Large-scale derivative-free optimization using random subspace methods

Joint work with Coralia Cartis (Oxford) & Clément Royer (Paris-Dauphine PSL)

Lindon Roberts, University of Sydney (`lindon.roberts@sydney.edu.au`)

OPTIMA Seminar
27 July 2022

Further Reading

This talk is based on:

- C. Cartis & L. Roberts, Scalable subspace methods for derivative-free nonlinear least-squares optimization, *Math. Prog.*, 2022.
- L. Roberts & C. W. Royer, Direct search based on probabilistic descent in reduced spaces, *arXiv:2204.01275*, 2022.

Our software packages are:

- DFBGN for nonlinear least-squares:
  `https://github.com/numericalalgorithmsgroup/dfbgn`
- directsearch for general problems:
  `https://github.com/lindonroberts/directsearch`

Outline

1. **Introduction to derivative-free optimization (DFO)**
2. Subspace DFO methods
3. Numerical results

Nonlinear Optimization

Interested in unconstrained nonlinear optimization

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

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

- f is possibly nonconvex and/or 'black-box'
  - In practice, allow inaccurate evaluations of f, e.g. noise, outcome of iterative process
- Seek local minimizer (actually, approximate stationary point: $\|\nabla f(\boldsymbol{x})\|_2 \le \epsilon$)

Lots of high-quality algorithms available:

- Linesearch, $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k H_k^{-1}\nabla f(\boldsymbol{x}_k)$ (e.g. GD, Newton, BFGS)
- Trust-region methods (adapt well to derivative-free setting)
- Others: cubic regularization, nonlinear CG, ...

Basic trust-region method

- Approximate f near $\boldsymbol{x}_k$ with a local quadratic (Taylor) model

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

- Get step by minimizing model in a neighborhood

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

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

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

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

[Conn, Gould & Toint, 2000]

Derivative-Free Optimization

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$$

$$m_k(s) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T \nabla^2 f(x_k) s$$

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

Derivative-Free Optimization

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

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

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

Derivative-Free Optimization

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\nabla^2 f(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k)$$

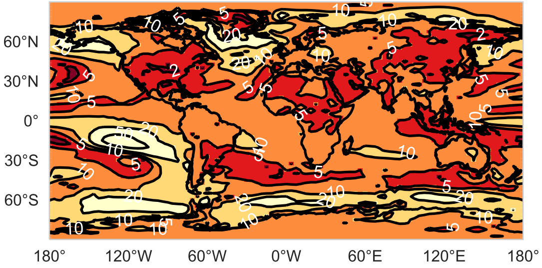
$$m_k(\mathbf{s}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \mathbf{s} + \frac{1}{2} \mathbf{s}^T \nabla^2 f(\mathbf{x}_k) \mathbf{s}$$

- How to calculate derivatives of f in practice?
  - Write code by hand
  - Finite differences
  - Algorithmic differentiation/backpropagation
- Difficulties when function evaluation is
  - Black-box
  - Noisy
  - Computationally expensive
- Alternative — derivative-free optimization (DFO)

Application 1: Climate Modelling

[Tett et al., 2022]

- Parameter calibration for global climate models
- One model run = simulate global climate for 5 years (expensive!)
- Very complicated, chaotic physics (black-box & noisy!)



Application 2: Adversarial Example Generation

[Alzantot et al., 2019]

- Find perturbations of neural network inputs which are misclassified
- Neural network structure assumed to be unknown (black-box!)
- Want to test very few examples ($\approx$ expensive!)

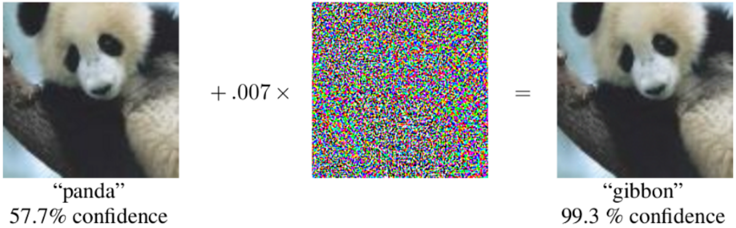


Image from [Goodfellow et al., 2015]

DFO Method 1: Model-Based DFO

- Using trust-region framework, build a model

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

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

DFO Method 1: Model-Based DFO

- Using trust-region framework, build a model

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

  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

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

DFO Method 1: Model-Based DFO

- Using trust-region framework, build a model

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

  and find $\boldsymbol{g}_k$ and H_k <u>without</u> using derivatives
- How? Interpolate f over a set of points — find $\boldsymbol{g}_k$, H_k such that

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

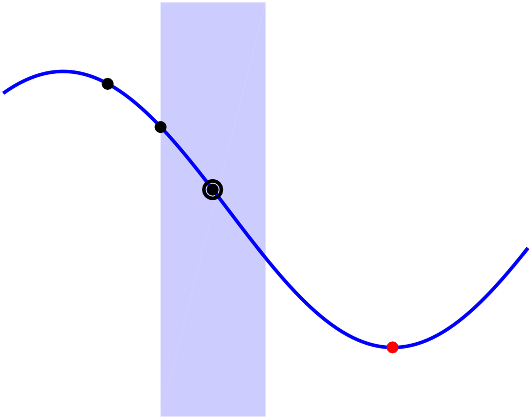
For convergence, need m_k to be fully linear:

$$|f(\boldsymbol{x}_k + \boldsymbol{s}) - m_k(\boldsymbol{s})| \leq \mathcal{O}(\Delta_k^2) \qquad \text{and} \qquad \|\nabla f(\boldsymbol{x}_k + \boldsymbol{s}) - \nabla m_k(\boldsymbol{s})\|_2 \leq \mathcal{O}(\Delta_k)$$

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

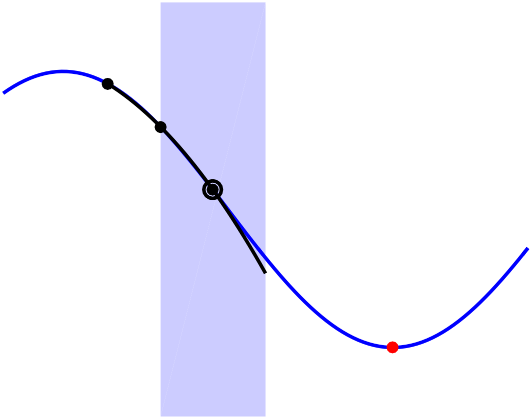
[Powell, 2003; Conn, Scheinberg & Vicente, 2009]

Example: Model-Based DFO



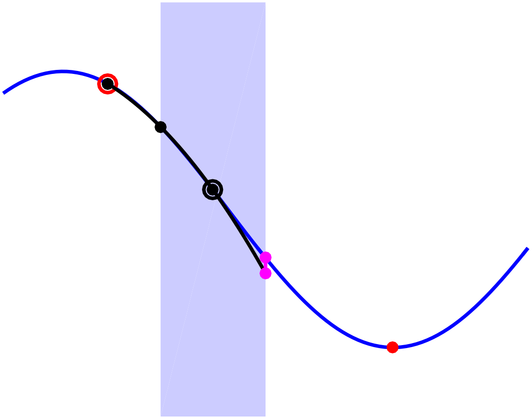
1. Choose interpolation set

Example: Model-Based DFO



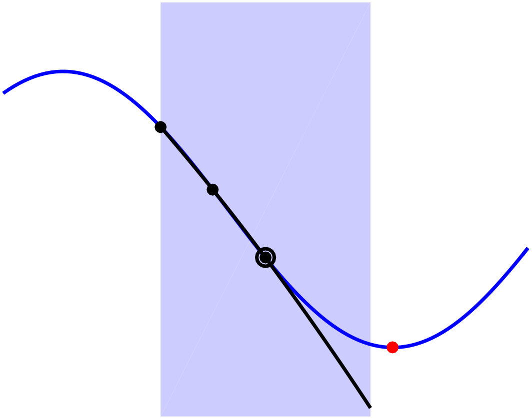
2. Interpolate & minimize...

Example: Model-Based DFO



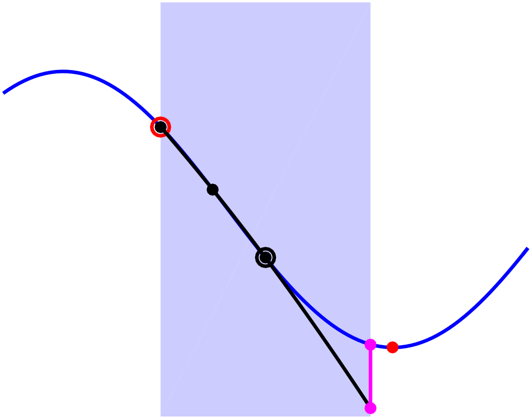
3. Add new point to interpolation set (replace a bad point)

Example: Model-Based DFO



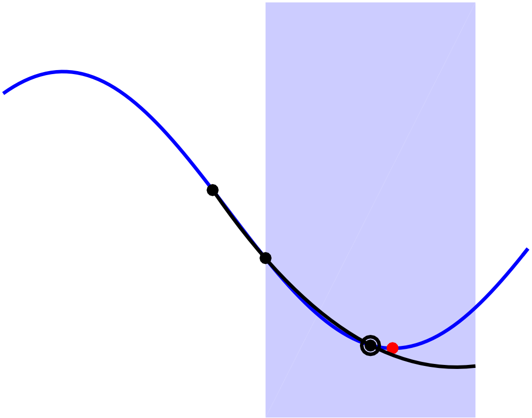
4. Repeat with new interpolation set & model

Example: Model-Based DFO



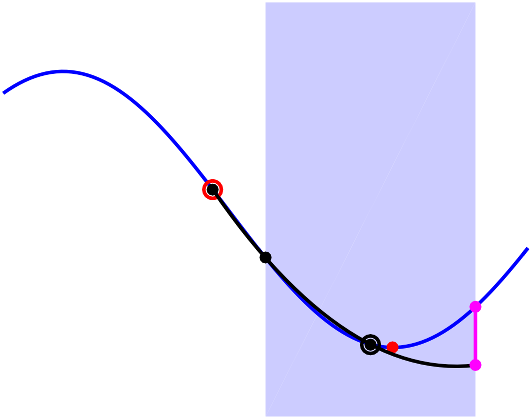
4. Repeat with new interpolation set & model

Example: Model-Based DFO



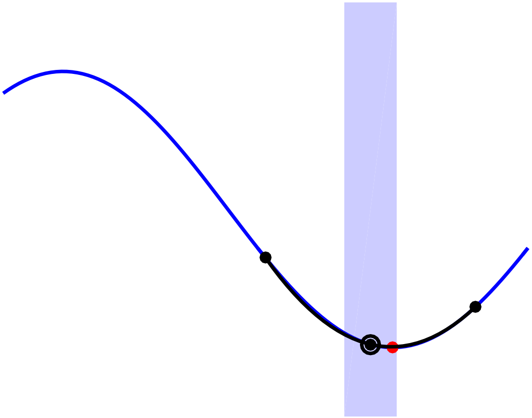
4. Repeat with new interpolation set & model

Example: Model-Based DFO

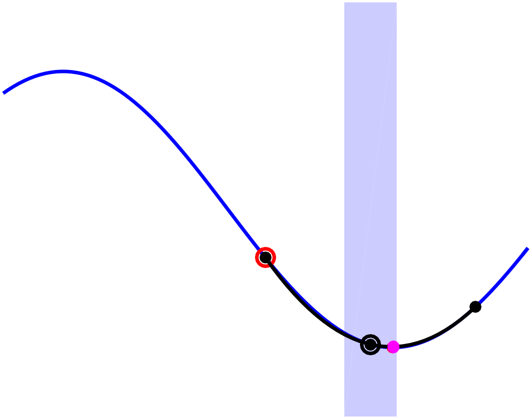


4. Repeat with new interpolation set & model

Example: Model-Based DFO

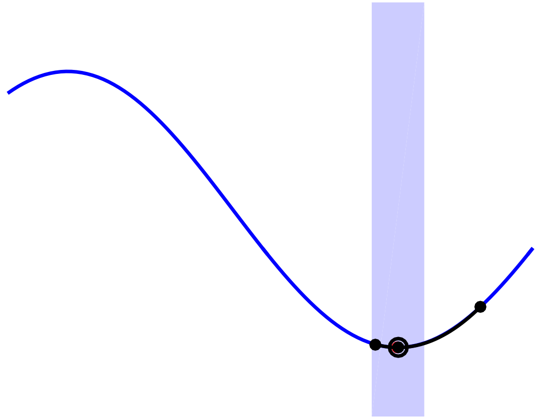


4. Repeat with new interpolation set & model



4. Repeat with new interpolation set & model

Example: Model-Based DFO



4. Repeat with new interpolation set & model

DFO Method 2: Direct Search

- Given $\boldsymbol{x}_k$ and Δ_k, choose a set $\mathcal{D}_k \subset \mathbb{R}^n$ of m vectors
- If there exists $\boldsymbol{d}_k \in \mathcal{D}_k$ with $f(\boldsymbol{x}_k + \Delta_k \boldsymbol{d}_k) < f(\boldsymbol{x}_k) - \frac{1}{2}\Delta_k^2 \|\boldsymbol{d}_k\|_2^2$:
  - Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta_k \boldsymbol{d}_k$ and increase Δ_k
- Otherwise, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and decrease Δ_k

DFO Method 2: Direct Search

- Given $\boldsymbol{x}_k$ and Δ_k, choose a set $\mathcal{D}_k \subset \mathbb{R}^n$ of m vectors
- If there exists $\boldsymbol{d}_k \in \mathcal{D}_k$ with $f(\boldsymbol{x}_k + \Delta_k \boldsymbol{d}_k) < f(\boldsymbol{x}_k) - \frac{1}{2}\Delta_k^2 \|\boldsymbol{d}_k\|_2^2$:
  - Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta_k \boldsymbol{d}_k$ and increase Δ_k
- Otherwise, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and decrease Δ_k

For convergence, need $\mathcal{D}_k$ to be κ-descent:

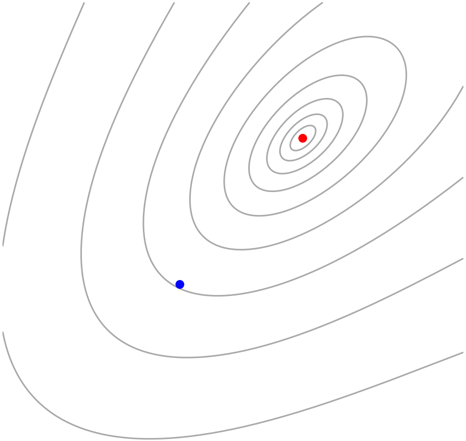
$$\max_{\boldsymbol{d} \in \mathcal{D}_k} \frac{-\boldsymbol{d}^T \nabla f(\boldsymbol{x}_k)}{\|\boldsymbol{d}\|_2 \cdot \|\nabla f(\boldsymbol{x}_k)\|_2} \geq \kappa \in (0, 1]$$

i.e. there is a vector $\boldsymbol{d}$ making an acute angle with $-\nabla f(\boldsymbol{x}_k)$ (descent direction).

Examples: $\{\pm \boldsymbol{e}_1, \ldots, \pm \boldsymbol{e}_n\}$ with $\kappa = 1/\sqrt{n}$ or $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n, -\boldsymbol{e}\}$ with $\kappa \sim 1/n$.

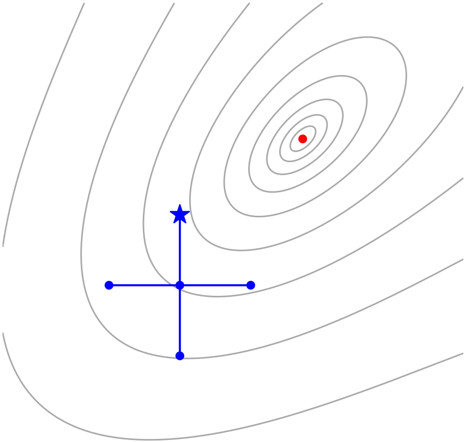
[Kolda, Lewis & Torczon, 2003; Conn, Scheinberg & Vicente, 2009]

Example: Direct Search



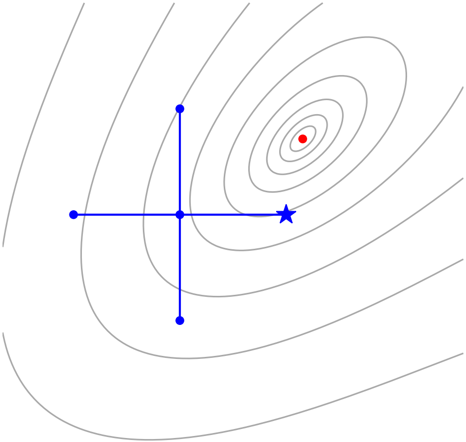
Modified from [Kolda, Lewis & Torczon, 2003]

Example: Direct Search



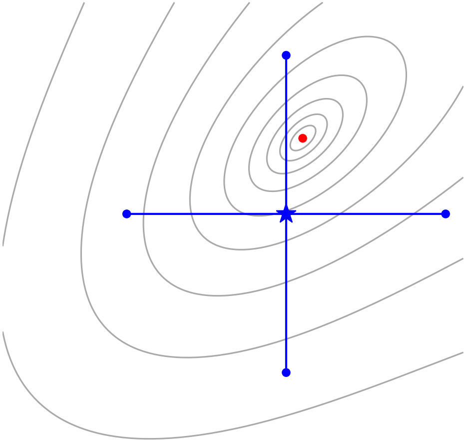
Modified from [Kolda, Lewis & Torczon, 2003]

Example: Direct Search



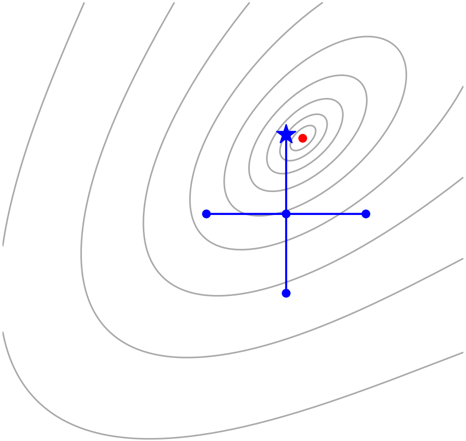
Modified from [Kolda, Lewis & Torczon, 2003]

Example: Direct Search



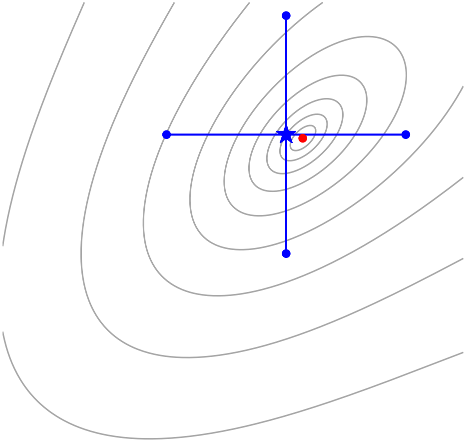
Modified from [Kolda, Lewis & Torczon, 2003]

Example: Direct Search



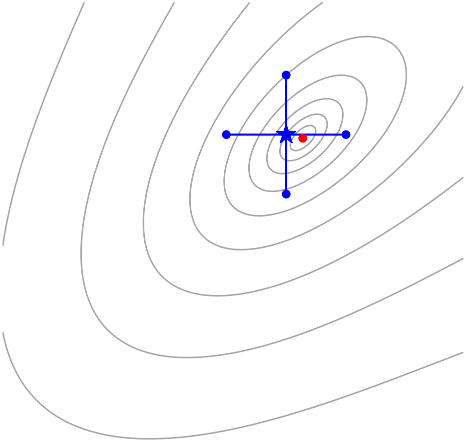
Modified from [Kolda, Lewis & Torczon, 2003]

Example: Direct Search



Modified from [Kolda, Lewis & Torczon, 2003]

Example: Direct Search



Modified from [Kolda, Lewis & Torczon, 2003]

Complexity Theory

Analyze methods using **worst-case complexity**: how long before $\|\nabla f(\mathbf{x}_k)\|_2 \leq \epsilon$?

Metric	Deriv-based	Model-based	Direct search
Iterations	$\mathcal{O}(\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$
Evaluations	$\approx \mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(n^2\epsilon^{-2})$	$\mathcal{O}(n^2\epsilon^{-2})$

[Cartis, Gould & Toint, 2010; Garmanjani, Júdice & Vicente, 2016; Vicente, 2013]

- Same ϵ dependency as derivative-based, but scales badly with problem dimension n
- Model-based DFO also has substantial linear algebra work for interpolation and geometry management: at least $\mathcal{O}(n^3)$ flops per iteration

Challenge

How can DFO methods be made scalable?

Analyze methods using **worst-case complexity**: how long before $\|\nabla f(\boldsymbol{x}_k)\|_2 \leq \epsilon$?

Metric	Deriv-based	Model-based	Direct search
Iterations	$\mathcal{O}(\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$
Evaluations	$\approx \mathcal{O}(n\epsilon^{-2})$	$\cancel{\mathcal{O}(n^2\epsilon^{-2})}$ $\mathcal{O}(n\epsilon^{-2})$	$\cancel{\mathcal{O}(n^2\epsilon^{-2})}$ $\mathcal{O}(n\epsilon^{-2})$

[Cartis, Gould & Toint, 2010; Garmanjani, Júdice & Vicente, 2016; Vicente, 2013]

- Same ϵ dependency as derivative-based, but ~~scales badly with problem dimension n~~
- Model-based DFO also has substantial linear algebra work for interpolation and geometry management: at least $\cancel{\mathcal{O}(n^3)}$ $\mathcal{O}(n)$ flops per iteration

Challenge

How can DFO methods be made scalable?

Outline

1. Introduction to derivative-free optimization (DFO)

2. **Subspace DFO methods**

3. Numerical results

Scalable DFO

Challenge

How can DFO methods be made scalable?

- Exploit known problem structure [Porcelli & Toint, 2020; Bandeira et al., 2012]
- Randomized finite differencing ('gradient sampling') [Nesterov & Spokoiny, 2017]

Applications for scalable DFO methods include:

- Machine learning [Salimans et al., 2017; Ughi et al., 2020]
- Image analysis [Ehrhardt & R., 2021]
- Proxy for global optimization methods [Cartis, R. & Sheridan-Methven, 2021]

Randomized DFO

Challenge

How can DFO methods be made scalable?

Randomization is a promising approach:

- Make model fully linear with probability < 1 [Gratton et al., 2017]
- Make search directions κ-descent with probability < 1 [Gratton et al., 2015]

Randomized DFO

Challenge

How can DFO methods be made scalable?

Randomization is a promising approach:

- Make model fully linear with probability < 1 [Gratton et al., 2017]
- Make search directions κ-descent with probability < 1 [Gratton et al., 2015]

Problem: Improves complexity for direct search, but not for model-based!

Why? Direct search formulation effectively allows dimensionality reduction (sample $\ll n$ directions).

Goal

Use dimensionality reduction techniques suitable for both DFO classes.

Randomization for Dimensionality Reduction

Lemma (Johnson-Lindenstrauss, 1984)

Suppose X is a set of N points in $\mathbb{R}^d$ and $\epsilon \in (0,1)$. Let $A \in \mathbb{R}^{p\times d}$ be a matrix with i.i.d. $N(0, p^{-2})$ entries and $p \sim \log(N)/\epsilon$. Then with high probability,

$$(1-\epsilon)\|x-y\|_2 \le \|Ax - Ay\|_2 \le (1+\epsilon)\|x-y\|_2, \qquad \forall x, y \in X.$$

Randomization for Dimensionality Reduction

Lemma (Johnson-Lindenstrauss, 1984)

Suppose X is a set of N points in $\mathbb{R}^d$ and $\epsilon \in (0,1)$. Let $A \in \mathbb{R}^{p\times d}$ be a matrix with i.i.d. $N(0,p^{-2})$ entries and $p \sim \log(N)/\epsilon$. Then with high probability,

$$(1-\epsilon)\|x-y\|_2 \leq \|Ax-Ay\|_2 \leq (1+\epsilon)\|x-y\|_2, \qquad \forall x,y \in X.$$

- Random projections approximately preserve distances (& inner products, norms, ...)
- Reduced dimension p depends only on # of points N, not the ambient dimension d!
- Other random constructions satisfy J-L Lemma (Haar subsampling, hashing, ...)

Subspace DFO

We use a subspace method: only search in low-dimensional subspaces of $\mathbb{R}^n$

- Related to coordinate descent methods [Wright, 2015; Patrascu & Necoara, 2015]
- Some implementations exist, but no theory [Gross & Parks, 2020; Neumaier et al., 2011]
- Build on recent derivative-based analysis [Cartis, Fowkes & Shao, 2020]

Subspace DFO

We use a subspace method: only search in low-dimensional subspaces of $\mathbb{R}^n$

- Related to coordinate descent methods [Wright, 2015; Patrascu & Necoara, 2015]
- Some implementations exist, but no theory [Gross & Parks, 2020; Neumaier et al., 2011]
- Build on recent derivative-based analysis [Cartis, Fowkes & Shao, 2020]

Subspace DFO framework:

- Generate subspace of dimension $p \ll n$ given by $\mathrm{col}(P_k)$ for random $P_k \in \mathbb{R}^{n \times p}$
- Model-based: build a low-dimensional model $\hat{m}_k(\hat{\boldsymbol{s}})$ which is fully linear for $\hat{f}(\hat{\boldsymbol{s}}) := f(\boldsymbol{x}_k + P_k \hat{\boldsymbol{s}}) : \mathbb{R}^p \to \mathbb{R}$
- Direct search: choose $\mathcal{D}_k \subset \mathbb{R}^p$ which is κ-descent for $P_k^T \nabla f(\boldsymbol{x}_k) \in \mathbb{R}^p$

Fewer interpolation/sample points needed, cheap linear algebra (everything in $\mathbb{R}^p$)

Subspace DFO — Subspace Quality

Choice of subspace: we need to make sure we search in ‘good’ subspaces (where there is potential to decrease f sufficiently).

The subspace at iteration k is well-aligned if

$$\|P_k^T \nabla f(\boldsymbol{x}_k)\|_2 \geq \alpha \|\nabla f(\boldsymbol{x}_k)\|_2, \qquad \text{for some } \alpha > 0.$$

i.e. if there is still work to do, then we know this by only inspecting f in the subspace.

Subspace DFO — Subspace Quality

Choice of subspace: we need to make sure we search in 'good' subspaces (where there is potential to decrease f sufficiently).

The subspace at iteration k is well-aligned if

$$\|P_k^T \nabla f(\boldsymbol{x}_k)\|_2 \geq \alpha \|\nabla f(\boldsymbol{x}_k)\|_2, \qquad \text{for some } \alpha > 0.$$

i.e. if there is still work to do, then we know this by only inspecting f in the subspace.

Key Assumption

The subspace P_k is well-aligned with probability $1 - \delta$.

Using J-L lemma, choose $p \sim (1-\alpha)^{-2}|\log \delta| = \mathcal{O}(1)$ independent of n.

Note: if randomly select p coordinates (block coordinate descent), need $p \sim \alpha n$.

Subspace DFO — Complexity

Theorem (Cartis & R., 2022; R. & Royer, 2022)

If f is sufficiently smooth and bounded below and ϵ sufficiently small, then

$$\mathbb{P}\left[K_{\epsilon} \leq C(p, \alpha, \delta)\epsilon^{-2}\right] \geq 1 - e^{-c(p,\alpha,\delta)\epsilon^{-2}},$$

where K_{ϵ} is the first iteration with $\|\nabla f(\boldsymbol{x}_k)\|_2 \leq \epsilon$.

- Implies $\mathbb{E}\left[K_{\epsilon}\right] = \mathcal{O}(\epsilon^{-2})$ and almost-sure convergence
- $\mathcal{O}(p)$ evaluations per iteration, so same bounds for evaluation complexity

Subspace DFO — Complexity

Standard methods:

Metric	Deriv-based	Model-based	Direct search
Iterations	$\mathcal{O}(\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$
Evaluations	$\approx \mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(n^2\epsilon^{-2})$	$\mathcal{O}(n^2\epsilon^{-2})$

Model-based DFO has $\mathcal{O}(n^3)$ linear algebra work per iteration.

Using random subspaces:

Metric	Deriv-based	Model-based	Direct search
Iterations	$\mathcal{O}(\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$
Evaluations	$\approx \mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(n\epsilon^{-2})$

Model-based DFO has $\mathcal{O}(n)$ linear algebra work per iteration.

Outline

1. Introduction to derivative-free optimization (DFO)

2. Subspace DFO methods

3. **Numerical results**

Software Packages

Open-source Python packages available on Github

Model-Based

DFBGN for nonlinear least-squares (`numericalalgorithmsgroup/dfbgn`)

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})\|_2^2 = \frac{1}{2}\sum_{i=1}^{m} r_i(\boldsymbol{x})^2$$

Subspace method with several heuristics to improve performance

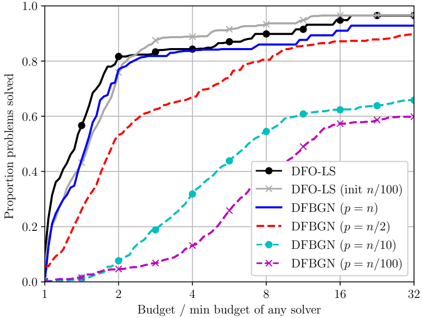
Direct Search

directsearch (`lindonroberts/directsearch`)

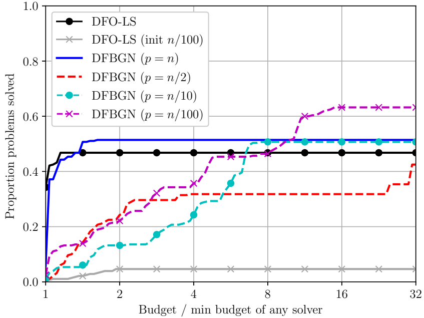
Many varieties of direct search methods (classical, random, subspaces) with multiple $\mathcal{D}_k$ generation methods.

Numerical Results — DFBGN

DFBGN vs. DFO-LS (low accuracy $\tau = 10^{-1}$) *[% problems solved vs. # evals]*



Medium-scale problems, $n \approx 100$

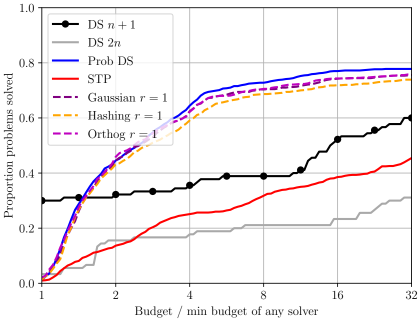


Large problems $n \approx 1000$, 12hr timeout

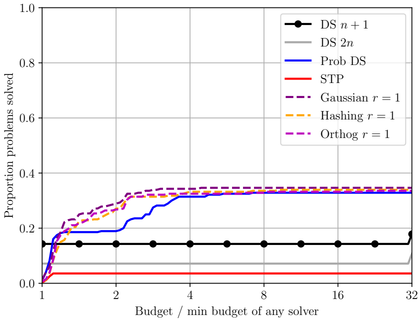
DFBGN is more suitable for low accuracy solutions, performance improves with larger p (except for timeouts!)

Numerical Results — Direct Search

Direct search comparisons (low accuracy $\tau = 10^{-1}$) *[% problems solved vs. # evals]*



Medium-scale problems, $n \approx 100$

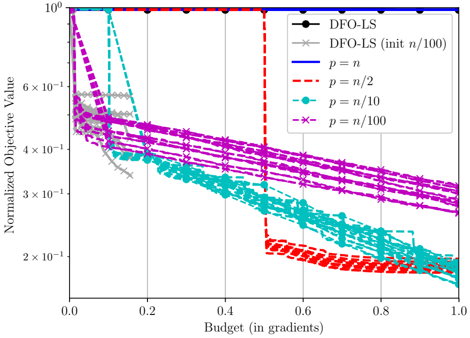


Large problems $n \approx 1000$

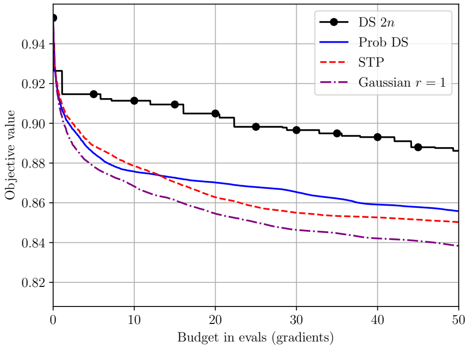
Subspace methods match randomized methods and outperform classical methods, performance best with small p

Numerical Results — low budget

Subspace methods progress after $p \ll n$ evaluations (important when n large)



DFBGN



directsearch

(normalized objective reduction vs. # evaluations, 12hr timeout)

Conclusions & Future Work

Conclusions

- Scalability of model-based DFO is currently limited (in theory & practice)
- Randomized projections are effective dimensionality reduction techniques
- New algorithms reduce linear algebra cost and iteration complexity
- Practical implementations available

Future Work

- Second-order complexity analysis
- Efficient implementation of subspace quadratic models (model-based)
- Problems with constraints
- Impact of noise in objective evaluations
  - New model-based work (2 weeks ago!): [Dhazini & Wild, 2022]

References i

M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C.-J. Hsieh, and M. B. Srivastava, *GenAttack: Practical black-box attacks with gradient-free optimization*, in Proceedings of the Genetic and Evolutionary Computation Conference, Prague, Czech Republic, 2019, ACM, pp. 1111–1119.

A. S. Bandeira, K. Scheinberg, and L. N. Vicente, *Computation of sparse low degree interpolating polynomials and their application to derivative-free optimization*, Mathematical Programming, 134 (2012), pp. 223–257.

C. Cartis, J. Fowkes, and Z. Shao, *A randomised subspace Gauss-Newton method for nonlinear least-squares*, in Workshop on "Beyond first-order methods in ML systems" at the 37th International Conference on Machine Learning, Vienna, Austria, 2020.

C. Cartis, N. I. M. Gould, and P. L. Toint, *On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization problems*, SIAM Journal on Optimization, 20 (2010), pp. 2833–2852.

C. Cartis and L. Roberts, *Scalable subspace methods for derivative-free nonlinear least-squares optimization*, Mathematical Programming, (2022).

References ii

C. Cartis, L. Roberts, and O. Sheridan-Methven, *Escaping local minima with local derivative-free methods: A numerical investigation*, Optimization, (2021).

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.

K. J. Dzahini and S. M. Wild, *Stochastic trust-region algorithm in random subspaces with convergence and expected complexity analyses*, arXiv preprint arXiv:2207.06452, (2022).

M. J. Ehrhardt and L. Roberts, *Inexact derivative-free optimization for bilevel learning*, Journal of Mathematical Imaging and Vision, 63 (2020), pp. 580–600.

R. Garmanjani, D. Júdice, and L. N. Vicente, *Trust-region methods without using derivatives: Worst case complexity and the nonsmooth case*, SIAM Journal on Optimization, 26 (2016), pp. 1987–2011.

I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, in 3rd International Conference on Learning Representations ICLR, San Diego, 2015.

References iii

S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang, *Direct search based on probabilistic descent*, SIAM Journal on Optimization, 25 (2015), pp. 1515–1541.

S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang, *Complexity and global rates of trust-region methods based on probabilistic models*, IMA Journal of Numerical Analysis, 38 (2017), pp. 1579–1597.

J. C. Gross and G. T. Parks, *Optimization by moving ridge functions: Derivative-free optimization for computationally intensive functions*, arXiv preprint arXiv:2007.04893, (2020).

T. G. Kolda, R. M. Lewis, and V. Torczon, *Optimization by direct search: New perspectives on some classical and modern methods*, SIAM Review, 45 (2003), pp. 385–482.

Y. Nesterov and V. Spokoiny, *Random gradient-free minimization of convex functions*, Foundations of Computational Mathematics, 17 (2017), pp. 527–566.

A. Neumaier, H. Fendl, H. Schilly, and T. Leitner, *VXQR: Derivative-free unconstrained optimization based on QR factorizations*, Soft Computing, 15 (2011), pp. 2287–2298.

A. Patrascu and I. Necoara, *Efficient random coordinate descent algorithms for large-scale structured nonconvex optimization*, Journal of Global Optimization, 61 (2015), pp. 19–46.

References iv

M. Porcelli and P. L. Toint, *Global and local information in structured derivative free optimization with BFO*, arXiv preprint arXiv:2001.04801, (2020).

M. J. D. Powell, *On trust region methods for unconstrained minimization without derivatives*, Mathematical Programming, 97 (2003), pp. 605–623.

L. Roberts and C. W. Royer, *Direct search based on probabilistic descent in reduced spaces*, arXiv preprint arXiv:2204.01275, (2022).

T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, *Evolution strategies as a scalable alternative to reinforcement learning*, arXiv preprint arXiv:1703.03864, (2017).

S. F. B. Tett, J. M. Gregory, N. Freychet, C. Cartis, M. J. Mineter, and L. Roberts, *Does model calibration reduce uncertainty in climate projections?*, Journal of Climate, 35 (2022), pp. 2585–2602.

G. Ughi, V. Abrol, and J. Tanner, *An empirical study of derivative-free-optimization algorithms for targeted black-box attacks in deep neural networks*, arXiv preprint arXiv:2012.01901, (2020).

L. N. Vicente, *Worst case complexity of direct search*, EURO Journal on Computational Optimization, 1 (2013), pp. 143–153.

References v

S. J. Wright, *Coordinate descent algorithms*, Mathematical Programming, 151 (2015), pp. 3–34.