Model-Based Derivative-Free Optimization: Algorithms and Approximation Theory

Part 2: Fundamentals of Model-Based DFO

Lindon Roberts, University of Melbourne (lindon.roberts@unimelb.edu.au)
Supported by the Australian Research Council (DE240100006)

MoCaO Lectures
28 July 2026

Model-Based DFO

We want to solve

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x})$$

where f is nonconvex and has L-Lipschitz continuous gradient (not accessible to algorithm!).

Model-Based DFO

We want to solve

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x})$$

where f is nonconvex and has L-Lipschitz continuous gradient (not accessible to algorithm!).

Basic approach draws on standard/derivative-based trust-region methods:

- Build a local quadratic interpolation model to approximate f near $\boldsymbol{x}_k$
- Calculate a step $\|\boldsymbol{s}_k\| \leq \Delta_k$ that decreases the model
- Evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$, and use this to accept/reject the step and choose Δ_{k+1}

e.g. [Conn et al., 2009]

Outline

1. **Main MBDFO Algorithm**
2. Convergence Theory
3. Interpolation Model Construction

Model Accuracy

Yesterday, we considered linear models based on finite differencing

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) := f(\boldsymbol{x}_k) + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k), \qquad \text{where} \quad [\boldsymbol{g}_k]_i = \frac{f(\boldsymbol{x}_k + h\boldsymbol{e}_i) - f(\boldsymbol{x}_k)}{h}$$

with error bounds

$$\|\boldsymbol{g}_k - \nabla f(\boldsymbol{x}_k)\| \leq \frac{\sqrt{n}Lh}{2} \qquad \text{and} \qquad |f(\boldsymbol{y}) - m_k(\boldsymbol{y})| \leq \frac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}_k\|^2 + \frac{\sqrt{n}Lh}{2}\|\boldsymbol{y} - \boldsymbol{x}_k\|,$$

Model Accuracy

Yesterday, we considered linear models based on finite differencing

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) := f(\boldsymbol{x}_k) + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k), \qquad \text{where} \quad [\boldsymbol{g}_k]_i = \frac{f(\boldsymbol{x}_k + h\boldsymbol{e}_i) - f(\boldsymbol{x}_k)}{h}$$

with error bounds

$$\|\boldsymbol{g}_k - \nabla f(\boldsymbol{x}_k)\| \leq \frac{\sqrt{n}Lh}{2} \qquad \text{and} \qquad |f(\boldsymbol{y}) - m_k(\boldsymbol{y})| \leq \frac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}_k\|^2 + \frac{\sqrt{n}Lh}{2}\|\boldsymbol{y} - \boldsymbol{x}_k\|,$$

For trust-region methods, our tentative step will be $\boldsymbol{x}_k + \boldsymbol{s}_k$, with $\|\boldsymbol{s}_k\| \leq \Delta_k$, and so

$$|f(\boldsymbol{x}_k + \boldsymbol{s}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)| \leq \frac{L}{2}\Delta_k^2 + \frac{\sqrt{n}Lh}{2}\Delta_k$$

Model Accuracy

Yesterday, we considered linear models based on finite differencing

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) := f(\boldsymbol{x}_k) + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k), \qquad \text{where} \quad [\boldsymbol{g}_k]_i = \frac{f(\boldsymbol{x}_k + h\boldsymbol{e}_i) - f(\boldsymbol{x}_k)}{h}$$

with error bounds

$$\|\boldsymbol{g}_k - \nabla f(\boldsymbol{x}_k)\| \leq \frac{\sqrt{n}Lh}{2} \qquad \text{and} \qquad |f(\boldsymbol{y}) - m_k(\boldsymbol{y})| \leq \frac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}_k\|^2 + \frac{\sqrt{n}Lh}{2}\|\boldsymbol{y} - \boldsymbol{x}_k\|,$$

For trust-region methods, our tentative step will be $\boldsymbol{x}_k + \boldsymbol{s}_k$, with $\|\boldsymbol{s}_k\| \leq \Delta_k$, and so

$$|f(\boldsymbol{x}_k + \boldsymbol{s}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)| \leq \frac{L}{2}\Delta_k^2 + \frac{\sqrt{n}Lh}{2}\Delta_k$$

For minimal error, pick h no bigger than Δ_k; e.g. if $h = \Delta_k$ we have

$$\|\boldsymbol{g}_k - \nabla f(\boldsymbol{x}_k)\| \leq \frac{\sqrt{n}L}{2}\Delta_k \qquad \text{and} \qquad |f(\boldsymbol{x}_k + \boldsymbol{s}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)| \leq \left(\frac{L}{2} + \frac{\sqrt{n}L}{2}\right)\Delta_k^2.$$

Model Accuracy

So far, we have two possible local, linear models for the objective, with error bounds:

Model Accuracy

So far, we have two possible local, linear models for the objective, with error bounds:

Linear Taylor Series

- Model:
  $m_k(\boldsymbol{y}) = f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^T(\boldsymbol{y} - \boldsymbol{x}_k)$
- Model error:
  $|f(\boldsymbol{y}) - m_k(\boldsymbol{y})| \leq \frac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}_k\|^2 = \mathcal{O}(\Delta_k^2)$
- Gradient error:
  $\|\nabla f(\boldsymbol{x}_k) - \nabla m_k(\boldsymbol{x}_k)\| = 0$

Finite Differencing, $h = \Delta_k$

- Model:
  $m_k(\boldsymbol{y}) = f(\boldsymbol{x}_k) + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k)$
- Model error:
  $|f(\boldsymbol{y}) - m_k(\boldsymbol{y})| = \mathcal{O}(\Delta_k^2)$
- Gradient error:
  $\|\nabla f(\boldsymbol{x}_k) - \nabla m_k(\boldsymbol{x}_k)\| \leq \mathcal{O}(\Delta_k)$

for all $\boldsymbol{y} \in B(\boldsymbol{x}_k, \Delta_k)$.

Fully Linear Models

Motivated by these, in our algorithm we assume we have a local, quadratic model for the objective

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) = f(\boldsymbol{x}_k) + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{y} - \boldsymbol{x}_k)^T H_k(\boldsymbol{y} - \boldsymbol{x}_k)$$

Fully Linear Models

Motivated by these, in our algorithm we assume we have a local, quadratic model for the objective

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) = f(\boldsymbol{x}_k) + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{y} - \boldsymbol{x}_k)^T H_k(\boldsymbol{y} - \boldsymbol{x}_k)$$

Our required notion of model accuracy is: [Conn et al., 2009]

Fully Linear Model

A quadratic model m_k is a fully linear approximation to f in $B(\boldsymbol{x}_k, \Delta_k)$ if:

$$|f(\boldsymbol{y}) - m_k(\boldsymbol{y})| \leq \kappa_{\text{mf}}\Delta_k^2 \qquad \text{and} \qquad \|\nabla f(\boldsymbol{x}_k) - \nabla m_k(\boldsymbol{x}_k)\| \leq \kappa_{\text{mg}}\Delta_k$$

for all $\boldsymbol{y} \in B(\boldsymbol{x}_k, \Delta_k)$, with $\kappa_{\text{mf}}, \kappa_{\text{mg}} \geq 0$ independent of k.

Fully Linear Models

Motivated by these, in our algorithm we assume we have a local, quadratic model for the objective

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) = f(\boldsymbol{x}_k) + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{y} - \boldsymbol{x}_k)^T H_k(\boldsymbol{y} - \boldsymbol{x}_k)$$

Our required notion of model accuracy is: [Conn et al., 2009]

Fully Linear Model

A quadratic model m_k is a fully linear approximation to f in $B(\boldsymbol{x}_k, \Delta_k)$ if:

$$|f(\boldsymbol{y}) - m_k(\boldsymbol{y})| \leq \kappa_{\text{mf}}\Delta_k^2 \qquad \text{and} \qquad \|\nabla f(\boldsymbol{x}_k) - \nabla m_k(\boldsymbol{x}_k)\| \leq \kappa_{\text{mg}}\Delta_k$$

for all $\boldsymbol{y} \in B(\boldsymbol{x}_k, \Delta_k)$, with $\kappa_{\text{mf}}, \kappa_{\text{mg}} \geq 0$ independent of k.

For example, linear Taylor series or finite differencing gradients, with $H_k = 0$.

Algorithm Preliminaries

To build our MBDFO algorithm, we just need to replace a Taylor-type local model with a fully linear model (constructed however we wish) then proceed as before: minimize the model to get step $\boldsymbol{s}_k$, evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and accept/reject, ...

Algorithm Preliminaries

To build our MBDFO algorithm, we just need to replace a Taylor-type local model with a fully linear model (constructed however we wish) then proceed as before: minimize the model to get step $\boldsymbol{s}_k$, evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and accept/reject, ...

Issue

In MBDFO, Δ_k is overloaded!

Algorithm Preliminaries

To build our MBDFO algorithm, we just need to replace a Taylor-type local model with a fully linear model (constructed however we wish) then proceed as before: minimize the model to get step $\boldsymbol{s}_k$, evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and accept/reject, ...

Issue

In MBDFO, Δ_k is overloaded!

- Controls length of step, $\|\boldsymbol{s}_k\| \leq \Delta_k$

Algorithm Preliminaries

To build our MBDFO algorithm, we just need to replace a Taylor-type local model with a fully linear model (constructed however we wish) then proceed as before: minimize the model to get step $\boldsymbol{s}_k$, evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and accept/reject, ...

Issue

In MBDFO, Δ_k is overloaded!

- Controls length of step, $\|\boldsymbol{s}_k\| \leq \Delta_k$
- Controls model accuracy, e.g. $|f(\boldsymbol{y}) - m_k(\boldsymbol{y})| = \mathcal{O}(\Delta_k^2)$

Algorithm Preliminaries

To build our MBDFO algorithm, we just need to replace a Taylor-type local model with a fully linear model (constructed however we wish) then proceed as before: minimize the model to get step $\boldsymbol{s}_k$, evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and accept/reject, ...

Issue

In MBDFO, Δ_k is overloaded!

- Controls length of step, $\|\boldsymbol{s}_k\| \leq \Delta_k$
- Controls model accuracy, e.g. $|f(\boldsymbol{y}) - m_k(\boldsymbol{y})| = \mathcal{O}(\Delta_k^2)$
- Controls termination: stop when $\Delta_k \leq \Delta_{\text{min}}$ (can't check $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon$, and checking $\|\boldsymbol{g}_k\| \leq \epsilon$ only works if model error—i.e. Δ_k—is small!)

Algorithm Preliminaries

To build our MBDFO algorithm, we just need to replace a Taylor-type local model with a fully linear model (constructed however we wish) then proceed as before: minimize the model to get step $\boldsymbol{s}_k$, evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and accept/reject, ...

Issue

In MBDFO, Δ_k is overloaded!

- Controls length of step, $\|\boldsymbol{s}_k\| \leq \Delta_k$
- Controls model accuracy, e.g. $|f(\boldsymbol{y}) - m_k(\boldsymbol{y})| = \mathcal{O}(\Delta_k^2)$
- Controls termination: stop when $\Delta_k \leq \Delta_{\min}$ (can't check $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon$, and checking $\|\boldsymbol{g}_k\| \leq \epsilon$ only works if model error—i.e. Δ_k—is small!)

This requires one small modification to the basic algorithm (*criticality check*)

MBDFO

With this, one iteration of the basic MBDFO algorithm is:

- Given $\boldsymbol{x}_k$ and Δ_k, build fully linear model m_k
- Calculate a step, $\boldsymbol{s}_k \approx \arg\min_{\|\boldsymbol{s}\| \leq \Delta_k} m_k(\boldsymbol{x}_k + \boldsymbol{s})$
- Evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and ratio $\rho_k = [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{s}_k)]/[m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)]$
- Accept/reject the step and update trust-region radius:

MBDFO

With this, one iteration of the basic MBDFO algorithm is:

- Given $\boldsymbol{x}_k$ and Δ_k, build fully linear model m_k
- Calculate a step, $\boldsymbol{s}_k \approx \arg\min_{\|\boldsymbol{s}\| \leq \Delta_k} m_k(\boldsymbol{x}_k + \boldsymbol{s})$
- Evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and ratio $\rho_k = [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{s}_k)]/[m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)]$
- Accept/reject the step and update trust-region radius:
  - (*Very successful*) If $\rho_k \geq 0.7$ and $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = 2\Delta_k$
  - (*Successful*) If $0.1 \leq \rho_k < 0.7$ and $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = \Delta_k$
  - (*Unsuccessful*) If $\rho_k < 0.1$ or $\|\boldsymbol{g}_k\| < \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and $\Delta_{k+1} = \frac{1}{2}\Delta_k$

MBDFO

With this, one iteration of the basic MBDFO algorithm is:

- Given $\boldsymbol{x}_k$ and Δ_k, build fully linear model m_k
- Calculate a step, $\boldsymbol{s}_k \approx \arg\min_{\|\boldsymbol{s}\| \leq \Delta_k} m_k(\boldsymbol{x}_k + \boldsymbol{s})$
- Evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and ratio $\rho_k = [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{s}_k)]/[m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)]$
- Accept/reject the step and update trust-region radius:
  - (*Very successful*) If $\rho_k \geq 0.7$ and $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = 2\Delta_k$
  - (*Successful*) If $0.1 \leq \rho_k < 0.7$ and $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = \Delta_k$
  - (*Unsuccessful*) If $\rho_k < 0.1$ or $\|\boldsymbol{g}_k\| < \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and $\Delta_{k+1} = \frac{1}{2}\Delta_k$

Criticality check: if $\|\boldsymbol{g}_k\| \ll \Delta_k$, shrink Δ_k (without calculating $\boldsymbol{s}_k$). Usually $\mu \ll 1$.

MBDFO

With this, one iteration of the basic MBDFO algorithm is:

- Given $\boldsymbol{x}_k$ and Δ_k, build fully linear model m_k
- Calculate a step, $\boldsymbol{s}_k \approx \arg\min_{\|\boldsymbol{s}\| \leq \Delta_k} m_k(\boldsymbol{x}_k + \boldsymbol{s})$
- Evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and ratio $\rho_k = [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{s}_k)]/[m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)]$
- Accept/reject the step and update trust-region radius:
  - (*Very successful*) If $\rho_k \geq 0.7$ and $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = 2\Delta_k$
  - (*Successful*) If $0.1 \leq \rho_k < 0.7$ and $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = \Delta_k$
  - (*Unsuccessful*) If $\rho_k < 0.1$ or $\|\boldsymbol{g}_k\| < \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and $\Delta_{k+1} = \frac{1}{2}\Delta_k$

Criticality check: if $\|\boldsymbol{g}_k\| \ll \Delta_k$, shrink Δ_k (without calculating $\boldsymbol{s}_k$). Usually $\mu \ll 1$.

i.e. If the algorithm thinks we're near a solution, shrink Δ_k to improve model accuracy, so we can be sure.

Step Calculation

Aside from model construction, the other key part of the algorithm is the trust-region subproblem (step calculation): e.g. [Conn et al., 2000]

$$\min_{\boldsymbol{s}\in\mathbb{R}^n} m(\boldsymbol{s}) := \boldsymbol{g}^T\boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T H\boldsymbol{s} \quad \text{s.t. } \|\boldsymbol{s}\| \leq \Delta$$

Step Calculation

Aside from model construction, the other key part of the algorithm is the trust-region subproblem (step calculation): e.g. [Conn et al., 2000]

$$\min_{\boldsymbol{s}\in\mathbb{R}^n} m(\boldsymbol{s}) := \boldsymbol{g}^T\boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T H\boldsymbol{s} \quad \text{s.t. } \|\boldsymbol{s}\| \leq \Delta$$

This problem is nonconvex (from H), but we can still characterize *global* minima.

Step Calculation

Aside from model construction, the other key part of the algorithm is the trust-region subproblem (step calculation): e.g. [Conn et al., 2000]

$$\min_{\boldsymbol{s}\in\mathbb{R}^n} m(\boldsymbol{s}) := \boldsymbol{g}^T\boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T H\boldsymbol{s} \quad \text{s.t. } \|\boldsymbol{s}\| \leq \Delta$$

This problem is nonconvex (from H), but we can still characterize *global* minima.

Theorem

A point $\boldsymbol{s}$ is a global minimizer if and only if the KKT conditions (with Lagrange multiplier $\lambda \in \mathbb{R}$)

$$(H + \lambda I)\boldsymbol{s} = -\boldsymbol{g},$$
$$\|\boldsymbol{s}\| \leq \Delta, \quad \lambda \geq 0, \quad \lambda(\Delta - \|\boldsymbol{s}\|) = 0,$$

hold and $H + \lambda I$ is positive semidefinite (unique minimizer if positive definite).

Step Calculation

This gives us a way to find a global minimizer of the trust-region subproblem:

Step Calculation

This gives us a way to find a global minimizer of the trust-region subproblem:

- Interior solution: $\boldsymbol{s} = -H^{-1}\boldsymbol{g}$ if H positive definite and $\|\boldsymbol{s}\| \leq \Delta$ (i.e. $\lambda = 0$)

Step Calculation

This gives us a way to find a global minimizer of the trust-region subproblem:

- Interior solution: $\boldsymbol{s} = -H^{-1}\boldsymbol{g}$ if H positive definite and $\|\boldsymbol{s}\| \leq \Delta$ (i.e. $\lambda = 0$)
- Boundary solution: $\lambda > 0$, so $\|\boldsymbol{s}\| = \Delta$ and $\boldsymbol{s} = -(H + \lambda I)^{-1}\boldsymbol{g}$ — 1D rootfinding problem in λ

Step Calculation

This gives us a way to find a global minimizer of the trust-region subproblem:

- Interior solution: $\boldsymbol{s} = -H^{-1}\boldsymbol{g}$ if H positive definite and $\|\boldsymbol{s}\| \leq \Delta$ (i.e. $\lambda = 0$)
- Boundary solution: $\lambda > 0$, so $\|\boldsymbol{s}\| = \Delta$ and $\boldsymbol{s} = -(H + \lambda I)^{-1}\boldsymbol{g}$ — 1D rootfinding problem in λ
- Hard case: if H negative semidefinite and $\boldsymbol{g}$ orthogonal to eigenspace of $\lambda_{\min}(H)$, then $(H + \lambda I)\boldsymbol{s} = -\boldsymbol{g}$ for $\lambda = -\lambda_{\min}(H)$ is singular but consistent (pick boundary solution)

Step Calculation

This gives us a way to find a global minimizer of the trust-region subproblem:

- Interior solution: $\boldsymbol{s} = -H^{-1}\boldsymbol{g}$ if H positive definite and $\|\boldsymbol{s}\| \leq \Delta$ (i.e. $\lambda = 0$)
- Boundary solution: $\lambda > 0$, so $\|\boldsymbol{s}\| = \Delta$ and $\boldsymbol{s} = -(H + \lambda I)^{-1}\boldsymbol{g}$ — 1D rootfinding problem in λ
- Hard case: if H negative semidefinite and $\boldsymbol{g}$ orthogonal to eigenspace of $\lambda_{\min}(H)$, then $(H + \lambda I)\boldsymbol{s} = -\boldsymbol{g}$ for $\lambda = -\lambda_{\min}(H)$ is singular but consistent (pick boundary solution)

For low-dimensional problems (e.g. $n = \mathcal{O}(100)$), finding a global minimizer is practical *and does improve the performance of MBDFO methods*.

Step Calculation

This gives us a way to find a global minimizer of the trust-region subproblem:

- Interior solution: $\boldsymbol{s} = -H^{-1}\boldsymbol{g}$ if H positive definite and $\|\boldsymbol{s}\| \leq \Delta$ (i.e. $\lambda = 0$)
- Boundary solution: $\lambda > 0$, so $\|\boldsymbol{s}\| = \Delta$ and $\boldsymbol{s} = -(H + \lambda I)^{-1}\boldsymbol{g}$ — 1D rootfinding problem in λ
- Hard case: if H negative semidefinite and $\boldsymbol{g}$ orthogonal to eigenspace of $\lambda_{\min}(H)$, then $(H + \lambda I)\boldsymbol{s} = -\boldsymbol{g}$ for $\lambda = -\lambda_{\min}(H)$ is singular but consistent (pick boundary solution)

For low-dimensional problems (e.g. $n = \mathcal{O}(100)$), finding a global minimizer is practical *and does improve the performance of MBDFO methods*.

However, for theory (and large-scale problems), approximate minimizers are acceptable...

Approximate Step Calculation

The simplest approximate minimizer is the Cauchy point — steepest descent with exact linesearch:

$$\boldsymbol{s}_C = -\alpha_C \boldsymbol{g}, \qquad \alpha_C = \underset{\alpha>0}{\arg\min}\, m(-\alpha \boldsymbol{g}) \quad \text{s.t. } \|\alpha \boldsymbol{g}\| \leq \Delta$$

Approximate Step Calculation

The simplest approximate minimizer is the Cauchy point — steepest descent with exact linesearch:

$$\boldsymbol{s}_C = -\alpha_C \boldsymbol{g}, \qquad \alpha_C = \underset{\alpha>0}{\arg\min}\, m(-\alpha \boldsymbol{g}) \quad \text{s.t. } \|\alpha \boldsymbol{g}\| \leq \Delta$$

If objective convex along this direction, get (constrained) minimizer, otherwise objective unbounded below (so must be at the boundary). Hence

$$\alpha_C = \begin{cases} \min\left(\frac{\|\boldsymbol{g}\|}{\boldsymbol{g}^T H \boldsymbol{g}}, \frac{\Delta}{\|\boldsymbol{g}\|}\right), & \boldsymbol{g}^T H \boldsymbol{g} > 0, \\ \frac{\Delta}{\|\boldsymbol{g}\|}, & \boldsymbol{g}^T H \boldsymbol{g} \leq 0 \end{cases}$$

Approximate Step Calculation

The simplest approximate minimizer is the Cauchy point — steepest descent with exact linesearch:

$$\boldsymbol{s}_C = -\alpha_C \boldsymbol{g}, \qquad \alpha_C = \underset{\alpha>0}{\arg\min}\, m(-\alpha \boldsymbol{g}) \quad \text{s.t. } \|\alpha \boldsymbol{g}\| \leq \Delta$$

If objective convex along this direction, get (constrained) minimizer, otherwise objective unbounded below (so must be at the boundary). Hence

$$\alpha_C = \begin{cases} \min\left(\frac{\|\boldsymbol{g}\|}{\boldsymbol{g}^T H \boldsymbol{g}}, \frac{\Delta}{\|\boldsymbol{g}\|}\right), & \boldsymbol{g}^T H \boldsymbol{g} > 0, \\ \frac{\Delta}{\|\boldsymbol{g}\|}, & \boldsymbol{g}^T H \boldsymbol{g} \leq 0 \end{cases}$$

which gives objective reduction of at least

$$m(\mathbf{0}) - m(\boldsymbol{s}_C) \geq \frac{1}{2}\|\boldsymbol{g}\| \min\left(\Delta, \frac{\|\boldsymbol{g}\|}{\|H\|+1}\right)$$

Approximate Step Calculation

The simplest approximate minimizer is the Cauchy point — steepest descent with exact linesearch:

$$\boldsymbol{s}_C = -\alpha_C \boldsymbol{g}, \qquad \alpha_C = \underset{\alpha>0}{\arg\min}\, m(-\alpha \boldsymbol{g}) \quad \text{s.t. } \|\alpha \boldsymbol{g}\| \leq \Delta$$

If objective convex along this direction, get (constrained) minimizer, otherwise objective unbounded below (so must be at the boundary). Hence

$$\alpha_C = \begin{cases} \min\left(\frac{\|\boldsymbol{g}\|}{\boldsymbol{g}^T H \boldsymbol{g}}, \frac{\Delta}{\|\boldsymbol{g}\|}\right), & \boldsymbol{g}^T H \boldsymbol{g} > 0, \\ \frac{\Delta}{\|\boldsymbol{g}\|}, & \boldsymbol{g}^T H \boldsymbol{g} \leq 0 \end{cases}$$

which gives objective reduction of at least

$$m(\mathbf{0}) - m(\boldsymbol{s}_C) \geq \frac{1}{2}\|\boldsymbol{g}\| \min\left(\Delta, \frac{\|\boldsymbol{g}\|}{\|H\| + 1}\right)$$

This is the minimum required by theory, but is not enough to be practically useful.

CG Solver

An intermediate alternative is to use the conjugate gradient method.

CG Solver

An intermediate alternative is to use the conjugate gradient method. CG solves symmetric positive definite linear systems $H\boldsymbol{s} = -\boldsymbol{g}$ via minimizing

$$\min_{\boldsymbol{s}} \boldsymbol{g}^T\boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T H\boldsymbol{s}$$

i.e. our subproblem without the constraint, and with H positive definite.

CG Solver

An intermediate alternative is to use the conjugate gradient method. CG solves symmetric positive definite linear systems $H\boldsymbol{s} = -\boldsymbol{g}$ via minimizing

$$\min_{\boldsymbol{s}} \boldsymbol{g}^T\boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T H\boldsymbol{s}$$

i.e. our subproblem without the constraint, and with H positive definite.

Idea: run CG without modification until:

CG Solver

An intermediate alternative is to use the conjugate gradient method. CG solves symmetric positive definite linear systems $\boldsymbol{Hs} = -\boldsymbol{g}$ via minimizing

$$\min_{\boldsymbol{s}} \boldsymbol{g}^T \boldsymbol{s} + \frac{1}{2} \boldsymbol{s}^T \boldsymbol{H} \boldsymbol{s}$$

i.e. our subproblem without the constraint, and with H positive definite.

Idea: run CG without modification until:

- Solution $\boldsymbol{Hs} = -\boldsymbol{g}$ reached (interior global minimizer)

CG Solver

An intermediate alternative is to use the conjugate gradient method. CG solves symmetric positive definite linear systems $\boldsymbol{Hs} = -\boldsymbol{g}$ via minimizing

$$\min_{\boldsymbol{s}} \boldsymbol{g}^T \boldsymbol{s} + \frac{1}{2} \boldsymbol{s}^T \boldsymbol{H} \boldsymbol{s}$$

i.e. our subproblem without the constraint, and with H positive definite.

Idea: run CG without modification until:

- Solution $\boldsymbol{Hs} = -\boldsymbol{g}$ reached (interior global minimizer)
- $\|\boldsymbol{s}\| > \Delta$ reached — shrink step to $\|\boldsymbol{s}\| = \Delta$ and terminate

CG Solver

An intermediate alternative is to use the conjugate gradient method. CG solves symmetric positive definite linear systems $H\boldsymbol{s} = -\boldsymbol{g}$ via minimizing

$$\min_{\boldsymbol{s}} \boldsymbol{g}^T\boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T H\boldsymbol{s}$$

i.e. our subproblem without the constraint, and with H positive definite.

Idea: run CG without modification until:

- Solution $H\boldsymbol{s} = -\boldsymbol{g}$ reached (interior global minimizer)
- $\|\boldsymbol{s}\| > \Delta$ reached — shrink step to $\|\boldsymbol{s}\| = \Delta$ and terminate
- Negative curvature detected — follow this direction until $\|\boldsymbol{s}\| = \Delta$ and terminate

CG Solver

An intermediate alternative is to use the conjugate gradient method. CG solves symmetric positive definite linear systems $H\boldsymbol{s} = -\boldsymbol{g}$ via minimizing

$$\min_{\boldsymbol{s}} \boldsymbol{g}^T\boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T H\boldsymbol{s}$$

i.e. our subproblem without the constraint, and with H positive definite.

Idea: run CG without modification until:

- Solution $H\boldsymbol{s} = -\boldsymbol{g}$ reached (interior global minimizer)
- $\|\boldsymbol{s}\| > \Delta$ reached — shrink step to $\|\boldsymbol{s}\| = \Delta$ and terminate
- Negative curvature detected — follow this direction until $\|\boldsymbol{s}\| = \Delta$ and terminate

This is cheap to implement for large-scale problems (Hessian-vector products only), and provides a reasonable model decrease in practice.

CG Solver

An intermediate alternative is to use the conjugate gradient method. CG solves symmetric positive definite linear systems $H\boldsymbol{s} = -\boldsymbol{g}$ via minimizing

$$\min_{\boldsymbol{s}} \boldsymbol{g}^T\boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T H\boldsymbol{s}$$

i.e. our subproblem without the constraint, and with H positive definite.

Idea: run CG without modification until:

- Solution $H\boldsymbol{s} = -\boldsymbol{g}$ reached (interior global minimizer)
- $\|\boldsymbol{s}\| > \Delta$ reached — shrink step to $\|\boldsymbol{s}\| = \Delta$ and terminate
- Negative curvature detected — follow this direction until $\|\boldsymbol{s}\| = \Delta$ and terminate

This is cheap to implement for large-scale problems (Hessian-vector products only), and provides a reasonable model decrease in practice. Always achieves Cauchy decrease.

Outline

1. Main MBDFO Algorithm
2. **Convergence Theory**
3. Interpolation Model Construction

Main Algorithm

Let's now prove global convergence (i.e. converges for any $\boldsymbol{x}_0$) of our algorithm

- Given $\boldsymbol{x}_k$ and Δ_k, build fully linear model m_k
- Calculate a step, $\boldsymbol{s}_k \approx \arg\min_{\|\boldsymbol{s}\| \leq \Delta_k} m_k(\boldsymbol{x}_k + \boldsymbol{s})$
- Evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and ratio $\rho_k = [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{s}_k)]/[m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)]$
- Accept/reject the step and update trust-region radius:
  - (*Very successful*) If $\rho_k \geq 0.7$ and $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = 2\Delta_k$
  - (*Successful*) If $0.1 \leq \rho_k < 0.7$ and $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = \Delta_k$
  - (*Unsuccessful*) If $\rho_k < 0.1$ or $\|\boldsymbol{g}_k\| < \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and $\Delta_{k+1} = \frac{1}{2}\Delta_k$

Main Algorithm

Let's now prove global convergence (i.e. converges for any $\boldsymbol{x}_0$) of our algorithm

- Given $\boldsymbol{x}_k$ and Δ_k, build fully linear model m_k
- Calculate a step, $\boldsymbol{s}_k \approx \arg\min_{\|\boldsymbol{s}\| \leq \Delta_k} m_k(\boldsymbol{x}_k + \boldsymbol{s})$
- Evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and ratio $\rho_k = [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{s}_k)]/[m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)]$
- Accept/reject the step and update trust-region radius:
  - (*Very successful*) If $\rho_k \geq 0.7$ and $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = 2\Delta_k$
  - (*Successful*) If $0.1 \leq \rho_k < 0.7$ and $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = \Delta_k$
  - (*Unsuccessful*) If $\rho_k < 0.1$ or $\|\boldsymbol{g}_k\| < \mu\Delta_k$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and $\Delta_{k+1} = \frac{1}{2}\Delta_k$

Assumptions:

- ∇f L-Lipschitz continuous and f bounded below, $f(\boldsymbol{x}) \geq f_{\text{low}}$ for all $\boldsymbol{x} \in \mathbb{R}^n$
- Step $\boldsymbol{s}_k$ always achieves Cauchy decrease in the model
- Uniformly bounded model Hessians, $\|H_k\| + 1 \leq \kappa_H$ for all k

Convergence: Step 1

Lemma

If $\boldsymbol{g}_k \neq \mathbf{0}$ and $\Delta_k \leq C_0 \|\boldsymbol{g}_k\|$, where $C_0 := \min\left(\frac{0.3}{4\kappa_{\text{mf}}}, \frac{1}{\kappa_H}, \frac{1}{\mu}\right)$, then iteration k is very successful (i.e. $\rho_k \geq 0.7$ and $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$).

Convergence: Step 1

Lemma

If $\boldsymbol{g}_k \neq \mathbf{0}$ and $\Delta_k \leq C_0 \|\boldsymbol{g}_k\|$, where $C_0 := \min\left(\frac{0.3}{4\kappa_{\text{mf}}}, \frac{1}{\kappa_H}, \frac{1}{\mu}\right)$, then iteration k is very successful (i.e. $\rho_k \geq 0.7$ and $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$).

Proof: By definition of C_0, we have $\Delta_k \leq \frac{\|\boldsymbol{g}_k\|}{\kappa_H} \leq \frac{\|\boldsymbol{g}_k\|}{\|H_k\|+1}$, so Cauchy decrease gives

$$m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k) \geq \frac{1}{2}\|\boldsymbol{g}_k\| \min\left(\Delta_k, \frac{\|\boldsymbol{g}_k\|}{\|H_k\|+1}\right) = \frac{1}{2}\|\boldsymbol{g}_k\|\Delta_k.$$

Convergence: Step 1

Lemma

If $\boldsymbol{g}_k \neq \mathbf{0}$ and $\Delta_k \leq C_0 \|\boldsymbol{g}_k\|$, where $C_0 := \min\left(\frac{0.3}{4\kappa_{\text{mf}}}, \frac{1}{\kappa_H}, \frac{1}{\mu}\right)$, then iteration k is very successful (i.e. $\rho_k \geq 0.7$ and $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$).

Proof: By definition of C_0, we have $\Delta_k \leq \frac{\|\boldsymbol{g}_k\|}{\kappa_H} \leq \frac{\|\boldsymbol{g}_k\|}{\|H_k\|+1}$, so Cauchy decrease gives

$$m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k) \geq \frac{1}{2}\|\boldsymbol{g}_k\| \min\left(\Delta_k, \frac{\|\boldsymbol{g}_k\|}{\|H_k\|+1}\right) = \frac{1}{2}\|\boldsymbol{g}_k\|\Delta_k.$$

Then, by definition of ρ_k and using fully linear models, we have

$$|\rho_k - 1| = \frac{|[f(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k)] - [f(\boldsymbol{x}_k + \boldsymbol{s}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)]|}{|m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)|} \leq \frac{2\kappa_{\text{mf}}\Delta_k^2}{\frac{1}{2}\|\boldsymbol{g}_k\|\Delta_k} \leq 4\kappa_{\text{mf}}C_0$$

Convergence: Step 1

Lemma

If $\boldsymbol{g}_k \neq \mathbf{0}$ and $\Delta_k \leq C_0 \|\boldsymbol{g}_k\|$, where $C_0 := \min\left(\frac{0.3}{4\kappa_{\text{mf}}}, \frac{1}{\kappa_H}, \frac{1}{\mu}\right)$, then iteration k is very successful (i.e. $\rho_k \geq 0.7$ and $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$).

Proof: By definition of C_0, we have $\Delta_k \leq \frac{\|\boldsymbol{g}_k\|}{\kappa_H} \leq \frac{\|\boldsymbol{g}_k\|}{\|H_k\|+1}$, so Cauchy decrease gives

$$m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k) \geq \frac{1}{2}\|\boldsymbol{g}_k\| \min\left(\Delta_k, \frac{\|\boldsymbol{g}_k\|}{\|H_k\|+1}\right) = \frac{1}{2}\|\boldsymbol{g}_k\|\Delta_k.$$

Then, by definition of ρ_k and using fully linear models, we have

$$|\rho_k - 1| = \frac{|[f(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k)] - [f(\boldsymbol{x}_k + \boldsymbol{s}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)]|}{|m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)|} \leq \frac{2\kappa_{\text{mf}}\Delta_k^2}{\frac{1}{2}\|\boldsymbol{g}_k\|\Delta_k} \leq 4\kappa_{\text{mf}}C_0$$

so $|\rho_k - 1| \leq 0.3$, which implies $\rho_k \geq 0.7$. $\|\boldsymbol{g}_k\| \geq \mu\Delta_k$ is trivial by definition of C_0.

Convergence: Step 2

Lemma

If $\|\nabla f(\boldsymbol{x}_k)\| \geq \epsilon$ for all $k = 0, \ldots, K-1$, then $\Delta_k \geq \Delta_{\min}(\epsilon)$ for all $k = 0, \ldots, K$, where $\Delta_{\min}(\epsilon) := \min(\Delta_0, \frac{\epsilon/2}{\kappa_{\text{mg}} + C_0^{-1}})$. Note $\Delta_{\min}(\epsilon) = \mathcal{O}(\epsilon)$.

Convergence: Step 2

Lemma

If $\|\nabla f(\boldsymbol{x}_k)\| \geq \epsilon$ for all $k = 0, \ldots, K-1$, then $\Delta_k \geq \Delta_{\min}(\epsilon)$ for all $k = 0, \ldots, K$, where $\Delta_{\min}(\epsilon) := \min(\Delta_0, \frac{\epsilon/2}{\kappa_{\text{mg}} + C_0^{-1}})$. Note $\Delta_{\min}(\epsilon) = \mathcal{O}(\epsilon)$.

Proof: Result is trivial for $k = 0$, suppose it holds for some $k = 0, \ldots, K-1$. For a contradiction, suppose $\Delta_{k+1} < \Delta_{\min}(\epsilon)$.

Convergence: Step 2

Lemma

If $\|\nabla f(\boldsymbol{x}_k)\| \geq \epsilon$ for all $k = 0, \ldots, K-1$, then $\Delta_k \geq \Delta_{\min}(\epsilon)$ for all $k = 0, \ldots, K$, where $\Delta_{\min}(\epsilon) := \min(\Delta_0, \frac{\epsilon/2}{\kappa_{\text{mg}} + C_0^{-1}})$. Note $\Delta_{\min}(\epsilon) = \mathcal{O}(\epsilon)$.

Proof: Result is trivial for $k = 0$, suppose it holds for some $k = 0, \ldots, K-1$. For a contradiction, suppose $\Delta_{k+1} < \Delta_{\min}(\epsilon)$.

Then $\Delta_{k+1} < \Delta_{\min}(\epsilon) \leq \Delta_k$, i.e. iteration k must be unsuccessful.

Convergence: Step 2

Lemma

If $\|\nabla f(\boldsymbol{x}_k)\| \geq \epsilon$ for all $k = 0, \ldots, K-1$, then $\Delta_k \geq \Delta_{\min}(\epsilon)$ for all $k = 0, \ldots, K$, where $\Delta_{\min}(\epsilon) := \min(\Delta_0, \frac{\epsilon/2}{\kappa_{\text{mg}} + C_0^{-1}})$. Note $\Delta_{\min}(\epsilon) = \mathcal{O}(\epsilon)$.

Proof: Result is trivial for $k = 0$, suppose it holds for some $k = 0, \ldots, K-1$. For a contradiction, suppose $\Delta_{k+1} < \Delta_{\min}(\epsilon)$.

Then $\Delta_{k+1} < \Delta_{\min}(\epsilon) \leq \Delta_k$, i.e. iteration k must be unsuccessful. By previous lemma, we must have $\Delta_k > C_0 \|\boldsymbol{g}_k\|$.

Convergence: Step 2

Lemma

If $\|\nabla f(\boldsymbol{x}_k)\| \geq \epsilon$ for all $k = 0, \ldots, K-1$, then $\Delta_k \geq \Delta_{\min}(\epsilon)$ for all $k = 0, \ldots, K$, where $\Delta_{\min}(\epsilon) := \min(\Delta_0, \frac{\epsilon/2}{\kappa_{\text{mg}} + C_0^{-1}})$. Note $\Delta_{\min}(\epsilon) = \mathcal{O}(\epsilon)$.

Proof: Result is trivial for $k = 0$, suppose it holds for some $k = 0, \ldots, K-1$. For a contradiction, suppose $\Delta_{k+1} < \Delta_{\min}(\epsilon)$.

Then $\Delta_{k+1} < \Delta_{\min}(\epsilon) \leq \Delta_k$, i.e. iteration k must be unsuccessful. By previous lemma, we must have $\Delta_k > C_0 \|\boldsymbol{g}_k\|$.

Since model is fully linear,

$$\epsilon \leq \|\nabla f(\boldsymbol{x}_k)\| \leq \|\boldsymbol{g}_k\| + \|\nabla f(\boldsymbol{x}_k) - \boldsymbol{g}_k\| < C_0^{-1}\Delta_k + \kappa_{\text{mg}}\Delta_k$$

Convergence: Step 2

Lemma

If $\|\nabla f(\boldsymbol{x}_k)\| \geq \epsilon$ for all $k = 0, \ldots, K-1$, then $\Delta_k \geq \Delta_{\min}(\epsilon)$ for all $k = 0, \ldots, K$, where $\Delta_{\min}(\epsilon) := \min(\Delta_0, \frac{\epsilon/2}{\kappa_{\text{mg}} + C_0^{-1}})$. Note $\Delta_{\min}(\epsilon) = \mathcal{O}(\epsilon)$.

Proof: Result is trivial for $k = 0$, suppose it holds for some $k = 0, \ldots, K-1$. For a contradiction, suppose $\Delta_{k+1} < \Delta_{\min}(\epsilon)$.

Then $\Delta_{k+1} < \Delta_{\min}(\epsilon) \leq \Delta_k$, i.e. iteration k must be unsuccessful. By previous lemma, we must have $\Delta_k > C_0 \|\boldsymbol{g}_k\|$.

Since model is fully linear,

$$\epsilon \leq \|\nabla f(\boldsymbol{x}_k)\| \leq \|\boldsymbol{g}_k\| + \|\nabla f(\boldsymbol{x}_k) - \boldsymbol{g}_k\| < C_0^{-1}\Delta_k + \kappa_{\text{mg}}\Delta_k$$

Hence $\Delta_k > \frac{\epsilon}{\kappa_{\text{mg}} + C_0^{-1}}$, so $\Delta_{k+1} = \frac{1}{2}\Delta_k \geq \Delta_{\min}(\epsilon)$, a contradiction.

Convergence: Main Result

We can now prove a global convergence and worst-case complexity bound. Suppose $\|\nabla f(\boldsymbol{x}_k)\| \geq \epsilon$ for all $k = 0, \ldots, K-1$ — we seek an upper bound on K.

Convergence: Main Result

We can now prove a global convergence and worst-case complexity bound. Suppose $\|\nabla f(\boldsymbol{x}_k)\| \geq \epsilon$ for all $k = 0, \ldots, K-1$ — we seek an upper bound on K.

Partition iterations into (very) successful and unsuccessful, $\{0, \ldots, K-1\} = \mathcal{S} \cup \mathcal{U}$, and note previous lemma gives $\Delta_k \geq \Delta_{\min}(\epsilon)$ for all such k.

Convergence: Main Result

We can now prove a global convergence and worst-case complexity bound. Suppose $\|\nabla f(\boldsymbol{x}_k)\| \geq \epsilon$ for all $k = 0, \ldots, K-1$ — we seek an upper bound on K.

Partition iterations into (very) successful and unsuccessful, $\{0, \ldots, K-1\} = \mathcal{S} \cup \mathcal{U}$, and note previous lemma gives $\Delta_k \geq \Delta_{\min}(\epsilon)$ for all such k.

Successful iterations: Cauchy decrease and criticality ($\|\boldsymbol{g}_k\| \geq \mu\Delta_k$) give

$$\begin{aligned} m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k) &\geq \frac{1}{2}\|\boldsymbol{g}_k\| \min\left(\Delta_k, \frac{\|\boldsymbol{g}_k\|}{\|H_k\|+1}\right) \\ &\geq \frac{1}{2}\mu\Delta_{\min}(\epsilon) \min\left(\Delta_{\min}(\epsilon), \frac{\mu\Delta_{\min}(\epsilon)}{\kappa_H}\right) \quad = \mathcal{O}(\Delta_{\min}(\epsilon)^2) \end{aligned}$$

Convergence: Main Result

We can now prove a global convergence and worst-case complexity bound. Suppose $\|\nabla f(\boldsymbol{x}_k)\| \geq \epsilon$ for all $k = 0, \ldots, K-1$ — we seek an upper bound on K.

Partition iterations into (very) successful and unsuccessful, $\{0, \ldots, K-1\} = \mathcal{S} \cup \mathcal{U}$, and note previous lemma gives $\Delta_k \geq \Delta_{\min}(\epsilon)$ for all such k.

Successful iterations: Cauchy decrease and criticality ($\|\boldsymbol{g}_k\| \geq \mu\Delta_k$) give

$$\begin{aligned} m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k) &\geq \frac{1}{2}\|\boldsymbol{g}_k\| \min\left(\Delta_k, \frac{\|\boldsymbol{g}_k\|}{\|H_k\|+1}\right) \\ &\geq \frac{1}{2}\mu\Delta_{\min}(\epsilon) \min\left(\Delta_{\min}(\epsilon), \frac{\mu\Delta_{\min}(\epsilon)}{\kappa_H}\right) \quad = \mathcal{O}(\Delta_{\min}(\epsilon)^2) \end{aligned}$$

So, for successful steps, $f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \geq 0.1[m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)] \geq C_1\Delta_{\min}(\epsilon)^2$.

Convergence: Main Result

Successful iterations: since $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ if unsuccessful, over K iterations we have

$$f(\boldsymbol{x}_0) - f(\boldsymbol{x}_K) = \sum_{k=0}^{K-1} [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})] = \sum_{k \in \mathcal{S}} [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})] \geq C_1 \Delta_{\text{min}}(\epsilon)^2 \cdot |\mathcal{S}|$$

Convergence: Main Result

Successful iterations: since $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ if unsuccessful, over K iterations we have

$$f(\boldsymbol{x}_0) - f(\boldsymbol{x}_K) = \sum_{k=0}^{K-1} [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})] = \sum_{k \in \mathcal{S}} [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})] \geq C_1 \Delta_{\min}(\epsilon)^2 \cdot |\mathcal{S}|$$

Bounding LHS by $f(\boldsymbol{x}_0) - f_{\text{low}}$ gives a bound on the number of successful iterations

$$|\mathcal{S}| \leq \frac{f(\boldsymbol{x}_0) - f_{\text{low}}}{C_1 \Delta_{\min}(\epsilon)^2} = \mathcal{O}(\epsilon^{-2})$$

Convergence: Main Result

Successful iterations: since $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ if unsuccessful, over K iterations we have

$$f(\boldsymbol{x}_0) - f(\boldsymbol{x}_K) = \sum_{k=0}^{K-1} [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})] = \sum_{k \in \mathcal{S}} [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})] \geq C_1 \Delta_{\text{min}}(\epsilon)^2 \cdot |\mathcal{S}|$$

Bounding LHS by $f(\boldsymbol{x}_0) - f_{\text{low}}$ gives a bound on the number of successful iterations

$$|\mathcal{S}| \leq \frac{f(\boldsymbol{x}_0) - f_{\text{low}}}{C_1 \Delta_{\text{min}}(\epsilon)^2} = \mathcal{O}(\epsilon^{-2})$$

Unsuccessful iterations: we have $\Delta_k \geq \Delta_{\text{min}}(\epsilon)$, and $\Delta_{k+1} \leq 2\Delta_k$ for $k \in \mathcal{S}$ and $\Delta_{k+1} = 0.5\Delta_k$ for $k \in \mathcal{U}$.

Convergence: Main Result

Successful iterations: since $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ if unsuccessful, over K iterations we have

$$f(\boldsymbol{x}_0) - f(\boldsymbol{x}_K) = \sum_{k=0}^{K-1} [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})] = \sum_{k \in \mathcal{S}} [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})] \geq C_1 \Delta_{\text{min}}(\epsilon)^2 \cdot |\mathcal{S}|$$

Bounding LHS by $f(\boldsymbol{x}_0) - f_{\text{low}}$ gives a bound on the number of successful iterations

$$|\mathcal{S}| \leq \frac{f(\boldsymbol{x}_0) - f_{\text{low}}}{C_1 \Delta_{\text{min}}(\epsilon)^2} = \mathcal{O}(\epsilon^{-2})$$

Unsuccessful iterations: we have $\Delta_k \geq \Delta_{\text{min}}(\epsilon)$, and $\Delta_{k+1} \leq 2\Delta_k$ for $k \in \mathcal{S}$ and $\Delta_{k+1} = 0.5\Delta_k$ for $k \in \mathcal{U}$. Over K iterations, we get

$$\Delta_{\text{min}}(\epsilon) \leq \Delta_K \leq 2^{|\mathcal{S}| - |\mathcal{U}|} \Delta_0$$

Convergence: Main Result

Successful iterations: since $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ if unsuccessful, over K iterations we have

$$f(\boldsymbol{x}_0) - f(\boldsymbol{x}_K) = \sum_{k=0}^{K-1} [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})] = \sum_{k \in \mathcal{S}} [f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})] \geq C_1 \Delta_{\min}(\epsilon)^2 \cdot |\mathcal{S}|$$

Bounding LHS by $f(\boldsymbol{x}_0) - f_{\text{low}}$ gives a bound on the number of successful iterations

$$|\mathcal{S}| \leq \frac{f(\boldsymbol{x}_0) - f_{\text{low}}}{C_1 \Delta_{\min}(\epsilon)^2} = \mathcal{O}(\epsilon^{-2})$$

Unsuccessful iterations: we have $\Delta_k \geq \Delta_{\min}(\epsilon)$, and $\Delta_{k+1} \leq 2\Delta_k$ for $k \in \mathcal{S}$ and $\Delta_{k+1} = 0.5\Delta_k$ for $k \in \mathcal{U}$. Over K iterations, we get

$$\Delta_{\min}(\epsilon) \leq \Delta_K \leq 2^{|\mathcal{S}| - |\mathcal{U}|} \Delta_0$$

So, $|\mathcal{U}| \leq |\mathcal{S}| + \log_2(\Delta_0 / \Delta_{\min}(\epsilon)) = \mathcal{O}(\epsilon^{-2}) + \mathcal{O}(\log(\epsilon^{-1})) = \mathcal{O}(\epsilon^{-2})$.

Convergence: Main Result

All together, we have a **worst-case complexity** bound and **global convergence**:

Theorem

$\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ first occurs after $\mathcal{O}(\epsilon^{-2})$ iterations. Hence $\liminf \|\nabla f(\boldsymbol{x}_k)\| = 0$.

Convergence: Main Result

All together, we have a **worst-case complexity** bound and **global convergence**:

Theorem

$\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ first occurs after $\mathcal{O}(\epsilon^{-2})$ iterations. Hence $\liminf \|\nabla f(\boldsymbol{x}_k)\| = 0$.

How does this compare to derivative-based methods?

Convergence: Main Result

All together, we have a **worst-case complexity** bound and **global convergence**:

Theorem

$\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ first occurs after $\mathcal{O}(\epsilon^{-2})$ iterations. Hence $\liminf \|\nabla f(\boldsymbol{x}_k)\| = 0$.

How does this compare to derivative-based methods?

- $\mathcal{O}(\epsilon^{-2})$ iterations is standard if we have exact gradients e.g. [Cartis et al., 2022]

Convergence: Main Result

All together, we have a **worst-case complexity** bound and **global convergence**:

Theorem

$\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ first occurs after $\mathcal{O}(\epsilon^{-2})$ iterations. Hence $\liminf \|\nabla f(\boldsymbol{x}_k)\| = 0$.

How does this compare to derivative-based methods?

- $\mathcal{O}(\epsilon^{-2})$ iterations is standard if we have exact gradients e.g. [Cartis et al., 2022]
- The penalty is in the size of the constant!

Convergence: Main Result

All together, we have a **worst-case complexity** bound and **global convergence**:

Theorem

$\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ first occurs after $\mathcal{O}(\epsilon^{-2})$ iterations. Hence $\liminf \|\nabla f(\boldsymbol{x}_k)\| = 0$.

How does this compare to derivative-based methods?

- $\mathcal{O}(\epsilon^{-2})$ iterations is standard if we have exact gradients e.g. [Cartis et al., 2022]
- The penalty is in the size of the constant! If $\kappa_{\text{m}} := \max(\kappa_{\text{mf}}, \kappa_{\text{mg}})$, we have
  - $\mathcal{O}((\kappa_{\text{m}} + \kappa_H)\epsilon^{-2})$ iterations for gradient-based trust-region methods
  - $\mathcal{O}(\kappa_H(\kappa_{\text{m}} + \kappa_H)^2\epsilon^{-2})$ iterations for MBDFO

Convergence: Main Result

All together, we have a **worst-case complexity** bound and **global convergence**:

Theorem

$\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ first occurs after $\mathcal{O}(\epsilon^{-2})$ iterations. Hence $\liminf \|\nabla f(\boldsymbol{x}_k)\| = 0$.

How does this compare to derivative-based methods?

- $\mathcal{O}(\epsilon^{-2})$ iterations is standard if we have exact gradients e.g. [Cartis et al., 2022]
- The penalty is in the size of the constant! If $\kappa_{\text{m}} := \max(\kappa_{\text{mf}}, \kappa_{\text{mg}})$, we have
  - $\mathcal{O}((\kappa_{\text{m}} + \kappa_H)\epsilon^{-2})$ iterations for gradient-based trust-region methods
  - $\mathcal{O}(\kappa_H(\kappa_{\text{m}} + \kappa_H)^2\epsilon^{-2})$ iterations for MBDFO

The difference comes from $\|\boldsymbol{g}_k\| \geq \mu\Delta_{\min}(\epsilon)$ vs. $\|\boldsymbol{g}_k\| = \|\nabla f(\boldsymbol{x}_k)\| \geq \epsilon$ in Cauchy decrease for successful steps.

Convergence: Main Result

All together, we have a **worst-case complexity** bound and **global convergence**:

Theorem

$\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ first occurs after $\mathcal{O}(\epsilon^{-2})$ iterations. Hence $\liminf \|\nabla f(\boldsymbol{x}_k)\| = 0$.

How does this compare to derivative-based methods?

- $\mathcal{O}(\epsilon^{-2})$ iterations is standard if we have exact gradients e.g. [Cartis et al., 2022]
- The penalty is in the size of the constant! If $\kappa_{\text{m}} := \max(\kappa_{\text{mf}}, \kappa_{\text{mg}})$, we have
  - $\mathcal{O}((\kappa_{\text{m}} + \kappa_H)\epsilon^{-2})$ iterations for gradient-based trust-region methods
  - $\mathcal{O}(\kappa_H(\kappa_{\text{m}} + \kappa_H)^2\epsilon^{-2})$ iterations for MBDFO

The difference comes from $\|\boldsymbol{g}_k\| \geq \mu\Delta_{\min}(\epsilon)$ vs. $\|\boldsymbol{g}_k\| = \|\nabla f(\boldsymbol{x}_k)\| \geq \epsilon$ in Cauchy decrease for successful steps. Also, κ_{m} is usually larger for MBDFO.

Termination

Since we don't know $\|\nabla f(\boldsymbol{x}_k)\|$, we can't terminate when this gets sufficiently small.

In practice, we terminate the algorithm:

Termination

Since we don't know $\|\nabla f(\boldsymbol{x}_k)\|$, we can't terminate when this gets sufficiently small.

In practice, we terminate the algorithm:

- After some maximum budget (usually # objective evaluations)
- When Δ_k is sufficiently small

Termination

Since we don't know $\|\nabla f(\boldsymbol{x}_k)\|$, we can't terminate when this gets sufficiently small.

In practice, we terminate the algorithm:

- After some maximum budget (usually # objective evaluations)
- When Δ_k is sufficiently small

The latter is justified by the following result:

Theorem

We have $\lim_{k\to\infty} \Delta_k = 0$, and when $\Delta_{k+1} < \Delta_{\text{min}}$ for the first time we have $\|\nabla f(\boldsymbol{x}_k)\| \leq \mathcal{O}(\Delta_{\text{min}})$.

Outline

1. Main MBDFO Algorithm
2. Convergence Theory
3. **Interpolation Model Construction**

Interpolation Models

Our algorithm now has convergence theory, provided we can construct fully linear models at every iteration.

Interpolation Models

Our algorithm now has convergence theory, provided we can construct fully linear models at every iteration.

In practice, we use linear or quadratic interpolation over points near $\boldsymbol{x}_k$ to define the model m_k.

Interpolation Models

Our algorithm now has convergence theory, provided we can construct fully linear models at every iteration.

In practice, we use linear or quadratic interpolation over points near $\boldsymbol{x}_k$ to define the model m_k. If the model is a good approximation to a Taylor series, then it is fully linear.

Interpolation Models

Our algorithm now has convergence theory, provided we can construct fully linear models at every iteration.

In practice, we use linear or quadratic interpolation over points near $\boldsymbol{x}_k$ to define the model m_k. If the model is a good approximation to a Taylor series, then it is fully linear.

Theorem [LR, 2025]

If m_k is a quadratic model such that

$$|m_k(\boldsymbol{y}) - f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_k)^T(\boldsymbol{y} - \boldsymbol{x}_k)| \leq \kappa \Delta_k^2$$

for all $\boldsymbol{y} \in B(\boldsymbol{x}_k, \Delta_k)$, and $\|H_k\| \leq \kappa_H$, then m_k is fully linear with $\kappa_{\text{mf}} = \kappa + \frac{L}{2}$ and $\kappa_{\text{mg}} = 2\kappa + L + 2\kappa_H$.

Linear Interpolation

First, let's build a **linear interpolation model**:

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) = c_k + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k).$$

Linear Interpolation

First, let's build a **linear interpolation model**:

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) = c_k + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k).$$

We have $p = n + 1$ degrees of freedom, so assume we have interpolation points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p$ where we know the value of f.

Linear Interpolation

First, let's build a **linear interpolation model**:

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) = c_k + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k).$$

We have $p = n + 1$ degrees of freedom, so assume we have interpolation points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p$ where we know the value of f.

The unique model satisfying $m_k(\boldsymbol{y}_t) = f(\boldsymbol{y}_t)$ is given by the $p \times p$ linear system

$$\begin{bmatrix} 1 & (\boldsymbol{y}_1 - \boldsymbol{x}_k)^T \\ \vdots & \vdots \\ 1 & (\boldsymbol{y}_p - \boldsymbol{x}_k)^T \end{bmatrix} \begin{bmatrix} c_k \\ \boldsymbol{g}_k \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{y}_1) \\ \vdots \\ f(\boldsymbol{y}_p) \end{bmatrix}$$

Linear Interpolation

First, let's build a **linear interpolation model**:

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) = c_k + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k).$$

We have $p = n + 1$ degrees of freedom, so assume we have interpolation points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p$ where we know the value of f.

The unique model satisfying $m_k(\boldsymbol{y}_t) = f(\boldsymbol{y}_t)$ is given by the $p \times p$ linear system

$$\begin{bmatrix} 1 & (\boldsymbol{y}_1 - \boldsymbol{x}_k)^T \\ \vdots & \vdots \\ 1 & (\boldsymbol{y}_p - \boldsymbol{x}_k)^T \end{bmatrix} \begin{bmatrix} c_k \\ \boldsymbol{g}_k \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{y}_1) \\ \vdots \\ f(\boldsymbol{y}_p) \end{bmatrix}$$

We shall see that we want $\|\boldsymbol{y}_t - \boldsymbol{x}_k\| = \mathcal{O}(\Delta_k)$.

Linear Interpolation

First, let's build a **linear interpolation model**:

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) = c_k + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k).$$

We have $p = n + 1$ degrees of freedom, so assume we have interpolation points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p$ where we know the value of f.

The unique model satisfying $m_k(\boldsymbol{y}_t) = f(\boldsymbol{y}_t)$ is given by the $p \times p$ linear system

$$\begin{bmatrix} 1 & (\boldsymbol{y}_1 - \boldsymbol{x}_k)^T \\ \vdots & \vdots \\ 1 & (\boldsymbol{y}_p - \boldsymbol{x}_k)^T \end{bmatrix} \begin{bmatrix} c_k \\ \boldsymbol{g}_k \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{y}_1) \\ \vdots \\ f(\boldsymbol{y}_p) \end{bmatrix}$$

We shall see that we want $\|\boldsymbol{y}_t - \boldsymbol{x}_k\| = \mathcal{O}(\Delta_k)$. But $\Delta_k \to 0$, so the matrix becomes increasingly ill-conditioned!

Linear Interpolation

Instead, we rescale the linear system and solve

$$\underbrace{\begin{bmatrix} 1 & \hat{\boldsymbol{s}}_1^T \\ \vdots & \vdots \\ 1 & \hat{\boldsymbol{s}}_p^T \end{bmatrix}}_{=:\hat{M}} \begin{bmatrix} c_k \\ \Delta_k \cdot \boldsymbol{g}_k \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{y}_1) \\ \vdots \\ f(\boldsymbol{y}_p) \end{bmatrix}$$

where $\hat{\boldsymbol{s}}_t := (\boldsymbol{y}_t - \boldsymbol{x}_k)/\Delta_k$.

Linear Interpolation

Instead, we rescale the linear system and solve

$$\underbrace{\begin{bmatrix} 1 & \hat{\boldsymbol{s}}_1^T \\ \vdots & \vdots \\ 1 & \hat{\boldsymbol{s}}_p^T \end{bmatrix}}_{=:\hat{M}} \begin{bmatrix} c_k \\ \Delta_k \cdot \boldsymbol{g}_k \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{y}_1) \\ \vdots \\ f(\boldsymbol{y}_p) \end{bmatrix}$$

where $\hat{\boldsymbol{s}}_t := (\boldsymbol{y}_t - \boldsymbol{x}_k)/\Delta_k$.

Is $\hat{M}$ ill-conditioned?

Linear Interpolation

Instead, we rescale the linear system and solve

$$\underbrace{\begin{bmatrix} 1 & \hat{\boldsymbol{s}}_1^T \\ \vdots & \vdots \\ 1 & \hat{\boldsymbol{s}}_p^T \end{bmatrix}}_{=:\hat{M}} \begin{bmatrix} c_k \\ \Delta_k \cdot \boldsymbol{g}_k \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{y}_1) \\ \vdots \\ f(\boldsymbol{y}_p) \end{bmatrix}$$

where $\hat{\boldsymbol{s}}_t := (\boldsymbol{y}_t - \boldsymbol{x}_k)/\Delta_k$.

Is $\hat{M}$ ill-conditioned? Choosing $\|\boldsymbol{y}_t - \boldsymbol{x}_k\| = \mathcal{O}(\Delta_k)$ ensures $\|\hat{M}\|$ is not too large, but $\|\hat{M}^{-1}\|$ may still be large!

Linear Interpolation

Instead, we rescale the linear system and solve

$$\underbrace{\begin{bmatrix} 1 & \hat{\boldsymbol{s}}_1^T \\ \vdots & \vdots \\ 1 & \hat{\boldsymbol{s}}_p^T \end{bmatrix}}_{=:\hat{M}} \begin{bmatrix} c_k \\ \Delta_k \cdot \boldsymbol{g}_k \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{y}_1) \\ \vdots \\ f(\boldsymbol{y}_p) \end{bmatrix}$$

where $\hat{\boldsymbol{s}}_t := (\boldsymbol{y}_t - \boldsymbol{x}_k)/\Delta_k$.

Is $\hat{M}$ ill-conditioned? Choosing $\|\boldsymbol{y}_t - \boldsymbol{x}_k\| = \mathcal{O}(\Delta_k)$ ensures $\|\hat{M}\|$ is not too large, but $\|\hat{M}^{-1}\|$ may still be large!

Our fully linear error bounds reflect this...

Linear Interpolation

Theorem

If $\|\boldsymbol{y}_t - \boldsymbol{x}_k\| \leq \beta\Delta_k$ for all t and $\hat{M}$ is invertible, then the model is fully linear with $\kappa_{\text{mf}}, \kappa_{\text{mg}} = \mathcal{O}(L\sqrt{n}\beta^2\|\hat{M}^{-1}\|_\infty)$.

Linear Interpolation

Theorem

If $\|\boldsymbol{y}_t - \boldsymbol{x}_k\| \leq \beta\Delta_k$ for all t and $\hat{M}$ is invertible, then the model is fully linear with $\kappa_{\text{mf}}, \kappa_{\text{mg}} = \mathcal{O}(L\sqrt{n}\beta^2\|\hat{M}^{-1}\|_\infty)$.

Proof: Since $f(\boldsymbol{y}_t) = m_k(\boldsymbol{y}_t)$, Taylor error bounds give

$$|m_k(\boldsymbol{y}_t) - f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_k)^T(\boldsymbol{y}_t - \boldsymbol{x}_k)| \leq \frac{L}{2}\|\boldsymbol{y}_t - \boldsymbol{x}_k\|^2 \leq \frac{L}{2}\beta^2\Delta_k^2.$$

Linear Interpolation

Theorem

If $\|\boldsymbol{y}_t - \boldsymbol{x}_k\| \leq \beta\Delta_k$ for all t and $\hat{M}$ is invertible, then the model is fully linear with $\kappa_{\text{mf}}, \kappa_{\text{mg}} = \mathcal{O}(L\sqrt{n}\beta^2\|\hat{M}^{-1}\|_\infty)$.

Proof: Since $f(\boldsymbol{y}_t) = m_k(\boldsymbol{y}_t)$, Taylor error bounds give

$$|m_k(\boldsymbol{y}_t) - f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_k)^T(\boldsymbol{y}_t - \boldsymbol{x}_k)| \leq \frac{L}{2}\|\boldsymbol{y}_t - \boldsymbol{x}_k\|^2 \leq \frac{L}{2}\beta^2\Delta_k^2.$$

Now pick any $\boldsymbol{y} \in B(\boldsymbol{x}_k, \Delta_k)$, with $\hat{\boldsymbol{s}} := (\boldsymbol{y} - \boldsymbol{x}_k)/\Delta_k \in B(\boldsymbol{0}, 1)$.

Linear Interpolation

Theorem

If $\|\boldsymbol{y}_t - \boldsymbol{x}_k\| \leq \beta\Delta_k$ for all t and $\hat{M}$ is invertible, then the model is fully linear with $\kappa_{\text{mf}}, \kappa_{\text{mg}} = \mathcal{O}(L\sqrt{n}\beta^2\|\hat{M}^{-1}\|_\infty)$.

Proof: Since $f(\boldsymbol{y}_t) = m_k(\boldsymbol{y}_t)$, Taylor error bounds give

$$|m_k(\boldsymbol{y}_t) - f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_k)^T(\boldsymbol{y}_t - \boldsymbol{x}_k)| \leq \frac{L}{2}\|\boldsymbol{y}_t - \boldsymbol{x}_k\|^2 \leq \frac{L}{2}\beta^2\Delta_k^2.$$

Now pick any $\boldsymbol{y} \in B(\boldsymbol{x}_k, \Delta_k)$, with $\hat{\boldsymbol{s}} := (\boldsymbol{y} - \boldsymbol{x}_k)/\Delta_k \in B(\boldsymbol{0}, 1)$. Since $\hat{M}$ is invertible, there exists a unique solution to

$$\hat{M}^T\boldsymbol{v} = \begin{bmatrix} 1 \\ \hat{\boldsymbol{s}} \end{bmatrix} \qquad \text{or} \qquad \sum_{t=1}^{p} v_t \begin{bmatrix} 1 \\ \boldsymbol{y}_t - \boldsymbol{x}_k \end{bmatrix} = \begin{bmatrix} 1 \\ \boldsymbol{y} - \boldsymbol{x}_k \end{bmatrix}$$

Theorem

If $\|\boldsymbol{y}_t - \boldsymbol{x}_k\| \leq \beta\Delta_k$ for all t and $\hat{M}$ is invertible, then the model is fully linear with $\kappa_{\text{mf}}, \kappa_{\text{mg}} = \mathcal{O}(L\sqrt{n}\beta^2\|\hat{M}^{-1}\|_\infty)$.

Proof: Since $f(\boldsymbol{y}_t) = m_k(\boldsymbol{y}_t)$, Taylor error bounds give

$$|m_k(\boldsymbol{y}_t) - f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_k)^T(\boldsymbol{y}_t - \boldsymbol{x}_k)| \leq \frac{L}{2}\|\boldsymbol{y}_t - \boldsymbol{x}_k\|^2 \leq \frac{L}{2}\beta^2\Delta_k^2.$$

Now pick any $\boldsymbol{y} \in B(\boldsymbol{x}_k, \Delta_k)$, with $\hat{\boldsymbol{s}} := (\boldsymbol{y} - \boldsymbol{x}_k)/\Delta_k \in B(\boldsymbol{0}, 1)$. Since $\hat{M}$ is invertible, there exists a unique solution to

$$\hat{M}^T\boldsymbol{v} = \begin{bmatrix} 1 \\ \hat{\boldsymbol{s}} \end{bmatrix} \qquad \text{or} \qquad \sum_{t=1}^{p} v_t \begin{bmatrix} 1 \\ \boldsymbol{y}_t - \boldsymbol{x}_k \end{bmatrix} = \begin{bmatrix} 1 \\ \boldsymbol{y} - \boldsymbol{x}_k \end{bmatrix}$$

Note $\|\boldsymbol{v}\|_1 \leq \|\hat{M}^{-T}\|_1(1 + \|\hat{\boldsymbol{s}}\|_1) \leq \|\hat{M}^{-1}\|_\infty(1 + \sqrt{n})$, using $\|\hat{\boldsymbol{s}}\| \leq 1$.

Linear Interpolation

$$
\begin{aligned}
&|m_k(\boldsymbol{y}) - f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_k)^T(\boldsymbol{y} - \boldsymbol{x}_k)| \\
&\quad = \left| \begin{bmatrix} c \\ \boldsymbol{g}_k \end{bmatrix}^T \begin{bmatrix} 1 \\ \boldsymbol{y} - \boldsymbol{x}_k \end{bmatrix} - \begin{bmatrix} f(\boldsymbol{x}_k) \\ \nabla f(\boldsymbol{x}_k) \end{bmatrix}^T \begin{bmatrix} 1 \\ \boldsymbol{y} - \boldsymbol{x}_k \end{bmatrix} \right| \\
&\quad = \left| \sum_{t=1}^{p} v_t \left(\begin{bmatrix} c \\ \boldsymbol{g}_k \end{bmatrix}^T \begin{bmatrix} 1 \\ \boldsymbol{y}_t - \boldsymbol{x}_k \end{bmatrix} - \begin{bmatrix} f(\boldsymbol{x}_k) \\ \nabla f(\boldsymbol{x}_k) \end{bmatrix}^T \begin{bmatrix} 1 \\ \boldsymbol{y}_t - \boldsymbol{x}_k \end{bmatrix} \right) \right| \\
&\quad = \left| \sum_{t=1}^{p} v_t [m_k(\boldsymbol{y}_t) - f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_k)^T(\boldsymbol{y}_t - \boldsymbol{x}_k)] \right| \\
&\quad \leq \frac{L}{2} \beta^2 \Delta_k^2 \cdot \|\boldsymbol{v}\|_1
\end{aligned}
$$

Linear Interpolation

$$
\begin{aligned}
&|m_k(\boldsymbol{y}) - f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_k)^T(\boldsymbol{y} - \boldsymbol{x}_k)| \\
&\quad = \left| \begin{bmatrix} c \\ \boldsymbol{g}_k \end{bmatrix}^T \begin{bmatrix} 1 \\ \boldsymbol{y} - \boldsymbol{x}_k \end{bmatrix} - \begin{bmatrix} f(\boldsymbol{x}_k) \\ \nabla f(\boldsymbol{x}_k) \end{bmatrix}^T \begin{bmatrix} 1 \\ \boldsymbol{y} - \boldsymbol{x}_k \end{bmatrix} \right| \\
&\quad = \left| \sum_{t=1}^{p} v_t \left(\begin{bmatrix} c \\ \boldsymbol{g}_k \end{bmatrix}^T \begin{bmatrix} 1 \\ \boldsymbol{y}_t - \boldsymbol{x}_k \end{bmatrix} - \begin{bmatrix} f(\boldsymbol{x}_k) \\ \nabla f(\boldsymbol{x}_k) \end{bmatrix}^T \begin{bmatrix} 1 \\ \boldsymbol{y}_t - \boldsymbol{x}_k \end{bmatrix} \right) \right| \\
&\quad = \left| \sum_{t=1}^{p} v_t [m_k(\boldsymbol{y}_t) - f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_k)^T(\boldsymbol{y}_t - \boldsymbol{x}_k)] \right| \\
&\quad \leq \frac{L}{2} \beta^2 \Delta_k^2 \cdot \|\boldsymbol{v}\|_1
\end{aligned}
$$

Then apply bound on $\|\boldsymbol{v}\|_1$ and "Taylor approximation implies fully linear" theorem.

Example Sets

For example, finite differencing with stepsize $h = \Delta_k$ is equivalent to linear interpolation with points $\{\boldsymbol{x}_k, \boldsymbol{x}_k + \Delta_k \boldsymbol{e}_1, \ldots, \boldsymbol{x}_k + \Delta_k \boldsymbol{e}_n\}$.

Example Sets

For example, finite differencing with stepsize $h = \Delta_k$ is equivalent to linear interpolation with points $\{\boldsymbol{x}_k, \boldsymbol{x}_k + \Delta_k \boldsymbol{e}_1, \ldots, \boldsymbol{x}_k + \Delta_k \boldsymbol{e}_n\}$. This gives $\beta = 1$ and $\|\hat{M}^{-1}\|_\infty = 2$, and so $\kappa_{\text{mf}}, \kappa_{\text{mg}} = \mathcal{O}(\sqrt{n})$.

Example Sets

For example, finite differencing with stepsize $h = \Delta_k$ is equivalent to linear interpolation with points $\{\boldsymbol{x}_k, \boldsymbol{x}_k + \Delta_k \boldsymbol{e}_1, \ldots, \boldsymbol{x}_k + \Delta_k \boldsymbol{e}_n\}$. This gives $\beta = 1$ and $\|\hat{M}^{-1}\|_\infty = 2$, and so $\kappa_{\text{mf}}, \kappa_{\text{mg}} = \mathcal{O}(\sqrt{n})$.

Corollary

Using this interpolation set, we get $\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ after at most $\mathcal{O}(n\epsilon^{-2})$ iterations or $\mathcal{O}(n^2\epsilon^{-2})$ evaluations.

Example Sets

For example, finite differencing with stepsize $h = \Delta_k$ is equivalent to linear interpolation with points $\{\boldsymbol{x}_k, \boldsymbol{x}_k + \Delta_k \boldsymbol{e}_1, \ldots, \boldsymbol{x}_k + \Delta_k \boldsymbol{e}_n\}$. This gives $\beta = 1$ and $\|\hat{M}^{-1}\|_\infty = 2$, and so $\kappa_{\text{mf}}, \kappa_{\text{mg}} = \mathcal{O}(\sqrt{n})$.

Corollary

Using this interpolation set, we get $\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ after at most $\mathcal{O}(n\epsilon^{-2})$ iterations or $\mathcal{O}(n^2\epsilon^{-2})$ evaluations.

By taking perturbations of size $\Delta_k/\sqrt{n}$, we can improve this (in terms of explicit n dependence) to a (likely) optimal result.

Example Sets

For example, finite differencing with stepsize $h = \Delta_k$ is equivalent to linear interpolation with points $\{\boldsymbol{x}_k, \boldsymbol{x}_k + \Delta_k \boldsymbol{e}_1, \ldots, \boldsymbol{x}_k + \Delta_k \boldsymbol{e}_n\}$. This gives $\beta = 1$ and $\|\hat{M}^{-1}\|_\infty = 2$, and so $\kappa_{\text{mf}}, \kappa_{\text{mg}} = \mathcal{O}(\sqrt{n})$.

Corollary

Using this interpolation set, we get $\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ after at most $\mathcal{O}(n\epsilon^{-2})$ iterations or $\mathcal{O}(n^2\epsilon^{-2})$ evaluations.

By taking perturbations of size $\Delta_k/\sqrt{n}$, we can improve this (in terms of explicit n dependence) to a (likely) optimal result.

Corollary

Using coordinate perturbations of size $\Delta_k/\sqrt{n}$, we get $\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ after at most $\mathcal{O}(\epsilon^{-2})$ iterations or $\mathcal{O}(n\epsilon^{-2})$ evaluations.

Quadratic Interpolation

In practice, linear models are not particularly good*, but quadratic models are generally quite effective.

* exc., e.g. [Cartis & LR, 2019]

Quadratic Interpolation

In practice, linear models are not particularly good*, but quadratic models are generally quite effective.

* exc., e.g. [Cartis & LR, 2019]

To build a quadratic model

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) := c_k + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{y} - \boldsymbol{x}_k)^T H_k(\boldsymbol{y} - \boldsymbol{x}_k)$$

requires $p = 1 + n + n(n+1)/2 = (n+1)(n+2)/2$ degrees of freedom/interpolation points.

Quadratic Interpolation

In practice, linear models are not particularly good*, but quadratic models are generally quite effective.

* exc., e.g. [Cartis & LR, 2019]

To build a quadratic model

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) := c_k + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{y} - \boldsymbol{x}_k)^T H_k(\boldsymbol{y} - \boldsymbol{x}_k)$$

requires $p = 1 + n + n(n+1)/2 = (n+1)(n+2)/2$ degrees of freedom/interpolation points.

Even for moderate n, this can mean many interpolation points, and $\mathcal{O}(p^3) = \mathcal{O}(n^6)$ cost of solving linear systems.

Quadratic Interpolation

In practice, linear models are not particularly good*, but quadratic models are generally quite effective.

* exc., e.g. [Cartis & LR, 2019]

To build a quadratic model

$$f(\boldsymbol{y}) \approx m_k(\boldsymbol{y}) := c_k + \boldsymbol{g}_k^T(\boldsymbol{y} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{y} - \boldsymbol{x}_k)^T H_k(\boldsymbol{y} - \boldsymbol{x}_k)$$

requires $p = 1 + n + n(n+1)/2 = (n+1)(n+2)/2$ degrees of freedom/interpolation points.

Even for moderate n, this can mean many interpolation points, and $\mathcal{O}(p^3) = \mathcal{O}(n^6)$ cost of solving linear systems.

However, we do get higher accuracy (fully quadratic $\approx$ 2nd order Taylor accurate) which allows convergence to second-order optimal points ($\|\nabla f(\boldsymbol{x})\| = 0$ and $\nabla^2 f(\boldsymbol{x}) \succeq 0$).

Quadratic Interpolation

A practically successful middle ground is underdetermined quadratic interpolation (i.e. $n+1 < p < (n+1)(n+2)/2$ interpolation points).

Quadratic Interpolation

A practically successful middle ground is underdetermined quadratic interpolation (i.e. $n+1 < p < (n+1)(n+2)/2$ interpolation points).

Interpolation system is underdetermined — infinitely many possible models satisfy the interpolation conditions. Two common choices, both with theory:

Quadratic Interpolation

A practically successful middle ground is underdetermined quadratic interpolation (i.e. $n+1 < p < (n+1)(n+2)/2$ interpolation points).

Interpolation system is underdetermined — infinitely many possible models satisfy the interpolation conditions. Two common choices, both with theory:

- Minimum norm solution of linear system [Conn et al., 2008]
- Minimum Frobenius norm (MFN) interpolation [Powell, 2004]

Quadratic Interpolation

A practically successful middle ground is underdetermined quadratic interpolation (i.e. $n+1 < p < (n+1)(n+2)/2$ interpolation points).

Interpolation system is underdetermined — infinitely many possible models satisfy the interpolation conditions. Two common choices, both with theory:

- Minimum norm solution of linear system [Conn et al., 2008]
- Minimum Frobenius norm (MFN) interpolation [Powell, 2004]

The most widely used MBDFO software use MFN interpolation.

Minimum Frobenius Norm Interpolation

For MFN, given interpolation points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p$ (with $n+1 \leq p \leq (n+1)(n+2)/2$), derive a quadratic model via

$$\min_{c_k, \boldsymbol{g}_k, H_k} \|H_k\|_F^2 \qquad \text{s.t.} \quad H_k = H_k^T \text{ and } m_k(\boldsymbol{y}_t) = f(\boldsymbol{y}_t) \; \forall t = 1, \ldots, p$$

Minimum Frobenius Norm Interpolation

For MFN, given interpolation points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p$ (with $n+1 \leq p \leq (n+1)(n+2)/2$), derive a quadratic model via

$$\min_{c_k, \boldsymbol{g}_k, H_k} \|H_k\|_F^2 \qquad \text{s.t.} \quad H_k = H_k^T \text{ and } m_k(\boldsymbol{y}_t) = f(\boldsymbol{y}_t) \, \forall t = 1, \ldots, p$$

In practice, minimum change Hessian, $\min \|H_k - H_{k-1}\|_F^2$ also works well, but no theory.

Minimum Frobenius Norm Interpolation

For MFN, given interpolation points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p$ (with $n+1 \leq p \leq (n+1)(n+2)/2$), derive a quadratic model via

$$\min_{c_k, \boldsymbol{g}_k, H_k} \|H_k\|_F^2 \qquad \text{s.t.} \quad H_k = H_k^T \text{ and } m_k(\boldsymbol{y}_t) = f(\boldsymbol{y}_t) \, \forall t = 1, \ldots, p$$

In practice, minimum change Hessian, $\min \|H_k - H_{k-1}\|_F^2$ also works well, but no theory. Idea motivated by quasi-Newton updating (min change in Hessian s.t. secant equation).

Minimum Frobenius Norm Interpolation

For MFN, given interpolation points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p$ (with $n+1 \leq p \leq (n+1)(n+2)/2$), derive a quadratic model via

$$\min_{c_k, \boldsymbol{g}_k, H_k} \|H_k\|_F^2 \qquad \text{s.t.} \quad H_k = H_k^T \text{ and } m_k(\boldsymbol{y}_t) = f(\boldsymbol{y}_t) \, \forall t = 1, \ldots, p$$

In practice, minimum change Hessian, $\min \|H_k - H_{k-1}\|_F^2$ also works well, but no theory. Idea motivated by quasi-Newton updating (min change in Hessian s.t. secant equation).

Thie problem is a convex, equality-constrained QP, so can be solved via a linear system (size $p + n + 1$, symmetric saddle point system).

Minimum Frobenius Norm Interpolation

For MFN, given interpolation points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p$ (with $n+1 \leq p \leq (n+1)(n+2)/2$), derive a quadratic model via

$$\min_{c_k, \boldsymbol{g}_k, H_k} \|H_k\|_F^2 \qquad \text{s.t.} \quad H_k = H_k^T \text{ and } m_k(\boldsymbol{y}_t) = f(\boldsymbol{y}_t) \, \forall t = 1, \ldots, p$$

In practice, minimum change Hessian, $\min \|H_k - H_{k-1}\|_F^2$ also works well, but no theory. Idea motivated by quasi-Newton updating (min change in Hessian s.t. secant equation).

Thie problem is a convex, equality-constrained QP, so can be solved via a linear system (size $p + n + 1$, symmetric saddle point system).

If $p = n + 1$ get linear models, if $p = (n+1)(n+2)/2$ get (unique) fully quadratic interpolant.

Minimum Frobenius Norm Interpolation

From saddle point system theory, we get

Lemma [Benzi et al., 2005]

If the MFN interpolation matrix $\hat{F}$ (scaled via $(\boldsymbol{y}_t - \boldsymbol{x}_k)/\Delta_k$) is invertible, then the equivalent linear regression system $\hat{M} \in \mathbb{R}^{p \times (n+1)}$ is full column rank.

Minimum Frobenius Norm Interpolation

From saddle point system theory, we get

Lemma [Benzi et al., 2005]

If the MFN interpolation matrix $\hat{F}$ (scaled via $(\boldsymbol{y}_t - \boldsymbol{x}_k)/\Delta_k$) is invertible, then the equivalent linear regression system $\hat{M} \in \mathbb{R}^{p \times (n+1)}$ is full column rank.

The fully linear bound is:

Theorem [LR, 2025], originally [Conn et al., 2008]

If all $\|\boldsymbol{y}_t - \boldsymbol{x}_k\| \leq \beta\Delta_k$ and $\hat{F}$ is invertible, then the MFN model:

Minimum Frobenius Norm Interpolation

From saddle point system theory, we get

Lemma [Benzi et al., 2005]

If the MFN interpolation matrix $\hat{F}$ (scaled via $(\boldsymbol{y}_t - \boldsymbol{x}_k)/\Delta_k$) is invertible, then the equivalent linear regression system $\hat{M} \in \mathbb{R}^{p \times (n+1)}$ is full column rank.

The fully linear bound is:

Theorem [LR, 2025], originally [Conn et al., 2008]

If all $\|\boldsymbol{y}_t - \boldsymbol{x}_k\| \leq \beta\Delta_k$ and $\hat{F}$ is invertible, then the MFN model:

- Has bounded Hessian, $\|H_k\| \leq \kappa_H = \mathcal{O}(Lp\beta^4\|\hat{F}^{-1}\|_\infty)$.

Minimum Frobenius Norm Interpolation

From saddle point system theory, we get

Lemma [Benzi et al., 2005]

If the MFN interpolation matrix $\hat{F}$ (scaled via $(\boldsymbol{y}_t - \boldsymbol{x}_k)/\Delta_k$) is invertible, then the equivalent linear regression system $\hat{M} \in \mathbb{R}^{p \times (n+1)}$ is full column rank.

The fully linear bound is:

Theorem [LR, 2025], originally [Conn et al., 2008]

If all $\|\boldsymbol{y}_t - \boldsymbol{x}_k\| \leq \beta\Delta_k$ and $\hat{F}$ is invertible, then the MFN model:

- Has bounded Hessian, $\|H_k\| \leq \kappa_H = \mathcal{O}(Lp\beta^4\|\hat{F}^{-1}\|_\infty)$.
- Is fully linear with $\kappa_{\text{mf}}, \kappa_{\text{mg}} = \mathcal{O}((L + \kappa_H)\sqrt{n}\beta^2\|\hat{M}^\dagger\|_\infty)$.

Minimum Frobenius Norm Interpolation

As a simple example, if we do MFN with $p = 2n + 1$ (a popular choice) and coordinate perturbations, $\{\boldsymbol{x}_k, \boldsymbol{x}_k \pm \Delta_k \boldsymbol{e}_1, \ldots, \boldsymbol{x}_k \pm \Delta_k \boldsymbol{e}_n\}$, then we get central difference gradients and diagonal H_k (3-point stencil for second derivatives).

Minimum Frobenius Norm Interpolation

As a simple example, if we do MFN with $p = 2n + 1$ (a popular choice) and coordinate perturbations, $\{\boldsymbol{x}_k, \boldsymbol{x}_k \pm \Delta_k \boldsymbol{e}_1, \ldots, \boldsymbol{x}_k \pm \Delta_k \boldsymbol{e}_n\}$, then we get central difference gradients and diagonal H_k (3-point stencil for second derivatives).

The above result gives $\kappa_H = \mathcal{O}(n^2)$ and $\kappa_{\text{mf}}, \kappa_{\text{mg}} = \mathcal{O}(\sqrt{n}\,\kappa_H)$, but a more refined analysis gives $\kappa_H = \mathcal{O}(1)$.

Minimum Frobenius Norm Interpolation

As a simple example, if we do MFN with $p = 2n + 1$ (a popular choice) and coordinate perturbations, $\{\boldsymbol{x}_k, \boldsymbol{x}_k \pm \Delta_k \boldsymbol{e}_1, \ldots, \boldsymbol{x}_k \pm \Delta_k \boldsymbol{e}_n\}$, then we get central difference gradients and diagonal H_k (3-point stencil for second derivatives).

The above result gives $\kappa_H = \mathcal{O}(n^2)$ and $\kappa_{\text{mf}}, \kappa_{\text{mg}} = \mathcal{O}(\sqrt{n}\,\kappa_H)$, but a more refined analysis gives $\kappa_H = \mathcal{O}(1)$.

Corollary

Using this interpolation set, we get $\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ after at most $\mathcal{O}(n\epsilon^{-2})$ iterations or $\mathcal{O}(n^2\epsilon^{-2})$ evaluations.

Minimum Frobenius Norm Interpolation

As a simple example, if we do MFN with $p = 2n + 1$ (a popular choice) and coordinate perturbations, $\{\boldsymbol{x}_k, \boldsymbol{x}_k \pm \Delta_k \boldsymbol{e}_1, \ldots, \boldsymbol{x}_k \pm \Delta_k \boldsymbol{e}_n\}$, then we get central difference gradients and diagonal H_k (3-point stencil for second derivatives).

The above result gives $\kappa_H = \mathcal{O}(n^2)$ and $\kappa_{\text{mf}}, \kappa_{\text{mg}} = \mathcal{O}(\sqrt{n}\,\kappa_H)$, but a more refined analysis gives $\kappa_H = \mathcal{O}(1)$.

Corollary

Using this interpolation set, we get $\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ after at most $\mathcal{O}(n\epsilon^{-2})$ iterations or $\mathcal{O}(n^2\epsilon^{-2})$ evaluations.

Similar finite differencing-type choices with perturbations scaled by n can again recover $\kappa_{\text{mf}}, \kappa_{\text{mg}} = \mathcal{O}(1)$, i.e. $\mathcal{O}(\epsilon^{-2})$ iterations and $\mathcal{O}(n\epsilon^{-2})$ evaluations.

Next steps

This approach to model construction is simple and gives nice theory, but requires resampling all interpolation points at every iteration (wasteful when evaluations are expensive).

Next steps

This approach to model construction is simple and gives nice theory, but requires resampling all interpolation points at every iteration (wasteful when evaluations are expensive).

Tomorrow, we look at more efficient interpolation set management — reusing many/most interpolation points from the previous iteration.

Next steps

This approach to model construction is simple and gives nice theory, but requires resampling all interpolation points at every iteration (wasteful when evaluations are expensive).

Tomorrow, we look at more efficient interpolation set management — reusing many/most interpolation points from the previous iteration.

We will also look at extensions of this algorithmic framework (e.g. constrained optimization).

References i

M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numerica, 14 (2005), pp. 1–137.

C. Cartis, N. I. M. Gould, and P. L. Toint, *Evaluation Complexity of Algorithms for Nonconvex Optimization: Theory, Computation and Perspectives*, no. 30 in MOS-SIAM Series on Optimization, MOS/SIAM, Philadelphia, 2022.

C. Cartis and L. Roberts, *A derivative-free Gauss–Newton method*, Mathematical Programming Computation, 11 (2019), pp. 631–674.

A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust-Region Methods*, vol. 1 of MPS-SIAM Series on Optimization, MPS/SIAM, Philadelphia, 2000.

A. R. Conn, K. Scheinberg, and L. N. Vicente, *Geometry of sample sets in derivative-free optimization: Polynomial regression and underdetermined interpolation*, IMA Journal of Numerical Analysis, 28 (2008), pp. 721–748.

A. R. Conn, K. Scheinberg, and L. N. Vicente, *Global convergence of general derivative-free trust-region algorithms to first- and second-order critical points*, SIAM Journal on Optimization, 20 (2009), pp. 387–415.

References ii

A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*, vol. 8 of MPS-SIAM Series on Optimization, MPS/SIAM, Philadelphia, 2009.

M. J. D. Powell, *Least Frobenius norm updating of quadratic models that satisfy interpolation conditions*, Mathematical Programming, 100 (2004), pp. 183–215.

L. Roberts, *Introduction to model-based derivative-free optimization*.
arXiv preprint arXiv:2510.04473, 2025.