Handling Noise in Model-Based Derivative-Free Optimization

Joint work with Nicole Felice & Sara Shashaani (North Carolina State University)

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

Sampling-Based Optimization in Robotics (RSS 2026 Workshop)
17 July 2026

Key references

This talk is based on the following papers:

- LR, Introduction to Model-Based Derivative-Free Optimization, *arXiv:2510.04473* (2025).
- Nicole Felice, LR & Sara Shashaani, Adaptive Sampling Trust-Region Optimization for Derivative-Free Stochastic Functions and Deterministic Equality Constraints, *Proc. Winter Simulation Conference* (2026).

Derivative-Free Optimization

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \qquad \Longrightarrow \text{ (e.g.)} \quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

Derivative-Free Optimization

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \qquad \Longrightarrow \text{ (e.g.)} \quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

- How to calculate derivatives of f?
  - Write code by hand
  - Finite differences, $f'(x) \approx \frac{f(x+h)-f(x)}{h}$
  - Algorithmic differentiation/backpropagation

Derivative-Free Optimization

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \qquad \Longrightarrow \text{ (e.g.)} \quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

- How to calculate derivatives of f?
  - Write code by hand
  - Finite differences, $f'(x) \approx \frac{f(x+h)-f(x)}{h}$
  - Algorithmic differentiation/backpropagation
- Impractical if function evaluation is black-box, computationally expensive or noisy

Derivative-Free Optimization

$$\min_{x \in \mathbb{R}^n} f(x) \qquad \Longrightarrow \text{ (e.g.)} \quad x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

- How to calculate derivatives of f?
  - Write code by hand
  - Finite differences, $f'(x) \approx \frac{f(x+h)-f(x)}{h}$
  - Algorithmic differentiation/backpropagation
- Impractical if function evaluation is black-box, computationally expensive or noisy
- How to pick stepsize/learning rate?
  - Calculate Lipschitz constant L of ∇f
  - Hyperparameter tuning
  - Adaptive procedures (e.g. linesearch)

Derivative-Free Optimization

$$\min_{x \in \mathbb{R}^n} f(x) \qquad \implies \text{(e.g.)} \quad x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

- How to calculate derivatives of f?
  - Write code by hand
  - Finite differences, $f'(x) \approx \frac{f(x+h)-f(x)}{h}$
  - Algorithmic differentiation/backpropagation
- Impractical if function evaluation is black-box, computationally expensive or noisy
- How to pick stepsize/learning rate?
  - Calculate Lipschitz constant L of ∇f
  - Hyperparameter tuning
  - Adaptive procedures (e.g. linesearch)
- Prefer adaptive procedures (no tuning, fits to local curvature, L often not known)

Derivative-Free Optimization

For black-box, computationally expensive and/or noisy functions, we cannot assume access to gradient information.

Alternative: derivative-free optimization (DFO)

Derivative-Free Optimization

For black-box, computationally expensive and/or noisy functions, we cannot assume access to gradient information.

Alternative: derivative-free optimization (DFO)

- Assume can evaluate $f(\boldsymbol{x})$ but not $\nabla f(\boldsymbol{x})$, etc. (but still assume f is differentiable)

Derivative-Free Optimization

For black-box, computationally expensive and/or noisy functions, we cannot assume access to gradient information.

Alternative: derivative-free optimization (DFO)

- Assume can evaluate $f(\boldsymbol{x})$ but not $\nabla f(\boldsymbol{x})$, etc. (but still assume f is differentiable)
  - Harder again are *comparison oracles*: given $\boldsymbol{x}$ and $\boldsymbol{y}$, say which is better (but no values of f), e.g. survey results [Cai et al., 2022; Scheinberg & Xiong, 2026]

Derivative-Free Optimization

For black-box, computationally expensive and/or noisy functions, we cannot assume access to gradient information.

Alternative: derivative-free optimization (DFO)

- Assume can evaluate $f(\boldsymbol{x})$ but not $\nabla f(\boldsymbol{x})$, etc. (but still assume f is differentiable)
  - Harder again are *comparison oracles*: given $\boldsymbol{x}$ and $\boldsymbol{y}$, say which is better (but no values of f), e.g. survey results [Cai et al., 2022; Scheinberg & Xiong, 2026]
- Focus on local minimization methods (actually, approximate stationary point: $\|\nabla f(\boldsymbol{x})\| \leq \epsilon$)

Derivative-Free Optimization

For black-box, computationally expensive and/or noisy functions, we cannot assume access to gradient information.

Alternative: derivative-free optimization (DFO)

- Assume can evaluate $f(\boldsymbol{x})$ but not $\nabla f(\boldsymbol{x})$, etc. (but still assume f is differentiable)
  - Harder again are *comparison oracles*: given $\boldsymbol{x}$ and $\boldsymbol{y}$, say which is better (but no values of f), e.g. survey results [Cai et al., 2022; Scheinberg & Xiong, 2026]
- Focus on local minimization methods (actually, approximate stationary point: $\|\nabla f(\boldsymbol{x})\| \leq \epsilon$)
- Consider efficient & adaptive methods with no need for parameter tuning/knowledge of Lipschitz constants

Derivative-Free Optimization

For black-box, computationally expensive and/or noisy functions, we cannot assume access to gradient information.

Alternative: derivative-free optimization (DFO)

- Assume can evaluate $f(\boldsymbol{x})$ but not $\nabla f(\boldsymbol{x})$, etc. (but still assume f is differentiable)
  - Harder again are *comparison oracles*: given $\boldsymbol{x}$ and $\boldsymbol{y}$, say which is better (but no values of f), e.g. survey results [Cai et al., 2022; Scheinberg & Xiong, 2026]
- Focus on local minimization methods (actually, approximate stationary point: $\|\nabla f(\boldsymbol{x})\| \leq \epsilon$)
- Consider efficient & adaptive methods with no need for parameter tuning/knowledge of Lipschitz constants

Applications (outside robotics): climate physics, quantum optimization, engineering design, ML, ... [Alarie et al., 2015; Lin et al., 2026]

Outline

1. **Model-Based DFO**
2. Bounded Noise
3. Stochastic Noise with Constraints

Model-Based DFO

There are many local DFO methods (e.g. Nelder–Mead, direct search). One popular and successful framework is model-based DFO (MBDFO).

Model-Based DFO

There are many local DFO methods (e.g. Nelder–Mead, direct search). One popular and successful framework is model-based DFO (MBDFO).

The basic approach behind an MBDFO algorithm is:

- Use evaluations at points near $\boldsymbol{x}_k$ to build a local model for the objective
- Perform one iteration of a trust-region algorithm (using this model instead of a Taylor-type approximation) to get $\boldsymbol{x}_{k+1}$ [Conn et al., 2000]

Model-Based DFO

There are many local DFO methods (e.g. Nelder–Mead, direct search). One popular and successful framework is model-based DFO (MBDFO).

The basic approach behind an MBDFO algorithm is:

- Use evaluations at points near $\boldsymbol{x}_k$ to build a local model for the objective
- Perform one iteration of a trust-region algorithm (using this model instead of a Taylor-type approximation) to get $\boldsymbol{x}_{k+1}$ [Conn et al., 2000]

Theoretically, the core idea is:

- Select 'good points' to build the model
- Model error $= \mathcal{O}$(Taylor series error)
- Standard trust-region convergence theory still holds

Model-Based DFO

The basic MBDFO algorithm is:

- Build a Taylor series-like local quadratic model for the objective

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

Model-Based DFO

The basic MBDFO algorithm is:

- Build a Taylor series-like local quadratic model for the objective

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

- Get step by minimizing model in a neighborhood

$$\boldsymbol{s}_k = \arg\min_{\boldsymbol{s} \in \mathbb{R}^n} m_k(\boldsymbol{x}_k + \boldsymbol{s}) \qquad \text{subject to } \|\boldsymbol{s}\| \leq \Delta_k$$

$\implies$ *'trust region' subproblem – specialized algorithms exist*

Model-Based DFO

The basic MBDFO algorithm is:

- Build a Taylor series-like local quadratic model for the objective

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

- Get step by minimizing model in a neighborhood

$$\boldsymbol{s}_k = \underset{\boldsymbol{s} \in \mathbb{R}^n}{\arg\min}\, m_k(\boldsymbol{x}_k + \boldsymbol{s}) \qquad \text{subject to } \|\boldsymbol{s}\| \leq \Delta_k$$

$\implies$ *'trust region' subproblem – specialized algorithms exist*

- Accept/reject step and adjust Δ_k based on quality of new point $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$

$$\boldsymbol{x}_{k+1} = \begin{cases} \boldsymbol{x}_k + \boldsymbol{s}_k, & \text{if sufficient decrease,} \quad \longleftarrow \text{(maybe increase } \Delta_k\text{)} \\ \boldsymbol{x}_k, & \text{otherwise.} \quad \longleftarrow \text{(decrease } \Delta_k\text{)} \end{cases}$$

Model-Based Optimisation

- Build a Taylor series-like model

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

  and find $\boldsymbol{g}_k$ and H_k without using derivatives

- How?

Model-Based Optimisation

- Build a Taylor series-like model

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

  and find $\boldsymbol{g}_k$ and H_k without using derivatives

- How? Interpolate f over a set of points — find $\boldsymbol{g}_k$, H_k such that

$$m_k(\boldsymbol{y}) = f(\boldsymbol{y}), \qquad \forall \boldsymbol{y} \in \mathcal{Y}$$

Model-Based Optimisation

- Build a Taylor series-like model

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

  and find $\boldsymbol{g}_k$ and H_k without using derivatives

- How? Interpolate f over a set of points — find $\boldsymbol{g}_k$, H_k such that

$$m_k(\boldsymbol{y}) = f(\boldsymbol{y}), \qquad \forall \boldsymbol{y} \in \mathcal{Y}$$

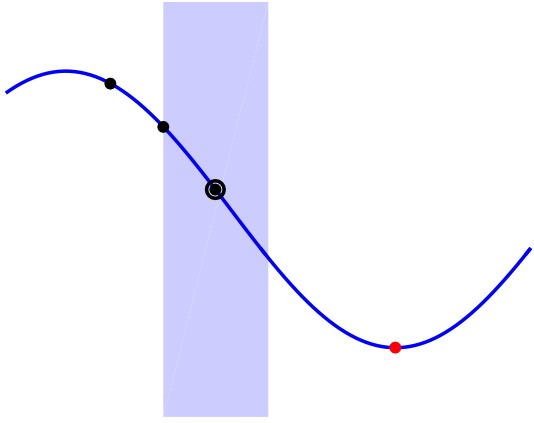
For convergence, need m_k to be fully linear:

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

Achievable if points in $\mathcal{Y}$ are well-spaced (in a specific sense).

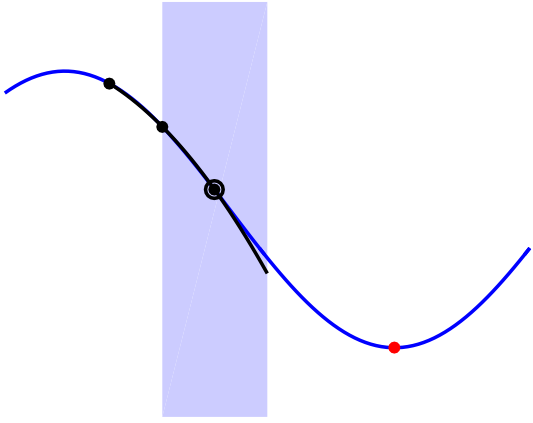
[Powell, 2004; Conn, Scheinberg & Vicente, 2009]

Example: Model-Based DFO



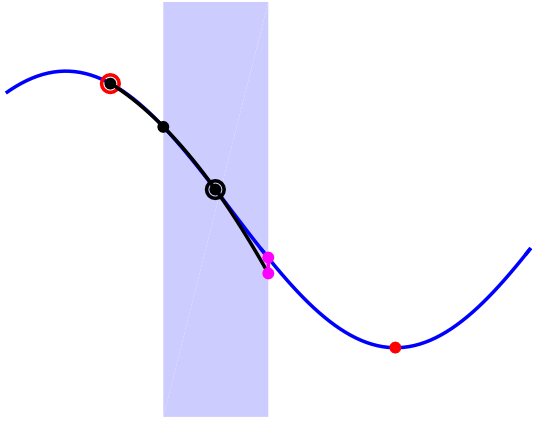
1. Choose interpolation set

Example: Model-Based DFO



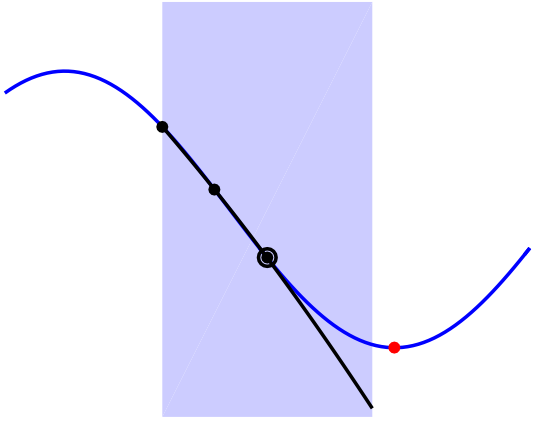
2. Interpolate & minimize...

Example: Model-Based DFO



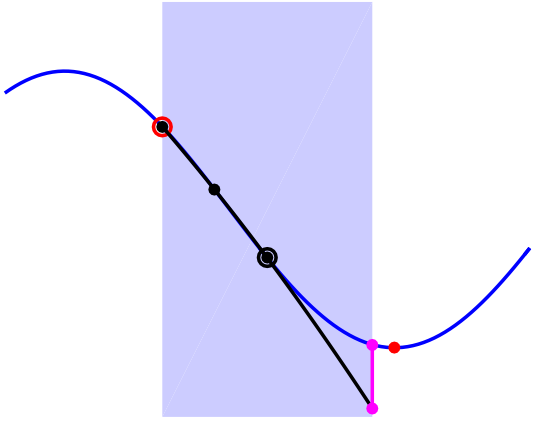
3. Add new point to interpolation set (replace a bad point)

Example: Model-Based DFO



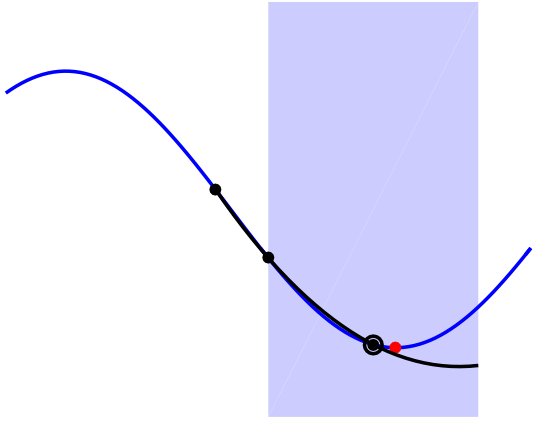
4. Repeat with new interpolation set & model

Example: Model-Based DFO



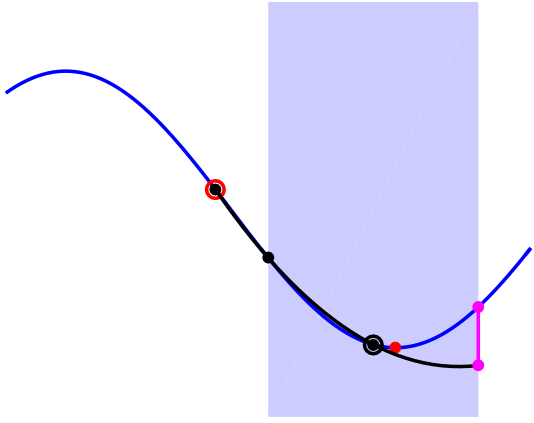
4. Repeat with new interpolation set & model

Example: Model-Based DFO



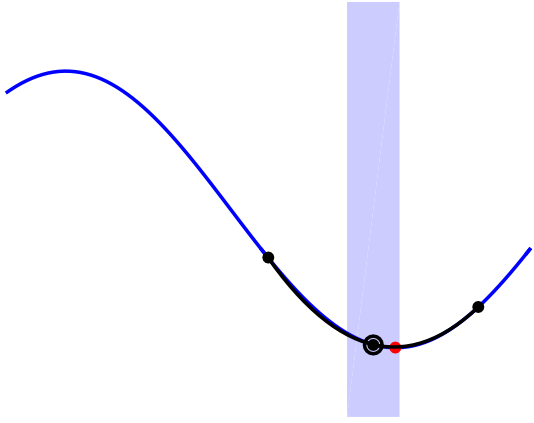
4. Repeat with new interpolation set & model

Example: Model-Based DFO



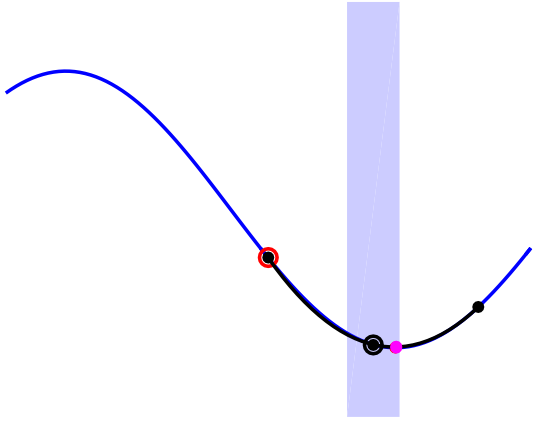
4. Repeat with new interpolation set & model

Example: Model-Based DFO



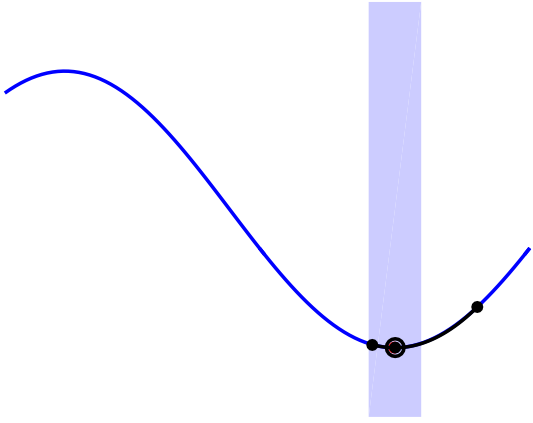
4. Repeat with new interpolation set & model

Example: Model-Based DFO



4. Repeat with new interpolation set & model

Example: Model-Based DFO



4. Repeat with new interpolation set & model

Practical MBDFO

The best-performing MBDFO software uses incremental geometry set updating:

Practical MBDFO

The best-performing MBDFO software uses incremental geometry set updating:

- Build local model m_k using current interpolation set

Practical MBDFO

The best-performing MBDFO software uses incremental geometry set updating:

- Build local model m_k using current interpolation set
- Calculate a step $\boldsymbol{s}_k$ and accept/reject

Practical MBDFO

The best-performing MBDFO software uses incremental geometry set updating:

- Build local model m_k using current interpolation set
- Calculate a step $\boldsymbol{s}_k$ and accept/reject
- Add $\boldsymbol{x}_k + \boldsymbol{s}_k$ to the interpolation set (replacing one 'bad' point)

Practical MBDFO

The best-performing MBDFO software uses incremental geometry set updating:

- Build local model m_k using current interpolation set
- Calculate a step $\boldsymbol{s}_k$ and accept/reject
- Add $\boldsymbol{x}_k + \boldsymbol{s}_k$ to the interpolation set (replacing one 'bad' point)
- If needed, change one more point the interpolation set, purely to improve the geometry

Practical MBDFO

The best-performing MBDFO software uses incremental geometry set updating:

- Build local model m_k using current interpolation set
- Calculate a step $\boldsymbol{s}_k$ and accept/reject
- Add $\boldsymbol{x}_k + \boldsymbol{s}_k$ to the interpolation set (replacing one 'bad' point)
- If needed, change one more point the interpolation set, purely to improve the geometry

Most popular in practice are Powell's algorithms (COBYLA, UOBYQA, NEWUOA, BOBYQA, LINCOA) — see Zaikun Zhang's PRIMA package on Github (or my Py-BOBYQA & DFO-LS codes).

Practical MBDFO

The best-performing MBDFO software uses incremental geometry set updating:

- Build local model m_k using current interpolation set
- Calculate a step $\boldsymbol{s}_k$ and accept/reject
- Add $\boldsymbol{x}_k + \boldsymbol{s}_k$ to the interpolation set (replacing one 'bad' point)
- If needed, change one more point the interpolation set, purely to improve the geometry

Most popular in practice are Powell's algorithms (COBYLA, UOBYQA, NEWUOA, BOBYQA, LINCOA) — see Zaikun Zhang's PRIMA package on Github (or my Py-BOBYQA & DFO-LS codes).

Typically measure performance by "# evaluations of objective required to achieve given objective improvement".

Recent Example

These methods were recently applied to robotics: [Landers et al., 2026]

- In this context, "maximum objective improvement given a fixed # evaluations" doesn't reflect reality

Recent Example

These methods were recently applied to robotics: [Landers et al., 2026]

- In this context, "maximum objective improvement given a fixed # evaluations" doesn't reflect reality
- Instead, measurement time for $f(\boldsymbol{x})$ can depend on *how many coordinates in* $\boldsymbol{x}$ *have been changed since the last measurement*

Recent Example

These methods were recently applied to robotics: [Landers et al., 2026]

- In this context, "maximum objective improvement given a fixed # evaluations" doesn't reflect reality
- Instead, measurement time for $f(\boldsymbol{x})$ can depend on *how many coordinates in* $\boldsymbol{x}$ *have been changed since the last measurement*
- Propose a modified trust-region step that biases (block) coordinate perturbations

Recent Example

These methods were recently applied to robotics: [Landers et al., 2026]

- In this context, "maximum objective improvement given a fixed # evaluations" doesn't reflect reality
- Instead, measurement time for $f(\boldsymbol{x})$ can depend on *how many coordinates in* $\boldsymbol{x}$ *have been changed since the last measurement*
- Propose a modified trust-region step that biases (block) coordinate perturbations

Improves over Powell's NEWUOA for optimizing angles of optical components (to maximize laser power).

Recent Example

These methods were recently applied to robotics: [Landers et al., 2026]

- In this context, "maximum objective improvement given a fixed # evaluations" doesn't reflect reality
- Instead, measurement time for $f(\boldsymbol{x})$ can depend on *how many coordinates in* $\boldsymbol{x}$ *have been changed since the last measurement*
- Propose a modified trust-region step that biases (block) coordinate perturbations

Improves over Powell's NEWUOA for optimizing angles of optical components (to maximize laser power).

Nelder–Mead and Bayesian Optimization tested, but much worse in practice.

Outline

1. Model-Based DFO
2. **Bounded Noise**
3. Stochastic Noise with Constraints

Bounded Noise

In reality, many situations do not have exact objective values available:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \qquad \text{but only have access to } \tilde{f}(\boldsymbol{x}) \in [f(\boldsymbol{x}) - \epsilon_f, f(\boldsymbol{x}) + \epsilon_f]$$

Bounded Noise

In reality, many situations do not have exact objective values available:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \qquad \text{but only have access to } \tilde{f}(\boldsymbol{x}) \in [f(\boldsymbol{x}) - \epsilon_f, f(\boldsymbol{x}) + \epsilon_f]$$

Assuming a constant, bounded noise level ϵ_f.

Bounded Noise

In reality, many situations do not have exact objective values available:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \qquad \text{but only have access to } \tilde{f}(\boldsymbol{x}) \in [f(\boldsymbol{x}) - \epsilon_f, f(\boldsymbol{x}) + \epsilon_f]$$

Assuming a constant, bounded noise level ϵ_f.

- Can be deterministic (e.g. floating-point errors) or stochastic noise (e.g. measurement error, Monte Carlo error)

Bounded Noise

In reality, many situations do not have exact objective values available:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \qquad \text{but only have access to } \tilde{f}(\boldsymbol{x}) \in [f(\boldsymbol{x}) - \epsilon_f, f(\boldsymbol{x}) + \epsilon_f]$$

Assuming a constant, bounded noise level ϵ_f.

- Can be deterministic (e.g. floating-point errors) or stochastic noise (e.g. measurement error, Monte Carlo error)
- Value of ϵ_f may not be known

Bounded Noise

In reality, many situations do not have exact objective values available:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \qquad \text{but only have access to } \tilde{f}(\boldsymbol{x}) \in [f(\boldsymbol{x}) - \epsilon_f, f(\boldsymbol{x}) + \epsilon_f]$$

Assuming a constant, bounded noise level ϵ_f.

- Can be deterministic (e.g. floating-point errors) or stochastic noise (e.g. measurement error, Monte Carlo error)
- Value of ϵ_f may not be known
  - Can be estimated, even for deterministic noise [Moré & Wild, 2011]

Bounded Noise

In reality, many situations do not have exact objective values available:

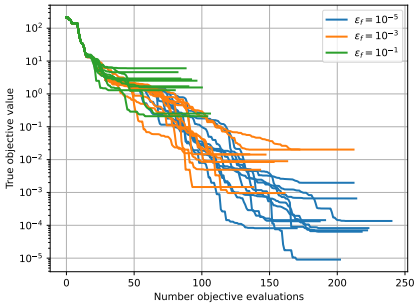
$$\min_{x \in \mathbb{R}^n} f(x) \qquad \text{but only have access to } \tilde{f}(x) \in [f(x) - \epsilon_f, f(x) + \epsilon_f]$$

Assuming a constant, bounded noise level ϵ_f.

- Can be deterministic (e.g. floating-point errors) or stochastic noise (e.g. measurement error, Monte Carlo error)
- Value of ϵ_f may not be known
  - Can be estimated, even for deterministic noise [Moré & Wild, 2011]
- Cannot expect convergence to a solution (e.g. $\|\nabla f(x_k)\| \to 0$), since can never be sure if $f(x_k + s_k) < f(x_k)$ when they are too close [Chaudhry & Scheinberg, 2025]

Example: Model-Based DFO

Fortunately, running an MBDFO algorithm with no modification still works well:



Minimizing Powell Singular function ($f^* = 0$) using Py-BOBYQA with stochastic noise $U([-\epsilon_f, \epsilon_f])$

Theory

In theory, bounded noise gives model accuracy bounds (assuming 'good' interpolation set):

$$|f(\boldsymbol{y}) - m_k(\boldsymbol{y})| \leq \mathcal{O}(\Delta_k^2 + \epsilon_f) \quad \text{and} \quad \|\nabla f(\boldsymbol{x}_k) - \nabla m_k(\boldsymbol{x}_k)\| \leq \mathcal{O}\left(\Delta_k + \frac{\epsilon_f}{\Delta_k}\right)$$

Theory

In theory, bounded noise gives model accuracy bounds (assuming 'good' interpolation set):

$$|f(\boldsymbol{y}) - m_k(\boldsymbol{y})| \leq \mathcal{O}(\Delta_k^2 + \epsilon_f) \quad \text{and} \quad \|\nabla f(\boldsymbol{x}_k) - \nabla m_k(\boldsymbol{x}_k)\| \leq \mathcal{O}\left(\Delta_k + \frac{\epsilon_f}{\Delta_k}\right)$$

This gives convergence to a neighborhood of a solution (with associated rates).

Theorem (LR, 2025)

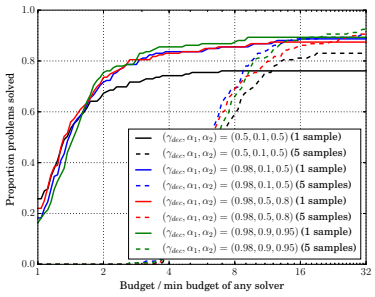
Under standard assumptions on f (e.g. Lipschitz continuous gradients), a standard MBDFO algorithm achieves $\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ after at most $\mathcal{O}(\epsilon^{-2})$ iterations/objective evaluations, provided $\epsilon > \epsilon_{\min} = \mathcal{O}(\sqrt{\epsilon_f})$.

Parameter Adjustment

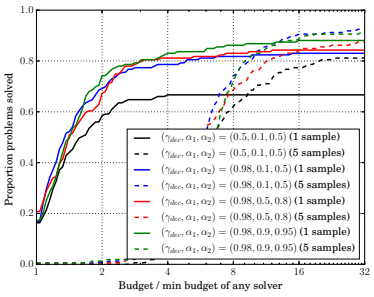
Useful heuristic: for noisy problems, decrease Δ_k more slowly on unsuccessful iterations ($\Delta_{k+1} = \gamma_{dec}\Delta_k$ for $\gamma_{dec} < 1$). [Cartis et al., 2019]

Parameter Adjustment

Useful heuristic: for noisy problems, decrease Δ_k more slowly on unsuccessful iterations ($\Delta_{k+1} = \gamma_{dec}\Delta_k$ for $\gamma_{dec} < 1$). [Cartis et al., 2019]



Uniform noise $\epsilon_f = 10^{-2}$



Gaussian noise $\sigma = 10^{-2}$

(Unpublished results, June 2017) — $\alpha_1 \leq \alpha_2 < 1$ are extra reduction parameters

Theory

Observation

The same convergence theory justifies this heuristic!

Theory

Observation

The same convergence theory justifies this heuristic!

Looking carefully at the constants in $\mathcal{O}(\cdot)$, we have: as $\gamma_{dec} \to 1^{-}$

Theory

Observation

The same convergence theory justifies this heuristic!

Looking carefully at the constants in $\mathcal{O}(\cdot)$, we have: as $\gamma_{dec} \to 1^-$

- ϵ_{min} decreases (still $\mathcal{O}(\sqrt{\epsilon_f})$) — higher accuracy achievable

Theory

Observation

The same convergence theory justifies this heuristic!

Looking carefully at the constants in $\mathcal{O}(\cdot)$, we have: as $\gamma_{dec} \to 1^-$

- ϵ_{min} decreases (still $\mathcal{O}(\sqrt{\epsilon_f})$) — higher accuracy achievable
- Upper bound on iterations/evaluations decreases (still $\mathcal{O}(\epsilon^{-2})$) — faster convergence

Observation

The same convergence theory justifies this heuristic!

Looking carefully at the constants in $\mathcal{O}(\cdot)$, we have: as $\gamma_{dec} \to 1^-$

- ϵ_{min} decreases (still $\mathcal{O}(\sqrt{\epsilon_f})$) — higher accuracy achievable
- Upper bound on iterations/evaluations decreases (still $\mathcal{O}(\epsilon^{-2})$) — faster convergence

For example, $\epsilon_{\text{min}} = \mathcal{O}\left(\frac{\sqrt{\epsilon_f}}{\sqrt{1-\left(\frac{1-\gamma_{dec}}{1+\gamma_{dec}}\right)^2}} \right)$

Outline

1. Model-Based DFO
2. Bounded Noise
3. **Stochastic Noise with Constraints**

Constrained Stochastic Optimization

Now consider the more complex problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) = \mathbb{E}_\omega[f(\boldsymbol{x},\omega)] \qquad \text{s.t.} \quad \boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{0}$$

i.e. stochastic noise in the objective with deterministic nonlinear constraints.

Constrained Stochastic Optimization

Now consider the more complex problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) = \mathbb{E}_\omega[f(\boldsymbol{x},\omega)] \qquad \text{s.t.} \quad \boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{0}$$

i.e. stochastic noise in the objective with deterministic nonlinear constraints.

Without noise, Sequential Quadratic Programming (SQP) methods are a popular choice

e.g. [Nocedal & Wright, 2006]

Constrained Stochastic Optimization

Now consider the more complex problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) = \mathbb{E}_\omega[f(\boldsymbol{x},\omega)] \qquad \text{s.t.} \quad \boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{0}$$

i.e. stochastic noise in the objective with deterministic nonlinear constraints.

Without noise, Sequential Quadratic Programming (SQP) methods are a popular choice e.g. [Nocedal & Wright, 2006]

- At $\boldsymbol{x}_k$, build linear models for the constraints and quadratic model for the objective/Lagrangian

Constrained Stochastic Optimization

Now consider the more complex problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) = \mathbb{E}_\omega[f(\boldsymbol{x},\omega)] \qquad \text{s.t.} \quad \boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{0}$$

i.e. stochastic noise in the objective with deterministic nonlinear constraints.

Without noise, Sequential Quadratic Programming (SQP) methods are a popular choice e.g. [Nocedal & Wright, 2006]

- At $\boldsymbol{x}_k$, build linear models for the constraints and quadratic model for the objective/Lagrangian
- Minimize the objective/Lagrangian model subject to *linearized constraints* (and trust region constraint)

Constrained Stochastic Optimization

Now consider the more complex problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) = \mathbb{E}_\omega[f(\boldsymbol{x},\omega)] \qquad \text{s.t.} \quad \boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{0}$$

i.e. stochastic noise in the objective with deterministic nonlinear constraints.

Without noise, Sequential Quadratic Programming (SQP) methods are a popular choice e.g. [Nocedal & Wright, 2006]

- At $\boldsymbol{x}_k$, build linear models for the constraints and quadratic model for the objective/Lagrangian
- Minimize the objective/Lagrangian model subject to *linearized constraints* (and trust region constraint)
- Determine sufficient decrease relative to a merit function, e.g. $\Phi(\boldsymbol{x},\mu) := f(\boldsymbol{x}) + \mu\|\boldsymbol{c}(\boldsymbol{x})\|_1$

Stochastic Noise

Question

How to handle stochastic noise in the objective (affects interpolation model accuracy & merit function accuracy/sufficient decrease check)?

Stochastic Noise

Question

How to handle stochastic noise in the objective (affects interpolation model accuracy & merit function accuracy/sufficient decrease check)?

In general, need all approximate objective values to have error $\mathcal{O}(\Delta_k^2)$.

Stochastic Noise

Question

How to handle stochastic noise in the objective (affects interpolation model accuracy & merit function accuracy/sufficient decrease check)?

In general, need all approximate objective values to have error $\mathcal{O}(\Delta_k^2)$.

Option 1: sample averaging/subsampling, $N_k = \mathcal{O}(\Delta_k^{-4})$ samples in iteration k, gives $\mathbb{P}[\text{good model or accurate merit function}] \geq 1 - \delta$ [Fang et al., 2024 & 2026]

Stochastic Noise

Question

How to handle stochastic noise in the objective (affects interpolation model accuracy & merit function accuracy/sufficient decrease check)?

In general, need all approximate objective values to have error $\mathcal{O}(\Delta_k^2)$.

Option 1: sample averaging/subsampling, $N_k = \mathcal{O}(\Delta_k^{-4})$ samples in iteration k, gives $\mathbb{P}[\text{good model or accurate merit function}] \geq 1 - \delta$ [Fang et al., 2024 & 2026]

Option 2: adaptive sampling — continue adding samples until we are confident we have a good estimate [Shashaani et al., 2018]

Stochastic Noise

Question

How to handle stochastic noise in the objective (affects interpolation model accuracy & merit function accuracy/sufficient decrease check)?

In general, need all approximate objective values to have error $\mathcal{O}(\Delta_k^2)$.

Option 1: sample averaging/subsampling, $N_k = \mathcal{O}(\Delta_k^{-4})$ samples in iteration k, gives $\mathbb{P}[\text{good model or accurate merit function}] \geq 1 - \delta$ [Fang et al., 2024 & 2026]

Option 2: adaptive sampling — continue adding samples until we are confident we have a good estimate [Shashaani et al., 2018]

Second option forms part of ASTRO software for simulation optimization, practically successful.

Theory

The adaptive sampling says: sample noisy objective $f(\boldsymbol{x}, \omega)$ independently N times and use sample average estimate

$$f(\boldsymbol{x}) \approx \overline{f}_N = \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}, \omega_i)$$

Theory

The adaptive sampling says: sample noisy objective $f(\boldsymbol{x}, \omega)$ independently N times and use sample average estimate

$$f(\boldsymbol{x}) \approx \overline{f}_N = \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}, \omega_i)$$

Keep adding new samples (i.e. increase N) until *we are confident our accuracy is* $\mathcal{O}(\Delta_k^2)$

$$\frac{\hat{\sigma}_N}{\sqrt{N}} \leq \mathcal{O}\left(\frac{\Delta_k^2}{\sqrt{\alpha_k}}\right) \qquad \text{and} \qquad N \geq \alpha_k = \mathcal{O}(k^{1+\epsilon})$$

where $\hat{\sigma}_N/\sqrt{N}$ is the estimated standard error in $\overline{f}_N$.

Theory

The adaptive sampling says: sample noisy objective $f(\boldsymbol{x}, \omega)$ independently N times and use sample average estimate

$$f(\boldsymbol{x}) \approx \overline{f}_N = \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}, \omega_i)$$

Keep adding new samples (i.e. increase N) until *we are confident our accuracy is* $\mathcal{O}(\Delta_k^2)$

$$\frac{\hat{\sigma}_N}{\sqrt{N}} \leq \mathcal{O}\left(\frac{\Delta_k^2}{\sqrt{\alpha_k}}\right) \qquad \text{and} \qquad N \geq \alpha_k = \mathcal{O}(k^{1+\epsilon})$$

where $\hat{\sigma}_N/\sqrt{N}$ is the estimated standard error in $\overline{f}_N$.

Theorem (Shashaani et al., 2018; Felice et al., 2026)

We have $\mathbb{P}[|\overline{f}_N - f(\boldsymbol{x})| > \mathcal{O}(\Delta_k^2) \ \ i.o.] = 0$

Theory

The adaptive sampling says: sample noisy objective $f(\boldsymbol{x}, \omega)$ independently N times and use sample average estimate

$$f(\boldsymbol{x}) \approx \overline{f}_N = \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}, \omega_i)$$

Keep adding new samples (i.e. increase N) until *we are confident our accuracy is* $\mathcal{O}(\Delta_k^2)$

$$\frac{\hat{\sigma}_N}{\sqrt{N}} \leq \mathcal{O}\left(\frac{\Delta_k^2}{\sqrt{\alpha_k}}\right) \qquad \text{and} \qquad N \geq \alpha_k = \mathcal{O}(k^{1+\epsilon})$$

where $\hat{\sigma}_N/\sqrt{N}$ is the estimated standard error in $\overline{f}_N$.

Theorem (Shashaani et al., 2018; Felice et al., 2026)

We have $\mathbb{P}[|\overline{f}_N - f(\boldsymbol{x})| > \mathcal{O}(\Delta_k^2) \ \textit{i.o.}] = 0$, *and so for the SQP algorithm we have* $\|\boldsymbol{c}(\boldsymbol{x}_k)\| \to 0$ *and* $\|\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)\| \to 0$ *almost surely.*

Conclusions & Future Work

Conclusions

- Model-based methods are a practical & theoretically justified DFO approach
- With bounded noise, get convergence to a neighborhood
- Theory justifies existing heuristic for noisy problems
- Adaptive sampling SQP method can handle stochastic objectives & deterministic nonlinear constraints

Future Work

- Theory for simple heuristic in unbounded noise case
- SQP extension to inequality constraints

Conclusions & Future Work

Conclusions

- Model-based methods are a practical & theoretically justified DFO approach
- With bounded noise, get convergence to a neighborhood
- Theory justifies existing heuristic for noisy problems
- Adaptive sampling SQP method can handle stochastic objectives & deterministic nonlinear constraints

Future Work

- Theory for simple heuristic in unbounded noise case
- SQP extension to inequality constraints

Intro to MBDFO: arXiv:2510.04473 (basic convergence, interpolation theory, constrained & noisy problems, software)

References i

S. Alarie, C. Audet, A. E. Gheribi, M. Kokkolaras, and S. Le Digabel, *Two decades of blackbox optimization applications*, EURO Journal on Computational Optimization, 9 (2021), p. 100011.

H. Cai, D. McKenzie, W. Yin, and Z. Zhang, *A one-bit, comparison-based gradient estimator*, Applied and Computational Harmonic Analysis, 60 (2022), pp. 242–266.

C. Cartis, J. Fiala, B. Marteau, and L. Roberts, *Improving the flexibility and robustness of model-based derivative-free optimization solvers*, ACM Transactions on Mathematical Software, 45 (2019), pp. 32:1–32:41.

A. Chaudhry and K. Scheinberg, *On complexity of model-based derivative-free methods*. **arXiv preprint arXiv:2510.14935, 2025.**

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, *Introduction to Derivative-Free Optimization*, vol. 8 of MPS-SIAM Series on Optimization, MPS/SIAM, Philadelphia, 2009.

References ii

Y. Fang, J. Lavaei, and S. Na, *High probability complexity bounds of trust-region stochastic sequential quadratic programming with heavy-tailed noise*, Mathematical Programming, (2026).

Y. Fang, S. Na, M. W. Mahoney, and M. Kolar, *Trust-region sequential quadratic programming for stochastic optimization with random models*.
arXiv preprint arXiv:2409.15734, 2024.

N. Felice, L. Roberts, and S. Shashaani, *Adaptive sampling trust region optimization for derivative-free stochastic functions and deterministic equality constraints*, in Proceedings of the Winter Simulation Conference Conference, 2026.

S. Landers, S. Pontula, S. Z. Uddin, S. Vaidya, M. Soljačić, and S. G. Johnson, *CLUSTER: Derivative-free optimization of smooth functions with parameter-change costs*.
arXiv preprint 2606.20498, 2026.

L. Lin, H. Ma, J. Wang, and S. Yang, *A survey on zeroth-order optimization for machine learning*, in Web Information Systems and Applications, S. Yang, J. Mao, M. Yu, C. Jin, and L. Chao, eds., vol. 16299, Springer Nature Singapore, Singapore, 2026, pp. 481–497.

References iii

J. J. Moré and S. M. Wild, *Estimating computational noise*, SIAM Journal on Scientific Computing, 33 (2011), pp. 1292–1314.

J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer, New York, 2nd ed., 2006.

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.

K. Scheinberg and Z. Xiong, *Function-free optimization via comparison oracles*.
arXiv preprint arXiv:2604.26867, 2026.

S. Shashaani, F. S. Hashemi, and R. Pasupathy, *ASTRO-DF: A class of adaptive sampling trust-region algorithms for derivative-free stochastic optimization*, SIAM Journal on Optimization, 28 (2018), pp. 3145–3176.