Blockwise Direct Search Methods for Derivative-free Optimization

Haitian LI

The Hong Kong Polytechnic University

collaborated with Zaikun ZHANG and Cunxin HUANG

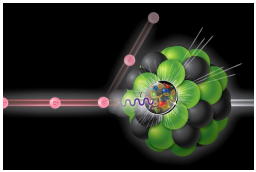
2026 Annual Meeting of the Guangdong Operations Research Society

Zeroth-order / Derivative-free optimization (DFO)

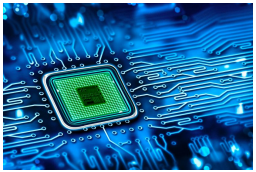


- Minimize f using function values, not 1st-order info., e.g., derivatives.
- Example 1: f is a black box given by simulations. No formula (yet).
- Example 2: 1st-order info. of f is possible to get, but too expensive.
- Even the evaluation of f is often expensive and noisy.
- Aim: Minimize/reduce f using as few function evaluations as possible.
- Examples of methods: Nelder–Mead, Powell's algorithms, SOLNP+, ...

Applications of DFO



Nuclear physics



Circuit design



Artificial intelligence

- Eldred, Larson, Padidar, Stein, and Wild, Derivative-free optimization of a rapid-cycling synchrotron, *Optim. Eng.*, 2022.
- Zeng, Liu, Zhang, et al., BBGP-sDFO: Batch Bayesian and Gaussian Process-enhanced subspace Derivative-Free Optimization for high-dimensional analog circuit synthesis, *IEEE TCAD*, 2024.
- Chen et al., Enhancing zeroth-order fine-tuning for language models with low-rank structures, *ICLR*, 2025.
- Wang et al., Zeroth-order federated fine-tuning for large AI models in resource-constrained wireless networks, *IEEE Trans. Wirel. Commun.*, 2026.

Two common classes of DFO methods

- Model-based methods: build a local approximation from function values, then use it to guide the search.
  - Model construction: interpolation or finite differences.
- Direct search methods: compare objective values directly, without constructing a model.
  - Example: Nelder–Mead simplex method

Other important DFO approaches include Bayesian optimization, evolutionary methods, simulated annealing, ...

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
- A modern implementation: PRIMA



PRIMA on GitHub

NEWUOA: powerful, but hard to implement and understand

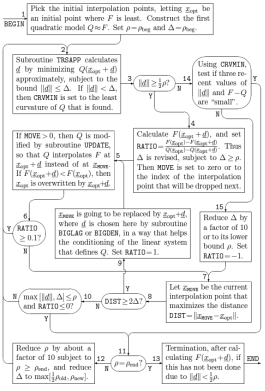


Figure 1: An outline of the method, where Y=Yes and N=No

Framework of NEWUOA

The development of NEWUOA has taken nearly three years. The work was very frustrating, due to severe damage from computer rounding errors in difficult cases, before the factorization (4.16) of Ω was introduced. There-

Powell (2006, p. 292)

NEWUOA: powerful, but hard to implement and understand

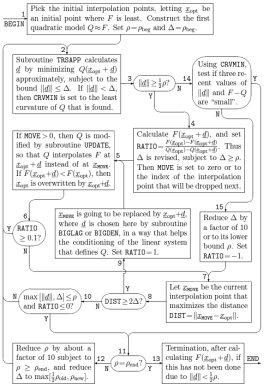


Figure 1: An outline of the method, where Y=Yes and N=No

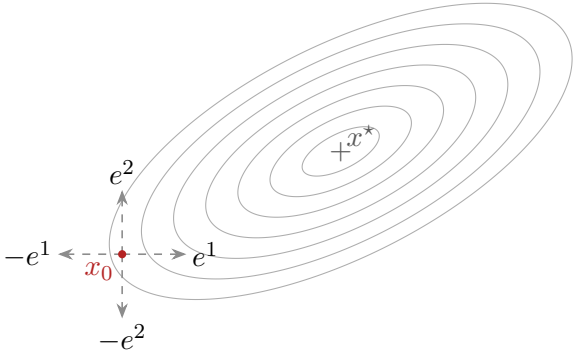
Framework of NEWUOA

The development of NEWUOA has taken nearly three years. The work was very frustrating, due to severe damage from computer rounding errors in difficult cases, before the factorization (4.16) of Ω was introduced. There-

Powell (2006, p. 292)

- The coupling between modeling and optimization makes the implementation complicated.

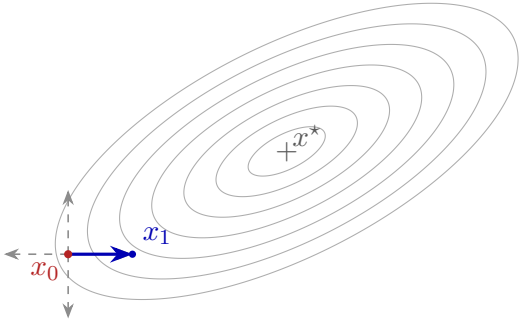
A 2D illustration of Direct Search (DS)



$$\mathcal{D} = \{e^1, -e^1, e^2, -e^2\}.$$

One shared step size, initially $\alpha_0 = 1$.

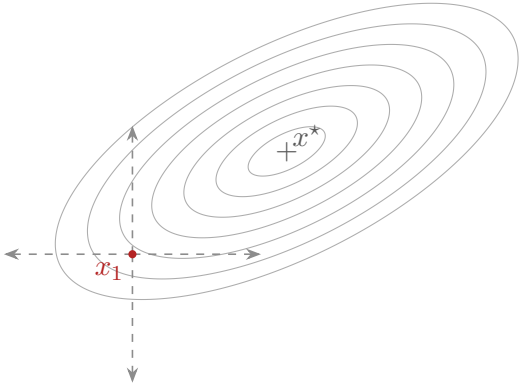
A 2D illustration of Direct Search (DS)



The e^1 trial gives enough decrease.

Accept $x_1 = x_0 + e^1$.

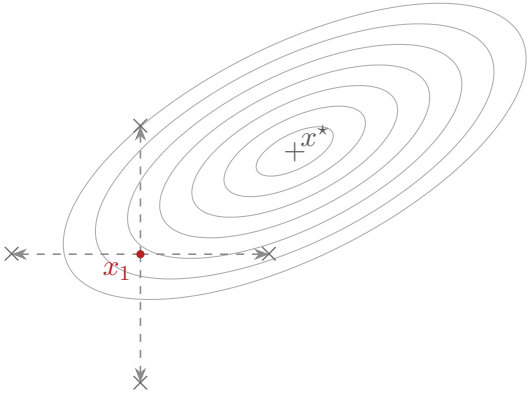
A 2D illustration of Direct Search (DS)



Expand the shared step size from 1 to 2.

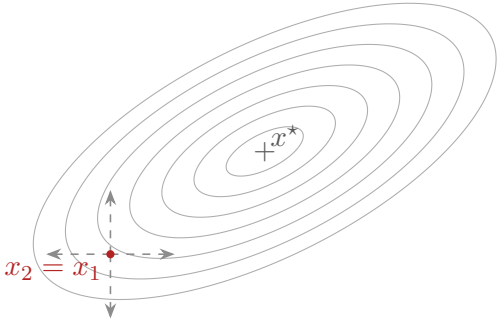
The trials in both coordinate directions become longer.

A 2D illustration of Direct Search (DS)



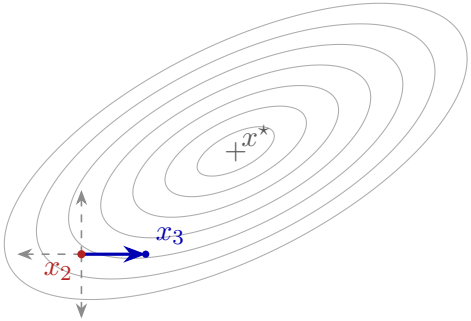
At step size 2, all four trials fail.

A 2D illustration of Direct Search (DS)



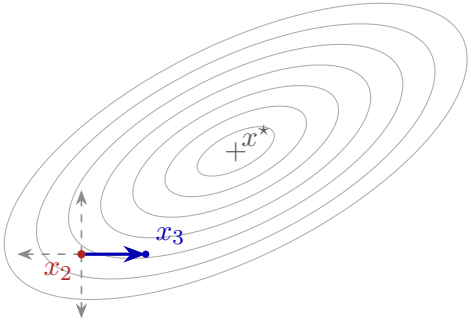
Shrink the shared step size from 2 to 1.

A 2D illustration of Direct Search (DS)



At step size 1, the e^1 trial succeeds.

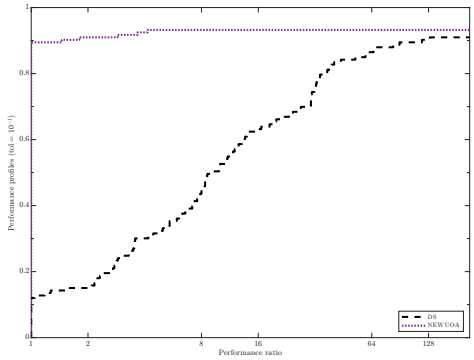
A 2D illustration of Direct Search (DS)



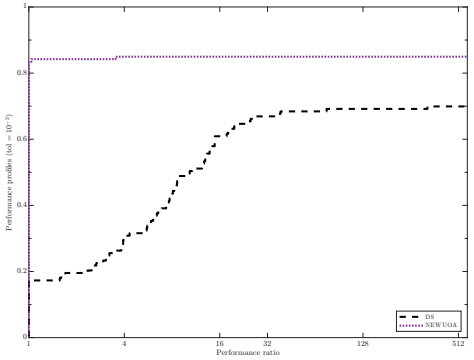
At step size 1, the e^1 trial succeeds.

Can different directions use different step sizes?

Unsatisfactory performance of classical direct search



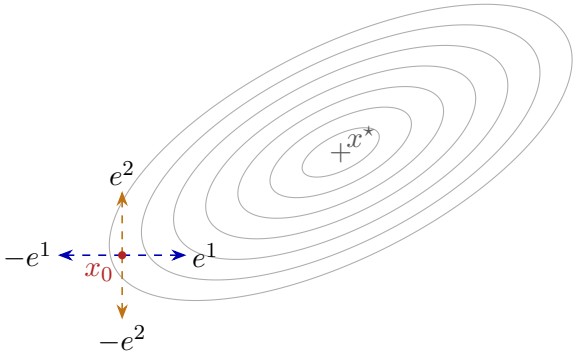
(a) $\tau = 10^{-1}$



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

133 unconstrained CUTEst problems, $6 \leq n \leq 50$

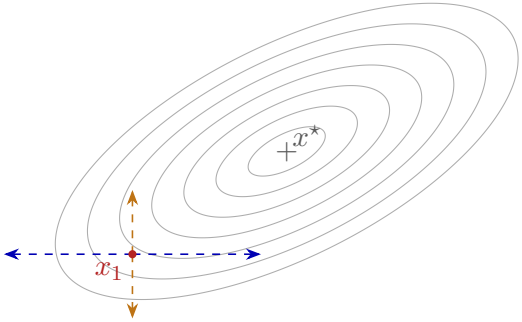
An improved direct-search method?



$\mathcal{D}^1 = \{e^1, -e^1\}, \quad \mathcal{D}^2 = \{e^2, -e^2\}.$

Same problem, same x_0, and $\alpha^1 = \alpha^2 = 1$.

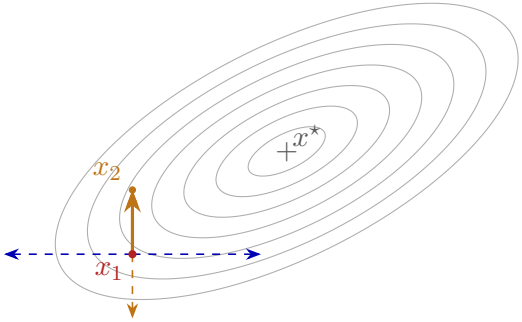
An improved direct-search method?



The e^1 trial succeeds. Expand only block 1.

$$\alpha^1 = 2, \qquad \alpha^2 = 1.$$

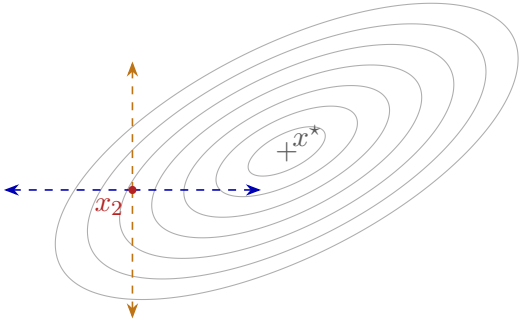
An improved direct-search method?



The e^2 trial succeeds.

Each block has its own step size.

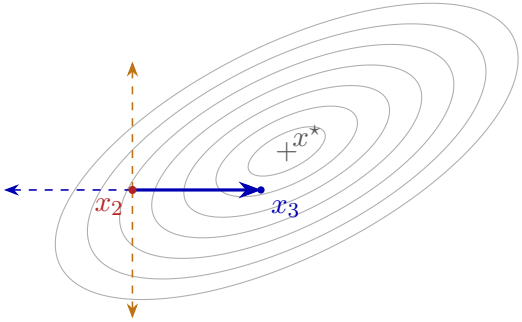
An improved direct-search method?



Expand only block 2 from 1 to 2.

$$\alpha^1 = 2, \qquad \alpha^2 = 2.$$

An improved direct-search method?



From this new point, the length-2 e^1 trial succeeds.

Accept $x_3 = x_2 + 2e^1$.

A General Framework for Blockwise Direct Search

Algorithm 1: Blockwise Direct Search

Input: $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \leq \gamma$, $\alpha_0^1, \ldots, \alpha_0^m > 0$, $c > 0$, and block direction sets $\mathcal{D}^1, \ldots, \mathcal{D}^m$.

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

  Select a block $i_k \in \{1, \ldots, m\}$

  if *there exists* $d \in \mathcal{D}^{i_k}$ *such that* $f(x_k + \alpha_k^{i_k} d) < f(x_k) - c(\alpha_k^{i_k})^2$ **then**

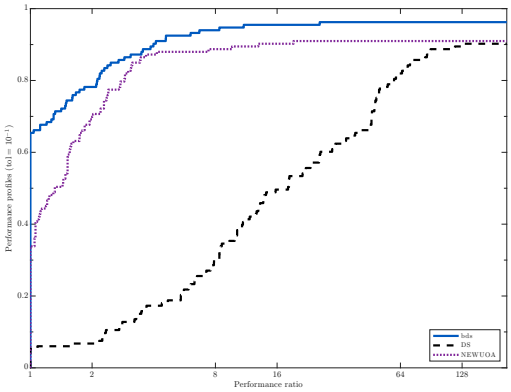
    Set $x_{k+1} = x_k + \alpha_k^{i_k} d$ and $\alpha_{k+1}^{i_k} = \gamma \alpha_k^{i_k}$

  else

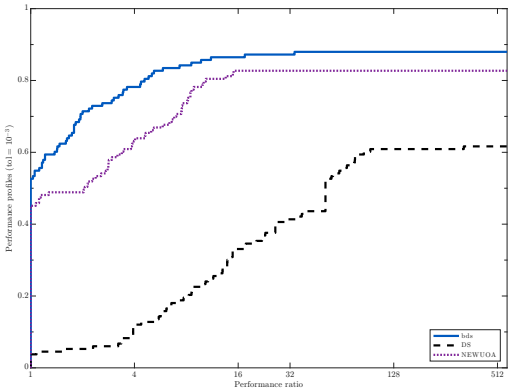
    Set $x_{k+1} = x_k$ and $\alpha_{k+1}^{i_k} = \theta \alpha_k^{i_k}$

- Step sizes of unvisited blocks remain unchanged: $\alpha_{k+1}^j = \alpha_k^j$ for every $j \neq i_k$.
- For example, visit $\mathcal{D}^i = \{e^i, -e^i\}$, $i = 1, \ldots, n$, cyclically.

Can we improve BDS further?



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

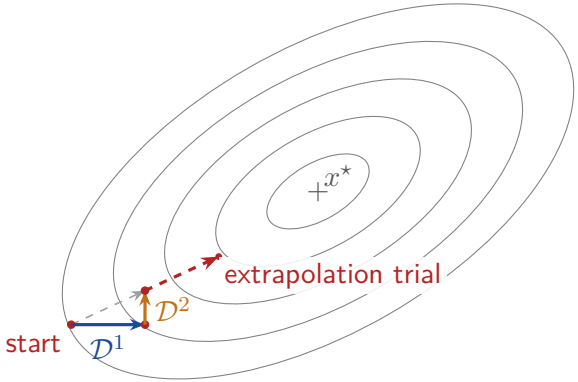


(b) $\tau = 10^{-3}$

133 unconstrained CUTEst problems, $6 \leq n \leq 50$

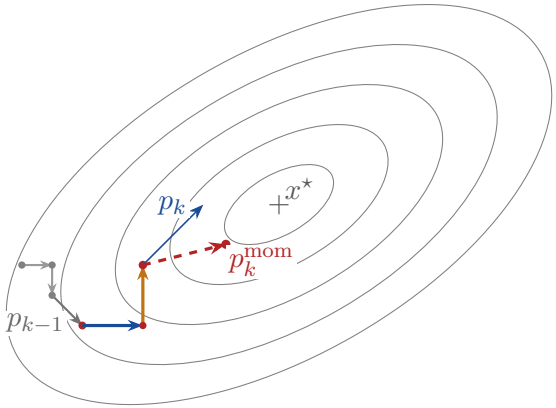
Directional extrapolation in BDS: a 2D illustration

For $n = 2$, visit $\mathcal{D}^1 = \{e^1, -e^1\}$ and $\mathcal{D}^2 = \{e^2, -e^2\}$ alternately.



- Directional extrapolation along $s = \alpha^1 e^1 + \alpha^2 e^2$.

Combining extrapolation directions: momentum

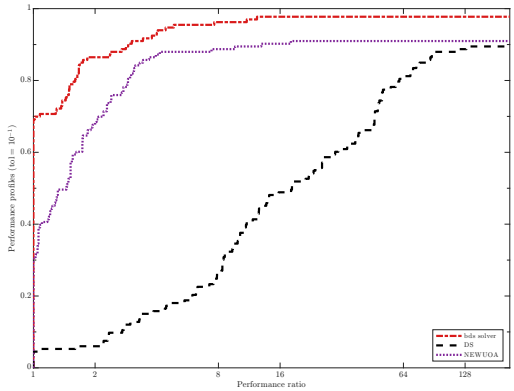


- Form a linear combination, e.g., $p_k^{\mathrm{mom}} = p_k + \beta\, p_{k-1}$.
- Try momentum first. If it gives no improvement, try the current extrapolation direction.

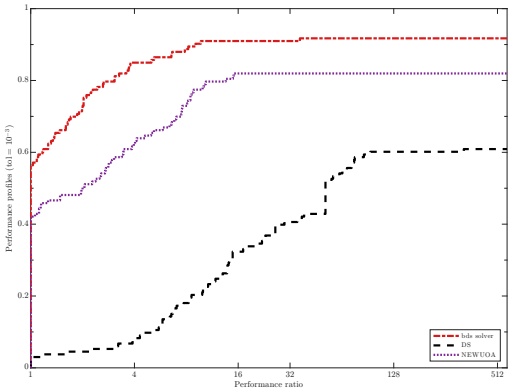
Other practical mechanisms

- Initialization: choose initial step sizes blockwise from the scale of x_0.
- Stopping: reuse existing trial values without extra function evaluations.
- Failed simulations: if a trial errors or returns NaN, assign $+\infty$ and continue.

Performance of the complete BDS solver



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



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

133 unconstrained CUTEst problems, $6 \leq n \leq 50$

Performance of the BDS solver under noise

Observed function value:

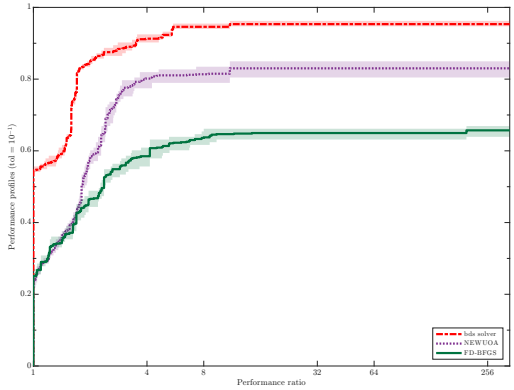
$$\widetilde{f}_\sigma(x) = f(x) + \max\{|f(x)|, 1\}\,\sigma Z, \qquad Z \sim \mathcal{N}(0,1).$$

Z is sampled at every function evaluation.

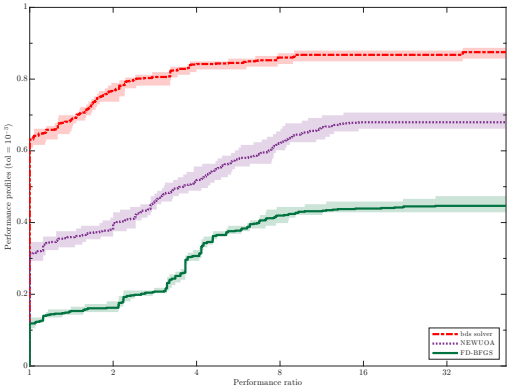
In our experiments:

- problem set: 133 unconstrained CUTEst problems
- dimensions: $6 \leq n \leq 50$
- noise level: $\sigma = 10^{-3}$
- budget: $500n$ function evaluations
- runs per noisy problem: 5

Performance of the BDS solver under noise



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



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

Adaptive forward-difference step for FD-BFGS: $h(x) = \sqrt{\sigma \max\{|\widetilde{f}(x)|, 1\}}$.

A General Framework for Blockwise Direct Search (Recap)

Algorithm 1: Blockwise Direct Search

Input: $x_0 \in \mathbb{R}^n$, $0 < \theta < 1 \leq \gamma$, $\alpha_0^1, \ldots, \alpha_0^m > 0$, $c > 0$, and block direction sets $\mathcal{D}^1, \ldots, \mathcal{D}^m$.

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

  Select a block $i_k \in \{1, \ldots, m\}$

  if *there exists* $d \in \mathcal{D}^{i_k}$ *such that* $f(x_k + \alpha_k^{i_k} d) < f(x_k) - c(\alpha_k^{i_k})^2$ **then**

    Set $x_{k+1} = x_k + \alpha_k^{i_k} d$ and $\alpha_{k+1}^{i_k} = \gamma \alpha_k^{i_k}$

  else

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

- Step sizes of unvisited blocks remain unchanged: $\alpha_{k+1}^j = \alpha_k^j$ for every $j \neq i_k$.

Is BDS convergent?

Is BDS convergent?

Yes.

Is BDS convergent?

Yes.

Problem assumptions

- $f : \mathbb{R}^n \to \mathbb{R}$ is convex and continuously differentiable.
- ∇f is globally Lipschitz continuous.
- the initial sublevel set is bounded.

Is BDS convergent?

Yes.

Problem assumptions

- $f : \mathbb{R}^n \to \mathbb{R}$ is convex and continuously differentiable.
- ∇f is globally Lipschitz continuous.
- the initial sublevel set is bounded.

Algorithmic assumptions

- Every block is visited infinitely often.
- $\bigcup_{i=1}^{m} \mathcal{D}^i$ positively spans $\mathbb{R}^n$.

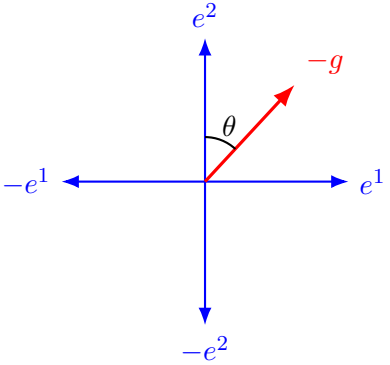
Flexible block selection

Every block is visited infinitely often.

$$\#\{k : i_k = i\} = \infty \qquad \text{for every block } i.$$

- No frequency or order is prescribed.
- Blocks may be selected deterministically or randomly.

What does positive spanning mean?



- $\mathcal{D}$ positively spans $\mathbb{R}^n$ if every $v \in \mathbb{R}^n$ can be written as

$$v = \sum_{r=1}^{q} \lambda_r d^r, \qquad d^r \in \mathcal{D}, \quad \lambda_r \geq 0.$$

- Consequently, for every nonzero gradient g, some $d \in \mathcal{D}$ satisfies $g^\top d < 0$, which means d is a descent direction.
- For any basis $\{b_1, \ldots, b_n\}$, its signed basis $\{b_1, -b_1, \ldots, b_n, -b_n\}$ is an example of a positive spanning set.

Conclusions

1. Blockwise Direct Search (BDS) is a substantial improvement over the classical direct search method.
2. BDS is robust under noise without any noise-handling techniques.
3. The BDS solver turns BDS into a practical black-box solver, with mechanisms for acceleration, initial step-size selection, and stopping.
4. Under smooth convex assumptions, BDS has a global convergence guarantee.

Future work

- An optimization agent built around the BDS solver
- Adaptive to structured problems (ℓ_1-regularized, etc.)



BDS on GitHub

- open-source and easy to use
- tested continuously via GitHub Actions
- tested on various platforms

Future work

- An optimization agent built around the BDS solver
- Adaptive to structured problems (ℓ_1-regularized, etc.)



BDS on GitHub

- open-source and easy to use
- tested continuously via GitHub Actions
- tested on various platforms

Thank you!