

# Blackbox Optimization using Mesh Adaptive Direct Search

MADS; method, algorithm and implementations

Ahmed Bayoumy

McGill University

November 29, 2022

# Overview

- What is blackbox optimization?
- Functions differentiability
- Direct search methods
- Mesh adaptive direct search method
- Implementations
- Running example

# Overview

- What is blackbox optimization?
- Functions differentiability
- Direct search methods
- Mesh adaptive direct search method
- Implementations
- Running example

# Blackbox Optimization Problems

We consider blackbox optimization problems:

$$\min_{x \in \Omega} f(x)$$

where evaluations of  $f$  and the functions defining  $\Omega$  are usually the result of a computer code (a blackbox).



Each call to the simulation is time-consuming

# Blackbox Optimization Problems

We consider blackbox optimization problems:

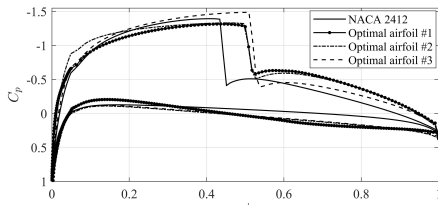
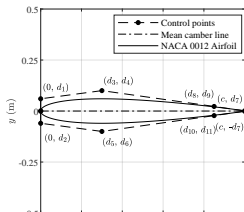
$$\min_{x \in \Omega} f(x)$$

where evaluations of  $f$  and the functions defining  $\Omega$  are usually the result of a computer code (a blackbox).



Each call to the simulation is time-consuming

*Transonic airfoil shape optimization using TSFOIL, SU2, OpenFoam, fluent ...etc.*



# Blackbox Optimization Problems

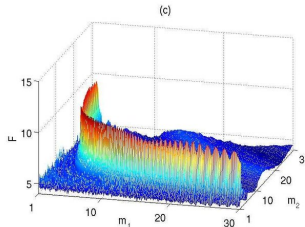
We consider blackbox optimization problems:

$$\min_{x \in \Omega} f(x)$$

where evaluations of  $f$  and the functions defining  $\Omega$  are usually the result of a computer code (a blackbox).



*Derivatives cannot be trusted or evaluated.*



# Blackbox Optimization Problems

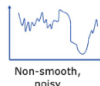
We consider blackbox optimization problems:

$$\min_{x \in \Omega} f(x)$$

where evaluations of  $f$  and the functions defining  $\Omega$  are usually the result of a computer code (a blackbox).



## BBO challenges



Possible bad properties of a blackbox from J. Simonis, 2009, *The Boeing Company*

# Blackbox Optimization Problems

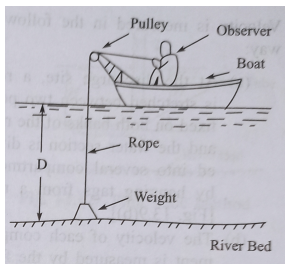
We consider blackbox optimization problems:

$$\min_{x \in \Omega} f(x)$$

where evaluations of  $f$  and the functions defining  $\Omega$  are usually the result of a computer code (a blackbox).

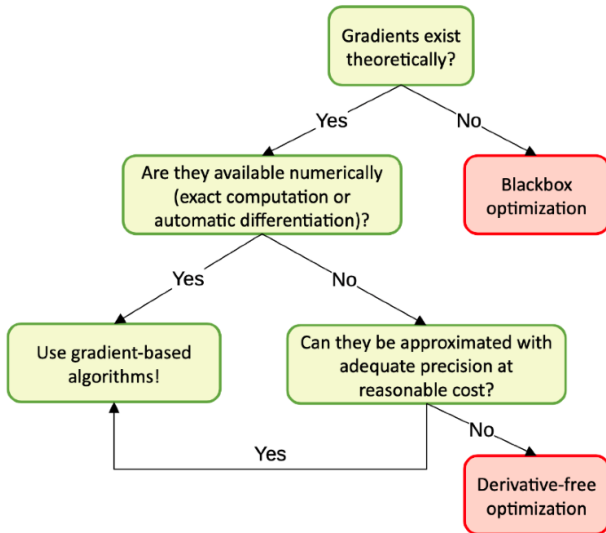


*John Dennis analogy [Powell 1994]*





# Gradients availability



# Blackbox Optimization Problems

## Definition: Blackbox optimization

*It refers to problems where objective and constraint functions cannot be exploited.*

Often the case when their evaluation requires the execution (usually time-consuming) simulation using computational models, typically inaccessible by the user.

# Blackbox Optimization Problems

## Definition: Blackbox optimization

*It refers to problems where objective and constraint functions cannot be exploited.*

Often the case when their evaluation requires the execution (usually time-consuming) simulation using computational models, typically inaccessible by the user.

## Definition: Derivative-free optimization

*It refers to the use of algorithms that utilize only function values because their derivatives are either not defined or not available.*

Gradient approximations may sometimes be obtained, but the amount of work required to ensure they are dependable may not be worth the effort.

# Types of constraints

The domain:  $\Omega = \{x \in X : c_j(x) \leq 0, j \in J\} \subset \mathbb{R}^n$

- The set  $X$  represents *unrelaxable constraints*
- $c_j(x) \leq 0$  are *relaxable constraints*
- *Hidden constraints*: when the simulation fails for points in  $\Omega$

# Properties of a function

- We consider a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ .
- We define some properties of  $f$  at  $x$ , a point of its domain.
- We say that these properties apply near  $x$  if the property is satisfied on some open neighborhood of  $x$ .
- We can also consider some properties on a domain  $\mathcal{X} \subset \mathbb{R}^n$ .
- $f$  is continuous at  $x \in \mathbb{R}^n$  if the limit  $\lim_{y \rightarrow x} f(y)$  exists and is equal to  $f(x)$ .

# Types of variables

The decision variables  $x$  can be any combination of

- Continuous  $\mathbb{R}$
- Integer  $\mathbb{N}$  or  $\mathbb{Z}$
- binary  $\{0, 1\}$
- granular  $\{0, 0.05, 0.10, \dots, 0.95, 1.00\}$
- Categorical  $\{0, 0.05, 0.10, \dots, 0.95, 1.00\}$ 
  - Ex: Hyper-parameter optimization of deep neural network

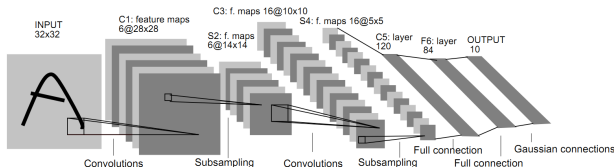


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

- No ordinal property
- The number of convolution layers impacts the total number of optimization variables

- What is blackbox optimization?
- **Functions differentiability**
- Direct search methods
- Mesh adaptive direct search method
- Implementations
- Running example

# Differentiability

Consider the function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ .

- $f$  is **differentiable** at  $x \in \mathbb{R}^n$  if there exists  $g \in \mathbb{R}^n$  such that

$$\lim_{y \rightarrow x} \frac{f(y) - f(x) - g^\top(y - x)}{\|y - x\|} = 0$$

- If this  $g$  exists, it is unique and is called the **gradient** of  $f$  at  $x$ , denoted  $\nabla f(x)$ .
- If  $f$  is differentiable at  $x$ , then  $f$  is continuous at  $x$ .
- **Partial derivatives:**

$$\nabla f(x) = \left( \frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \dots, \frac{\partial f(x)}{\partial x_n} \right)^\top$$



# Differentiability classes

- A function  $f: \mathbb{R}^n \rightarrow \mathbb{R}$  is said **of class  $\mathcal{C}^k$** , denoted  $f \in \mathcal{C}^k$ , with  $0 \leq k \leq \infty$ , if all the possible partial derivatives of the form

$$\frac{\partial^k f}{\partial x_{i_1} \partial x_{i_2} \cdots \partial x_{i_k}}$$

exist and are continuous, where  $i_\ell \in \{1, 2, \dots, n\}$  for all  $\ell \in \{1, 2, \dots, k\}$ .

- $\mathcal{C}^0$ : Continuous functions.
- $\mathcal{C}^1$ : **Continuously differentiable** functions.
- $\mathcal{C}^\infty$ : **Smooth** functions.

# Lipschitz functions

- $f$  is **Lipschitz** on the set  $\mathcal{X} \subset \mathbb{R}^n$  if there exists a scalar  $K > 0$  such that

$$|f(x) - f(y)| \leq K \|x - y\| \text{ for all } x, y \in \mathcal{X}$$

- $K$  is called the **Lipschitz constant**.

- Examples of non-Lipschitz functions:

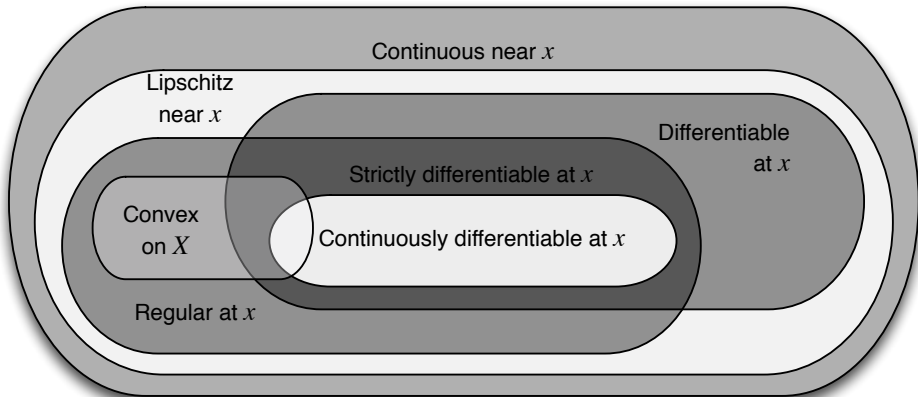
- $f(x) = \sqrt{x}$  for  $x \geq 0$ .
- Discontinuous functions,  $\tan(x)$  for  $x \in (-\pi/2, \pi/2)$ ,  $\frac{1}{x}$  for  $x \in \mathbb{R}$ .
- $f(x) = x^2$  for  $x \in \mathbb{R}$ .

Examples of Lipschitz functions:

- $f(x) = \sqrt{x^2 + 5}$  for  $x \in \mathbb{R}$ ,  $K = 1$ .
- $\sin(x)$  for  $x \in \mathbb{R}$ ,  $K = 1$ .
- $f(x) = |x|$  for  $x \in \mathbb{R}$ ,  $K = 1$ .

## Summary of function types

$f : X \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$  is...



## Convergence analysis (1/2)

- An optimization algorithm is not considered **heuristic** when it is backed by a **convergence analysis** which ensures some properties at the resulting solution  $\hat{x}$ .
- This analysis typically depends on some assumptions made about the nature of the problem. For example: differentiability of  $f$ , convexity of  $\Omega$ , etc.
- Usually, these properties are given as **necessary** or **sufficient optimality conditions**.
- Recall that **global convergence** refers to independence of the starting point.

## Convergence analysis (2/2)

- In DFO, we expect global convergence to solutions satisfying some local and necessary optimality conditions, when the function is supposed Lipschitz.
- However, a blackbox has no exploitable property and cannot be proven Lipschitz.
- **But** consider the following choice between two algorithms to apply to such a problem:
  - Algorithm  $\mathcal{A}$  is a heuristic; it **may** yield a point  $\hat{x}$  where  $\nabla f(\hat{x}) \neq 0$  when  $f$  is differentiable.
  - Algorithm  $\mathcal{B}$  guarantees  $\nabla f(\hat{x}) = 0$  when  $f$  is differentiable.

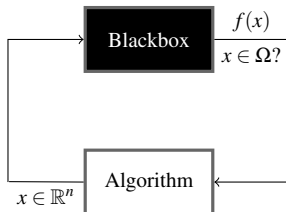
The choice is obvious.

- What is blackbox optimization?
- Functions differentiability
- **Direct search methods**
- Mesh adaptive direct search method
- Implementations
- Running example

## General scheme

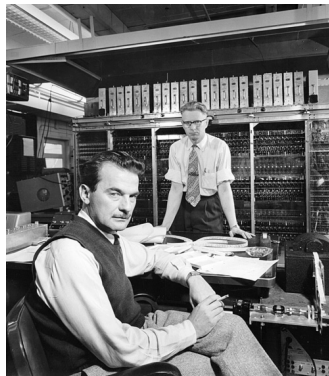
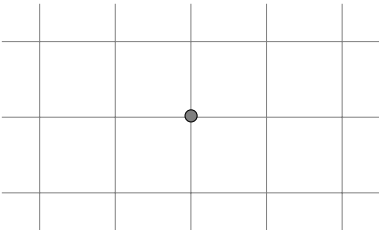
Direct search methods launch the blackbox simulation at tentative trial points in hopes of improving the current best solution.

\* The way that the trial points are generated defines the method



# Unconstrained optimization - Coordinate search

1952 Fermi and Metropolis  
Coordinate Search algorithm.

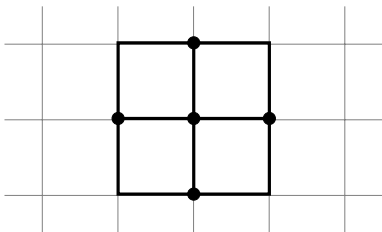


Metropolis and Richardson in front of the MANIAC computer  
Source <http://www.ominous-valve.com/maniac.html>



# Unconstrained optimization - Coordinate search

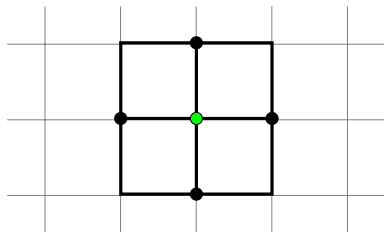
1952 Fermi and Metropolis  
Coordinate Search algorithm.



- **INITIALIZATION:**  
 $x_0$  : starting point in  $\mathbb{R}^n$   
 $\Delta_0 > 0$  : initial step size
- **POLL STEP:** for  $k = 0, 1, \dots$   
if  $f(t) < f(x_k)$  for  $t \in P_k := \{x_k \pm \Delta_k e_i : i = 1, 2, \dots, n\}$ :  
     $x_{k+1} \leftarrow t$   
     $\Delta_{k+1} \leftarrow \Delta_k$   
else (failure):  $x_k$  is a local minimum relatively to  $P_k$ :  
     $x_{k+1} \leftarrow x_k$   
     $\Delta_{k+1} \leftarrow \Delta_k/2$

# Unconstrained optimization - Coordinate search

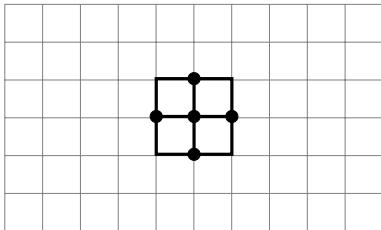
1952 Fermi and Metropolis  
Coordinate Search algorithm.



- **INITIALIZATION:**  
 $x_0$  : starting point in  $\mathbb{R}^n$   
 $\Delta_0 > 0$  : initial step size
- **POLL STEP:** for  $k = 0, 1, \dots$   
if  $f(t) < f(x_k)$  for  $t \in P_k := \{x_k \pm \Delta_k e_i : i = 1, 2, \dots, n\}$ :  
     $x_{k+1} \leftarrow t$   
     $\Delta_{k+1} \leftarrow \Delta_k$   
else (failure):  $x_k$  is a local minimum relatively to  $P_k$ :  
     $x_{k+1} \leftarrow x_k$   
     $\Delta_{k+1} \leftarrow \Delta_k/2$

# Unconstrained optimization - Coordinate search

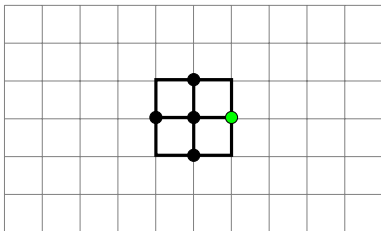
1952 Fermi and Metropolis  
Coordinate Search algorithm.



- **INITIALIZATION:**  
 $x_0$  : starting point in  $\mathbb{R}^n$   
 $\Delta_0 > 0$  : initial step size
- **POLL STEP:** for  $k = 0, 1, \dots$   
if  $f(t) < f(x_k)$  for  $t \in P_k := \{x_k \pm \Delta_k e_i : i = 1, 2, \dots, n\}$ :  
     $x_{k+1} \leftarrow t$   
     $\Delta_{k+1} \leftarrow \Delta_k$   
else (failure):  $x_k$  is a local minimum relatively to  $P_k$ :  
     $x_{k+1} \leftarrow x_k$   
     $\Delta_{k+1} \leftarrow \Delta_k/2$

# Unconstrained optimization - Coordinate search

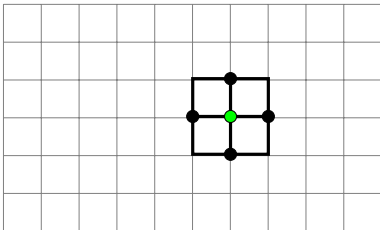
1952 Fermi and Metropolis  
Coordinate Search algorithm.



- **INITIALIZATION:**  
 $x_0$  : starting point in  $\mathbb{R}^n$   
 $\Delta_0 > 0$  : initial step size
- **POLL STEP:** for  $k = 0, 1, \dots$   
if  $f(t) < f(x_k)$  for  $t \in P_k := \{x_k \pm \Delta_k e_i : i = 1, 2, \dots, n\}$ :  
     $x_{k+1} \leftarrow t$   
     $\Delta_{k+1} \leftarrow \Delta_k$   
else (failure):  $x_k$  is a local minimum relatively to  $P_k$ :  
     $x_{k+1} \leftarrow x_k$   
     $\Delta_{k+1} \leftarrow \Delta_k/2$

# Unconstrained optimization - Coordinate search

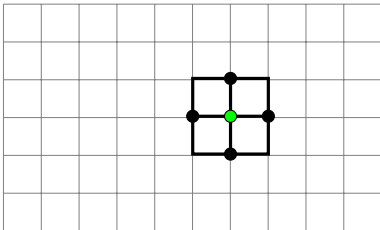
1952 Fermi and Metropolis  
Coordinate Search algorithm.



- **INITIALIZATION:**  
 $x_0$  : starting point in  $\mathbb{R}^n$   
 $\Delta_0 > 0$  : initial step size
- **POLL STEP:** for  $k = 0, 1, \dots$   
if  $f(t) < f(x_k)$  for  $t \in P_k := \{x_k \pm \Delta_k e_i : i = 1, 2, \dots, n\}$ :  
     $x_{k+1} \leftarrow t$   
     $\Delta_{k+1} \leftarrow \Delta_k$   
else (failure):  $x_k$  is a local minimum relatively to  $P_k$ :  
     $x_{k+1} \leftarrow x_k$   
     $\Delta_{k+1} \leftarrow \Delta_k/2$

# Unconstrained optimization - Coordinate search

1952 Fermi and Metropolis  
Coordinate Search algorithm.



- **INITIALIZATION:**  
 $x_0$  : starting point in  $\mathbb{R}^n$   
 $\Delta_0 > 0$  : initial step size
- **POLL STEP:** for  $k = 0, 1, \dots$   
if  $f(t) < f(x_k)$  for  $t \in P_k := \{x_k \pm \Delta_k e_i : i = 1, 2, \dots, n\}$ :  
     $x_{k+1} \leftarrow t$   
     $\Delta_{k+1} \leftarrow \Delta_k$   
else (failure):  $x_k$  is a local minimum relatively to  $P_k$ :  
     $x_{k+1} \leftarrow x_k$   
     $\Delta_{k+1} \leftarrow \Delta_k/2$

- **Remarks**

- Evaluation of  $x_0$  is counted
- Extension with bounds: if the algorithm generates a trial point outside of the boundary, then do not evaluate and consider as failure
- If the algorithm generates a trial point  $x$  found in the cache, then do not evaluate and consider as failure

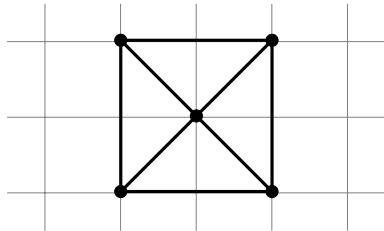
## CS variants

Complete			Opportunist			Ordered			
$k$	trial point $t^\dagger$	$f(t)$	$k$	trial point $t^\dagger$	$f(t)$	$k$	trial point $t^\dagger$	$f(t)$	
0	[3, 2]	29286	0	[3, 2]	29286	0	[3, 2]	29286	
	[2, 3]	4772		[2, 3]	4772		[2, 3]	4772	
	$x^1 = [1, 2]$	166		$x^1 = [1, 2]$	166		$x^1 = [1, 2]$	166	
1	[2, 1]	4176	1	[2, 2]	4401	1	$x^2 = [0, 2]$	81	
	[2, 2]	4401		[1, 3]	262		[−1, 2]	2646	
	[1, 3]	262		$x^2 = [0, 2]$	81	[0, 3]	152		
	$x^2 = [0, 2]$	81	2	[1, 2]	166	[1, 2]	166		
	[1, 1]	106		[0, 3]	152	$x^3 = [0, 1]$	36		
2	[1, 2]	166	2	[−1, 2]	2646	3	$x^4 = [0, 0]$	17	
	[0, 3]	152		$x^3 = [0, 1]$	36		[0, −1]	24	
	[−1, 2]	2646		3	[1, 1]	106	[0, 1]	36	
	$x^3 = [0, 1]$	36	[0, 2]		81	[1, 0]	82		
3	[1, 1]	106	3	[−1, 1]	2466	4	[−1, 0]	2402	
	[0, 2]	81		$x^4 = [0, 0]$	17		[0, −0.5]	17.25	
	[−1, 1]	2466		4	[1, 0]		82	[0, 0.5]	23.25
$x^4 = [0, 0]$	17	[0, 1]	36		$x^6 = [0.5, 0]$	1.0625			
4	[1, 0]	82	4	[−1, 0]	2402	6	[1, 0]	82	
	[0, 1]	36		[0, −1]	24		$x^7 = [0.5, −0.5]$	0.375	
	[−1, 0]	2402		5	$x^6 = [0.5, 0]$	1.0625	7	[0.5, −1]	4.3125
	[0, −1]	24			[1, 0]	82		[1, −0.5]	83.5
Incumbent solutions after a total of 100 function evaluations									
[0.398, −0.332] $2.2 \times 10^{-5}$			[0.3999, −0.3330] $9.8 \times 10^{-7}$			[0.4000, −0.3334] $1.7 \times 10^{-8}$			

# CS evolution

1961 Hooke and Jeeves - Pattern Search  
1997 Torczon - Generalized Pattern Search

Mesh coarsens on successful iterations



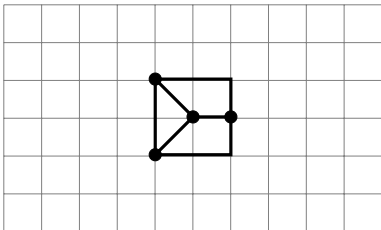


# CS evolution

1961 Hooke and Jeeves - Pattern Search

1997 Torczon - Generalized Pattern Search

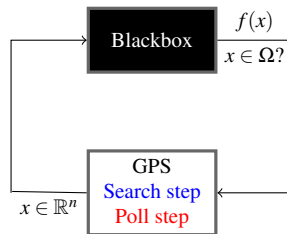
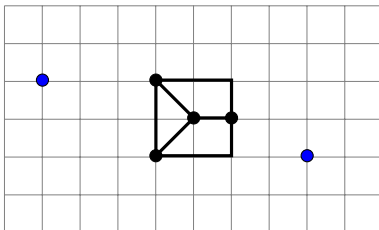
Mesh refines on unsuccessful iterations



# CS evolution

1961 Hooke and Jeeves - Pattern Search

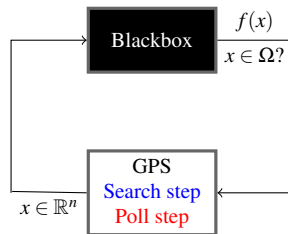
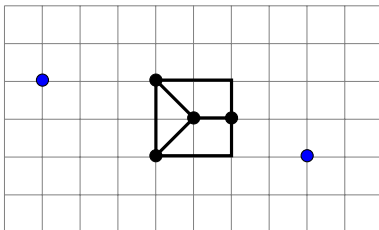
1997 Torczon - Generalized Pattern Search



# CS evolution

1961 Hooke and Jeeves - Pattern Search

1997 Torczon - Generalized Pattern Search



## Theorem (Convergence analysis 2003 - Clark calculus)

*Let  $\hat{x}$  be an accumulation point of the sequence of mesh local optimizers on meshes that get infinitely fine.*

*If  $f$  is Lipschitz near  $\hat{x}$ , then  $f^\circ(\hat{x}; d) \geq 0$  for all directions  $d$  used infinitely often.*

*The set of such directions forms a positive basis for  $\mathbb{R}^n$*

**[0] Initializations** ( $x_0, \Delta_0$ )**[1] Iteration  $k$** **[1.1] (global) Search**

select a finite number of mesh points  
sort these points  
evaluate candidates opportunistically

**[1.2] (local) Poll** (if the Search failed)

construct poll set  $P_k = \{x_k + \Delta_k d : d \in D_k\}$   
sort( $P_k$ )  
evaluate candidates opportunistically

**[2] Updates**

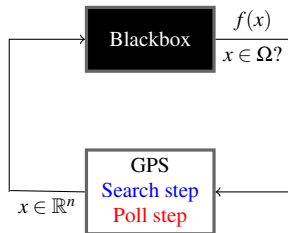
if success

$x_{k+1} \leftarrow$  success point  
possibly increase  $\Delta_k$

else

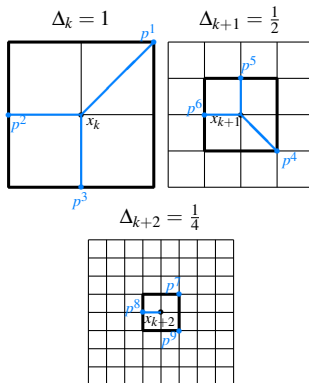
$x_{k+1} \leftarrow x_k$   
decrease  $\Delta_k$

$k \leftarrow k + 1$ , stop or go to [1]



# GPS: Example of poll directions

$P_k = \{x_k + \Delta_k d : d \in D_k\}$ ;  $n + 1$  mesh points at distance  $\Delta_k$  from  $x_k$



14 different ways of defining  $D_k$  on this mesh

- What is blackbox optimization?
- Functions differentiability
- Direct search methods
- **Mesh adaptive direct search method**
- Implementations
- Running example

# The MADS algorithm

---

## MADS main algorithm

---

### 1: Initialization

$x^0 \in \mathbb{R}^n$       initial point  
 $\Delta^0 \in (0, \infty)$       initial frame size parameter

### 2: Parameter Update

set the mesh size parameter to  $\delta^k \leq \Delta^k$   
define the mesh  $M^k$

### 3: “Optional” global Search on mesh $M^k$

if  $f(t) < f(x^k)$  for some  $t$  in a finite subset of the mesh  $M^k$   
set  $x^{k+1} \leftarrow t$  and  $\Delta^{k+1} \leftarrow 2\Delta^k$  and go to Step 5.

### 4: Local Poll on Mesh $M^k$

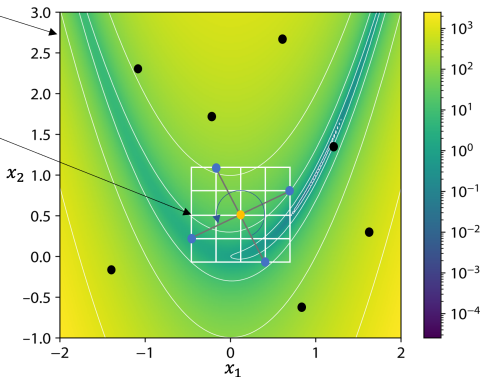
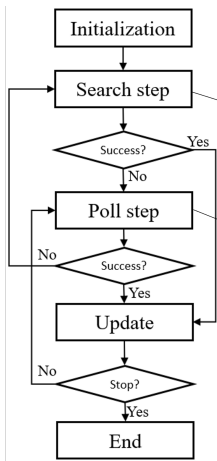
select a positive spanning set  $D_\Delta^k$   
if  $f(t) < f(x^k)$  for some  $t \in P^k = \{x^k + \delta^k d : d \in D_\Delta^k\}$  in a finite subset of the mesh  $M^k$   
set  $x^{k+1} \leftarrow t$  and  $\Delta^{k+1} \leftarrow 2\Delta^k$   
else set  $x^{k+1} \leftarrow x^k$  and  $\Delta^{k+1} \leftarrow \frac{1}{2}\Delta^k$

### 5: Termination check

otherwise  $k \leftarrow k + 1$  and return to Step 2.

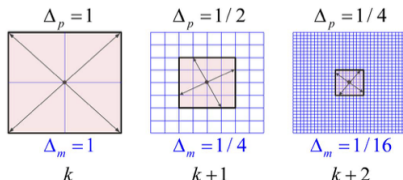
---

# MADS algorithm workflow





# Mesh and frame definitions



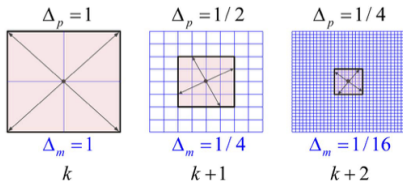
## Definition: Mesh and Mesh Size Parameter

Let  $G \in \mathbb{R}^{n \times n}$  be an invertible matrix and the columns of  $Z \in \mathbb{Z}^{n \times p}$  form a positive spanning set for  $\mathbb{R}^n$ . Define  $D = GZ$ . The mesh of coarseness  $\delta^k > 0$  generated by  $D$ , centered at the incumbent solution  $x_k \in \mathbb{R}^n$ , is defined by  $M^k := \{x^k + \delta^k D_y : y \in \mathbb{N}^p \subset \mathbb{R}^n, \}$  where  $\delta^k$  is called the mesh size parameter.

## Definition: Frame and Frame Size Parameter

Let  $G \in \mathbb{R}^{n \times n}$  be an invertible matrix and the columns of  $Z \in \mathbb{Z}^{n \times p}$  form a positive spanning set for  $\mathbb{R}^n$ . Define  $D = GZ$ . Select a mesh size parameter  $\delta^k > 0$  and let  $\Delta^k$  be such that  $\delta^k \leq \Delta^k$ . The frame of extent  $\Delta^k$  generated by  $D$ , centered at the incumbent solution  $x_k \in \mathbb{R}^n$ , is defined by  $F^k := \{x \in M^k : \|x - x^k\|_\infty \leq \Delta^k b\}$  with  $b = \max\{\|d'\|_\infty : d' \in \mathbb{D}\}$  and  $\Delta^k$  is called the frame size parameter.

## Dense (rich) set of polling directions



### Definition: Asymptotically Dense

The set  $V \subseteq \mathbb{R}^n$  is said to be asymptotically dense if the normalised set  $\{v/\|v\| : v \in V\}$  is dense on the *unit sphere*  $S = \{w \in \mathbb{R}^n : |w| = 1\}$ .

### Definition: Householder Matrix

Let  $V \subseteq \mathbb{R}^n$  be a *normalized vector*. The Householder matrix  $H$  associated with  $v$  is  $H := I - 2vv^T \in \mathbb{R}^{n \times n}$ , where  $I$  is the  $n \times n$  *identity matrix*.

- What is blackbox optimization?
- Functions differentiability
- Direct search methods
- Mesh adaptive direct search method
- **Implementations**
- Running example

## MADS implementations (1/2)

- MADS is a general framework. It defines the conditions on the directions, but do not define the direction themselves
- There are several implementations:
  - **LT-MADS**: Based on Lower-Triangular random matrices
  - **QR-MADS**: Based on the QR decomposition and on normally distributed directions
  - **OrthoMADS**: Quasi-random, deterministic, and orthogonal directions. Current default in NOMAD

## MADS implementations (2/2)

- Several programs that implement MADS are available:
  - **MATLAB:** NOMADm: [https://github.com/khbalhandawi/MECH559\\_notebooks/blob/master/MATLAB\\_Algorithms/NomadM/nomadm.m](https://github.com/khbalhandawi/MECH559_notebooks/blob/master/MATLAB_Algorithms/NomadM/nomadm.m)
  - **Python:** OMADS: <https://ahmed-bayoumy.github.io/OMADS/>
  - **C++ (with interfaces to various languages):** NOMAD:  
<https://nomad-4-user-guide.readthedocs.io/en/latest/index.html>  
This implements the state-of-the-art in MADS development.

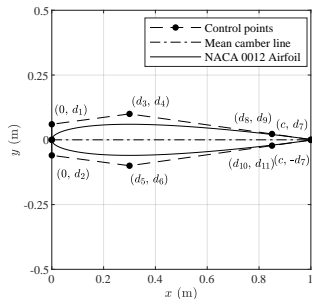
- What is blackbox optimization?
- Functions differentiability
- Direct search methods
- Mesh adaptive direct search method
- Implementations
- Running example

# Transonic airfoil shape optimization

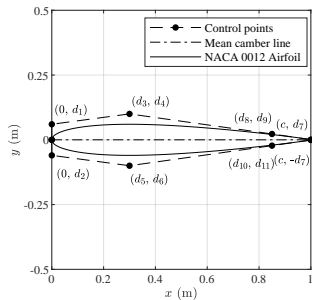
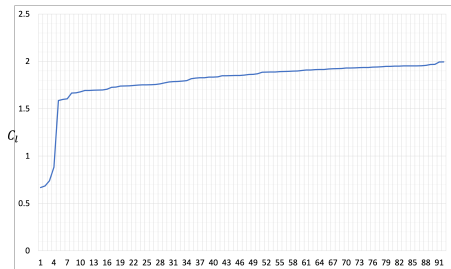
$$\min_{\mathbf{d} \in \Omega} -C_l(\mathbf{d})$$

$$\text{s.t.} \quad 0.075 - A_{cs}(\mathbf{d}) \leq 0,$$

$$C_d(\mathbf{d}) - 0.006 \leq 0$$



# Transonic airfoil shape optimization





# Transonic airfoil shape optimization

