# Blockwise Direct-Search Methods

Haitian Li

The Hong Kong Polytechnic University

Joint work with Zaikun Zhang

ISMP 2024, Montreal, Canada

# Blockwise Direct-Search Methods

Zaikun Zhang (replacing Haitian Li)

The Hong Kong Polytechnic University

Joint work with Haitian Li

ISMP 2024, Montreal, Canada

# Derivative-free optimization (DFO): what and when?

## What is DFO?

Solve an optimization problem

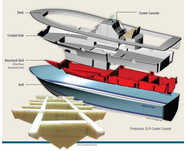
$$\min_{x \in \mathbb{R}^n} f(x)$$

using function values but not derivatives (classical or generalized).

## When do we use DFO?

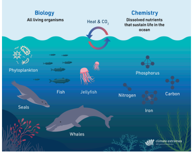
- Derivatives are not available even though $f$ may be smooth.
- "not available": the evaluation is impossible or too expensive.

# Applications of DFO




Ship Design

Machine Learning




Geosciences

1. Campana, Diez, Iemma, Liuzzi, Lucidi, Rinaldi, and Serani, Derivative-free global ship design optimization using global/local hybridization of the DIRECT algorithm. *Optim. Eng.*, 2016.
2. Ghanbari and Scheinberg, Black-box optimization in machine learning with trust region based derivative free algorithm. *arXiv:1703.06925*, 2017.
3. Oliver, Cartis, Kriest, Tett, and Khatiwala, A derivative-free optimisation method for global ocean biogeochemical models. *Geosci. Model Dev.*, 2022.

# Two classes of DFO methods

1. Model-based methods based on
   - trust region
   - line search
2. Direct-search methods based on
   - simplex
   - directions

# Model-based methods v.s. Direct-search methods

| | Model-based | Direct-search |
|---|---|---|
| Performance | good | less satisfactory |
| Implementation | complicated | relatively simple |

1. Model-based methods:
   - The optimization process is guided by models.
   - The coupling between modeling and optimization makes the implementation complicated.
2. Direct-search methods:
   - Iterates are decided by comparing the function values of samples.
   - No need to construct models.

# An example of model-based methods: NEWUOA

- A model-based DFO solver for unconstrained problems
- Developed by M.J.D. Powell
- Widely used by engineers and scientists
- A popular benchmark in the DFO community [1]
- The modernized version: PRIMA (https://github.com/libprima)

[1] Benchmarking derivative-free optimization algorithms, Moré, J. J. and Wild, S. M., SIAM Journal on Optimization, 2009.

# NEWUOA: implementation and understanding are HARD

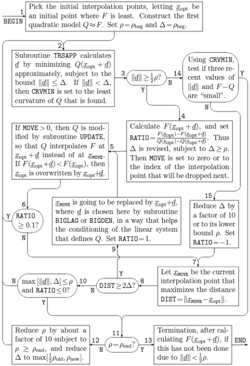



Figure 1: An outline of the method, where Y = Yes and N = No

Framework of NEWUOA

# NEWUOA: implementation and understanding are HARD

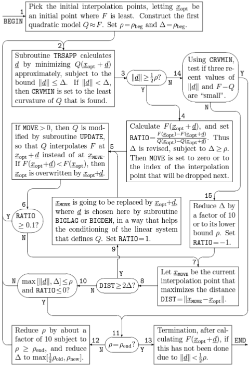



Figure 1: An outline of the method, where Y = Yes and N = No

Framework of NEWUOA

**Powell (2006)**

The development of NEWUOA has taken nearly three years. The work was very frustrating ...

# A much simpler algorithm: Direct Search (DS)

**Algorithm 1:** DS based on sufficient decrease

**Input:** $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \leq \gamma$, $\alpha_0 > 0$, a forcing function $\rho$, and a search direction set $\mathcal{D} \subset \mathbb{R}^n$.

**for** $k = 0, 1, \dots$ **do**

  **if** $f(x_k + \alpha_k d_k) < f(x_k) - \rho(\alpha_k)$ for some $d_k \in \mathcal{D}$ **then**

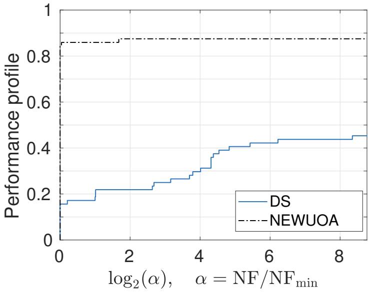
    Set $x_{k+1} = x_k + \alpha_k d_k$ and $\alpha_{k+1} = \gamma \alpha_k$

  **else**

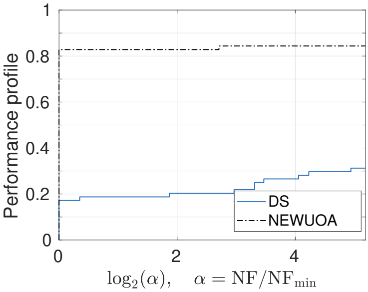
    Set $x_{k+1} = x_k$ and $\alpha_{k+1} = \theta \alpha_k$

**N.B.**: The MADS family is another important class of direct-search methods based on integer lattices without imposing sufficient decrease.

# Unsatisfactory performance of DS



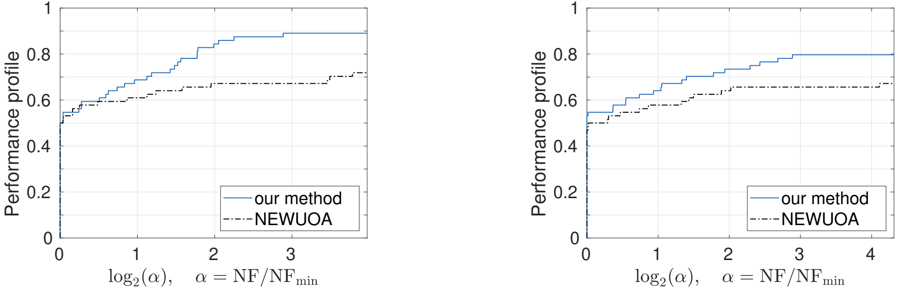

(a) $\tau = 10^{-3}$




(b) $\tau = 10^{-5}$

Unconstrained CUTEst problems, $6 \leq n \leq 200$

# Performance of the new method we will introduce




(a) $\tau = 10^{-3}$ (b) $\tau = 10^{-5}$

Unconstrained CUTEst problems, $6 \leq n \leq 200$

# Flaws of DS?

**Algorithm 1:** DS based on sufficient decrease (recapped)

**Input:** $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \leq \gamma$, $\alpha_0 > 0$, a forcing function $\rho$,
and a search direction set $\mathcal{D} \subset \mathbb{R}^n$.

**for** $k = 0, 1, \ldots$ **do**

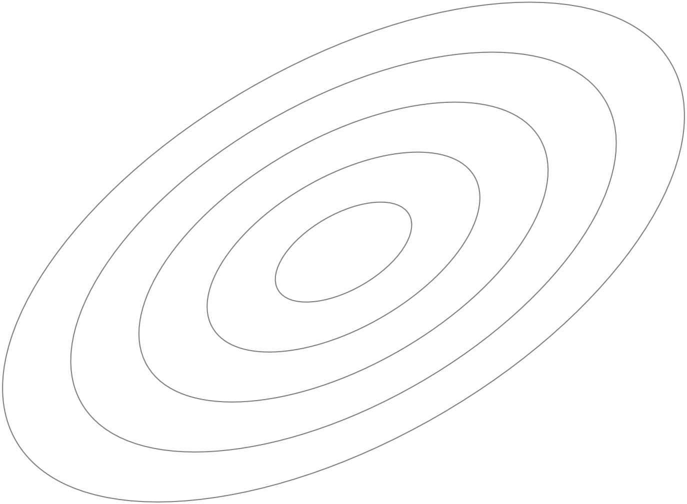
  **if** $f(x_k + \alpha_k d_k) < f(x_k) - \rho(\alpha_k)$ for some $d_k \in \mathcal{D}$ **then**

    Set $x_{k+1} = x_k + \alpha_k d_k$ and $\alpha_{k+1} = \gamma\alpha_k$

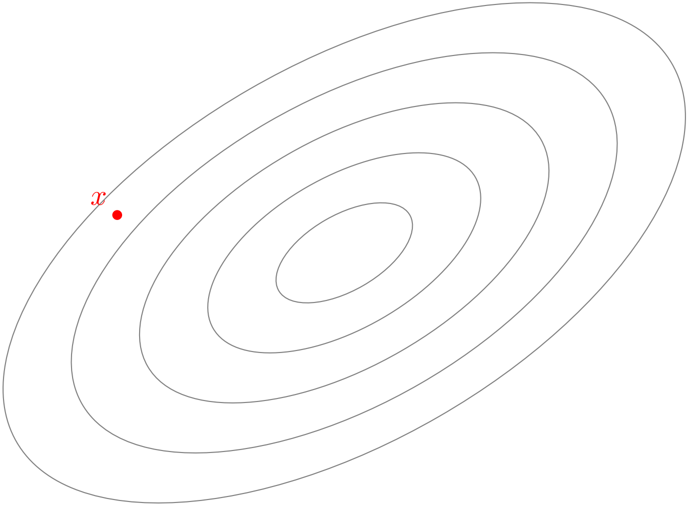
  **else**

    Set $x_{k+1} = x_k$ and $\alpha_{k+1} = \theta\alpha_k$

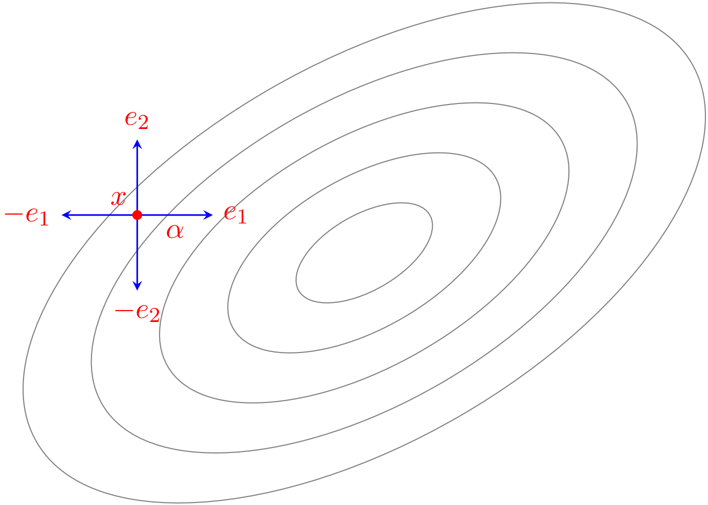
# An illustration of DS



# An illustration of DS

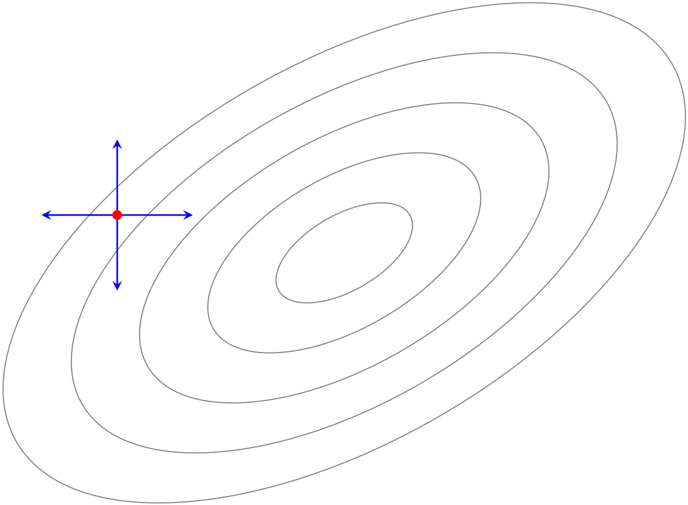


# An illustration of DS

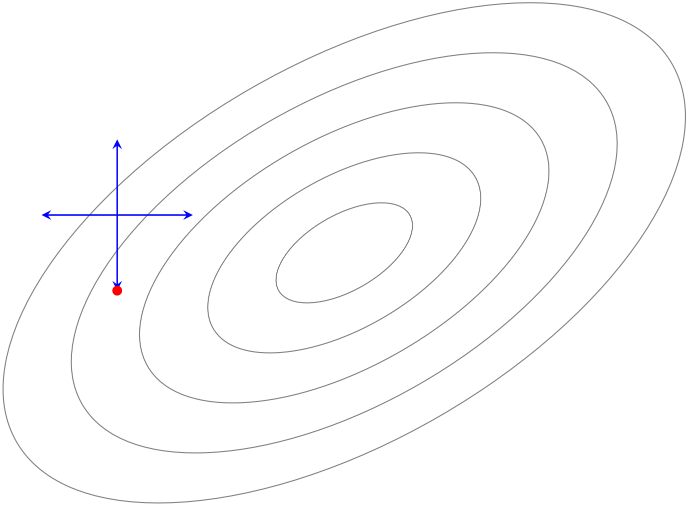



$$\mathcal{D} = \{e_1, -e_1, e_2, -e_2\}$$

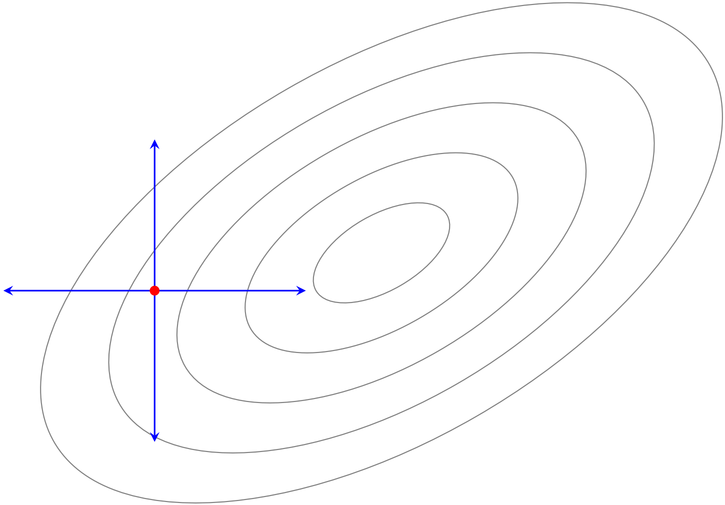
# An illustration of DS



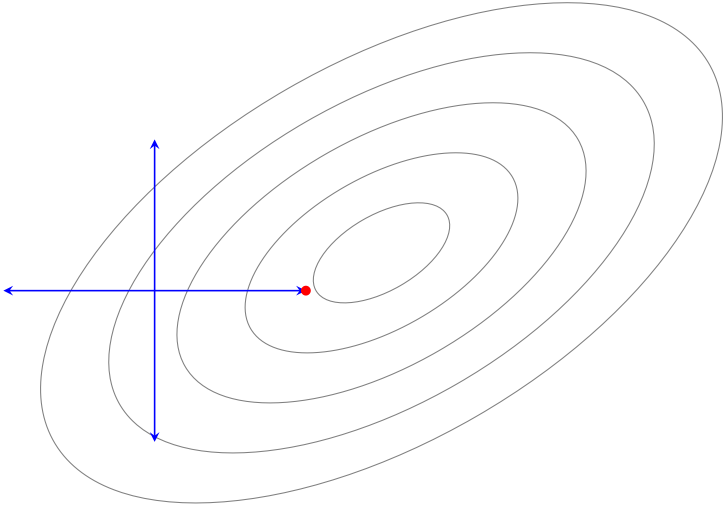
# An illustration of DS



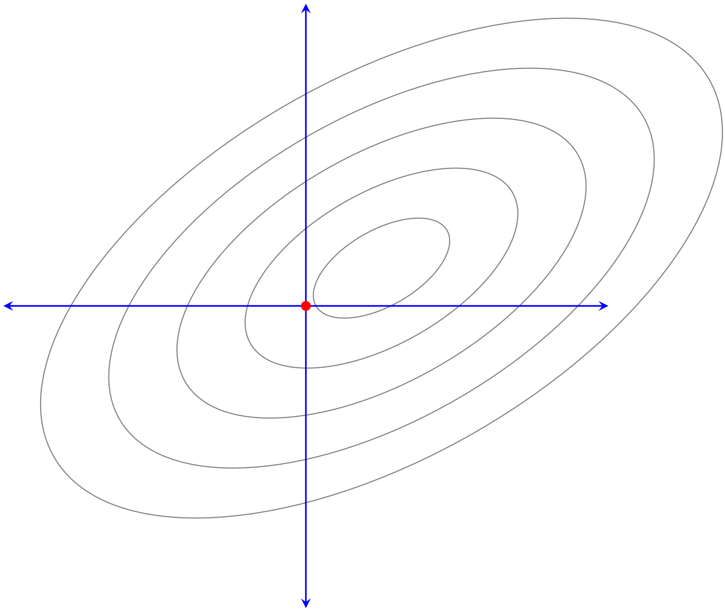
# An illustration of DS



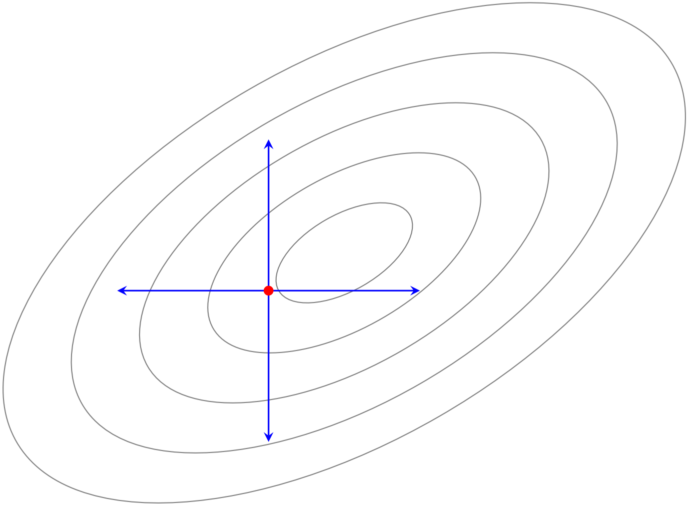
# An illustration of DS



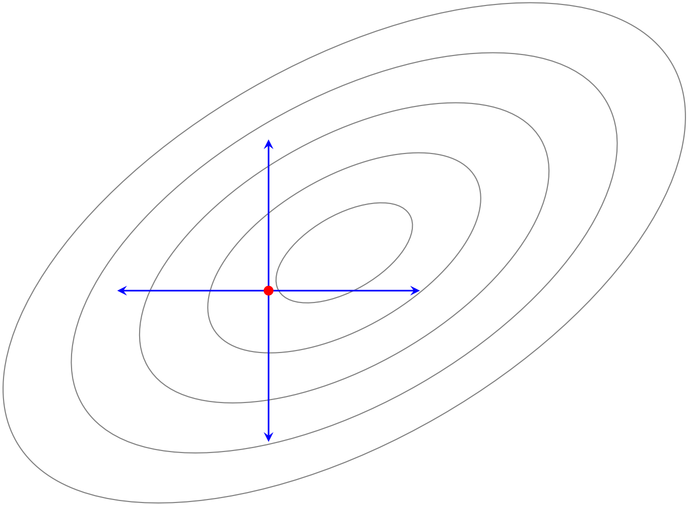
# An illustration of DS



# An illustration of DS

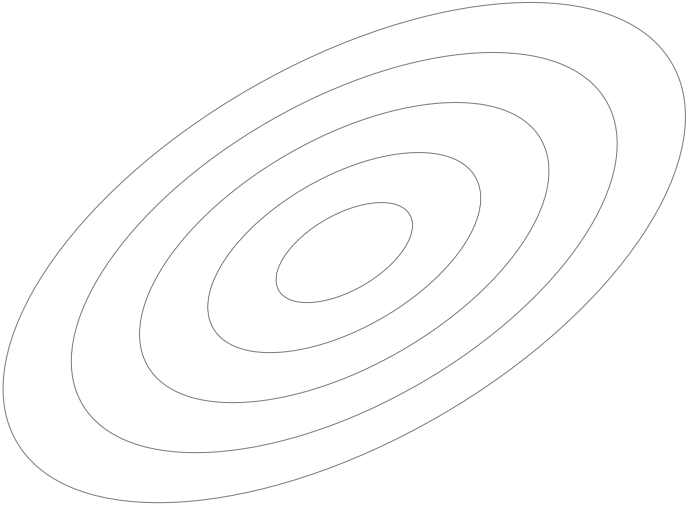


# An illustration of DS

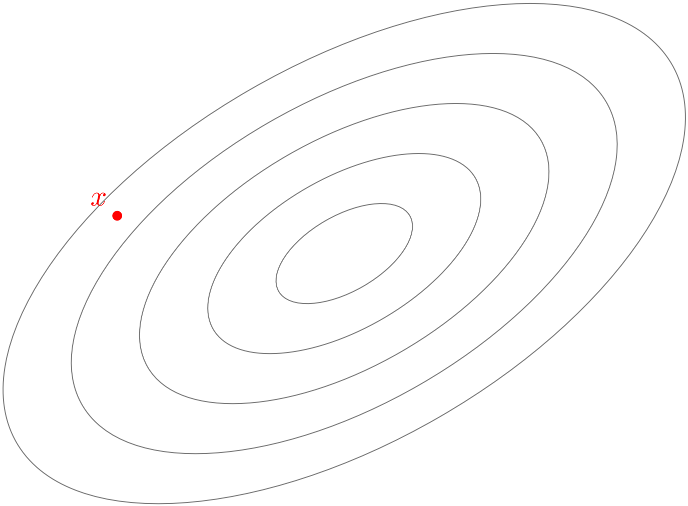


It is not reasonable to have one single stepsize for all directions!

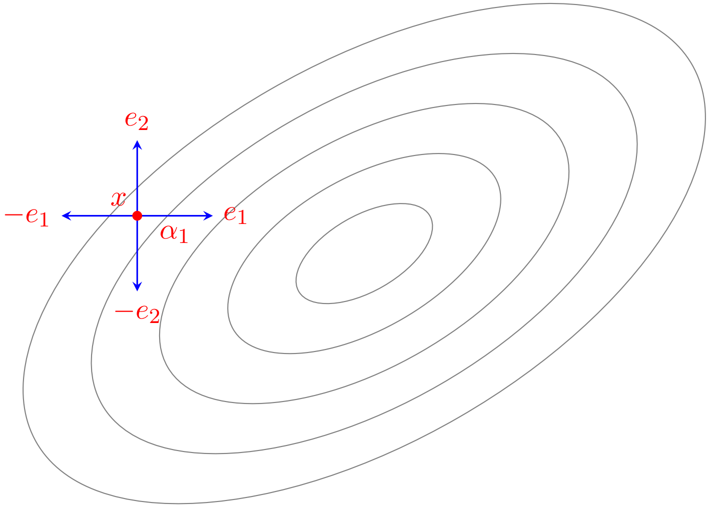
# An improved direct-search method?



# An improved direct-search method?

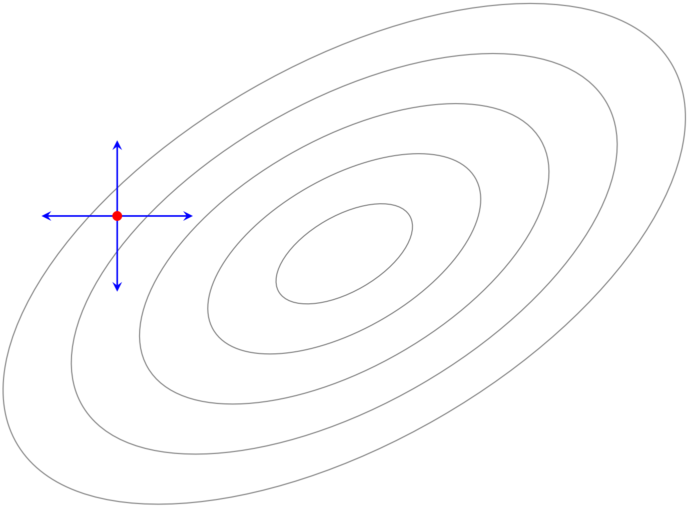


# An improved direct-search method?

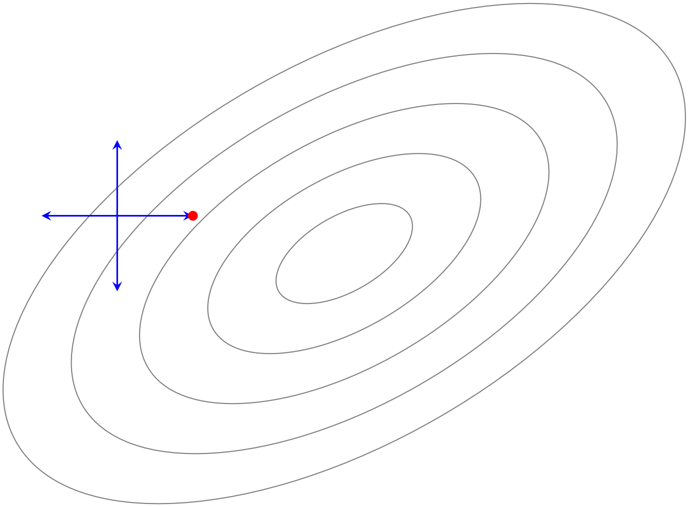



$\mathcal{D}_1 = \{e_1, -e_1\}$ and $\mathcal{D}_2 = \{e_2, -e_2\}$

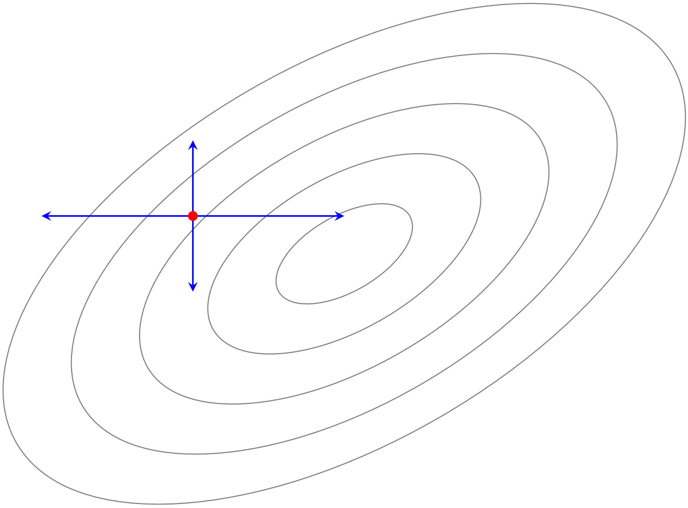
# An improved direct-search method?



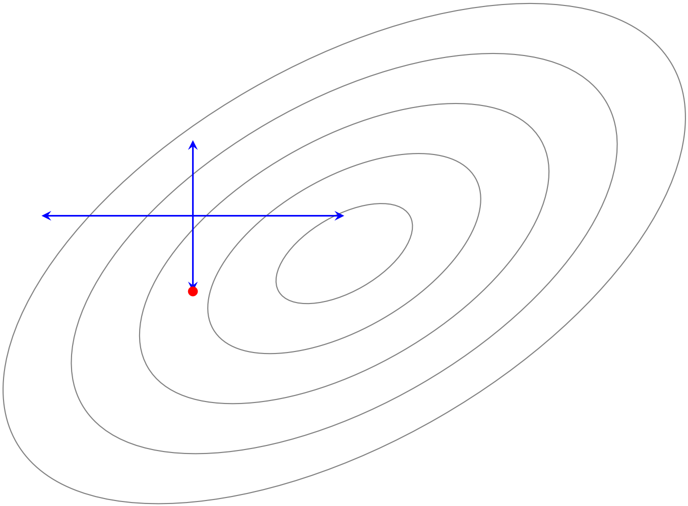
# An improved direct-search method?



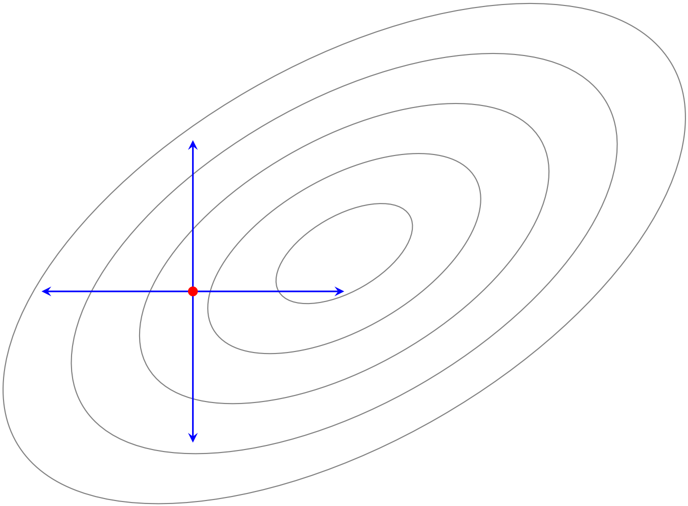
# An improved direct-search method?



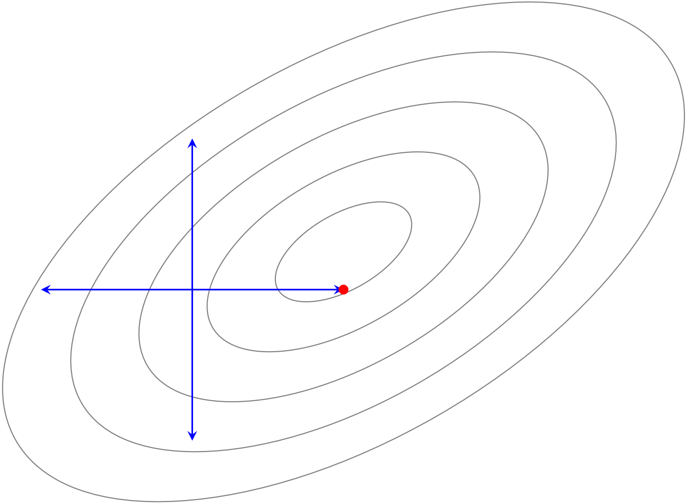
# An improved direct-search method?



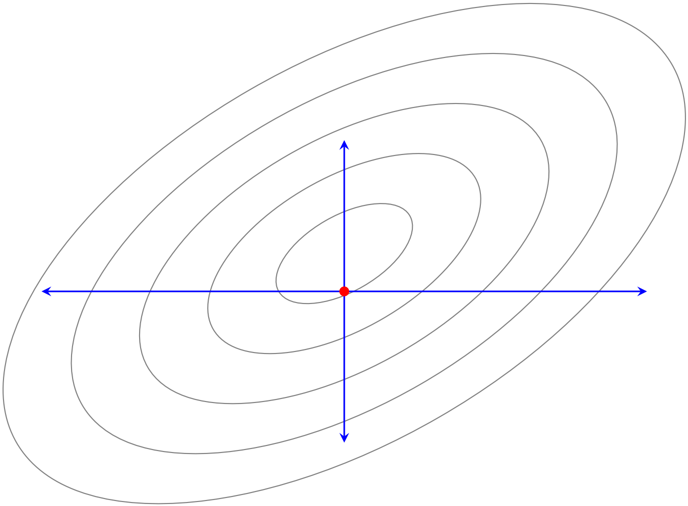
# An improved direct-search method?



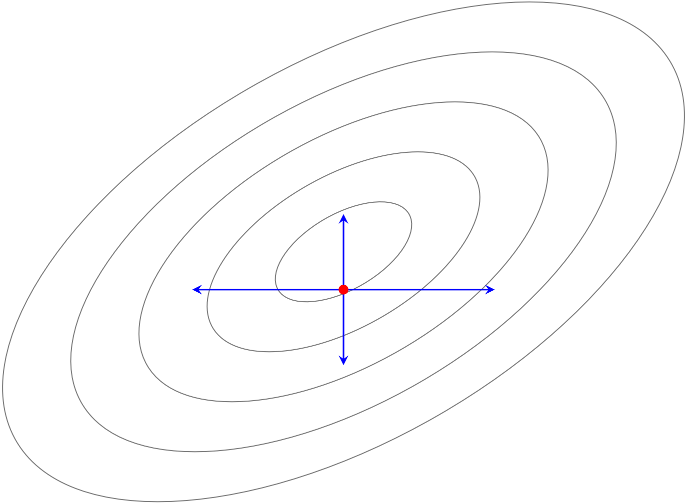
# An improved direct-search method?



# An improved direct-search method?



# An improved direct-search method?



# Blockwise direct-search method

**Algorithm 2:** Blockwise Direct Search (BDS)

**Input:** $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \leq \gamma$, $\alpha_0^1, \dots, \alpha_0^m \in (0, \infty)$, a forcing function $\rho$, and a search direction set $\mathcal{D} = \cup_{i=1}^m \mathcal{D}^i \subset \mathbb{R}^n$.

**for** $k = 0, 1, \dots$ **do**
- Set $y_k^1 = x_k$
- **for** $i = 1, \dots, m$ **do**
  - **if** $f(y_k^i + \alpha_k^i d_k^i) < f(y_k^i) - \rho(\alpha_k^i)$ for some $d_k^i \in \mathcal{D}^i$ **then**
    - Set $y_k^{i+1} = y_k^i + \alpha_k^i d_k^i$ and $\alpha_{k+1}^i = \gamma \alpha_k^i$
  - **else**
    - Set $y_k^{i+1} = y_k^i$ and $\alpha_{k+1}^i = \theta \alpha_k^i$
- Set $x_{k+1} = y_k^{m+1}$

# Blockwise direct-search method

**Algorithm 2:** Blockwise Direct Search (BDS)

**Input:** $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \leq \gamma$, $\alpha_0^1, \dots, \alpha_0^m \in (0, \infty)$, a forcing function $\rho$, and a search direction set $\mathcal{D} = \cup_{i=1}^m \mathcal{D}^i \subset \mathbb{R}^n$.

**for** $k = 0, 1, \dots$ **do**
- Set $y_k^1 = x_k$
- **for** $i = 1, \dots, m$ **do**
  - **if** $f(y_k^i + \alpha_k^i d_k^i) < f(y_k^i) - \rho(\alpha_k^i)$ for some $d_k^i \in \mathcal{D}^i$ **then**
    - Set $y_k^{i+1} = y_k^i + \alpha_k^i d_k^i$ and $\alpha_{k+1}^i = \gamma \alpha_k^i$
  - **else**
    - Set $y_k^{i+1} = y_k^i$ and $\alpha_{k+1}^i = \theta \alpha_k^i$
- Set $x_{k+1} = y_k^{m+1}$

# The difference from DS

- The only difference between BDS and DS: blocks
- No backtracking/extrapolating line search like in
  - Lucidi and Sciandrone, *SIAM J. Optim.*, 2002
  - Brilli, Kimiaei, Liuzzi, and Lucidi, *arXiv:2302.05274*, 2023

# “Blocking”: A classic idea

It is obviously a classic idea to divide search directions into blocks and treat them differently.

- Blockwise Coordinate Descent
- Audet, Le Digabel, and Tribes, Dynamic scaling in the mesh adaptive direct search algorithm for blackbox optimization, *Optim. Eng.*, 2015

# Flexibility of the framework

- The search direction set: a positive spanning set.
- The division of blocks: any ("fits" the problem as much as possible).
- The scheme of visiting blocks: cyclic (Gauss-Seidel), Jacobi, random.

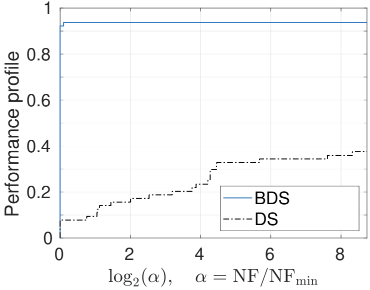
# Flexibility of the framework

- The search direction set: a positive spanning set.
- The division of blocks: any ("fits" the problem as much as possible).
- The scheme of visiting blocks: cyclic (Gauss-Seidel), Jacobi, random.

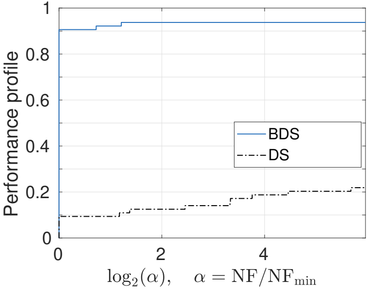
Our implementation takes the following setting as the default:

- $\mathcal{D} = \{e_1, -e_1, \ldots, e_n, -e_n\}$
- $\mathcal{D}^i = \{e_i, -e_i\}$
- Gauss-Seidel scheme

# Comparison between BDS and DS



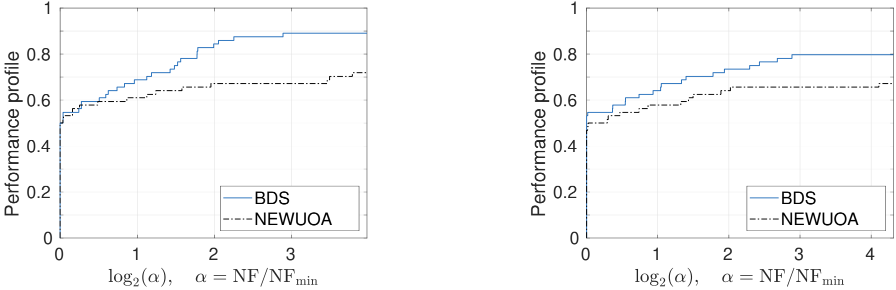

(a) $\tau = 10^{-3}$




(b) $\tau = 10^{-5}$

Unconstrained CUTEst problems, $6 \leq n \leq 200$

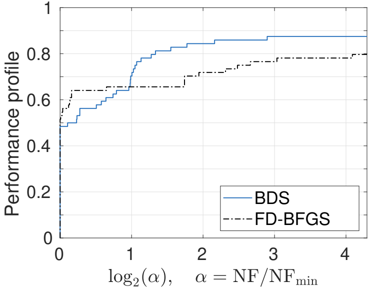
# Comparison betwen BDS and NEWUOA (recapped)



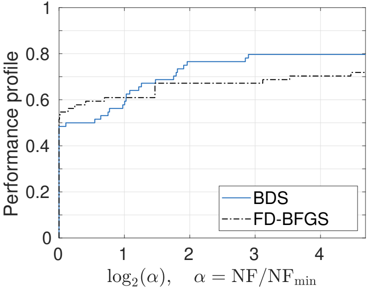

(a) $\tau = 10^{-3}$ (b) $\tau = 10^{-5}$

Unconstrained CUTEst problems, $6 \leq n \leq 200$

# Comparison between BDS and FD-BFGS




(a) $\tau = 10^{-3}$




(b) $\tau = 10^{-5}$

Unconstrained CUTEst problems, $6 \leq n \leq 200$

- FD-BFGS: Forward-finite-difference BFGS (fminunc in MATLAB).

# Performance of BDS under noise

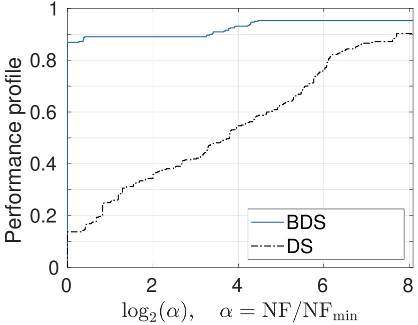
Observed function value:

$$\widetilde{f}(x) = f(x)[1 + \sigma r(x)],$$

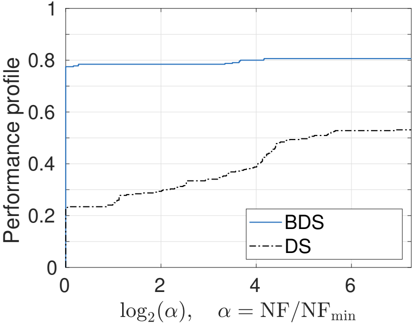
where $r(x) \sim \mathcal{N}(0, 1)$.
In our experiments:

- problem set: unconstrained problems from CUTEst
- dimensions: $6 \leq n \leq 200$
- noise level: $\sigma = 10^{-3}$
- budget: $500n$ function evaluations
- number of random experiments: $5$

# Comparison between BDS and DS

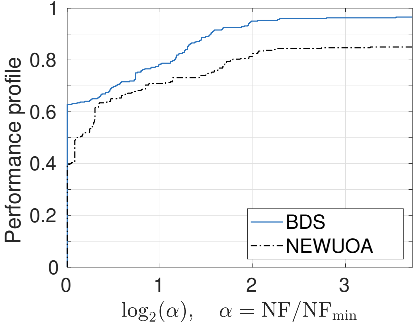



$\tau = 10^{-1}$

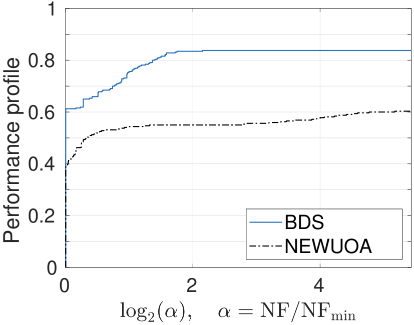



$\tau = 10^{-3}$

# Comparison between BDS and NEWUOA

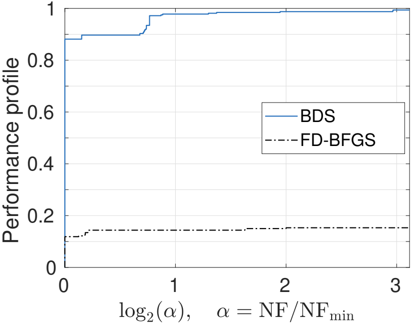



$\tau = 10^{-1}$

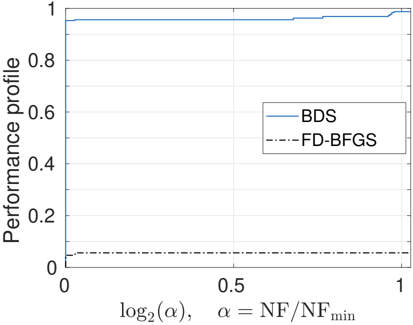



$\tau = 10^{-3}$

# Comparison between BDS and FD-BFGS (fminunc)

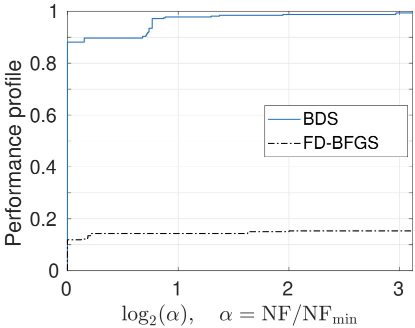



$\tau = 10^{-1}$

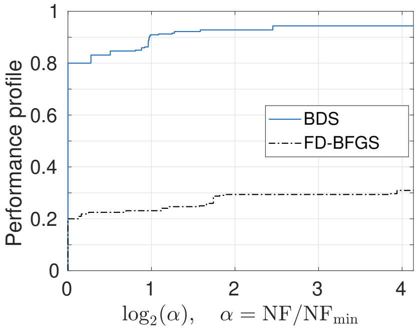



$\tau = 10^{-3}$

# Comparison between BDS and adaptive FD-BFGS



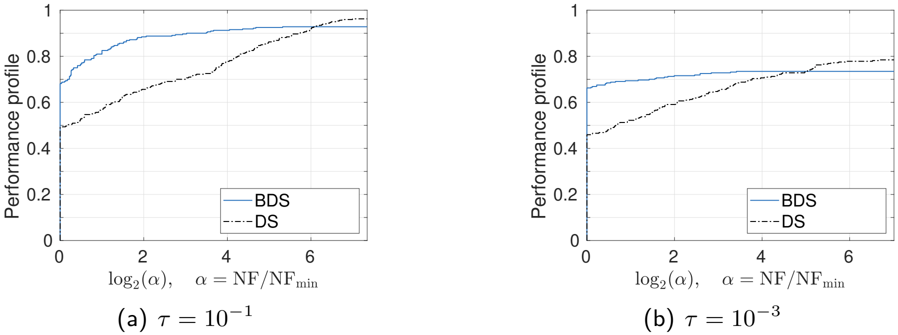

$\tau = 10^{-1}$




$\tau = 10^{-3}$

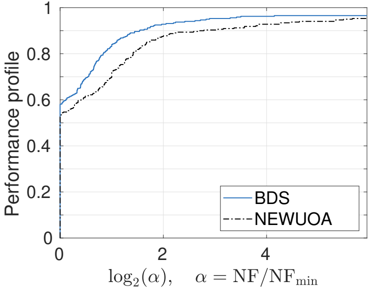
Adaptive stepsize for FD-BFGS: $h = \sqrt{(\max|f|, 1)\sigma}$

# BDS v.s. DS (under rotation)

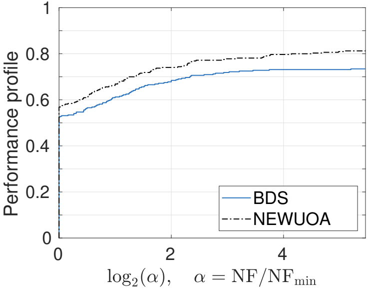



$\widetilde{f}(x) = f(Ux)[1 + \sigma r(x)]$, where $U$ is a random orthogonal matrix

# BDS v.s. NEWUOA (under rotation)



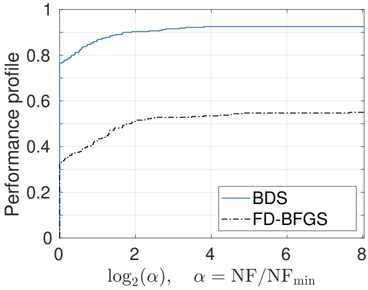

(c) $\tau = 10^{-1}$



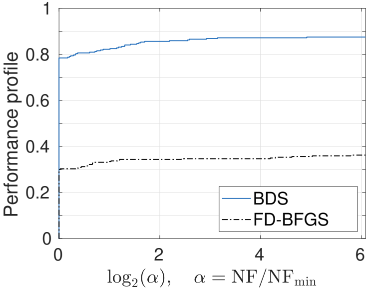

(d) $\tau = 10^{-3}$

$\widetilde{f}(x) = f(Ux)[1 + \sigma r(x)]$, where $U$ is a random orthogonal matrix

# BDS v.s. adaptive FD-BFGS (under rotation)




(e) $\tau = 10^{-1}$



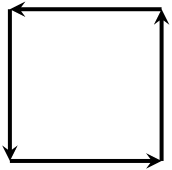

(f) $\tau = 10^{-3}$

$\widetilde{f}(x) = f(Ux)[1 + \sigma r(x)]$, where $U$ is a random orthogonal matrix

Adaptive stepsize for FD-BFGS: $h = \sqrt{(\max|f|, 1)\sigma}$

# Is BDS convergent?

- The analysis of cyclic methods is challenging.
- Powell's non-convergent example of cyclic coordinate descent method [2].



limiting behavior of Powell's example

- We do not know whether BDS is convergent yet.
- Is it possible that the vanilla version of BDS is not convergent?

[2] On search directions for minimization algorithms, Mathematical programming, 1973, Powell, M. J. D.

# Conclusions

1. Blockwise Direct Search (BDS) is a substantial improvement over the classical direct search method based on sufficient decrease
2. BDS is robust under noise without any noise-handling techniques

# Future work

- Convergence and worst-case complexity (an adapted framework?)
- Make use of the existing iterates (finite difference or interpolation)
- Extend our implementation to other languages (Python, Julia, etc.)

BDS on GitHub

- open-source and easy to use
- tested continuously via GitHub Actions
- tested under different platforms

Merci Beaucoup !

# References I


- C. Audet and J. E. Dennis Jr.
  Analysis of generalized pattern searches.
  *SIAM J. Optim.*, 13:889–903, 2002.
- C. Audet and J. E. Dennis Jr.
  Mesh adaptive direct search algorithms for constrained optimization.
  *SIAM J. Optim.*, 17:188–217, 2006.
- C. Audet and D. Orban.
  Finding optimal algorithmic parameters using derivative-free optimization.
  *SIAM J. Optim.*, 17:642–664, 2006.
- A. S. Bandeira, K. Scheinberg, and L. N. Vicente.
  Convergence of trust-region methods based on probabilistic models.
  *SIAM J. Optim.*, 24:1238–1264, 2014.

# References II


- E. F. Campana, M. Diez, U. Iemma, G. Liuzzi, S. Lucidi, F. Rinaldi, and A. Serani.
  Derivative-free global ship design optimization using global/local hybridization of the direct algorithm.
  *Optim. Eng.*, 17:127–156, 2016.
- N. I. M. Gould, D. Orban, and Ph. L. Toint.
  CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization.
  *Comput. Optim. Appl.*, 60:545–557, 2015.
- S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang.
  Direct search based on probabilistic descent.
  *SIAM J. Optim.*, 25:1515–1541, 2015.

- T. Gu, W. Li, A. Zhao, Z. Bi, X. Li, F. Yang, C. Yan, W. Hu, D. Zhou, T. Cui, X. Liu, Z. Zhang, and X. Zeng.
  BBGP-sDFO: Batch Bayesian and Gaussian process enhanced subspace derivative free optimization for high-dimensional analog circuit synthesis.
  *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 43:417–430, 2023.
- T. G. Kolda, R. M. Lewis, and V. Torczon.
  Optimization by direct search: New perspectives on some classical and modern methods.
  *SIAM Rev.*, 45:385–482, 2003.
- S. Oliver, C. Cartis, I. Kriest, S. F. B. Tett, and S. Khatiwala.
  A derivative-free optimisation method for global ocean biogeochemical models.
  *Geosci. Model Dev.*, 15:3537–3554, 2022.

# References IV


- M. Porcelli and Ph. L. Toint.
  BFO, a trainable derivative-free brute force optimizer for nonlinear bound-constrained optimization and equilibrium computations with continuous and discrete variables.
  *ACM Trans. Math. Software*, 44:6:1–6:25, 2017.
- M. J. D. Powell.
  On search directions for minimization algorithms.
  *Math. Program.*, 4:193–201, 1973.
- M. J. D. Powell.
  The NEWUOA software for unconstrained optimization without derivatives.
  In G. Di Pillo and M. Roma, editors, *Large-scale Nonlinear Optimization*, pages 255–297. Springer, Boston, 2006.