Multiscale Methods

Optimization of Discretized Continuous Functions

5 Jun 2026 | SIAM OP26 | Edinburgh, UK

Nicholas J. E. Richardson[1], Noah Marusenko[2], & Michael P. Friedlander[1,2]

Departments of [1]Mathematics and [2]Computer Science

INSTITUTE OF APPLIED MATHEMATICS
University of British Columbia

siam 2026 | Conference on Optimization

THE UNIVERSITY OF BRITISH COLUMBIA

Paper [Richardson *et al.* 2025]

https://arxiv.org/abs/2512.13993

Slides

https://njericha.github.io/research

The Problem

What we want

Solve

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

with

- objective $\mathcal{L} : \mathcal{C} \to \mathbb{R}$
- continuous function $f : \mathcal{D} \to \mathbb{R}$
- domain $\mathcal{D} \subseteq \mathbb{R}$
- constraint $\mathcal{C} \subseteq \mathcal{F} := \{f : \mathcal{D} \to \mathbb{R} \mid f \text{ is continuous}\}$.

How can we solve P on a computer?

Some approaches

Solve over a (truncated) basis for f

- e.g. wavelets [Benedetto *et al.* 1993], polynomials [Trefethen 2019]

Parametrize f with an expressive family

- e.g. Gaussians [Vershynin 2018], neural networks [Gurney 2018]

Use variational calculus techniques [Gelfand *et al.* 2000]

Use duality [Chuna *et al.* 2025]

Some approaches

Solve over a (truncated) basis for f

- e.g. wavelets [Benedetto *et al.* 1993], polynomials [Trefethen 2019]

Parametrize f with an expressive family

- e.g. Gaussians [Vershynin 2018], neural networks [Gurney 2018]

Use variational calculus techniques [Gelfand *et al.* 2000]

Use duality [Chuna *et al.* 2025]

Our approach: Solve P on progressively finer discretizations of f

- see multigrid [Trottenberg *et al.* 2001], MGProx [Ang *et al.* 2024]

What's new: general framework, and iterate & computational analysis

A Numerical Example

The Multiscale Method: Example

Recover continuous probability density $p : [-1, 1] \to \mathbb{R}_+$ from polynomial measurements $y \in \mathbb{R}^M$ with regularized least-squares:

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

with

- measurements $\mathcal{A}(p)[m] := \langle a_m, p \rangle = \int_{-1}^{1} a_m(t) p(t)\, dt$
- Legendre polynomials a_m of degree m
- regularizer $\|p'\|_2^2 = \int_{-1}^{1} p'(t)^2\, dt$ to promote smooth p.

The Multiscale Method: Discretized Example

Uniformly discretize $[-1,1]$ with I points $t[i] = 2\frac{i-1}{I-1} - 1$ to get

$$\min_x \left\{ \frac{1}{2}\|Ax - y\|_2^2 + \frac{1}{2}\lambda x^\top G x \;\middle|\; \|x\|_1 = 1 \text{ and } x \geq 0 \right\}$$

where $x[i] = p(t[i])\Delta t$ and

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

Single-Scale Solution Method

Run projected gradient descent on x^k starting at a random x^0.

Single-Scale Solution Method

Run projected gradient descent on x^k starting at a random x^0.

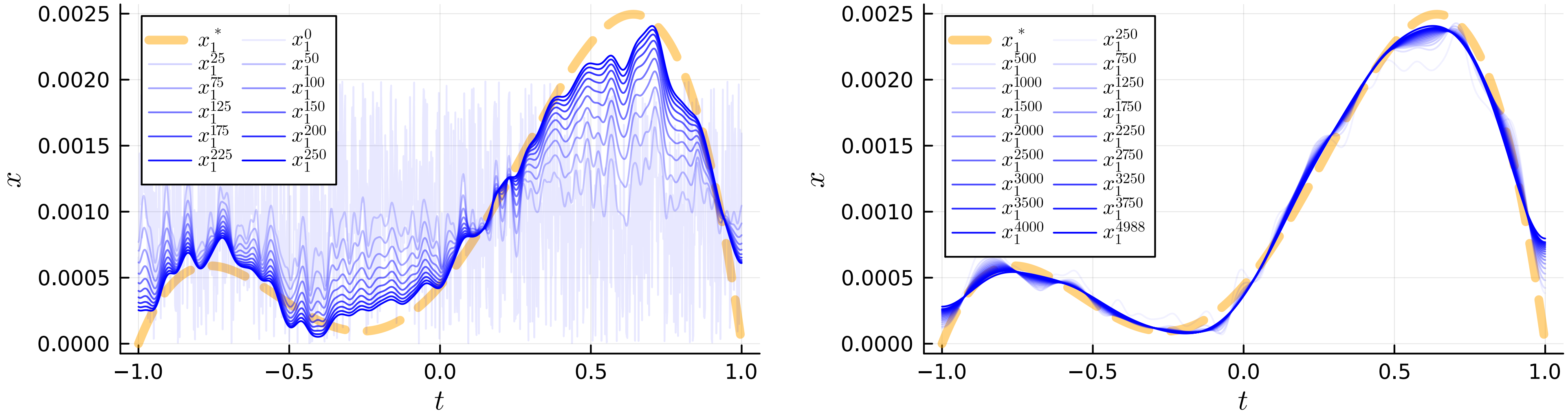

Fig 1: Progression of iterates x^k at iteration k. The clean ground truth is plotted behind in the thick dashed lined. Iterations $k = 1, 25, \ldots, 250$ (left) and $250, 500, \ldots, 5\,000$ (right).

A Faster Way: Discretize at Multiple Scale

Discretize at multiple scales $s = S, S-1, ..., 1$ to get

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

where

- $x_s = x[1 : 2^{s-1} : I]$ (sub-vector of x only taking every 2^{s-1} points)
- $A_s = 2^{s-1} A[:, 1 : 2^{s-1} : I]$
- $G_s = 2^{1-s} G[1 : 2^{s-1} : I, 1 : 2^{s-1} : I]$

so that $A_s x_s \approx Ax$ and $x_s^\top G_s x_s \approx x^\top G x$ at all scales.

Discretization at multiple scales

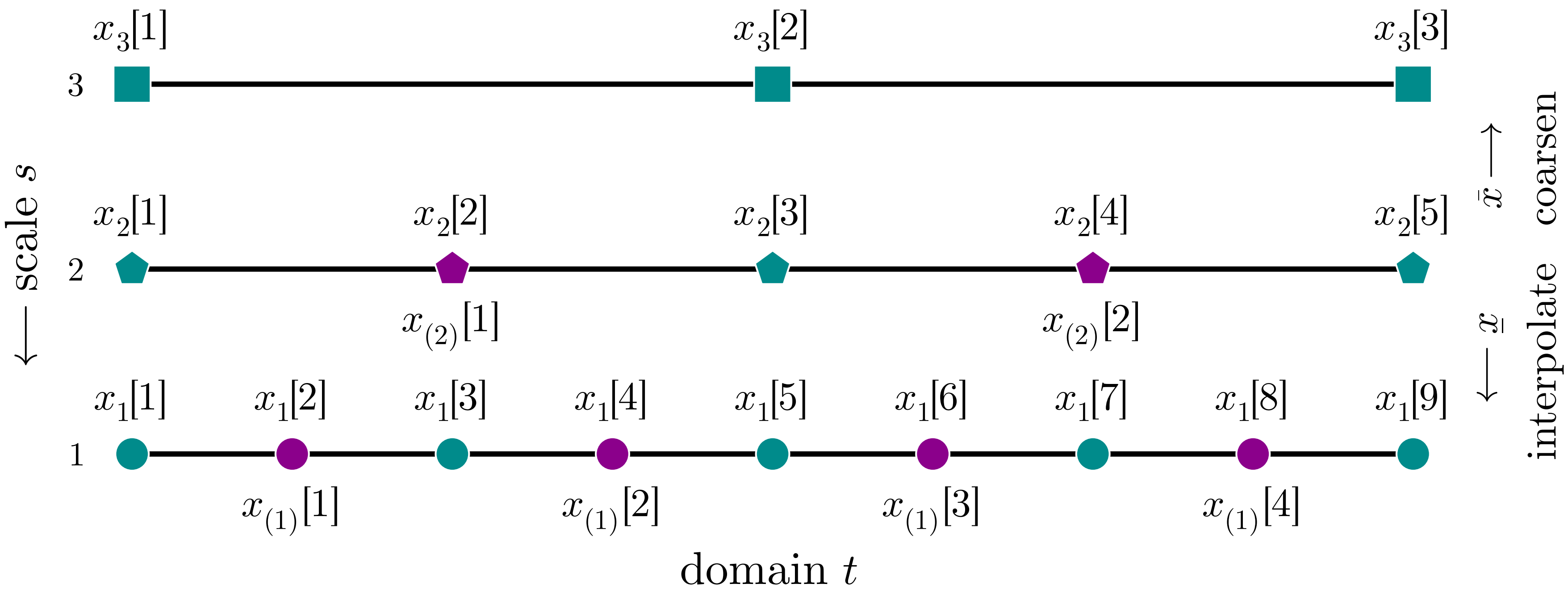

Fig 2: Discretization of an interval with $S = 3$ scales. Scale s has $2^{S-s+1} + 1$ points; new points (purple) and existing points (teal).

Multiscale Solution Method

Run 1 step of projected gradient descent on x_s^k at scales $s = 10, 9, ..., 2$, then run projected gradient descent on x_1^k for many iterations.

Multiscale Solution Method

Run 1 step of projected gradient descent on x_s^k at scales $s = 10, 9, \ldots, 2$, then run projected gradient descent on x_1^k for many iterations.

The trick: Linearly interpolate x_s^1 to warm start the initialization x_{s-1}^0.

Multiscale Solution Method

Run 1 step of projected gradient descent on x_s^k at scales $s = 10, 9, \ldots, 2$, then run projected gradient descent on x_1^k for many iterations.

The trick: Linearly interpolate x_s^1 to warm start the initialization x_{s-1}^0.

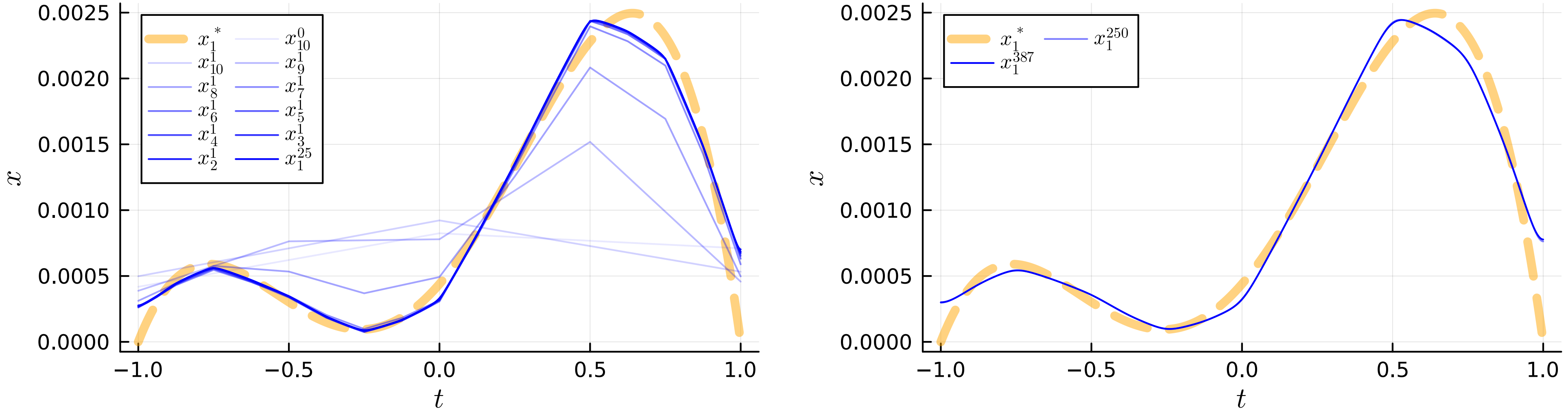

Fig 3: x_s^k for $k = 1$ and $s = 10, \ldots, 1$ (left), $k = 400$ and $s = 1$ (right).

Faster you say?

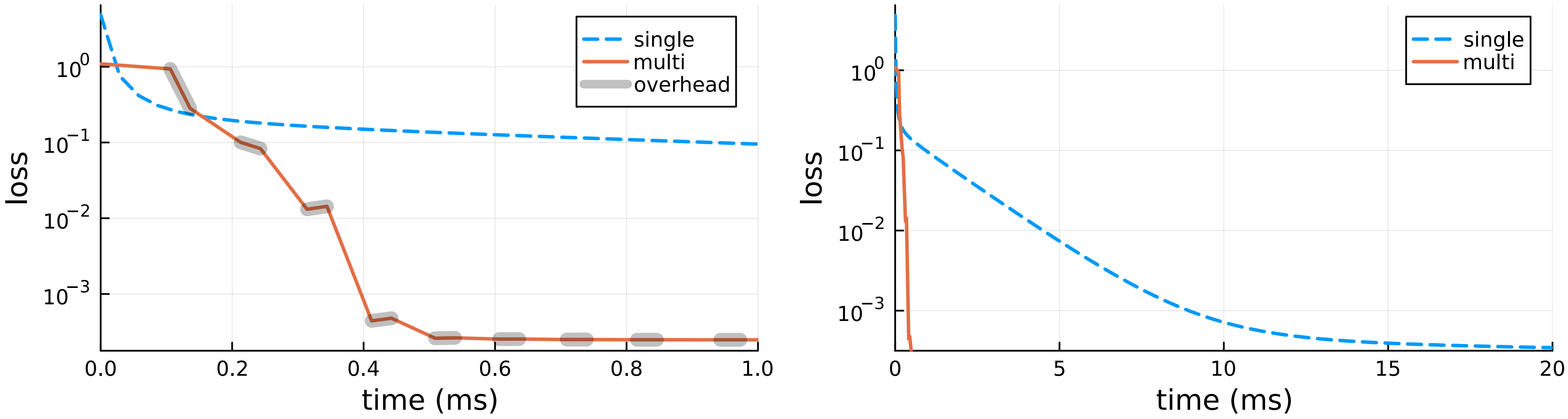

Fig 4: Objective $\tilde{\mathcal{L}}$ convergence using single scale (dashed blue) and multiscale (solid orange) for the first millisecond (left) and first 20 milliseconds (right). Grey regions highlight overhead: interpolation, allocations, and extra function calls.

How does it scale?

For large problems, turns seconds to milliseconds!

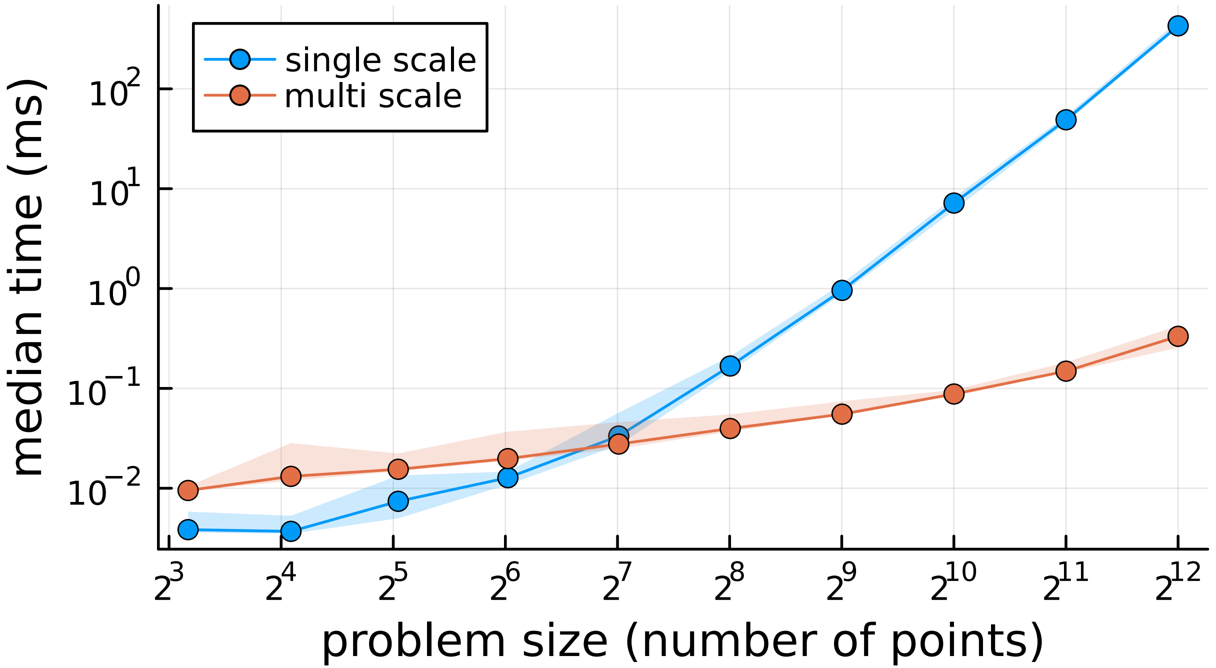

Fig 5: Median convergence times (within $5\,\%$ of optimal value) from 100 random initializations for problem size I.

The Theory

Sufficient Conditions For…

…Iterate Convergence

- Solution functions $f^* \in \arg\min_{f \in \mathcal{C}} \mathcal{L}(f)$ are L_{f^*}-Lipschitz
- Run K_s iterations at scale s using a q-linear update $x_s^{k+1} = U\left(x_s^k\right)$,

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

Sufficient Conditions For…

…Iterate Convergence

- Solution functions $f^* \in \arg\min_{f \in \mathcal{C}} \mathcal{L}(f)$ are L_{f^*}-Lipschitz
- Run K_s iterations at scale s using a q-linear update $x_s^{k+1} = U\left(x_s^k\right)$,

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

…Multiscale To Beat Single Scaled

- Initialize with i.i.d. standard normal entries $\mathcal{N}(0, 1)$
- Cost $C = DI^p$ of one iteration scales polynomially in problem size I
- Run one iteration $K_s = 1$ for coarse scales $s = S, S - 1, \ldots, 2$
- Finest scale has enough points I

Multiscale Iterate Convergence

Theorem 1: *The multiscale method returns an iterate* $x_1^{K_1}$ *satisfying*

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

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

Final error is bounded

Multiscale Iterate Convergence

Theorem 1: *The multiscale method returns an iterate* $x_1^{K_1}$ *satisfying*

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

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

Final error is bounded by the sum of the

- inner algorithm rate q times the initial error

Multiscale Iterate Convergence

Theorem 1: *The multiscale method returns an iterate* $x_1^{K_1}$ *satisfying*

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

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

Final error is bounded by the sum of the

- inner algorithm rate q times the initial error
- cumulative linear interpolation error from each scale

Conditions for Multiscale to be Better Than Single

Theorem 2: *Assume the finest scale problem has $I = 2^S + 1$ points, where the number of scales S satisfies*

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

Then, multiscale simultaneously achieves lower total cost and a tighter expected final error bound than single scale.

Conditions for Multiscale to be Better Than Single

Theorem 2: *Assume the finest scale problem has $I = 2^S + 1$ points, where the number of scales S satisfies*

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

Then, multiscale simultaneously achieves lower total cost and a tighter expected final error bound than single scale.

For small q (easier problems!), need $I \gtrsim \frac{1}{q^2}$ for multiscale to be worth it.

Note $q = 0 \implies$ can solve the fine scale problem in one iteration.

But does the discrete solution approximately solve P?

Yes!

Theorem 3: *Let* $f^* : [\ell, u] \to \mathbb{R}$ *have uniform discretization* $x^* \in \mathbb{R}^I$. *Construct the piecewise linear function* $\hat{f}$ *from* $x_1^{K_1} \in \mathbb{R}^I$.

There exist constants $C, D > 0$ *such that* $\forall \epsilon > 0$,

$$\left\| x_1^{K_1} - x^* \right\|_2^2 < C\epsilon \quad \text{and} \quad I > D\epsilon^{-1} + 1 \quad \Longrightarrow \quad \left\| \hat{f} - f^* \right\|_2 < \epsilon.$$

Solve the discrete problem accurately

But does the discrete solution approximately solve P?

Yes!

Theorem 3: *Let* $f^* : [\ell, u] \to \mathbb{R}$ *have uniform discretization* $x^* \in \mathbb{R}^I$. *Construct the piecewise linear function* $\hat{f}$ *from* $x_1^{K_1} \in \mathbb{R}^I$.

There exist constants $C, D > 0$ *such that* $\forall \epsilon > 0$,

$$\left\| x_1^{K_1} - x^* \right\|_2^2 < C\epsilon \quad \textit{and} \quad I > D\epsilon^{-1} + 1 \quad \Longrightarrow \quad \left\| \hat{f} - f^* \right\|_2 < \epsilon.$$

Solve the discrete problem accurately with enough points

But does the discrete solution approximately solve P?

Yes!

Theorem 3: *Let* $f^* : [\ell, u] \to \mathbb{R}$ *have uniform discretization* $x^* \in \mathbb{R}^I$. *Construct the piecewise linear function* $\hat{f}$ *from* $x_1^{K_1} \in \mathbb{R}^I$.

There exist constants $C, D > 0$ *such that* $\forall \epsilon > 0$,

$$\left\| x_1^{K_1} - x^* \right\|_2^2 < C\epsilon \text{ and } I > D\epsilon^{-1} + 1 \implies \left\| \hat{f} - f^* \right\|_2 < \epsilon.$$

Solve the discrete problem accurately with enough points
$\implies$ construct a function that accurately solves the continuous problem

Example:
Tensor Factorization

Factorizing at Multiple Scales (see [Gillis *et al.* 2012])

Separate mixtures $Y[i, :, :, :]$ of 3D densities $B[r, :, :, :]$ by solving

$$\min_{A, B_s} \|Y_s - AB_s\|_F^2 \quad \text{s.t.} \quad A, B_s \geq 0 \text{ and } \|A[i, :]\|_1 = 1$$

at scales $s = S, S-1, \ldots, 1$ using progressively finer resolution of slices.

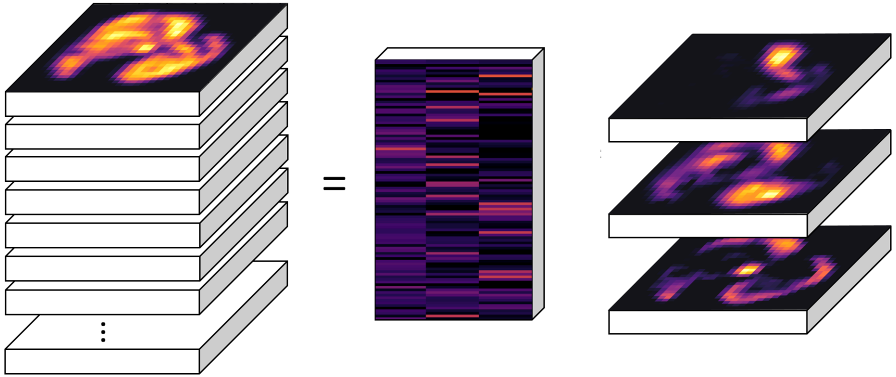

Fig 9: Example of a Tucker-1 factorization for a stack of 2D densities.

Convergence vs Number of Coarse Scale Iterations

Allowing for multiple inner iterations $K_s \implies$ drop from $2.4\,\mathrm{s}$ to $0.3\,\mathrm{s}$!

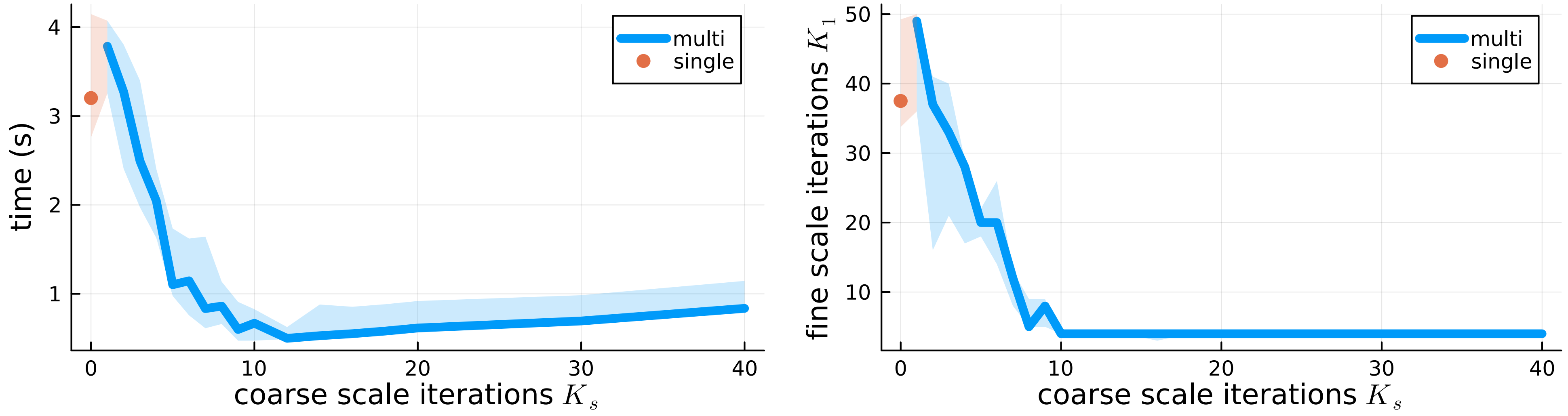

Fig 10: (Left) Median time (20 trials) to convergence at the finest scale vs number of fixed iterations K_s at coarser scales $s = S, S-1, ..., 2$. (Right) Median number of iterations K_1 at the finest scale $s = 1$.

Conclusion

- Speed up optimization problems of underlying continuous functions
- Approximate coarse solutions (cheap) to warm start fine scale iterates
- Better than single scale iterating when the problem size is big enough

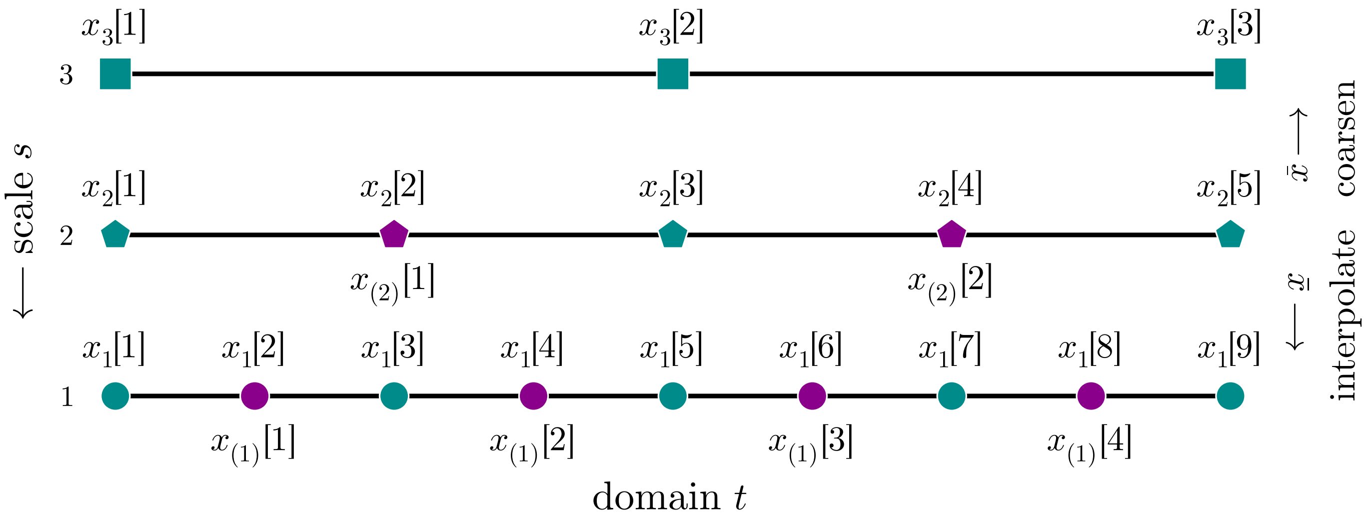

Fig 11: Discretization scheme

Thank You!

Paper [Richardson *et al.* 2025]

https://arxiv.org/abs/2512.13993

Slides

https://njericha.github.io/research

References

[1] N. J. E. Richardson, N. Marusenko, and M. P. Friedlander, “Multiple Scale Methods For Optimization Of Discretized Continuous Functions,” Dec. 2025. http://arxiv.org/abs/2512.13993

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

[3] L. N. Trefethen, *Approximation theory and approximation practice*, Extended edition., no. 164. Philadelphia: Society for Industrial, Applied Mathematics, 2019.

[4] R. Vershynin, *High-Dimensional Probability*. University of California, Irvine: Cambridge University Press, 2018.

[5] K. Gurney, *An Introduction to Neural Networks*. London: CRC Press, 2018.

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

[7] 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. http://arxiv.org/abs/2501.01869

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

[9] A. Ang, H. De Sterck, and S. Vavasis, "MGProx: A Nonsmooth Multigrid Proximal Gradient Method with Adaptive Restriction for Strongly Convex Optimization," *SIAM Journal on Optimization*, vol. 34, no. 3, Sept. 2024.

[10] N. Gillis and F. Glineur, "A multilevel approach for nonnegative matrix factorization," *Journal of Computational and Applied Mathematics*, vol. 236, no. 7, Jan. 2012.

Additional Slides

The Multiscale Algorithm

Algorithm 1: Line 6 specifies Greedy or Lazy versions.

1: Randomly initialize x_s^0 at the coarsest scale $s = S$.

2: Perform K_S iterations on x_S^0 to approx. solve P_S and obtain $x_S^{K_S}$.

3: **for** scales $s = S - 1, S - 2, \ldots, 1$ **do**

4:     Interpolate $x_{s+1}^{K_{s+1}}$ to obtain $\underline{x}_{s+1}^{K_{s+1}}$ according to Definition 2.

5:     Initialize next scale using previous solution: $x_s^0 = \underline{x}_{s+1}^{K_{s+1}}$.

6:     Perform K_s iterations on x_s^0 (Greedy) or $x_{(s)}^0$ (Lazy) to approximately solve P_s at scale s and obtain $x_s^{K_s}$.

7: **end**

8: **return** Approximate solution $x_1^{K_1}$ for P_s at $s = 1$

Coarsening and Interpolating

Definition 1 (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:*

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

Definition 2 (Interpolating): *Midpoint linear interpolation inserts a point between entries in* $x \in \mathbb{R}^I$ *to produce* $\underline{x} \in \mathbb{R}^{2I-1}$*:*

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

Continuous vs Discretize Problem

Continuous	Discrete
$\min_{f}\{\mathcal{L}(f) \mid f \in \mathcal{C}\} \quad (P)$	$\min_{x_s}\{\tilde{\mathcal{L}}_s(x_s) \mid x_s \in \tilde{\mathcal{C}}_s\} \ (P_s)$
f continuous	$x_s \in \mathbb{R}^{I_s}$
$f : \mathcal{D} \subseteq \mathbb{R} \to \mathbb{R}$	$x_s[i] = f(t_s[i])$
$\mathcal{L} : \{f : \mathcal{D} \to \mathbb{R}\} \to \mathbb{R}$	$\tilde{\mathcal{L}}_s : \mathbb{R}^{I_s} \to \mathbb{R}$
$\mathcal{C} \subseteq \{f : \mathcal{D} \to \mathbb{R}\}$	$\tilde{\mathcal{C}}_s \subseteq \mathbb{R}^{I_s}$

Updates with q-Linear Convergence

Definition 3 (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 initialization $x^0 \in \tilde{\mathcal{C}}$ and minimizer x^* of $\tilde{\mathcal{L}}(x)$ over $\tilde{\mathcal{C}}$ that may depend on x^0.

Example: projected gradient descent with stepsize $\frac{2}{\mathcal{S}+\mu}$ for $\mathcal{S}$-smooth and μ-strongly convex objective $\tilde{\mathcal{L}}$, and convex $\tilde{\mathcal{C}}$.

Handling Constraints At Multiple Scales

Type	Continuous $\mathcal{C}$	Discrete $\tilde{\mathcal{C}}_s$
nonnegative	$f : \mathcal{D} \to \mathbb{R}_+$	$x_s \geq 0$
restricted range	$f : \mathcal{D} \to \mathcal{R} \subseteq \mathbb{R}$	$x_s \in \mathcal{R}^{I_s}$
p-norm	$\\|f\\|_p = c$	$\\|x_s\\|_p = c \cdot (I_s / I_1)^{1/p}$
linear	$\langle g, f \rangle = c$ $\left(\int_{\mathcal{D}} g(t) f(t) \, dt \right)$	$\langle a, x_s \rangle = c \cdot (I_s / I_1)$ $a[i] = g(t[i]) \sqrt{\Delta t}$ $x_s[i] = f(t[i]) \sqrt{\Delta t}$

Legendre Polynomials

In the motivating example, we use normalized polynomials,

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

where the binomial coefficient

$$\binom{m}{k} = \frac{m!}{k!(m-k)!}.$$

Legendre Polynomials