approx A_1 x_1 \quad \text{and} \quad x_s^\top G_s x_s \approx x_1^\top G_1 x_1.$$

This balancing ensures each scaled problem is regularized similarly by λ.

2.3. Motivating Example: Numerics. To emphasise the concepts behind multiscale, we compare an instance of the multiscale method (see Algorithm 2.2) against single scale optimization on the motivating example from subsection 2.1. Figure 1 illustrates the improvement in computation time versus problem size achieved by the multiscale approach. Additional numerics and details are provided in section 7.

We compare total runtime for $S = 3, 4, \dots, 12$, which yields finest discretizations with $I_1 = 2^S + 1$ points. We initialize by uniformly discretizing the polynomial $p(t) = -2.625t^4 - 1.35t^3 + 2.4t^2 + 1.35t + 0.225$ on $[-1, 1]$ to obtain the initial density approximation $x_S[i] = f(t_S[i])\Delta t_S$ at the coarsest scale.

At each scale $s > 1$, we perform one ($K_s = 1$) iteration; at the finest scale ($s = 1$), and we iterate until the objective is within 5% of its optimal value. We generate Legendre measurements for $m = 1, \dots, 5$ using the measurement operator A_1 from subsection 2.1 and add 5% Gaussian noise to obtain y. The regularization parameter $\lambda = 10^{-4}$ balances data fidelity against smoothness.

In Figure 2, we show the typical convergence of iterates x_s^k for single scale and multiscale approaches. We run the motivating example from subsection 2.1 with $2^{10} + 1$ points at the finest scale $s = 1$. The top-left plot shows every 25th iterate k starting with the initialization at $k = 0$ and ending at $k = 250$ when running projected gradient descent at the finest scale ($s = 1$). The top-right plot shows every 250th iterate k, ending with the final iterate $k = 4988$ which satisfies the stopping criteria $\tilde{\mathcal{L}}_1(x_1^k) < 0.00024$. The bottom-left plot shows x_s^k at each scale from the coarsest scale $s = 10$ to the second finest $s = 2$, and the iterate at the finest scale $s = 1$ after $k = 25$ iterations. We can make a fair comparison between x_1^{25} on the top-left and bottom-left plots since they are both the result of running $k = 25$ iterations of projected gradient descent at the finest scale. Using multiscale results in a smoother iterate since x_1^0 was constructed from the chain of interpolations, rather than the completely random initialization used by the single scale approach. Because of this, the bottom-left plot highlights multiscale only needs $k = 387$ iterations at the finest scale to also achieve the same stopping criteria $\tilde{\mathcal{L}}_1(x_1^k) < 0.00024$.

We call multiscale's tendency to smooth iterates an implicit regularization. This helps smooth solutions by carrying information from farther away points to speed up convergence. In this problem, the graph laplacian regularizer only looks at the immediate neighbouring points. So working at coarser scales lets us compare points that are not immediately neighbouring at the finest scale.

In Figure 3, we show the typical loss convergence for the single scale and multiscale methods. These are shown as a function of time rather than number of iterations.

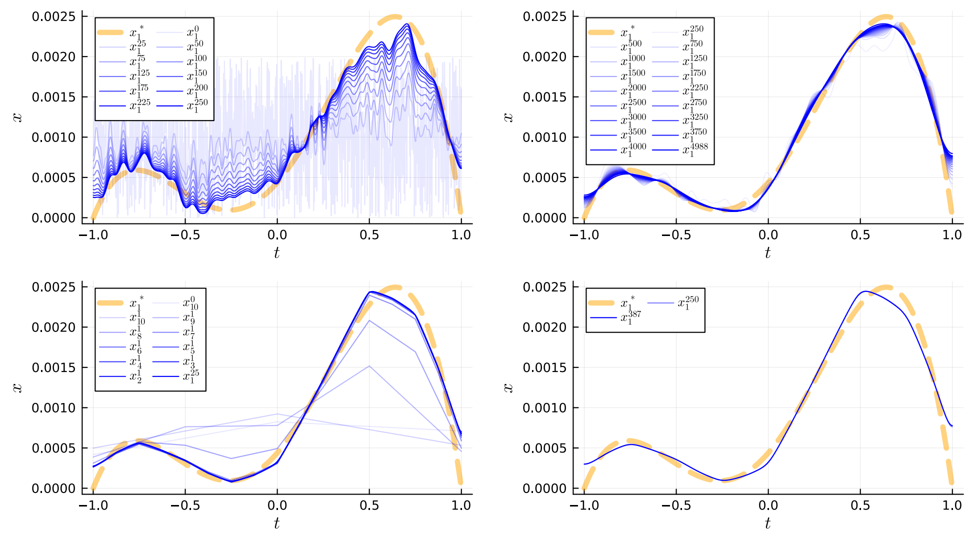


Fig. 2: Typical progression of iterates x_s^k using single scale (top row) and multiscale (bottom row) approachs for the motivating example from subsection 2.1. The clean ground truth is plotted behind in the thick dashed lined.

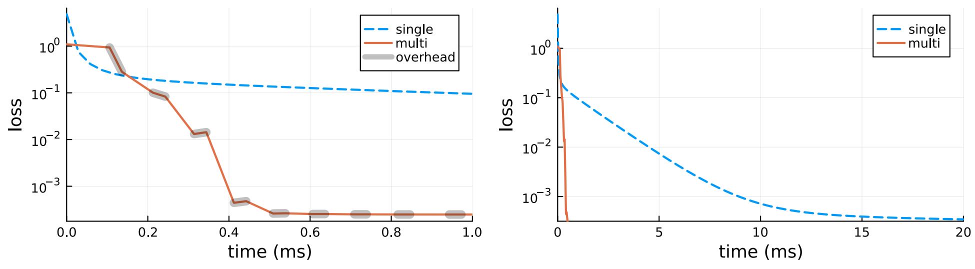


Fig. 3: Typical loss convergence using single scale (dashed blue) and multiscale (solid orange) approachs for the motivating example from subsection 2.1. The left plot shows the first millisecond, and the right plot shows the first 20 milliseconds. The left plot also highlights in grey regions that are not from projected gradient descent steps. These are the result of additional overhead with the multiscale method such as interpolation, allocations, and extra function calls.

Comparing iterations at different scales is not fair since we expect one projected gradient step to be cheaper at coarser scales. We also want to highlight the minimal effect of interpolation, allocations, and additional function calls. The loss $\tilde{\mathcal{L}}_s(x_s^k)$ for the multiscale method is calculated at the respective scale s of the iterate, whereas the single scale only calculates the the loss $\tilde{\mathcal{L}}_1(x_1^k)$ at the finest scale $s = 1$. This explains why the loss fluctuates up and down during interpolations for coarser scales s when the approximation $\tilde{\mathcal{L}}_s$ for $\mathcal{L}$ is worse. The coarsest scales also take longer than expected because of the additional function call overhead. Despite these drawbacks, multiscale more than makes up for these by reducing the loss significantly faster once iterating on the finer scales. This is make clear by the right plot in Figure 3.

It could be argued that for optimal convergence rates with a first order method, we should not be using a fixed stepsize based on the global Lipschitz constant [18]. Using accelerated methods would almost certainly accelerate convergence and not require almost 5000 iterations. But any sort of acceleration could also be used to speed up convergence for the multiscale method which already has a leg-up since the fine scale iterations start closer to the solution.

2.4. Descritization, Coarsening, and Interpolation. Consider a nested sequence of discretizations

$$\mathcal{D}_S \subset \mathcal{D}_{S-1} \subset \cdots \subset \mathcal{D}_1 \subset \mathcal{D} \subseteq \mathbb{R}^N \tag{2.3}$$

of the continuous domain $\mathcal{D}$. Scale $s = S$ represents the coarsest grid. Each discretization $\mathcal{D}$ contains $|\mathcal{D}_s| = \prod_{n=1}^N I_s^n$ points. The tensor T with dimension $(I_s^1 \times \cdots \times I_s^N)$ contains the discretization points from $\mathcal{D}_s$. The optimization variable X_s represents the function values at these points:

$$X_s[i_1, \dots, i_N] = f(T_s[i_1, \dots, i_N]).$$

Unless otherwise stated, we restrict the remainder of the paper to scalar functions $f : [\ell, u] \to \mathbb{R}$ on the 1-dimensional domain $\mathcal{D} = [\ell, u]$ with uniform discretization

$$x_s[i] = f(t_s[i]), \qquad \text{with} \qquad t_s[i] = \ell + \frac{i-1}{I_s - 1}(u - \ell).$$

This specialization simplifies our exposition and is without loss of generality because the multiscale framework extends to tensor-valued functions by vectorizing $X_s \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ as $x_s \in \mathbb{R}^{I_1 \cdots I_N}$ and adapting interpolation operators accordingly. Section 7 demonstrates this extension.

The multiscale framework requires operators to transfer solutions of ($\mathrm{P_s}$) between scales and to eventually construct approximate solutions to the continuous problem (P). Any coarsening and interpolation method could be used, including splines [7] and Chebyshev and Lagrange polynomials [24]. We focus on dyadic coarsening, where each coarse grid comprises every second point, and midpoint linear interpolation. These are defined below. Figure 4 illustrates this discretization hierarchy and midpoint interpolation scheme.

Definition 2.1 (Dyadic Coarsening). *Dyadic coarsening maps a vector $x \in \mathbb{R}^I$ to $\overline{x} \in \mathbb{R}^{\lfloor (I+1)/2 \rfloor}$ by selecting odd-indexed entries:*

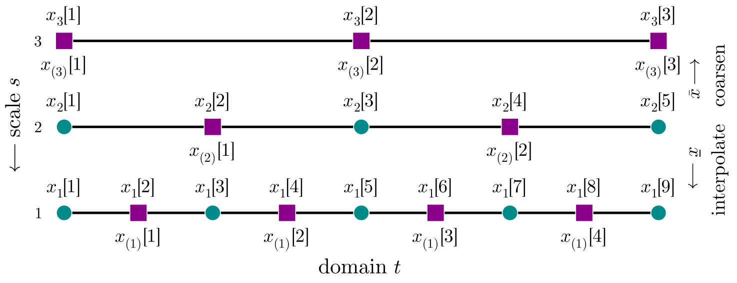
$$\overline{x}[i] = x[2i-1], \qquad i = 1, \dots, \lfloor (I+1)/2 \rfloor.$$

Definition 2.2 (Midpoint Linear Interpolation). *Midpoint linear interpolation inserts a linearly interpolated point between each adjacent pair of entries in $x \in \mathbb{R}^I$ to produce $\underline{x} \in \mathbb{R}^{2I-1}$:*

$$\underline{x}[i] = \begin{cases} x[\frac{i+1}{2}] & \text{if } i \text{ is odd,} \\ \frac{1}{2}\left(x[\frac{i}{2}] + x[\frac{i}{2}+1]\right) & \text{if } i \text{ is even.} \end{cases}$$

The operators in the coarsening and interpolation definitions translate between scales s with the notation

$$\overline{x}_s = x_{s+1} \qquad \text{and} \qquad \underline{x}_s = x_{s-1}.$$



$x_3[1]$ $x_3[2]$ $x_3[3]$
3
$x_{(3)}[1]$ $x_{(3)}[2]$ $x_{(3)}[3]$
$x_2[1]$ $x_2[2]$ $x_2[3]$ $x_2[4]$ $x_2[5]$
2
$x_{(2)}[1]$ $x_{(2)}[2]$
$x_1[1]$ $x_1[2]$ $x_1[3]$ $x_1[4]$ $x_1[5]$ $x_1[6]$ $x_1[7]$ $x_1[8]$ $x_1[9]$
1
$x_{(1)}[1]$ $x_{(1)}[2]$ $x_{(1)}[3]$ $x_{(1)}[4]$
domain t
$\longleftarrow$ scale s
$\bar{x} \longrightarrow$ coarsen
interpolate $\underline{x} \longrightarrow$

Fig. 4: Example uniform discretization of an interval with $S = 3$ scales. Each scale s has $2^{S-s+1} + 1$ points. The free variables $x_{(s)}$ (magenta squares) resemble a "W-cycle" from multigrid [25].

The overline operator $\overline{(\cdot)}$ maps x_s to the coarser scale $s + 1$; the underline operator $\underline{(\cdot)}$ maps to the finer scale $s - 1$; see Figure 4.

When analyzing the multiscale algorithm, we need to refer to the newly added interpolated points. We call this vector the *free variables* $x_{(s)}$, a term justified in subsection 2.5.

Definition 2.3 (Vector of Free Variables). *Given the* (I_{s+1})*-vector* x_{s+1} *and its interpolation* $\underline{x}_{s+1} = x_s$*, the* $(I_{s+1} - 1)$*-vector of* free variables *has entries*

$$x_{(s)}[i] = \tfrac{1}{2}(x_{s+1}[i] + x_{s+1}[i+1]), \quad i = 1, \dots, I_{s+1} - 1.$$

These are the newly added points in x_s *when interpolating* x_{s+1}*. At the coarsest scale* $s = S$*, we set* $x_{(S)} = x_S$*.*

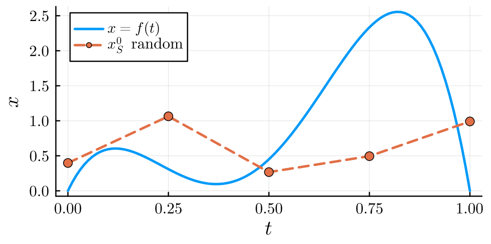
To convert a discrete solution x_1 of ($\mathrm{P_s}$) at the finest scale $s = 1$ to an approximate solution to the continuous problem (P), we define a piecewise linear approximation. This approximation stitches secants through (possibly approximate) sample points [7, Ch. I, Eq. 9].

Definition 2.4 (Piecewise Linear Approximation). *Let the* I*-vector* x *represent (possibly approximate) samples of a function* f *at locations* $t_1 < \cdots < t_I$*. The piecewise linear approximation constructed from* x *is*

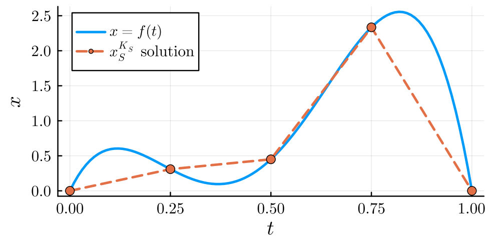
$$\hat{f}_x(t) = \begin{cases} \frac{x[2]-x[1]}{t_2-t_1}(t - t_1) + x[1] & \text{if } t_1 \le t \le t_2, \\ \vdots & \\ \frac{x[I]-x[I-1]}{t_I-t_{I-1}}(t - t_{I-1}) + x[I-1] & \text{if } t_{I-1} \le t \le t_I. \end{cases}$$

Remark 1. *The subscript in* $\hat{f}_x$ *identifies the vector* x *that defines the approximation. When* x *contains samples from a function, that is,* $x[i] = f(t_i)$*, we omit the subscript and write* $\hat{f}(t)$*.*

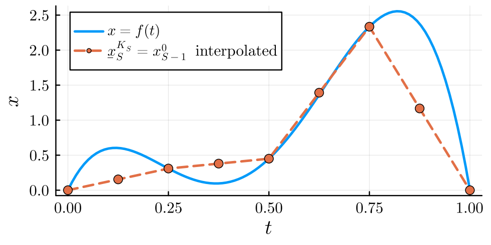
2.5. Greedy and Lazy MultiScale Algorithms. We present two variants of the multiscale method outlined in subsection 2.2 for solving (P): the *greedy* variant (Algorithm 2.2), which optimizes all grid points at each scale, and the *lazy* variant (Algorithm 2.3), which optimizes only the newly interpolated points at each finer scale. Both variants apply an update rule $x^{k+1} = U(x^k)$ at each scale, and our convergence analysis holds for any such update rule that achieves q-linear iterate convergence.



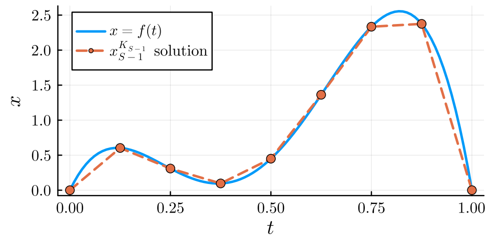
(a)



(b)



(c)



(d)

Fig. 5: One hypothetical round of multiscale optimization. In all four subplots, the solid blue curve represents the solution to the continuous problem. The orange dots represent a) a random coarse initialization with four points, b) the solution at the coarse scale, c) the interpolation of the coarse solution to the fine grid, and d) the solution at the fine scale.

Definition 2.5 (Q-Linear Iterate Convergence). *An iterative algorithm with update rule $x^{k+1} = U(x^k)$ has global q-linear iterate convergence with rate $q \in [0, 1)$ when*

$$\left\|x^k - x^*\right\|_2 \leq (q)^k \left\|x^0 - x^*\right\|_2$$

for any feasible initialization $x^0 \in \tilde{\mathcal{C}}$ and some minimizer x^* of $\tilde{\mathcal{L}}(x)$ over $\tilde{\mathcal{C}}$ that may depend on x^0.

Figure 5 illustrates one round of multiscale optimization. Suppose the solution to the continuous optimization problem is

$$f(t) = -84t^4 + 146.4t^3 - 74.4t^2 + 12t$$

on the interval $0 \leq t \leq 1$. The figure displays the multiscale flow: initialize at the coarse scale with five points, optimize at the coarse scale, interpolate to the fine grid (adding four points), and optimize at the fine scale. In this particular example, the greedy and lazy variants share this structure but differ in the final optimization: greedy optimizes all nine fine-scale points, while lazy optimizes only the four newly interpolated points.

Our software implementation, described in section 7, uses projected gradient descent as the update rule, which achieves q-linear iterate convergence for smooth and strongly convex objectives $\tilde{\mathcal{L}}(x)$ with closed and convex constraints $\tilde{\mathcal{C}}$ [2, Sec. 10.2].

Algorithm 2.2 Greedy multiscale method

```
1: Randomly initialize $x_s^0$ at the coarsest scale $s = S$.
2: Perform $K_s$ iterations of projected gradient descent (2.4) on $x_s^0$ to approximately
   solve (P_s) at scale $s = S$ and obtain $x_s^{K_s}$.
3: for scales $s = S-1, S-2, \dots, 1$ do
4:     Interpolate $x_{s+1}^{K_{s+1}}$ to obtain $\underline{x}_{s+1}^{K_{s+1}}$ according to Definition 2.2.
5:     Initialize the next finest scale $x_s^0 = \underline{x}_{s+1}^{K_{s+1}}$ using the previous scale's interpolated
       solution.
6:     Perform $K_s$ iterations of projected gradient descent (2.4) on $x_s^0$ to approximately
       solve (P_s) at scale $s$ and obtain $x_s^{K_s}$.
7: end for
8: return Approximate solutions $x_1^{K_1}$ and $\hat{f}_{x_1^{K_1}}$ for (P_s) at $s = 1$ and (P) respectively.
```

This update rule takes the form

$$x^{k+1} \leftarrow U(x^k) = P_{\tilde{\mathcal{C}}}\left(x^k - \alpha \nabla \tilde{\mathcal{L}}(x^k)\right) \tag{2.4}$$

where

$$P_{\tilde{\mathcal{C}}}(x) = \operatorname{argmin}\left\{ \, \|x - y\|_2 \;\middle|\; y \in \tilde{\mathcal{C}} \, \right\}$$

denotes the projection of x onto $\tilde{\mathcal{C}}$. The standard analysis of projected gradient descent gives the following q-linear rate for strongly-convex objectives.

Lemma 2.6 (Descent Lemma [18, Theorem 2.1.15]). *Projected gradient descent with stepsize α has iterate convergence with rate*

$$q(\alpha) = \sqrt{1 - \frac{2\alpha \mathcal{S}_{\tilde{\mathcal{L}}} \mu_{\tilde{\mathcal{L}}}}{\mathcal{S}_{\tilde{\mathcal{L}}} + \mu_{\tilde{\mathcal{L}}}}}.$$

This is minimized at a stepsize of $\alpha = 2/(\mathcal{S}_{\tilde{\mathcal{L}}} + \mu_{\tilde{\mathcal{L}}})$, where we obtain the iterate convergence

$$q(\alpha) = \frac{c-1}{c+1}$$

for a condition number *of $c = \mathcal{S}_{\tilde{\mathcal{L}}}/\mu_{\tilde{\mathcal{L}}}$.*

Although we can prescribe the number of iterations K_s at each scale in advance, in practice we may also use additional convergence criteria (such as a small gradient) at each scale in addition to this predetermined maximum.

The lazy variant freezes previously computed entries of x and adjusts only the newly interpolated values $x_{(s)}$ (Definition 2.3) during the algorithm update step. Figure 4 illustrates these free variables. Algorithm 2.3 implements this approach, differing from Algorithm 2.2 only in lines 1, 2, and 6.

3. Discretizing Constraints.

The multiscale approach requires discretizing (P) at multiple scales. Given a discretization of the continuous constraint $\mathcal{C}$ from (P) to the finest scale $\tilde{\mathcal{C}}_1$ in (P$_s$), we must determine the appropriate discretized constraint sequence $\tilde{\mathcal{C}}_s$ at coarser scales $s = S-1, S-2, \dots, 1$.

Pointwise constraints $\mathcal{C} = \{f : \mathcal{D} \to \mathbb{R} \mid f(t) \in \mathcal{X}\}$ transfer verbatim to any scale: $\tilde{\mathcal{C}}_s = \left\{x_s \in \mathbb{R}^{I_s} \mid x[i] \in \mathcal{X}\right\}$. However, linear and moment constraints require careful scaling to maintain consistency across discretization levels.

Algorithm 2.3 Lazy multiscale method

```
1: Randomly initialize $x^0_{(s)}$ (Definition 2.3) at the coarsest scale $s = S$.
2: Perform $K_s$ iterations of projected gradient descent (2.4) on $x^0_{(s)}$ to approximately
   solve (P$_\text{s}$) at scale $s = S$ and obtain $x^{K_s}_{(s)}$.
3: for scales $s = S-1, S-2, \ldots, 1$ do
4:     Interpolate $x^{K_{s+1}}_{s+1}$ to obtain $\underline{x}^{K_{s+1}}_{s+1}$ according to Definition 2.2.
5:     Initialize the next finest scale $x^0_s = \underline{x}^{K_{s+1}}_{s+1}$ using the previous scale's interpolated
       solution.
6:     Perform $K_s$ iterations of projected gradient descent (2.4) on $x^0_{(s)}$ to approximately
       solve (P$_\text{s}$) at scale $s$ and obtain $x^{K_s}_{(s)}$.
7: end for
8: return Approximate solutions $x^{K_1}_1$ and $\hat{f}_{x^{K_1}_1}$ for (P$_\text{s}$) at $s = 1$ and (P) respectively.
```

Consider a p-norm constraint on f:

$$\mathcal{C} = \left\{ f : \mathcal{D} \to \mathbb{R} \;\middle|\; \|f\|_p = c \right\}.$$

The natural discretization

$$\tilde{\mathcal{C}}_1 = \left\{ x_1 \in \mathbb{R}^{I_1} \;\middle|\; \|x_1\|_p = c \right\}$$

correctly approximates $\mathcal{C}$ when we scale the entries of x_1 by the grid size:

$$x_1[i] = f(t_1[i])(\Delta t)^{1/p}.$$

With this scaling, the discrete norm approximates the continuous norm:

$$c^p := \|x_1\|_p^p = \sum_{i=1}^{I_1} f(t_1[i])^p \Delta t \approx \int_{\mathcal{D}} f(t)^p \, dt = \|f\|_p^p ,$$

as $I_1 \to \infty$.

At coarse scale s, we modify the constraint to

$$\tilde{\mathcal{C}}_s = \left\{ x_s \in \mathbb{R}^{I_s} \;\middle|\; \|x_s\|_p = c \cdot (I_s/I_1)^{1/p} \right\}.$$

This scaling preserves the constraint across scales. The Lipschitz property of f ensures neighboring entries in x_s remain similar, so

$$\frac{I_1}{I_s} \sum_{i=1}^{I_s} (x_s[i])^p \approx \sum_{i=1}^{I_1} (x_1[i])^p,$$

which gives

$$c^p = \frac{I_1}{I_s} \|x_s\|_p^p \approx \|x_1\|_p^p .$$

Linear constraints require analogous scaling. For L_{g_k}-Lipschitz functions $g_k : \mathcal{D} \to \mathbb{R}$, the continuous constraint

$$\mathcal{C} = \{ f : \mathcal{D} \to \mathbb{R} \mid \langle g_k, f \rangle = b[k],\, k \in [K] \} \quad \text{where} \quad b \in \mathbb{R}^K,$$

discretizes at scale s as

$$\tilde{\mathcal{C}}_s = \left\{ x_s \in \mathbb{R}^{I_s} \;\middle|\; A_s x_s = b \cdot (I_s/I_1) \right\},$$

where

$$A_s[k,i] = g_k(t_s[i])(\Delta t)^{1/2} \quad \text{and} \quad x_s[i] = f(t_s[i])(\Delta t)^{1/2}.$$

We bound the discretization error to justify our constraint modification across scales. We assume grid points double at each scale refinement ($I_s/I_{s+1} = 2$) and use a uniform grid $\{t_i\}$ on the domain $\mathcal{D} = [\ell, u]$ to sample an L_f-Lipschitz function f according to $x[i] = f(t_i)$. The following proposition quantifies how linear constraints transform under coarsening.

Proposition 3.1 (Single Linear Constraint Scaling). *Let $a \in \mathbb{R}^I$ discretize a bounded L_g-Lipschitz function $g : [\ell, u] \to \mathbb{R}$, and let x discretize a bounded L_f-Lipschitz function f. Define the Lipschitz constant for the product function fg as $L_{fg} = L_f \|g\|_\infty + L_g \|f\|_\infty$.*

If $\langle a, x\rangle = b$, then the coarsened constraint satisfies the bounds

$$\begin{aligned} \left|\langle \overline{a}, \overline{x}\rangle - \tfrac{1}{2}b\right| &\leq \tfrac{1}{4}L_{fg}(u-l)\left(\frac{I}{I-1}\right) && \textit{for even } I,\\ \left|\langle \overline{a}, \overline{x}\rangle - \tfrac{1}{2}\frac{I+1}{I}b\right| &\leq \tfrac{1}{2}L_{fg}(u-l) && \textit{for odd } I. \end{aligned}$$

Proof. See subsection SM2.1. ∎

Proposition 3.1 extends naturally to multiple linear constraints, as described in the following corollary.

Corollary 3.2 (Matrix Linear Constraint Scaling). *Let $g_k : [\ell, u] \to \mathbb{R}$ be bounded L_{g_k}-Lipschitz functions for $k \in [K]$. Let $A \in \mathbb{R}^{K\times I}$ be the discretization $A[k,i] = g_k(t[i])$, and define $\overline{A} = A[:, \texttt{begin} : 2 : \texttt{end}]$ as the $(K \times \lceil I \rceil_e /2)$-submatrix with every other column removed, where $\lceil I \rceil_e$ rounds up to the nearest even integer. Define $L_{fg} = \max_{k\in[K]}(L_{fg_k}) = \max_{k\in[K]}(L_f \|g_k\|_\infty + L_{g_k}\|f\|_\infty)$ as the maximum Lipschitz constant of the product functions $\{fg_k\}_{k\in[K]}$.*

If $Ax = b \in \mathbb{R}^K$, then

$$\begin{aligned} \left\|\overline{A}\overline{x} - \tfrac{1}{2}b\right\|_2 &\leq \sqrt{K}\cdot \tfrac{1}{4}L_{fg}(u-l)\left(\frac{I}{I-1}\right) && \textit{for even } I,\\ \left\|\overline{A}\overline{x} - \frac{I+1}{I}\tfrac{1}{2}b\right\|_2 &\leq \sqrt{K}\cdot \tfrac{1}{2}L_{fg}(u-l) && \textit{for odd } I. \end{aligned}$$

Proof. See subsection SM2.2. ∎

The 1-norm constraint admits analogous scaling bounds.

Proposition 3.3 (L1-norm Constraint Scaling). *Let $x \in \mathbb{R}^I$ discretize a bounded L_f-Lipschitz function f. If $\|x\|_1 = b$, then the coarsened constraint satisfies the bounds*

$$\begin{aligned} \left|\|\overline{x}\|_1 - \tfrac{1}{2}b\right| &\leq \tfrac{1}{4}L_f(u-l)\left(\frac{I}{I-1}\right) && \textit{for even } I,\\ \left|\|\overline{x}\|_1 - \frac{I+1}{I}\tfrac{1}{2}b\right| &\leq \tfrac{1}{2}L_f(u-l) && \textit{for odd } I. \end{aligned}$$

Proof. The proof is similar to the proof of Proposition 3.1 (see subsection SM2.1) by considering the vector $a = \operatorname{sign}(x)$, and using the Lipschitz constant L_f for f to bound $|x[i] - x[i+1]|$ in place of L_{fg}. ∎

Proposition 3.3 specializes to a clean asymptotic behavior in the normalized case. For $z = x/\|x\|_1$, the bound for even I becomes

$$\left|\|\overline{z}\|_1 - \tfrac{1}{2}\right| \leq \tfrac{1}{4b} L_f(u-l)\left(\frac{I}{I-1}\right).$$

As the discretization refines ($I \to \infty$), we have $b = \|x\|_1 \to \infty$, so the bound vanishes and $\|\overline{z}\|_1 \to 1/2$. The same limit holds for odd I, and also extends to other p-norms.

4. Convergence of a Multiscale Method.

Both the standard and multiscale approaches execute many iterations at the finest scale and converge because, by assumption, they both apply a convergent algorithm at the finest scale. The multiscale achieves faster convergence by warm starting the fine-scale iterations with solutions from coarser scales. The next subsections develop this analysis.

Subsection 4.1 establishes tight bounds on the error between smooth and strongly convex Lipschitz functions and their linear interpolation. Subsection 4.2 proves convergence when we solve the problem along the entire new discretization at each scale. Subsection 4.3 proves convergence when we freeze previously computed points and solve only at the newly interpolated points for each scale.

4.1. Functional Analysis Lemmas.

To analyze multiscale method convergence, we first establish bounds on discretizing Lipschitz functions and approximating them with linear interpolations. We focus on solving the discretized problem ($\mathrm{P_s}$) in one dimension with any algorithm that achieves iterate convergence at the rate specified in Definition 2.5.

The following lemma provides the foundation for analyzing linear interpolations of Lipschitz functions and, ultimately, our multiscale algorithm.

Lemma 4.1 (Lipschitz Function Interpolation). *Let $f : \mathbb{R}^N \to \mathbb{R}$ be L_f-Lipschitz. For all $a, b \in \mathbb{R}^N$ and $\lambda_1, \lambda_2 \geq 0$, $\lambda_1 + \lambda_2 = 1$, the error between f evaluated at a convex combination of a and b and the linear interpolation of $f(a)$ and $f(b)$ satisfies*

$$|f(\lambda_1 a + \lambda_2 b) - (\lambda_1 f(a) + \lambda_2 f(b))| \leq 2L_f \lambda_1 \lambda_2 \|a - b\|_2 . \tag{4.1}$$

Proof. See subsection SM3.1. ∎

Since Lemma 4.1 is central to the analysis of linear interpolations of Lipschitz functions and ultimately our multiscale algorithm, we require a tight bound on the interpolation error. The bound in (4.1) is optimal among all bounds independent of the endpoint values $f(a)$ and $f(b)$, as demonstrated by the explicit construction in the proof of Lemma 4.2.

Lemma 4.2 (Lipschitz Interpolation Tightness). *Given $\lambda_1, \lambda_2 \geq 0$ with $\lambda_1 + \lambda_2 = 1$, an interval $[\ell, u]$, and constant $L_f > 0$, there exists an L_f-Lipschitz function on $[\ell, u]$ such that (4.1) holds with equality.*

Proof. Construct

$$f(t) = L_f \left|t - (\lambda_1 \ell + \lambda_2 u)\right|.$$

This function is L_f-Lipschitz with end-point values $f(\ell) = L_f \lambda_2 |\ell - u|$ and

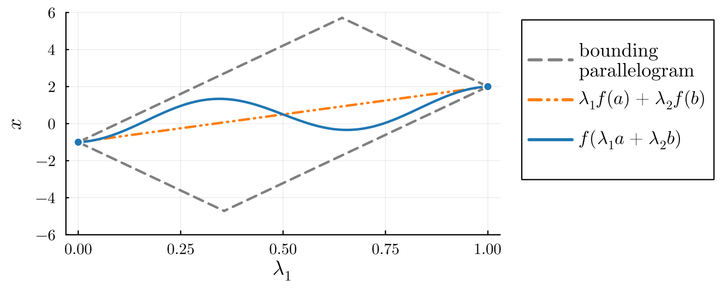


Fig. 6: Example for Lemma 4.1 with the function $x = f(t) = t - \cos(3\pi t)$ on the interval $[\ell, u] = [0, 1]$. The function f must lie within the dashed parallelogram since it is $L_f = 1 + 3\pi$ Lipschitz. Lemma 4.1 uses the parallelogram constraint to bound the distance between f and the linear interpolation $\lambda_1 f(a) + \lambda_2 f(b)$. Lemma 4.2 shows that the bound in (4.1) is tight for any given $\lambda_1, \lambda_2 \geq 0$ with $\lambda_1 + \lambda_2 = 1$.

$f(u) = L_f \lambda_1 |\ell - u|$, and $f(\lambda_1 \ell + \lambda_2 u) = 0$. Therefore,

$$\begin{aligned} &|\lambda_1 f(\ell) + \lambda_2 f(u) - f(\lambda_1 \ell + \lambda_2 u)| \\ &\quad = \big|\lambda_1 L_f \lambda_2 |\ell - u| + \lambda_2 L_f \lambda_1 |\ell - u|\,\big| \\ &\quad = 2 L_f \lambda_1 \lambda_2 |\ell - u|\,. \end{aligned}$$ ∎

To bound the error introduced when interpolating a coarse-scale solution to a finer scale, we apply Lemma 4.1 repeatedly with centre point interpolation ($\lambda_1 = \lambda_2 = 1/2$). This bounds the error between an exact fine-scale discretization $x_s^*[j] = f(t_s[j])$ and the linear interpolation $x_s = \underline{x}_{s+1}$ obtained from a coarser discretization $x_{s+1}[i] = f(t_{s+1}[i])$. See Figure 7 (top).

Lemma 4.3 (Exact Interpolation). *Let $f : [\ell, u] \to \mathbb{R}$ be L_f-Lipschitz continuous. On the interval $[\ell, u]$, let t_s be a fine grid with $J = 2I - 1$ points, and t_{s+1} be a coarse grid with I points. Discretize the function exactly at both scales to obtain vectors $x_s^* \in \mathbb{R}^J$ and $x_{s+1}^* \in \mathbb{R}^I$ where $x_s^*[j] = f(t_s[j])$ and $x_{s+1}^*[i] = f(t_{s+1}[i])$. Let $x_{s+1} = x_{s+1}^* \in \mathbb{R}^I$ and linearly interpolate to obtain $x_s = \underline{x}_{s+1} \in \mathbb{R}^J$ where*

$$\underline{x}_{s+1}[j] = \begin{cases} x_{s+1}\left[\frac{j+1}{2}\right] & \text{if } j \text{ is odd,} \\ \frac{1}{2}\left(x_{s+1}\left[\frac{j}{2}\right] + x_{s+1}\left[\frac{j}{2} + 1\right]\right) & \text{if } j \text{ is even.} \end{cases}$$

Then the difference between the interpolated x_s and exact values x_s^* is bounded by

$$\|x_s - x_s^*\|_2 \leq L_f \frac{1}{2\sqrt{I-1}} |u - \ell|\,. \tag{4.2}$$

Proof. See subsection SM3.2. ∎

When the coarse discretization contains error, the interpolation analysis extends naturally. Suppose we interpolate not from exact values $x_{s+1}[i] = x_{s+1}^*[i] = f(t_{s+1}[i])$, but from *approximate values* $x_{s+1}[i] = x_{s+1}^*[i] + \delta[i] = f(t_{s+1}[i]) + \delta[i]$, where δ is an I-vector that represents the error at the coarse scale. Figure 7 (bottom) visualizes this scenario.

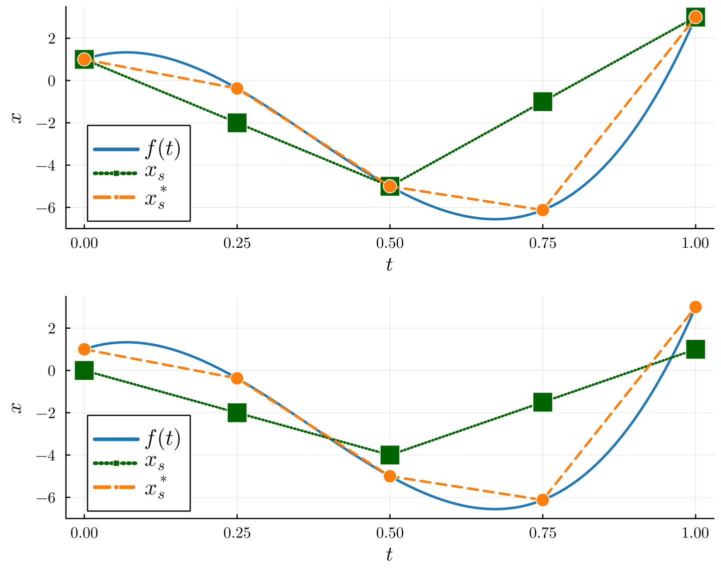


Fig. 7: Illustration of Lemma 4.3 (top) and Lemma 4.4 (bottom) for the function $f(t) = 72t^3 - 80t^2 + 10t + 1$. Both lemmas bound the error between the true fine-scale discretization x_s^* (orange circles) and the linear interpolation $x_s = \underline{x}_{s+1}$ (green squares) obtained from a coarser-scale discretization. The top figure shows (4.2), where $x_{s+1} = x_{s+1}^*$ is exact. The bottom figure shows (4.3), where $x_{s+1} = x_{s+1}^* + \delta$ contains error.

Lemma 4.4 (Inexact Interpolation). *Consider the setup of Lemma 4.3 with one modification: the coarse discretization $x_{s+1} = x_{s+1}^* + \delta$ contains error $\delta \in \mathbb{R}^I$ before we interpolate to obtain $x_s = \underline{x}_{s+1}$. The difference between the interpolated values x_s and exact values x_s^* satisfies*

$$\left\|x_s - x_s^*\right\|_2 \leq \sqrt{2}\left\|\delta\right\|_2 + L_f \frac{1}{2\sqrt{I-1}}\left|u - \ell\right|. \tag{4.3}$$

Proof. See subsection SM3.3. ∎

This bound decomposes into two terms corresponding to distinct error sources. The first term captures the error from interpolating approximate values $x_{s+1} = x_{s+1}^* + \delta$ rather than exact values x_{s+1}^*. The second term captures the linear interpolation error as in Lemma 4.3.

4.2. Greedy Multiscale. We use the following approach to show convergence of the greedy multiscale algorithm described in Algorithm 2.2. At each scale s, we use the iterate convergence (Definition 2.5) to bound the error between our iterate $x_s^{K_s}$ and the solution x_s^* at that scale. Then we use the inexact interpolation lemma (Lemma 4.4) to bound the error between our iterate $x_s^{K_s}$ and interpolation $\underline{x}_S^{K_S}$. This interpolated iterate becomes the initialization $x_{S-1}^0 = \underline{x}_S^{K_S}$ at the next finest scale.

Theorem 4.5 (Greedy Multiscale Error Bound). *Let q denote the rate of convergence of an algorithm applied to a loss function $\tilde{\mathcal{L}}$, let L_{f^*} be the Lipschitz constant of the solution function f^*, and let S denote the coarsest scale. The greedy multiscale* *method in Algorithm* 2.2 *returns iterate $x_1^{K_1}$ at the finest scale satisfying*

$$\left\|x_1^{K_1} - x_1^*\right\|_2 \leq \sqrt{2^{S-1}}(q)^{r_S} \left\|x_S^0 - x_S^*\right\|_2 + L_{f^*} \frac{|u-\ell|}{2\sqrt{2^{S+1}}} \sum_{s=1}^{S-1} 2^s (q)^{r_s},$$

where $r_s = \sum_{t=1}^{s} K_t$ denotes the cumulative iteration count through scale s.

Proof. See subsection SM4.1. ∎

4.3. Lazy Multiscale. We establish convergence of the lazy multiscale algorithm using the framework from Theorem 4.5. The key modification is to freeze the interpolated values $\underline{x}_s^{K_s}$ that correspond to points from the previous iteration $x_s^{K_s}$. The algorithm updates only the newly interpolated values, denoted $x_{(s)}^k$ (see Definition 2.3).

THEOREM 4.6 (Lazy Multiscale Error Bound). *Let q denote the convergence rate of an algorithm with iterate convergence on a loss function $\tilde{\mathcal{L}}$. Under the setting of subsection* 4.1*, the lazy multiscale method satisfies*

$$\begin{aligned}\left\|x_1^{K_1} - x_1^*\right\| \leq{}& \left\|x_S^0 - x_S^*\right\| (q)^{K_S} \prod_{s=1}^{S-1} (1 + (q)^{K_s}) \\ &+ \tfrac{1}{2} L_{f^*} |u-\ell| \sum_{s=1}^{S-1} \frac{1}{\sqrt{2^{S-s}}} (q)^{K_s} \prod_{j=1}^{s-1} \left(1 + (q)^{K_j}\right).\end{aligned}$$

When using constant iterations $K_s = K$ at each scale, this reduces to

$$\begin{aligned}\left\|x_1^{K_1} - x_1^*\right\| \leq{}& \left\|x_S^0 - x_S^*\right\| d(K)(d(K)+1)^{S-1} \\ &+ \tfrac{1}{2} L_{f^*} |u-\ell| \frac{d(K)\left(\sqrt{2^S}\,(d(K)+1)^S - \sqrt{2}\,(d(K)+1)\right)}{\sqrt{2^S}\,(d(K)+1)\left(\sqrt{2}d(K) + \sqrt{2} - 1\right)}\end{aligned}$$

where $d(K) = (q)^K \in (0,1)$.

Proof. See subsection SM4.2. ∎

We summarize the convergence with Corollary 4.7.

COROLLARY 4.7 (Multiscale Convergence). *Suppose the number of iterations K_1 at the finest scale grows unbounded $K_1 \to \infty$. Then under the settings described by Theorems* 4.5 *and* 4.6*, both multiscale algorithms (Algorithms* 2.2 *and* 2.3*) converge to a solution x_1^* that solves* ($\mathrm{P_s}$) *at the finest scale $s = 1$.*

5. Relation Between the Discretized and Continuous Problems.

This section relates solutions of the discretized and continuous problems ($\mathrm{P_s}$) and (P). We show that a small error $\|x_1^{K_1} - x^*\|_2$ in the fine-scale discretized problem solution implies a small error $\|\hat{f}_{x_1^{K_1}} - f^*\|_2$ for the continuous problem, where f^* solves (P) and $\hat{f}_{x_1^{K_1}}(t)$ denotes the piecewise linear function constructed from $x_1^{K_1}$ according to Definition 2.4.

We assume $t \in \mathbb{R}^I$ forms a uniform grid on the interval $[\ell, u]$ with spacing $\Delta t = t[i+1] - t[i] = \frac{u-\ell}{I-1}$.

LEMMA 5.1 (Distance Between Piecewise Linear Functions). *Let $f, g : \mathbb{R} \to \mathbb{R}$, and let $x, y \in \mathbb{R}^I$ be samples $x[i] = f(t[i])$ and $y[i] = g(t[i])$. Construct piecewise linear approximations (Definition* 2.4*) $\hat{f}$ of f and $\hat{g}$ of g. Then*

$$\|\hat{f} - \hat{g}\|_2^2 \leq \tfrac{2}{3} \Delta t \|x - y\|_2^2.$$

Proof. See subsection SM5.1. ∎

We bound the L^2 distance between a Lipschitz function and its piecewise linear approximation.

Lemma 5.2 (Piecewise Linear Function Approximation). *Let f be an L_f-Lipschitz function on $[\ell, u]$, and let $\hat{f}$ denote its piecewise linear approximation (Definition 2.4) on the uniform grid $t \in \mathbb{R}^I$. Then*

$$\|\hat{f} - f\|_2^2 \leq \tfrac{2}{15}(u-\ell)\Delta t^2 L_f^2.$$

Proof. See subsection SM5.2. ∎

Lemma 5.2 compares favourably with existing results. De Boor [7, Ch. III, Eq. 17] establishes the bound

$$\|\hat{f} - f\|_2^2 \leq \tfrac{1}{16}\Delta t^4 \|f''\|_2^2,$$

which converges faster as the interval width $\Delta t \to 0$ (equivalently, $I \to \infty$), but requires $f \in C^2([\ell, u])$ with bounded second derivative. Lipschitz continuity and second-order differentiability doesn't suffice: for example, the parametrized soft-plus function $f(x) = \ln(1 + \exp(cx))/c$ is 1-Lipschitz for all $c > 0$, but has unbounded second derivative $f''(0) = c/2$ as $c \to \infty$.

De Boor showed that the error vanishes for continuous functions (*not* necessarily Lipschitz), though possibly a slower rate than Lemma 5.2 [7, Ch. III, Eq. 18]. Kunoth et al. [14, Sec. 1.5, Th. 18] generalizes the approximation error to functions in Sobolev spaces, achieving for *differentiable* Lipschitz functions a rate comparable to Lemma 5.2:

$$\|\hat{f} - f\|_2^2 \leq K\Delta t^2 \|f'\|_2^2 \tag{5.1}$$

for some constant $K > 0$.

We cannot expect a better rate because Lipschitz functions are differentiable almost everywhere [12]. For Lipschitz functions (possibly not differentiable), the derivative, where it exists, satisfies $|f'(x)| \leq L_f$, giving $\|f'\|_2^2 \leq (u-\ell)L_f^2$. Thus, Lemma 5.2 explicitly calculates the constant $K = 2/15$ in (5.1).

Theorem 5.3 (Connection Between the Continuous and Discrete Problems). *Let $f : [\ell, u] \to \mathbb{R}$ be L_f-Lipschitz with discretization $x^* \in \mathbb{R}^I$. Let $x \in \mathbb{R}^I$, and construct the corresponding piecewise linear function $\hat{f}_x$ (Definition 2.4). For $\epsilon > 0$, if*

$$I > C\epsilon^{-1} + 1 \quad \textit{and} \quad \|x - x^*\|_2^2 < D\epsilon$$

then

$$\|\hat{f}_x - f\|_2 < \epsilon,$$

where $C = \sqrt{\frac{8}{15}}(u-\ell)^{3/2}L_f$ and $D = \sqrt{\frac{3}{40}}(u-\ell)^{1/2}L_f$.

Proof. See subsection SM5.3. ∎

Theorem 5.3 shows that an approximate solution x to the discrete problem ($\mathrm{P_s}$) constructs a piecewise linear function $\hat{f}_x$ that approximately solves the continuous problem (P), provided the grid is sufficiently fine. The constant C grows with the Lipschitz constant L_f, which reflects the expected behaviour that functions with greater variation require finer grids.

6. Comparison with Projected Gradient Descent. Corollary 4.7 established that two versions of the multiscale algorithm (lazy and greedy) converge to a solution of ($\mathrm{P_s}$). We now demonstrate that this convergence improves upon solving ($\mathrm{P_s}$) only at the finest scale $s = 1$. We compare against projected gradient descent with fixed stepsize $\alpha = 2/(\mathcal{S}_{\tilde{\mathcal{L}}} + \mu_{\tilde{\mathcal{L}}})$, where the loss $\tilde{\mathcal{L}}$ is $\mathcal{S}_{\tilde{\mathcal{L}}}$-smooth and $\mu_{\tilde{\mathcal{L}}}$-strongly convex (see Lemma 2.6). While more sophisticated approaches exist for minimizing smooth and strongly convex objectives [18], and these conditions may be relaxed [16], we focus on demonstrating that optimizing over multiple scales minimizes an objective faster than optimization at a single scale. This comparison extends in principle to other methods.

Lemma 2.6 establishes iterate convergence (cf. Definition 2.5) for projected gradient descent. We use this to derive expected convergence bounds for three approaches: projected gradient descent at scale $s = 1$ (Theorem 6.1), greedy multiscale descent (Theorem 6.3), and lazy multiscale descent (Theorem 6.8). After presenting these results, we demonstrate that both multiscale variants achieve tighter error bounds with lower computational cost when the problem size and Lipschitz constant are sufficiently large.

THEOREM 6.1 (Expected Projected Gradient Descent Convergence). *Consider problem* ($\mathrm{P_s}$) *with* $I = 2^S + 1$ *discretization points for some* $S \geq 1$*, solved at the finest resolution (scale* $s = 1$*). Assume solutions* $x_1^* \in \mathbb{R}^I$ *are either normalized* $\|x_1^*\|_2 = 1$ *or centered* $\|x_1^*\|_2 = 0$*. Given initialization* $x_1^0 \in \mathbb{R}^I$ *with i.i.d. standard normal entries* $x_1^0[i] \sim \mathcal{N}(0,1)$*, iterating projected gradient descent* K *times yields the expected error*

$$\mathbb{E}\left\|x_1^K - x_1^*\right\| \leq (q)^K \sqrt{2^S + 2}. \tag{6.1}$$

COROLLARY 6.2 (Expected Number of Iterations). *If we iterate projected gradient descent* K *times, where*

$$K \geq \frac{\log(1/\epsilon) + \log(2^S + 2)/2}{-\log q},$$

then the expected error is bounded as

$$\mathbb{E}\left\|x_1^k - x_1^*\right\|_2 \leq \epsilon.$$

Proof. See subsection SM4.3. ∎

THEOREM 6.3 (Expected Greedy Multiscale Convergence). *Assume the same setting as Theorem* 6.1*, but use the greedy multiscale descent method starting at scale* $s = S$*. At the coarsest scale, the expected initial error satisfies*

$$\mathbb{E}\left\|x_S^0 - x_S^*\right\|_2 \leq 2.$$

Performing K_s *iterations at each scale yields the expected final error*

$$\mathbb{E}\|x_1^{K_1} - x_1^*\|_2 \leq (q)^{K_1}\sqrt{2^{S+1}}\left((q)^{r_S} + \frac{L_{f^*}|u-\ell|}{2^{S+2}}\sum_{s=1}^{S-1} 2^s (q)^{r_s}\right) \tag{6.2}$$

where $r_s = \sum_{t=2}^{s} K_t$ *with the base case* $r_1 = 0$.

Proof. See subsection SM4.4. ∎

Determining the iteration count required to achieve a specified accuracy is less direct for multiscale methods than for projected gradient descent. Moreover, a fair comparison requires accounting for computational cost: an iteration at coarse scale S requires fewer floating-point operations than an iteration at the finest scale $s = 1$. We therefore configure each iteration of multiscale to simultaneously achieve lower total cost than fine-scale-only optimization while obtaining a tighter expected error bound. Specifically, we require (6.2) to be smaller than (6.1). This requires costing multiscale in terms of one projected gradient descent iteration at the finest scale $s = 1$.

Lemma 6.4 (Cost of Greedy Multiscale). *Suppose projected gradient descent with K iterations at the finest scale $s = 1$ has total cost $C_{\mathrm{GD}} = AK$, where $A > 0$ is the cost per iteration at that scale. Assume the cost of one projected gradient descent iteration scales polynomially with problem size as $C_s = C(I_s)^p$ for $I_s = 2^{S-s+1} + 1$ points and constants $C > 0$ and $p \geq 1$. Then greedy multiscale descent with K_s iterations at scale s has total cost C_{GM} that satisfies*

$$C_{\mathrm{GM}} \leq A \sum_{s=1}^{S} K_s \left(\frac{3}{5}\right)^{s-1}.$$

Proof. See subsection SM4.5. ∎

The factor $3/5$ is the ratio of the number of points at the two coarsest scales: $I_S/I_{S-1} = 3/5$.

The assumption that iteration cost scales as CI^p for $C > 0$ and $p \geq 1$ holds for common projects and gradient computations. For example, projection of a vector $y \in \mathbb{R}^I$ onto the sphere $\mathcal{S}^{I-1}$, given by

$$\min\left\{ \tfrac{1}{2} \|x - y\|_2^2 \;\middle|\; x \in \mathcal{S}^{I-1} \right\}$$

scales linearly in I ($p = 1$). The least-squares gradient update for $A \in \mathbb{R}^{I\times I}$,

$$\min\left\{ \tfrac{1}{2} \|Ax - y\|_2^2 \;\middle|\; x \in \mathbb{R}^I \right\},$$

scales quadratically in I ($p = 2$).

Lemma 6.4 establishes a selection of iterations counts K_s at each scale such that greedy multiscale is cheaper than projected gradient descent.

Corollary 6.5 (Conditions for Greedy Multiscale to be Cheaper). *Consider projected gradient descent at scale $s = 1$ with K iterations. Greedy multiscale with K_s iterations at scale s achieves lower cost under either iteration plan:*

- *$K_s = \lceil 2K/5 \rceil - 1$ iterations at every scale; or*
- *$K_s = 1$ iterations at every scale except the finest scale, where $K_1 = K - 2$.*

Proof. This follows from Lemma 6.4 combined with bounds on the sums

$$\textstyle\sum_{s=1}^{S} \left(\frac{3}{5}\right)^{s-1} = \frac{1-(3/5)^S}{1-(3/5)} < 5/2$$

and

$$\textstyle\sum_{s=2}^{S} \left(\frac{3}{5}\right)^{s-1} = \frac{3/5-(3/5)^S}{1-(3/5)} < 3/2 < 2.$$ ∎

The analysis in Corollary 6.5 ignores the cost of interpolation and memory allocation when changing iterate dimensions between scales. These costs become negligible as the iteration counts increase. Empirical results in section 7 confirm that greedy multiscale remains cheaper in practice.

Having established conditions that ensure greedy multiscale is cheaper than projected gradient descent, we now establish conditions that ensure tighter expected error bounds.

Corollary 6.6 (Sufficient Conditions for Greedy Multiscale to be Better). *Assume the base algorithm has iterate convergence (Definition 2.5) with rate $q \in (0, 1/2)$. Assume the finest scale problem has $I = 2^S + 1$ points, where the number of scales S satisfies*

$$S \geq \max \left\{ 4,\, \log_2 \left(\frac{L_{f^*} \left| u - \ell \right|}{\sqrt{2}(q)^2(1-2q)(1-\sqrt{2}q)} \right) \right\}.$$

Then greedy multiscale with one iteration $K_s = 1$ at each scale except the finest (where $K_1 = K - 2$) simultaneously achieves lower total cost and a tighter expected final error bound than projected gradient descent with K iterations.

Proof. See subsection SM4.7. ∎

We establish analogous results for lazy multiscale.

Lemma 6.7 (Cost of lazy Multiscale). *Suppose projected gradient descent with K iterations has total cost $C_{\text{GD}} = AK$, where $A > 0$ is the cost per iteration at the finest scale $s = 1$. Under the polynomial cost scaling assumption $C_s = \Theta(I_s^p)$ for $I_s = 2^{S-s+1} + 1$ points and $p \geq 1$, lazy multiscale descent with K_s iterations at scale s has total cost C_{LM} that satisfies*

$$C_{\text{LM}} \leq A \left(\sum_{s=1}^{S-1} 2^{-s} K_s + 2^{-S} 3 K_S \right).$$

With $\left\lceil \frac{4}{5} K \right\rceil - 1$ iterations at each scale,

$$C_{\text{LM}} < C_{\text{GD}}.$$

Proof. See subsection SM4.6 ∎

Theorem 6.8 (Expected Lazy Multiscale Convergence). *Assume the same setting as Theorem 6.1, but use the lazy multiscale starting at coarsest scale S with $I = 2^S + 1$ points at the finest scale. The expected initial error satisfies*

$$\mathbb{E} \left\| x_S^0 - x_S^* \right\|_2 = 2.$$

Performing constant $K_s = K$ iterations at each scale yields the expected final error

$$\begin{aligned} &\mathbb{E} \left\| x_1^K - x_1^* \right\|_2 \\ &\leq d(K) \left(2(d(K)+1)^{S-1} + \frac{L_{f^*}}{2} \left| u - \ell \right| \frac{\left(\sqrt{2^S} \left(d(K)+1 \right)^S - \sqrt{2} \left(d(K)+1 \right) \right)}{\sqrt{2^S} \left(d(K)+1 \right) \left(\sqrt{2} d(K) + \sqrt{2} - 1 \right)} \right), \end{aligned}$$

where $d(K) = (q)^K$ and the solution function f^* is L_{f^*}-Lipschitz on $[\ell, u]$.

Proof. See subsection SM4.8. ∎

We establish sufficient conditions that ensure lazy multiscale achieves both lower cost and tighter error bounds than projected gradient descent.

Corollary 6.9 (Sufficient Conditions for Lazy Multiscale to be Better). *Assume the base algorithm has iterate convergence (Definition 2.5) with rate $q \in (0, 1)$. Assume projected gradient decent runs for $K > 5(\log_q(\sqrt{2} - 1) - 1)/4$ iterations, and lazy multiscale runs for $K' = \lceil 4K/5 \rceil - 1$ iterations at every scale. If the finest scale has at* *least $I = 2^S + 1$ points where*

$$S \geq \frac{\log\left(\frac{q^{-K/5-1}}{1+q^{K'}}\left(2 + \frac{L_{f^*}|u-\ell|}{2(\sqrt{2}(1+q^{K'})-1)}\right)\right)}{\log\left(\frac{\sqrt{2}}{1+q^{K'}}\right)},$$

then lazy multiscale is cheaper and achieves a tighter expected error bounds than projected gradient descent.

Proof. See subsection SM4.9. ∎

Corollaries 6.6 and 6.9 reveal several insights. Both multiscale variants require a minimum number of scales S (equivalently a minimum sized problem I) for the multiscale optimization to outperform single scale methods. Theorem 5.3 establishes that arbitrarily small error between continuous and discretized solutions requires arbitrarily large problem size I. Both corollaries show that the required I increases with the Lipschitz constant L_{f^*} and the domain size $|u - \ell|$, consistent with Theorem 5.3. A subtle requirement is that the base algorithm cannot have iterative convergence with $q = 0$, which would enable solving the discretized problem in one iteration, and thus eliminate the benefit of multiscale.

Corollaries 6.6 and 6.9 differ in their constraints on the convergence rate q. Greedy multiscale with one iteration per scale requires convergence rate $q \in (0, 1/2)$ because interpolation error must remain smaller than the algorithmic progress at each scale. Lazy multiscale, on the other hand, allows $q \in (0, 1)$ but requires a minimum of $K' > \log_q(\sqrt{2} - 1)$ iterations at each scale to sufficiently control error accumulation. This is a mild requirement: two iterations per scale suffice for lazy multiscale to improve on the base algorithm when $0 < q < (\sqrt{2}-1)^{1/2} \approx 0.6436$, and both algorithms require arbitrarily many iterations to drive the final error to zero.

7. Numerical Experiments and Benchmarks.

We evaluate the greedy multiscale method on two density demixing problems and compare it to standard projected gradient descent. These problems seek to recover mixtures of continuous probability densities from noisy samples. We consider synthetic data in subsection 7.2 and geological survey data in subsection 7.3.

The project-gradient multiscale and single-scale algorithms are implemented in the Julia package `BlockTensorFactorization.jl` [20]: the function `factorize` implements projected gradient descent for the Tucker-1 decomposition problem at a single scale, and `multiscale_factorize` implements the greedy multiscale method (Algorithm 2.2). All experiments ran on an Intel Core i7-1185G7 with 32GB of RAM, without parallelization, matching the hardware used for the example experiment in subsection 2.3.

7.1. Density Demixing as Optimization.

Graham et. al. [10] provides a full treatment of this formulation which we summarize here. Additional details are given in section SM6.

Given I mixtures $y_i : \mathcal{D} \subseteq \mathbb{R}^N \to \mathbb{R}_+$ of R source probability density functions $b_r : \mathcal{D} \subseteq \mathbb{R}^N \to \mathbb{R}_+$,

$$y_i(x) = \textstyle\sum_{r\in[R]} a_{i,r} b_r(x),$$

we seek to recover the sources $\{b_r\}$ and their mixing coefficients $a_{i,r}$. We assume $R < I$ and that the source densities $\{b_r\}$ are linearly independent, which ensures this is a well-posed problem.

The continuous problem formulation is

$$\min_{\{a_{i,r}\},\{b_r\}} \tfrac{1}{2} \textstyle\sum_{i\in[I]} \left\| \sum_{r\in[R]} a_{i,r} b_r - y_i \right\|_2^2$$

such that for all $i \in [I]$,

$$1 = \textstyle\sum_{r\in[R]} a_{i,r}, \qquad a_{i,r} \geq 0, \tag{7.1a}$$

$$1 = \textstyle\int_{\mathcal{D}} b_r(x)\, dx, \qquad b_r(x) \geq 0 \quad \forall x \in \mathcal{D},\ \forall r \in [R]. \tag{7.1b}$$

Discretizing yields the Tucker-1 decomposition problem (see Definition SM6.1)

$$\min \left\{ \tfrac{1}{2} \|B \times_1 A - Y\|_F^2 \;\middle|\; A \in \Delta_R^I,\ B \in \Delta_{K_1\dots K_N}^R \right\}$$

where

$$\Delta_R^I = \left\{ A \in \mathbb{R}_+^{I\times R} \;\middle|\; \sum_{r=1}^{R} A[i,r] = 1,\ \forall i \in [I] \right\}, \tag{7.2a}$$

(7.2b)

$$\Delta_{K_1\dots K_N}^R = \left\{ B \in \mathbb{R}_+^{R\times K_1\times\cdots\times K_N} \;\middle|\; \sum_{k_1,\dots,k_N=1}^{K_1\dots K_N} B[r,k_1,\dots,k_N] = 1,\ \forall r \in [R] \right\}.$$

In the geological data example (subsection 7.3), each dimension of b_r is independent. This allows a simpler discretization, so that B becomes a third-order tensor with

$$B \in \Delta_K^{RJ} = \left\{ B \in \mathbb{R}_+^{R\times J\times K} \;\middle|\; \sum_{k=1}^{K} B[r,j,k] = 1,\ \forall (r,j) \in [R]\times[J] \right\},$$

and we reinterpret $Y \in \mathbb{R}_+^{I\times J\times K}$ accordingly.

7.2. Synthetic Data. We generate three source distributions for a synthetic test. Each distribution is a 3-dimensional product of standard continuous distributions; see section SM7 for details, with full code available in this paper's GitHub repository [21]. Benchmarking the two approaches yields the following for the single-scale `factorize` function:

```
BenchmarkTools.Trial: 20 samples with 1 evaluation per sample.
Range (min...max):  1.305 s...3.487 s      │ GC (min...max): 15.07%
Time  (median):     2.412 s                │ GC (median):    27.67%
Time  (mean ± σ):   2.418 s ± 628.420 ms   │ GC (mean ± σ):  25.08%
Memory estimate: 1.46 GiB, allocs estimate: 2189884.
```

and for the factorization at multiple scales with the `multiscale_factorize` function:

```
BenchmarkTools.Trial: 20 samples with 1 evaluation per sample.
Range (min...max):  231.299 ms...1.583 s     │ GC (min...max):  8.10%
Time  (median):     334.698 ms               │ GC (median):    11.61%
Time  (mean ± σ):   455.869 ms ± 313.358 ms  │ GC (mean ± σ):  23.90%
Memory estimate: 366.81 MiB, allocs estimate: 569530.
```

The function `multiscale_factorize` runs roughly seven times faster and uses about one-fourth of the memory. This comparison includes overhead such as interpolating the tensor and repeatedly calling the internal `factorize` function, which represents

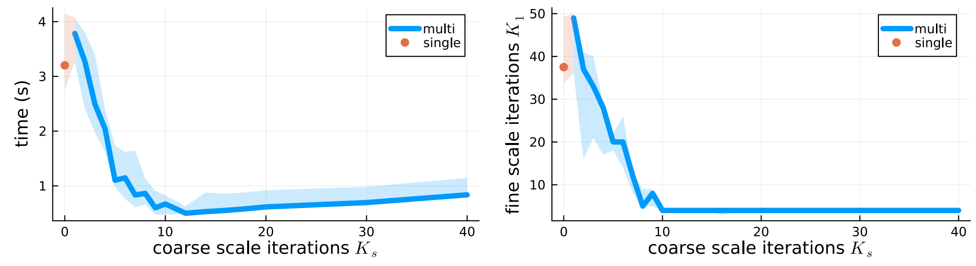


Fig. 8: (Left) Median time until convergence at the finest scale $s = 1$ as a function of number of fixed iterations K_s at coarser scales $s = S, S-1, \ldots, 2$. (Right) Median number of iterations K_1 at the finest scale $s = 1$. Shaded ribbon shows the bottom and top quartile times (left) and iterations (right).

approximately 4% of the total time. The specific numbers are less important than the qualitative comparison: both algorithms could be optimized and benchmarked on state-of-the-art hardware, but this basic implementation demonstrates that multiscale methods can accelerate single-scaled algorithm.

In these `multiscale_factorize` benchmarks, rather than performing one iteration at each scale coarser than the finest scale, we advance to the next scale once sufficient progress has been made; see section SM7 for details. This differs from our analysis in Corollary 6.6, which suggests using exactly $K_s = 1$ at coarse scales. However, running additional iterations at coarser scales often reduces the total number of iterations required at the finest scale.

Figure 8 (left) shows median runtime as a function of the number of fixed coarse iterations K_s in multiscale for $s = S, S-1, \ldots, 2$, where we run iterations at the finest scale until the objective falls below 10^{-6}. This figure demonstrates that performing more than one iteration at coarser scales improves multiscale performance, but only to a point: in Figure 8 (left), more than 12 iterations at each coarse scale wastes time. Figure 8 (right) illustrates this by showing the number of fine-scale iterations K_1 needed to converge to a minimum value. Our implementation of `multiscale_factorize` [20] addresses this by allowing multiple iterations at coarser scales and advancing to the next scale when other criteria are met, such as small gradients or objective values.

Both plots in Figure 8 show that multiscale with exactly $K_s = 1$ iterations at coarse scales $s = S, S-1, \ldots, 2$ performs slightly worse than single-scale optimization. We attribute this to interpolation and to overhead costs in the multiscale approach. The problem size of $2^6 + 1 = 65$ points along each continuous dimension may be too small to see immediate benefit from multiscale when only one iteration is performed at each scale; larger problems may needed to realize the advantage illustrated by Figure 1. However, flexible criteria to advance between scales may make multiscale competitive even for smaller problem.

7.3. Real Geological Data. We use the same sedimentary data and Tucker-1 model described in Graham et. al. [10] to factorize a tensor containing mixtures of estimated densities. Subsection SM7.1 provides the details, and the full code appears in `distribution_unmixing_geology.jl` in this paper's GitHub repository [21]. For `factorize`, we obtain

```
BenchmarkTools.Trial: 20 samples with 1 evaluation per sample.
Range (min...max):  174.316 ms...593.727 ms  │ GC (min...max): 19.75%
Time  (median):     416.816 ms               │ GC (median):    19.91%
Time  (mean ± σ):   389.401 ms ± 108.955 ms  │ GC (mean ± σ):  19.90%
Memory estimate: 361.43 MiB, allocs estimate: 139855.
```

and for `multiscale_factorize`, we observe

```
BenchmarkTools.Trial: 20 samples with 1 evaluation per sample.
Range (min...max):  78.320 ms...153.810 ms  │ GC (min...max):  0.00%
Time  (median):     91.376 ms               │ GC (median):     0.00%
Time  (mean ± σ):   95.542 ms ±  19.154 ms  │ GC (mean ± σ):  11.24%
Memory estimate: 105.80 MiB, allocs estimate: 1444133.
```

By every metric, the multiscale approach is faster and uses less memory than the standard single-scale factorization. Comparing median time and memory estimates, the multiscale method runs roughly four times as fast and uses about a third of the memory. These wall clock-times include significant garbage collection (GC), suggesting that design improvements such as preallocated arrays could reduce memory usage further. Such improvements would accelerate both `factorize` and `multiscale_factorize`. Nevertheless, we observe that `multiscale_factorize` triggers garbage collection less frequently because more computations occur on smaller arrays.

8. Conclusion.

We developed a multiscale approach for optimizing Lipschitz continuous functions by solving discretized problems at progressively finer scales. Our convergence analysis establishes that both greedy and lazy variants achieve the finest-scale solution through per-scale descent combined with controlled interpolation error. The framework extends to linear and norm constraints and to tensor-valued functions.

The explicit error bounds that we derive quantify the tradeoff between discretization granularity and solution accuracy, and prove that multiscale can outperform direct fine-scale optimization when the problem size and Lipschitz constant are sufficiently large.

Several directions merit further investigation. Our analysis assumes a fixed number of base algorithm iterations at each scale. However, an adaptive iteration count that responds to convergence progress could improve efficiency. With regard to the discretization approach, the dyadic scheme might be replaced by adaptive grid refinement based on local Lipschitz estimates. Second-order base algorithms present another avenue, though their cost would be better characterized in terms of linear system solves rather than gradient evaluations. Finally, interpreting the pointwise discretization as a discrete wavelet transform suggests generalizations to other wavelet expansions.

REFERENCES

[1] B. Adcock, M. J. Colbrook, and M. Neyra-Nesterenko, *Restarts Subject to Approximate Sharpness: A Parameter-Free and Optimal Scheme For First-Order Methods*, Foundations of Computational Mathematics, (2025), https://doi.org/10.1007/s10208-024-09673-8.

[2] A. Beck, *First-Order Methods in Optimization*, MOS-SIAM Series on Optimization, Society for Industrial and Applied Mathematics, Oct. 2017, https://doi.org/10.1137/1.9781611974997.

[3] J. J. Benedetto and M. W. Frazier, *Wavelets: Mathematics and Applications*, CRC Press, Boca Raton, 1st ed., 1993.

[4] S. Chakraborty, *Generating discrete analogues of continuous probability distributions-A survey of methods and constructions*, Journal of Statistical Distributions and Applications, 2 (2015), p. 6, https://doi.org/10.1186/s40488-015-0028-6.

[5] Y.-C. Chen, *A tutorial on kernel density estimation and recent advances*, Biostatistics & Epidemiology, 1 (2017), pp. 161–187, https://doi.org/10.1080/24709360.2017.1396742. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/24709360.2017.1396742.

[6] T. Chuna, N. Barnfield, T. Dornheim, M. P. Friedlander, and T. Hoheisel, *Dual formulation of the maximum entropy method applied to analytic continuation of quantum Monte Carlo data*, Jan. 2025, https://doi.org/10.48550/arXiv.2501.01869. arXiv:2501.01869 [physics].

[7] C. De Boor, *A practical guide to splines*, no. v. 27 in Applied mathematical sciences, Springer, New York, rev. ed ed., 2001.

[8] I. Dhillon, P. Ravikumar, and A. Tewari, *Nearest Neighbor based Greedy Coordinate Descent*, in Advances in Neural Information Processing Systems, vol. 24, Curran Associates, Inc., 2011.

[9] I. M. Gelfand and S. V. Fomin, *Calculus of variations*, Dover Publications, Mineola, N.Y, 2000.

[10] N. Graham, N. Richardson, M. P. Friedlander, and J. Saylor, *Tracing Sedimentary Origins in Multivariate Geochronology via Constrained Tensor Factorization*, Mathematical Geosciences, (2025), https://doi.org/10.1007/s11004-024-10175-0.

[11] K. Gurney, *An Introduction to Neural Networks*, CRC Press, London, Oct. 2018, https://doi.org/10.1201/9781315273570.

[12] J. Heinonen, *Lectures On Lipschitz Analysis.* Aug. 2004.

[13] R. Kondor, N. Teneva, and V. Garg, *Multiresolution Matrix Factorization*, in Proceedings of the 31st International Conference on Machine Learning, PMLR, June 2014, pp. 1620–1628.

[14] A. Kunoth, T. Lyche, G. Sangalli, and S. Serra-Capizzano, *Splines and PDEs: From Approximation Theory to Numerical Linear Algebra: Cetraro, Italy 2017*, vol. 2219 of Lecture Notes in Mathematics, Springer International Publishing, Cham, 2018.

[15] I. Loshchilov and F. Hutter, *SGDR: Stochastic Gradient Descent with Warm Restarts*, May 2017, https://doi.org/10.48550/arXiv.1608.03983. arXiv:1608.03983 [cs].

[16] I. Necoara, Y. Nesterov, and F. Glineur, *Linear convergence of first order methods for non-strongly convex optimization*, Mathematical Programming, 175 (2019), pp. 69–107, https://doi.org/10.1007/s10107-018-1232-1.

[17] Y. Nesterov, *Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems*, SIAM Journal on Optimization, 22 (2012), pp. 341–362, https://doi.org/10.1137/100802001. Publisher: Society for Industrial and Applied Mathematics.

[18] Y. Nesterov, *Smooth Convex Optimization*, in Lectures on Convex Optimization, Y. Nesterov, ed., Springer Optimization and Its Applications, Springer International Publishing, Cham, 2018, pp. 59–137, https://doi.org/10.1007/978-3-319-91578-4_2.

[19] S. Ramírez-Gallego, S. García, H. Mouriño-Talín, D. Martínez-Rego, V. Bolón-Canedo, A. Alonso-Betanzos, J. M. Benítez, and F. Herrera, *Data discretization: taxonomy and big data challenge*, WIREs Data Mining and Knowledge Discovery, 6 (2016), pp. 5–21, https://doi.org/10.1002/widm.1173.

[20] N. Richardson, N. Marusenko, and M. P. Friedlander, *BlockTensorFactorization.jl v0.4.0*. https://github.com/MPF-Optimization-Laboratory/BlockTensorFactorization.jl, 2025.

[21] N. Richardson, N. Marusenko, and M. P. Friedlander, *richardson-multiscale-2025*. https://github.com/MPF-Optimization-Laboratory/richardson-multiscale-2025, 2025.

[22] H. J. Stetter, *Analysis of Discretization Methods for Ordinary Differential Equations*, no. 23 in Springer Tracts in Natural Philosophy, Springer, Berlin Heidelberg, 1973, https://doi.org/10.1007/978-3-642-65471-8.

[23] V. Strassen, *Gaussian elimination is not optimal*, Numerische Mathematik, 13 (1969), pp. 354–356, https://doi.org/10.1007/BF02165411.

[24] L. N. Trefethen, *Approximation theory and approximation practice*, no. 164 in Other titles in applied mathematics, Society for Industrial and Applied Mathematics, Philadelphia, extended edition ed., 2019, https://doi.org/10.1137/1.9781611975949.

[25] U. Trottenberg, C. W. Oosterlee, and A. Schuller, *Multigrid Methods*, Academic Press, 2001.

[26] R. Vershynin, *High-Dimensional Probability*, Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press.

[27] Y. Xu and W. Yin, *A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion*, 6, pp. 1758–1789, https://doi.org/10.1137/120887795.

[28] Y. Yan, K. Wang, and P. Rigollet, *Learning Gaussian Mixtures Using the Wasserstein-Fisher-Rao Gradient Flow*, Jan. 2023, https://doi.org/10.48550/arXiv.2301.01766. arXiv:2301.01766 [math].

SUPPLEMENTARY MATERIALS: MULTIPLE SCALE METHODS FOR OPTIMIZATION OF DISCRETIZED CONTINUOUS FUNCTIONS[*]

NICHOLAS J. E. RICHARDSON[†], NOAH MARUSENKO[‡], AND MICHAEL P. FRIEDLANDER[*†]

SM1. Notation and Definition Reference.

SM1.1. Functional Analysis.

Definition SM1.1 (Gradient). *The gradient $\nabla f : \mathbb{R}^{I_1\times\cdots\times I_N} \to \mathbb{R}^{I_1\times\cdots\times I_N}$ of a (differentiable) function $f : \mathbb{R}^{I_1\times\cdots\times I_N} \to \mathbb{R}$ is defined entry-wise for a tensor $A \in \mathbb{R}^{I_1\times\cdots\times I_N}$ by*

$$\nabla f(A)[i_1,\ldots,i_N] = \frac{\partial f}{\partial A[i_1,\ldots,i_N]}(A).$$

Definition SM1.2 (Lipschitz Function). *A function $g : \mathcal{D} \subseteq \mathbb{R}^{I_1\times\cdots\times I_N} \to \mathbb{R}^{J_1\times\cdots\times J_M}$ is L_g-Lipschitz when*

$$\|g(A) - g(B)\|_F \le L_g \|A - B\|_F\,, \quad \forall A, B \in \mathcal{D}.$$

We call the smallest such L_g the *Lipschitz constant of g.*

Definition SM1.3 (Smooth Function). *A differentiable function $f : \mathcal{D} \subseteq \mathbb{R}^{I_1\times\cdots\times I_N} \to \mathbb{R}$ is $\mathcal{S}_f$-smooth when its gradient $g = \nabla f$ is $L_g = \mathcal{S}_f$-Lipschitz,*

$$\|\nabla f(A) - \nabla f(B)\|_F \le \mathcal{S}_f \|A - B\|_F\,, \quad \forall A, B \in \mathcal{D}.$$

A useful fact is that the sum, composition, and (if also bounded) product of Lipschitz functions are also Lipschitz.

Lemma SM1.4 (Product of Lipschitz Functions). *Let $f, g : \mathcal{D} \subseteq \mathbb{R}^{I_1\times\cdots\times I_N} \to \mathbb{R}$ be L_f and L_g Lipschitz respectively. Also assume f and g are bounded with $f(A) \le \|f\|_\infty$ and $g(A) \le \|g\|_\infty$ for all $a \in \mathcal{D}$. Then the product function $(fg)(A) = f(A)g(A)$ is Lipschitz with constant*

$$L_{fg} = L_f \|g\|_\infty + L_g \|f\|_\infty\,.$$

Proof. Let $f, g : \mathcal{D} \to \mathbb{R}^{I_1\times\cdots\times I_N}$, and $A, B \in \mathcal{D}$.

$$\begin{aligned} |f(A)g(A) - f(B)g(B)| &= |f(A)g(A) - f(A)g(B) + f(A)g(B) - f(B)g(B)| \\ &\le |f(A)|\,|g(A) - g(B)| + |g(B)|\,|f(A) - f(B)| \\ &\le |f(A)|\, L_g \|A - B\|_F + |g(B)|\, L_f \|A - B\|_F \\ &\le \|f\|_\infty L_g \|A - B\|_F + \|g\|_\infty L_f \|A - B\|_F \\ &= (\|f\|_\infty L_g + \|g\|_\infty L_f)\, \|A - B\|_F \end{aligned}$$ ∎

We can extend Lemma SM1.4 to tensor-valued functions $f, g : \mathcal{D} \subseteq \mathbb{R}^{I_1\times\cdots\times I_N} \to \mathbb{R}^{J_1\times\cdots\times J_M}$ given any product $f \times g : \mathcal{D} \to \mathbb{R}^{K_1\times\cdots\times K_P}$ that is bilinear,

$$\begin{aligned} f \times (tg + sh) &= t(f \times g) + s(f \times h) \\ (tf + sg) \times h &= t(f \times h) + s(g \times h), \end{aligned}$$

[*]This work was funded by the Natural Sciences and Engineering Research Council of Canada. Manuscript Date: December 11, 2025.

[†]Department of Mathematics, University of British Columbia, Vancouver, BC, Canada

[‡]Department of Computer Science, University of British Columbia, Vancouver, BC, Canada/

for all $t, s \in \mathbb{R}$, and ensures the Frobenius norm is sub-multiplicative,

$$\|f(A) \times g(B)\|_F \leq \|f(A)\|_F \|g(B)\|_F .$$

The functions f and g must still be bounded with

$$\|f(A)\|_F \leq \sup_{A \in \mathcal{D}} \|f(A)\|_F = \|f\|_\infty < \infty,$$

and similarly with g.

SM1.2. Convex Analysis.

DEFINITION SM1.5 (Convex Function). *A function $f : \mathcal{D} \subseteq \mathbb{R}^I \to \mathbb{R}$ is convex when*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y),$$

for all $0 \leq \lambda \leq 1$ and $x, y \in \mathcal{D}$.

When f is differentiable, f is convex when

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle$$

for all $x, y \in \mathcal{D}$.

DEFINITION SM1.6 (Strongly Convex Function). *A function $f : \mathcal{D} \subseteq \mathbb{R}^I \to \mathbb{R}$ is μ_f-strongly convex for $\mu_f > 0$ when $g(x) = f(x) - \frac{\mu_f}{2} \|x\|_2^2$ is convex.*

When f is differentiable, f is μ_f-strongly convex when

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu_f}{2} \|x - y\|_2^2$$

for all $x, y \in \mathcal{D}$.

The vector space $\mathbb{R}^I$ in Definitions SM1.3 and SM1.6 and (2.4) can be extended to tensor spaces $\mathbb{R}^{I_1 \times \cdots \times I_N}$ with the Frobenius inner product $\langle X, Y \rangle_F$ and norm $\|\cdot\|_F$ more generally.

SM2. Constraint Rescaling Proofs. In the following proofs, we have an L_f-Lipschitz function $f : [l, u] \to \mathbb{R}$ that is discretized according to

$$x[i] = f(t[i]),$$

where

$$t[i] = l + (i-1)\Delta t = l + (i-1)\frac{u-l}{I-1}.$$

This ensures neighboring entries of x are close together. Specifically, we have

$$|x_i - x_{i+1}| = |f(t_i) - f(t_{i+1})| \leq L_f |t_i - t_{i+1}| = L_f \Delta t = L_f \frac{u-l}{I-1} := C_x.$$

We can formalize this idea by extending the definition of Lipschitz continuity to vectors.

DEFINITION SM2.1 (Lipschitz Vector). *A vector $x \in \mathbb{R}^I$ is L_x-Lipschitz when, for every $i, j \in [I]$,*

$$|x[i] - x[j]| \leq L_x |i - j| .$$

This is equivalent to the property that neighbouring entries are at most L_x apart.

We also get a corespondent between Lipschitz functions and vectors with this definition.

Corollary SM2.2 (Lipschitz Vectors and Functions). *Given I samples $x[i] = f(t[i])$ of an L_f-Lipschitz function $f : \mathbb{R} \to \mathbb{R}$ on a uniformly spaced grid $\{t[i]\}$, the vector $x \in \mathbb{R}^I$ is $L_x = L_f \Delta t$ Lipschitz.*

Proof. Let $i, j \in [I]$. A uniformly spaced grid on $[l, u]$ will have points $t[i] = l + (i-1)\Delta t$ for $\Delta t = \frac{u-l}{I-1}$. We then have

$$\begin{aligned}
|x[i] - x[j]| &= |f(t[i]) - f(t[j])| \\
&\le L_f \, |t[i] - t[j]| \\
&= L_f \, |l + (i-1)\Delta t - l - (j-1)\Delta t| \\
&= L_f \Delta t \, |i - j| \, . \qquad \square
\end{aligned}$$

SM2.1. Linear Constraint Scaling. Proof of Proposition 3.1.
First, assume an even number of points I.

$$\begin{aligned}
|2\,\langle \overline{a}, \overline{x}\rangle - b| &= \left| 2 \sum_{i \text{ odd}} a_i x_i - \sum_{i=1}^{I} a_i x_i \right| \\
&= \left| \sum_{i \text{ odd}} a_i x_i - \sum_{i \text{ even}} a_i x_i \right| \\
&= \left| \sum_{i \text{ odd}} a_i x_i - \sum_{i \text{ odd}} a_{i+1} x_{i+1} \right| \\
&\le \sum_{i \text{ odd}} |a_i x_i - a_{i+1} x_{i+1}| \\
&\le \sum_{i \text{ odd}} L_{fg} \Delta t \, |i - (i+1)| \\
&= \sum_{i \text{ odd}} L_{fg} \frac{u-l}{I-1} \, |1| \\
&= \frac{I}{2} (L_f \, \|g\|_\infty + L_g \, \|f\|_\infty) \frac{u-l}{I-1}
\end{aligned}$$

The second inequality follows from Corollary SM2.2 in combination with Lemma SM1.4. This uses the fact that $a_i x_i$ are samples of the product $a_i x_i = f(t_i) g(t_i)$ which is Lipschitz with constant

$$L_{fg} = (L_f \, \|g\|_\infty + L_g \, \|f\|_\infty).$$

Dividing the inequality by 2 proves the first statement.

For odd I, we have

$$\begin{aligned}
\left|2\langle\overline{a},\overline{x}\rangle - \frac{I+1}{I}b\right| &= \left|2\sum_{i\text{ odd}} a_i x_i - \frac{I+1}{I}b\right| \\
&= \left|2\sum_{i\text{ odd}} a_i x_i - b - \frac{1}{I}b\right| \\
&= \left|\sum_{i\text{ odd}} a_i x_i + \sum_{i\text{ odd}} a_i x_i - \sum_{i\text{ odd}} a_i x_i - \sum_{i\text{ even}} a_i x_i - \frac{1}{I}b\right| \\
&= \left|\sum_{i\text{ odd}} a_i x_i - \sum_{i\text{ even}} a_i x_i - \frac{1}{I}b\right| \\
&= \left|\sum_{j=1}^{(I+1)/2} a_{2j-1}x_{2j-1} - \sum_{j=1}^{(I-1)/2} a_{2j}x_{2j} - \frac{1}{I}b\right| \\
&= \left|\sum_{j=1}^{(I-1)/2} a_{2j-1}x_{2j-1} + a_I x_I - \sum_{j=1}^{(I-1)/2} a_{2j}x_{2j} - \frac{1}{I}b\right| \\
&\le \left|\sum_{j=1}^{(I-1)/2} a_{2j-1}x_{2j-1} - \sum_{j=1}^{(I-1)/2} a_{2j}x_{2j}\right| + \left|a_I x_I - \frac{1}{I}b\right| \\
&\le \sum_{j=1}^{(I-1)/2} \left|a_{2j-1}x_{2j-1} - a_{2j}x_{2j}\right| + \frac{1}{I}\left|I a_I x_I - b\right| \\
&= \sum_{j=1}^{(I-1)/2} \left|a_{2j-1}x_{2j-1} - a_{2j}x_{2j}\right| + \frac{1}{I}\left|\sum_{i=1}^{I} a_I x_I - \sum_{i=1}^{I} a_i x_i\right| \\
&= \sum_{j=1}^{(I-1)/2} \left|a_{2j-1}x_{2j-1} - a_{2j}x_{2j}\right| + \frac{1}{I}\sum_{i=1}^{I}\left|a_I x_I - a_i x_i\right|
\end{aligned}$$

The first term in the last line is bounded by

$$\sum_{j=1}^{(I-1)/2} \left|a_{2j-1}x_{2j-1} - a_{2j}x_{2j}\right| \le \frac{I-1}{2} L_{fg}\frac{u-l}{I-1} = L_{fg}\frac{u-l}{2}$$

in a similar manner to the case with even I.

The second term has the following bound

$$\begin{aligned}
\frac{1}{I}\sum_{i=1}^{I}\left|a_I x_I - a_i x_i\right| &\le \frac{1}{I}\sum_{i=1}^{I} L_{fg}\Delta t\left|I - i\right| \\
&= \frac{1}{I} L_{fg}\Delta t \sum_{i=1}^{I}(I-i) \\
&= \frac{1}{I} L_{fg}\frac{u-l}{I-1}\frac{I(I-1)}{2} \\
&= L_{fg}\frac{u-l}{2}.
\end{aligned}$$

Putting these terms together, we have

$$\begin{aligned}\left|2\left\langle\overline{a},\overline{x}\right\rangle-\frac{I+1}{I}b\right|&\leq L_{fg}\frac{u-l}{2}+L_{fg}\frac{u-l}{2}\\&=(L_f\left\|g\right\|_\infty+L_g\left\|f\right\|_\infty)(u-l).\end{aligned}$$

Again, dividing by two completes the proof.

SM2.2. Matrix Constraint Scaling. Proof of Corollary 3.2.

Let $a_k=A[k,:]\in\mathbb{R}^I$ be the kth row vector in A. By Proposition 3.1, we have for even I,

$$\begin{aligned}\left\|\overline{A}\overline{x}-\frac{b}{2}\right\|_2^2&=\sum_{k=1}^K\left|\langle\overline{a}_k,\overline{x}\rangle-\frac{b[k]}{2}\right|^2\\&\leq\sum_{k=1}^K\left(\frac{I}{I-1}\frac{L_{fg_k}(u-l)}{4}\right)^2\\&=\left(\frac{I}{I-1}\frac{u-l}{4}\right)^2\sum_{k=1}^K(L_{fg_k})^2\\&\leq\left(\frac{I}{I-1}\frac{u-l}{4}\right)^2\sum_{k=1}^K\max_k\{(L_{fg_k})^2\}\\&=\left(\frac{I}{I-1}\frac{u-l}{4}\right)^2(L_{fg})^2K.\end{aligned}$$

Taking square roots completes this case, and the case for odd I is similar.

SM3. Functional Analysis Lemmas Proofs. In the following proofs, we take the unlabeled norm $\|\cdot\|=\|\cdot\|_2$ to be the 2-norm.

SM3.1. Lipschitz-Interpolation Proof. Proof of Lemma 4.1.

Let $f:\mathbb{R}^n\to\mathbb{R}$ be L_f-Lipschitz, $a,b\in\mathbb{R}^n$, and $\lambda=\lambda_1$, $1-\lambda=\lambda_2$ for $\lambda\in[0,1]$. Our goal is to bound

$$\left|((1-\lambda)f(a)+\lambda f(b))-f((1-\lambda)a+\lambda b)\right|.$$

We achieve the tightest bound[1] by separating

$$f((1-\lambda)a+\lambda b)=(1-\lambda)f((1-\lambda)a+\lambda b)+\lambda f((1-\lambda)a+\lambda b)$$

[1]A naive approach would be to separate $(1-\lambda)f(a)=f(a)-\lambda f(a)$ and separately bound the terms $f(a)-f((1-\lambda)a+\lambda b)$ and $-\lambda f(a)+\lambda f(b)$. But this only gives a bound of $2L_f\left\|a-b\right\|$ at $\lambda=1$ instead of 0. Swapping $\lambda\leftrightarrow 1-\lambda$ (or equivalently $a\leftrightarrow b$) gives a mirrored bound that, when combined with the unswapped variables, can only tighten the bound to $2\min(\lambda,1-\lambda)L_f\left\|a-b\right\|$. The bound $2\lambda(1-\lambda)L_f\left\|a-b\right\|$ is the tightest bound that does *not* depend on the value of the function at the endpoints $f(a)$ and $f(b)$. More precisely, for a given λ, there is an L_f-Lipschitz function that will hit this tighter bound.

and using triangle inequality to get

$$\begin{aligned}
&|((1-\lambda)f(a)+\lambda f(b))-f((1-\lambda)a+\lambda b)| \\
&=|(1-\lambda)f(a)+\lambda f(b)-(1-\lambda)f((1-\lambda)a+\lambda b)-\lambda f((1-\lambda)a+\lambda b)| \\
&\leq |(1-\lambda)f(a)-(1-\lambda)f((1-\lambda)a+\lambda b)|+|\lambda f(b)-\lambda f((1-\lambda)a+\lambda b)| \\
&=(1-\lambda)\left|f(a)-f((1-\lambda)a+\lambda b)\right|+\lambda\left|f(b)-f((1-\lambda)a+\lambda b)\right| \\
&\leq (1-\lambda)L_f\left\|a-((1-\lambda)a+\lambda b)\right\|_2+\lambda L_f\left\|b-((1-\lambda)a+\lambda b)\right\|_2 \\
&=(1-\lambda)L_f\lambda\left\|a-b\right\|_2+\lambda L_f(1-\lambda)\left\|b-a\right\|_2 \\
&=2\lambda(1-\lambda)L_f\left\|a-b\right\|_2.
\end{aligned}$$

SM3.2. Exact Interpolation Proof. Proof of Lemma 4.3.

Let $t_{s+1}\in\mathbb{R}^I$ and $t_s\in\mathbb{R}^J$ be uniform grids on $[\alpha,\beta]$ where $J=2I-1$ and $t_s[1]=t_{s+1}[1]=\alpha$ and $t_s[J]=t_{s+1}[I]=\beta$ at some scale s.

Let $x^*_{s+1}[i]=f(t_{s+1}[i])$ for some L_f-Lipchitz function $f:\mathbb{R}\to\mathbb{R}$, let $x_{s+1}=x^*_{s+1}$, and linearly interpolate the function values

$$\underline{x}_{s+1}[j]=\begin{cases} x_{s+1}\left[\frac{j+1}{2}\right], & \text{if } j \text{ is odd,}\\ \frac{1}{2}\left(x_{s+1}\left[\frac{j}{2}\right]+x_{s+1}\left[\frac{j}{2}+1\right]\right), & \text{if } j \text{ is even,}\end{cases}$$

where the true values are given by $x^*_s[j]=f(t_s[j])$. Our approximate on the finder grid becomes $x_s=\underline{x}_{s+1}$.

We wish to show the difference between the interpolated x_s and exact values x^*_s is bounded by

$$\left\|x_s-x^*_s\right\|_2\leq\frac{L_f}{2\sqrt{I-1}}\left|\alpha-\beta\right|_2.$$

Using Lemma 4.1 with $\lambda=1/2$, we have for even j,

$$\begin{aligned}
|x_s[j]-x^*_s[j]| &= \left|\tfrac{1}{2}\left(x_{s+1}\left[\tfrac{j}{2}\right]+x_{s+1}\left[\tfrac{j}{2}+1\right]\right)-f(t_s\left[j\right])\right| \\
&=\left|\tfrac{1}{2}\left(f\left(t_{s+1}\left[\tfrac{j}{2}\right]\right)+f\left(t_{s+1}\left[\tfrac{j}{2}+1\right]\right)\right)\right. \\
&\qquad \left.-f\left(\tfrac{1}{2}\left(t_{s+1}\left[\tfrac{j}{2}\right]+t_{s+1}\left[\tfrac{j}{2}+1\right]\right)\right)\right| \\
&\leq\tfrac{L}{2}\left|t_{s+1}\left[\tfrac{j}{2}\right]-t_{s+1}\left[\tfrac{j}{2}+1\right]\right| \\
&=\tfrac{L}{2(I-1)}\left|\alpha-\beta\right|.
\end{aligned}$$

For odd j, the interpolated values match the function exactly

$$x_s[j]=\underline{x}_{s+1}[j]=x_{s+1}\left[\tfrac{j+1}{2}\right]=x^*_{s+1}\left[\tfrac{j+1}{2}\right]=f\left(t_{s+1}\left[\tfrac{j+1}{2}\right]\right)=f(t_s[j])=x^*_s[j].$$

Summing over all j, we have a bound on squared error

$$\begin{aligned}
\left\|x_s-x^*_s\right\|_2^2 &= \sum_{j=1}^{J}\left|x_s[j]-x^*_s[j]\right|^2=\sum_{j\text{ even}}\left|x_s[j]-x^*_s[j]\right|^2 \\
&\leq\sum_{j\text{ even}}\frac{L_f^2}{4(I-1)^2}\left|\alpha-\beta\right|^2 \\
&\leq (I-1)\frac{L_f^2}{4(I-1)^2}\left|\alpha-\beta\right|^2 \\
&=\frac{L_f^2}{4(I-1)}\left|\alpha-\beta\right|^2.
\end{aligned}$$

Taking square roots completes the proof.

SM3.3. Inexact Interpolation Proof. Proof of Lemma 4.4.

We let the inexact values be $x_{s+1}[i] = x^*_{s+1}[i] + \delta[i]$, and we interpolate the inexact values to get $x_s = \underline{x}_{s+1}$:

$$\begin{aligned}
\underline{x}_{s+1}[j] &= \begin{cases} x_{s+1}[\frac{j+1}{2}], & \text{if } j \text{ is odd,} \\ \frac{1}{2}\left(x_{s+1}[\frac{j}{2}] + x_{s+1}[\frac{j}{2}+1]\right), & \text{if } j \text{ is even} \end{cases} \\
&= \begin{cases} x^*_{s+1}[\frac{j+1}{2}] + \delta[\frac{j+1}{2}], & \text{if } j \text{ is odd,} \\ \frac{1}{2}\left(x^*_{s+1}[\frac{j}{2}] + x^*_{s+1}[\frac{j}{2}+1]\right) + \frac{1}{2}\left(\delta[\frac{j}{2}] + \delta[\frac{j}{2}+1]\right), & \text{if } j \text{ is even.} \end{cases}
\end{aligned}$$

So for odd j,

$$\left|\underline{x}_{s+1}[j] - \underline{x}^*_{s+1}[j]\right| = \left|\delta\left[\frac{j+1}{2}\right]\right| := e[j],$$

and even j,

$$\left|\underline{x}_{s+1}[j] - \underline{x}^*_{s+1}[j]\right| = \frac{1}{2}\left|\delta\left[\frac{j}{2}\right] + \delta\left[\frac{j}{2}+1\right]\right| := e[j].$$

Note $\underline{x}^*_{s+1}$ is the interpolated vector of x^*_{s+1} where $x^*_{s+1}[i] = f(t_{s+1}[i])$, and is possibly different from the exact discretization $x^*_s[j] = f(t_s[j])$. Of course the entries $\underline{x}^*_{s+1}[j] = x^*_s[j]$ for odd j.

So we have

$$\left\|\underline{x}_{s+1} - \underline{x}^*_{s+1}\right\|_2 = \sqrt{\sum_{j\in[J]} \left|x_s[j] - \underline{x}^*_{s+1}[j]\right|^2} = \sqrt{\sum_{j\in[J]} |e[j]|^2} = \|e\|_2$$

We would like some bound on $\|e\|$ in terms of the closeness of x_{s+1} and x^*_{s+1}, i.e. in terms of δ.

$$
\begin{aligned}
\|e\|^2 &= \sum_{j \text{ odd}} |e[j]|^2 + \sum_{j \text{ even}} |e[j]|^2 \\
&= \sum_{j \text{ odd}} \left|\delta\left[\frac{j+1}{2}\right]\right|^2 + \sum_{j \text{ even}} \left|\frac{1}{2}\left(\delta\left[\frac{j}{2}\right] + \delta\left[\frac{j}{2}+1\right]\right)\right|^2 \\
&= \sum_{i=1}^{I} |\delta[i]|^2 + \frac{1}{4}\sum_{i=1}^{I-1} (\delta[i] + \delta[i+1])^2 \\
&= \|\delta\|^2 + \frac{1}{4}\sum_{i=1}^{I-1} \left(\delta[i]^2 + \delta[i+1]^2 + 2\delta[i]\delta[i+1]\right) \\
&= \|\delta\|^2 + \frac{1}{4}\left(\sum_{i=1}^{I-1} \delta[i]^2 + \sum_{i=1}^{I-1} \delta[i+1]^2 + 2\sum_{i=1}^{I-1} \delta[i]\delta[i+1]\right) \\
&\leq \|\delta\|^2 + \frac{1}{4}\left(\sum_{i=1}^{I} \delta[i]^2 + \sum_{i=0}^{I-1} \delta[i+1]^2 + 2\sum_{i=1}^{I-1} \delta[i]\delta[i+1]\right) \\
&= \|\delta\|^2 + \frac{1}{4}\left(\|\delta\|^2 + \|\delta\|^2 + 2\sum_{i=1}^{I-1} \delta[i]\delta[i+1]\right) \\
&= \frac{3}{2}\|\delta\|^2 + \frac{1}{2}\sum_{i=1}^{I-1} \delta[i]\delta[i+1] \\
&\leq \frac{3}{2}\|\delta\|^2 + \frac{1}{2}\sqrt{\sum_{i=1}^{I-1} \delta[i]^2}\sqrt{\sum_{i=1}^{I-1} \delta[i+1]^2} \quad \text{(Cauchy–Schwarz)} \\
&\leq \frac{3}{2}\|\delta\|^2 + \frac{1}{2}\sqrt{\sum_{i=1}^{I} \delta[i]^2}\sqrt{\sum_{i=0}^{I-1} \delta[i+1]^2} \\
&= \frac{3}{2}\|\delta\|^2 + \frac{1}{2}\|\delta\|\,\|\delta\| \\
&= 2\|\delta\|^2
\end{aligned}
\tag{SM3.1}
$$

Therefore,

$$\|e\| \leq \sqrt{2}\,\|\delta\|$$

or substituting our notation,

$$\left\|\underline{x}_{s+1} - \underline{x}^*_{s+1}\right\| \leq \sqrt{2}\left\|x_{s+1} - x^*_{s+1}\right\|.$$

This is saying that the error in our fine grid e is bounded by a factor of $\sqrt{2}$ of the error in the coarse grid δ when we (nearly) double the number of points.

Now we can use the triangle inequality to bound the difference between the interpolated approximate values $x_s = \underline{x}_{s+1}$ and the true values on the finer grid x^*_s:

$$
\begin{aligned}
\|x_s - x^*_s\|_2 &\leq \left\|x_s - \underline{x}^*_{s+1}\right\|_2 + \left\|\underline{x}^*_{s+1} - x^*_s\right\|_2 \\
&= \left\|\underline{x}_{s+1} - \underline{x}^*_{s+1}\right\|_2 + \left\|\underline{x}^*_{s+1} - x^*_s\right\|_2 \\
&\leq \sqrt{2}\left\|x_{s+1} - x^*_{s+1}\right\|_2 + \frac{L_f}{2\sqrt{I-1}}|u - \ell|.
\end{aligned}
$$

SM4. Multi-scale Convergence Proofs. In the following proofs, we take the unlabeled norm $\|\cdot\| = \|\cdot\|_2$ to be the 2-norm.

SM4.1. Greedy Multi-scale Descent Error Proof. Proof of Theorem 4.5.

Using Lemma 4.4, for an L_{f^*}-Lipchitz function f^* discretized with I points, we have

$$\|x_s - x_s^*\| \leq \sqrt{2}\left\|x_{s+1} - x_{s+1}^*\right\| + \frac{L_{f^*}|u-\ell|}{2\sqrt{I-1}}.$$

Translating this into our notation for the multi-scaled descent, we have

$$\left\|x_{s-1}^0 - x_{s-1}^*\right\| = \left\|\underline{x}_s^{K_s} - x_{s-1}^*\right\| \leq \sqrt{2}\left\|x_s^{K_s} - x_s^*\right\| + \frac{L_{f^*}|u-\ell|}{2\sqrt{2^{S-s+1}}}.$$

Note that at scale s, we have $2^{S-s+1}+1$ many points in our discretization, where we have 2^S+1 many points in our finest scale, and the discretization is over the interval $a \leq t \leq b$ for an L_{f^*}-Lipschitz function f^*.

Combining this with our convergence for gradient descent gives us the inequality

$$\left\|x_{s-1}^0 - x_{s-1}^*\right\| \leq \sqrt{2}\left\|x_s^{K_s} - x_s^*\right\| + \frac{L_{f^*}|u-\ell|}{2\sqrt{2^{S-s+1}}} \leq \sqrt{2}(q)^{K_s}\left\|x_s^0 - x_s^*\right\| + \frac{L|u-\ell|}{2\sqrt{2^{S-s+1}}}.$$

Writing it in terms of $s+1$ give the descent

$$\left\|x_s^0 - x_s^*\right\| \leq \sqrt{2}(q)^{K_{s+1}}\left\|x_{s+1}^0 - x_{s+1}^*\right\| + \sqrt{2^s}\frac{L_{f^*}|u-\ell|}{2\sqrt{2^S}}.$$

Let $e_s = \left\|x_s^0 - x_s^*\right\|$ and $C = \frac{L_{f^*}|u-\ell|}{2\sqrt{2^{S+1}}}$ to clean up notation

$$e_s \leq \sqrt{2}(q)^{K_{s+1}}e_{s+1} + \sqrt{2^{s+1}}C.$$

Now we recurse! Starting with the final error at the finest scale $\left\|x_1^{K_1} - x_1^*\right\|$, we have

$$\begin{aligned}
&\left\|x_1^{K_1} - x_1^*\right\| \\
&\leq (q)^{K_1} \left\|x_1^0 - x_1^*\right\| \\
&= (q)^{K_1} e_1 \\
&\leq (q)^{K_1} \left(\sqrt{2}(q)^{K_2} e_2 + \sqrt{2^2}C\right) \\
&= \sqrt{2}(q)^{K_1+K_2} e_2 + (q)^{K_1}\sqrt{2^2}C \\
&\leq \sqrt{2}(q)^{K_1+K_2}\left(\sqrt{2}(q)^{K_3} e_3 + \sqrt{2^3}C\right) + (q)^{K_1}\sqrt{2^2}C \\
&= \sqrt{2}^2(q)^{K_1+K_2+K_3} e_3 + \sqrt{2}(q)^{K_1+K_2}\sqrt{2^3}C + (q)^{K_1}\sqrt{2^2}C \\
&\leq \sqrt{2}^2(q)^{K_1+K_2+K_3}\left(\sqrt{2}(q)^{K_4} e_4 + \sqrt{2^4}C\right) + \sqrt{2}(q)^{K_1+K_2}\sqrt{2^3}C + (q)^{K_1}\sqrt{2^2}C \\
&= \sqrt{2}^3(q)^{K_1+K_2+K_3+K_4} e_4 + \sqrt{2}^2(q)^{K_1+K_2+K_3}\sqrt{2^4}C \\
&\quad + \sqrt{2}(q)^{K_1+K_2}\sqrt{2^3}C + (q)^{K_1}\sqrt{2^2}C \\
&\vdots \\
&\leq \sqrt{2}^{S-1}(q)^{\sum_{s=1}^S K_s} e_S + \sum_{s=1}^{S-1} \sqrt{2}^{s-1}(q)^{\sum_{t=1}^s K_t}\sqrt{2^{s+1}}C \\
&= \sqrt{2^{S-1}}(q)^{\sum_{s=1}^S K_s} e_S + C\sum_{s=1}^{S-1} 2^s (q)^{\sum_{t=1}^s K_t}.
\end{aligned}$$

Substituting back $e_s = \left\|x_s^0 - x_s^*\right\|$ and $C = \frac{L_{f^*}|u-\ell|}{2\sqrt{2^{S+1}}}$ gives us the result

$$\left\|x_1^{K_1} - x_1^*\right\| \leq \sqrt{2^{S-1}}(q)^{\sum_{s=1}^S K_s}\left\|x_S^0 - x_S^*\right\| + \frac{L_{f^*}|u-\ell|}{2\sqrt{2^{S+1}}}\sum_{s=1}^{S-1} 2^s (q)^{\sum_{t=1}^s K_t}.$$

SM4.2. Lazy Multi-scale Descent Error Proof. Proof of Theorem 4.6. We first prove the following lemma.

LEMMA SM4.1 (Inexact Lazy Interpolation). *Let $e = x - x^*$ be the error between x and the solution values $x^*[i] = f^*(t[i])$, and let $e_{(s)}$ be the error in the newly added points at the scale s (see Definition 2.3). In the lazy multi-scale setting for an L_f-Lipchitz function f on $[\ell, u]$, we have the interpolated error bound at scale s,*

$$\left\|e_{(s-1)}\right\| \leq \frac{L_{f^*}|u-\ell|}{2\sqrt{2^{S-s}}} + \left\|e_s\right\|.$$

Proof. The proof is similar to Lemma 4.4 for greedy multi-scale and differs by a factor of $\sqrt{2}$ on the error term $\|e_s\|$. This can be shown by modifying (SM3.1) to exclude the odd indexes since we only look at the error for the newly interpolated points. ∎

Using Lemma SM4.1, we have

$$\left\|e_{(s)}^0\right\| \leq \frac{L_{f^*}|u-\ell|}{2\sqrt{2^{S-s}}} + \left\|e_{s+1}^{K_{s+1}}\right\|.$$

Putting this back into the iteration convergence equation (Definition 2.5) gets us

$$\left\| e_{(s)}^{K_s} \right\| \leq (q)^{K_s} \left(\frac{L_{f^*} \left| u - \ell \right|}{2\sqrt{2^{S-s}}} + \left\| e_{s+1}^{K_{s+1}} \right\| \right).$$

And finally, we get the recursion relation

$$\left\| e_s^{K_s} \right\|^2 \leq (q)^{2K_s} \left(\frac{L_{f^*} \left| u - \ell \right|}{2\sqrt{2^{S-s}}} + \left\| e_{s+1}^{K_{s+1}} \right\| \right)^2 + \left\| e_{s+1}^{K_{s+1}} \right\|^2.$$

We will use big $C = \frac{L_{f^*}}{2} \left| u - \ell \right|$ to clean up the notation.

$$\left\| e_s^{K_s} \right\|^2 \leq (q)^{2K_s} \left(\frac{C}{\sqrt{2^{S-s}}} + \left\| e_{s+1}^{K_{s+1}} \right\| \right)^2 + \left\| e_{s+1}^{K_{s+1}} \right\|^2$$

We can remove the squares because, for $a, b, c \geq 0$,

$$a^2 \leq b^2 + c^2 \leq b^2 + 2bc + c^2 = (b + c)^2 \implies a \leq b + c.$$

So we are left with a looser bound,

$$\begin{aligned}\left\| e_s^{K_s} \right\| &\leq (q)^{K_s} \left(\frac{C}{\sqrt{2^{S-s}}} + \left\| e_{s+1}^{K_{s+1}} \right\| \right) + \left\| e_{s+1}^{K_{s+1}} \right\| \\ &= \left(1 + (q)^{K_s}\right) \left\| e_{s+1}^{K_{s+1}} \right\| + C(q)^{K_s} \frac{1}{\sqrt{2^{S-s}}}.\end{aligned}$$

We have the relation

$$\left\| e_s^{K_s} \right\| \leq \left(1 + (q)^{K_s}\right) \left\| e_{s+1}^{K_{s+1}} \right\| + C(q)^{K_s} \frac{1}{\sqrt{2^{S-s}}}$$

for every $s = 1, \ldots, S$. So let's expand this.

$$\begin{aligned}\left\| e_1^{K_1} \right\| &\leq \left(1 + (q)^{K_1}\right) \left\| e_2^{K_2} \right\| + C(q)^{K_1} \frac{1}{\sqrt{2^{S-1}}} \\ &\leq \left(1 + (q)^{K_1}\right) \left(\left(1 + (q)^{K_2}\right) \left\| e_3^{K_3} \right\| + C(q)^{K_2} \frac{1}{\sqrt{2^{S-2}}} \right) + C(q)^{K_1} \frac{1}{\sqrt{2^{S-1}}} \\ &= \left(1 + (q)^{K_1}\right) \left(1 + (q)^{K_2}\right) \left\| e_3^{K_3} \right\| \\ &\quad + \ C \left(1 + (q)^{K_1}\right) (q)^{K_2} \frac{1}{\sqrt{2^{S-2}}} + C(q)^{K_1} \frac{1}{\sqrt{2^{S-1}}} \\ &\leq \left(1 + (q)^{K_1}\right) \left(1 + (q)^{K_2}\right) \left(\left(1 + (q)^{K_3}\right) \left\| e_4^{K_4} \right\| + C(q)^{K_3} \frac{1}{\sqrt{2^{S-3}}} \right) \\ &\quad + \ C \left(1 + (q)^{K_1}\right) (q)^{K_2} \frac{1}{\sqrt{2^{S-2}}} + C(q)^{K_1} \frac{1}{\sqrt{2^{S-1}}} \\ &= \left(1 + (q)^{K_1}\right) \left(1 + (q)^{K_2}\right) \left(1 + (q)^{K_3}\right) \left\| e_4^{K_4} \right\| \\ &\quad + \ C \left(1 + (q)^{K_1}\right) \left(1 + (q)^{K_2}\right) (q)^{K_3} \frac{1}{\sqrt{2^{S-3}}} \\ &\quad + \ C \left(1 + (q)^{K_1}\right) (q)^{K_2} \frac{1}{\sqrt{2^{S-2}}} + C(q)^{K_1} \frac{1}{\sqrt{2^{S-1}}}\end{aligned}$$

So the general formula goes to

$$\left\| e_1^{K_1} \right\| \leq \left\| e_S^{K_S} \right\| \prod_{s=1}^{S-1} (1 + (q)^{K_s}) + C \sum_{s=1}^{S-1} \frac{1}{\sqrt{2^{S-s}}} (q)^{K_s} \prod_{j=1}^{s-1} \left(1 + (q)^{K_j}\right).$$

After performing gradient descent on the coarsest scale, we get

$$\left\| e_1^{K_1} \right\| \leq \left\| e_S^{0} \right\| (q)^{K_S} \prod_{s=1}^{S-1} (1 + (q)^{K_s}) + C \sum_{s=1}^{S-1} \frac{1}{\sqrt{2^{S-s}}} (q)^{K_s} \prod_{j=1}^{s-1} \left(1 + (q)^{K_j}\right).$$

This gives us the first upper bound for any plan of iterations K_s. If we now assume each scale uses the name number of iterations $K_s = K$ we get the following.

$$\left\| e_1^{K_1} \right\| \leq \left\| e_S^{0} \right\| (q)^{K} (1 + (q)^{K})^{S-1} + C \sum_{s=1}^{S-1} \frac{1}{\sqrt{2^{S-s}}} (q)^{K} (1 + (q)^{K})^{s-1}.$$

Via WolframAlpha, we have

$$\begin{aligned}
&\sum_{s=1}^{S-1} \frac{1}{\sqrt{2^{S-s}}} (q)^K ((q)^K + 1)^{s-1} \\
&= \frac{(q)^K \left(\sqrt{2^S} \left((q)^K + 1\right)^S - \sqrt{2}\left((q)^K + 1\right) \right)}{\sqrt{2^S}\left((q)^K + 1\right)\left(\sqrt{2}(q)^K + \sqrt{2} - 1\right)}
\end{aligned}$$

Setting $d(K) = (q)^K$ gives the final upper bound in the theorem.

SM4.3. Expected Projected Gradient Descent Convergence Proof. Proof of Theorem 6.1.

Case 1: $\|x_1^*\|_2 = 1$.

Without loss of generality, assume (by symmetry) that $x_1^*[I] = 1$ and $x_1^*[i] = 0$ (fix a point on the sphere at the pole) so that $x_1^* = e_I$, the unit vector $e_I = (0, \ldots, 0, 1)$. We'll use this to illustrate how a warm start from these interpolations can do better than a random start. Let entries of our initialization be standard Gaussian $x_1^0[i] = g[i] \sim \mathcal{N}(0, 1)$.

$$\begin{aligned}
\mathbb{E}_{g[i]\sim\mathcal{N}} \|g - e_I\|_2^2 &= \mathbb{E}_{g[i]\sim\mathcal{N}} \left[\sum_{i=1}^{I} (g[i] - e_I[i])^2 \right] \\
&= \mathbb{E}_{g[i]\sim\mathcal{N}} \left[\sum_{i=1}^{I-1} (g[i] - 0)^2 + (g[I] - 1)^2 \right] \\
&= \sum_{i=1}^{I-1} \mathbb{E}_{g[i]\sim\mathcal{N}} \left[g[i]^2 \right] + \mathbb{E}_{g[i]\sim\mathcal{N}} \left[(g[I] - 1)^2 \right] \\
&= \sum_{i=1}^{I-1} 1 + \mathbb{E}_{g[i]\sim\mathcal{N}} \left[g[I]^2 - 2g[I] + 1 \right] \\
&= I - 1 + (1) - 2(0) + 1 \\
&= I + 1.
\end{aligned}$$

With some Gaussian concentration, we can square root both sides.

Case 2: $\|x_1^*\|_2 = 0$.

Assume the solution is centred so that $x_1^* = 0 \in \mathbb{R}^I$. Here, we have the well-known result

$$\mathbb{E}\left[\left\|x_1^0 - x_1^*\right\|\right] = \mathbb{E}\left[\|g - 0\|\right] = \sqrt{I}.$$

So in either case, our initial error for a scaled or centred problem goes like $\sim \sqrt{I}$. Since the number of points we have is I is one plus a power of two $I = 2^S + 1$, we *expect* (that is, the expectation is less than the bound with high probability) the following convergence

$$\left\|x_1^K - x_1^*\right\| \leq (q)^K \sqrt{2^S + 2}.$$

So to ensure $\left\|x_1^K - x_1^*\right\| \leq \epsilon$ in expectation, we need

$$\begin{aligned}
(q)^K \sqrt{2^S + 2} &\leq \epsilon \\
\frac{\sqrt{2^S + 2}}{\epsilon} &\leq (q)^{-K} \\
\log\left(\frac{\sqrt{2^S + 2}}{\epsilon}\right) &\leq K \log\left((q)^{-1}\right) \\
\frac{\log\left(\frac{\sqrt{2^S+2}}{\epsilon}\right)}{-\log q} &\leq K \\
\frac{\frac{1}{2}\log\left(2^S + 2\right) + \log(1/\epsilon)}{-\log q} &\leq K.
\end{aligned}$$

SM4.4. Expected Greedy Multi-scale Descent Convergence Proof. Proof of Theorem 6.3.

First we need to have a handle on how close we are to our coarsest discretization at scale $s = s_0$. The coarsest we could go would be when $s_0 = S$ which would result in $2^{S-s_0+1} + 1 = 3$ points total at $t = \ell, \frac{1}{2}(\ell + u)$, and u.

For the centred problem, $x_{s_0}^* = 0 \in \mathbb{R}^{2^{S-s_0+1}+1}$, so we would expect a Gaussian initialization to have an initial error of

$$\mathbb{E}\left[\left\|x_{s_0}^0 - x_{s_0}^*\right\|\right] = \sqrt{2^{S-s_0+1} + 1}.$$

If we used the scaled problem, it's a bit tricker to consider what happens when we only include $2^{S-s_0+1} + 1$ many points, out of the total $2^S + 1$ possible points.

We will actually work out a bound of $\sqrt{2^{S-s_0+1} + 2}$ by considering the "smoothest" case where $(x_1^*)[i] = \frac{1}{\sqrt{2^S+1}}$ (normalized perfectly to a diagonal), and the "roughest" case where $x_1^* = e_{2^S+1} \in \mathbb{R}^{2^S+1}$ was aligned to the pole. After discretization, the smooth case still has all the same entries, but there are just less of them ($2^{S-s_0+1} + 1$ many). In the rough case, we stay aligned to a pole $x_{s_0}^* = e_{2^{S-s_0+1}+1} \in \mathbb{R}^{2^{S-s_0+1}+1}$ since the last entry stays at a one, and the rest are still zero. We use a standard normal initialization of $g \in \mathbb{R}^{2^{S-s_0+1}+1}$ in the smaller space. This is identical to choosing the same initialization on the finer grid, and dropping our the coordinates that we skip.

So this really is a fair comparison to the regular gradient descent. In the smooth case,

$$\begin{aligned}
\mathbb{E}\left[\left\|x_{s_0}^0 - x_{s_0}^*\right\|^2\right] &= \sum_{i=1}^{2^{S-s_0+1}+1} \mathbb{E}\left(g[i] - \frac{1}{\sqrt{2^S+1}}\right)^2 \\
&= \sum_{i=1}^{2^{S-s_0+1}+1} \left(\mathbb{E}\left[g[i]^2\right] - 2\frac{1}{\sqrt{2^S+1}}\mathbb{E}\left[g[i]\right] + \frac{1}{2^S+1}\mathbb{E}\left[1\right]\right) \\
&= \sum_{i=1}^{2^{S-s_0+1}+1} \left(1 - 0 + \frac{1}{2^S+1}\right) \\
&= (2^{S-s_0+1}+1)\left(1 + \frac{1}{2^S+1}\right) \\
&= (2^{S-s_0+1}+1)\left(\frac{2^S+2}{2^S+1}\right).
\end{aligned}$$

In the rough case

$$\begin{aligned}
\mathbb{E}\left[\left\|x_{s_0}^0 - x_{s_0}^*\right\|^2\right] &= \sum_{i=1}^{2^{S-s_0+1}+1-1} \mathbb{E}\left(g[i] - 0\right)^2 + \mathbb{E}\left(g[2^{S-s_0+1}+1] - 1\right)^2 \\
&= 2^{S-s_0+1} + 2.
\end{aligned}$$

For $2^{S-s_0+1} \geq 3$ (which is the case), $2^{S-s_0+1} + 2 > (2^{S-s_0+1}+1)\left(\frac{2^S+2}{2^S+1}\right)$ (in fact, it is bigger by 1 asymptotically as S grows large).

So we are justified in using $\sqrt{2^{S-s_0+1}+2}$ as our *expected* bound on $\left\|x_{s_0}^0 - x_{s_0}^*\right\|$.

This means our descent is

$$\begin{aligned}
&\left\|x_1^{K_1} - x_1^*\right\| \\
&\leq \sqrt{2^{S-1}}(q)^{\sum_{s=1}^S K_s} e_S + C\sum_{s=1}^{S-1} 2^s (q)^{\sum_{t=1}^s K_t} \\
&\leq \sqrt{2^{S-1}}(q)^{\sum_{s=1}^S K_s}\sqrt{2^{S-S+1}+2} + C\sum_{s=1}^{S-1} 2^s (q)^{\sum_{t=1}^s K_t} \\
&= \sqrt{2^{S+1}}(q)^{\sum_{s=1}^S K_s} + \frac{L\left|u-\ell\right|}{2\sqrt{2^{S+1}}}\sum_{s=1}^{S-1} 2^s (q)^{\sum_{t=1}^s K_t} \\
&= (q)^{K_1}\sqrt{2^{S+1}}\left((q)^{\sum_{s=2}^S K_s} + \frac{1}{\sqrt{2^{S+1}}}\frac{L_{f^*}\left|u-\ell\right|}{2\sqrt{2^{S+1}}}\sum_{s=1}^{S-1} 2^s (q)^{\sum_{t=2}^s K_t}\right) \\
&= (q)^{K_1}\sqrt{2^{S+1}}\left((q)^{\sum_{s=2}^S K_s} + \frac{L_{f^*}\left|u-\ell\right|}{2^{S+2}}\sum_{s=1}^{S-1} 2^s (q)^{\sum_{t=2}^s K_t}\right)
\end{aligned}$$

with the understanding that $\sum_{t=2}^1 K_t = 0$.

SM4.5. Cost of Projected Gradient Descent vs Greedy Multi-scale Proof.

Proof of Lemma 6.4.

The total cost of regular projected gradient descent is given by

$$C_{\mathrm{GD}} = AK.$$

For multiscale, we add up the cost at each scale giving us

$$C_{\mathrm{GM}} = \sum_{s=1}^{S} C_s K_s.$$

In the case of greedy multi-scale, one iteration at the finest scale $s = 1$ costs the same for regular projected gradient descent and multi-scale $A = C_1$.

We assume that the cost of projected gradient descent grows at least linearly in the number of points $I = 2^{S-s+1} + 1$ at each scale s

$$C_s = CI^p = C\left(2^{S-s+1} + 1\right)^p$$

for some constant $C > 0$, power $p \geq 1$, and coarsest scale S, where there are $2^S + 1$ points at the finest scale $s = 1$.

We wish to find a lower bound to the ratio

$$\frac{C_s}{C_{s+1}} = \frac{C\left(2^{S-s+1} + 1\right)^p}{C\left(2^{S-(s+1)+1} + 1\right)^p} = \left(\frac{2^{S-s+1} + 1}{2^{S-s} + 1}\right)^p$$

for $1 \leq s \leq S - 1$. Letting $x = S - s$, the function $g(x) = \frac{2^{x+1}+1}{2^x+1}$ is increasing so it is minimized at $x = 1$ on $1 \leq x = S - s \leq S - 1$. This means $g(x) \geq g(1) = 5/3$ and, since $p \geq 1$,

$$\frac{C_s}{C_{s+1}} \geq \left(\frac{5}{3}\right)^p \geq \frac{5}{3}.$$

This gives us the chain,

$$C_1 \geq \frac{5}{3}C_2 \geq \left(\frac{5}{3}\right)^2 C_3 \geq \cdots \geq \left(\frac{5}{3}\right)^{S-1} C_S$$

or

$$C_S \leq \left(\frac{3}{5}\right) C_{S-1} \leq \cdots \leq \left(\frac{3}{5}\right)^{S-2} C_2 \leq \left(\frac{3}{5}\right)^{S-1} C_1.$$

This means

$$C_{\mathrm{GM}} = \sum_{s=1}^{S} C_s K_s \leq C_1 \sum_{s=1}^{S} \left(\frac{3}{5}\right)^{s-1} K_s.$$

Noting $A = C_1$ completes the proof.

SM4.6. Cost of Projected Gradient Descent vs Lazy Multi-scale Proof. Proof of Lemma 6.7.

Similar to the proof of Lemma 6.4 in subsection SM4.5, we suppose the total cost of regular projected gradient descent is given by

$$C_{\mathrm{GD}} = AK,$$

for some $A > 0$ and number of iterations K, and the cost of lazy multi-scale is

$$C_{\mathrm{LM}} = \sum_{s=1}^{S} C_s K_s.$$

We again assume that the cost of projected gradient descent grows at least linearly in the number of points $I = 2^{S-s}$ at each scale s for $1 \leq s \leq S-1$, and $I = 3$ points at the coarsest scale $s = S$ so that

$$C_s = CI^p = C\left(2^{S-s}\right)^p, \text{ for } 1 \leq s \leq S-1, \quad \& \quad C_S = C3^p$$

for some constant $C > 0$, power $p \geq 1$. This also applies to the cost of projected gradient descent applied to each point at the finest scale $I = 2^S + 1$ which is

$$A = CI^p = C\left(2^S + 1\right)^p.$$

We note the ratios

$$\frac{A}{C_s} = \frac{C\left(2^S\right)^p}{C\left(2^{S-s}\right)^p} = \left(2^s\right)^p > 2^s$$

for $1 \leq s \leq S-1$, and

$$\frac{A}{C_S} = \frac{C\left(2^S\right)^p}{C3^p} > \frac{2^S}{3}$$

since $S \geq 2$ and $p \geq 1$.

So the cost of lazy multi-scale is bounded by

$$C_{\mathrm{LM}} = \sum_{s=1}^{S} C_s K_s \leq A\left(\sum_{s=1}^{S-1} 2^{-s} K_s + 2^{-S} 3 K_S\right).$$

If we perform the same number of iterations $K_s = K'$ at each scale, then the cost is bounded by

$$C_{\mathrm{LM}} \leq AK'\left(\sum_{s=1}^{S-1} 2^{-s} + 2^{-S}3\right) = AK'\left(1 - 2^{-S}2 + 2^{-S}3\right) = AK'\left(1 + 2^{-S}\right).$$

To ensure this is less than AK, we need

$$K'\left(1 + 2^{-S}\right) < K$$
$$K' < \frac{1}{1 + 2^{-S}} K.$$

The right side factor is maximized at $S = 2$ on $2 \leq S$, so we can set

$$K' = \left\lceil \frac{1}{1 + 2^{-2}} K \right\rceil - 1 = \left\lceil \frac{4}{5} K \right\rceil - 1$$

and ensure that $C_{\mathrm{LM}} < C_{\mathrm{GD}}$.

SM4.7. Greedy Multi-scale Descent Cost Proof. Proof of Corollary 6.6.

From Corollary 6.5, we know performing one iteration at every scale except performing $K_1 = K - 2$ iterations at the finest scale will be cheaper than K iterations of projected gradient descent at the finest scale. We now need to show this plan results in a tighter upper bound on the expected final error for greedy multi-scale.

By Theorem 6.1, the upper bound on the expected error is

$$(q)^K \sqrt{2^S + 2}.$$

By Theorem 6.3, the upper bound on the expected error for this plan is

$$\begin{aligned}
&(q)^{K_1}\sqrt{2^{S+1}}\left((q)^{\sum_{s=2}^{S} K_s} + \frac{L_{f^*}\,|u-\ell|}{2^{S+2}}\sum_{s=1}^{S-1} 2^s (q)^{\sum_{t=2}^{s} K_t}\right)\\
&= (q)^{K_1}\sqrt{2^{S+1}}\left((q)^{S-1} + \frac{L_{f^*}\,|u-\ell|}{2^{S+2}}\sum_{s=1}^{S-1} 2^s (q)^{s-1}\right)\\
&= (q)^{K_1}\sqrt{2^{S+1}}\left((q)^{S-1} + \frac{L_{f^*}\,|u-\ell|}{2^{S+2}q}\sum_{s=1}^{S-1} (2q)^{s}\right)\\
&= (q)^{K_1}\sqrt{2^{S+1}}\left((q)^{S-1} + \frac{L_{f^*}\,|u-\ell|}{2^{S+2}q}\frac{2q-(2q)^S}{1-2q}\right).
\end{aligned}$$

Already we have assumed $q \neq 1/2$ so we can express the sum in the above closed-form expression.

Our goal will be to show

$$(q)^{K}\sqrt{2^S+2} > (q)^{K_1}\sqrt{2^{S+1}}\left((q)^{S-1} + \frac{L_{f^*}\,|u-\ell|}{2^{S+2}q}\frac{2q-(2q)^S}{1-2q}\right).$$

Assume $0 < q < \frac{1}{2}$, and that the finest scale problem is discretized with at least $I = 2^S + 1$ points where the number of scales S is bigger than

$$S \geq \max\left\{4,\, \log_2\left(\frac{L_{f^*}\,|u-\ell|}{\sqrt{2}(1-2q)((q)^2-\sqrt{2}(q)^3)}\right)\right\}.$$

This ensures $S \geq 4$, and

$$\begin{aligned}
S &\geq \log_2\left(\frac{L_{f^*}\,|u-\ell|}{\sqrt{2}(1-2q)((q)^2-\sqrt{2}(q)^3)}\right)\\
2^S &\geq \frac{L_{f^*}\,|u-\ell|}{\sqrt{2}(1-2q)((q)^2-\sqrt{2}(q)^3)}\\
2^S &\geq \frac{L_{f^*}\,|u-\ell|\sqrt{2}}{2(1-2q)((q)^2-\sqrt{2}(q)^3)}.
\end{aligned}$$

We use two tricks to relax this inequality. First, because $S-1 \geq 3 > 0$ and $0 < q < 1/2$, we have $0 < (2q)^{S-1} < 1$ so

$$1 > 1-(2q)^{S-1} > 0.$$

We also have

$$\begin{aligned}
\frac{1}{q^2\sqrt{2^{-1}+2^{-S}}-q^{S-1}} &\overset{(1)}{<} \frac{1}{q^2\sqrt{2^{-1}}-q^{S-1}}\\
&= \frac{q^{-2}\sqrt{2}}{1-\sqrt{2}q^{S-3}}\\
&\overset{(2)}{\leq} \frac{q^{-2}\sqrt{2}}{1-\sqrt{2}q}\\
&= \frac{\sqrt{2}}{q^2-\sqrt{2}q^3}.
\end{aligned}$$

We have the inequality (1) since $2^{-S} > 0$, and (2) since $S-3 \geq 1$ implies $q^{S-3} \leq q$. Using these inequalities, we have

$$\begin{aligned}
2^S &\geq \frac{L_{f^*}\left|u-\ell\right|\sqrt{2}}{2(1-2q)((q)^2-\sqrt{2}(q)^3)}\\
2^S &> \frac{L_{f^*}\left|u-\ell\right|(1-(2q)^{S-1})}{2(1-2q)(q^2\sqrt{2^{-1}+2^{-S}}-q^{S-1})}\\
(q^2\sqrt{2^{-1}+2^{-S}}-q^{S-1})2^S &> \frac{L_{f^*}\left|u-\ell\right|(1-(2q)^{S-1})}{2(1-2q)}\\
q^2\sqrt{2^{-1}+2^{-S}} &> q^{S-1}+\frac{L_{f^*}\left|u-\ell\right|(2q-(2q)^S)}{2^{S+2}q(1-2q)}\\
q^2\sqrt{\frac{2^S+2}{2^{S+1}}} &> q^{S-1}+\frac{L_{f^*}\left|u-\ell\right|(2q-(2q)^S)}{2^{S+2}q(1-2q)}\\
q^K\sqrt{2^S+2} &> q^{K-2}\sqrt{2^{S+1}}\left(q^{S-1}+\frac{L_{f^*}\left|u-\ell\right|(2q-(2q)^S)}{2^{S+2}q(1-2q)}\right).
\end{aligned}$$

Since $K_1 = K-2$, this completes the proof.

SM4.8. Expected Lazy Multi-scale Descent Convergence Proof.

Proof of Theorem 6.8.

For multi-scale, we start out with only $I = 2^{S-s_0+1}+1$ points. Taking the largest scale we can $s_0 = S$, we have an initial error bound of $\sqrt{2+2} = 2$ giving us the expected error bound

$$\left\|e_1^{K_1}\right\| \leq 2d(K)(1+d(K))^{S-1} + C\frac{d(K)\left(\sqrt{2^S}\left(d(K)+1\right)^S-\sqrt{2}\left(d(K)+1\right)\right)}{\sqrt{2^S}\left(d(K)+1\right)\left(\sqrt{2}d(K)+\sqrt{2}-1\right)}$$

for $C = \frac{L_{f^*}}{2}\left|u-\ell\right|$ (see Theorem 6.1).

If we factor out a $d(K)$, we get

$$\left\|e_1^{K_1}\right\| \leq d(K)\left(2(1+d(K))^{S-1} + C\frac{\left(\sqrt{2^S}\left(d(K)+1\right)^S-\sqrt{2}\left(d(K)+1\right)\right)}{\sqrt{2^S}\left(d(K)+1\right)\left(\sqrt{2}d(K)+\sqrt{2}-1\right)}\right).$$

SM4.9. Lazy Multi-scale Descent Cost Proof.

Proof of Corollary 6.9.

Recall our expected regular descent error have the bound

$$\left\|x_1^K - x_1^*\right\| \leq (q)^K\sqrt{2^S+2} = d(K)\sqrt{2^S+2}. \tag{SM4.1}$$

From Theorem 6.8, we have the expected error bound for lazy multi-scale

$$\left\|e_1^{K_1}\right\| \leq d(K')\left(2(1+d(K'))^{S-1} + C\frac{\left(\sqrt{2^S}\left(d(K')+1\right)^S-\sqrt{2}\left(d(K')+1\right)\right)}{\sqrt{2^S}\left(d(K')+1\right)\left(\sqrt{2}d(K')+\sqrt{2}-1\right)}\right), \tag{SM4.2}$$

where we run each scale for $K_s = K' = \lceil 4K/5\rceil - 1$ many iterations. From Lemma 6.7, we know lazy multi-scale will be cheaper. It remains to show that the bound on the error at the final scale on the right-hand side of (SM4.2) is less than (SM4.1).

Let $C = L_{f^*} |u - \ell| /2$, $h = 1 + q^{K'}$, and $g = q^{-K/5-1}$. To ensure first line bellow is valid, we cannot have $\sqrt{2}/h = 1$. Otherwise, the logarithm in the denominator becomes zero. Moreover, we assume $K > 5(\log_q(\sqrt{2}-1)-1)/4$ so that $\sqrt{2}/h > 1$ and we can write the inequality (1).

$$
\begin{aligned}
S &\geq \frac{\log\left(\frac{g}{h}\left(2 + C\frac{1}{h\sqrt{2}-1}\right)\right)}{\log\left(\frac{\sqrt{2}}{h}\right)} \\
\log\left(\frac{\sqrt{2}}{h}\right) S &\overset{(1)}{\geq} \log\left(\frac{g}{h}\left(2 + C\frac{1}{h\sqrt{2}-1}\right)\right) \\
(\sqrt{2})^S h^{-S} &\geq \frac{g}{h}\left(2 + C\frac{1}{h\sqrt{2}-1}\right) \\
\sqrt{2^S} &\geq g h^{S-1}\left(2 + C\frac{1}{h\sqrt{2}-1}\right) \\
\sqrt{2^S + 2} &\overset{(2)}{>} g h^{S-1}\left(2 + C\frac{1-(\sqrt{2}h)^{-(S-1)}}{h\sqrt{2}-1}\right) \\
\sqrt{2^S + 2} &> g\left(2h^{S-1} + C\frac{h^{S-1} - \sqrt{2^{-(S-1)}}}{h\sqrt{2}-1}\right) \\
\sqrt{2^S + 2} &> q^{-K/5-1}\left(2h^{S-1} + C\frac{h^S\sqrt{2^S} - h\sqrt{2}}{\sqrt{2^S}h(h\sqrt{2}-1)}\right) \\
q^K\sqrt{2^S + 2} &> q^{4K/5-1}\left(2h^{S-1} + C\frac{h^S\sqrt{2^S} - h\sqrt{2}}{\sqrt{2^S}h(h\sqrt{2}-1)}\right) \\
q^K\sqrt{2^S + 2} &\overset{(3)}{>} q^{\lceil 4K/5\rceil-1}\left(2h^{S-1} + C\frac{h^S\sqrt{2^S} - h\sqrt{2}}{\sqrt{2^S}h(h\sqrt{2}-1)}\right) \\
q^K\sqrt{2^S + 2} &> q^{K'}2(1+q^{K'})^{S-1} \\
&\qquad + q^{K'}\frac{L_{f^*}}{2}|u-\ell|\frac{(1+q^{K'})^S\sqrt{2^S} - (1+q^{K'})\sqrt{2}}{\sqrt{2^S}(1+q^{K'})(\sqrt{2}(1+q^{K'})-1)}
\end{aligned}
$$

For inequality (2), we use the fact that $\sqrt{2^S+2} > \sqrt{2^S}$ and $0 < 1-(\sqrt{2}h)^{-(S-1)} < 1$ to relax the previous line. For inequality (3), we use the fact that $0 < q < 1$ so that possibly multiplying by at most a factor of q only makes the right-hand side smaller.

SM5. Relation between Discretizations and Continuous Problems Proofs.

SM5.1. Piecewise Function Distance Proof.

Proof of Lemma 5.1.
In the following, we use a substitution

$$s = \left(\frac{f(t_{i+1}) - f(t_i)}{\Delta t} - \frac{g(t_{i+1}) - g(t_i)}{\Delta t}\right)(t - t_i) + (f(t_i) - g(t_i)) = m_i(t - t_1) + b_i$$

so that $ds = m_i dt$. We can evaluate

$$\begin{aligned}
\left\|\hat{f}_x - \hat{g}_y\right\|_2^2 &= \int_{t_1}^{t_I} (\hat{f}_x(t) - \hat{g}_x(t))^2 \, dt \\
&= \sum_{i=1}^{I-1} \int_{t_i}^{t_{i+1}} \left(\left(\frac{f(t_{i+1}) - f(t_i)}{\Delta t} - \frac{g(t_{i+1}) - g(t_i)}{\Delta t} \right) (t - t_i) \right. \\
&\qquad \left. + (f(t_i) - g(t_i)) \right)^2 \, dt \\
&= \sum_{i=1}^{I-1} \int_{t_i}^{t_{i+1}} (m_i(t - t_i) + b_i)^2 \, dt \\
&= \sum_{i=1}^{I-1} \frac{1}{m_i} \int_{b_i}^{m_i \Delta t + b_i} s^2 \, ds \\
&= \sum_{i=1}^{I-1} \frac{1}{m_i} \frac{1}{3} (m_i \Delta t + b_i)^3 - b_i^3) \\
&= \sum_{i=1}^{I-1} \frac{1}{m_i} \frac{1}{3} (m_i^3 \Delta t^3 + 2 m_i^2 \Delta t^2 b_i + 2 m_i \Delta t b_i^2 + b_i^3 - b_i^3) \\
&= \sum_{i=1}^{I-1} \frac{\Delta t}{3} (m_i^2 \Delta t^2 + 2 m_i \Delta t b_i + 2 b_i^2) \\
&= \sum_{i=1}^{I-1} \frac{\Delta t}{3} ((m_i \Delta t + b_i)^2 + b_i^2) \\
&= \frac{\Delta t}{3} \sum_{i=1}^{I-1} \left((f(t_{i+1}) - g(t_{i+1}))^2 + (f(t_i) - g(t_i))^2 \right) \\
&= \frac{\Delta t}{3} \left(2 \sum_{i=1}^{I} (f(t_i) - g(t_i))^2 - (f(t_1) - g(t_1))^2 - (f(t_I) - g(t_I))^2 \right) \\
&= \frac{\Delta t}{3} \left(2 \left\| x - y \right\|_2^2 - (x[1] - y[1])^2 - (x[I] - y[I])^2 \right) \\
&\leq \frac{2\Delta t}{3} \left\| x - y \right\|_2^2 .
\end{aligned}$$

Here we have discretized vectors $x, y \in \mathbb{R}^I$ with entries $x_i = f(t_i)$ and $y_i = g(t_i)$. Note the I and not the $I - 1$ like the piecewise constant interpolation.

We also have the additional result that, when the end points are equal $x[1] = y[1]$ and $x[I] = y[I]$, we have equality

$$\left\| f - g \right\|_2^2 = \frac{2\Delta t}{3} \left\| x - y \right\|_2^2 .$$

SM5.2. Piecewise Linear Function Approximation Proof. Proof of Lemma 5.2. Using Lemma 4.1 with $a \mapsto t_{i+1}$, $b \mapsto t_i$, and $t \mapsto (t - t_i)/\Delta t$ where $\Delta t = t_{i+1} - t_i$

and $t_i \leq t \leq t_{i+1}$ gives us

$$\begin{aligned}\frac{t-t_i}{\Delta t}t_{i+1} + \left(1 - \frac{t-t_i}{\Delta t}\right)t_i &= \frac{t-t_i}{\Delta t}t_{i+1} + \frac{t_{i+1}-t}{\Delta t}t_i \\ &= \frac{t \cdot t_{i+1} - t_i t_{i+1} + t_{i+1}t_i - t \cdot t_i}{\Delta t} \\ &= t\frac{t_{i+1}-t_i}{\Delta t} \\ &= t\end{aligned}$$

so that

$$\begin{aligned}\left|\frac{t-t_i}{\Delta t}f(t_{i+1}) + \left(1 - \frac{t-t_i}{\Delta t}\right)f(t_i) - f\left(\frac{t-t_i}{\Delta t}t_{i+1} + \left(1 - \frac{t-t_i}{\Delta t}\right)t_i\right)\right| \\ \leq 2L_f\frac{t-t_i}{\Delta t}\left(1 - \frac{t-t_i}{\Delta t}\right)|t_{i+1}-t_i| \\ \left|\frac{f(t_{i+1})-f(t_i)}{\Delta t}(t-t_i) + f(t_i) - f(t)\right| \leq \frac{L_f}{\Delta t}(t-t_i)(t_{i+1}-t) \\ \left|\hat{f}(t) - f(t)\right| \leq \frac{L_f}{\Delta t}(t-t_i)(t_{i+1}-t).\end{aligned}$$

We can then bound the error, using an integration substitution of $s = t - t_i$,

$$\begin{aligned}\left\|\hat{f} - f\right\|_2^2 &= \int_{t_1}^{t_I}(\hat{f}(t) - f(t))^2\,dt \\ &= \sum_{i=1}^{I-1}\int_{t_i}^{t_{i+1}}(\hat{f}(t) - f(t))^2\,dt \\ &\leq \sum_{i=1}^{I-1}\int_{t_i}^{t_{i+1}}\left(\frac{2L}{\Delta t}(t-t_i)(t_{i+1}-t)\right)^2\,dt \\ &= \frac{4L^2}{\Delta t^2}\sum_{i=1}^{I-1}\int_{t_i}^{t_{i+1}}(t-t_i)^2(t_{i+1}-t)^2\,dt \\ &= \frac{4L^2}{\Delta t^2}\sum_{i=1}^{I-1}\int_0^{\Delta t}s^2(\Delta t - s)^2\,ds \\ &= \frac{4L^2}{\Delta t^2}(I-1)\int_0^{\Delta t}\Delta t^2 s^2 - 2\Delta t s^3 + s^4\,ds \\ &= \frac{4L^2}{\Delta t^2}(I-1)\left(\frac{1}{3}\Delta t^2\Delta t^3 - \frac{1}{4}2\Delta t\Delta t^4 + \frac{1}{5}\Delta t^5\right) \\ &= 4L^2\Delta t^2\frac{t_I - t_1}{I-1}(I-1)\left(\frac{1}{3} - \frac{1}{2} + \frac{1}{5}\right) \\ &= \frac{2}{15}L^2\Delta t^2(t_I - t_1).\end{aligned}$$

SM5.3. Continuous and Discrete Problem Connection Proof. Proof of Theorem 5.3.

Let $\epsilon > 0$, $I > \sqrt{\frac{8}{15}}(u-\ell)^{3/2}L\epsilon^{-1} + 1$, and $\|x - x^*\|_2^2 < \sqrt{\frac{3}{40}}(u-\ell)^{1/2}L\epsilon$. This means

$$\frac{1}{I-1} < \frac{\epsilon\sqrt{15}}{\sqrt{8}L(u-\ell)^{3/2}}.$$

We wish to bound

$$\left\|\hat{f}_x - f\right\|_2 < \epsilon.$$

Our approach will be to use triangle inequality with the piecewise linear approximation $\hat{f}_{x^*} = \hat{f}$ of f where $x^*[i] = f(t[i])$ is the true discretization of f. This gives us

$$\begin{aligned}\left\|\hat{f}_x - f\right\|_2 &\leq \left\|\hat{f}_x - \hat{f}_{x^*}\right\|_2 + \left\|\hat{f}_{x^*} - f\right\|_2 \\ &= \left\|\hat{f}_x - \hat{f}_{x^*}\right\|_2 + \left\|\hat{f} - f\right\|_2.\end{aligned}$$

By Lemma 5.1, we bound the first term by

$$\begin{aligned}\left\|\hat{f}_x - \hat{f}_{x^*}\right\|_2^2 &\leq \frac{2\Delta t}{3}\left\|x - x^*\right\|_2^2 \\ &< \frac{2\Delta t}{3}\left(L\epsilon\sqrt{\frac{3(u-\ell)}{40}}\right) \\ &= \frac{2(u-\ell)}{3(I-1)}\left(L\epsilon\sqrt{\frac{3(u-\ell)}{40}}\right) \\ &< \frac{2(u-\ell)}{3}\frac{\epsilon\sqrt{15}}{\sqrt{8}L(u-\ell)^{3/2}}\left(L\epsilon\sqrt{\frac{3(u-\ell)}{40}}\right) \\ &= \frac{\epsilon^2}{4}.\end{aligned}$$

By Lemma 5.2, we bound the second by

$$\begin{aligned}\left\|\hat{f} - f\right\|_2^2 &\leq \frac{2}{15}(u-\ell)L^2\Delta t^2 \\ &= \frac{2}{15}(u-\ell)L^2\left(\frac{u-\ell}{I-1}\right)^2 \\ &< \frac{2}{15}(u-\ell)L^2\left(u-\ell\right)^2\frac{\epsilon^2 15}{8L^2(u-\ell)^3} \\ &= \frac{\epsilon^2}{4}.\end{aligned}$$

Taking square roots and combining these bound gives us our desired result,

$$\begin{aligned}\left\|\hat{f}_x - f\right\|_2 &\leq \left\|\hat{f}_x - \hat{f}_{x^*}\right\|_2 + \left\|\hat{f} - f\right\|_2 \\ &< \frac{\epsilon}{2} + \frac{\epsilon}{2} \\ &= \epsilon.\end{aligned}$$

Remark 1. *The statement we proved, that for all $\epsilon > 0$,*

$$I > \sqrt{\frac{8}{15}}(u-\ell)^{3/2}L\epsilon^{-1} + 1 \text{ and } \left\|x - x^*\right\|_2^2 < \sqrt{\frac{3}{40}}(u-\ell)^{1/2}L\epsilon \implies \left\|\hat{f}_x - f\right\|_2 < \epsilon$$

is artificially complicated. We prove this statement to have a clean final line. But the whole statement can be rewritten by substituting ϵ for $(u-\ell)\epsilon$ to show that, again for all $\epsilon > 0$,

$$I > \sqrt{\frac{8}{15}}(u-\ell)L\epsilon^{-1} + 1 \text{ and } \|x - x^*\|_2^2 < \sqrt{\frac{3}{40}}(u-\ell)L\epsilon \implies \left\| \frac{\hat{f}_x - f}{u-\ell} \right\|_2 < \epsilon.$$

Rewritten, it is clearer that what matters is the effective Lipschitz constant $(u-\ell)L$, and not L and $(u-\ell)$ separately, and the right side is now expressed in terms of the average error.

SM6. Additional Density Unmixing with Tucker-1 Tensor Factorization Details.

SM6.1. Tucker-1 Tensor Decomposition. A tensor decomposition is a factorization of a tensor into multiple (usually smaller) tensors, that can be recombined into the original tensor. Section 7 uses the Tucker-1 decomposition defined in Definition SM6.1.

Definition SM6.1 (Tucker-1 Decomposition). *A rank-R Tucker-1 decomposition of a tensor $Y \in \mathbb{R}^{I_1\times\cdots\times I_N}$ produces a matrix $A \in \mathbb{R}^{I_1\times R}$, and core tensor $B \in \mathbb{R}^{R\times I_2\times\cdots\times I_N}$ such that*

$$Y[i_1,\dots,i_N] = \sum_{r=1}^{R} A[i_1,r]B[r,i_2,\dots,i_N] \tag{SM6.1}$$

entry-wise or more compactly,

$$Y = B \times_1 A.$$

Tensor decompositions are not necessarily unique. It should be clear that scaling one factor by $c \neq 0$ and dividing another by c yields the same original tensor. Furthermore, slices can be permuted without affecting the the original tensor. Up to these manipulations, for a fixed rank, there exist criteria that ensures their decompositions are unique [SM5, SM6, SM1].

SM6.2. Formulating Density Unmixing as a Tucker-1 Decomposition Problem. In both the synthetic and geological data, the task is to recover mixtures of probability densities. We accomplish this task by formulating a discretized version of the problem as a tensor decomposition problem.

In the case of the synthetic example in subsection 7.2, we have access to the true probability density mixtures. For the real-world example in subsection 7.3, the continuous probability density mixtures are estimated from sampling using kernel density estimation [SM2, SM4].

We use the 2-norm/Frobenius loss in the formulation of the continuous decomposition problem (7.1), but alternative losses can be used such as the KL divergence and their corresponding discrete versions based on how the error between the data and model are distributed [SM3]. The constraints (7.1a) and (7.1b) ensure each mixture in the model is indeed a probability density function.

We interpret the tensors $Y \in \mathbb{R}_+^{I\times K_1\times\cdots\times K_N}$ and $B \in \mathbb{R}_+^{R\times K_1\times\cdots\times K_N}$ as samples of the underlying continuous probability density functions

$$Y[i,k_1,\dots,k_N] = y_i(x[k_1,\dots,k_N])\Delta x, \text{ and } B[r,k_1,\dots,k_N] = b_r(x[k_1,\dots,k_N])\Delta x,$$

with a uniformly spaced grid[2] $\{x[k_1,\dots,k_N]\}\subset\mathcal{D}$. We scale by the volume element of the grid,

$$\Delta x=\prod_{n=1}^{N}\Delta_n x=\prod_{n=1}^{N}(x[1,\dots,1,\underbrace{2}_{n},1,\dots 1]-x[1,\dots,1,\underbrace{1}_{n},1,\dots 1]),$$

to ensure the tensors are normalized, $Y\in\Delta^{I}_{K_1\dots K_N}$ and $B\in\Delta^{R}_{K_1\dots K_N}$. This leads to the convenient notation of switching from integrals to summations when we discretize;

$$1=\int_{\mathcal{D}}y_i(x)dx\quad\text{discretizes to}\quad 1\approx\sum_{k_1,\dots,k_N=1}^{K_1\dots K_N}Y[i,k_1,\dots,k_N]\Delta x.$$

In higher dimensions when N is large, it can be expensive to compute the full N-dimensional kernel density estimation[3] for each b_r and memory intensive to store the full N dimensional tensor Y. We can instead approximate the full distribution $y_i:\mathcal{D}\subseteq\mathbb{R}^N\to\mathbb{R}_+$ as a product distribution of $J=N$ one-dimensional distributions $y_i^j:\mathcal{D}^j\subseteq\mathbb{R}\to\mathbb{R}_+$:

$$y_i(x)=\prod_{j=1}^{J}y_i^j(x[j]),\quad\text{where}\quad\mathcal{D}=\mathcal{D}^1\times\cdots\times\mathcal{D}^J,$$

and similarly with b_r and b_r^j. This lets us discretize the probability density functions as 3rd order tensors regardless of the number of dimensions N,

$$Y[i,j,k]=y_i^j(x_k^j)\Delta x^j,\quad\text{and}\quad B[r,j,k]=b_r^j(x_k^j)\Delta x^j,$$

with a 1D grid $\{x_k^j\}\subset\mathcal{D}^j$ for each $j\in[J]$. The constraint on B also gets similarly modified to

$$B\in\Delta_K^{RJ}=\left\{B\in\mathbb{R}_+^{R\times J\times K}\;\middle|\;\forall(r,j)\in[R]\times[J],\sum_{k=1}^{K}B[r,j,k]=1\right\}.$$

We use the full 3-dimensional representation of the probability density mixtures for the synthetic data in subsection 7.2, and use the compressed representation of the 7-dimensional mixtures for the real word data in subsection 7.3.

SM6.3. Solution Algorithm. Let

$$f(A,B):=\frac{1}{2}\left\|B\times_1 A-Y\right\|_F^2$$

be the objective function we wish to minimize in (7.2). Following Xu and Yin [SM10], the general approach we take to minimize f is to apply block coordinate descent using each factor as a different block. Let A^t be the tth iteration of A and let

$$f_A^t(A):=\frac{1}{2}\left\|B^t\times A-Y\right\|_F^2\tag{SM6.2a}$$

$$f_B^t(B):=\frac{1}{2}\left\|B\times A^t-Y\right\|_F^2\tag{SM6.2b}$$

[2]If the domain $\mathcal{D}$ is unbounded, we may restrict the domain to the support of y_i, or where $y_i(x)\geq\epsilon$ for some $\epsilon>0$.

[3]Despite the existence of fast algorithms for kernel density estimation [SM8], it can still be cheaper to compute N one-dimensional kernel density estimations than one N-dimensional kernel density estimation.

be the (partially updated) objective function at iteration t for the factor A, and similarly with B.

Given initial factors A^0 and B^0, we alternate through the factors and perform the updates

$$A^{t+1} \leftarrow P_{\Delta_R^I}\left(A^t - \frac{1}{\mathcal{S}_{f_A^t}} \nabla f_A^t(A^t)\right) \tag{SM6.3a}$$

$$B^{t+1} \leftarrow P_{\Delta^R_{K_1\dots K_N}}\left(B^t - \frac{1}{\mathcal{S}_{f_B^t}} \nabla f_B^t(B^t)\right) \tag{SM6.3b}$$

for $t = 1, 2, \dots$ until some convergence criterion is satisfied. In our case, we iterate until the relative error,

$$\frac{\|B^t \times A^t - Y\|_F}{\|Y\|_F},$$

or mean relative error,

$$\frac{1}{I \cdot K_1 \cdot \dots \cdot K_N} \sum_{k_1,\dots,k_N} \frac{|(B^t \times A^t)[k_1,\dots,k_N] - Y[k_1,\dots,k_N]|}{Y[k_1,\dots,k_N]},$$

is less than some specified tolerance δ. In practice to avoid dividing by small entries of Y, we may take the mean relative error only on the subarray of Y where entries are larger than some threshold. Given Y is in the constraint set Δ_K

We choose a stepsize of $\alpha = 1/\mathcal{S}_{f_A^t}$, since it is a sufficient condition to guarantee $f_A^t(A^{t+1}) \leq f_A^t(A_A^t)$, but other stepsizes can be used in theory [SM7, Sec. 1.2.3]. This choice also agrees with the best stepsize given in Lemma 2.6 since the least-square loss in (SM6.2a) is $\mathcal{S}_{f_A^t}$-smooth and $\mu_{f_A^t}$ strongly convex where $\mu_{f_A^t} = \mathcal{S}_{f_A^t}$. A similar idea holds with B in (SM6.2b).

SM7. Synthetic Data Setup.

This section details the data used in subsection 7.2. The main setup is provided here, and the full file can be viewed at `julia_experiments/distribution_unmixing_synthetic.jl` in this paper's GitHub repository [SM9].

```julia
using Distributions

source1a, source1b, source1c = Normal(4, 1), Uniform(-7, 2), Uniform(-1, 1)
source2a, source2b, source2c = Normal(0, 3), Uniform(-2, 2), Exponential(2)
source3a, source3b, source3c = Exponential(1), Normal(0, 1), Normal(0, 3)

source1 = product_distribution([source1a, source1b, source1c])
source2 = product_distribution([source2a, source2b, source2c])
source3 = product_distribution([source3a, source3b, source3c])

sources = (source1, source2, source3)
```

We generate the following 5×3 mixing matrix

```julia
p1 = [0.0, 0.4, 0.6]
p2 = [0.3, 0.3, 0.4]
p3 = [0.8, 0.2, 0.0]
p4 = [0.2, 0.7, 0.1]
p5 = [0.6, 0.1, 0.3]

A_true = hcat(p1,p2,p3,p4,p5)'
```

and use it to construct 5 mixture distributions.

```
distribution1 = MixtureModel([sources...], p1)
distribution2 = MixtureModel([sources...], p2)
distribution3 = MixtureModel([sources...], p3)
distribution4 = MixtureModel([sources...], p4)
distribution5 = MixtureModel([sources...], p5)
distributions =
    [distribution1, distribution2, distribution3, distribution4, distribution5]
```

These are discretized into $65 \times 65 \times 65$ sample tensors, and stacked into a $5 \times 65 \times 65 \times 65$ tensor Y. We normalize the 1-slices so that they sum to one.

```
sinks = [pdf.((d,), xyz) for d in distributions]
Y = cat(sinks...; dims=4)
# reorder so the first dimension indexes mixtures
Y = permutedims(Y, (4,1,2,3))
Y_slices = eachslice(Y, dims=1)
correction = sum.(Y_slices) # normalize slices to 1
Y_slices ./= correction
```

The options used for both the single and multi scaled factorization are the following.

```
options = (
  rank=3,
  momentum=false,
  model=Tucker1,
  tolerance=(0.05, 1e-6),
  converged=(MeanRelError, ObjectiveValue),
  do_subblock_updates=true,
  constrain_init=true,
  constraints=[nonnegative!, simplex_rows!],
  stats=[Iteration, ObjectiveValue, GradientNNCone, RelativeError, MeanRelError]
      ,
  mean_rel_error_tol = 1e-4,
  maxiter=50
)
```

At each scale, the algorithm will continue to iterate until one of the following three occur. The mean relative error is below 5%, the objective value is below 10^{-6}, or 50 iterations have occurred. At this point, the iteration at the next smaller scale will start.

To create Figure 8, we modify the options so that we only use the objective value at the convergence criteria. This allows us to make a fairer comparison between different number of coarse iterations, and the analysis in the paper. The code for this figure can be found in `julia_experiments/distribution_unmixing_synthetic_figures.jl`.

SM7.1. Geological Data Factorization Details. We discretize the densities with $K = 2^{10} + 1 = 1025$ points to obtain an input tensor $Y \in \mathbb{R}_+^{20\times 7\times 1025}$ and normalize the depth fibres so that $\sum_{k\in[K]} Y[i,j,k] = 1$ for all $i \in [20]$ and $j \in [7]$.

We run the multiscale factorization algorithm with the following call

```
multiscale_factorize(Y; continuous_dims=3, options...)
```

from `BlockTensorFactorization`. The third dimension is specified as continuous since each depth fibre $Y[i,j,:]$ is a discretized continuous probability density function.

This is compared to the regular factorization algorithm

```
factorize(Y; options...)
```

using the Julia package `BenchmarkingTools`. This runs the algorithm as many times as it can within a default time window. Note that a new random initialization is generated for each run. After running the following,

```
using BenchmarkingTools

# Run each function once so they compile
factorize(Y; options...);
multiscale_factorize(Y; continuous_dims=3,options...);

benchmark1 = @benchmark factorize(Y; options...)
display(benchmark1)

benchmark2 = @benchmark multiscale_factorize(Y; continuous_dims=3,options...)
display(benchmark2)
```

we observe the two benchmarks shown in subsection 7.3.

We run the multiscale and single scale factorization algorithms with the following options.

```
options = (
    rank=3,
    momentum=false,
    do_subblock_updates=false,
    model=Tucker1,
    tolerance=(0.12),
    converged=(RelativeError),
    constrain_init=true,
    constraints=[l1scale_average12slices! ∘ nonnegative!, nnonnegative!],
    stats=[Iteration, ObjectiveValue, GradientNNCone, RelativeError],
    maxiter=200
)
```

We use the same convergence criteria at each scale and iterate until the relative error between the input Y and our model X is at most 12%, or until 200 iterations have passed. We use 12% because this is roughly the error Graham et. al. observe in the final factorization [SM4], suggesting this is the roughly the smallest amount of error we can expect in this factorization.

REFERENCES

[SM1] A. Bhaskara, M. Charikar, and A. Vijayaraghavan, *Uniqueness of Tensor Decompositions with Applications to Polynomial Identifiability*, in Proceedings of The 27th Conference on Learning Theory, PMLR, May 2014, pp. 742–778. ISSN: 1938-7228.

[SM2] Y.-C. Chen, *A tutorial on kernel density estimation and recent advances*, Biostatistics & Epidemiology, 1 (2017), pp. 161–187, https://doi.org/10.1080/24709360.2017.1396742. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/24709360.2017.1396742.

[SM3] N. Gillis, *NMF models*, in Nonnegative Matrix Factorization, Data Science, Society for Industrial and Applied Mathematics, Jan. 2020, pp. 157–193, https://doi.org/10.1137/1.9781611976410.ch5.

[SM4] N. Graham, N. Richardson, M. P. Friedlander, and J. Saylor, *Tracing Sedimentary Origins in Multivariate Geochronology via Constrained Tensor Factorization*, Mathematical Geosciences, (2025), https://doi.org/10.1007/s11004-024-10175-0.

[SM5] T. G. Kolda and B. W. Bader, *Tensor Decompositions and Applications*, 51, pp. 455–500, https://doi.org/10.1137/07070111X.

[SM6] J. B. Kruskal, *Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics*, Linear Algebra and its Applications, 18 (1977), pp. 95–138, https://doi.org/10.1016/0024-3795(77)90069-6.

[SM7] Y. Nesterov, *Nonlinear Optimization*, in Lectures on Convex Optimization, Y. Nesterov, ed., Springer Optimization and Its Applications, Springer International Publishing, pp. 3–58, https://doi.org/10.1007/978-3-319-91578-4_1.

[SM8] T. A. O'Brien, K. Kashinath, N. R. Cavanaugh, W. D. Collins, and J. P. O'Brien, *A fast and objective multidimensional kernel density estimation method: fastKDE*, Computational Statistics & Data Analysis, 101 (2016), pp. 148–160, https://doi.org/10.1016/j.csda.2016.02.014.

[SM9] N. Richardson, N. Marusenko, and M. P. Friedlander, *richardson-multiscale-2025*. https://github.com/MPF-Optimization-Laboratory/richardson-multiscale-2025, 2025.

[SM10] Y. Xu and W. Yin, *A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion*, 6, pp. 1758–1789, https://doi.org/10.1137/120887795.