Derivative-free optimisation for least-squares problems

Joint work with Coralia Cartis (Oxford), Jan Fiala & Benjamin Marteau (NAG Ltd.), Simon Tett (Edinburgh), Matthias Ehrhardt (Bath)

Lindon Roberts, ANU (`lindon.roberts@anu.edu.au`)

Applied Mathematics Seminar, UNSW
16 April 2020

Supported by EPSRC (EP/L015803/1) & NAG Ltd.

Outline

1. Introduction to derivative-free optimisation (DFO)
2. DFO for nonlinear least-squares
3. Software implementation
4. Application: parameter tuning of climate models
5. Application: learning image denoising parameters

Outline

1. **Introduction to derivative-free optimisation (DFO)**
2. DFO for nonlinear least-squares
3. Software implementation
4. Application: parameter tuning of climate models
5. Application: learning image denoising parameters

Nonlinear Optimisation

Interested in nonlinear, nonconvex optimisation

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

where objective function $f : \mathbb{R}^n \to \mathbb{R}$.

- Ubiquitous in quantitative disciplines, but very difficult to solve in general
- No information about structure of f, except assumed smoothness
  - Problem constants (e.g. bounds on derivatives) unknown
  - Allow inaccurate evaluation of f, e.g. stochastic noise, iterative process
- Unconstrained, but software allows bounds $a_i \leq x_i \leq b_i$
- Seek local minimiser: $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x}$ close to $\boldsymbol{x}^*$ (not all $\boldsymbol{x} \in \mathbb{R}^n$)
  - Actually, seek (approximate) stationary point $\|\nabla f(\boldsymbol{x}^*)\|_2 \leq \epsilon$
- Gap between 'textbook' algorithms and state-of-the-art performance is large

Nonlinear Optimisation

Basic trust-region method:

- Approximate f near $\boldsymbol{x}_k$ with quadratic model

$$f(\boldsymbol{x}_k + \boldsymbol{s}) \approx m_k(\boldsymbol{s}) = f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^\top \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}$$

- Minimise model (set $\nabla m_k = 0$) gives Newton's method

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k \qquad \text{where} \qquad [\nabla^2 f(\boldsymbol{x}_k)]\boldsymbol{s}_k = -\nabla f(\boldsymbol{x}_k)$$

  but may not converge!

- One way to guarantee convergence: restrict the step size

$$\boldsymbol{s}_k = \underset{\boldsymbol{s}\in\mathbb{R}^n}{\arg\min}\, m_k(\boldsymbol{s}) \qquad \text{subject to } \|\boldsymbol{s}\|_2 \leq \Delta_k$$

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

Nonlinear Optimisation

Basic iterative method:

1. Given $\boldsymbol{x}_k$ and $\Delta_k > 0$, evaluate $f(\boldsymbol{x}_k)$, $\nabla f(\boldsymbol{x}_k)$, $\nabla^2 f(\boldsymbol{x}_k)$ and construct model m_k
2. Solve trust region subproblem to get step $\boldsymbol{s}_k$
3. Evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and determine quality of step

$$\rho_k := \frac{\text{actual decrease}}{\text{predicted decrease}} = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{s}_k)}{m_k(0) - m_k(\boldsymbol{s}_k)}$$

4. Accept/reject step and update Δ_k:
   - If $\rho_k \geq 0.7$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = 2\Delta_k$ *[very successful]*
   - If $\rho_k \in [0.1, 0.7)$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = \Delta_k$ *[successful]*
   - If $\rho_k < 0.1$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and $\Delta_{k+1} = \Delta_k/2$ *[unsuccessful]*

Standard algorithm with theoretical guarantees (e.g. $\lim_{k\to\infty} \|\nabla f(\boldsymbol{x}_k)\|_2 = 0$)

Derivative-Free Optimisation

$$f(\boldsymbol{x}_k + \boldsymbol{s}) \approx m_k(\boldsymbol{s}) = f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^\top \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}$$

- How to calculate derivatives of f in practice?

Derivative-Free Optimisation

$$f(\boldsymbol{x}_k + \boldsymbol{s}) \approx m_k(\boldsymbol{s}) = f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^\top \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}$$

- How to calculate derivatives of f in practice?
  - Write code by hand
  - Finite differences
  - Algorithmic differentiation

Derivative-Free Optimisation

$$f(\boldsymbol{x}_k + \boldsymbol{s}) \approx m_k(\boldsymbol{s}) = f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^\top \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}$$

- How to calculate derivatives of f in practice?
  - Write code by hand
  - Finite differences
  - Algorithmic differentiation
- Difficulties when function evaluation is
  - 'Black-box'
  - Noisy
  - Computationally expensive

Derivative-Free Optimisation

$$f(\boldsymbol{x}_k + \boldsymbol{s}) \approx m_k(\boldsymbol{s}) = f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^\top \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}$$

- How to calculate derivatives of f in practice?
  - Write code by hand
  - Finite differences
  - Algorithmic differentiation
- Difficulties when function evaluation is
  - 'Black-box'
  - Noisy
  - Computationally expensive
- Alternative — derivative-free optimisation (DFO)
- Many applications: finance, climate, engineering design, experimental design, ...

Model-Based DFO — Basic Ideas

Many different approaches: model-based, Nelder-Mead, pattern/direct search, genetic algorithms, ...

- Previously,

$$f(\boldsymbol{x}_k + \boldsymbol{s}) \approx m_k(\boldsymbol{s}) = f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^\top \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}$$

- Instead, approximate

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

- Find $\boldsymbol{g}_k$ and H_k <u>without</u> using derivatives

Model-Based DFO — Basic Ideas

Many different approaches: model-based, Nelder-Mead, pattern/direct search, genetic algorithms, ...

- Previously,

$$f(\boldsymbol{x}_k + \boldsymbol{s}) \approx m_k(\boldsymbol{s}) = f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^\top \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}$$

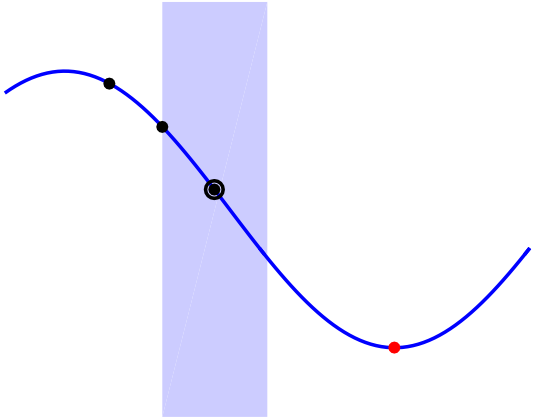
- Instead, approximate

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

- Find $\boldsymbol{g}_k$ and H_k <u>without</u> using derivatives
- How? Interpolate f over a set of points
- Same trust region approach as before

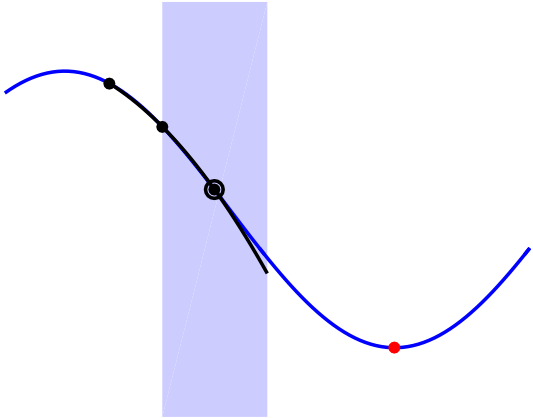
[Conn, Powell, Scheinberg, Toint, Vicente, ...]

Example: Model-Based DFO



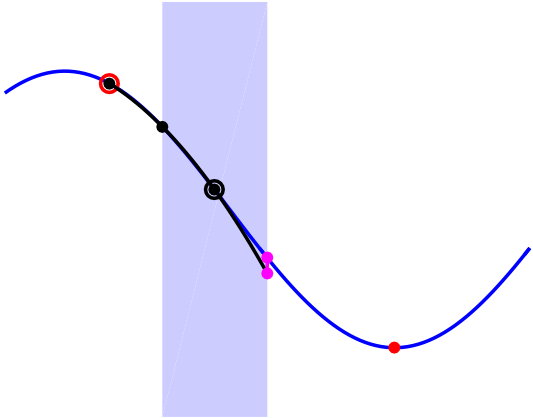
1. Choose interpolation set

Example: Model-Based DFO



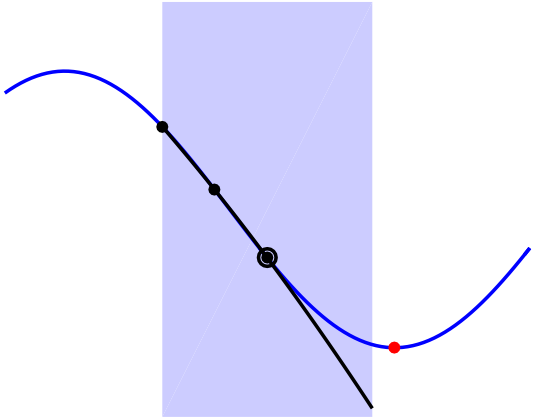
2. Interpolate & minimise...

Example: Model-Based DFO



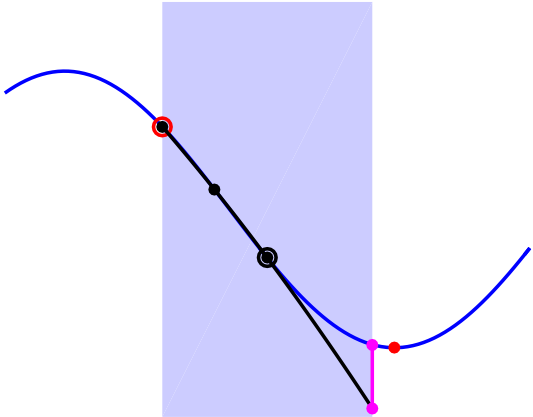
3. Add new point to interpolation set (replace a bad point)

Example: Model-Based DFO



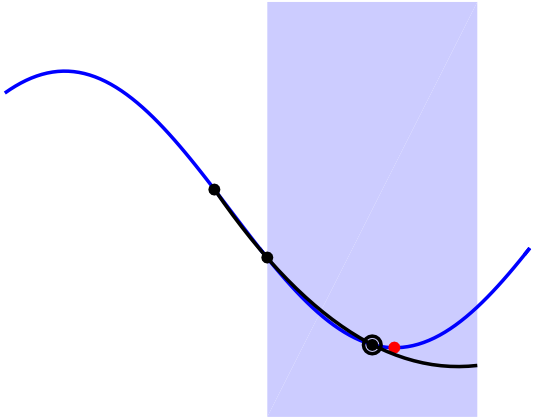
4. Repeat with new interpolation set & model

Example: Model-Based DFO



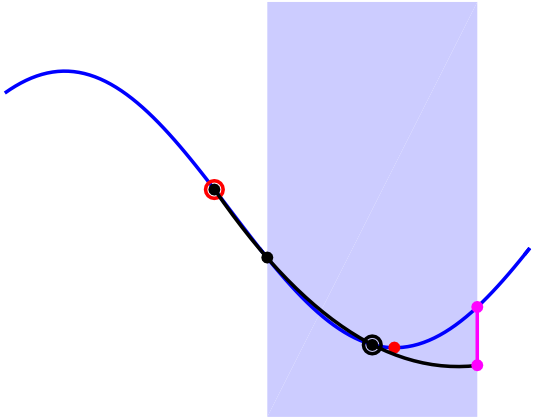
4. Repeat with new interpolation set & model

Example: Model-Based DFO



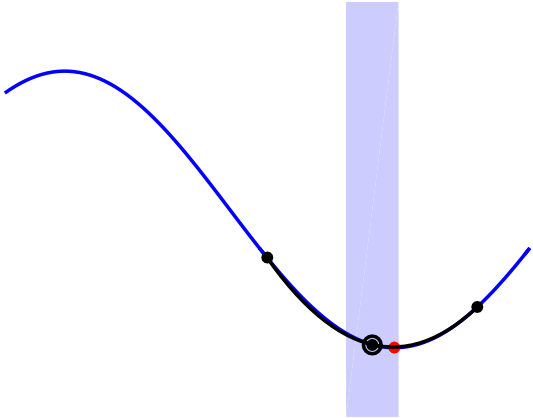
4. Repeat with new interpolation set & model

Example: Model-Based DFO



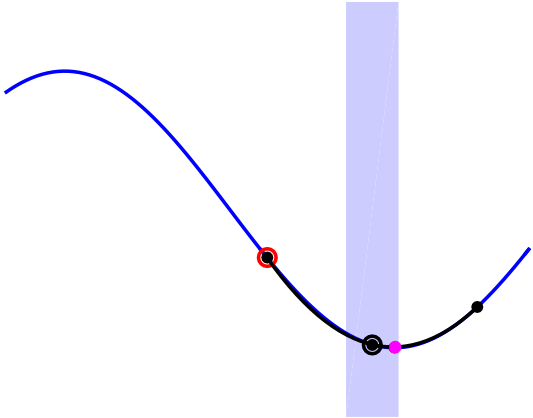
4. Repeat with new interpolation set & model

Example: Model-Based DFO



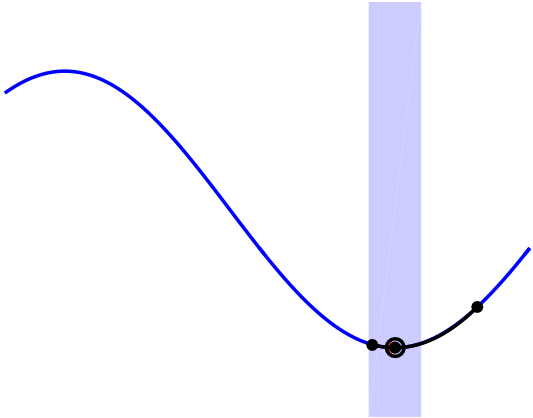
4. Repeat with new interpolation set & model

Example: Model-Based DFO



4. Repeat with new interpolation set & model

Example: Model-Based DFO



4. Repeat with new interpolation set & model

Model-Based DFO — Theory

What about theory?

- If geometry of interpolation points is good (in a specific sense), then model has same accuracy order as first-order Taylor series

$$|f(\boldsymbol{x}_k + \boldsymbol{s}) - m_k(\boldsymbol{s})| \leq \kappa_{\mathrm{f}} \Delta_k^2,$$
$$\|\nabla f(\boldsymbol{x}_k + \boldsymbol{s}) - \nabla m_k(\boldsymbol{s})\|_2 \leq \kappa_{\mathrm{g}} \Delta_k,$$

  for all $\|\boldsymbol{s}\|_2 \leq \Delta_k$.
- Need to modify algorithm to fix geometry, when needed
- Get similar convergence results as derivative-based methods
  - Global convergence to stationary points: $\lim_{k\to\infty} \|\nabla f(\boldsymbol{x}_k)\|_2 = 0$
  - Worst-case complexity: need at most $\mathcal{O}(\epsilon^{-2})$ iterations to get $\|\nabla f(\boldsymbol{x}_k)\|_2 \leq \epsilon$

Model-Based DFO — Theory

Geometry Requirement

- Interpolation set is $\{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_p\}$, usually with $\boldsymbol{y}_0 := \boldsymbol{x}_k$
- Build model by imposing $f(\boldsymbol{y}_t) = m_k(\boldsymbol{y}_t - \boldsymbol{x}_k)$ for all t
- Lagrange polynomials: $\ell_t(\boldsymbol{y}_s) = \delta_{s,t}$ for all s, t
- 'Good' geometry if all ℓ_t small $\Leftrightarrow$ interpolation problem well-conditioned

Theorem (Conn, Scheinberg & Vicente, 2008)

If $|\ell_t(\boldsymbol{x}_k + \boldsymbol{s})| \leq C_1$ for all $\|\boldsymbol{s}\|_2 \leq \Delta_k$ and all $\|\boldsymbol{y}_t - \boldsymbol{x}_k\|_2 \leq C_2\Delta_k$, then the interpolation model is "fully linear" (Taylor-accurate) inside the trust region.

Simple algorithms can check/ensure these conditions (maybe moving some points).

Model-Based DFO — Algorithm

Basic model-based DFO method:

1. Given $\boldsymbol{x}_k$ and $\Delta_k > 0$, construct interpolation model m_k
2. Solve trust region subproblem to get step $\boldsymbol{s}_k$
3. Evaluate $f(\boldsymbol{x}_k + \boldsymbol{s}_k)$ and determine quality of step

$$\rho_k := \frac{\text{actual decrease}}{\text{predicted decrease}} = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{s}_k)}{m_k(0) - m_k(\boldsymbol{s}_k)}$$

4. Accept/reject step and update Δ_k:
   - If $\rho_k \geq 0.7$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = 2\Delta_k$, add $\boldsymbol{x}_{k+1}$ to model *[very successful]*
   - If $\rho_k \in [0.1, 0.7)$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\Delta_{k+1} = \Delta_k$, add $\boldsymbol{x}_{k+1}$ to model *[successful]*
   - If $\rho_k < 0.1$ and model not fully linear, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, $\Delta_{k+1} = \Delta_k$ and make model fully linear *[model-improving]*
   - If $\rho_k < 0.1$ and model fully linear, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and $\Delta_{k+1} = \Delta_k/2$ *[unsuccessful]*

Outline

1. Introduction to derivative-free optimisation (DFO)
2. **DFO for nonlinear least-squares**
3. Software implementation
4. Application: parameter tuning of climate models
5. Application: learning image denoising parameters

DFO for Least-Squares — Basic Framework

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})\|_2^2, \qquad \boldsymbol{r}(\boldsymbol{x}) \in \mathbb{R}^m$$

DFO for Least-Squares — Basic Framework

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})\|_2^2, \qquad \boldsymbol{r}(\boldsymbol{x}) \in \mathbb{R}^m$$

Classical Gauss-Newton | **Derivative-Free Gauss-Newton**

- Linearise $\boldsymbol{r}$ at $\boldsymbol{x}_k$ using Jacobian

$$\boldsymbol{r}(\boldsymbol{x}_k+\boldsymbol{s}) \approx \boldsymbol{m}_k(\boldsymbol{s}) = \boldsymbol{r}(\boldsymbol{x}_k)+J(\boldsymbol{x}_k)\boldsymbol{s}$$

- Approximation: $\nabla^2 f(\boldsymbol{x}_k) = J(\boldsymbol{x}_k)^\top J(\boldsymbol{x}_k) + \cancel{\sum_{i=1}^m r_i(\boldsymbol{x}_k)\nabla^2 r_i(\boldsymbol{x}_k)}$

DFO for Least-Squares — Basic Framework

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})\|_2^2, \qquad \boldsymbol{r}(\boldsymbol{x}) \in \mathbb{R}^m$$

Classical Gauss-Newton

- Linearise $\boldsymbol{r}$ at $\boldsymbol{x}_k$ using Jacobian

$$\boldsymbol{r}(\boldsymbol{x}_k+\boldsymbol{s}) \approx \boldsymbol{m}_k(\boldsymbol{s}) = \boldsymbol{r}(\boldsymbol{x}_k)+J(\boldsymbol{x}_k)\boldsymbol{s}$$

- Approximation: $\nabla^2 f(\boldsymbol{x}_k) = J(\boldsymbol{x}_k)^\top J(\boldsymbol{x}_k) + \cancel{\sum_{i=1}^m r_i(\boldsymbol{x}_k)\nabla^2 r_i(\boldsymbol{x}_k)}$

Derivative-Free Gauss-Newton

- Jacobian not available, use

$$\boldsymbol{m}_k(\boldsymbol{s}) = \boldsymbol{r}(\boldsymbol{x}_k) + J_k\boldsymbol{s}$$

- Find J_k by interpolation

DFO for Least-Squares — Basic Framework

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})\|_2^2, \qquad \boldsymbol{r}(\boldsymbol{x}) \in \mathbb{R}^m$$

Classical Gauss-Newton

- Linearise $\boldsymbol{r}$ at $\boldsymbol{x}_k$ using Jacobian

$$\boldsymbol{r}(\boldsymbol{x}_k+\boldsymbol{s}) \approx \boldsymbol{m}_k(\boldsymbol{s}) = \boldsymbol{r}(\boldsymbol{x}_k)+J(\boldsymbol{x}_k)\boldsymbol{s}$$

- Approximation: $\nabla^2 f(\boldsymbol{x}_k) = J(\boldsymbol{x}_k)^\top J(\boldsymbol{x}_k) + \cancel{\sum_{i=1}^m r_i(\boldsymbol{x}_k)\nabla^2 r_i(\boldsymbol{x}_k)}$

Derivative-Free Gauss-Newton

- Jacobian not available, use

$$\boldsymbol{m}_k(\boldsymbol{s}) = \boldsymbol{r}(\boldsymbol{x}_k) + J_k\boldsymbol{s}$$

- Find J_k by interpolation

In both cases, solve trust region subproblem with simplified quadratic model

$$f(\boldsymbol{x}_k + \boldsymbol{s}) \approx m_k(\boldsymbol{s}) = \frac{1}{2}\|\boldsymbol{m}_k(\boldsymbol{s})\|_2^2$$

DFO Gauss-Newton

Previous works use quadratic models for $\boldsymbol{r}(\boldsymbol{x})$ [Zhang, Conn & Scheinberg (2010), Wild (2017)]

Advantages of linear models

- Match global convergence guarantees
- Fewer evaluations of $\boldsymbol{r}(\boldsymbol{x})$ to build first model
- Lower linear algebra cost ($\approx 7\times$ speedup) and improved scalability
- Explicit connection between geometry and linear algebra
  - When adding interpolation point $\boldsymbol{x}_{k+1}$, delete point $\boldsymbol{y}_t$ with large $|\ell_t(\boldsymbol{x}_{k+1})|$
  - Gives rank-1 update of interpolation matrix A, hence

$$A_{\text{new}}^{-1} = A_{\text{old}}^{-1} + \frac{1}{\sigma_t}\boldsymbol{u}_t\boldsymbol{v}_t^\top \qquad \text{where } \sigma_t = \ell_t(\boldsymbol{x}_{k+1})$$

Outline

1. Introduction to derivative-free optimisation (DFO)
2. DFO for nonlinear least-squares
3. **Software implementation**
4. Application: parameter tuning of climate models
5. Application: learning image denoising parameters

DFO-LS

DFO-LS (Derivative-Free Optimisation for Least-Squares)

Open-source Python package (NLLS with bounds)

– Github: `numerical algorithms group/dfols`

Key Features:

- Flexible model construction

  $\Longrightarrow$ *enables both reduced initialisation cost and regression models*

- Robust to noisy objectives using multiple restarts

  $\Longrightarrow$ *effective alternative to sample averaging, regression models*

- Reduced initialisation cost for expensive objectives

  $\Longrightarrow$ *progress from 2 evaluations, if desired*

Flexible Model Construction

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})\|_2^2, \qquad \boldsymbol{r}(\boldsymbol{x}) \in \mathbb{R}^m$$

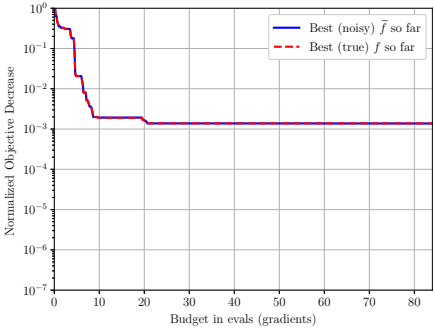
- Have $p+1$ interpolation points $\{\boldsymbol{y}_0 = \boldsymbol{x}_k, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_p\}$
- Find model $\boldsymbol{m}_k(\boldsymbol{s}) = \boldsymbol{r}_k + J_k\boldsymbol{s}$ by solving

$$\min_{\boldsymbol{r}_k, J_k} \sum_{t=0}^{p} \|\boldsymbol{m}_k(\boldsymbol{y}_t - \boldsymbol{x}_k) - \boldsymbol{r}(\boldsymbol{y}_t)\|_2^2$$

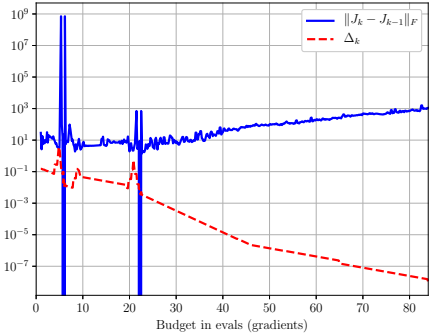
  $\Rightarrow$ *one* $(p+1)\times(n+1)$ *system, different RHS for each residual* r_i, $i = 1, \ldots, m$
- Works for any $p \geq 1$
  - Unique interpolant if $p = n$ (usual case)
  - Regression model if $p > n$ (e.g. noisy objective)
  - Select minimal-norm solution if $p < n$ (used for reduced initialisation cost)

Noisy Problems — Example of Stagnation

- TR radius $\Delta_k \to 0$, so interpolation points eventually get close together
- Objective values all within noise level $\implies$ interpolated model only captures noise
- This is one of the main use cases for DFO!



Normalised Objective Decrease



Convergence Details

Robustness to Noise

Common approaches:

- Sample averaging [Deng & Ferris (2006), Chen, Menickelly & Scheinberg (2016)]
- Regression models [Conn, Scheinberg & Vicente (2009), Billups, Larson & Graf (2013)]
- Both available in DFO-LS if desired (c.f. flexible model construction)

Robustness to Noise

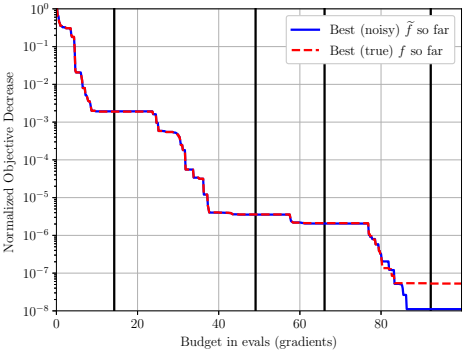
Common approaches:

- Sample averaging [Deng & Ferris (2006), Chen, Menickelly & Scheinberg (2016)]
- Regression models [Conn, Scheinberg & Vicente (2009), Billups, Larson & Graf (2013)]
- Both available in DFO-LS if desired (c.f. flexible model construction)

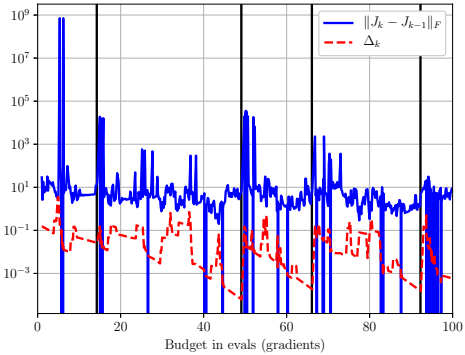
Alternative strategy: multiple restarts

- When $\Delta_k \leq \Delta_{min}$, reset $\Delta_{k+1} = \Delta_0$ initial TR radius
- Update interpolation set for new trust region:
  - Move $\boldsymbol{x}_k$ plus $N \approx 2$ points closest to $\boldsymbol{x}_k$ to geometry-improving points in $B(\boldsymbol{x}_k, \Delta_{k+1})$
- *Auto-detection:* call restart if Δ_k consistently decreasing and $\|J_k - J_{k-1}\|_F$ consistently increasing for several iterations

Multiple Restarts Strategy — Example



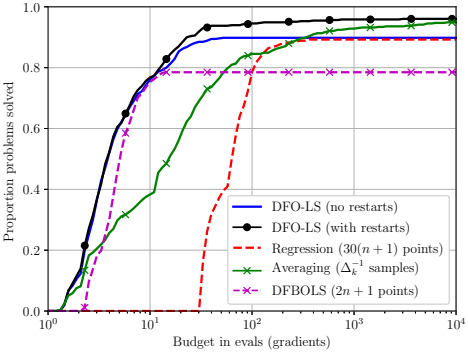
Normalised Objective Decrease



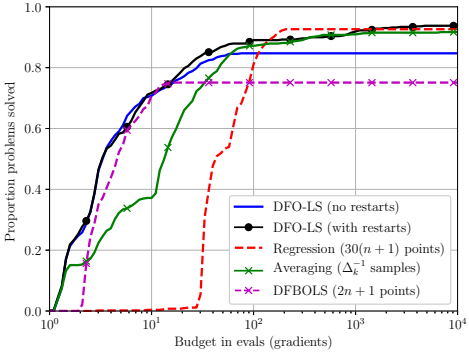
Convergence Details

DFO-LS — Comparison of Noise Robustness Strategies

Data profiles (using 53 test problems from [Moré & Wild, 2009], $\tau = \max(10^{-5}, \hat{\tau})$)



Mult. Gaussian noise ($\pm 1\%$)



Add. Gaussian noise (± 0.01)

% test problems solved with a given # objective evaluations; higher values are better

Reduced Initialisation Cost for Expensive Objectives

- Start by evaluating objective at $\boldsymbol{x}_0$ and p random orthogonal directions $\boldsymbol{y}_t = \boldsymbol{x}_0 + \Delta_0 \boldsymbol{q}_t$ $(t = 1, \ldots, p)$ — usually $p = n$
- **Expensive objective?** Want to see progress with very few evaluations

Reduced Initialisation Cost for Expensive Objectives

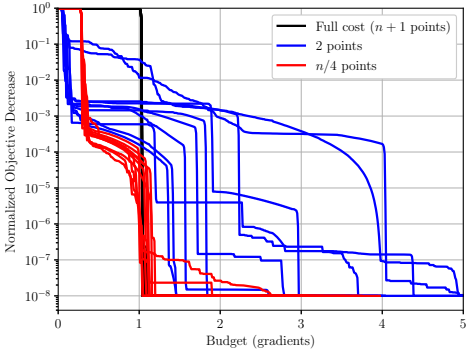
- Start by evaluating objective at $\boldsymbol{x}_0$ and p random orthogonal directions $\boldsymbol{y}_t = \boldsymbol{x}_0 + \Delta_0 \boldsymbol{q}_t$ $(t = 1, \ldots, p)$ — usually $p = n$
- **Expensive objective?** Want to see progress with very few evaluations
- For $p < n$ directions, use interpolating model with minimal norm
- Problem: J_k not full rank, so $\boldsymbol{s}_k \in \operatorname{span}\{\boldsymbol{y}_1 - \boldsymbol{x}_k, \ldots, \boldsymbol{y}_p - \boldsymbol{x}_k\}$
  $\implies$ can never search outside the initial subspace of directions

Reduced Initialisation Cost for Expensive Objectives

- Start by evaluating objective at $\boldsymbol{x}_0$ and p random orthogonal directions $\boldsymbol{y}_t = \boldsymbol{x}_0 + \Delta_0 \boldsymbol{q}_t$ $(t = 1, \ldots, p)$ — usually $p = n$
- **Expensive objective?** Want to see progress with very few evaluations
- For $p < n$ directions, use interpolating model with minimal norm
- Problem: J_k not full rank, so $\boldsymbol{s}_k \in \operatorname{span}\{\boldsymbol{y}_1 - \boldsymbol{x}_k, \ldots, \boldsymbol{y}_p - \boldsymbol{x}_k\}$
  $\implies$ can never search outside the initial subspace of directions
- Solution: Artificially perturb J_k to make it full rank
  - Floor singular values at $\sigma_p > 0$
- Sometimes this will give descent, but always expands the search space

Reduced Initialisation Cost — Example

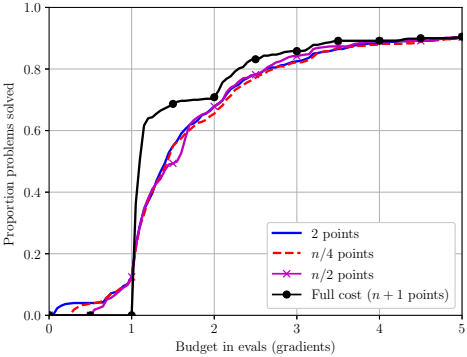
Can make reasonable progress with $< n + 1$ evaluations, but usually better to wait (if possible)



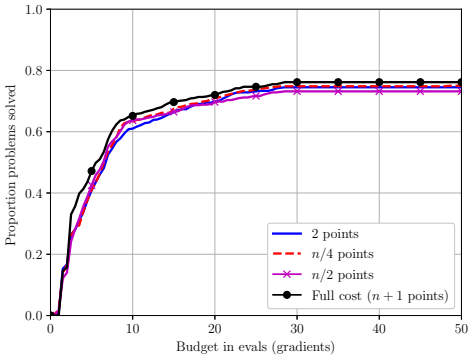
Objective decrease for BROWNALE ($n = 100$)

Reduced Initialisation Cost — Performance

Data profiles (60 test problems with $n \approx 100$)



Short budget, $\tau = 10^{-1}$



Long budget, $\tau = 10^{-5}$

% test problems solved with a given # objective evaluations; higher values are better

Outline

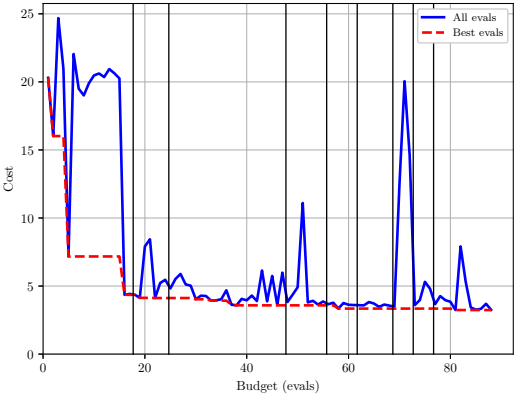
1. Introduction to derivative-free optimisation (DFO)
2. DFO for nonlinear least-squares
3. Software implementation
4. **Application: parameter tuning of climate models**
5. Application: learning image denoising parameters

Climate Parameter Tuning

- Aim: tuning models of global atmospheric physics
- Fit to observations (e.g. average temperature, humidity, radiation)
- Difficulty: simulations are expensive (multi-year global climate simulation) and noisy (underlying physics is chaotic)
- Standard approach for tuning climate models is manual:
  - Generate different sets of parameters
  - Evaluate fit to observations
  - Select parameters with best fit (and possibly perturb these to generate new sets)
- Alternative: apply DFO-LS with multiple restarts

Climate Parameter Tuning — Results

Example Results (HadAM3, 14 parameters, budget 90 evaluations)



Cost vs. # objective evaluations (cost ≤ 5 considered broadly in line with observations)

Climate Parameter Tuning — Results

- Start DFO-LS from 5 different starting locations
- Find 5 different parameter combinations, all in line with observations (and genuinely different local minima)
- Outperforms other solvers: approximate finite differencing, surrogate modelling

Climate research question

Q: How are these local minima different from a climatology perspective?

A: Not very! *Uncertainty in climate predictions largely driven by modelling choices (processes included & parametrisations) not parameter tuning.*

Outline

1. Introduction to derivative-free optimisation (DFO)
2. DFO for nonlinear least-squares
3. Software implementation
4. Application: parameter tuning of climate models
5. **Application: learning image denoising parameters**

Bilevel Learning

Many image processing problems can be posed in the form

$$\min_{\boldsymbol{x}} \mathcal{D}(A\boldsymbol{x}, \boldsymbol{y}) + \mathcal{R}(\boldsymbol{x}),$$

where $\mathcal{D}$ measures data fidelity ($A\boldsymbol{x} \approx \boldsymbol{y}$) and $\mathcal{R}$ is regulariser; e.g. denoising

$$\min_{\boldsymbol{x}} \underbrace{\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2}_{\mathcal{D}(\boldsymbol{x},\boldsymbol{y})} + \alpha \underbrace{\sum_j \sqrt{\|\nabla x_j\|_2^2 + \epsilon^2}}_{\approx \mathrm{TV}(\boldsymbol{x})} + \frac{\eta}{2}\|\boldsymbol{x}\|_2^2$$

This problem is smooth and strongly convex, can be solved effectively with iterative methods (gradient descent, NAG, FISTA, etc.).

Bilevel Learning

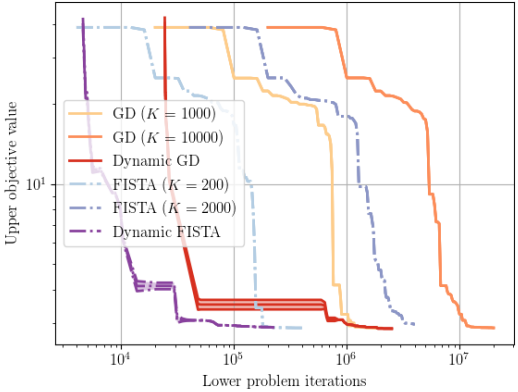
- Unclear how to choose parameters $\theta := [\alpha, \epsilon, \eta]^\top$
- One option: learn parameters from example problems $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$:

$$\begin{aligned} \min_{\theta} \quad & \sum_i \|\hat{\boldsymbol{x}}_i(\theta) - \boldsymbol{x}_i\|_2^2 \\ \text{s.t.} \quad & \hat{\boldsymbol{x}}_i \text{ solves denoising problem with } \theta \end{aligned}$$

- Bilevel optimisation problem, requires computing $\partial_\theta \hat{\boldsymbol{x}}_i(\theta)$
  - Requires very high accuracy solves of denoising problem
  - Don't know in advance what accuracy is required (educated guess)
- Alternative: modify DFO-LS to allow dynamic accuracy on objective evaluations (i.e. ask for $\hat{\boldsymbol{x}}_i(\theta)$ correct to within some error δ_x)

Bilevel Learning

Example results (gradient descent & FISTA as lower-level solvers):



More efficient learning, without requiring heuristics for lower-level accuracy

Conclusion & Future Work

Conclusions

- DFO methods suitable when objective is expensive and/or noisy
- DFO equivalent of Gauss-Newton gives an effective algorithm for least-squares
- Reduced initialisation cost if desired — can start progressing after 2 evaluations
- Effective for tuning global climate models and bilevel learning

Future work:

- Scalability: dimensionality reduction, sparsity, inexact interpolation solves, etc.
- Local convergence rates
- General objective and constrained problems

References i

Stephen C. Billups, Jeffrey W. Larson, and Peter Graf.
Derivative-free optimization of expensive functions with computational error using weighted regression.
SIAM Journal on Optimization, 23(1):27–53, 2013.

Coralia Cartis, Jan Fiala, Benjamin Marteau, and Lindon Roberts.
Improving the flexibility and robustness of model-based derivative-free optimization solvers.
ACM Transactions on Mathematical Software, 45(3):32:1–32:41, 2019.

References ii

Coralia Cartis and Lindon Roberts.
A derivative-free Gauss-Newton method.
Mathematical Programming Computation, 2019.

Ruobing Chen, Matt Menickelly, and Katya Scheinberg.
Stochastic optimization using a trust-region method and random models.
Mathematical Programming, 169(2):447–487, 2018.

Andrew R. Conn, Katya Scheinberg, and Luís N. Vicente.
Geometry of interpolation sets in derivative free optimization.
Mathematical Programming, 111(1-2):141–172, 2007.

References iii

Andrew R. Conn, Katya Scheinberg, and Luís N. Vicente.
Introduction to Derivative-Free Optimization, volume 8 of MPS-SIAM Series on Optimization.
MPS/SIAM, Philadelphia, 2009.

Matthias J. Ehrhardt and Lindon Roberts.
Inexact derivative free optimization for bi-level learning.
in preparation, 2020.

Simon F. B. Tett, Jonathan M. Gregory, Nicolas Freychet, Coralia Cartis, Michael J. Mineter, and Lindon Roberts.
Does model calibration reduce uncertainty in climate projections?
submitted, 2020.

References iv

Stefan M. Wild.
POUNDERS in TAO: Solving derivative-free nonlinear least-squares problems with POUNDERS.
In T. Terlaky, M. F. Anjos, and S. Ahmed, editors, *Advances and Trends in Optimization with Engineering Applications*, volume 24 of *MOS-SIAM Book Series on Optimization*, pages 529–539. MOS/SIAM, Philadelphia, 2017.

Hongchao Zhang, Andrew R. Conn, and Katya Scheinberg.
A derivative-free algorithm for least-squares minimization.
SIAM Journal on Optimization, 20(6):3555–3576, 2010.