Derivative-Free Optimization: Theory and Practice

Charles Audet, École Polytechnique de Montréal

Luís Nunes Vicente, Universidade de Coimbra

Mini-tutorial – SIOPT 2008 – Boston

May 2008

Presentation Outline

Presentation Outline

1. Introduction
2. Unconstrained optimization
   - Directional direct search methods
   - Simplicial direct search and line-search methods
   - Interpolation-based trust-region methods
3. Optimization under general constraints
   - Nonsmooth optimality conditions
   - MADS and the extreme barrier for closed constraints
   - Filters and progressive barrier for open constraints
4. Surrogates, global DFO, software, and references
   - The surrogate management framework
   - Constrained TR interpolation-based methods
   - Towards global DFO optimization
   - Software and references

Why derivative-free optimization

Some of the reasons to apply Derivative-Free Optimization are the following:

- Growing sophistication of computer hardware and mathematical algorithms and software (which opens new possibilities for optimization).
- Function evaluations costly and noisy (one cannot trust derivatives or approximate them by finite differences).
- Binary codes (source code not available or owned by a company) — making automatic differentiation impossible to apply.
- Legacy codes (written in the past and not maintained by the original authors).
- Lack of sophistication of the user (users need improvement but want to use something simple).

Examples of problems where derivatives are unavailable

Tuning of algorithmic parameters:

- Most numerical codes depend on a number of critical parameters.
- One way to automate the choice of the parameters (to find optimal values) is to solve:

$$\min_{p \in \mathbb{R}^{n_p}} f(p) = CPU(p; solver) \quad \text{s.t.} \quad p \in P,$$

or

$$\min_{p \in \mathbb{R}^{n_p}} f(p) = \#iterations(p; solver) \quad \text{s.t.} \quad p \in P,$$

where n_p is the number of parameters to be tuned and P is of the form $\{p \in \mathbb{R}^{n_p} : \ell \leq p \leq u\}$.

- It is hard to calculate derivatives for such functions f (which are likely noisy and non-differentiable).

Examples of problems where derivatives are unavailable

Automatic error analysis:

- A process in which the computer is used to analyze the accuracy or stability of a numerical computation.
- How large can the growth factor for GE be for a pivoting strategy? Given n and a pivoting strategy, one maximizes the growth factor:

$$\max_{A \in \mathbb{R}^{n \times n}} f(A) = \frac{\max_{i,j,k} |a_{ij}^{(k)}|}{\max_{i,j} |a_{ij}|}.$$

  A starting point could be $A_0 = I_n$.
- When no pivoting is chosen, f is defined and continuous at all points where GE does not break down (possibly non-differentiability).
- For partial pivoting, the function f is defined everywhere (because GE cannot break down) but it can be discontinuous when a tie occurs.

Examples of problems where derivatives are unavailable

Engineering design:

- A case study in DFO is the helicopter rotor blade design problem:
  - The goal is to find the structural design of the rotor blades to minimize the vibration transmitted to the hub. The variables are the mass, center of gravity, and stiffness of each segment of the rotor blade.
  - The simulation code is multidisciplinary, including dynamic structures, aerodynamics, and wake modeling and control.
  - The problem includes upper and lower bounds on the variables, and some linear constraints have been considered such as an upper bound on the sum of masses.
  - Each function evaluation requires simulation and can take from minutes to days of CPU time.
- Other examples are wing platform design, aeroacoustic shape design, and hydrodynamic design.

Examples of problems where derivatives are unavailable

Other known applications:

- Circuit design (tuning parameters of relatively small circuits).
- Molecular geometry optimization (minimization of the potential energy of clusters).
- Groundwater community problems.
- Medical image registration.
- Dynamic pricing.

Limitations of derivative-free optimization

iteration	$\|x_k - x_*\|$
0	1.8284e+000
1	5.9099e-001
2	1.0976e-001
3	5.4283e-003
4	1.4654e-005
5	1.0737e-010
6	1.1102e-016

- Newton methods converge quadratically (locally) but require first and second order derivatives (gradient and Hessian).

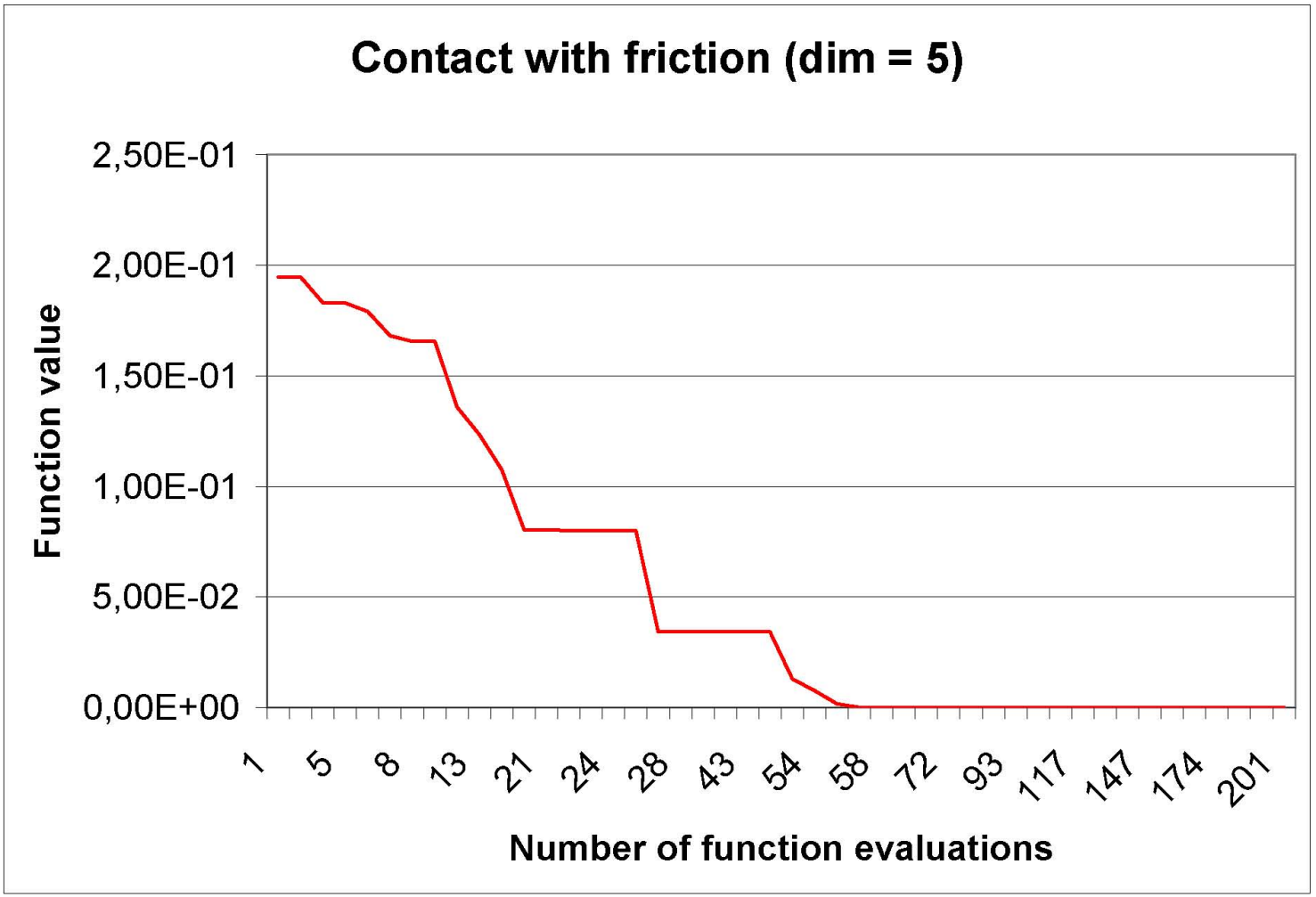
Limitations of derivative-free optimization

iteration	$\|x_k - x_*\|$
0	3.0000e+000
1	2.0002e+000
2	6.4656e-001
⋮	⋮
6	1.4633e-001
7	4.0389e-002
8	6.7861e-003
9	6.5550e-004
10	1.4943e-005
11	8.3747e-008
12	8.8528e-010

- Quasi Newton (secant) methods converge superlinearly (locally) but require first order derivatives (gradient).

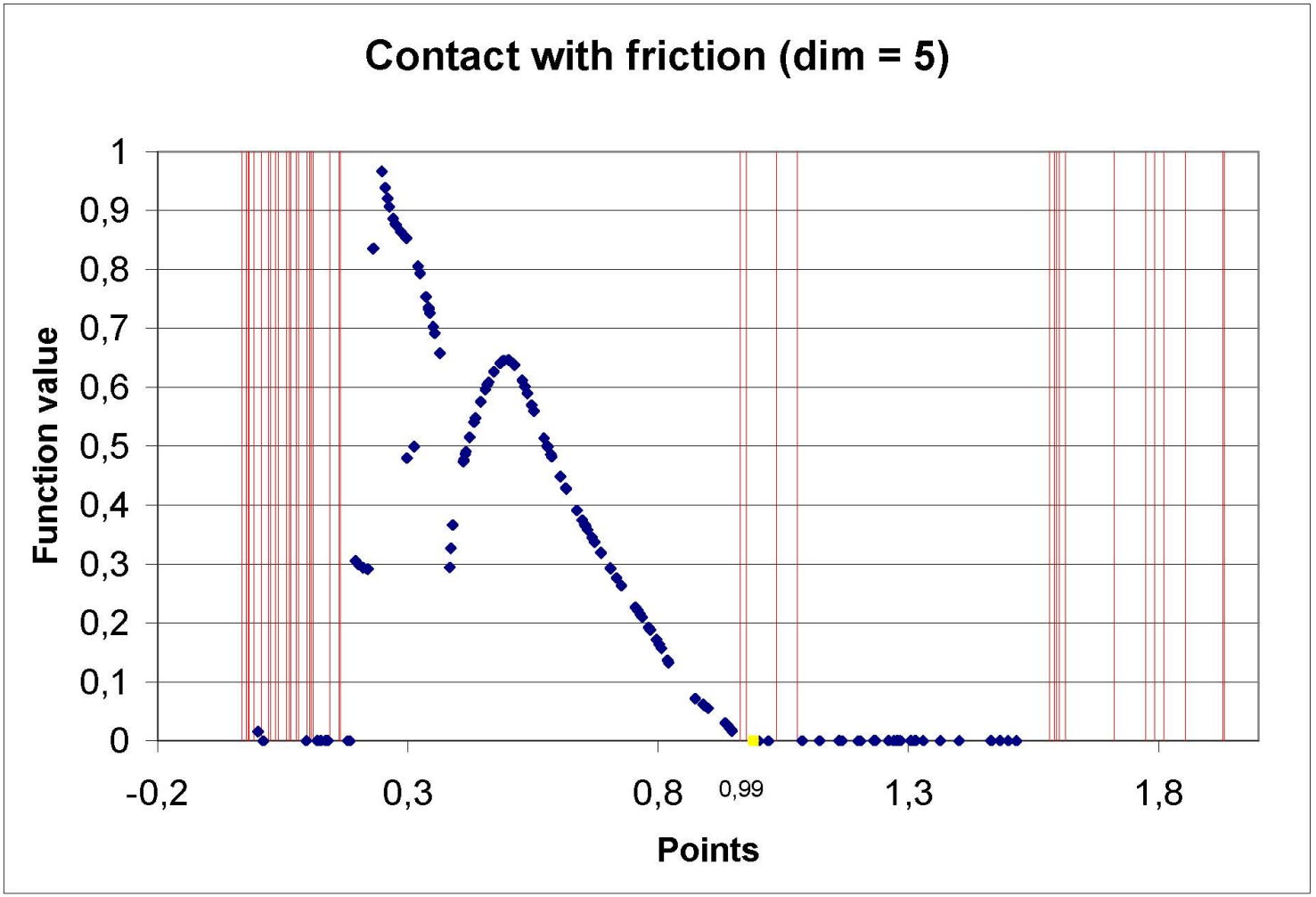
Limitations of derivative-free optimization

In DFO convergence/stopping is typically slow (per function evaluation):



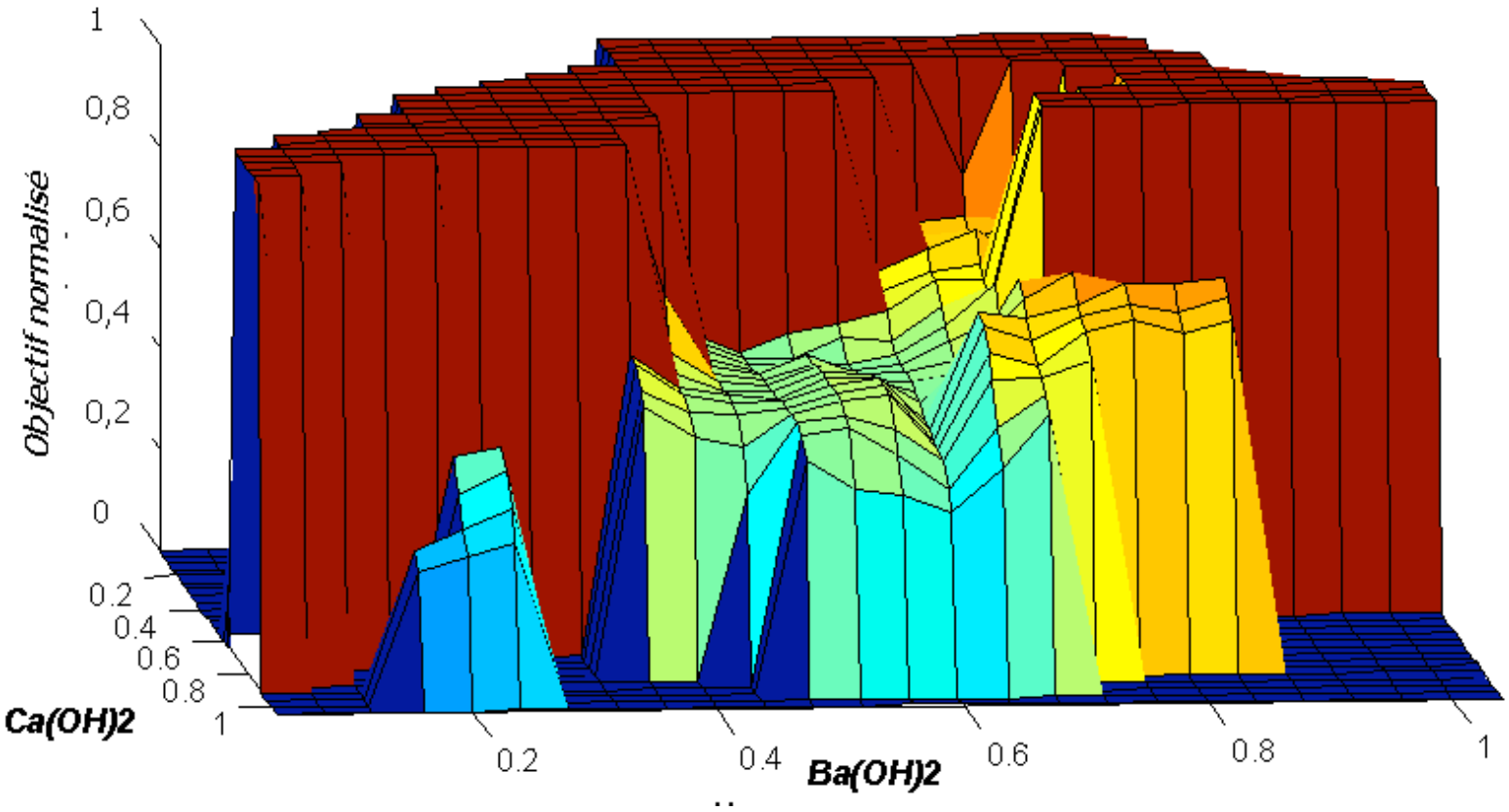
Pitfalls

The objective function might not be continuous or even well defined:



Pitfalls

The objective function might not be continuous or even well defined:



What can we solve

With the current state-of-the-art DFO methods one can expect to successfully address problems where:

- The evaluation of the function is expensive and/or computed with noise (and for which accurate finite-difference derivative estimation is prohibitive and automatic differentiation is ruled out).
- The number of variables does not exceed, say, a hundred (in serial computation).
- The functions are not excessively non-smooth.
- Rapid asymptotic convergence is not of primary importance.
- Only a few digits of accuracy are required.

What can we solve

In addition we can expect to solve problems:

- With hundreds of variables using a parallel environment or exploiting problem information.
- With a few integer or categorical variables.
- With a moderate level of multimodality:

  It is hard to minimize non-convex functions without derivatives.

  However, it is generally accepted that derivative-free optimization methods have the ability to find 'good' local optima.

  DFO methods have a tendency to: (i) go to generally low regions in the early iterations; (ii) 'smooth' the function in later iterations.

Classes of algorithms (globally convergent)

Over-simplifying, all globally convergent DFO algorithms must:

- Guarantee some form of descent away from stationarity.
- Guarantee some control of the geometry of the sample sets where the objective function is evaluated.
- Imply convergence of step size parameters to zero, indicating global convergence to a stationary point.

By global convergence we mean convergence to some form of stationarity from arbitrary starting points.

Classes of algorithms (globally convergent)

(Directional) Direct Search:

- Achieve descent by using positive bases or positive spanning sets and moving in the directions of the best points (in patterns or meshes).
- Examples are coordinate search, pattern search, generalized pattern search (GPS), generating set search (GSS), mesh adaptive direct search (MADS).

Classes of algorithms (globally convergent)

(Simplicial) Direct Search:

- Ensure descent from simplex operations like reflections, by moving in the direction away from the point with the worst function value.
- Examples are the Nelder-Mead method and several modifications to Nelder-Mead.

Classes of algorithms (globally convergent)

Line-Search Methods:

- Aim to get descent along negative simplex gradients (which are intimately related to polynomial models).
- Examples are the implicit filtering method.

Classes of algorithms (globally convergent)

Trust-Region Methods:

- Minimize trust-region subproblems defined by fully-linear or fully-quadratic models (typically built from interpolation or regression).
- Examples are methods based on polynomial models or radial basis functions models.

Presentation outline

Direct search methods

- Use the function values directly.
- Do not require derivatives.
- Do not attempt to estimate derivatives.
- MADS is guaranteed to produce solutions that satisfy hierarchical optimality conditions depending on local smoothness of the functions.
- Examples: DIRECT, MADS, Nelder-Mead, Pattern Search.

Coordinate search (ancestor of pattern search).

Consider the unconstrained problem

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

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$.

Coordinate search (ancestor of pattern search).

Consider the unconstrained problem

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

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$.

- INITIALIZATION:
  x_0 : initial point in $\mathbb{R}^n$ such that $f(x_0) < \infty$
  $\Delta_0 > 0$: initial mesh size.

Coordinate search (ancestor of pattern search).

Consider the unconstrained problem

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

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$.

- INITIALIZATION:
  x_0 : initial point in $\mathbb{R}^n$ such that $f(x_0) < \infty$
  $\Delta_0 > 0$: initial mesh size.
- POLL STEP: for $k = 0, 1, \ldots$
  If $f(t) < f(x_k)$ for some $t \in P_k := \{x_k \pm \Delta_k e_i : i \in N\}$,
  then set $x_{k+1} = t$
  and $\Delta_{k+1} = \Delta_k$;

Coordinate search (ancestor of pattern search).

Consider the unconstrained problem

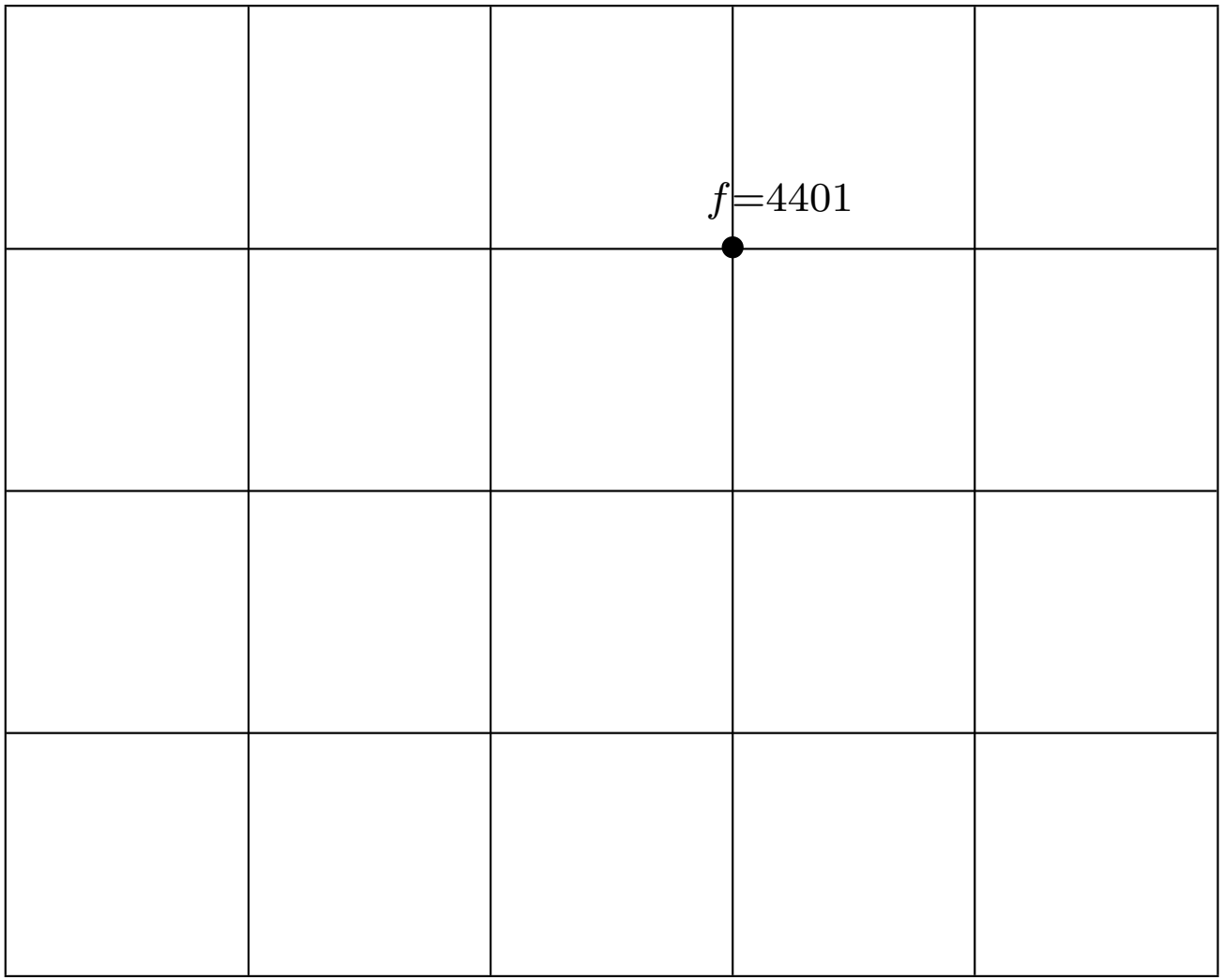
$$\min_{x \in \mathbb{R}^n} \quad f(x)$$

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$.

- INITIALIZATION:
  x_0 : initial point in $\mathbb{R}^n$ such that $f(x_0) < \infty$
  $\Delta_0 > 0$: initial mesh size.
- POLL STEP: for $k = 0, 1, \ldots$
  If $f(t) < f(x_k)$ for some $t \in P_k := \{x_k \pm \Delta_k e_i : i \in N\}$,
  then set $x_{k+1} = t$
  and $\Delta_{k+1} = \Delta_k$;
  otherwise x_k is a local minimum with respect to P_k,
  set $x_{k+1} = x_k$
  and $\Delta_{k+1} = \frac{\Delta_k}{2}$.

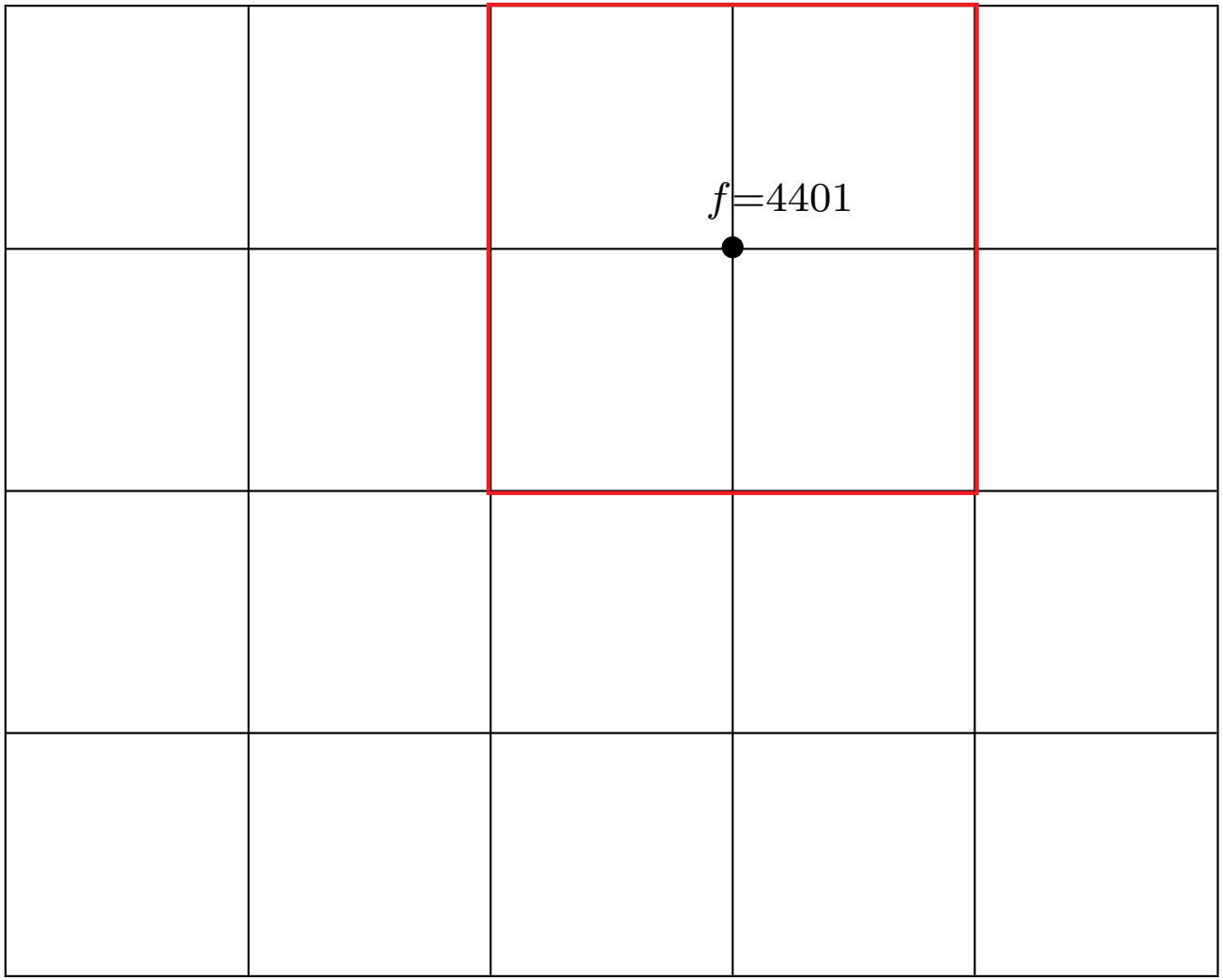
Coordinate search

$x_0=(2,2)^T, \Delta_0=1$



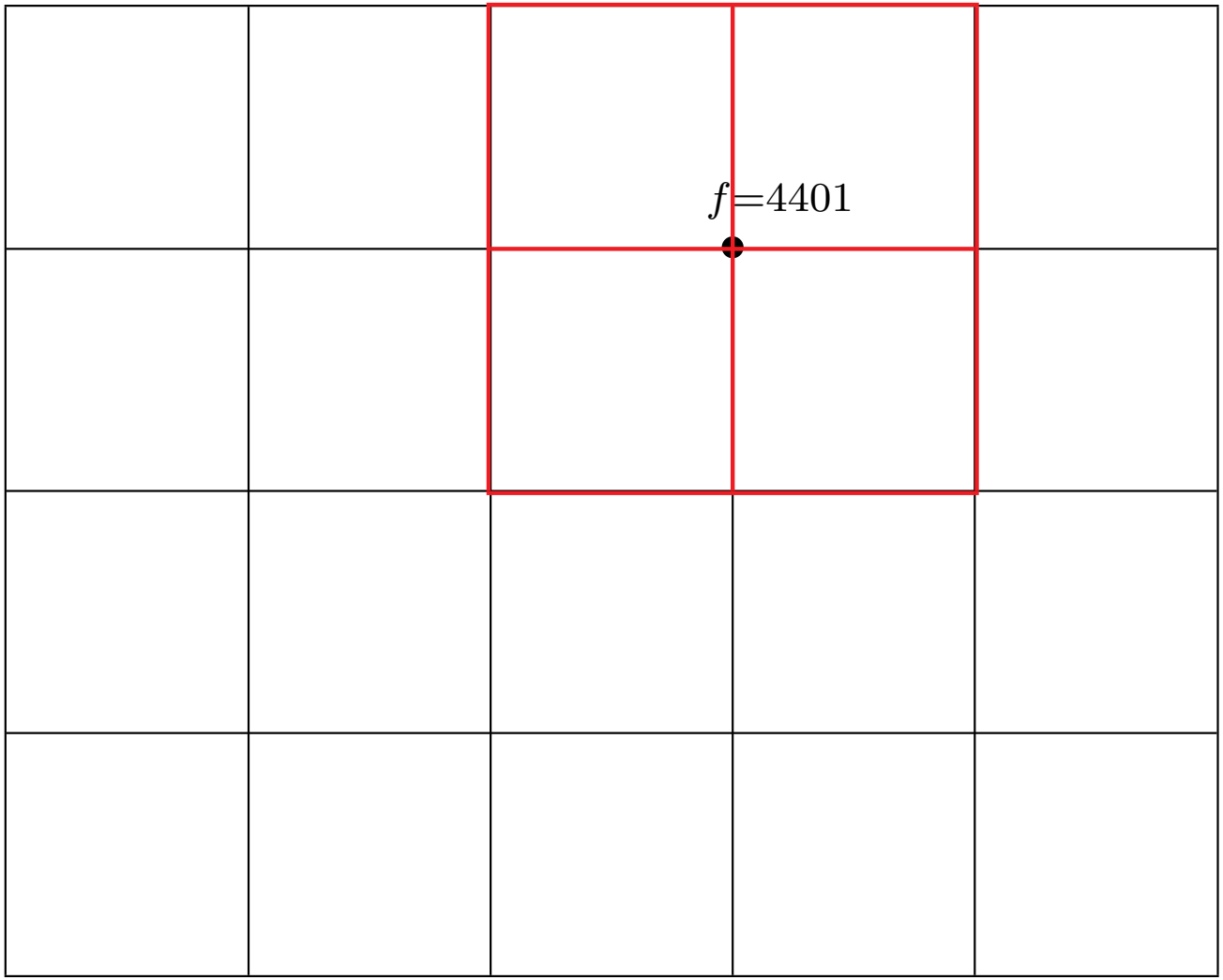
Coordinate search

$x_0=(2,2)^T, \Delta_0=1$



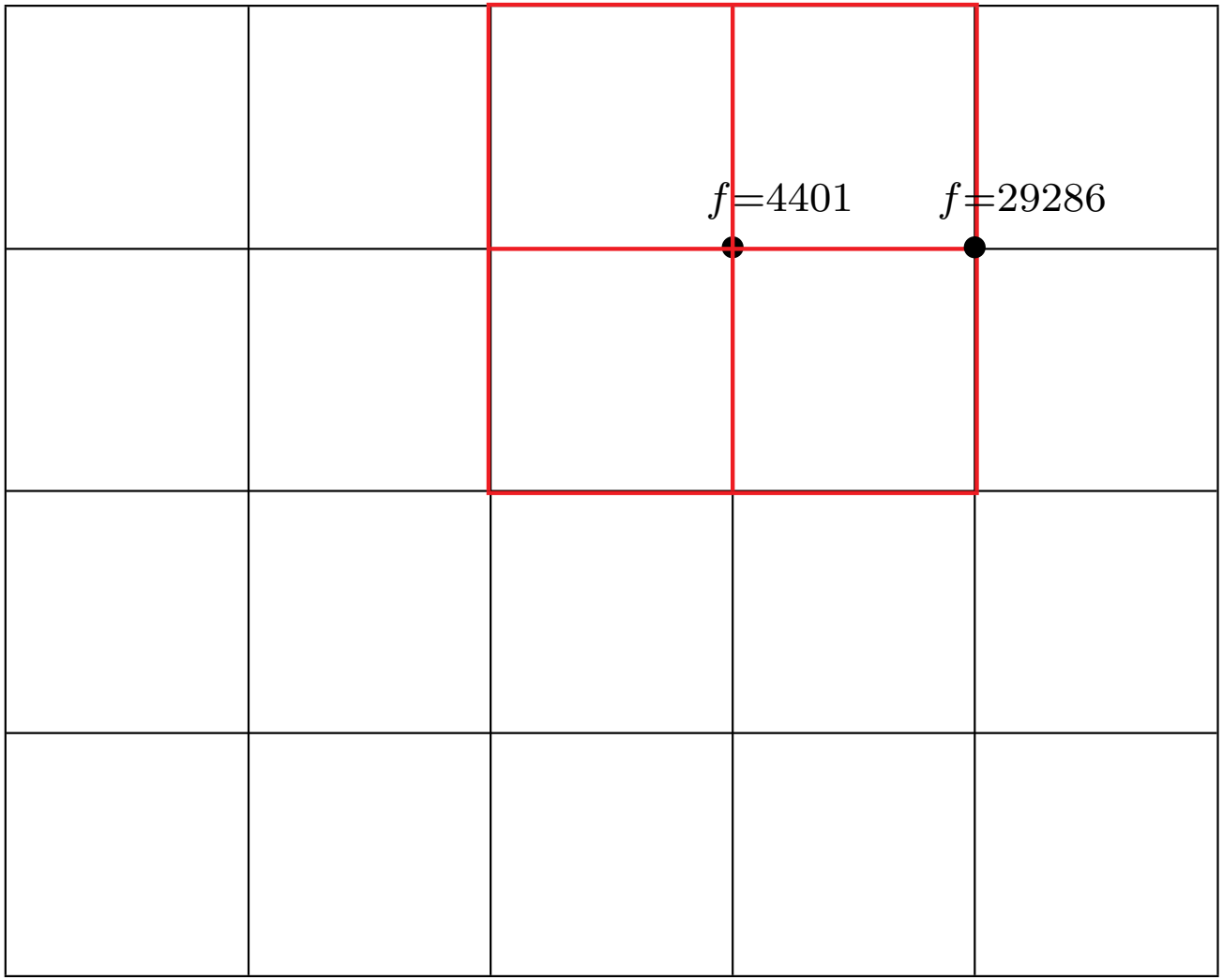
Coordinate search

$x_0=(2,2)^T, \Delta_0=1$



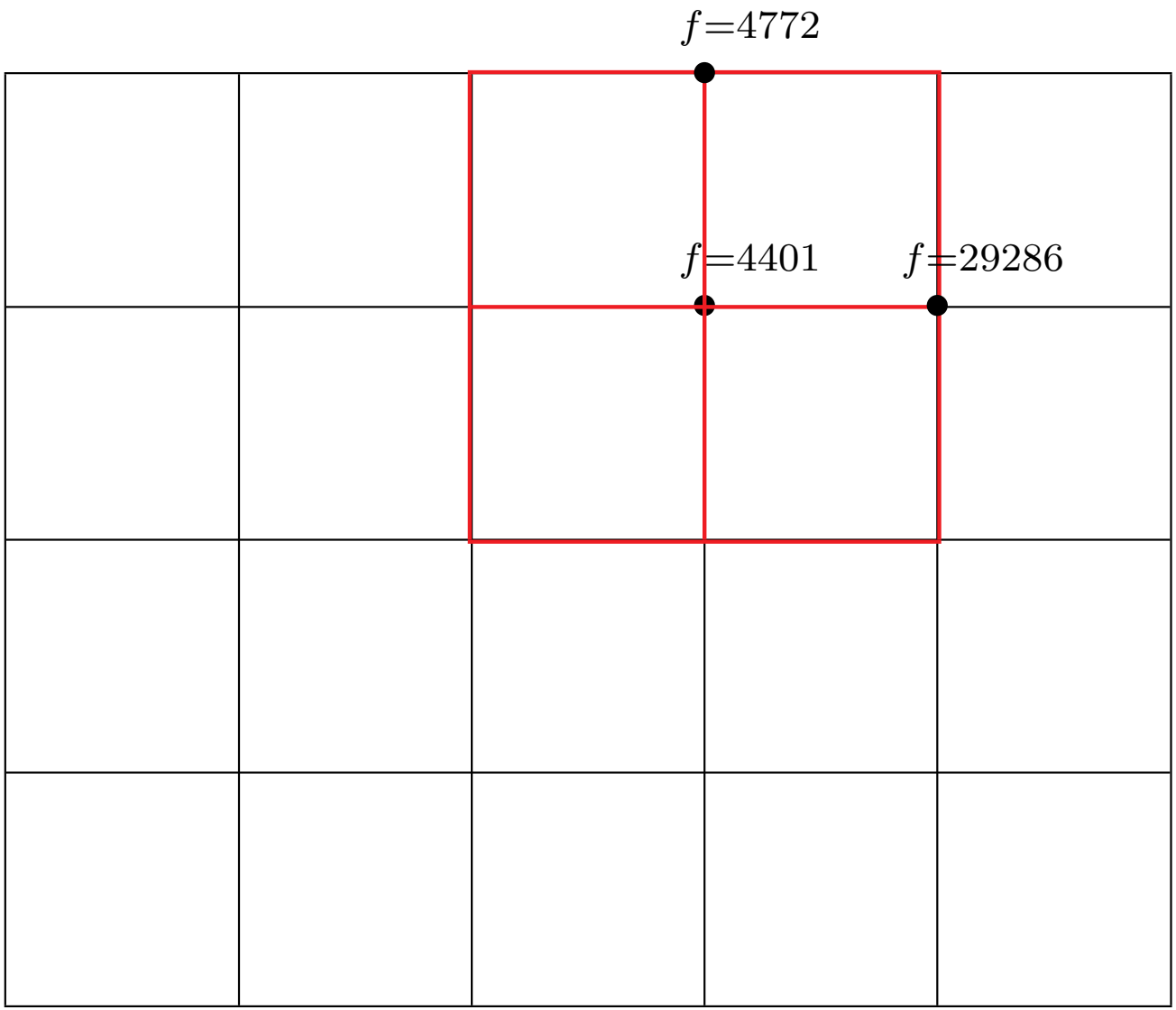
Coordinate search

$x_0=(2,2)^T, \Delta_0=1$



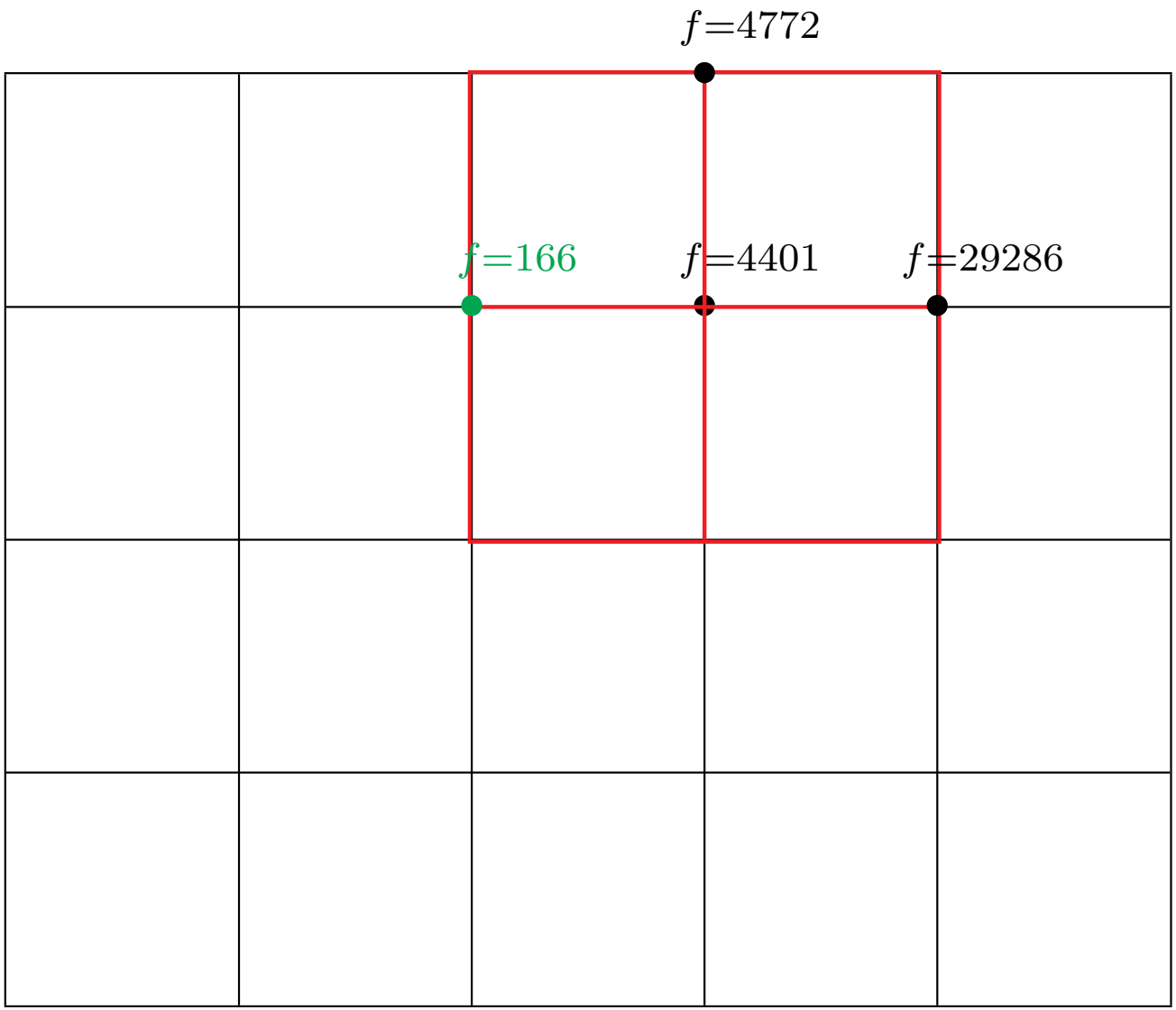
Coordinate search

$x_0=(2,2)^T, \Delta_0=1$



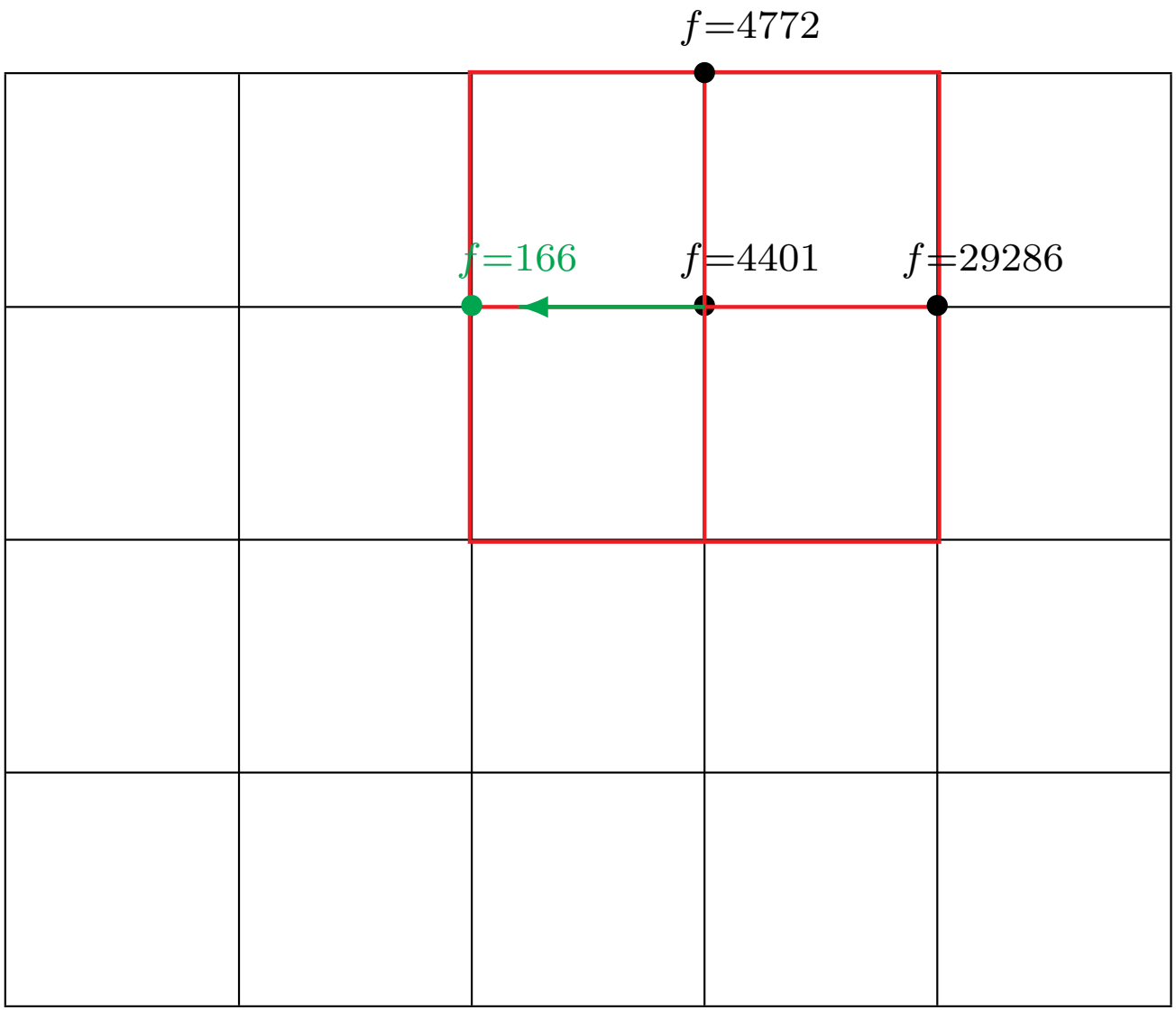
Coordinate search

$x_0=(2,2)^T, \Delta_0=1$



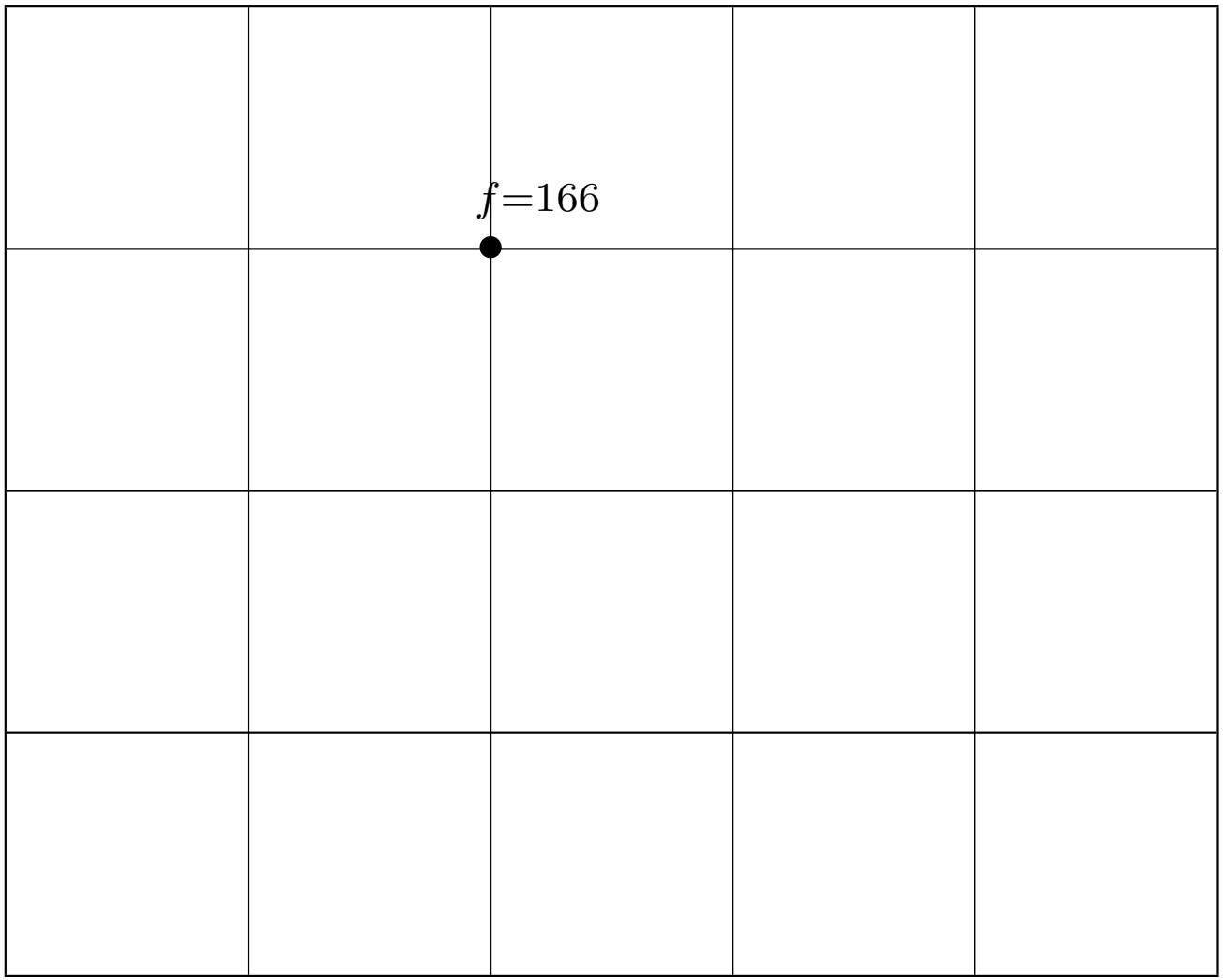
Coordinate search

$x_0=(2,2)^T, \Delta_0=1$



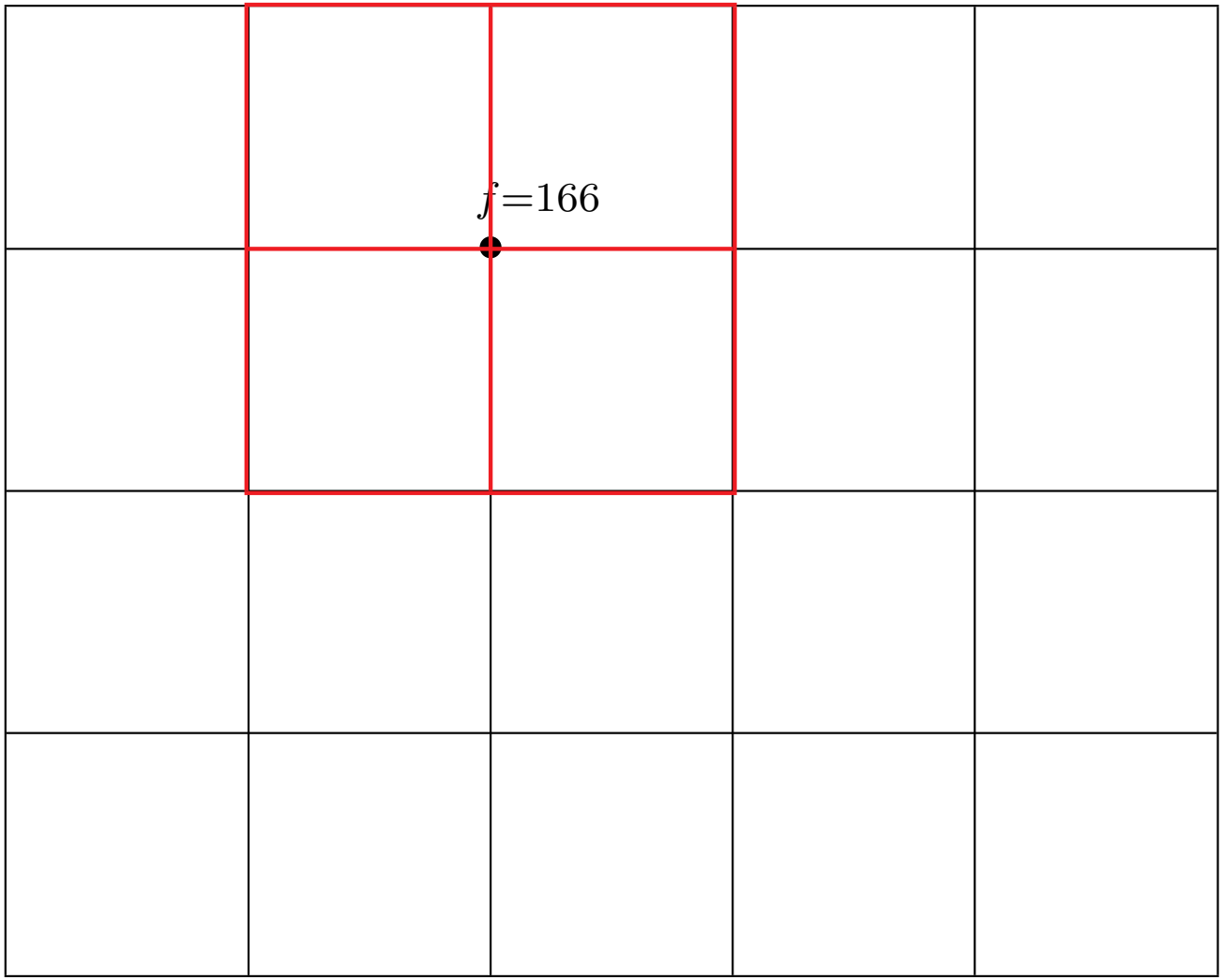
Coordinate search

$x_1=(1,2)^T, \Delta_1=1$



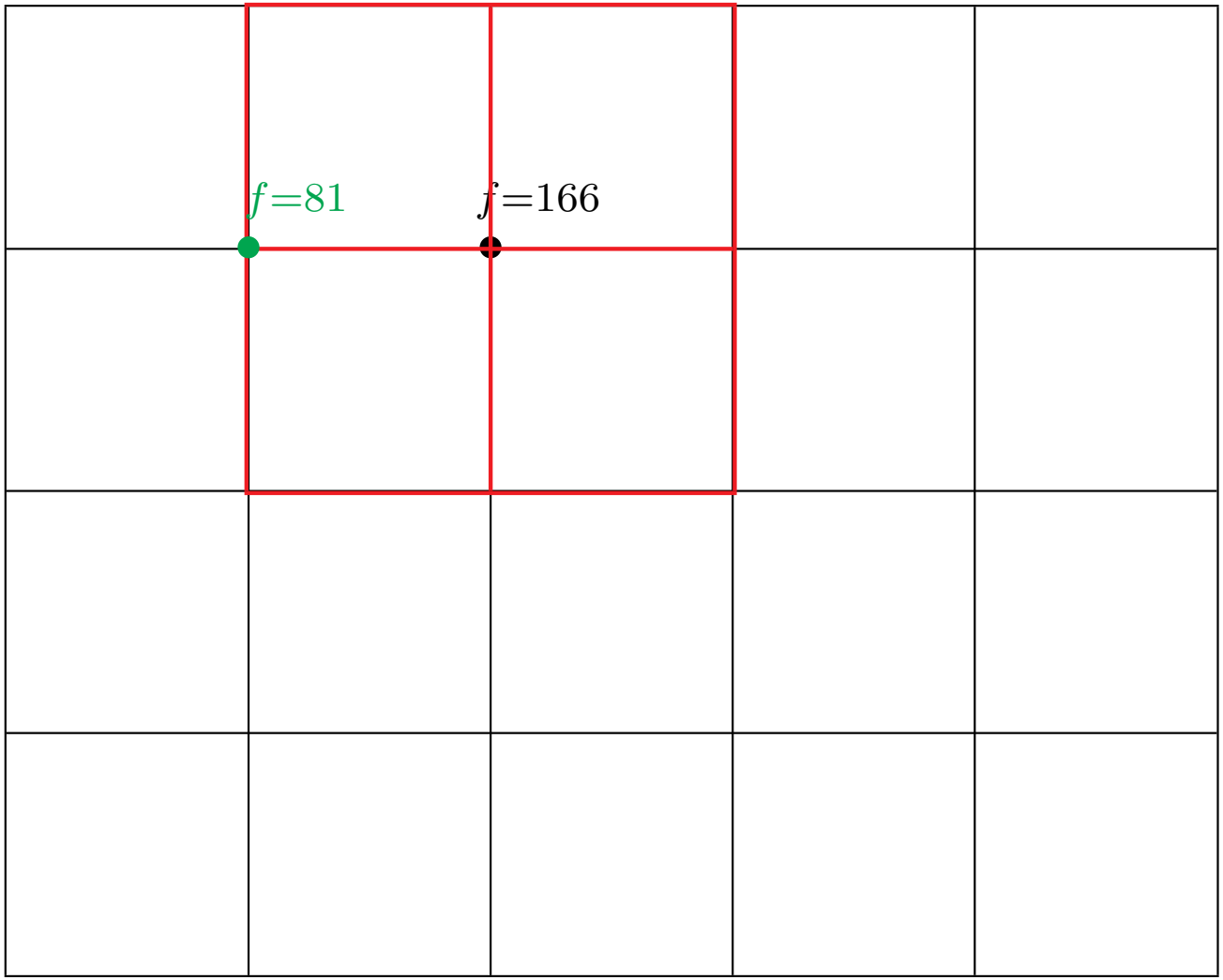
Coordinate search

$x_1=(1,2)^T, \Delta_1=1$



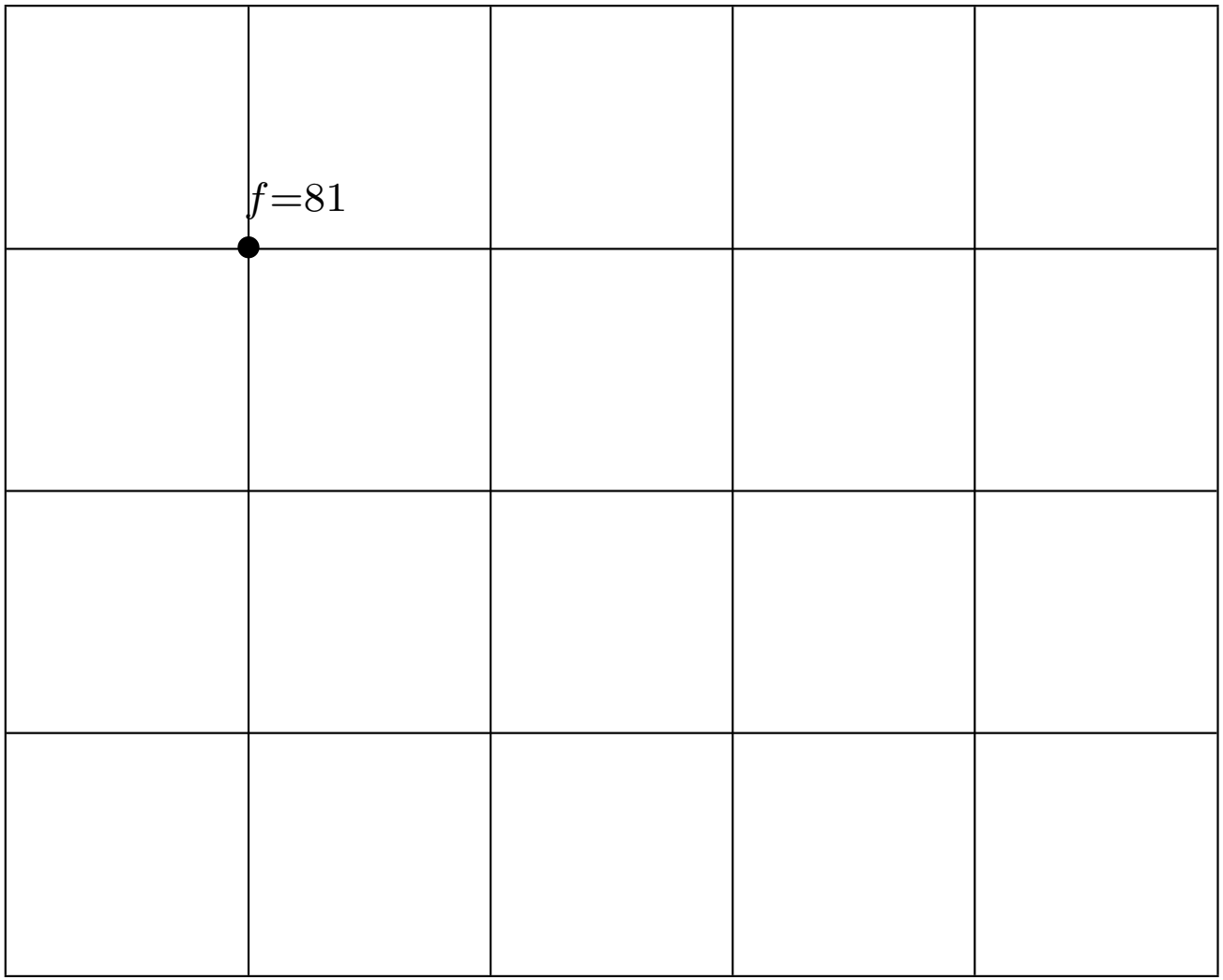
Coordinate search

$x_1=(1,2)^T, \Delta_1=1$

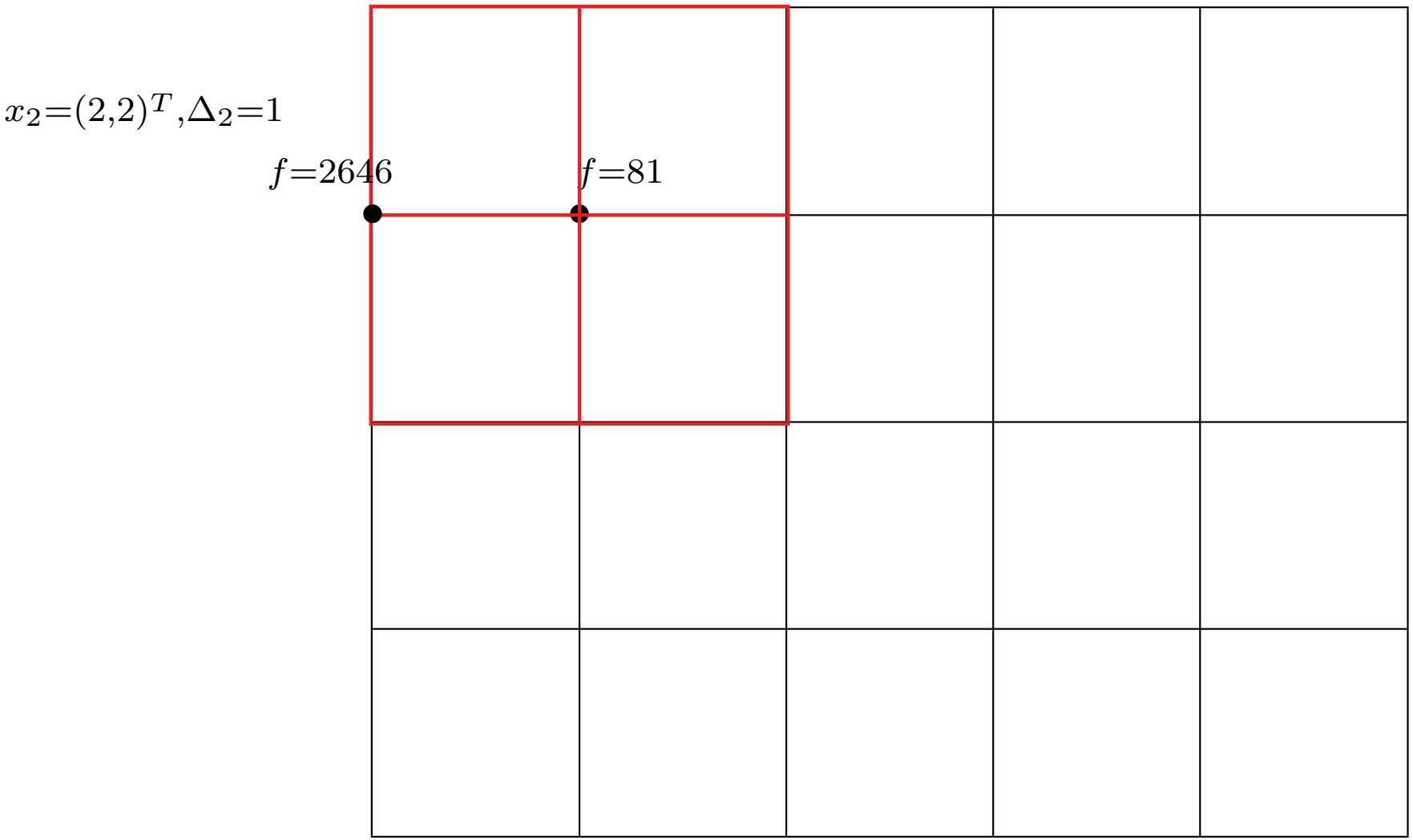


Coordinate search

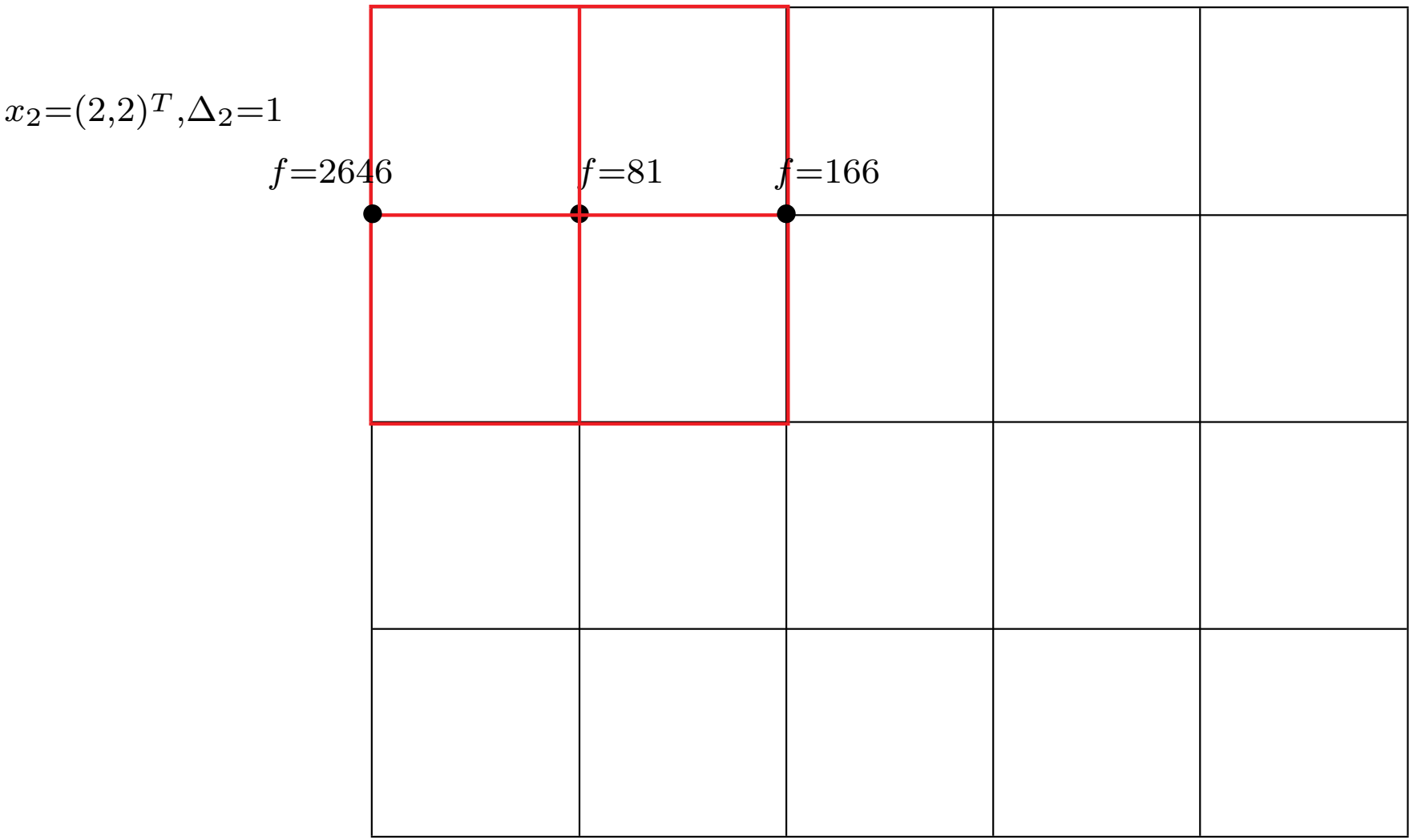
$x_2=(2,2)^T, \Delta_2=1$



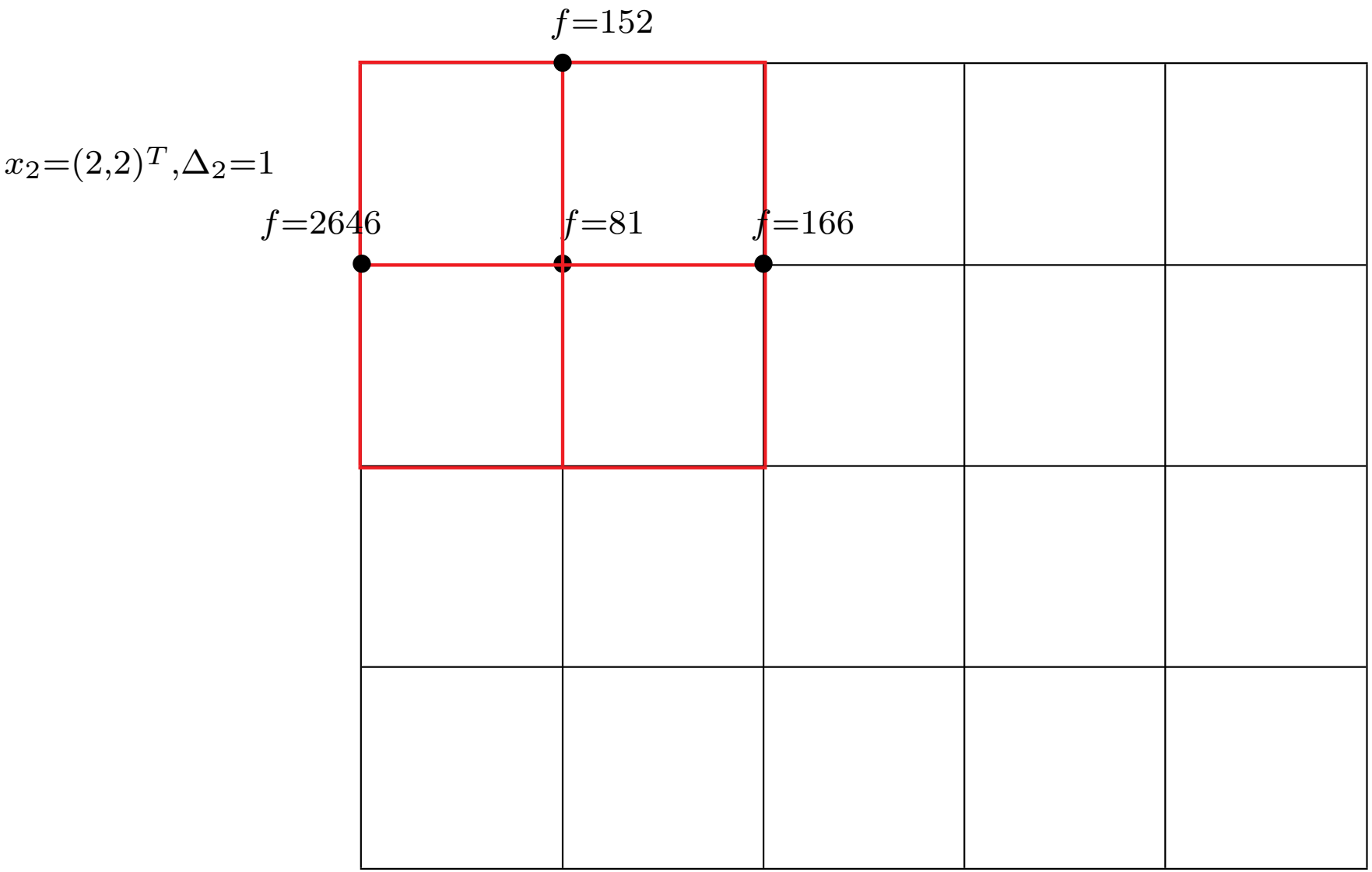
Coordinate search



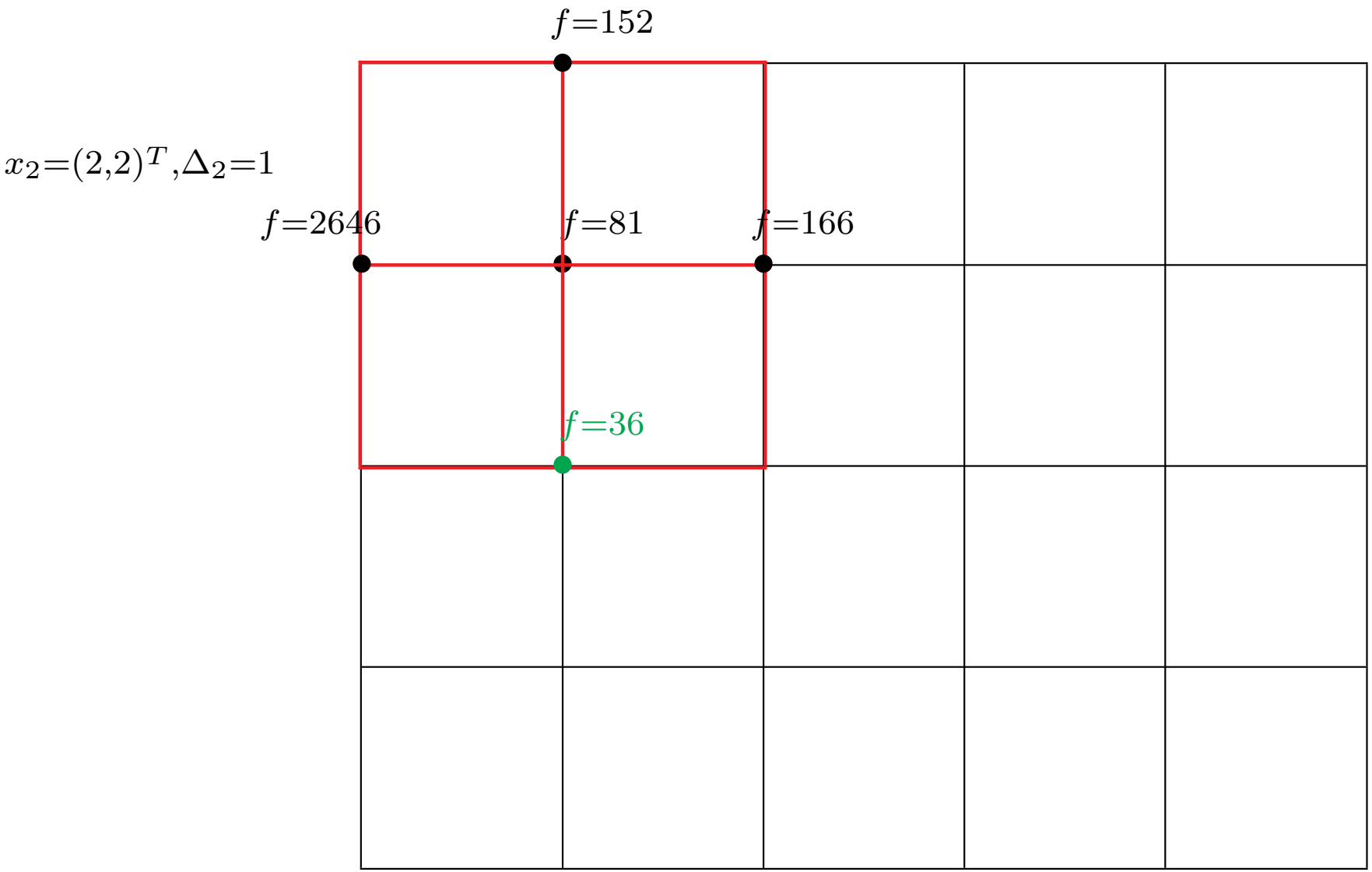
Coordinate search



Coordinate search

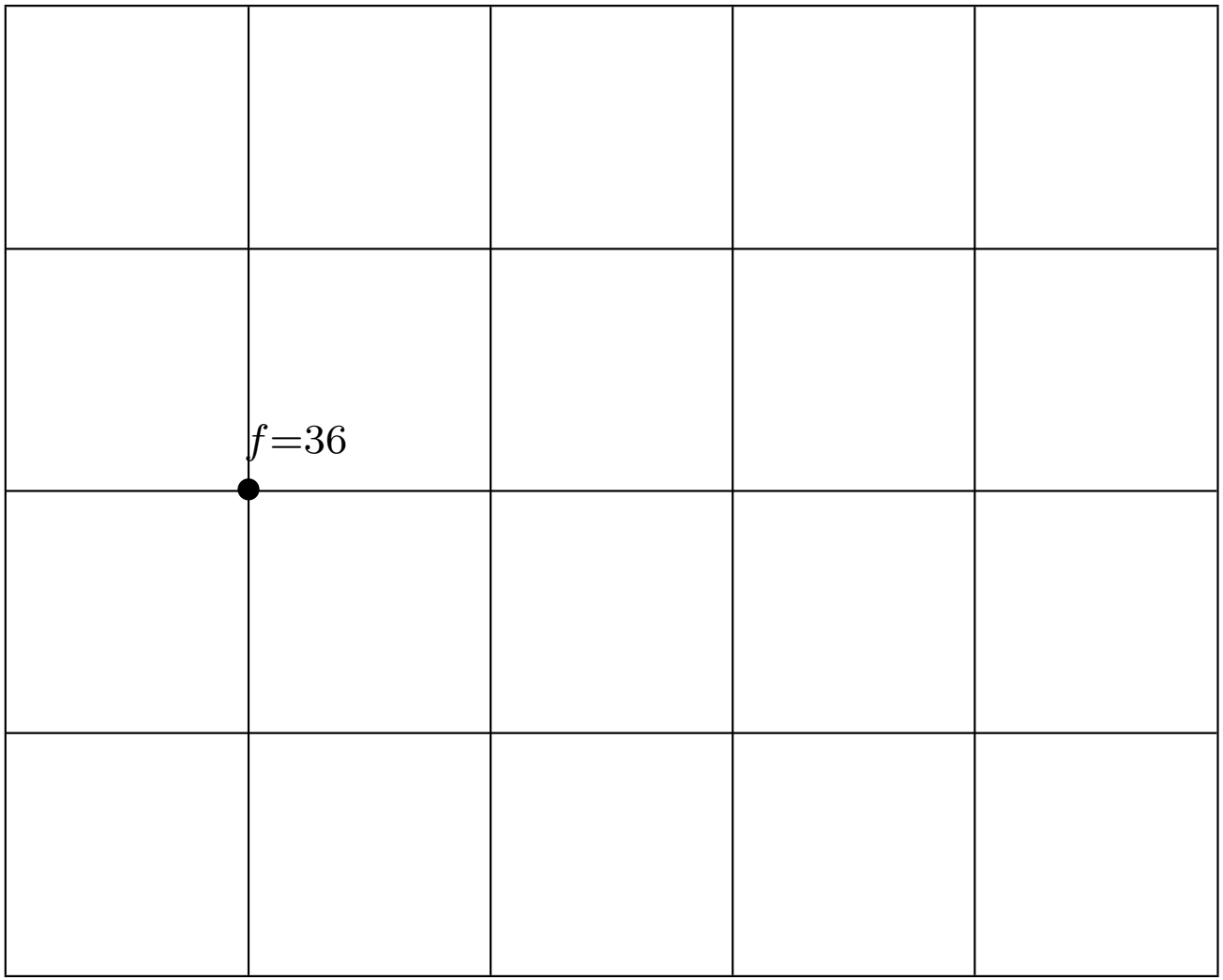


Coordinate search



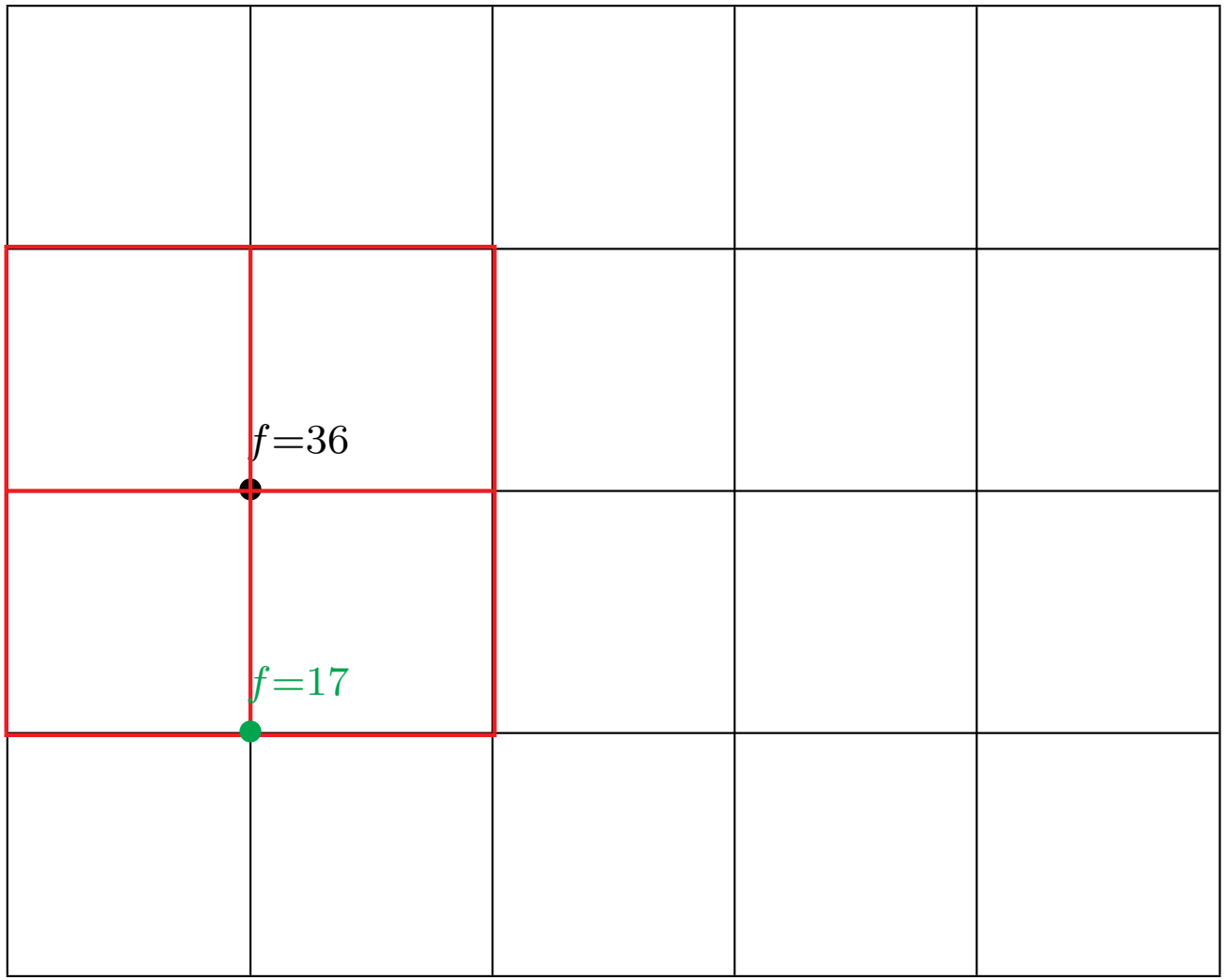
Coordinate search

$x_3=(0,1)^T, \Delta_3=1$



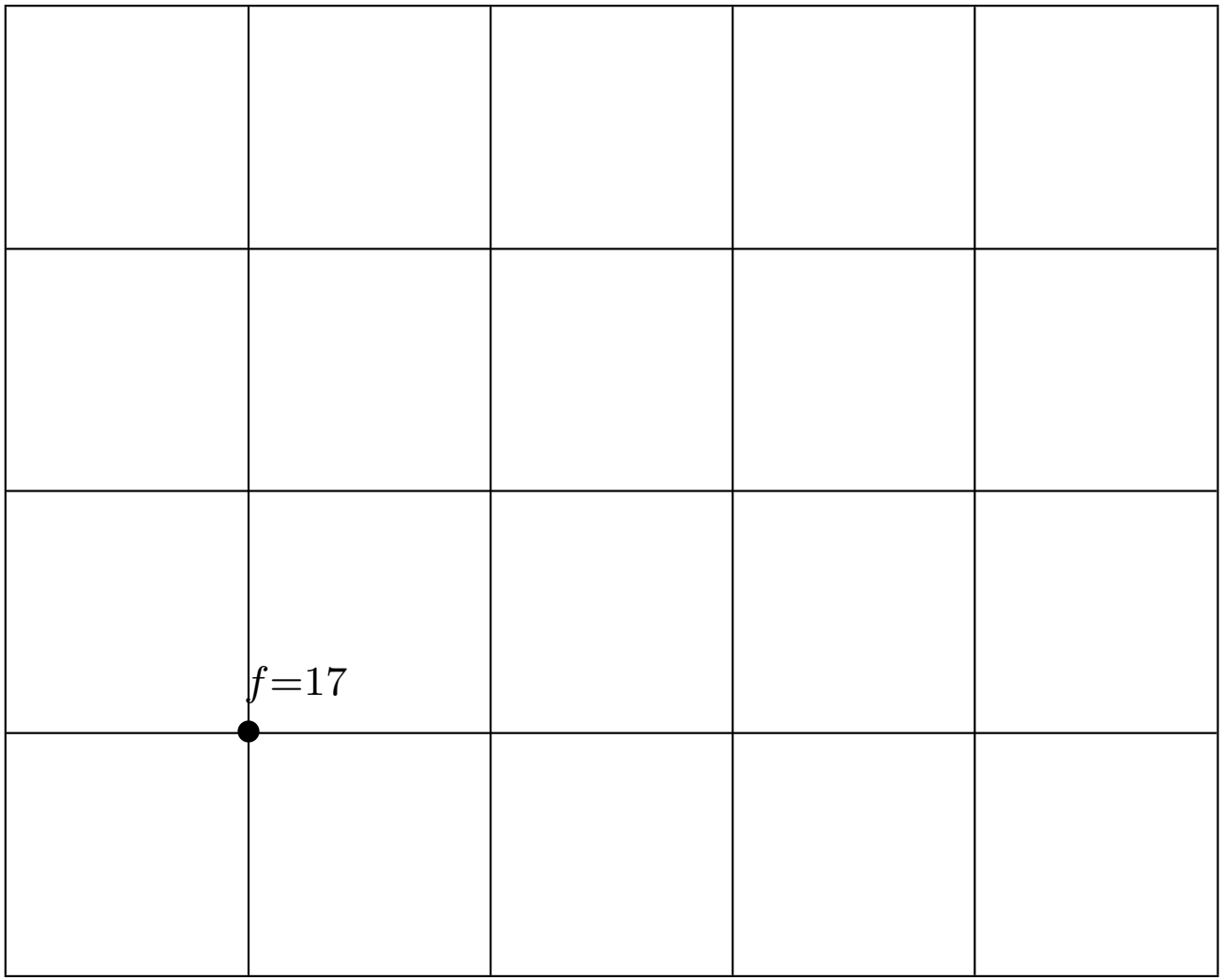
Coordinate search

$x_3=(0,1)^T, \Delta_3=1$

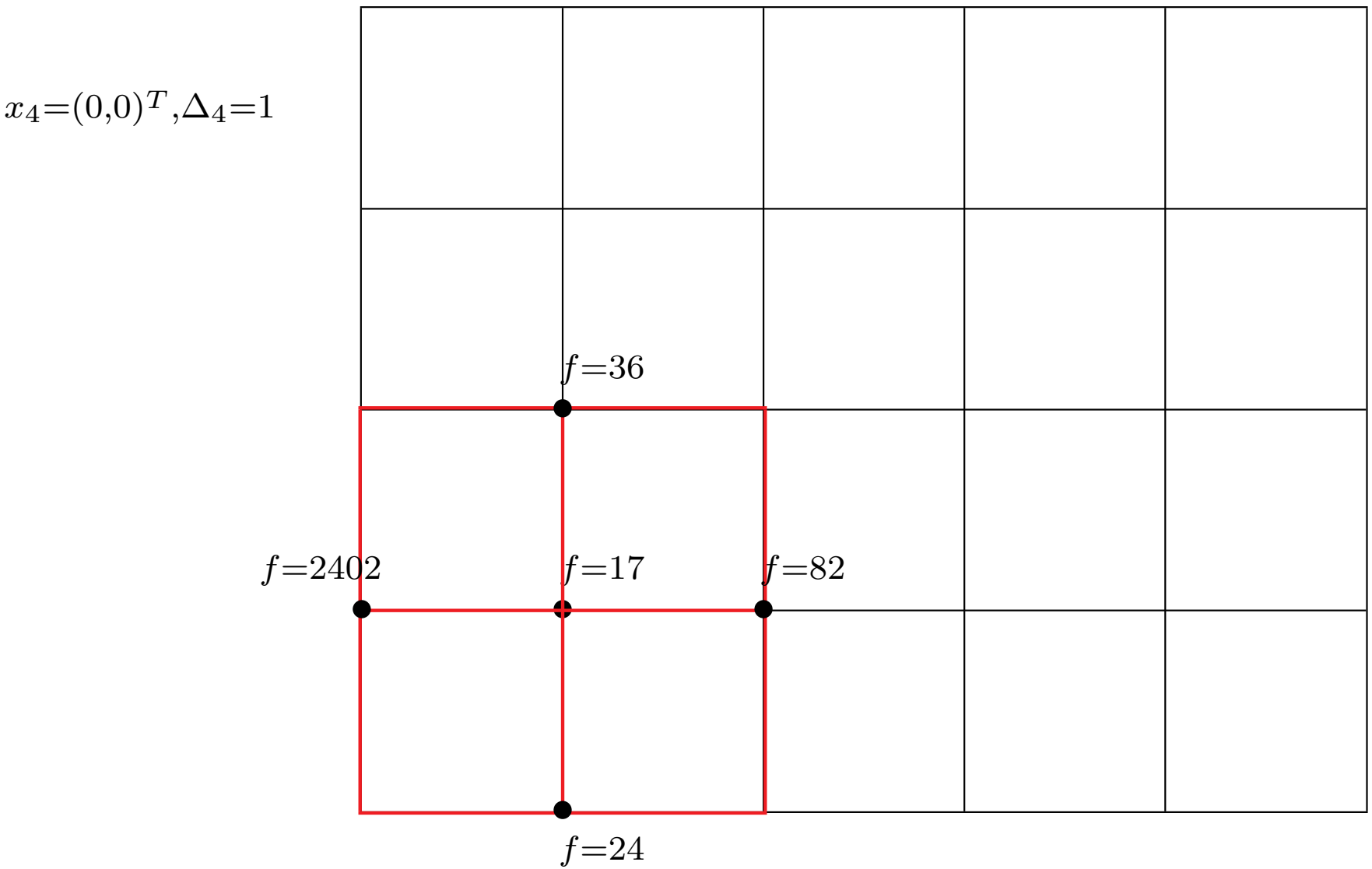


Coordinate search

$x_4=(0,0)^T, \Delta_4=1$

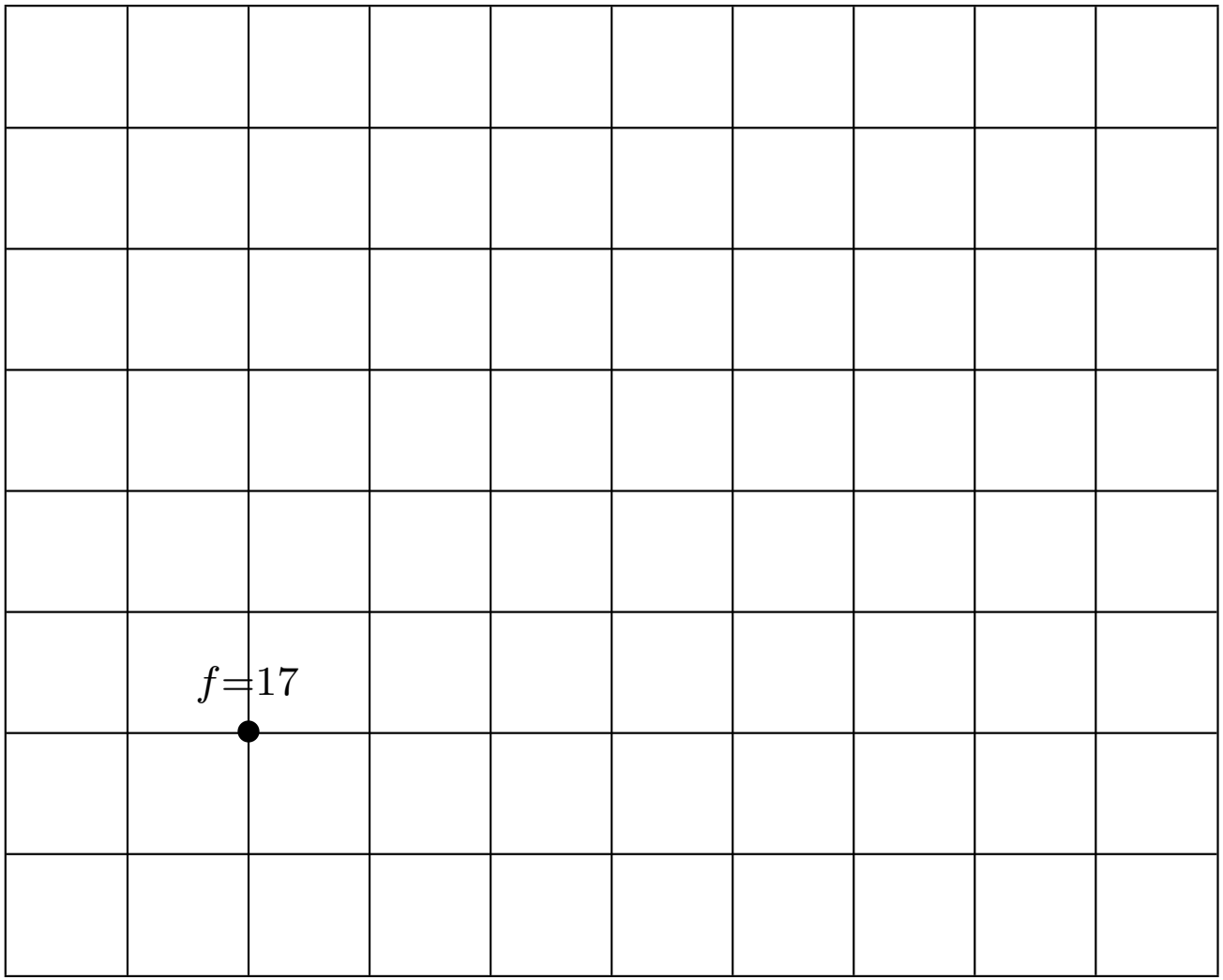


Coordinate search



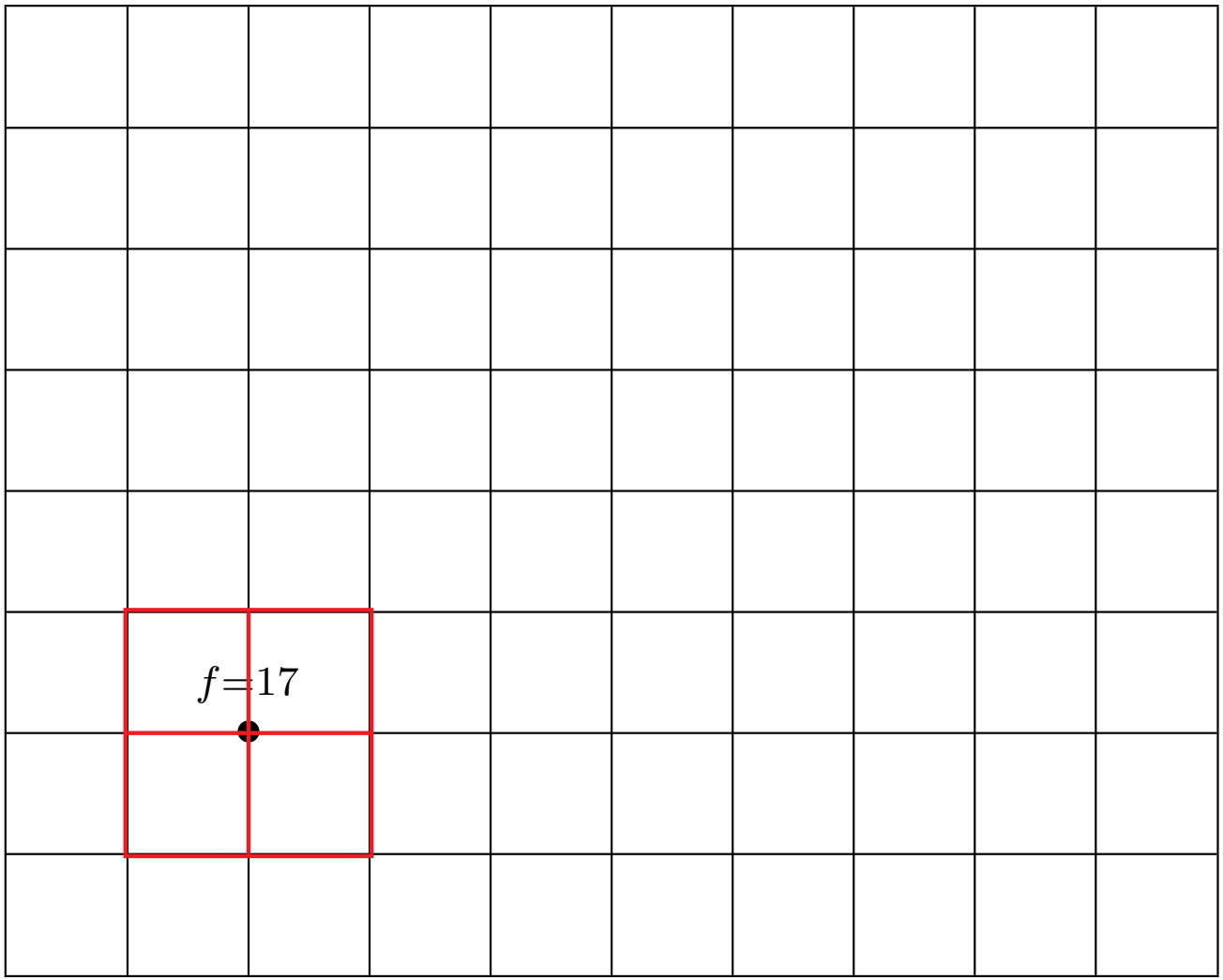
Coordinate search

$x_5 = (0,0)^T, \Delta_5 = \frac{1}{2}$



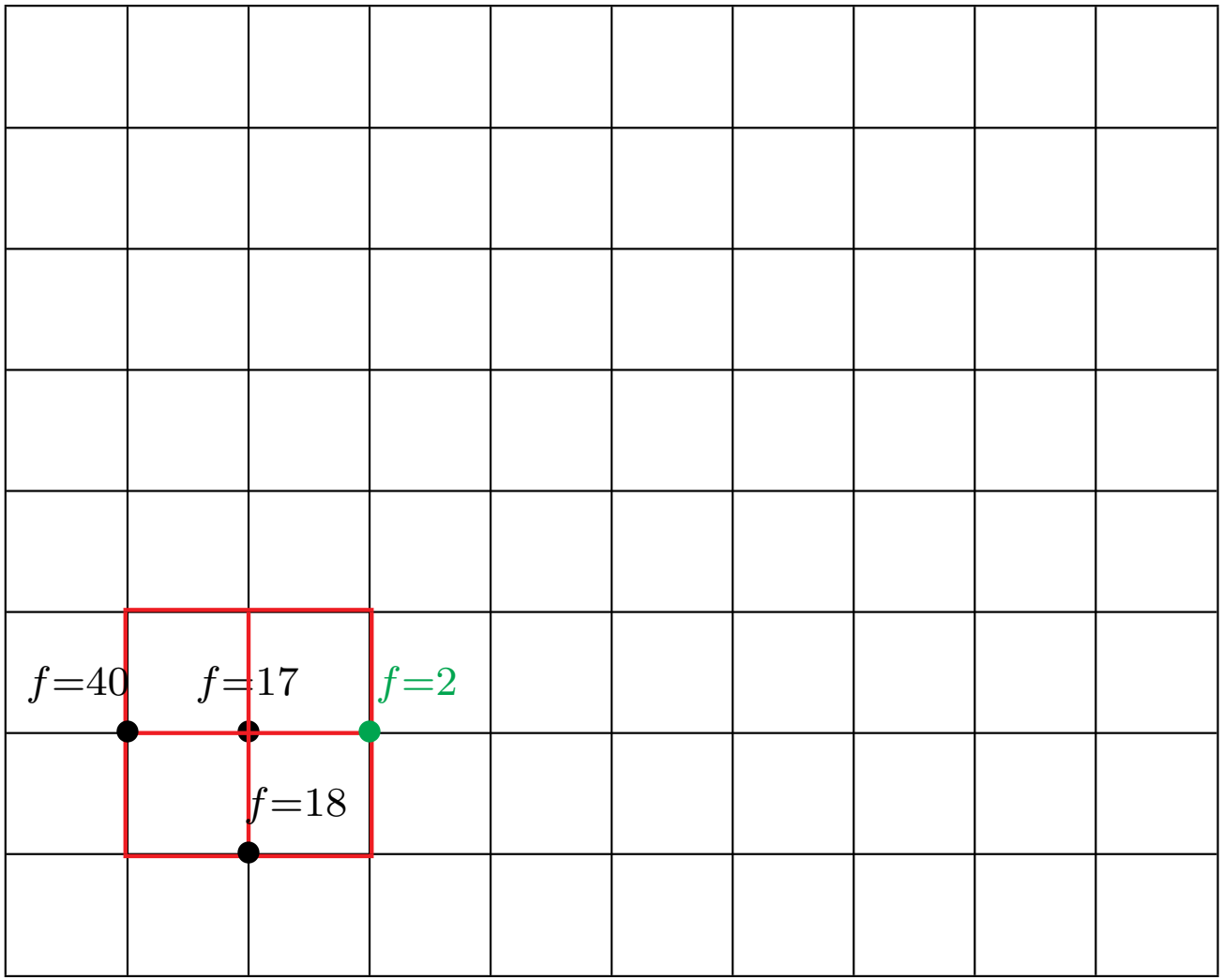
Coordinate search

$x_5=(0,0)^T, \Delta_5=\frac{1}{2}$



Coordinate search

$x_5=(0,0)^T, \Delta_5=\frac{1}{2}$



Generalized pattern search – Torczon 1996

- Initialization:
  x_0 : initial point in $\mathbb{R}^n$ such that $f(x_0) < \infty$
  $\Delta_0 > 0$: initial mesh size.
  D : finite positive spanning set of directions.

Generalized pattern search – Torczon 1996

- Initialization:
  x_0 : initial point in $\mathbb{R}^n$ such that $f(x_0) < \infty$
  $\Delta_0 > 0$: initial mesh size.
  D : finite positive spanning set of directions.

Definition

$D = \{d_1, d_2, \ldots, d_p\}$ is a positive spanning set if

$$\left\{ \sum_{i=1}^{p} \alpha_i d_i \ : \ \alpha_i \geq 0, i = 1, 2, \ldots p \right\} = \mathbb{R}^n.$$

Remark : $p \geq n + 1$.

Definition (Davis)

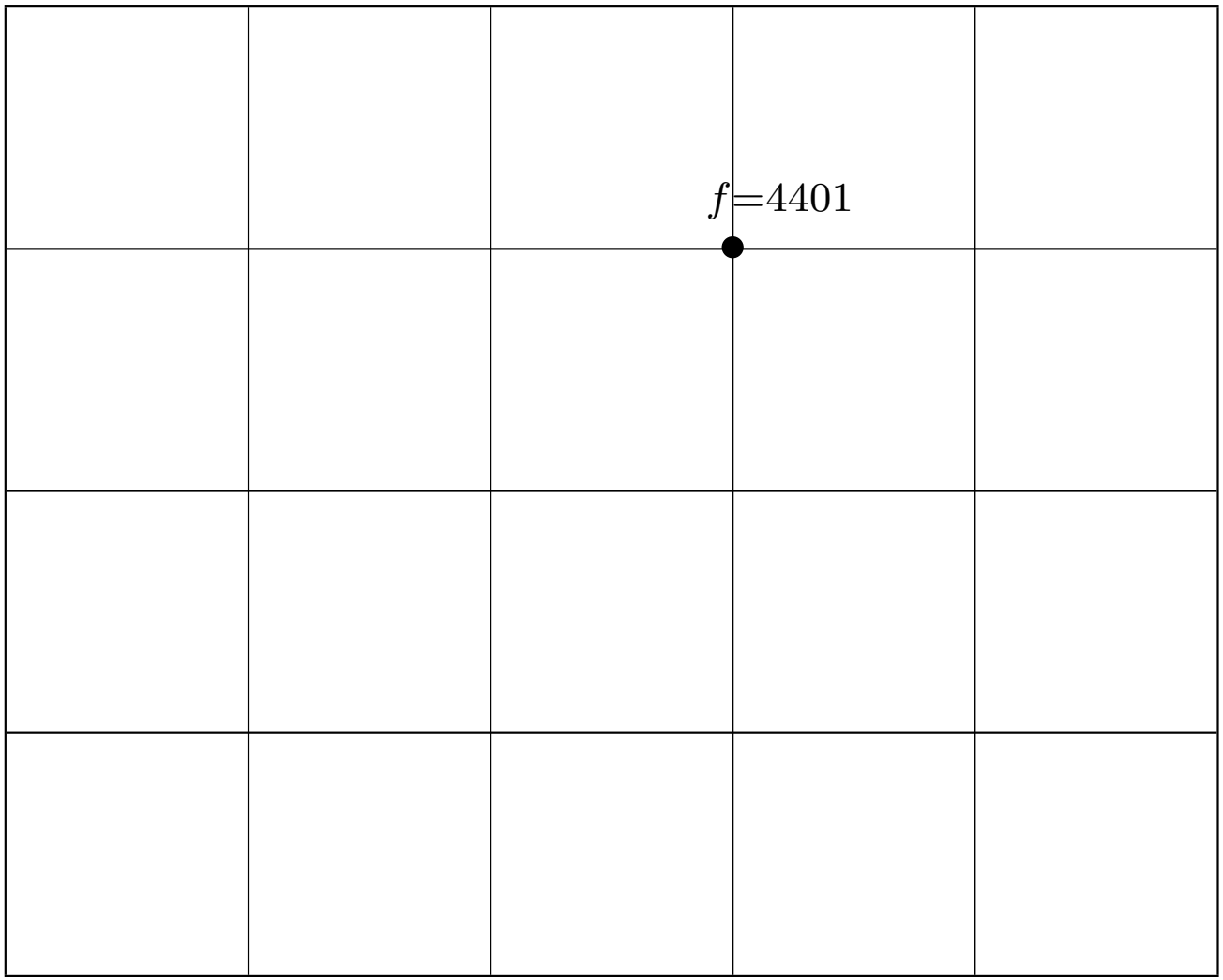
$D = \{d_1, d_2, \ldots, d_p\}$ is a positive basis is it is a positive spanning set and no proper subset of D is a positive spanning set.
Remark : $n + 1 \leq p \leq 2n$.

Generalized pattern search – Torczon 1996

- Initialization:
  x_0 : initial point in $\mathbb{R}^n$ such that $f(x_0) < \infty$
  $\Delta_0 > 0$: initial mesh size.
  D : finite positive spanning set of directions.
- For $k = 0, 1, \ldots$
  - Search step: Evaluate f at a finite number of mesh points.
  - Poll step: If the search failed, evaluate f at the poll points $\{x_k + \Delta_k d \ : \ d \in D_k\}$ where $D_k \subset D$ is a positive spanning set.
  - Parameter update:
    Set $\Delta_{k+1} < \Delta_k$ if no new incumbent was found,
    otherwise set $\Delta_{k+1} \geq \Delta_k$, and call x_{k+1} the new incumbent.

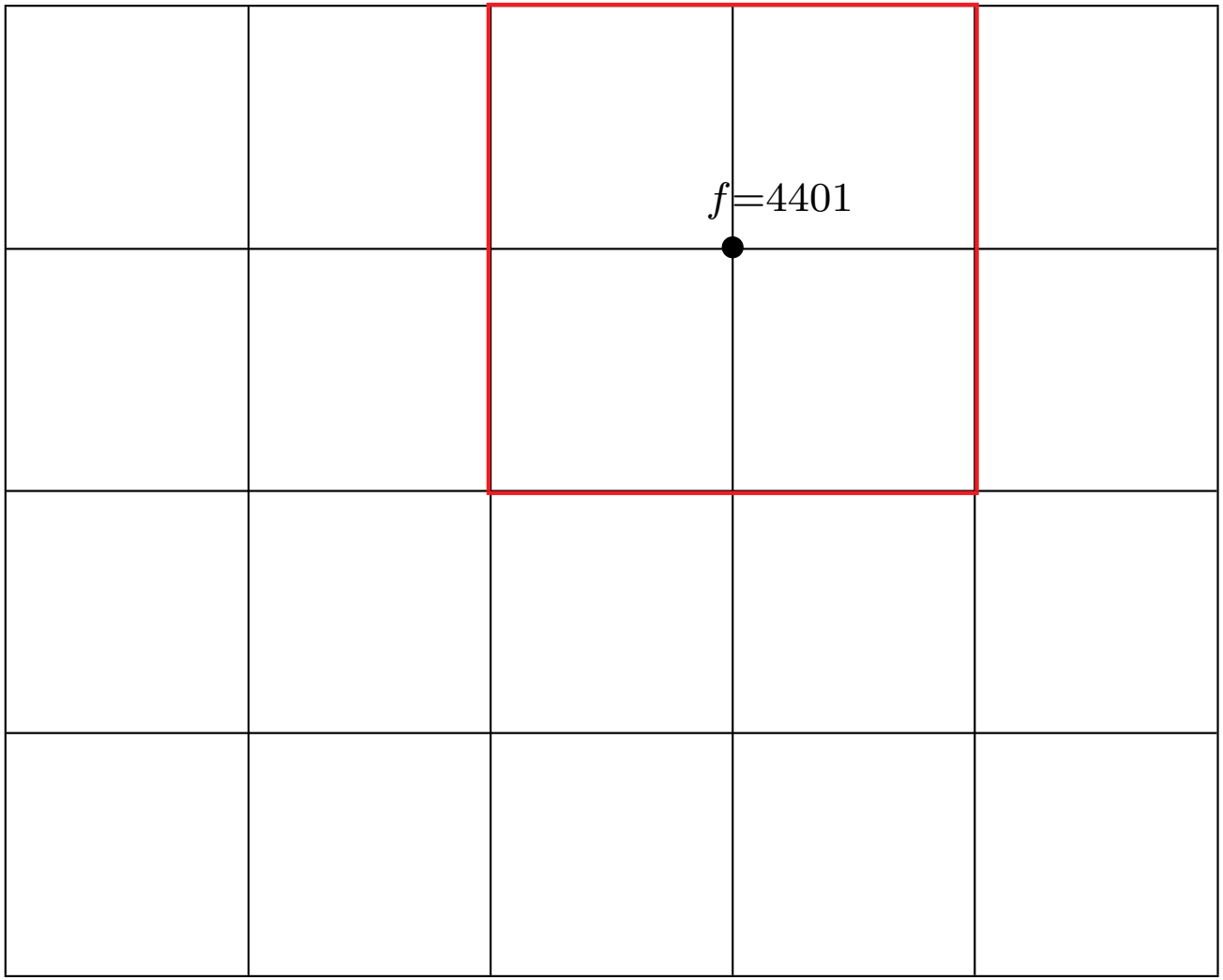
Generalized pattern search

$x_0=(2,2)^T, \Delta_0=1$



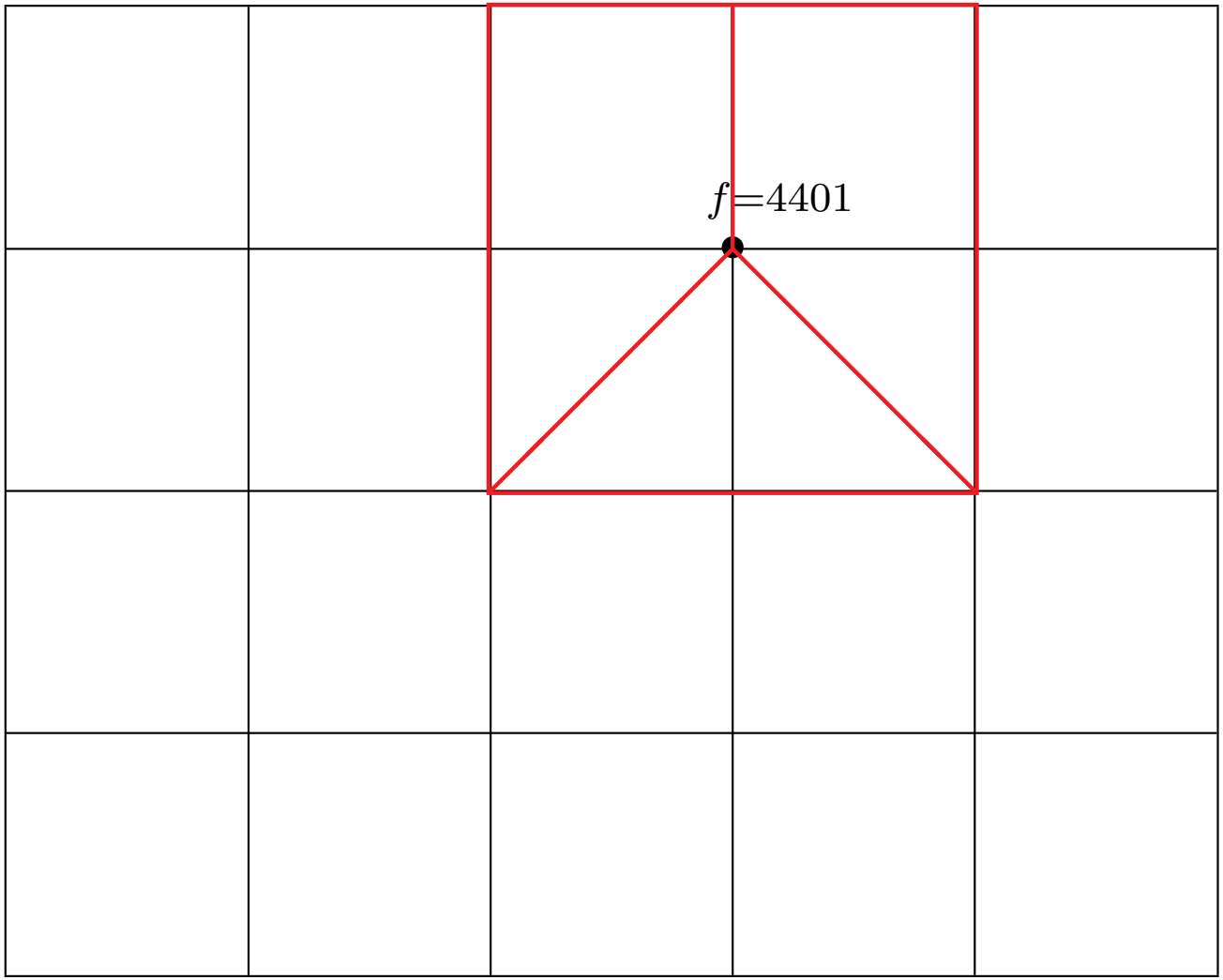
Generalized pattern search

$x_0=(2,2)^T, \Delta_0=1$



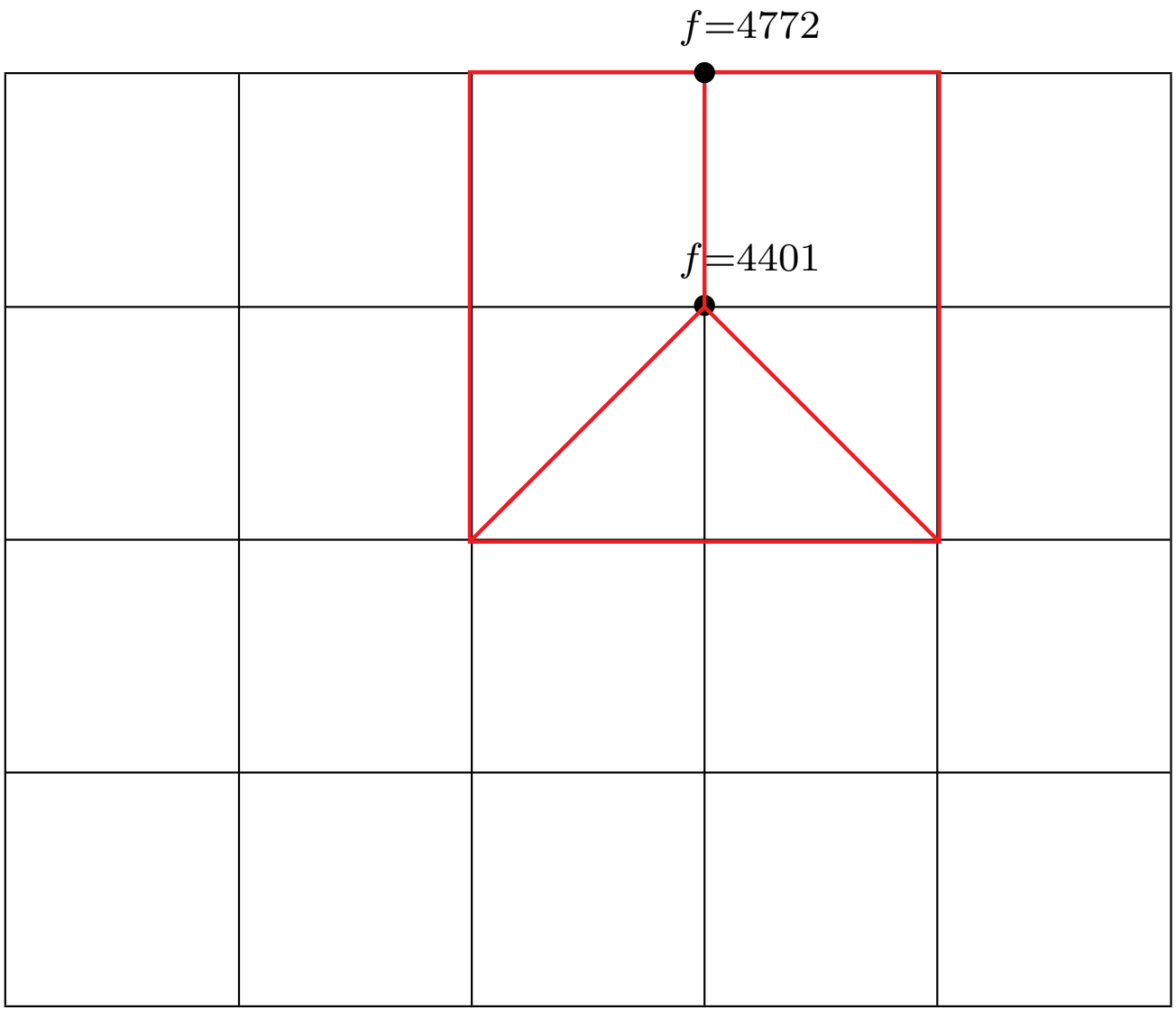
Generalized pattern search

$x_0=(2,2)^T, \Delta_0=1$



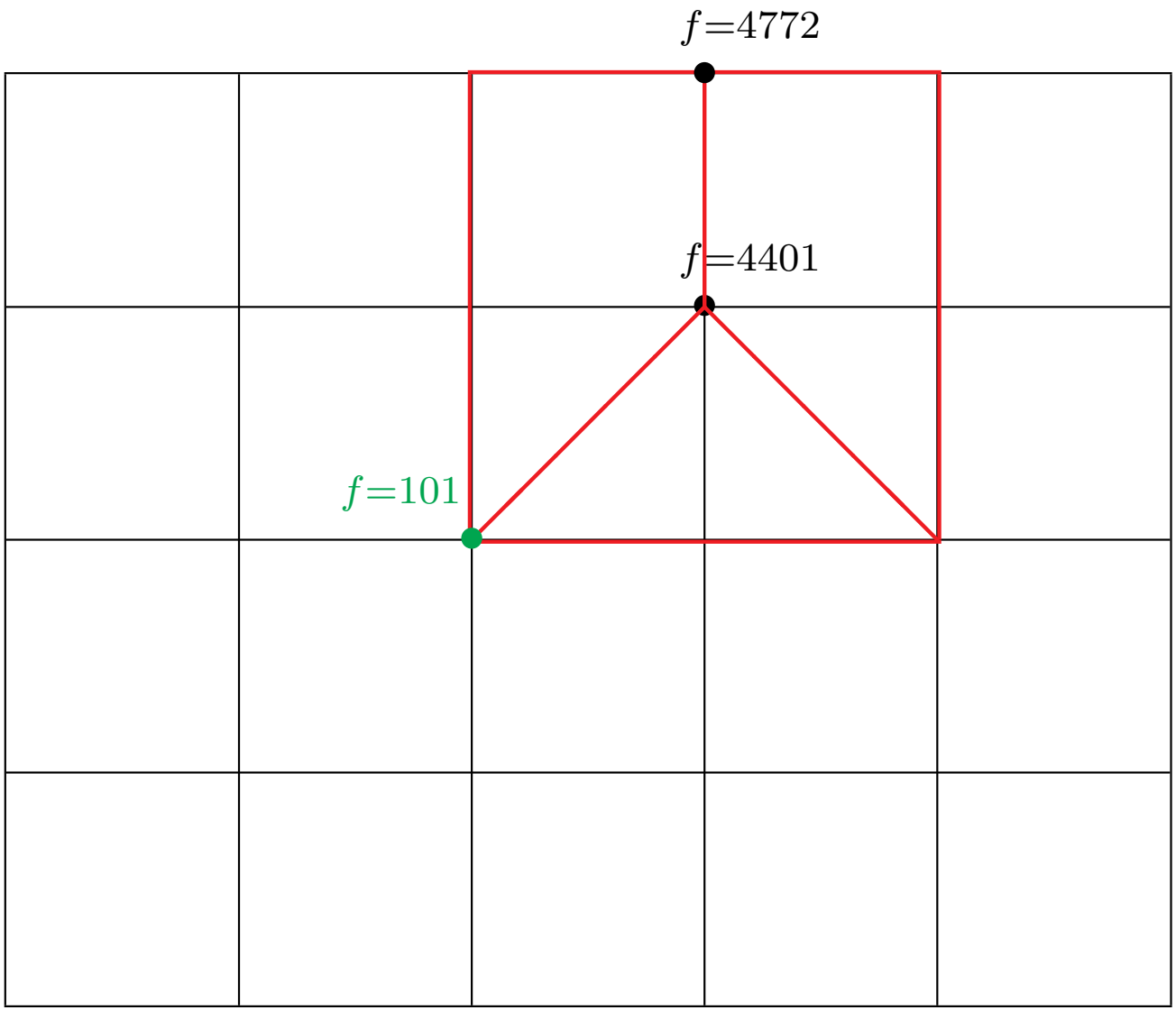
Generalized pattern search

$x_0=(2,2)^T, \Delta_0=1$



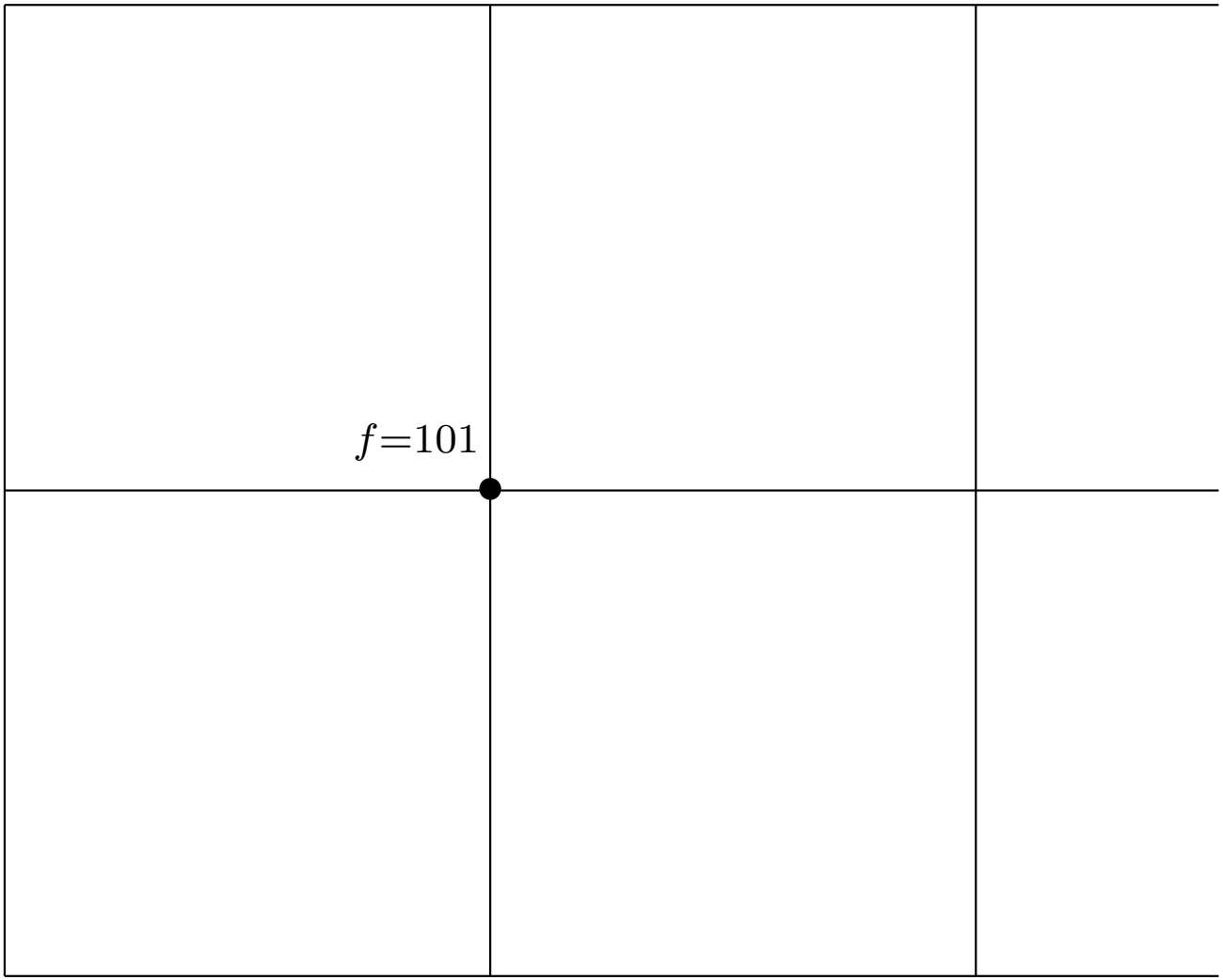
Generalized pattern search

$x_0=(2,2)^T, \Delta_0=1$



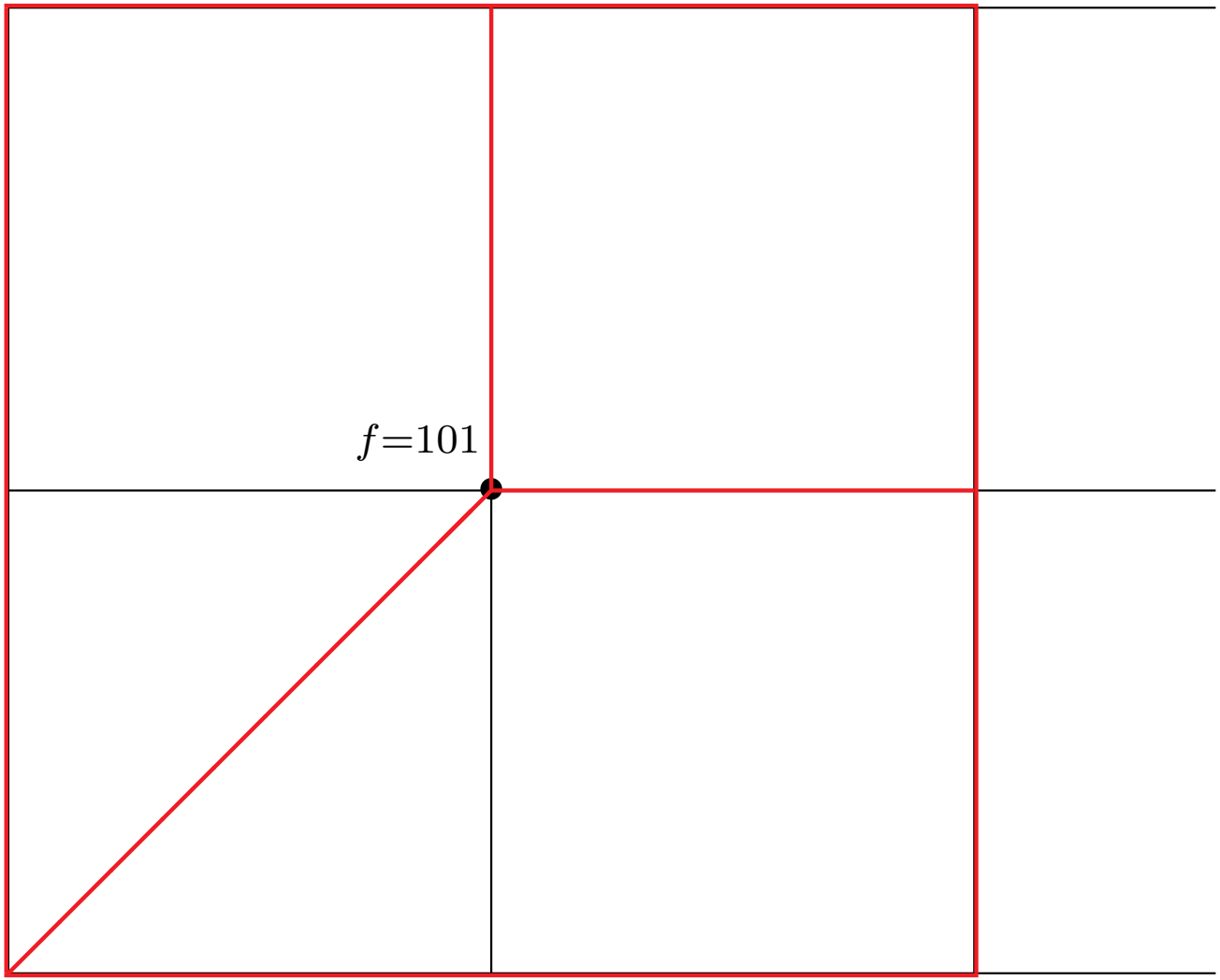
Generalized pattern search

$x_1=(1,2)^T, \Delta_1=2$

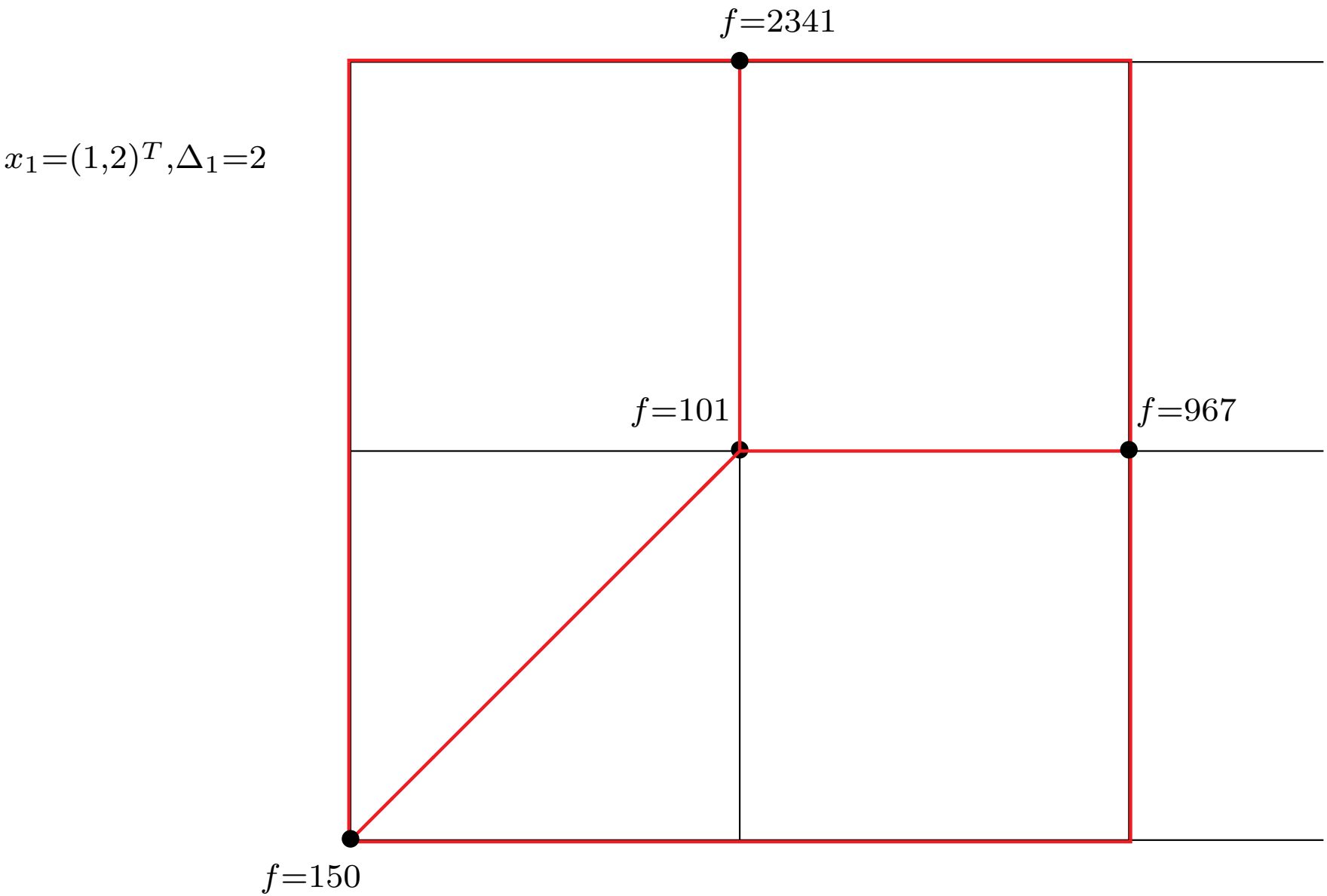


Generalized pattern search

$x_1=(1,2)^T, \Delta_1=2$

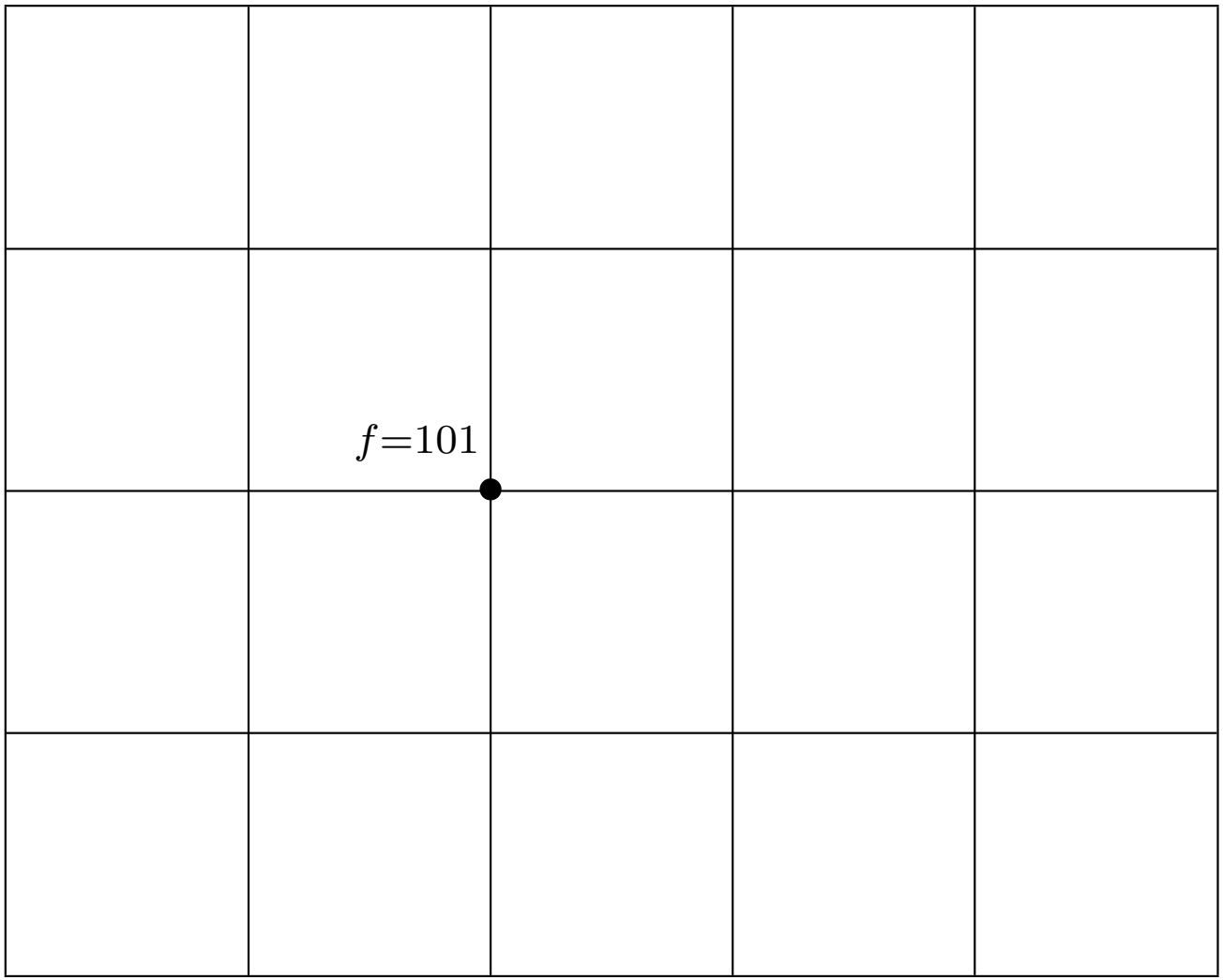


Generalized pattern search



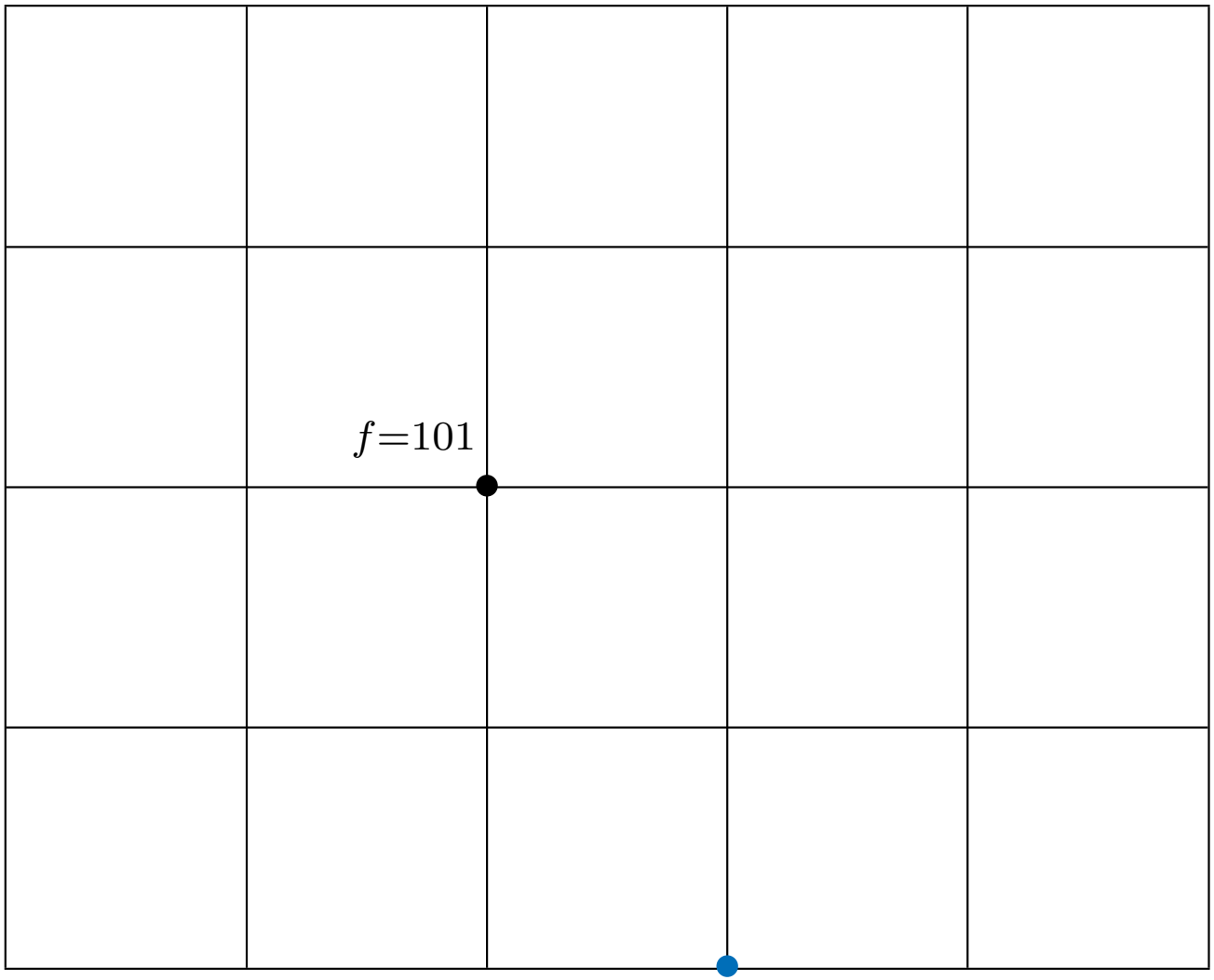
Generalized pattern search

$x_2=(2,2)^T, \Delta_2=1$



Generalized pattern search

$x_2=(2,2)^T, \Delta_2=1$



A SEARCH trial point. Ex: built using a simplex gradient

Convergence analysis

- If all iterates belong to a bounded set, then $\liminf_k \Delta_k = 0$.

Corollary

There is a convergent subsequence of iterates $x_k \to \hat{x}$ of mesh local optimizers, on meshes that get infinitely fine.

Convergence analysis

- If all iterates belong to a bounded set, then $\liminf_k \Delta_k = 0$.

Corollary

There is a convergent subsequence of iterates $x_k \to \hat{x}$ of mesh local optimizers, on meshes that get infinitely fine.

- GPS methods are directional. The analysis is tied to the fixed set of directions D: $f(x_k) \leq f(x_k + \Delta_k d)$ for every $d \in D_k \subseteq D$.

Theorem

If f is Lipschitz near $\hat{x}$, then for every direction $d \in D$ used infinitely often,

$$\begin{aligned} f^\circ(\hat{x}; d) &:= \limsup_{y \to \hat{x}, t \to 0} \frac{f(y + td) - f(y)}{t} \\ &\geq \limsup_k \frac{f(x_k + \Delta_k d) - f(x_k)}{\Delta_k} \geq 0. \end{aligned}$$

Convergence analysis

- If all iterates belong to a bounded set, then $\liminf_k \Delta_k = 0$.

Corollary

There is a convergent subsequence of iterates $x_k \to \hat{x}$ of mesh local optimizers, on meshes that get infinitely fine.

- GPS methods are directional. The analysis is tied to the fixed set of directions D: $f(x_k) \leq f(x_k + \Delta_k d)$ for every $d \in D_k \subseteq D$.

Corollary

If f is regular near $\hat{x}$, then for every direction $d \in D$ used infinitely often,

$$f'(\hat{x}; d) \;:=\; \lim_{t \to 0} \frac{f(x + td) - f(x)}{t} \;\geq\; 0.$$

Convergence analysis

- If all iterates belong to a bounded set, then $\liminf_k \Delta_k = 0$.

Corollary

There is a convergent subsequence of iterates $x_k \to \hat{x}$ of mesh local optimizers, on meshes that get infinitely fine.

- GPS methods are directional. The analysis is tied to the fixed set of directions D: $f(x_k) \leq f(x_k + \Delta_k d)$ for every $d \in D_k \subseteq D$.

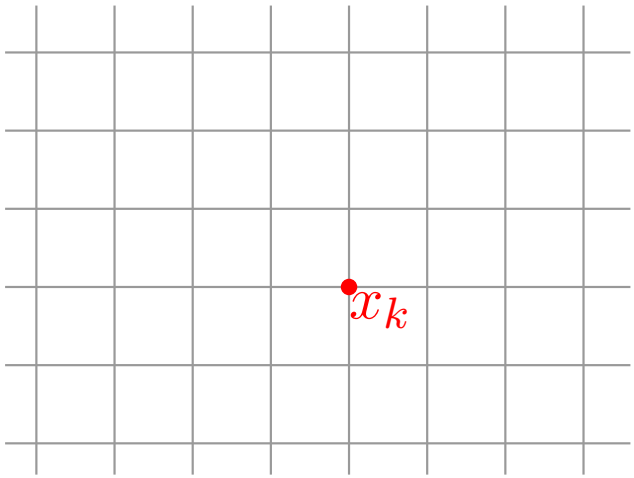
Corollary

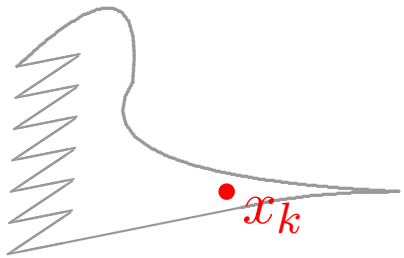
If f is strictly differentiable near $\hat{x}$, then

$$\nabla f(\hat{x}) = 0.$$

Related methods

- The GPS iterates at each iteration are confined to reside on the mesh.
- The mesh gets finer and finer, but still, some users want to evaluate the functions at non-mesh points.
- Frame-based methods, and Generalized Set Search methods remove this restriction to the mesh. The trade-off is that they require a minimal decrease on the objective to accept new solutions.





GPS: trial points on the mesh GSS: trial points away from x_k

Related methods

- The GPS iterates at each iteration are confined to reside on the mesh.
- The mesh gets finer and finer, but still, some users want to evaluate the functions at non-mesh points.
- Frame-based methods, and Generalized Set Search methods remove this restriction to the mesh. The trade-off is that they require a minimal decrease on the objective to accept new solutions.
- DIRECT partitions the space of variables into hyperrectangles, and iteratively refines the most promising ones.

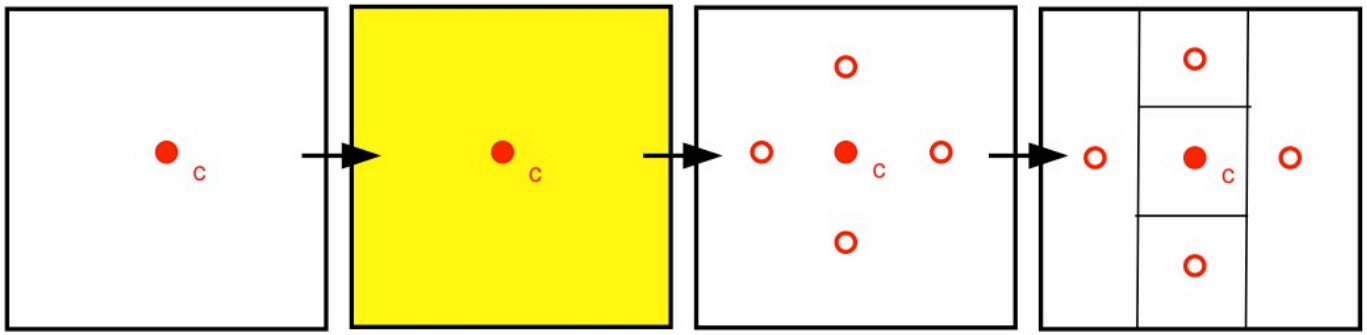
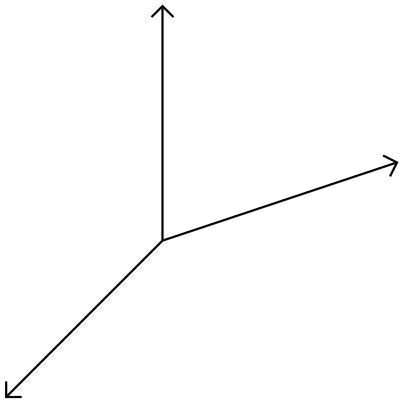


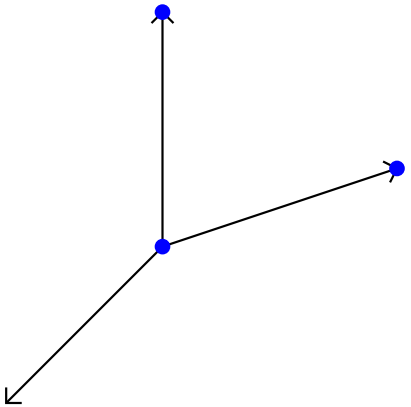
Figure taken from Finkel and Kelley CRSC-TR04-30.

Presentation outline

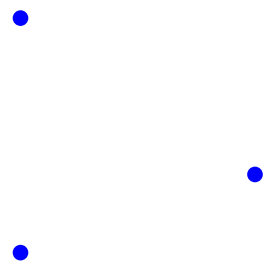
Positive bases vs. simplices



Positive bases vs. simplices



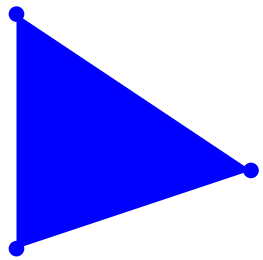
Positive bases vs. simplices



The $3 = n + 1$ points form an affinely independent set.

A set of $m + 1$ points $Y = \{y^0, y^1, \ldots, y^m\}$ is said to be affinely independent if its affine hull $\text{aff}(y^0, y^1, \ldots, y^m)$ has dimension m.

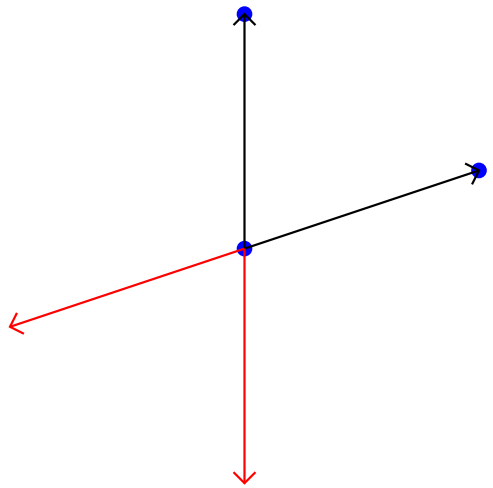
Positive bases vs. simplices



Their convex hull is a simplex of dimension $n = 2$.

Given an affinely independent set of points $Y = \{y^0, y^1, \ldots, y^m\}$,
its convex hull is called a simplex of dimension m.

Positive bases vs. simplices



Its reflection produces a (maximal) positive basis.

Simplices

The volume of a simplex of vertices $Y = \{y^0, y^1, \ldots, y^n\}$ is defined by

$$\text{vol}(Y) \;=\; \frac{|\det(S(Y))|}{n!}$$

where

$$S(Y) \;=\; \left[y^1 - y^0 \cdots y^n - y^0\right].$$

Note that $\text{vol}(Y) > 0$ (since the vertices are affinely independent).

A measure of geometry for simplices is the normalized volume:

$$\text{von}(Y) \;=\; \text{vol}\left(\frac{1}{\text{diam}(Y)} Y\right).$$

Simplicial Direct-Search Methods

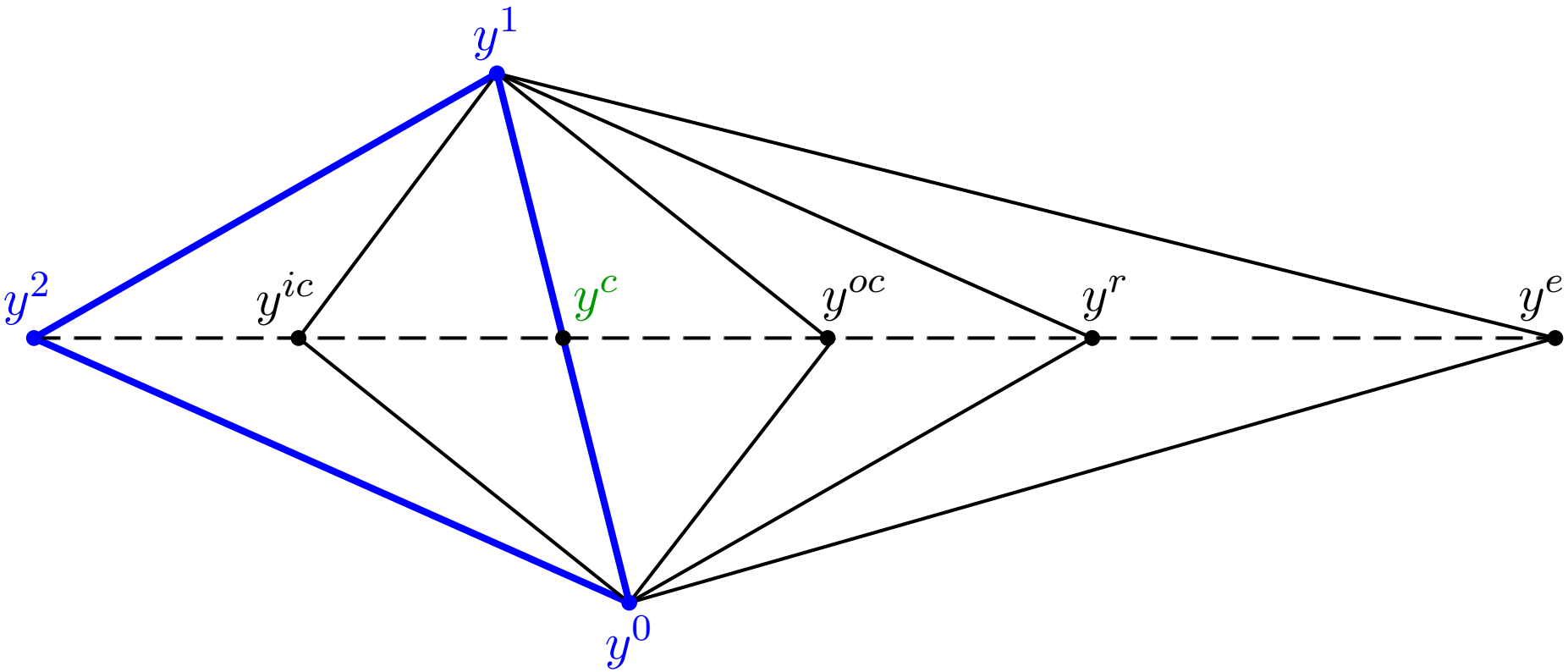
The Nelder-Mead method:

- Considers a simplex at each iteration, trying to replace the worst vertex by a new one.
- For that it performs one of the following simplex operations: reflexion, expansion, outside contraction, inside contraction.

  $\longrightarrow$ Costs 1 or 2 function evaluations (per iteration).
- If they all the above fail the simplex is shrunk.

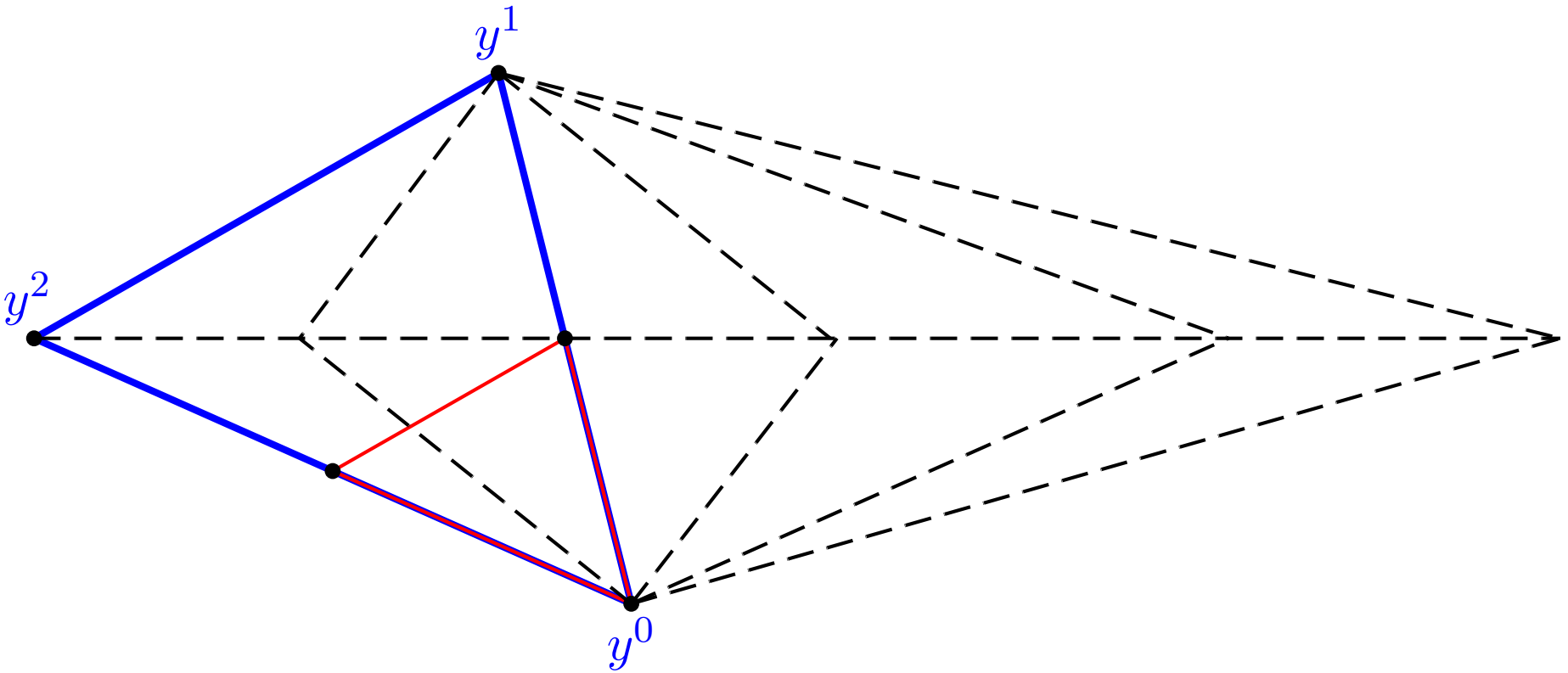
  $\longrightarrow$ Additional n function evaluations (per iteration).

Nelder Mead simplex operations (reflections, expansions, outside contractions, inside contractions)



y^c is the centroid of the face opposed to the worse vertex y^2.

Nelder Mead simplex operations (shrinks)

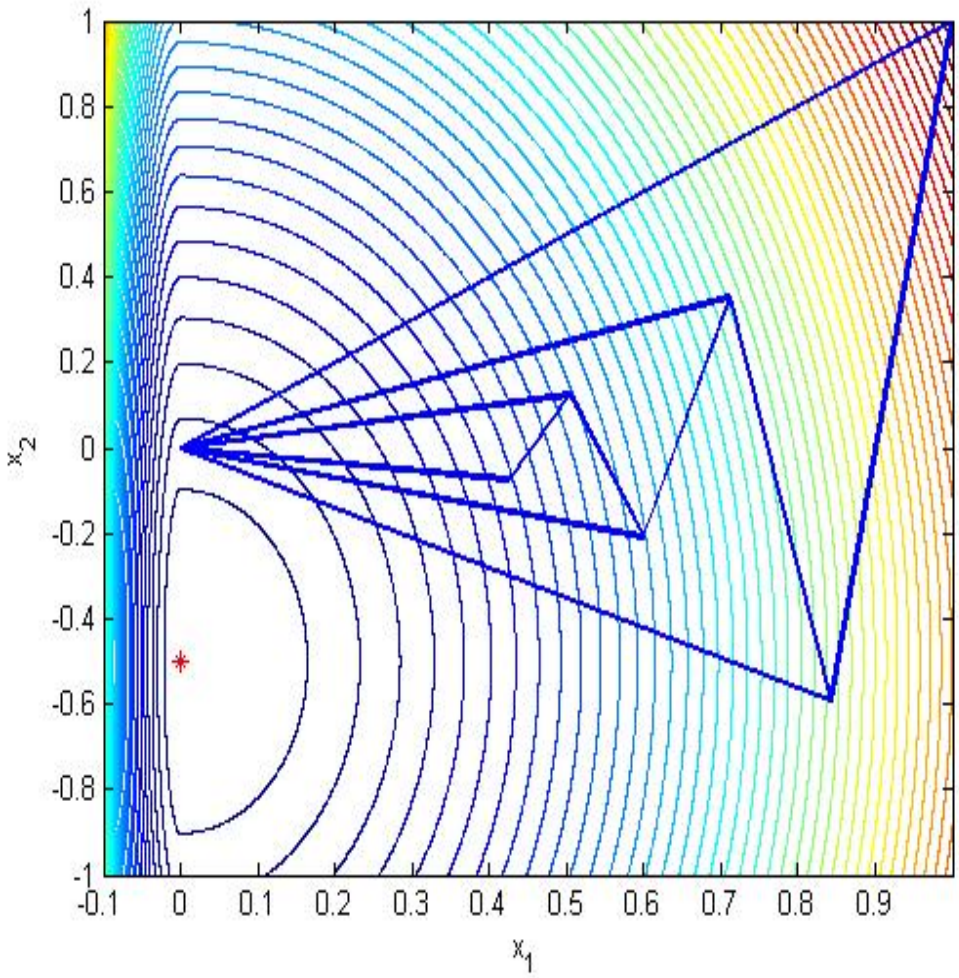


McKinnon counter-example:

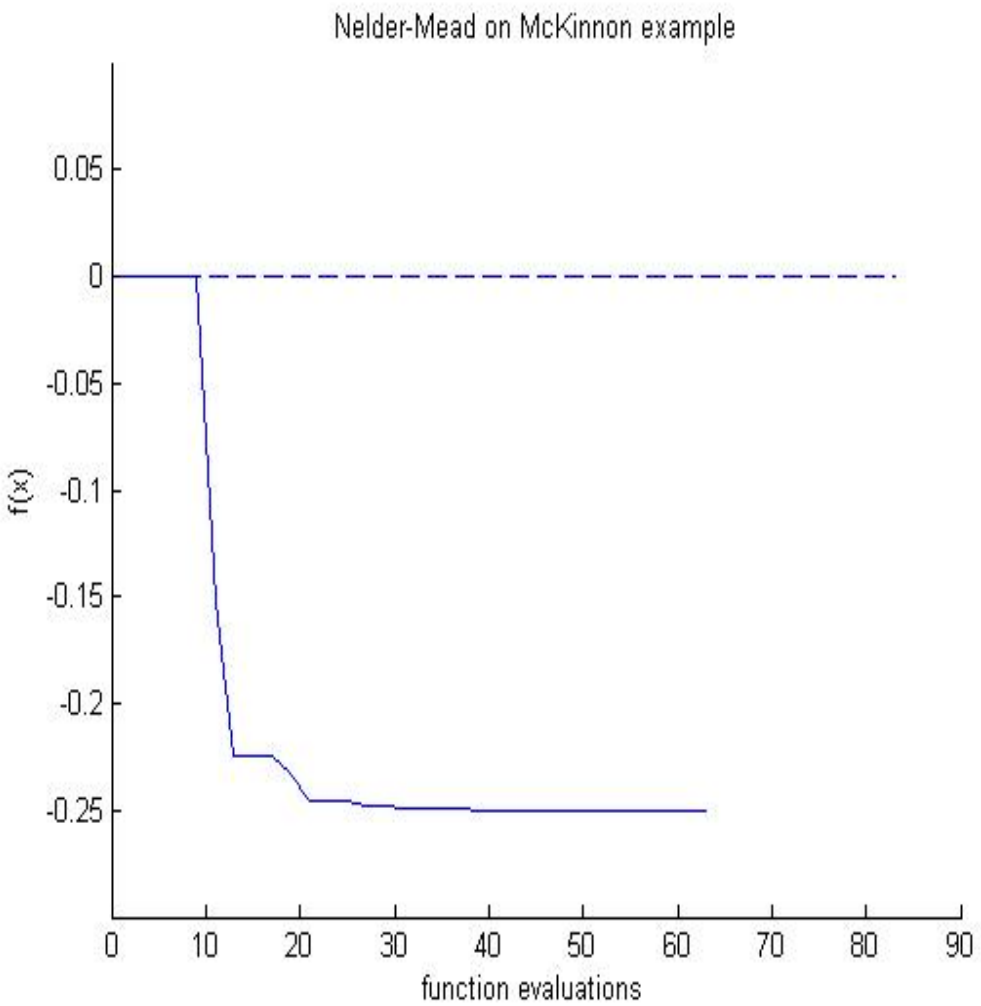
The Nelder-Mead method:

- Attempts to capture the curvature of the function.
- Is globally convergent when $n = 1$.
- Can fail for $n > 1$ (e.g. due to repeated inside contractions).

McKinnon counter-example



Nelder-Mead on McKinnon example (two different starting simplices)



Modified Nelder-Mead methods
For Nelder-Mead to globally converge one must:

- Control the internal angles (normalized volume) in all simplex operations but shrinks.

  CAUTION (very counterintuitive): Isometric reflections only preserve internal angles when $n = 2$ or the simplices are equilateral.

  $\longrightarrow$ Need for a back-up polling.

- Impose sufficient decrease instead of simple decrease for accepting new iterates:

$$f(\text{new point}) \leq f(\text{previous point}) - o(\text{simplex diameter}).$$

Modified Nelder-Mead methods

Let $\{Y_k\}$ be the sequence of simplices generated.
Let f be bounded from below and uniformly continuous in $\mathbb{R}^n$.

Theorem (Step size going to zero)

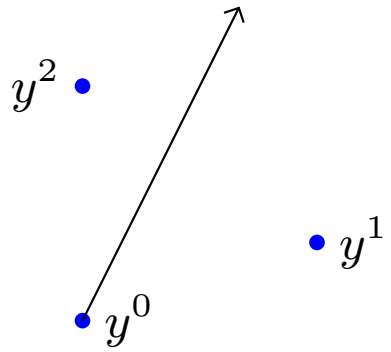
The diameters of the simplices converge to zero:

$$\lim_{k \longrightarrow +\infty} \mathit{diam}(Y_k) = 0.$$

Theorem (Global convergence)

If f is continuously differentiable in $\mathbb{R}^n$ and $\{Y_k\}$ lies in a compact set then $\{Y_k\}$ has at least one stationary limit point x_*.

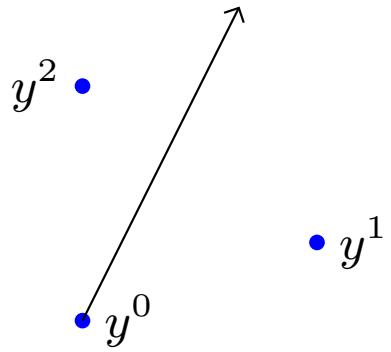
Simplex gradients



It is possible to build a simplex gradient:

$$\nabla_s f(y^0) \;=\; \left[\; y^1 - y^0 \quad y^2 - y^0 \;\right]^{-\top} \left[\; \begin{array}{c} f(y^1) - f(y^0) \\ f(y^2) - f(y^0) \end{array} \;\right].$$

Simplex gradients



It is possible to build a simplex gradient:

$$\nabla_s f(y^0) \;=\; S^{-\top}\delta(f).$$

Simplex gradients

Simple Fact (What is a simplex gradient)

The simplex gradient (based on $n+1$ affinely independent points) is the gradient of the corresponding linear interpolation model:

$$\begin{aligned} f(y^0) + \langle \nabla_s f(y^0), y^i - y^0 \rangle &= f(y^0) + \left(S^{-\top}\delta(f)\right)^\top (Se_i) \\ &= f(y^0) + \delta(f)_i \\ &= f(y^i). \end{aligned}$$

$\longrightarrow$ Simplex derivatives are the derivatives of the polynomial models.

Line-search methods

For instance, the implicit filtering method:

- Computes a simplex gradient (per iteration).

  $\longrightarrow$ The function evaluations can be computed in parallel.

  $\longrightarrow$ It can use regression with more than $n+1$ points.

- Improves the simplex gradient by applying a quasi-Newton update.
- Performs a line search along the negative computed direction.

The noise is filtered: (i) by the simplex gradient calculation (especially when using regression); (ii) by not performing an accurate line search.

Presentation outline

Trust-Region Methods for DFO (basics)

Trust-region methods for DFO typically:

- attempt to form quadratic models (by interpolation/regression and using polynomials or radial basis functions)

$$m(\Delta x) \;=\; f(x_k) + \langle g_k, \Delta x\rangle + 1/2\langle \Delta x, H_k \Delta x\rangle$$

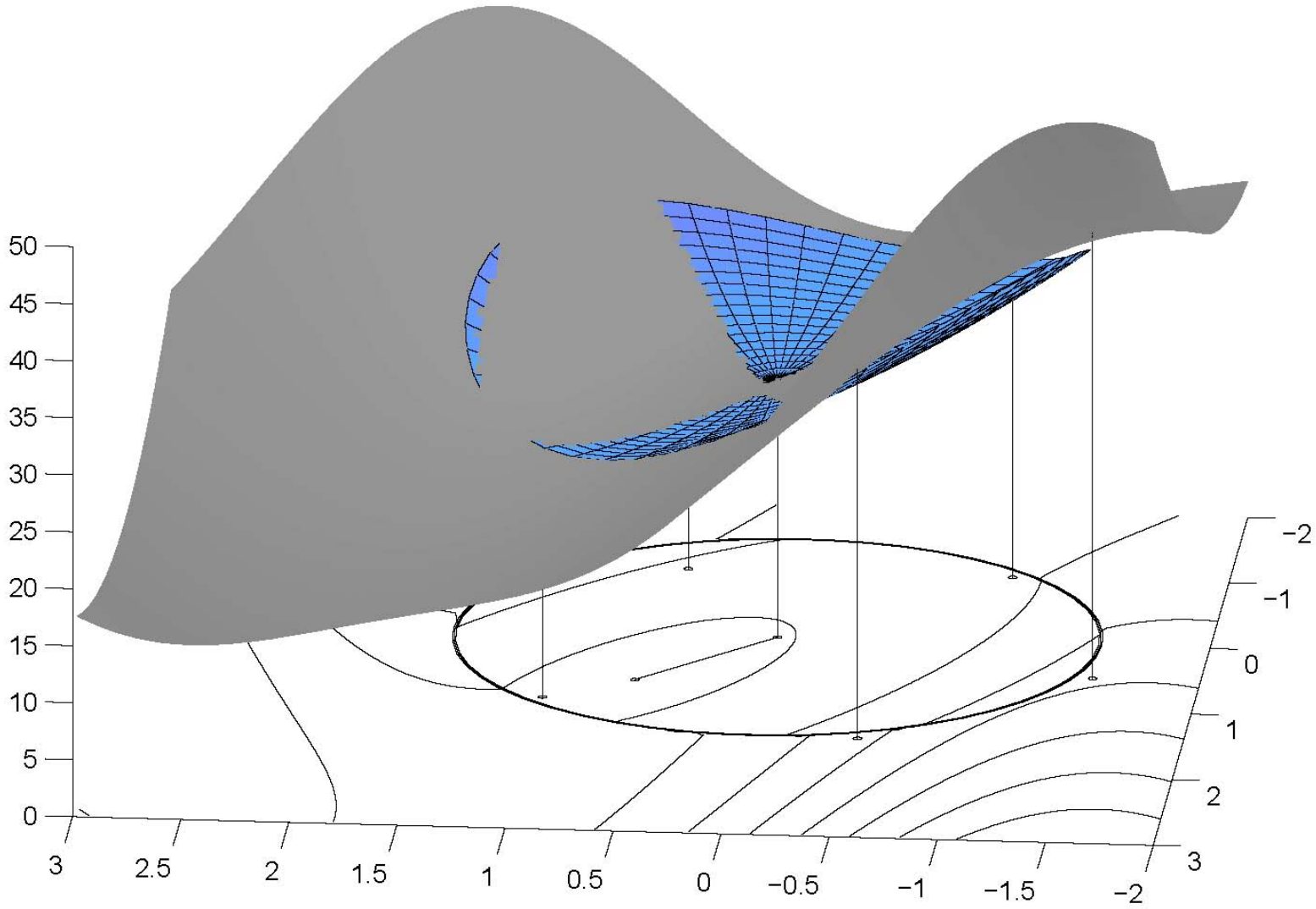
  based on well-poised sample sets.

$\longrightarrow$ Well poisedness ensures fully-linear or fully quadratic models.

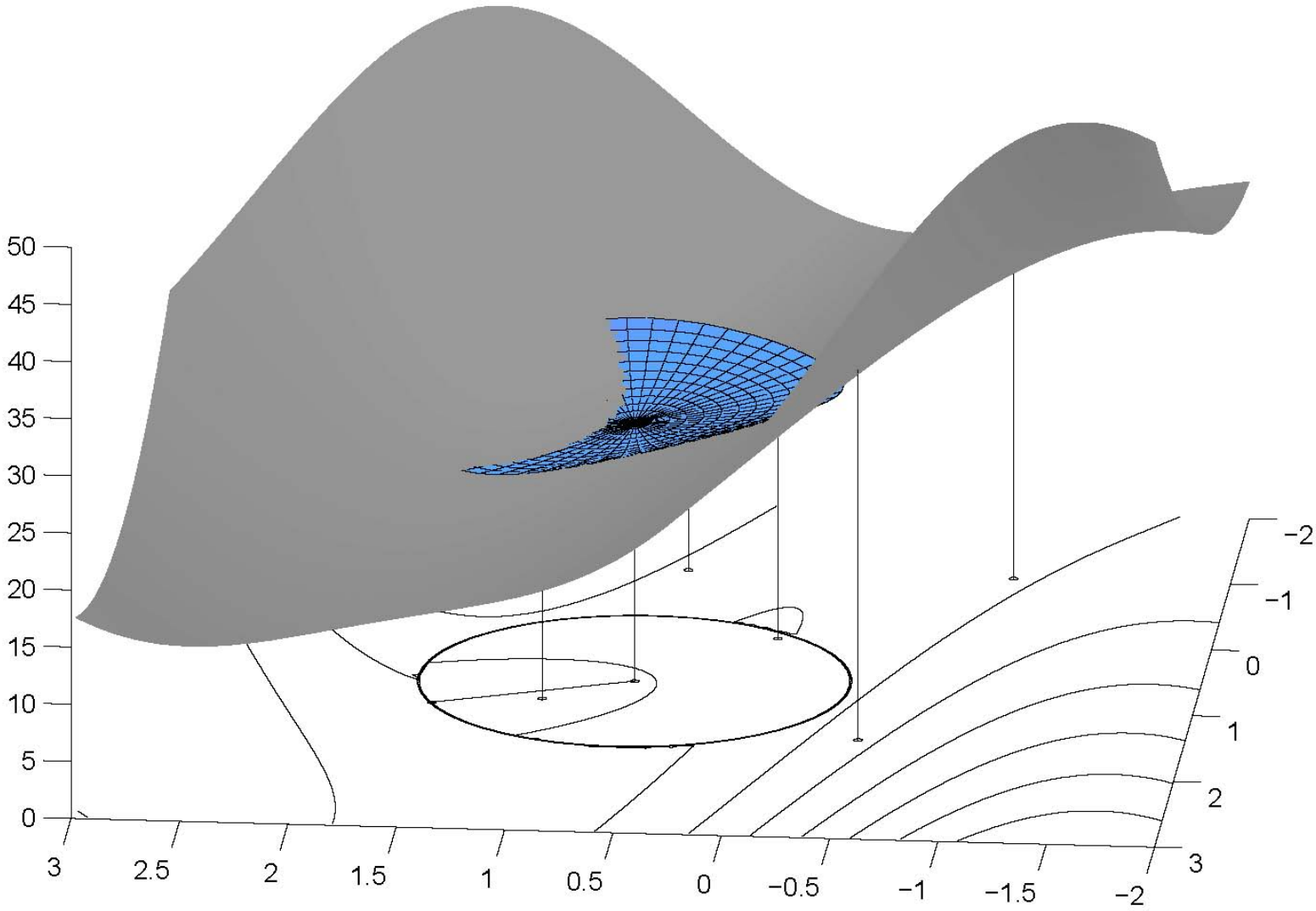
- Calculate a step Δx_k by solving

$$\min_{\Delta x \in B(x_k;\Delta_k)} \quad m(\Delta x).$$

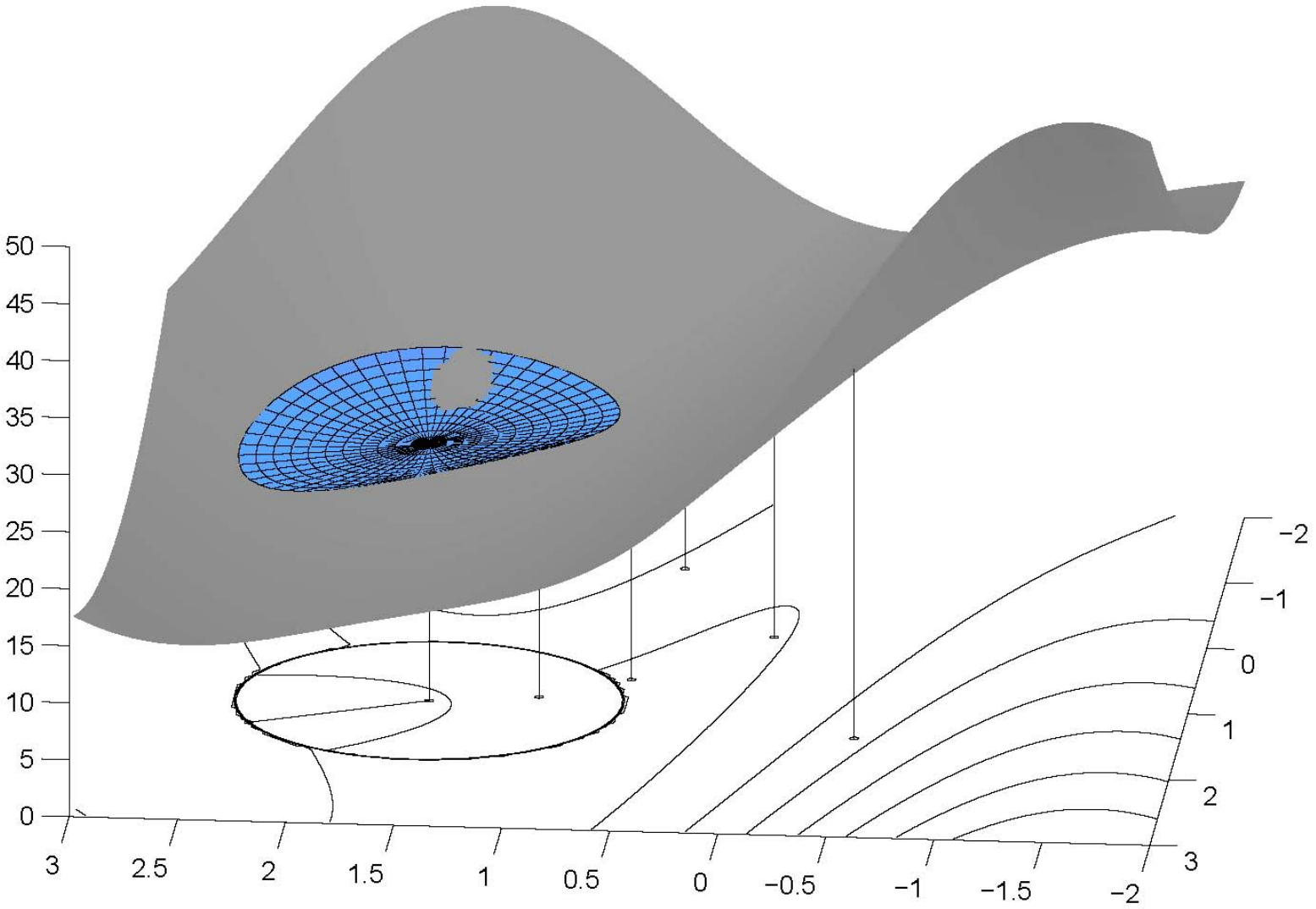
Example of ill poisedness



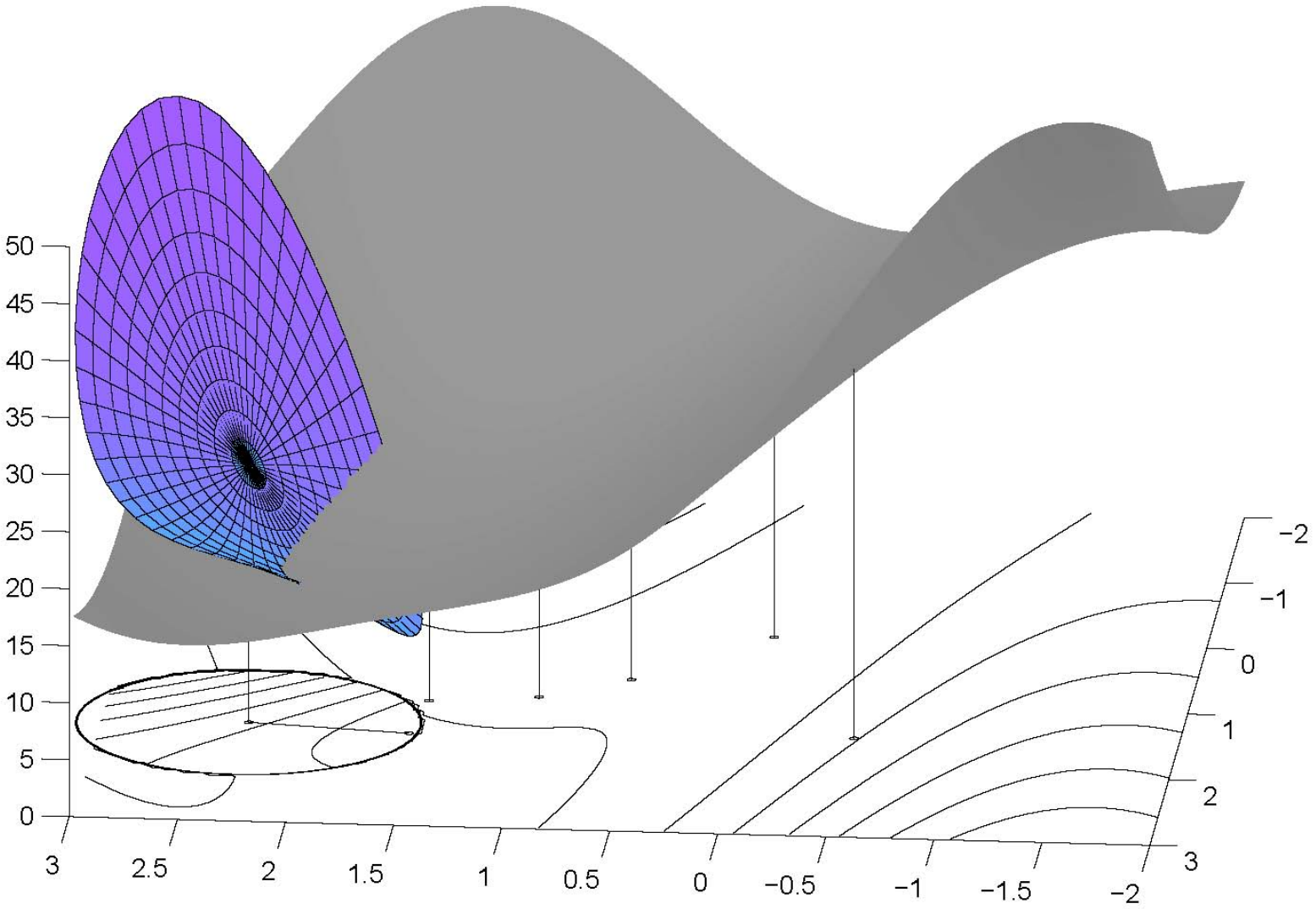
Example of ill poisedness



Example of ill poisedness



Example of ill poisedness



Fully-linear models

Given a point x and a trust-region radius Δ, a model $m(y)$ around x is called fully linear if

- It is continuous differentiable with Lipschitz continuous first derivatives.
- The following error bounds hold:

$$\|\nabla f(y) - \nabla m(y)\| \leq \kappa_{eg}\,\Delta \qquad \forall y \in B(x;\Delta)$$

and

$$|f(y) - m(y)| \leq \kappa_{ef}\,\Delta^2 \qquad \forall y \in B(x;\Delta).$$

For a class of fully-linear models, the (unknown) constants $\kappa_{ef}, \kappa_{eg} > 0$ must be independent of x and Δ.

Fully-linear models

For a class of fully-linear models, one must also guarantee that:

- There exists a model-improvement algorithm, that in a finite, uniformly bounded (with respect to x and Δ) number of steps can:

  — certificate that a given model is fully linear on $B(x;\Delta)$,

  — or (if the above fails), find a model that is fully linear on $B(x;\Delta)$.

Fully-quadratic models

Given a point x and a trust-region radius Δ, a model $m(y)$ around x is called fully quadratic if

- It is twice continuous differentiable with Lipschitz continuous second derivatives.
- The following error bounds hold (...):

$$\|\nabla^2 f(y) - \nabla^2 m(y)\| \leq \kappa_{eh}\,\Delta \qquad \forall y \in B(x;\Delta)$$

$$\|\nabla f(y) - \nabla m(y)\| \leq \kappa_{eg}\,\Delta^2 \qquad \forall y \in B(x;\Delta)$$

and

$$|f(y) - m(y)| \leq \kappa_{ef}\,\Delta^3 \qquad \forall y \in B(x;\Delta).$$

TR methods for DFO (basics)

- Set x_{k+1} to $x_k + \Delta x_k$ (success) or to x_k (unsuccess) and update Δ_k depending on the value of

$$\rho_k = \frac{f(x_k) - f(x_k + \Delta x_k)}{m(0) - m(\Delta x_k)}.$$

- Attempt to accept steps based on simple decrease, i.e., if

$$\rho_k > 0 \iff f(x_k + \Delta x_k) < f(x_k).$$

TR methods for DFO (main features)

- Reduce Δ_k only if ρ_k is small and the model is FL/FQ.
- Accept new iterates based on simple decrease ($\rho_k > 0$) as long as the model is FL/FQ
- Allow for model-improving iterations (when ρ_k is not large enough and the model is not certifiably FL/FQ).

  $\longrightarrow$ Do not reduce Δ_k.
- Incorporate a criticality step (1st or 2nd order) when the 'stationarity' of the model is small.

  $\longrightarrow$ Internal cycle of reductions of Δ_k.

TR methods for DFO (global convergence)

Theorem (Step size going to zero)

The trust-region radius converges to zero:

$$\Delta_k \longrightarrow 0.$$

Theorem (Global convergence (1st order) — TRM)

If f has Lipschitz continuous first derivatives then

$$\|\nabla f(x_k)\| \longrightarrow 0.$$

$\longrightarrow$ Compactness of $L(x_0)$ is not necessary.

$\longrightarrow$ True for simple decrease.

$\longrightarrow$ Use of fully-linear models when necessary.

TR methods for DFO (global convergence)

Theorem (Global convergence (2nd order) — TRM)

If f has Lipschitz continuous second derivatives then

$$\max\left\{\|\nabla f(x_k)\|, -\lambda_{min}[\nabla^2 f(x_k)]\right\} \longrightarrow 0.$$

$\longrightarrow$ Compactness of $L(x_0)$ is not necessary.

$\longrightarrow$ True for simple decrease.

$\longrightarrow$ Use of fully-quadratic models when necessary.

Polynomial models

Given a sample set $Y = \{y^0, y^1, \ldots, y^p\}$ and a polynomial basis ϕ, one considers a system of linear equations:

$$M(\phi, Y)\alpha \;=\; f(Y),$$

where

$$M(\phi, Y) = \begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \cdots & \phi_p(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \cdots & \phi_p(y^1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(y^p) & \phi_1(y^p) & \cdots & \phi_p(y^p) \end{bmatrix} \quad f(Y) = \begin{bmatrix} f(y^0) \\ f(y^1) \\ \vdots \\ f(y^p) \end{bmatrix}.$$

Example: $\phi = \{1, x_1, x_2, x_1^2/2, x_2^2/2, x_1x_2\}$.

Polynomial models

The system

$$M(\phi, Y)\alpha = f(Y)$$

can be

- Determined when (# points) $=$ (# basis components).
- Overdetermined when (# points) $>$ (# basis components).

  $\longrightarrow$ least-squares regression solution.
- Undetermined when (# points) $<$ (# basis components).

  $\longrightarrow$ minimum-norm solution

  $\longrightarrow$ minimum Frobenius norm solution

Error bounds for polynomial models

Given an interpolation set Y, the error bounds ($\#$ points $\geq n+1$) are of the form

$$\|\nabla f(y) - \nabla m(y)\| \leq [C(n)C(f)C(Y)]\,\Delta \qquad \forall y \in B(x;\Delta)$$

where

- $C(n)$ is a small constant depending on n.
- $C(f)$ measures the smoothness of f (in this case the Lipschitz constant of ∇f).
- $C(Y)$ is related to the geometry of Y.

Geometry constants

Let $\{\mathcal{L}_y(z),\, y \in Y\}$ be the set of Lagrange polynomials associated with Y.

The geometry constant is typically given by

$$C(Y) \;=\; \max_{y \in Y} \; \max_{z \in B(x;\Delta)} \; |\mathcal{L}_y(z)| \,.$$

$\longrightarrow$ leading to model-improvement algorithms based on the maximization of Lagrange polynomials.

In fact, we say that Y is Λ–poised when

$$C(Y) \;=\; \max_{y \in Y} \; \max_{z \in B(x;\Delta)} \; |\mathcal{L}_y(z)| \;\leq\; \Lambda.$$

Model improvement (Lagrange polynomials)

Choose $\Lambda > 1$. Let Y be a poised set.

Each iteration of a model-improvement algorithm consists of:

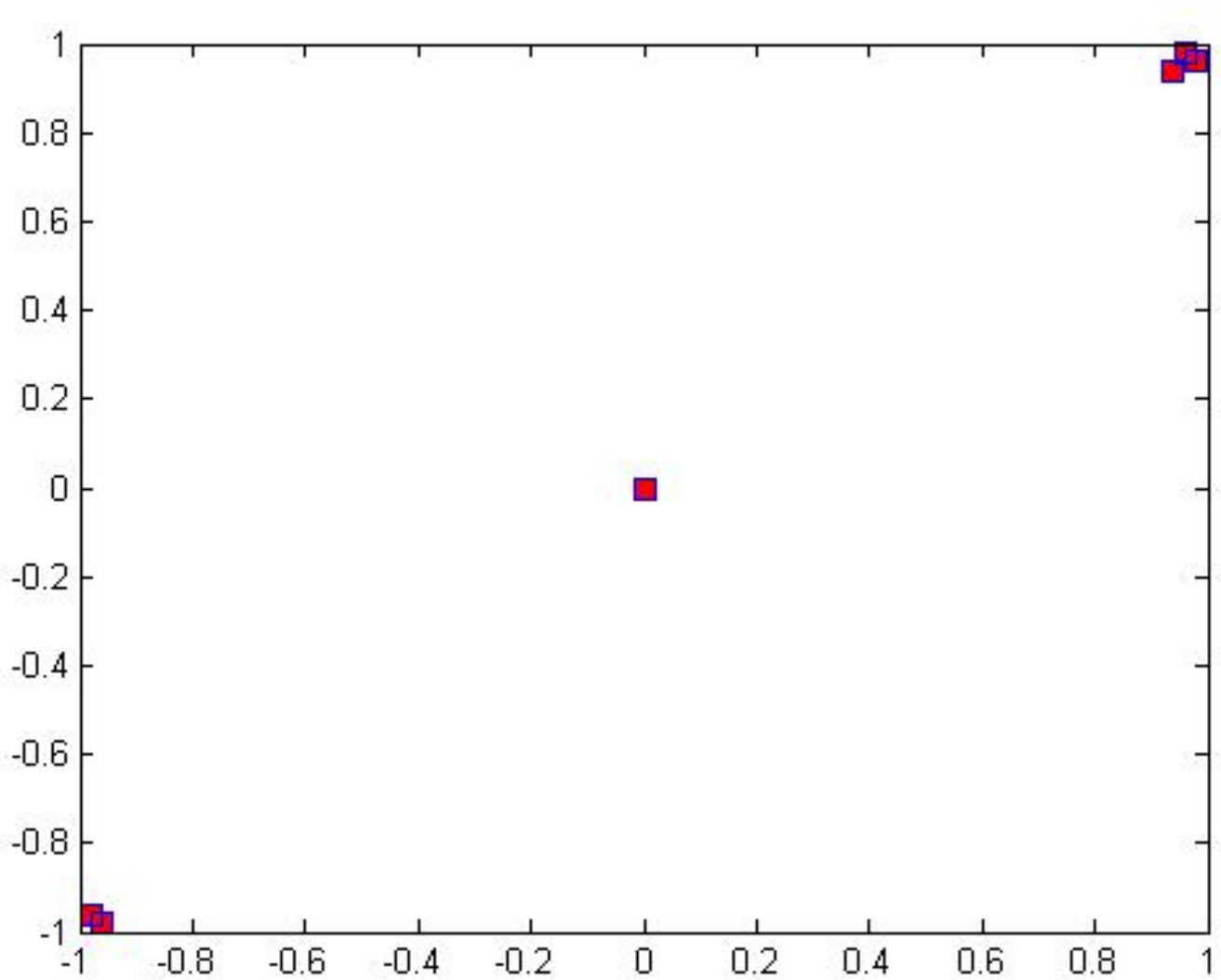
- Estimate
$$C \;=\; \max_{y \in Y} \max_{z \in B} |\mathcal{L}_y(z)|.$$
- If $C > \Lambda$ then let y^{out} correspond to the polynomial where the maximum was attained. Let
$$y^{in} \;\in\; \text{argmax}_{z \in B} |\mathcal{L}_{y^{out}}(z)|.$$
Update Y (and the Lagrange polynomials):
$$Y \;\leftarrow\; Y \cup \{y^{in}\} \setminus \{y^{out}\}.$$
- Otherwise (i.e., $C \leq \Lambda$), Y is Λ–poised and stop.

Model improvement (Lagrange polynomials)

Theorem

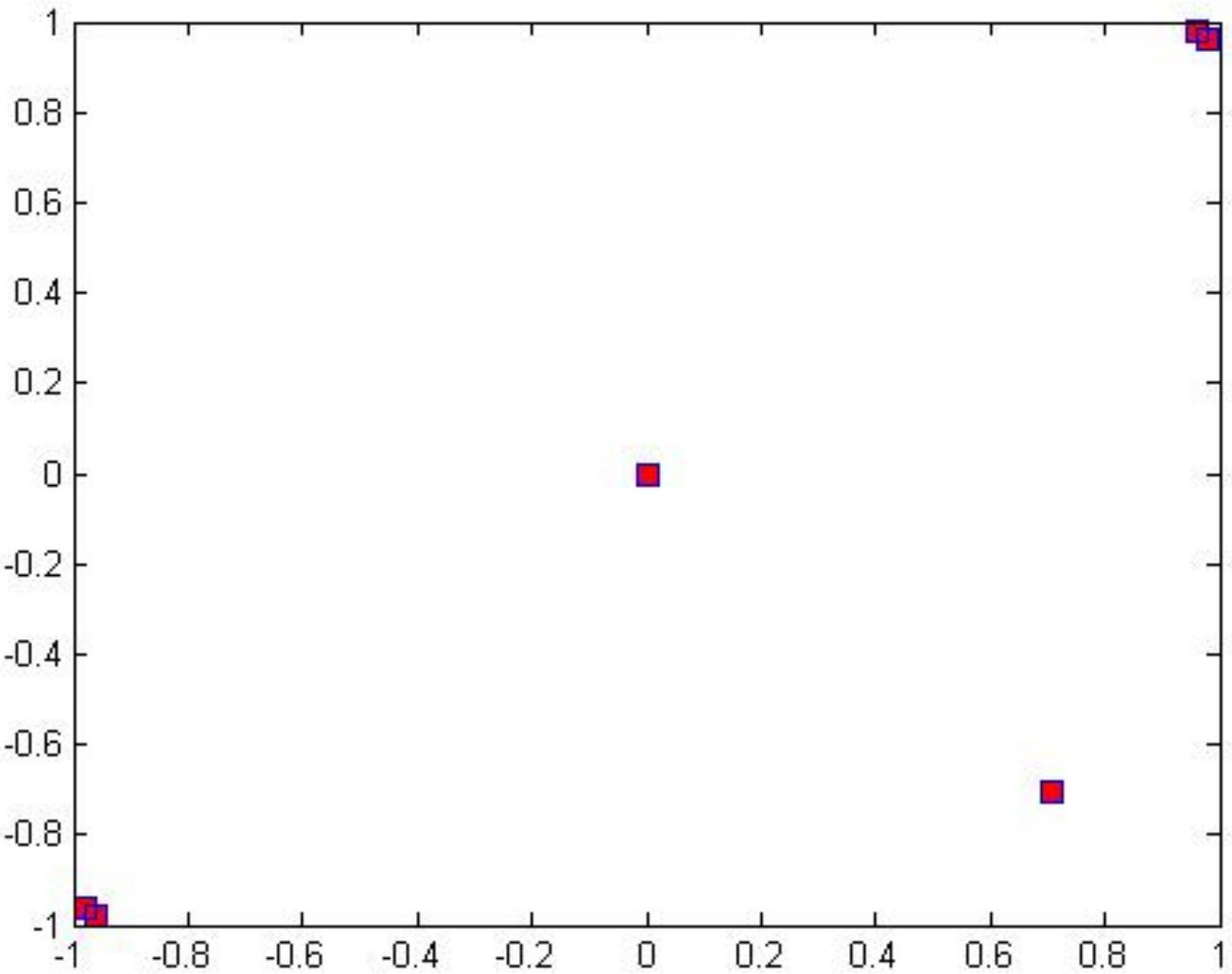
For any given $\Lambda > 1$ and a closed ball B, the previous model-improvement algorithm terminates with a Λ–poised set Y after at most $N = N(\Lambda)$ iterations.

Example (model improvement based on LP)



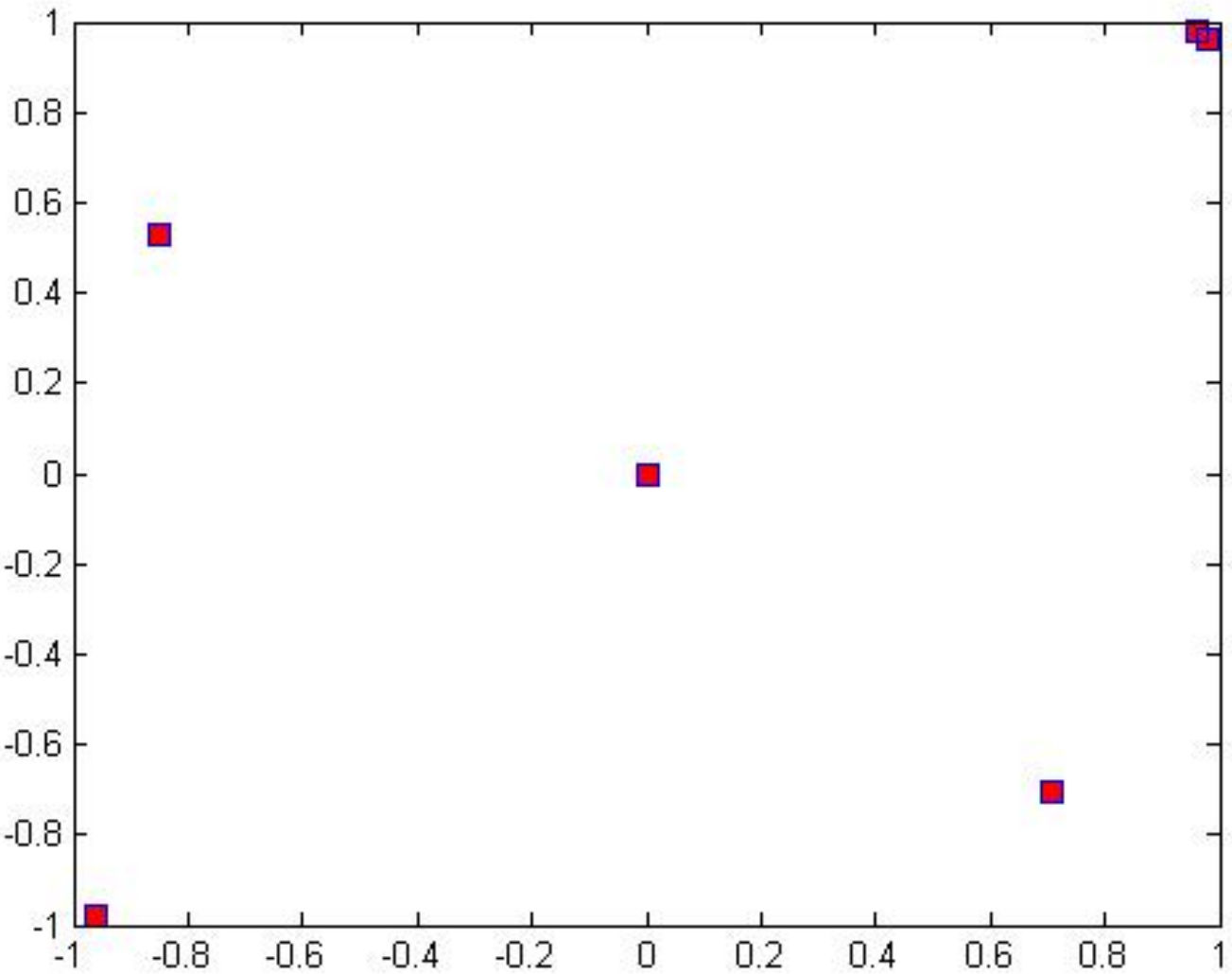
$C = 5324$

Example (model improvement based on LP)



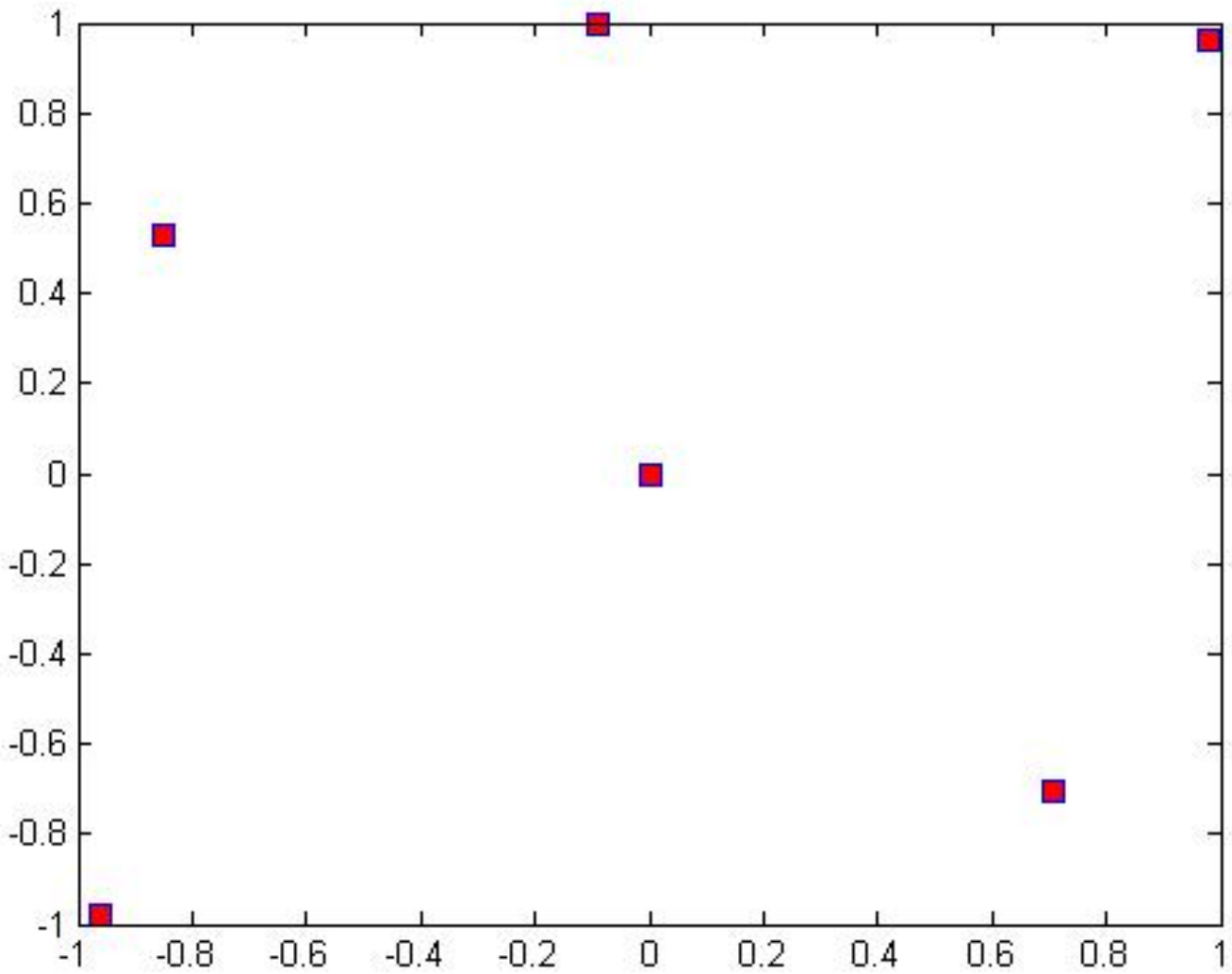
$C = 36.88$

Example (model improvement based on LP)



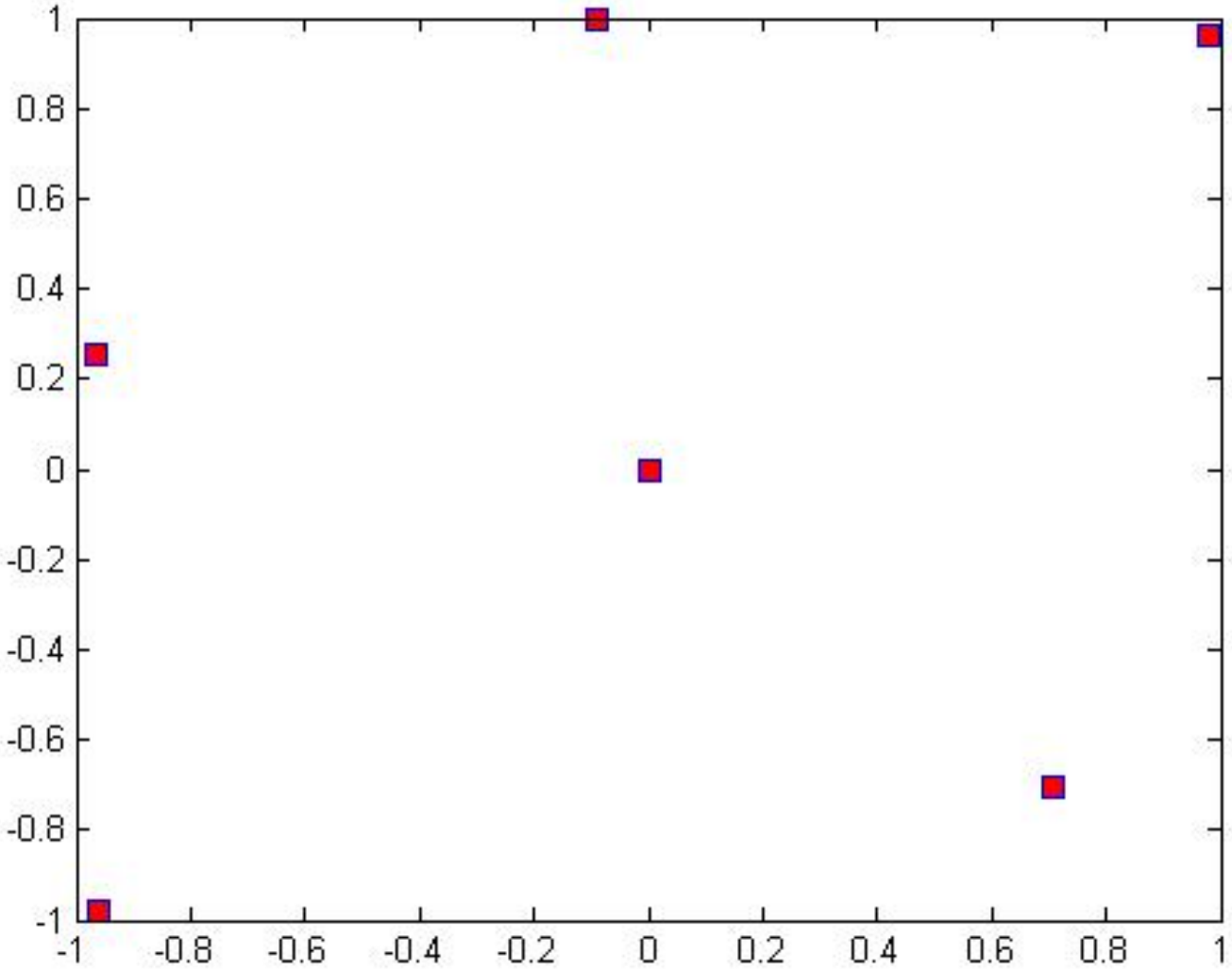
$C = 15.66$

Example (model improvement based on LP)



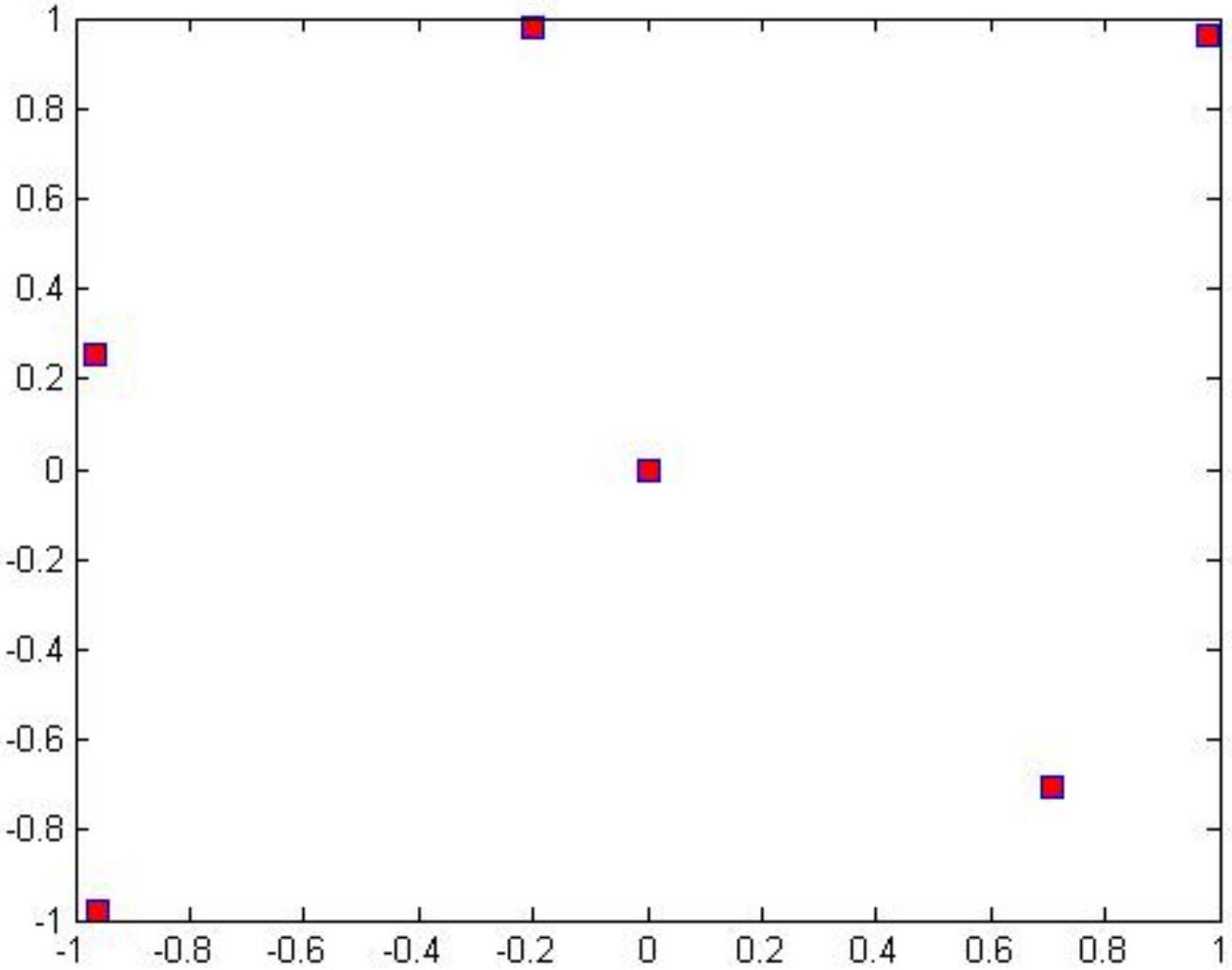
$C = 1.11$

Example (model improvement based on LP)



$C = 1.01$

Example (model improvement based on LP)



$C = 1.001$

Geometry constants

The geometry constant can also be given by

$$C(Y) \;=\; \text{cond}(\text{scaled interpolation matrix } M)$$

$\longrightarrow$ leading to model-improvement algorithms based on pivotal factorizations (LU/QR or Newton polynomials).

These algorithms yield:

$$\|M^{-1}\| \;\leq\; C(n)\,\frac{\varepsilon_{growth}}{\xi}$$

where

- ε_{growth} is the growth factor of the factorization.
- $\xi > 0$ is a (imposed) lower bound on the absolute value of the pivots.

  $\longrightarrow$ one knows that $\xi \leq 1/4$ for quadratics and $\xi \leq 1$ for linears.

Underdetermined polynomial models

Consider a underdetermined quadratic polynomial model built with less than $(n+1)(n+2)/2$ points.

Theorem

If Y is $C(Y)$–poised for linear interpolation or regression then

$$\|\nabla f(y) - \nabla m(y)\| \leq C(Y)\,[C(f) + \|H\|]\,\Delta \qquad \forall y \in B(x;\Delta)$$

where H is the Hessian of the model.

$\longrightarrow$ Thus, one should minimize the norm of H.

Minimum Frobenius norm models

Motivation comes from:

- The models are built by minimizing the entries of the Hessian (in the Frobenius norm) subject to the interpolation conditions.

  In this case, one can prove that H is bounded:

$$\|H\| \leq C(n)C(f)C(Y).$$

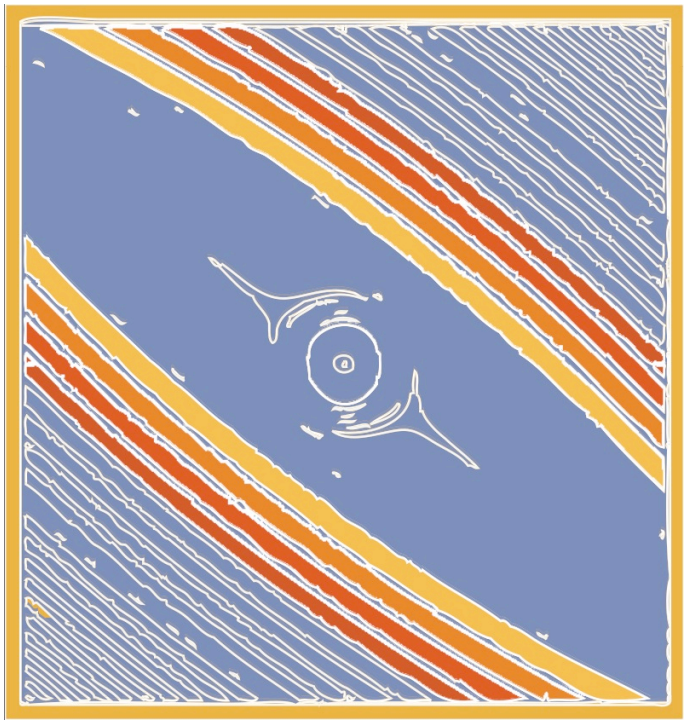
- Or they can be built by minimizing the difference between the current and previous Hessians (in the Frobenius norm) subject to the interpolation conditions.

  In this case, one can show that if f is itself a quadratic then:

$$\|H - \nabla^2 f\| \leq \|H^{old} - \nabla^2 f\|.$$

Presentation outline

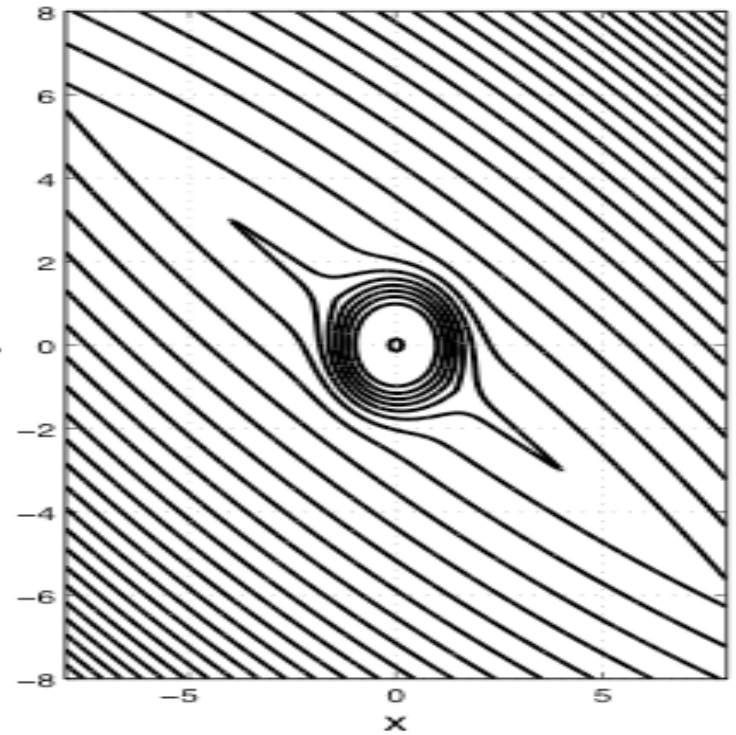
Limitations of GPS

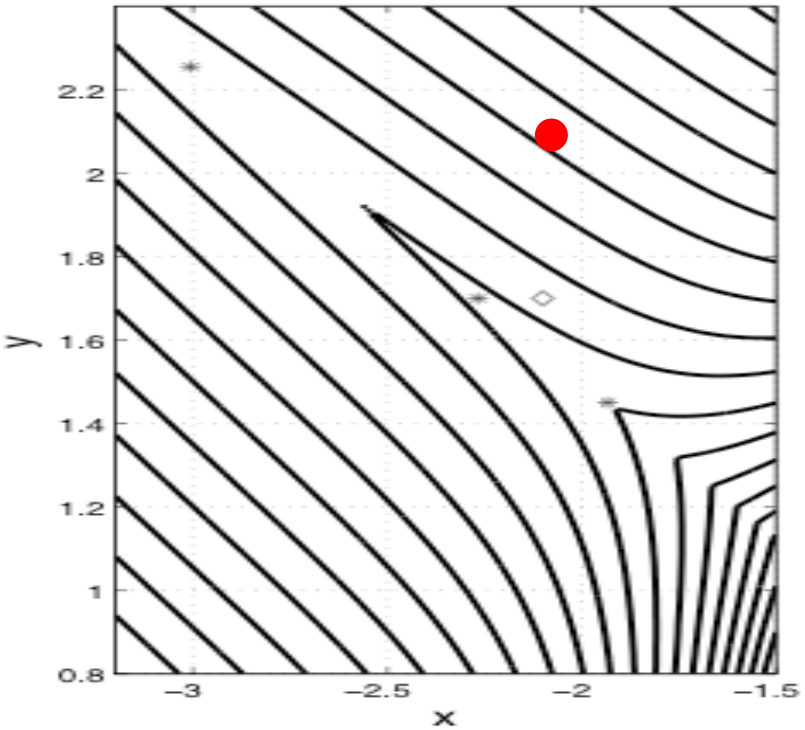


Optimization applied to the meeting logo.

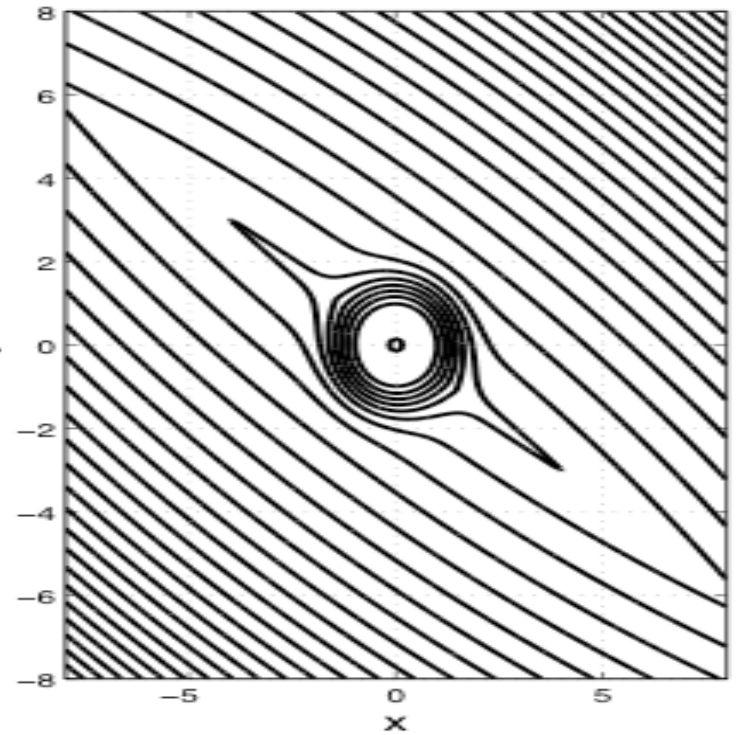
Level sets of $f : \mathbb{R}^2 \to \mathbb{R}$: $f(x) = \left(1 - e^{-\|x\|^2}\right) \max\{\|x - c\|^2, \|x - d\|^2\}$

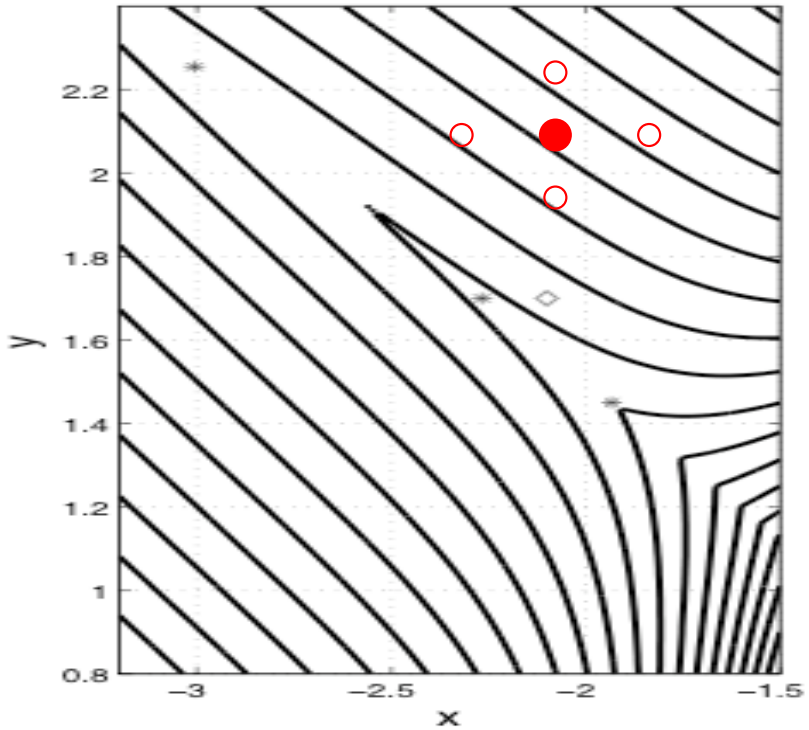
Limitations of GPS



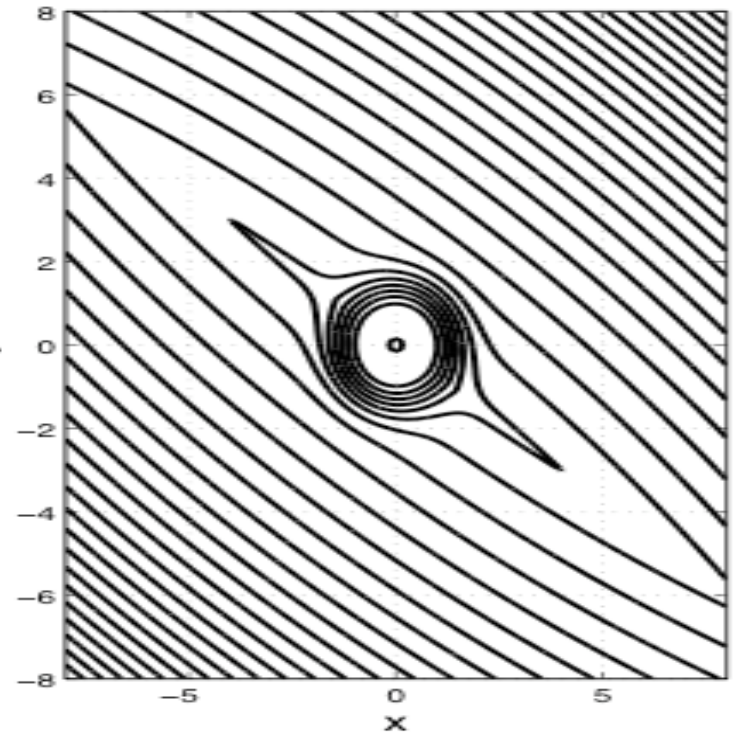


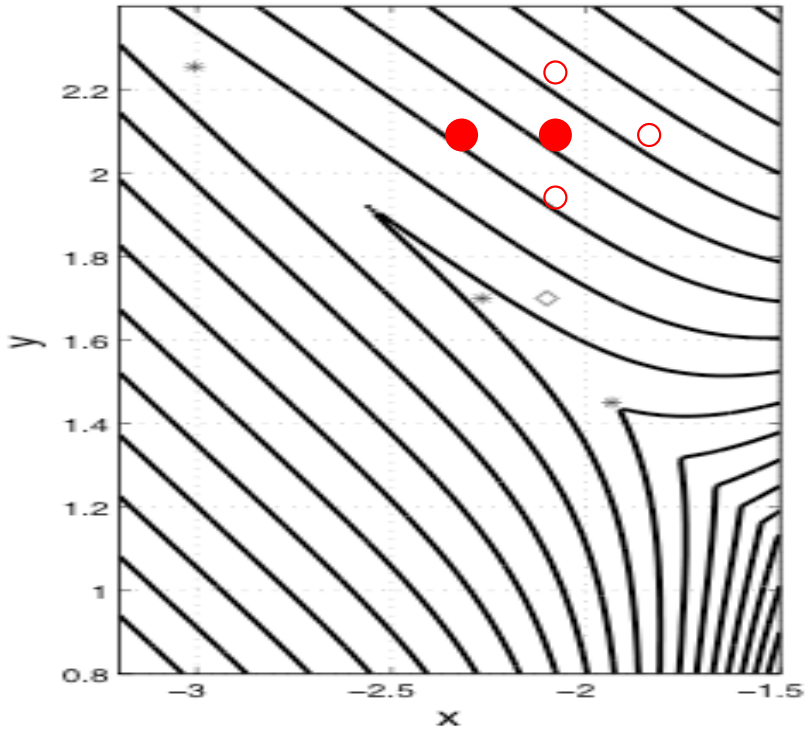
Limitations of GPS



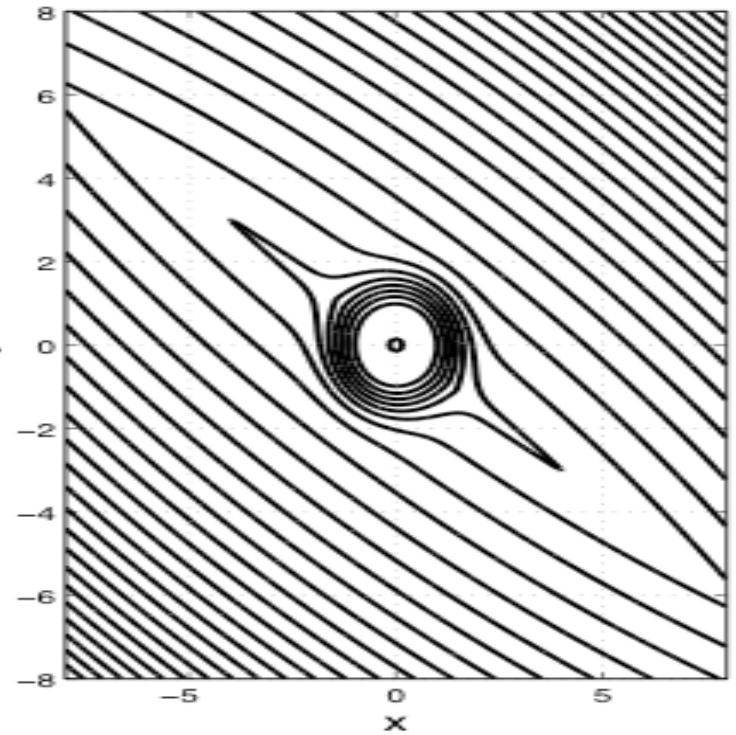


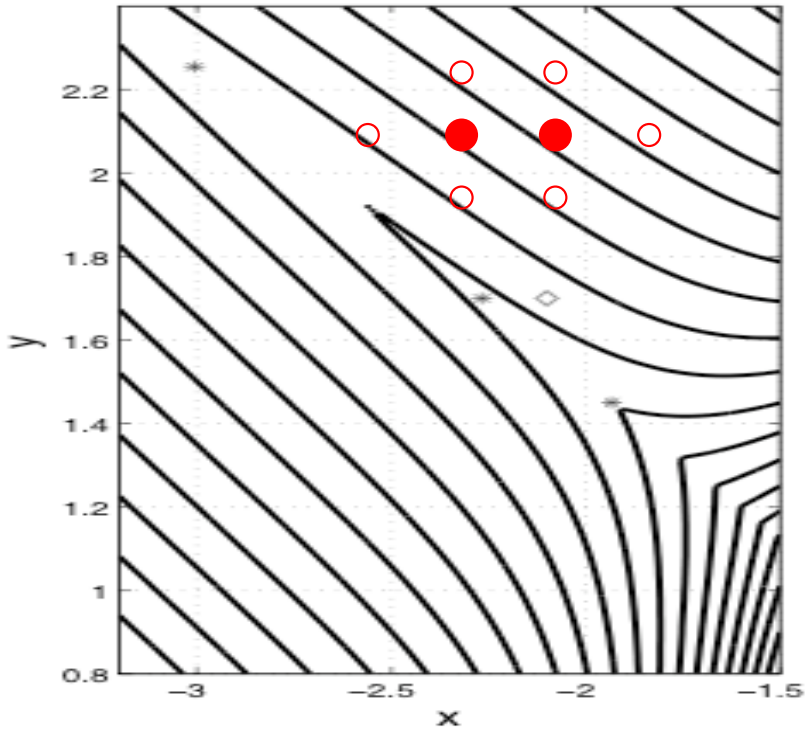
Limitations of GPS



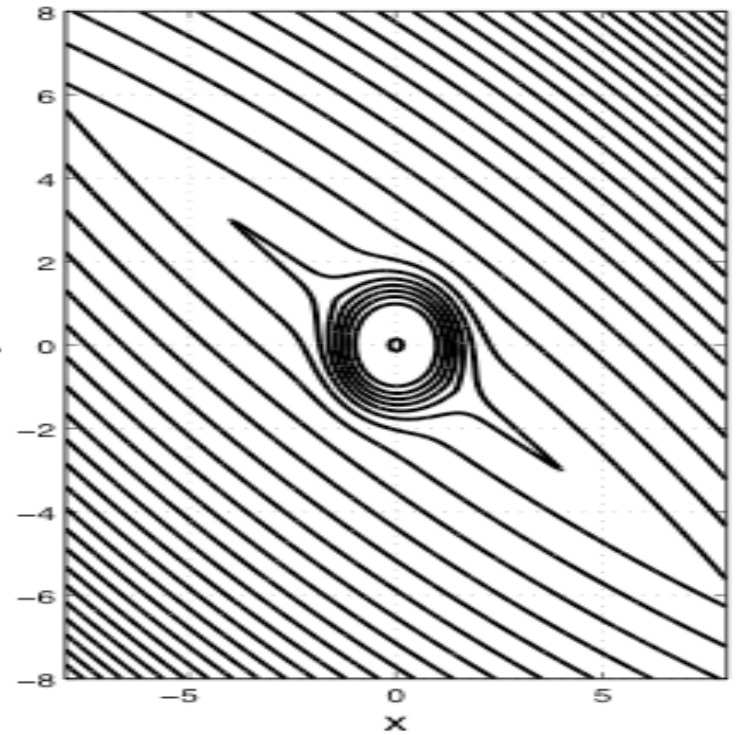


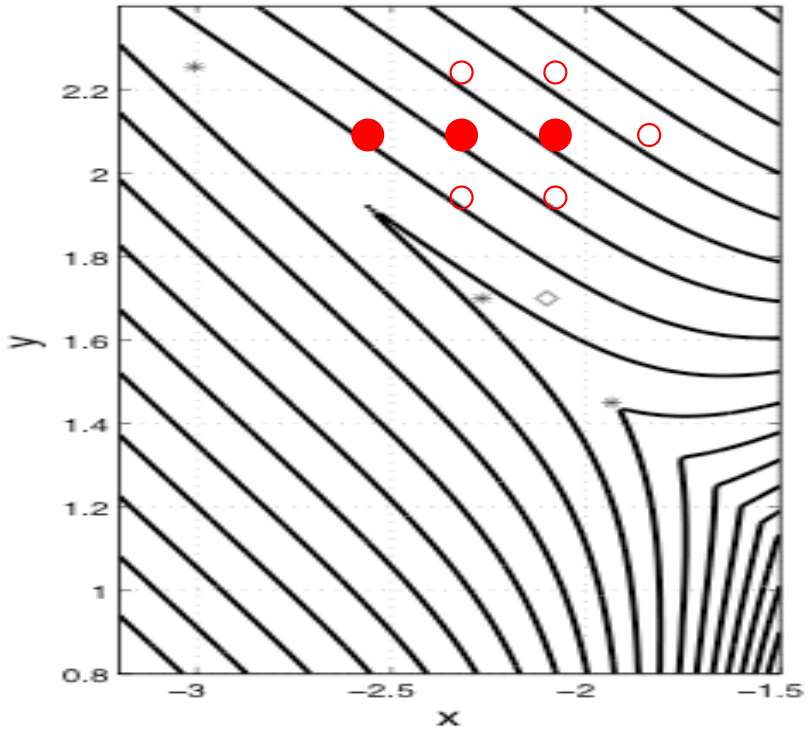
Limitations of GPS



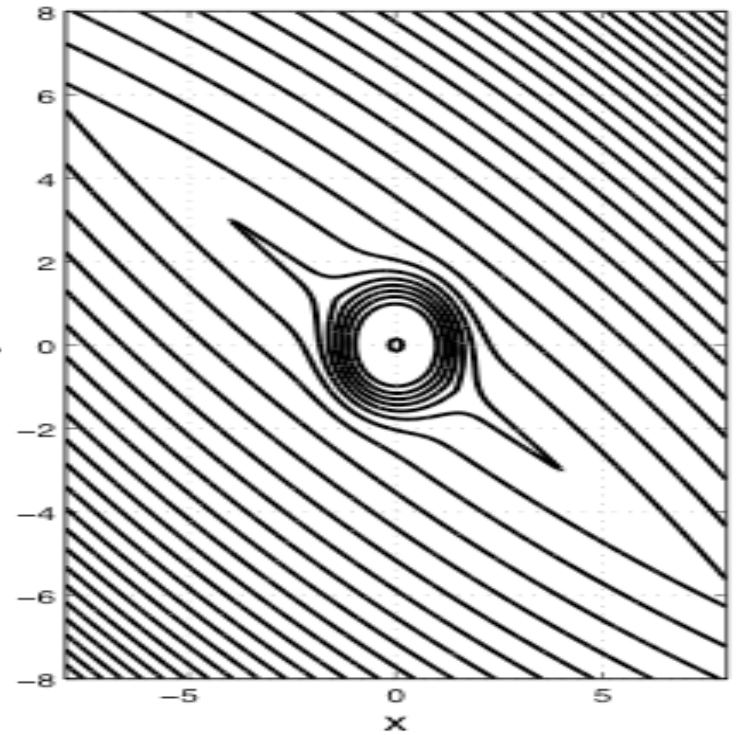


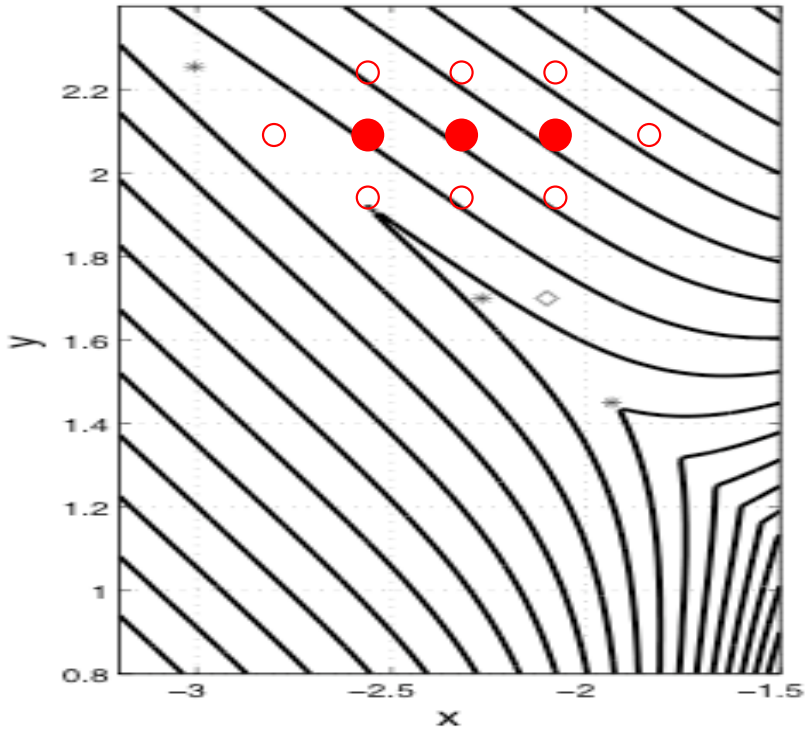
Limitations of GPS



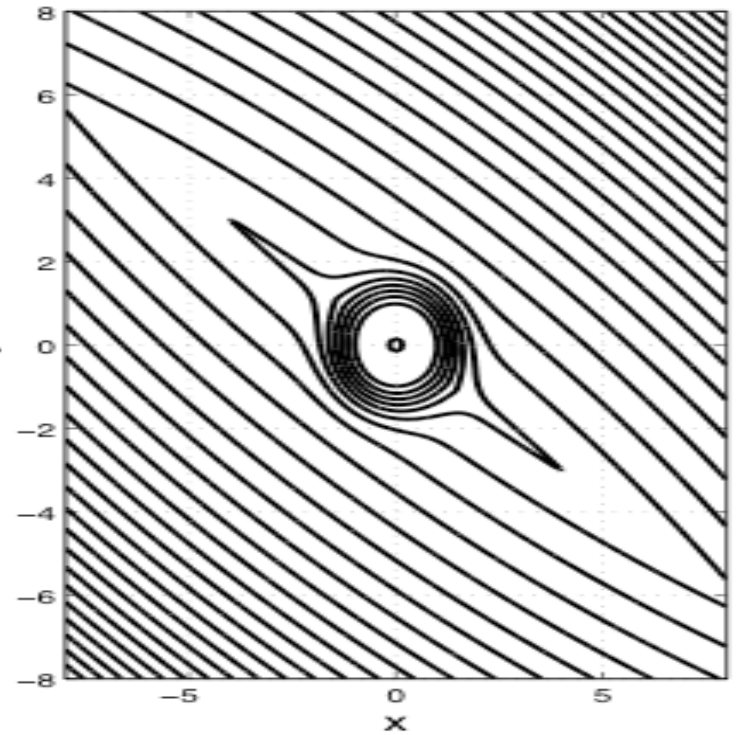


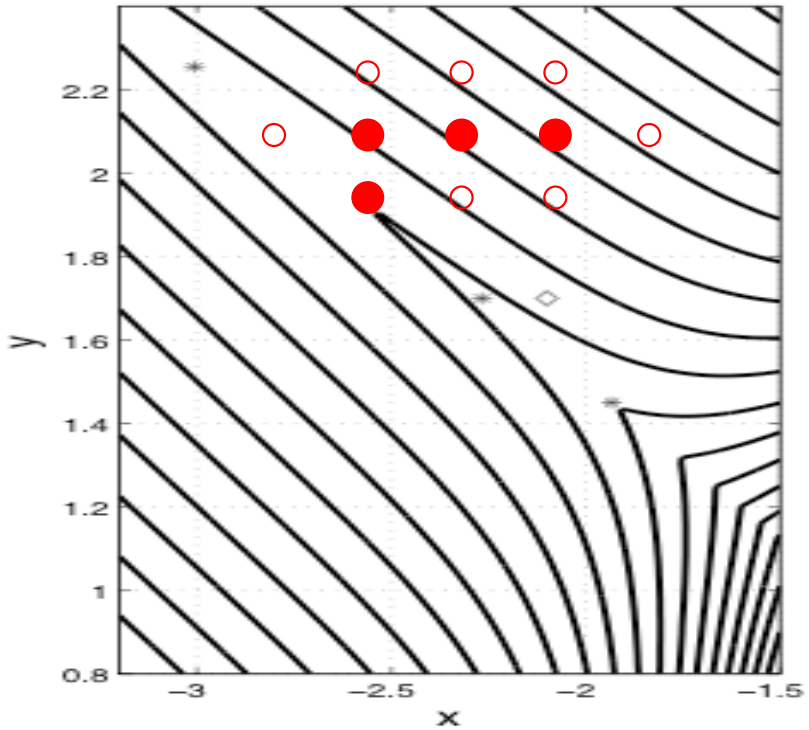
Limitations of GPS



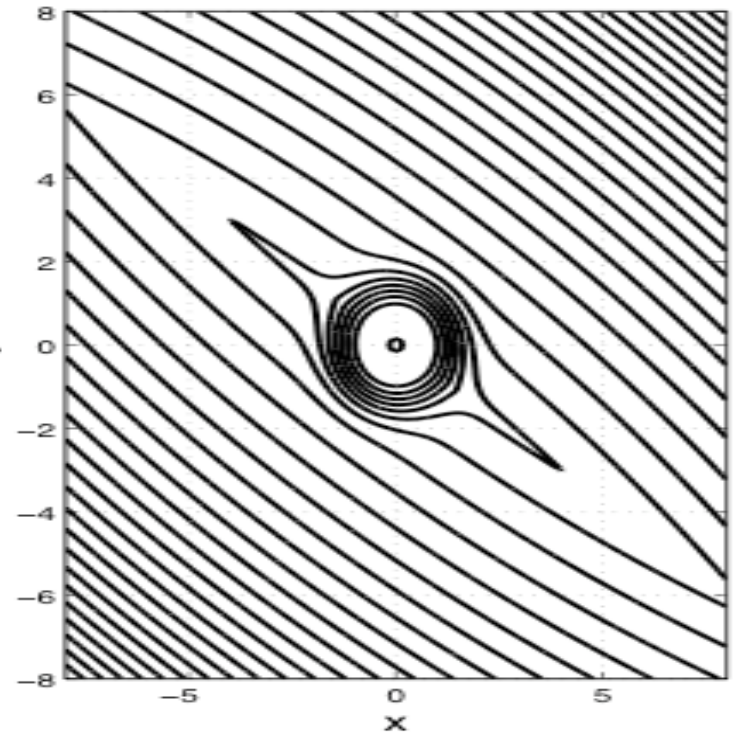


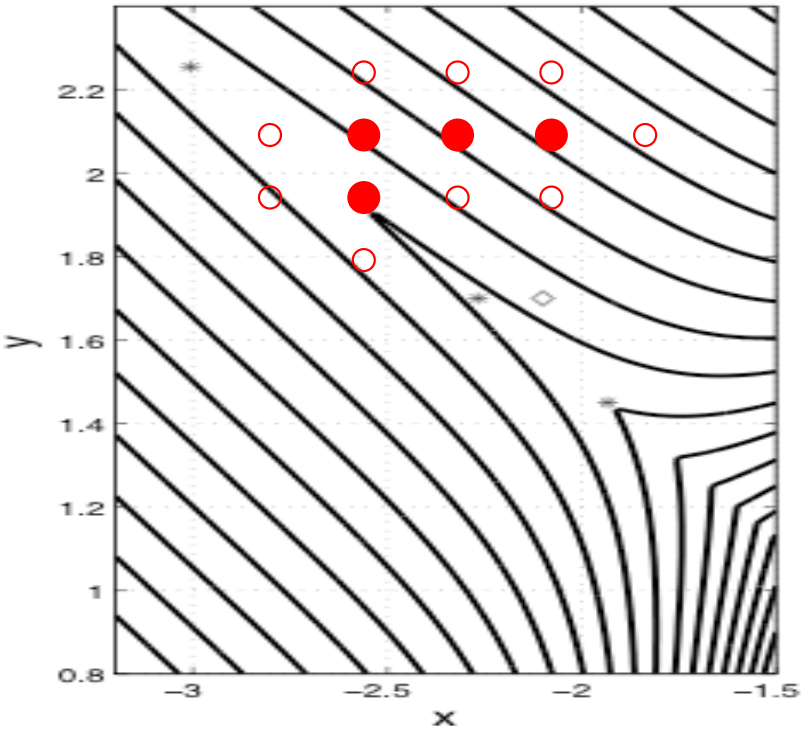
Limitations of GPS



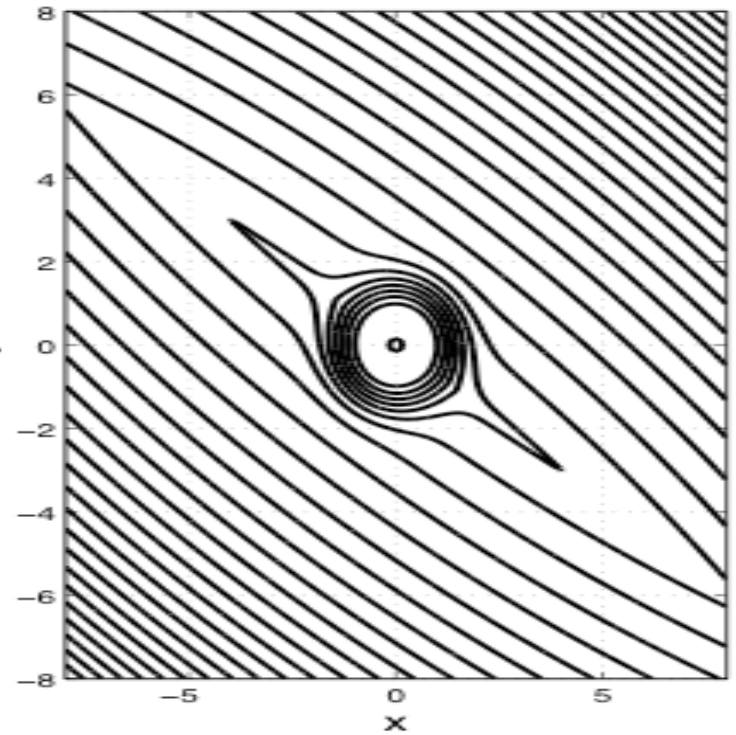


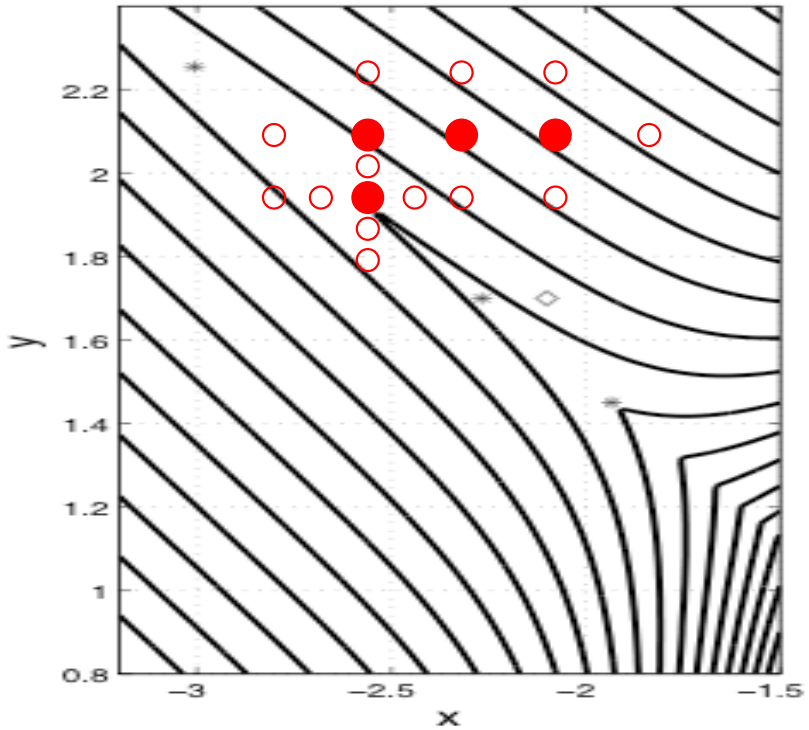
Limitations of GPS



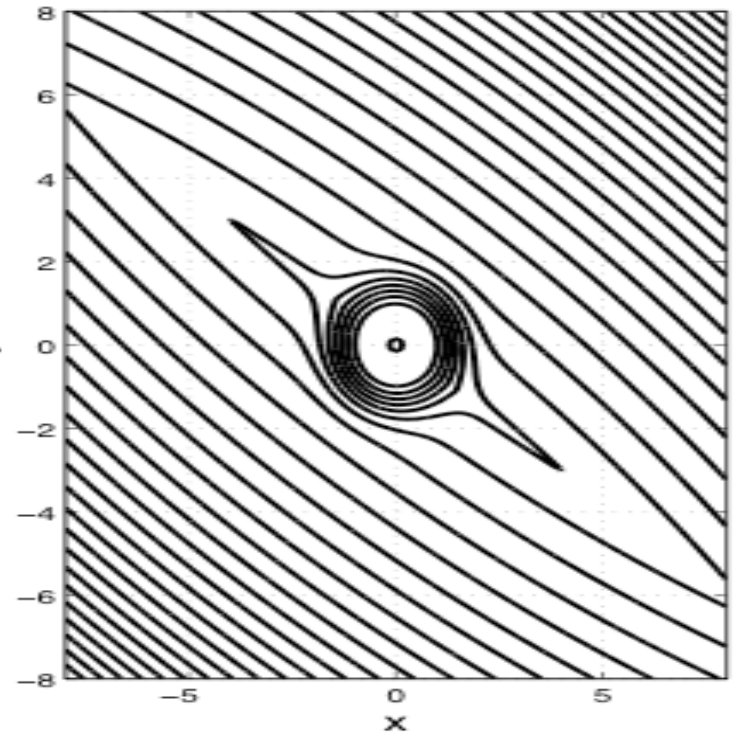


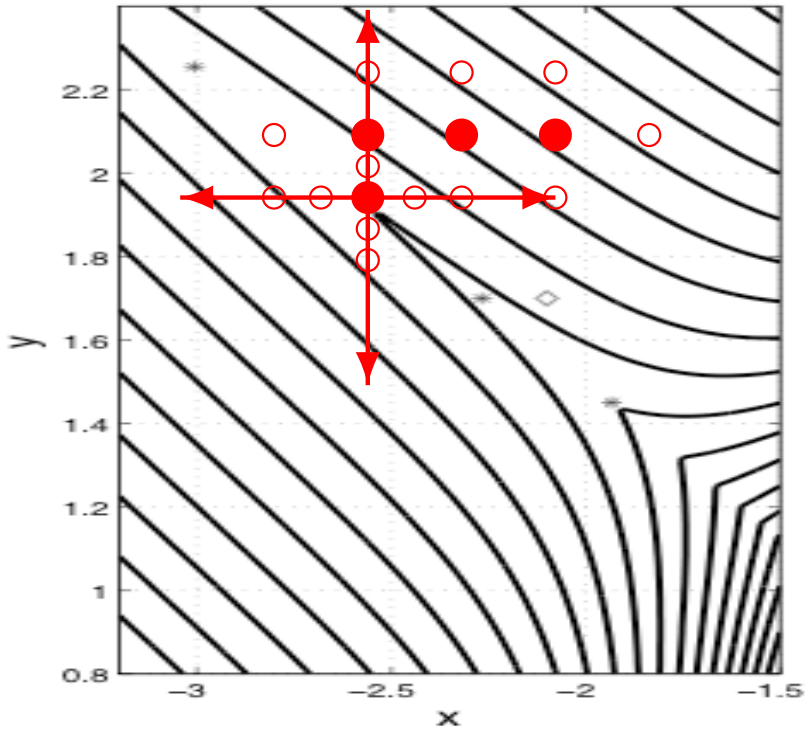
Limitations of GPS





Limitations of GPS

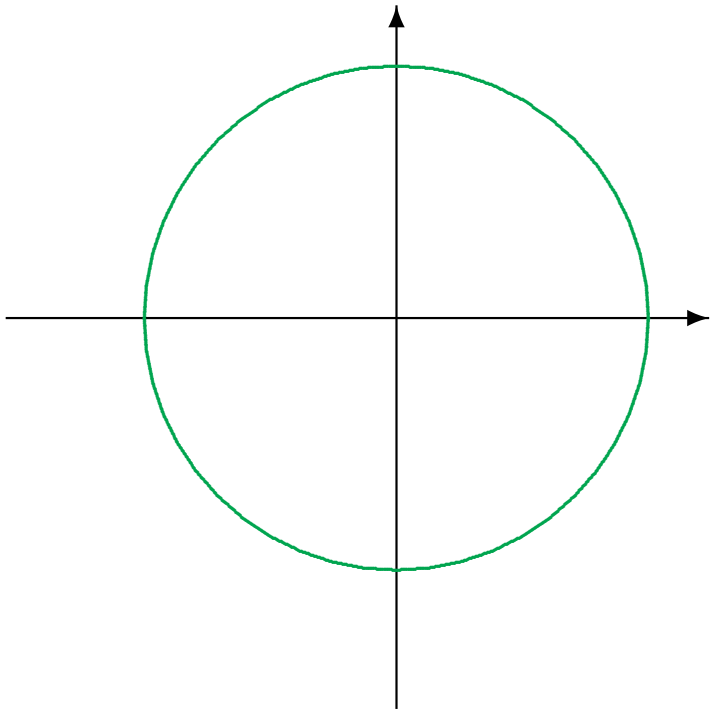




For almost all starting points, the iterates of GPS with the coordinate directions converge to a point $\hat{x}$ on the diagonal, where f is not differentiable, with $f'(\hat{x}; \pm e_i) \geq 0$
but $f'(\hat{x}; d) < 0$ with $d = (1, -1)^T$.

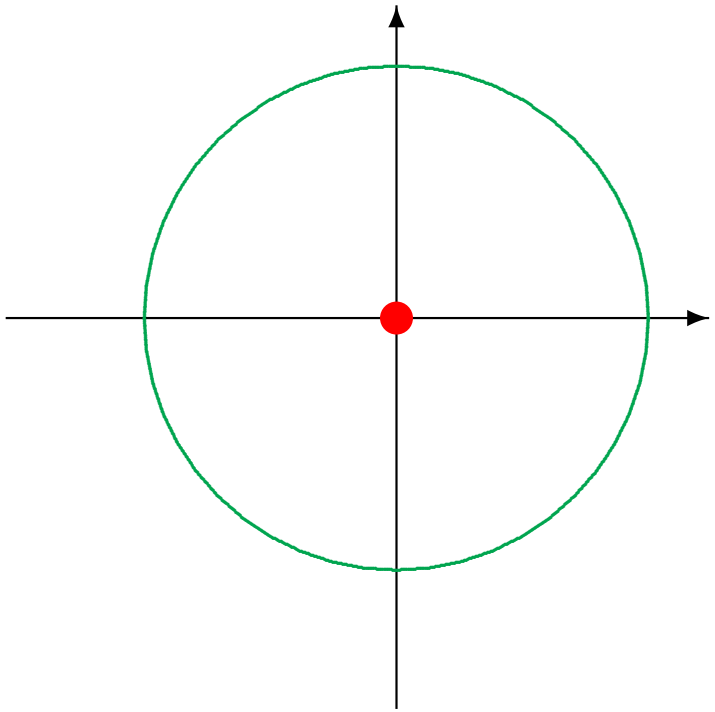
Limitations of GPS - a convex problem

Minimize $f(x) = x_1 + x_2$ on a disc



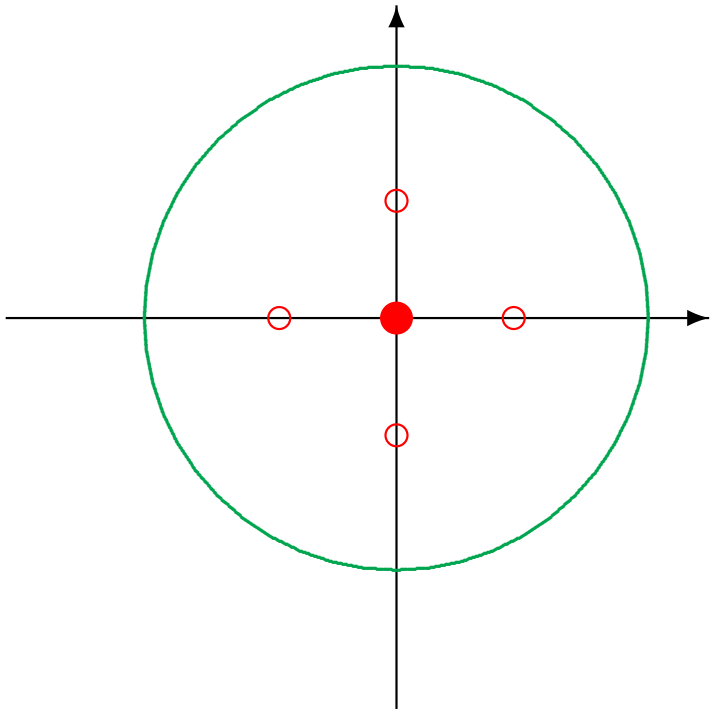
Limitations of GPS - a convex problem

Minimize $f(x) = x_1 + x_2$ on a disc



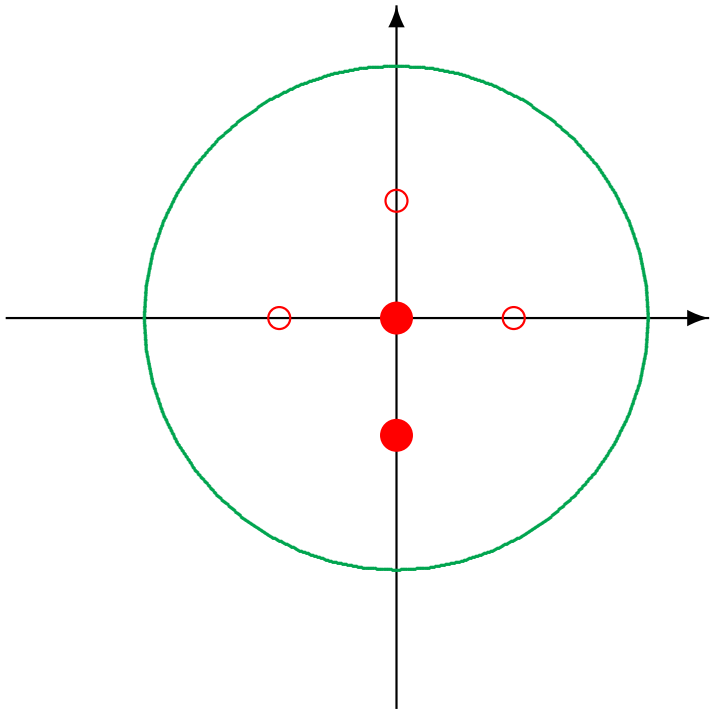
Limitations of GPS - a convex problem

Minimize $f(x) = x_1 + x_2$ on a disc



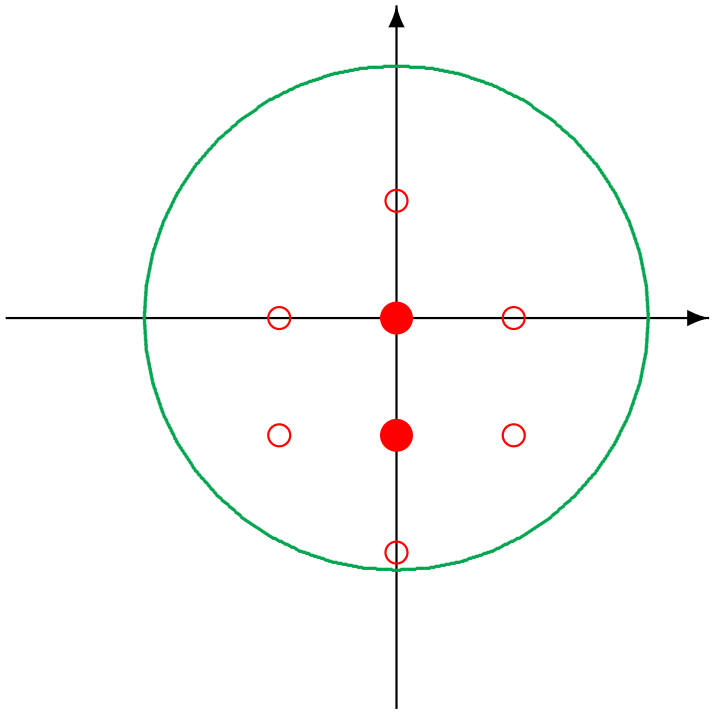
Limitations of GPS - a convex problem

Minimize $f(x) = x_1 + x_2$ on a disc



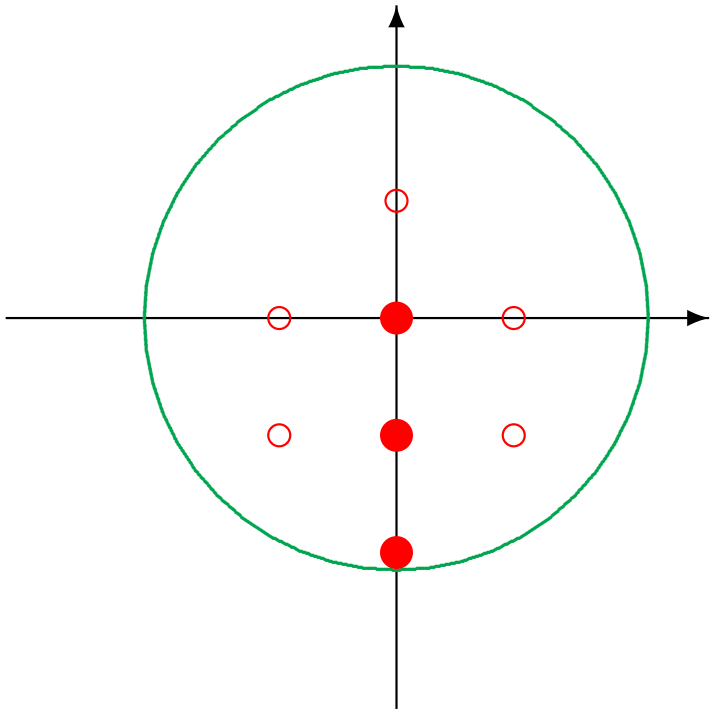
Limitations of GPS - a convex problem

Minimize $f(x) = x_1 + x_2$ on a disc



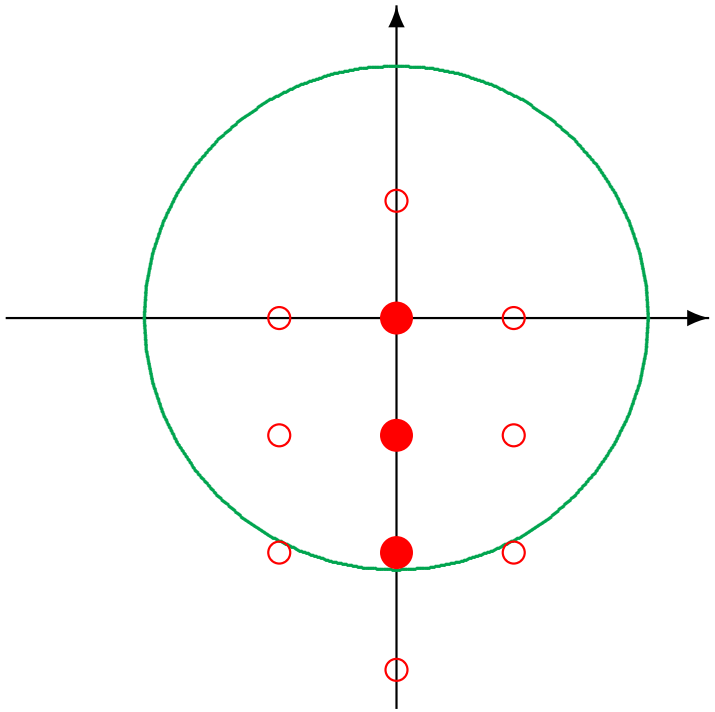
Limitations of GPS - a convex problem

Minimize $f(x) = x_1 + x_2$ on a disc



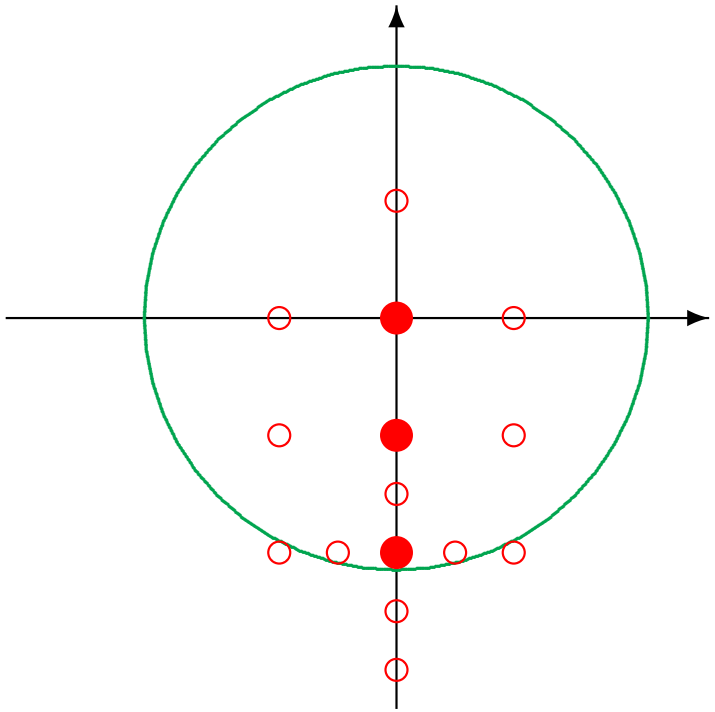
Limitations of GPS - a convex problem

Minimize $f(x) = x_1 + x_2$ on a disc



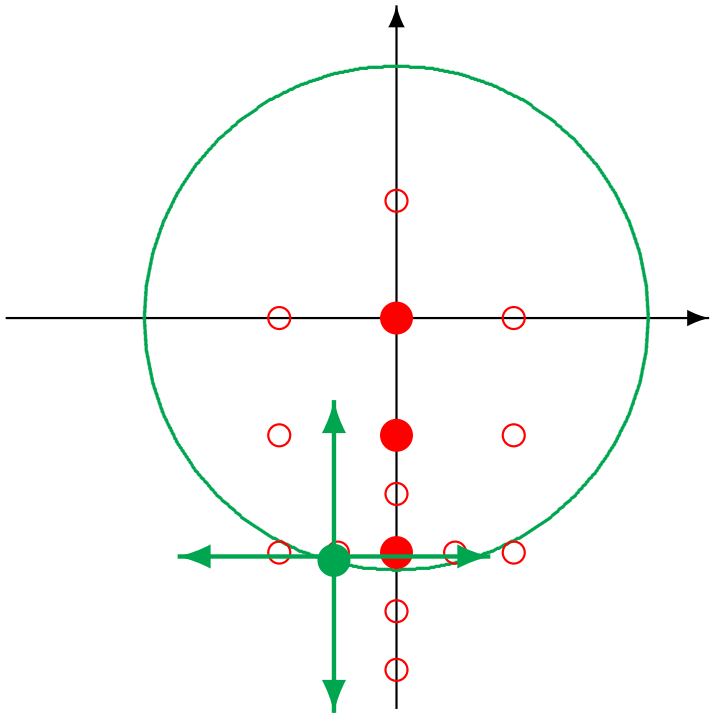
Limitations of GPS - a convex problem

Minimize $f(x) = x_1 + x_2$ on a disc



Limitations of GPS - a convex problem

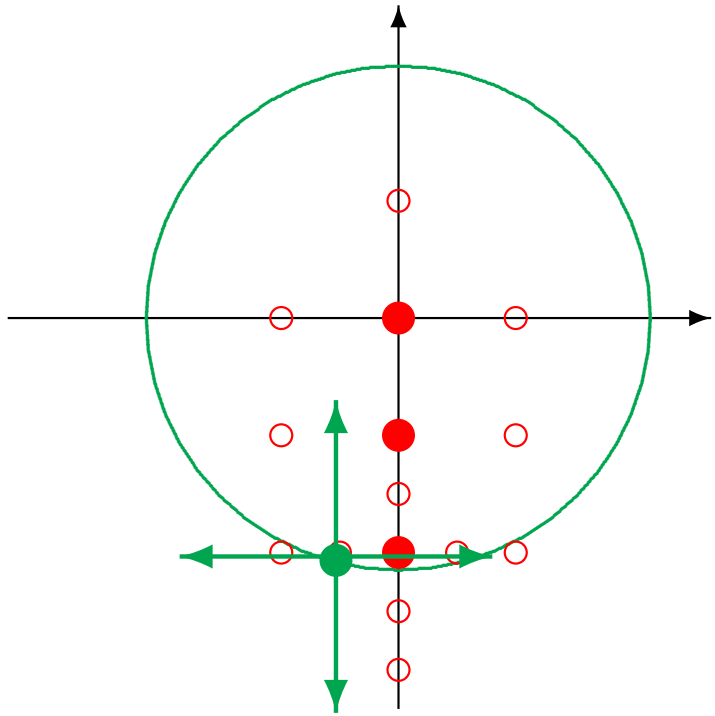
Minimize $f(x) = x_1 + x_2$ on a disc



Infeasible trial points are simply rejected from consideration.
This is called the extreme barrier approach.

Limitations of GPS - a convex problem

Minimize $f(x) = x_1 + x_2$ on a disc



Infeasible trial points are simply rejected from consideration.
This is called the extreme barrier approach.
The GPS iterates with the coordinate directions converge to a suboptimal point $\hat{x}$ on the boundary, where $f'(\hat{x}; +e_i) \geq 0$ and $\hat{x} - e_i \notin \Omega$.

Unconstrained nonsmooth optimality conditions

Given any unconstrained optimization problem (NLP), we desire an algorithm that produces a solution $\hat{x}$

Unconstrained nonsmooth optimality conditions

Given any unconstrained optimization problem (NLP), we desire an algorithm that produces a solution $\hat{x}$

Unconstrained optimization hierarchy of optimality conditions :

if f is $\mathcal{C}^1$	then $0 = \nabla f(\hat{x})$	$\Leftrightarrow$	$f'(\hat{x}; d) \geq 0 \ \forall d \in \mathbb{R}^n$
if f is regular	then		$f'(\hat{x}; d) \geq 0 \ \forall d \in \mathbb{R}^n$
if f is convex	then $0 \in \underline{\partial} f(\hat{x})$		
if f is Lipschitz near $\hat{x}$	then $0 \in \partial f(\hat{x})$	$\Leftrightarrow$	$f^\circ(\hat{x}; d) \geq 0 \ \forall d \in \mathbb{R}^n$.

Clarke derivatives and generalized gradient

Let f be Lipschitz near $x \in \mathbb{R}^n$.

- The Clarke generalized devivative of f at x in the direction $v \in \mathbb{R}^n$ is

$$f^\circ(x;v) = \limsup_{y \to x,\ t \downarrow 0} \frac{f(y+tv) - f(y)}{t}.$$

- The generalized gradient of f at x is defined to be

$$\begin{aligned} \partial f(x) &= \{\zeta \in \mathbb{R}^n : f^\circ(x;v) \geq v^T\zeta \text{ for every } v \in \mathbb{R}^n\} \\ &= \text{co}\{\lim \nabla f(x_i) \ : \ x_i \to x \text{ and } \nabla f(x_i) \text{ exists } \}. \end{aligned}$$

Clarke derivatives and generalized gradient

Let f be Lipschitz near $x \in \mathbb{R}^n$.

- The Clarke generalized devivative of f at x in the direction $v \in \mathbb{R}^n$ is

$$f^\circ(x;v) = \limsup_{y \to x,\ t \downarrow 0} \frac{f(y+tv) - f(y)}{t}.$$

- The generalized gradient of f at x is defined to be

$$\begin{aligned} \partial f(x) &= \{\zeta \in \mathbb{R}^n : f^\circ(x;v) \geq v^T\zeta \text{ for every } v \in \mathbb{R}^n\} \\ &= \text{co}\{\lim \nabla f(x_i) \ : \ x_i \to x \text{ and } \nabla f(x_i) \text{ exists } \}. \end{aligned}$$

Properties:

- If f is convex, $\partial f(x) =$ sub-gradient.
- f is *strictly differentiable* at x if $\partial f(x)$ contains a single element, and that element is $\nabla f(x)$, and $f'(x;\cdot) = f^\circ(x;\cdot)$.

Constrained optimization optimality conditions

Necessary optimality condition

If $\hat{x} \in \Omega$ is a local minimizer of a differentiable function f over a convex set $\Omega \subset \mathbb{R}^n$, then

$$f'(\hat{x}; d) \geq 0 \quad \forall\ d \in T_\Omega(\hat{x}),$$

where $f'(\hat{x}; d) = \lim_{t \to 0} \dfrac{f(\hat{x} + td) - f(\hat{x})}{t} = d^T \nabla f(\hat{x})$

and $T_\Omega(\hat{x})$ is the tangent cone to Ω at $\hat{x}$.

Constrained optimization optimality conditions

Necessary optimality condition

If $\hat{x} \in \Omega$ is a local minimizer of a differentiable function f over a convex set $\Omega \subset \mathbb{R}^n$, then

$$f'(\hat{x}; d) \geq 0 \quad \forall\, d \in T_\Omega(\hat{x}),$$

where $f'(\hat{x}; d) = \lim_{t \to 0} \dfrac{f(\hat{x} + td) - f(\hat{x})}{t} = d^T \nabla f(\hat{x})$
and $T_\Omega(\hat{x})$ is the tangent cone to Ω at $\hat{x}$.

Necessary optimality condition

If $\hat{x} \in \Omega$ is a local minimizer of the function f over the set $\Omega \subset \mathbb{R}^n$, then

$$f^\circ(\hat{x}; d) \geq 0 \quad \forall\, d \in T_\Omega^H(\hat{x}),$$

where $f^\circ(\hat{x}; d)$ is a generalization of the directional derivative,
and $T_\Omega^H(\hat{x})$ is a generalization of the tangent cone.

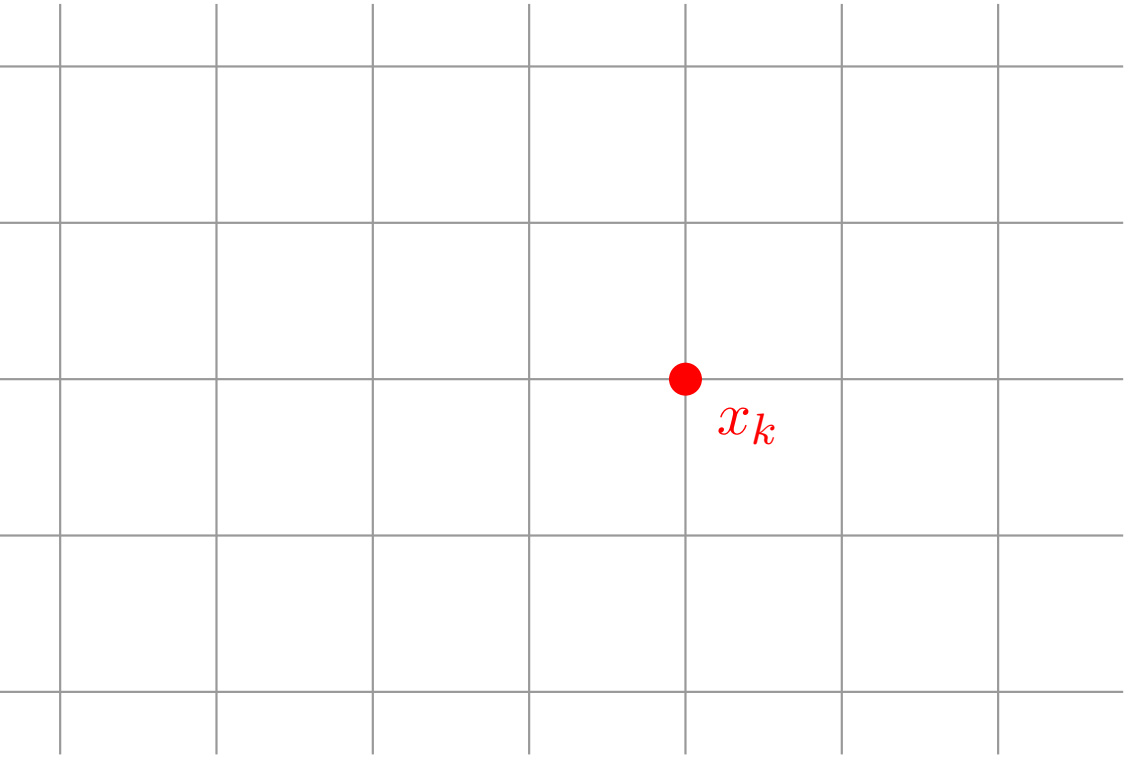
Extending GPS to handle nonlinear constraints

- For unconstrained optimization, GPS garantees (under a local Lipschitz assumption) to produce a limit point x such that $f^\circ(x;d) \geq 0$ for every d used infinitely often.
  Unfortunately, the d's are selected from a fixed finite set D, and GPS may miss important directions. The effect is more pronounced as the dimension increases.
- Torczon and Lewis show how to adapt GPS to explicit bound or linear inequalities. The directions in D are constructed using the nearby active constraints.
- GPS is not suited for nonlinear constraints.

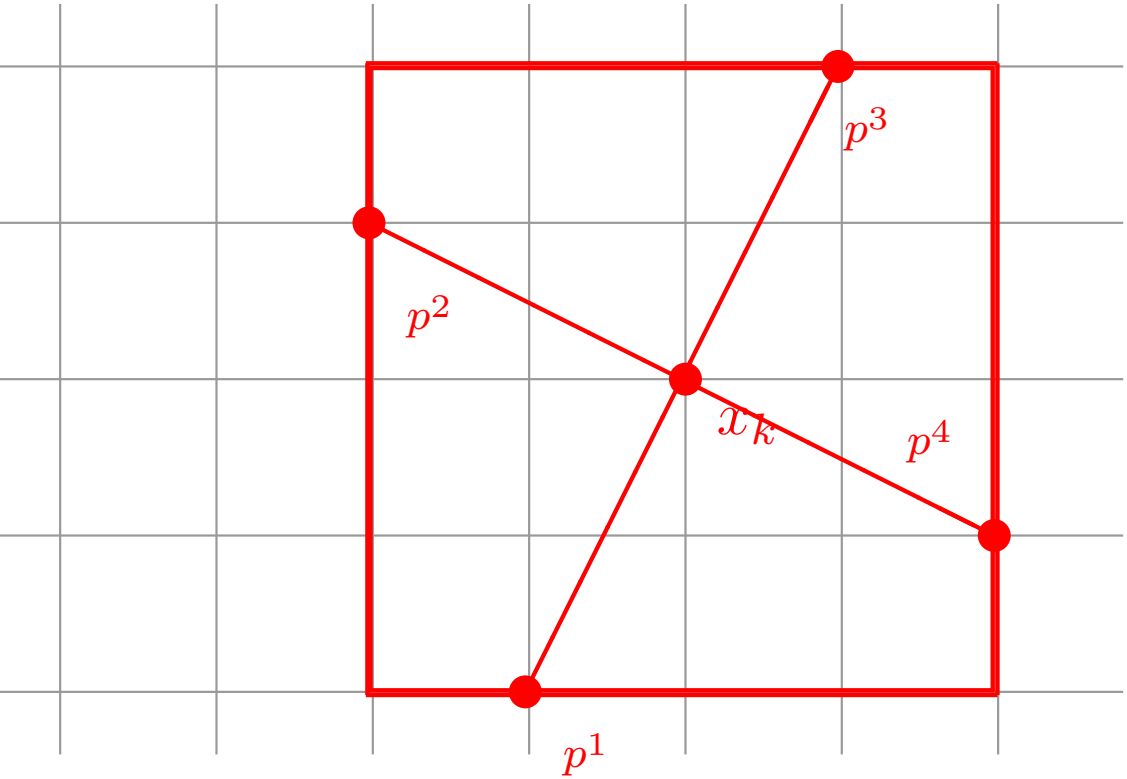
Extending GPS to handle nonlinear constraints

- For unconstrained optimization, GPS garantees (under a local Lipschitz assumption) to produce a limit point x such that $f^\circ(x;d) \geq 0$ for every d used infinitely often.
  Unfortunately, the d's are selected from a fixed finite set D, and GPS may miss important directions. The effect is more pronounced as the dimension increases.
- Torczon and Lewis show how to adapt GPS to explicit bound or linear inequalities. The directions in D are constructed using the nearby active constraints.
- GPS is not suited for nonlinear constraints.
- Recently, Kolda, Lewis and Torczon proposed an augmented Lagrangean GSS approach, analyzed under the assumption that the objective and constraints be twice continuously differentiable.
- MADS generalizes GPS by allowing more directions. MADS is designed for both constrained or unconstrained optimization, and does not require any smoothness assumptions.

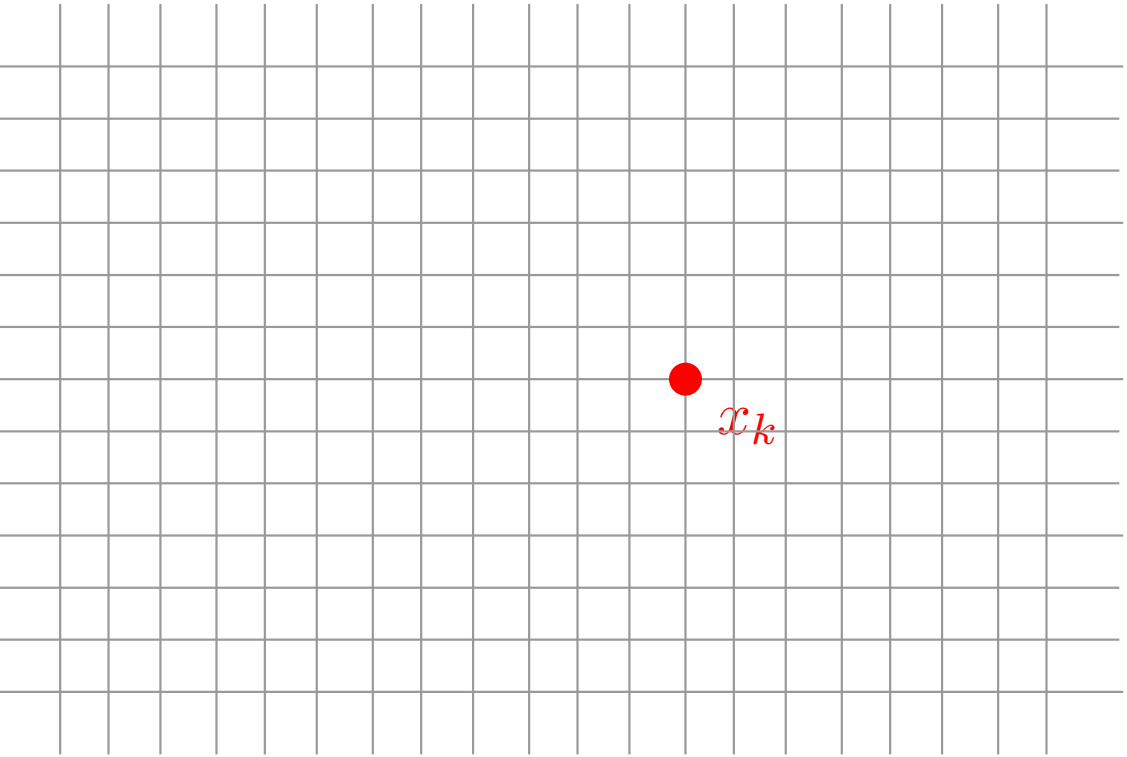
OrthoMads – 2008



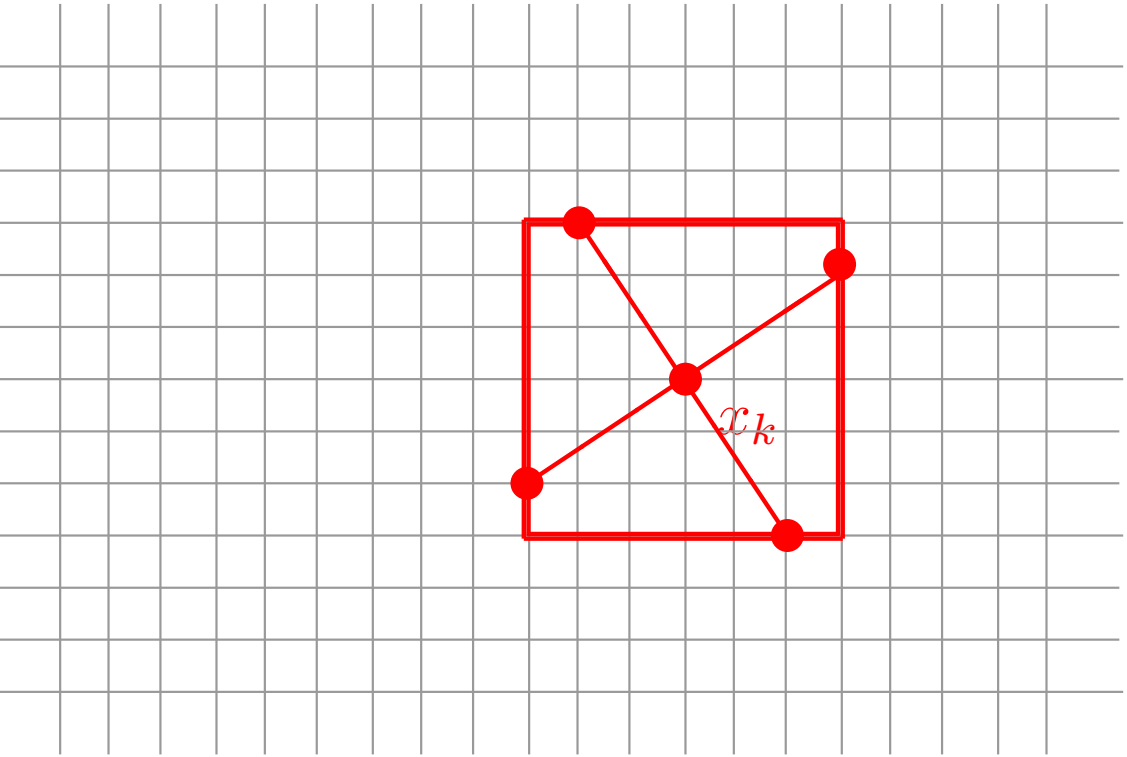
OrthoMads – 2008



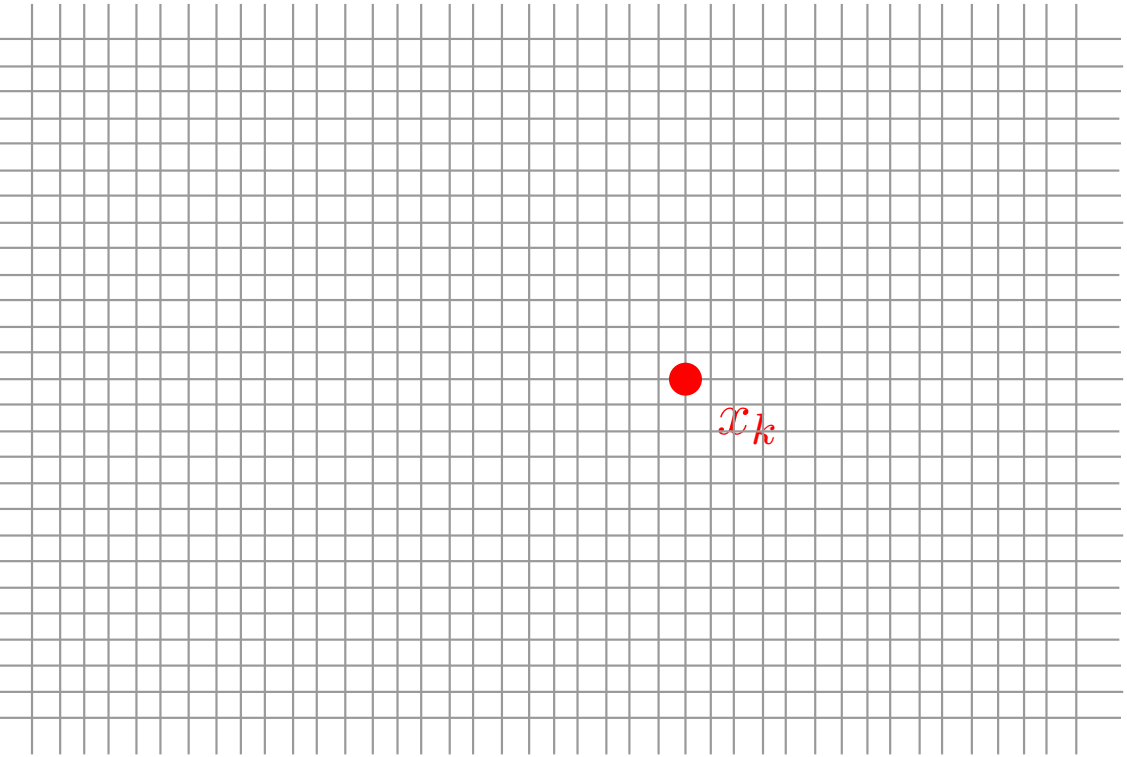
OrthoMads – 2008



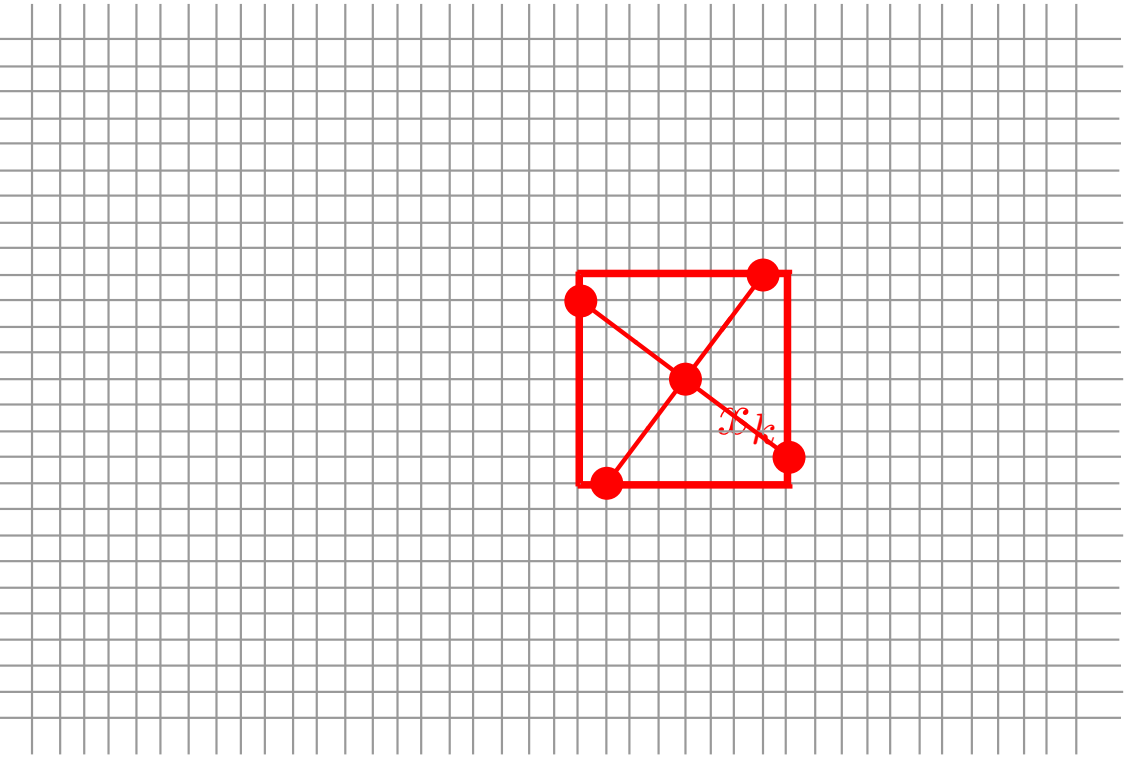
OrthoMads – 2008



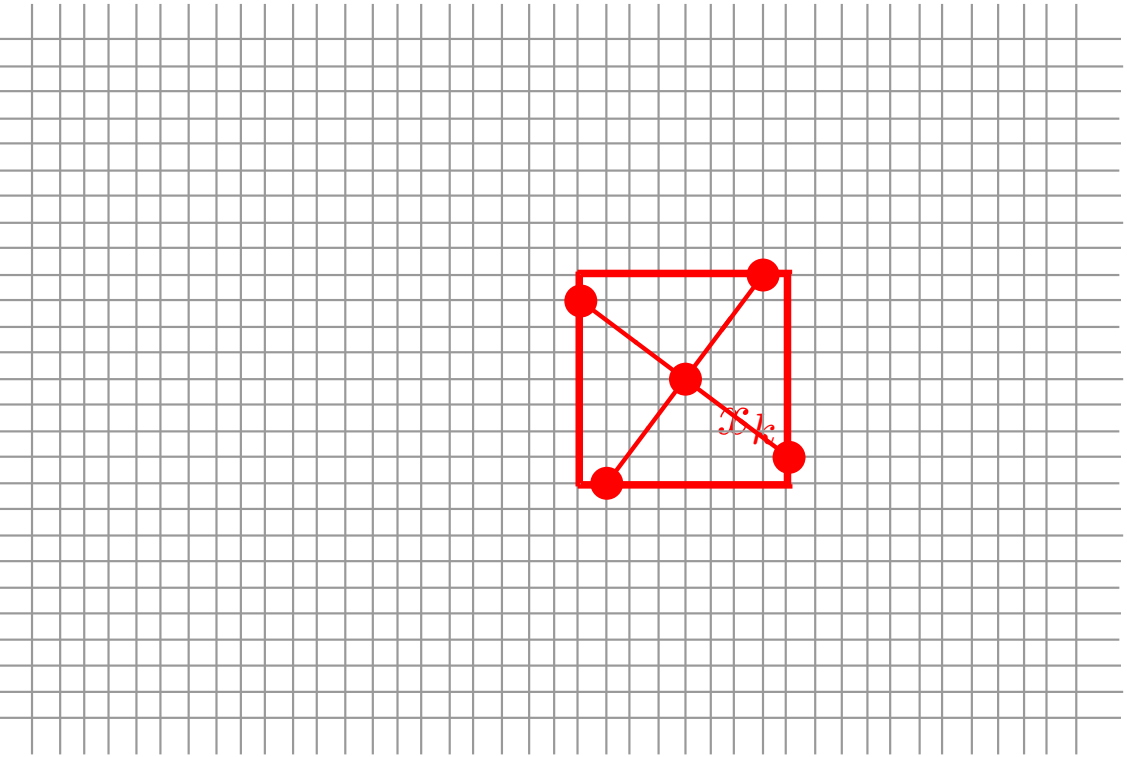
OrthoMads – 2008



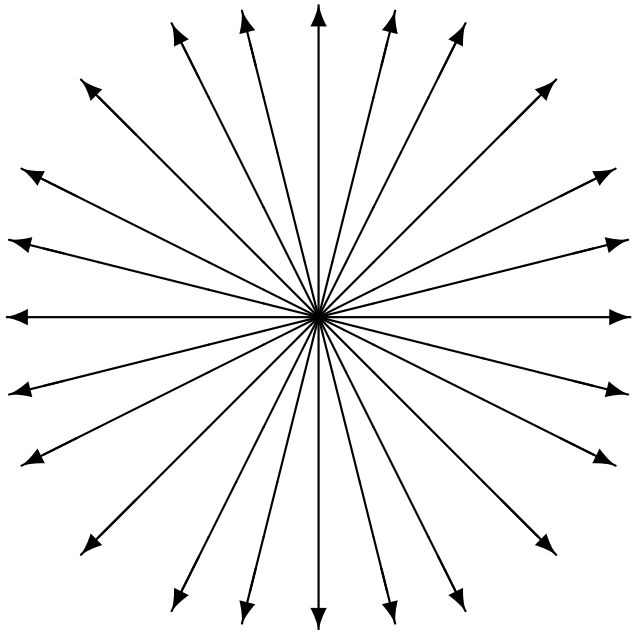
OrthoMads – 2008



OrthoMads – 2008



Union of all normalized directions grows dense in the unit sphere

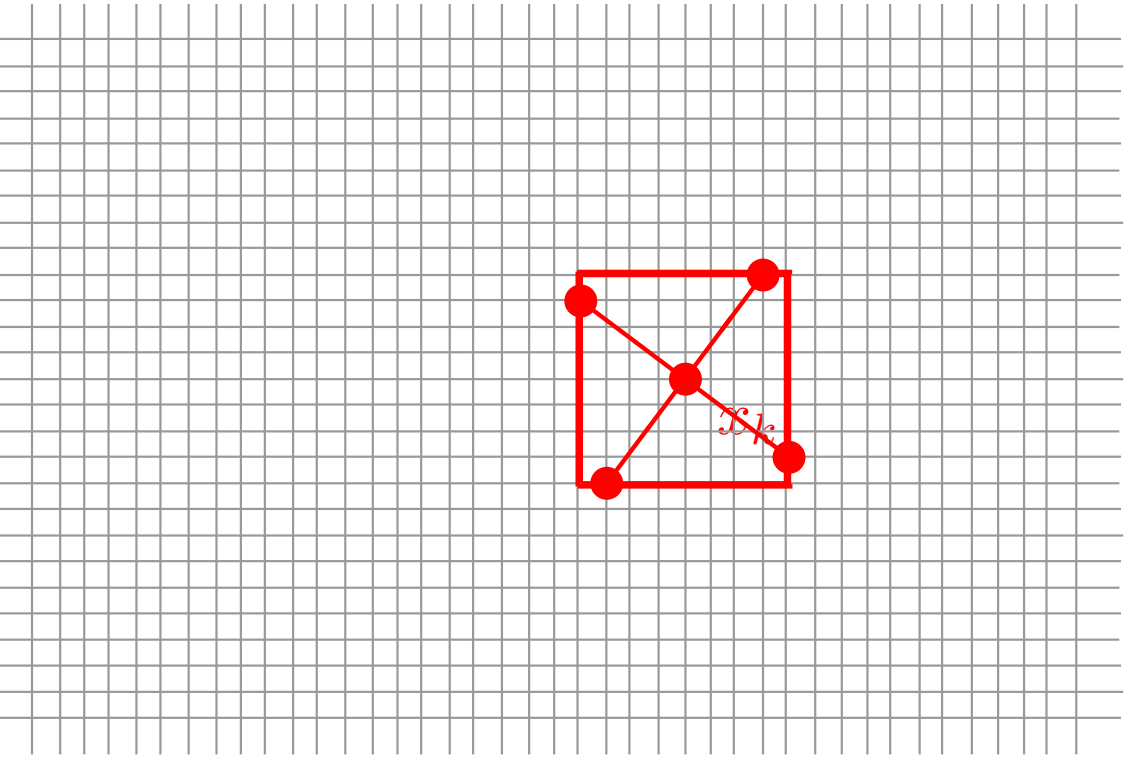


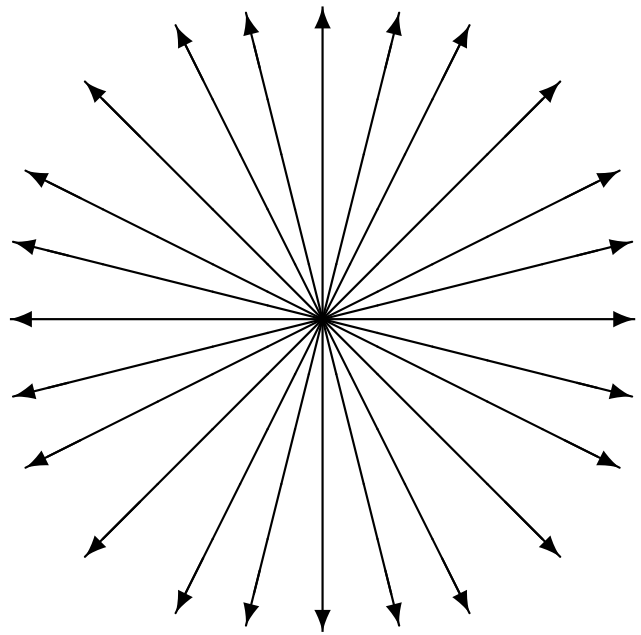
infinite number of directions

From Coordinate Search to OrthoMads

OrthoMads – 2008

Union of all normalized directions grows dense in the unit sphere



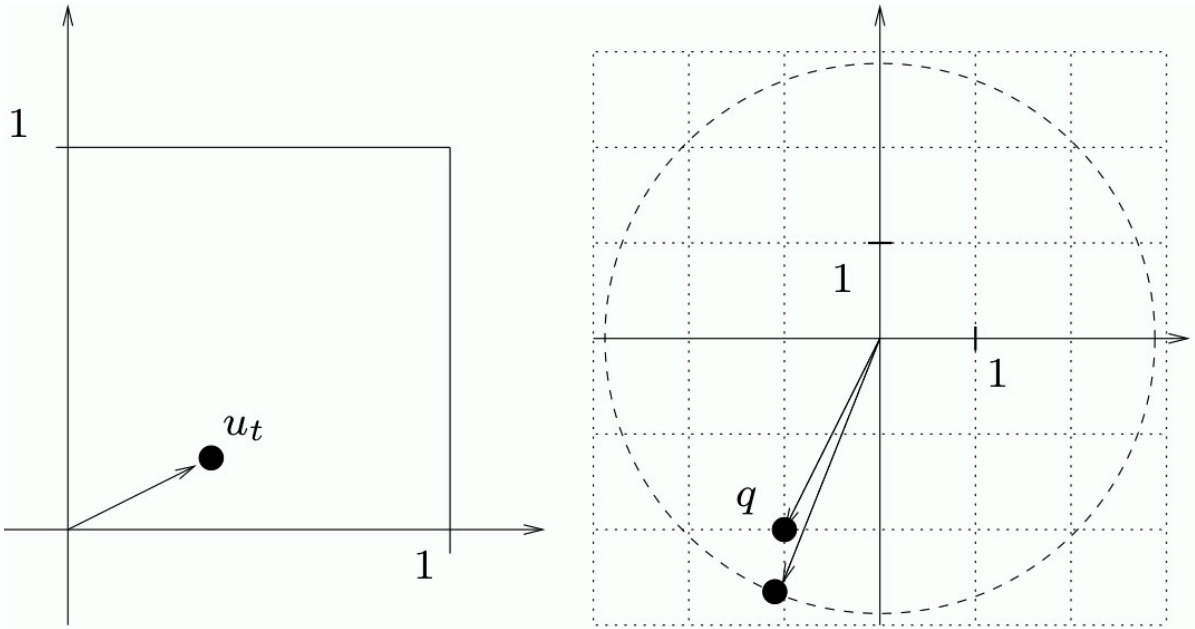


infinite number of directions

- OrthoMads is deterministic.
  Results are reproducible on any machine.
- At any given iteration, the directions are orthogonal.

OrthoMads: Halton Pseudo-random sequence

- OrthoMads uses the pseudo-random Halton sequence (1960) to generate a sequence $\{u_t\}_{t=1}^{\infty}$ of vectors in $\mathbb{R}^n$ that is dense in the hypercube $[0,1]^n$.
- The Mads convergence analysis requires that all trial points belong to a mesh that gets finer and finer as the algorithm unfolds. The direction u_t is translated and scaled to $\frac{2u_t - e}{\|2u_t - e\|}$ so that it belongs to the unit sphere (e is the vector of ones).
- The direction is rounded to the nearest mesh direction q.



OrthoMads: Householder orthogonal matrix

- The Householder transformation is then applied to the integer direction q:

$$H = \|q\|^2 I_n - 2qq^T,$$

where I_n is the identity matrix.

- By construction, H is an integer orthogonal basis of $\mathbb{R}^n$.
- The poll directions for OrthoMads are defined to be the columns of H and $-H$.
- A lower bound on the cosine of the maximum angle between any arbitrary nonzero vector $v \in \mathbb{R}^n$ and the set of directions in D is defined as

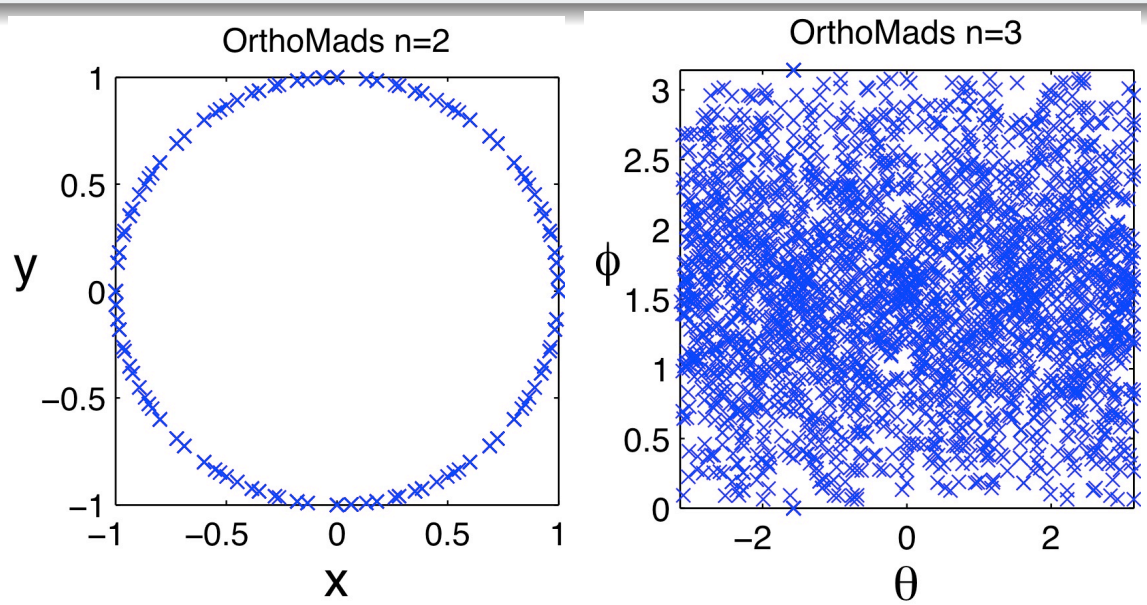
$$\kappa(D) = \min_{0 \neq v \in \mathbb{R}^n} \max_{d \in D} \frac{v^T d}{\|v\| \|d\|}.$$

With OrthoMads the measure $\kappa(D) = \frac{1}{\sqrt{n}}$ is maximized over all positive bases.

Convergence analysis - OrthoMads with extreme barrier

Theorem

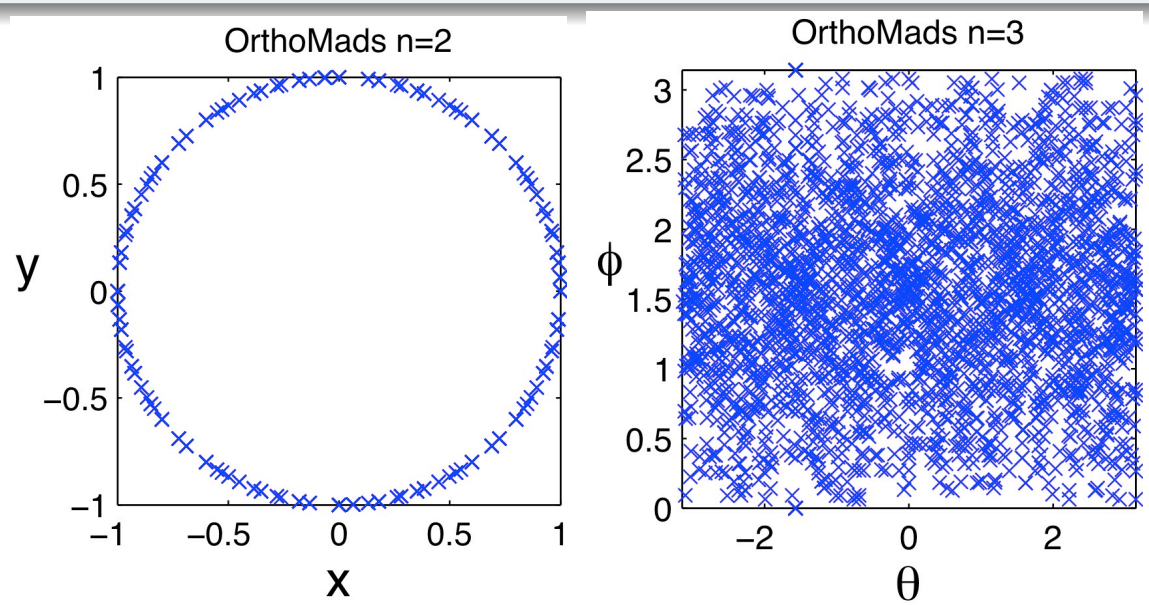
As $k \to \infty$, the set of OrthoMads *normalized poll directions is dense in the unit sphere.*



Convergence analysis - OrthoMads with extreme barrier

Theorem

As $k \to \infty$, the set of OrthoMads *normalized poll directions is dense in the unit sphere.*



Theorem

Let $\hat{x}$ be the the limit of a subsequence of mesh local optimizers on meshes that get infinitely fine. If f is Lipschitz near $\hat{x}$, then $f^\circ(\hat{x}, v) \geq 0$ for all $v \in T_\Omega^H(\hat{x})$.

Assuming more smoothness, Abramson studies second order convergence.

Open, closed and hidden constraints

Consider the toy problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & x_1^2 - \sqrt{x_2} \\ \text{s.t.} \quad & -x_1^2 + x_2^2 \leq 1 \\ & x_2 \geq 0 \end{aligned}$$

Open, closed and hidden constraints

Consider the toy problem:
$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & x_1^2 - \sqrt{x_2} \\ \text{s.t.} \quad & -x_1^2 + x_2^2 \leq 1 \\ & x_2 \geq 0 \end{aligned}$$

- Closed constraints *must* be satisfied at every trial vector of decision variables in order for the functions to evaluate.
  Here $x_2 \geq 0$ is a closed constraint, because if it is violated, the objective function will fail.

Open, closed and hidden constraints

Consider the toy problem:
$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & x_1^2 - \sqrt{x_2} \\ \text{s.t.} \quad & -x_1^2 + x_2^2 \leq 1 \\ & x_2 \geq 0 \end{aligned}$$

- Closed constraints *must* be satisfied at every trial vector of decision variables in order for the functions to evaluate.
  Here $x_2 \geq 0$ is a closed constraint, because if it is violated, the objective function will fail.
- Open constraints must be satisfied at the solution, but an optimization algorithm may generate iterates that violate it. Here $-x_1^2 + x_2^2 \leq 1$ is an open constraint.

Open, closed and hidden constraints

Consider the toy problem:
$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & x_1^2 - \ln(x_2) \\ \text{s.t.} \quad & -x_1^2 + x_2^2 \leq 1 \\ & x_2 \geq 0 \end{aligned}$$

- Closed constraints *must* be satisfied at every trial vector of decision variables in order for the functions to evaluate.
  Here $x_2 \geq 0$ is a closed constraint, because if it is violated, the objective function will fail.
- Open constraints must be satisfied at the solution, but an optimization algorithm may generate iterates that violate it. Here $-x_1^2 + x_2^2 \leq 1$ is an open constraint.
- Lets change the objective. $x_2 \neq 0$ is now an hidden constraint.
  f is set to ∞ when $x \in \Omega$ but x fails to satisfy an hidden contraint.

Open, closed and hidden constraints

Consider the toy problem:
$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & x_1^2 - \ln(x_2) \\ \text{s.t.} \quad & -x_1^2 + x_2^2 \leq 1 \\ & x_2 \geq 0 \end{aligned}$$

- Closed constraints *must* be satisfied at every trial vector of decision variables in order for the functions to evaluate.
  Here $x_2 \geq 0$ is a closed constraint, because if it is violated, the objective function will fail.
- Open constraints must be satisfied at the solution, but an optimization algorithm may generate iterates that violate it. Here $-x_1^2 + x_2^2 \leq 1$ is an open constraint.
- Lets change the objective. $x_2 \neq 0$ is now an hidden constraint.
  f is set to ∞ when $x \in \Omega$ but x fails to satisfy an hidden contraint.
- This terminology differs from *soft and hard* constraints which mean that satisfaction might allow, or might not, for some tolerance on the right hand side of $c_j(x) \leq 0$.

Constrained optimization

Consider the constrained problem

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

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$
and $\Omega = \{x \in X \subset \mathbb{R}^n : C(x) \leq 0\}$ with $C : \mathbb{R}^n \to (\mathbb{R} \cup \{\infty\})^m$.

Constrained optimization

Consider the constrained problem

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

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$
and $\Omega = \{x \in X \subset \mathbb{R}^n : C(x) \leq 0\}$ with $C : \mathbb{R}^n \to (\mathbb{R} \cup \{\infty\})^m$.

Hypothesis

An initial $x_0 \in X$ with $f(x_0) < \infty$, and $C(x) < \infty$ is provided.

Filter approach to constraints (Based on Fletcher - Leyffer)

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

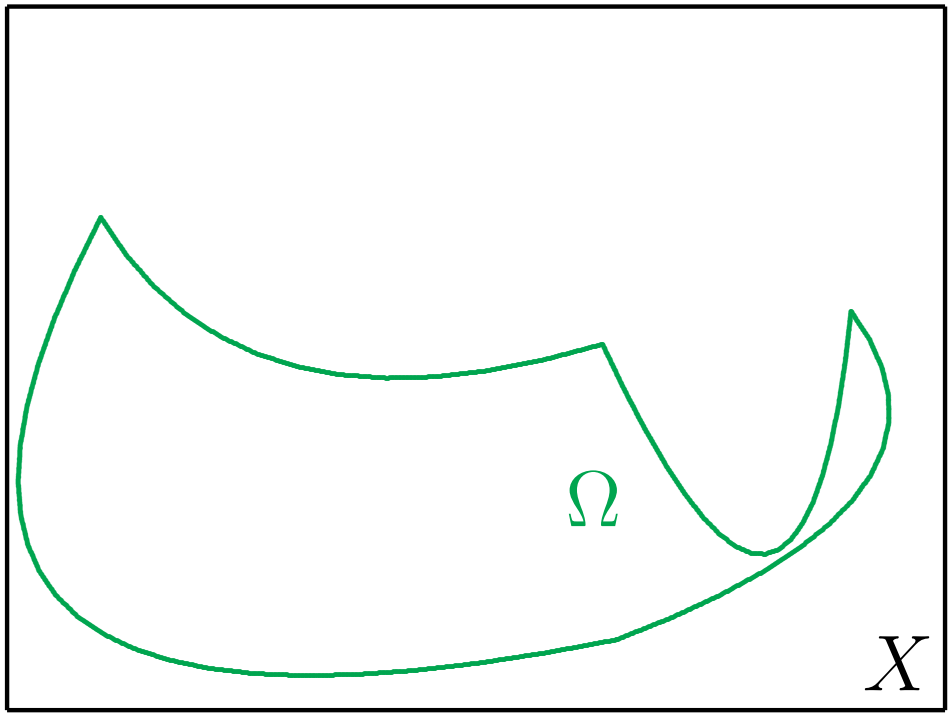
The extreme barrier handles the closed and hidden constraints X.
A filter handles $C(x) \leq 0$.

Define the nonnegative constraint violation function

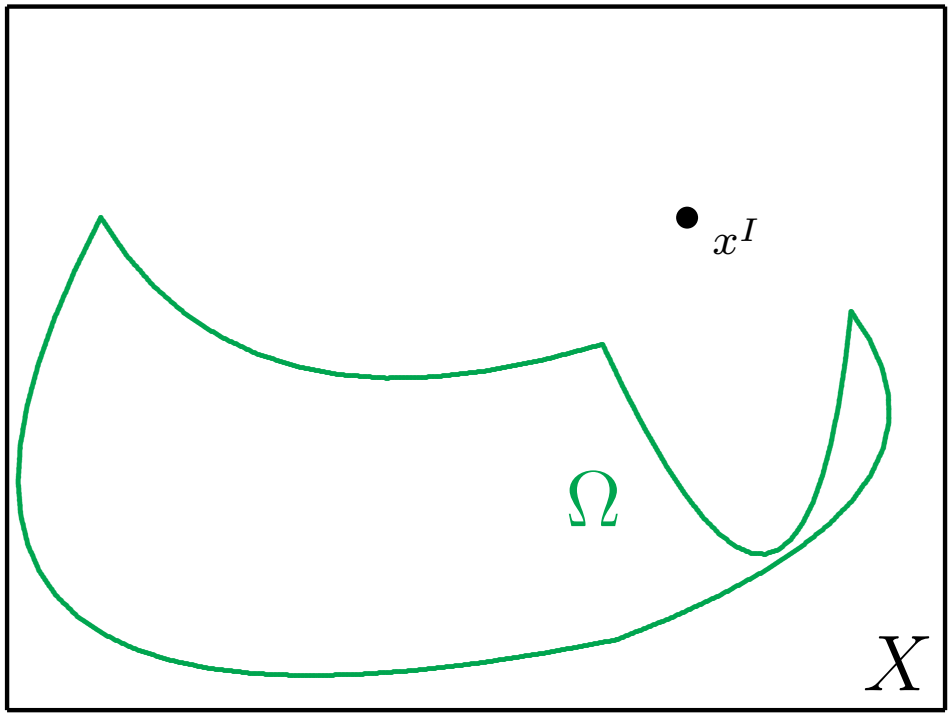
$$h(x) := \begin{cases} \sum_j \max(0, c_j(x))^2 & \text{if } x \in X \text{ and } f(x) < \infty, \quad \Leftarrow \text{ open constraints} \\ +\infty & \text{otherwise.} \quad \Leftarrow \text{ closed constraints} \end{cases}$$

$h(x) = 0$ if and only if $x \in \Omega$.
The constrained optimization problem is then viewed as a biobjective one: to minimize f and h, with a priority to h. This allows trial points that violate the open constraints.

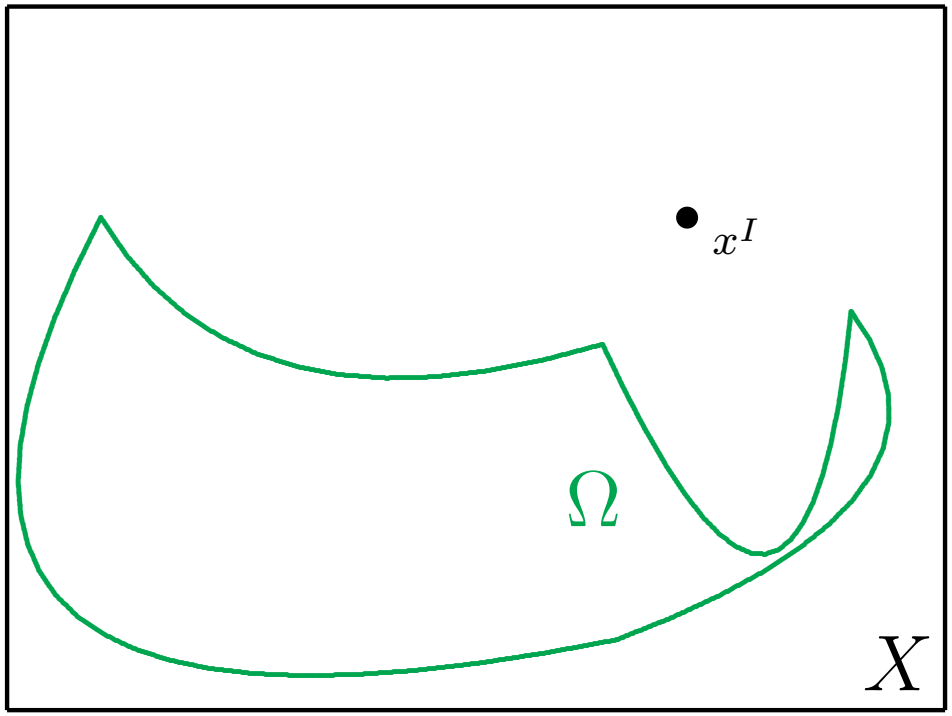
Filter approach to constraints

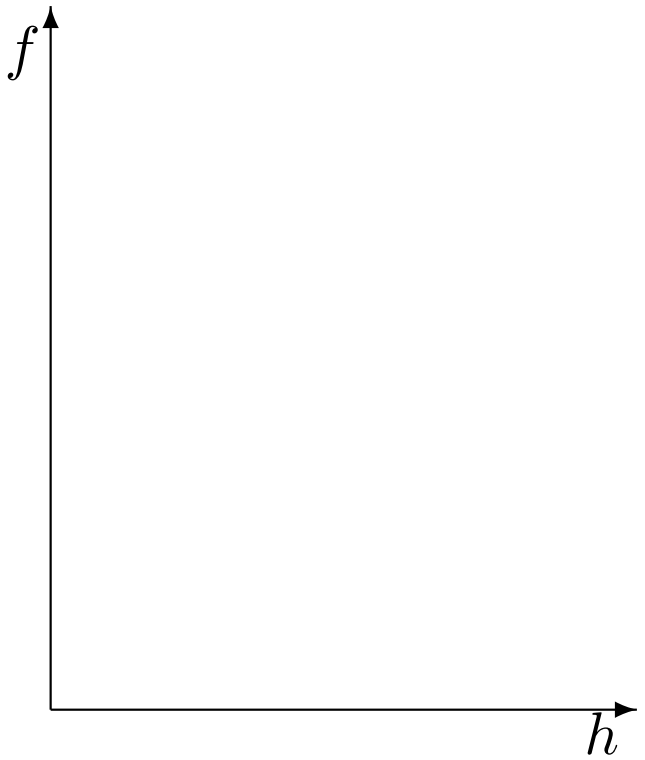


Filter approach to constraints

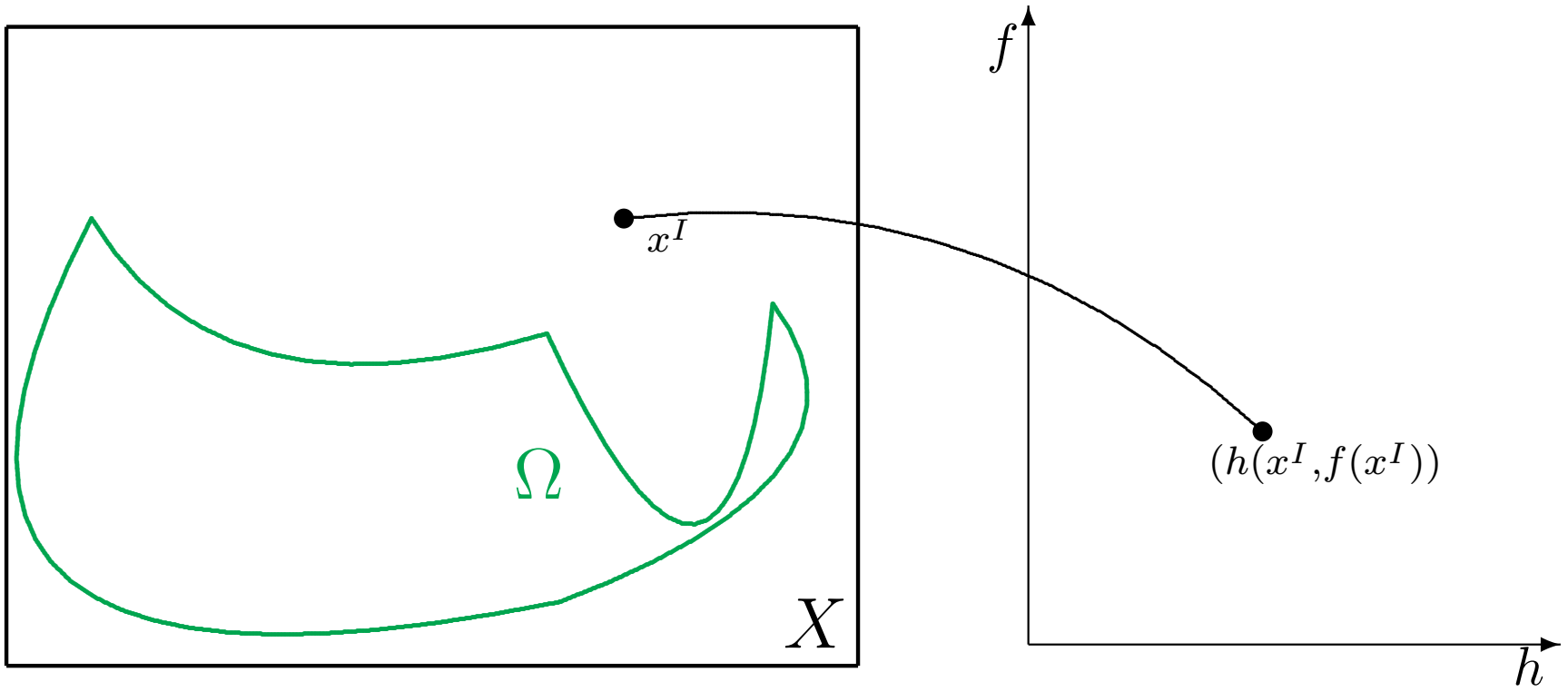


Filter approach to constraints

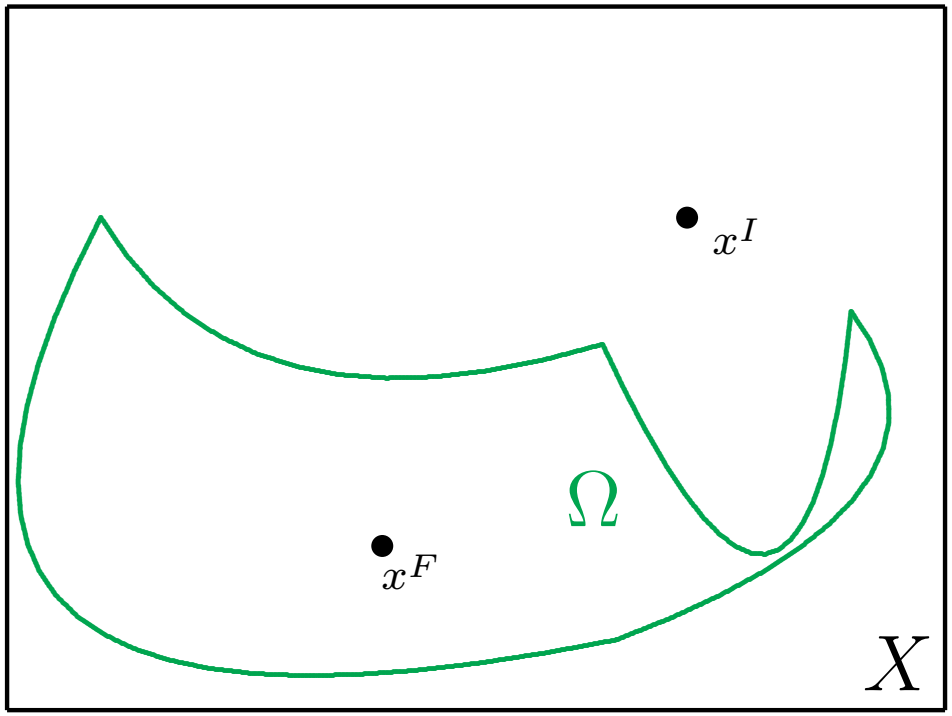


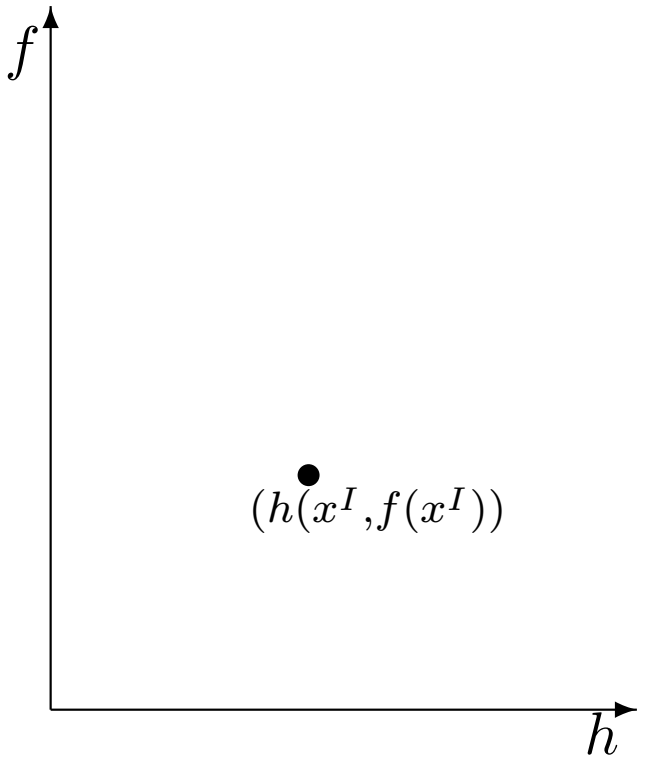


Filter approach to constraints

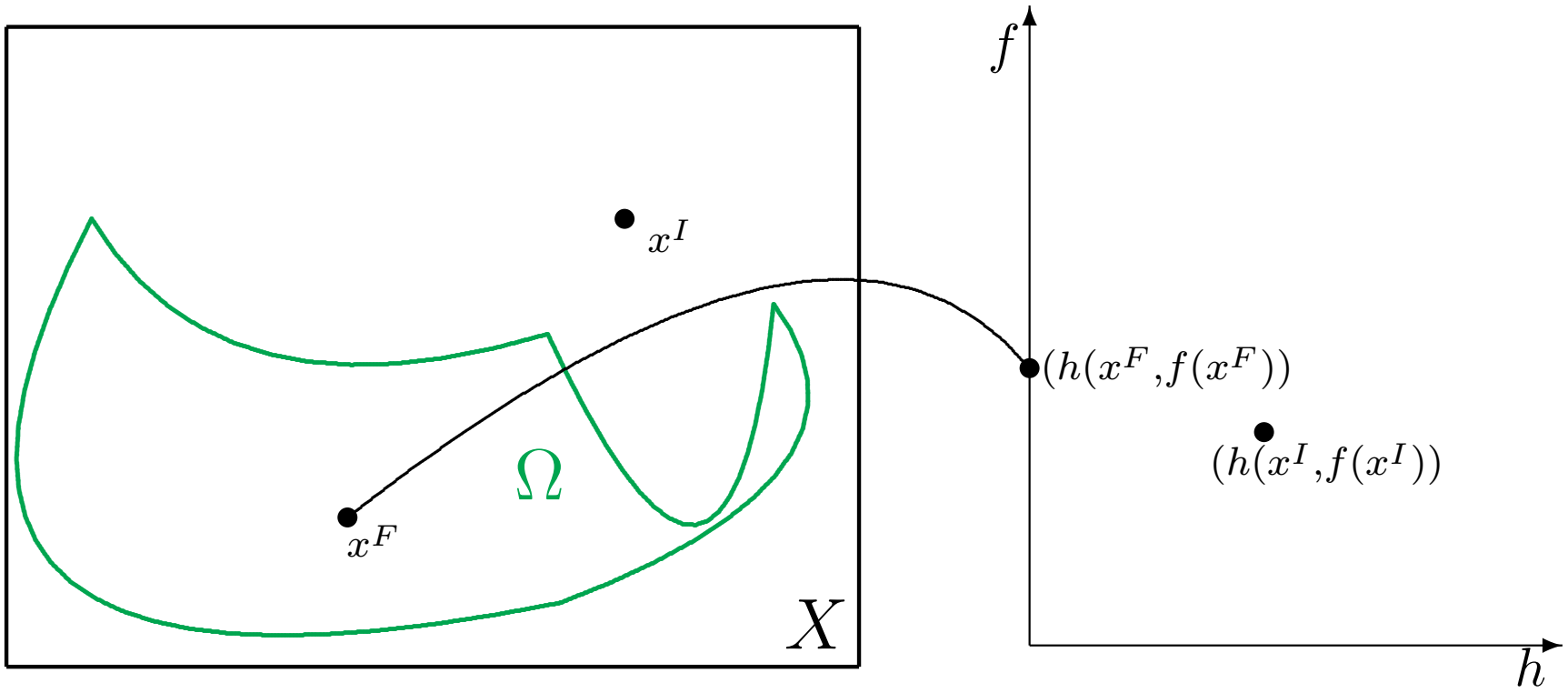


Filter approach to constraints

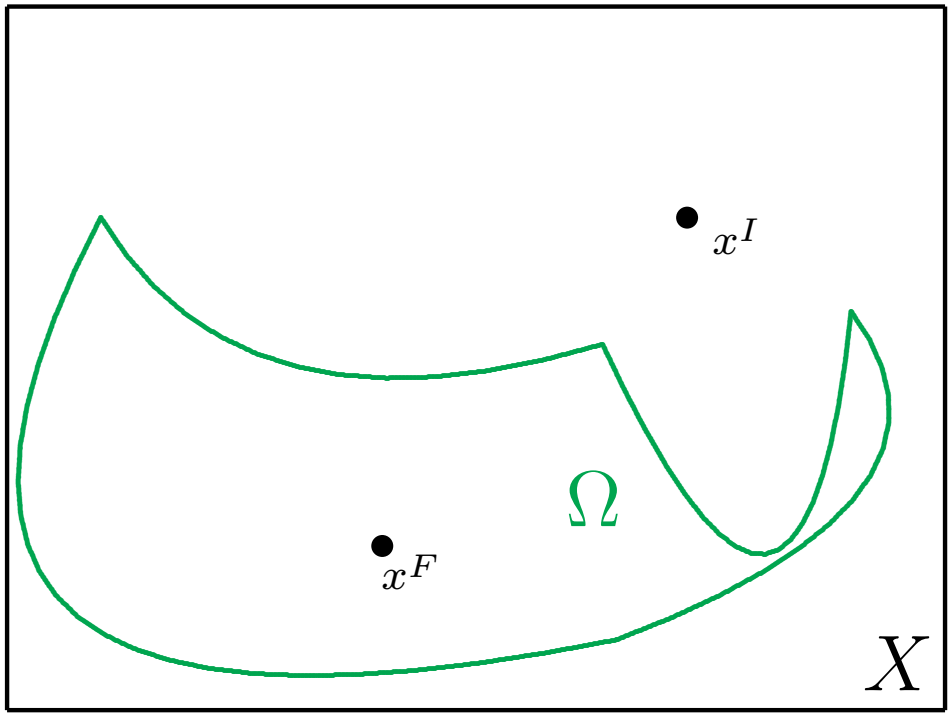


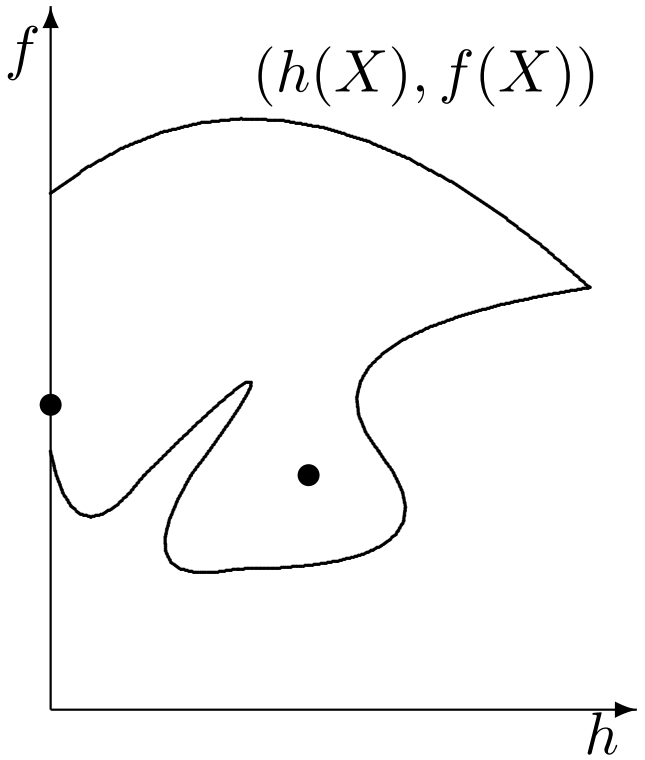


Filter approach to constraints

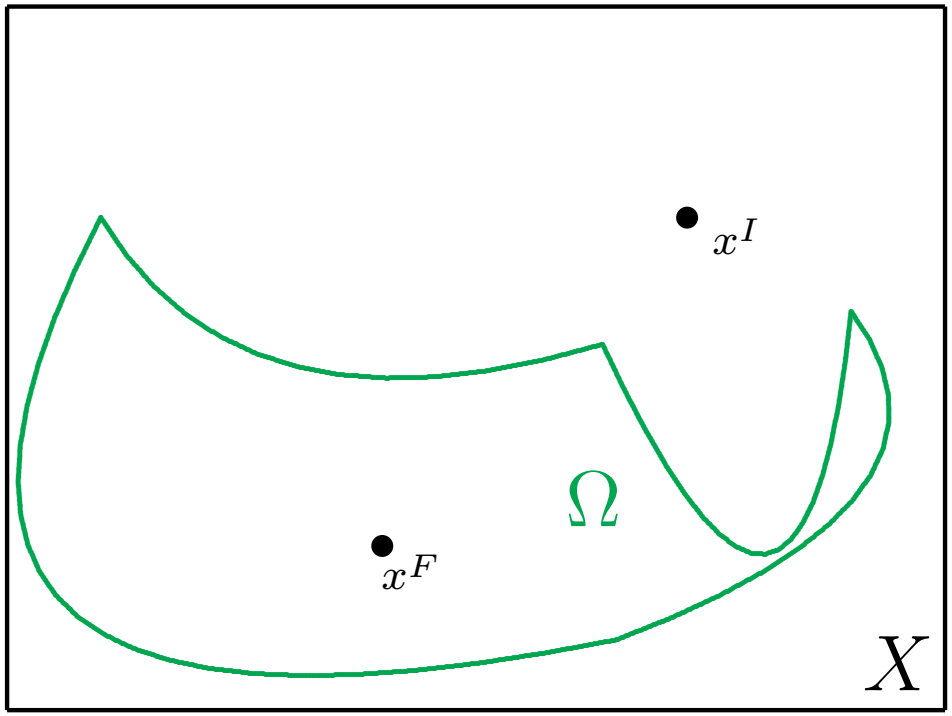


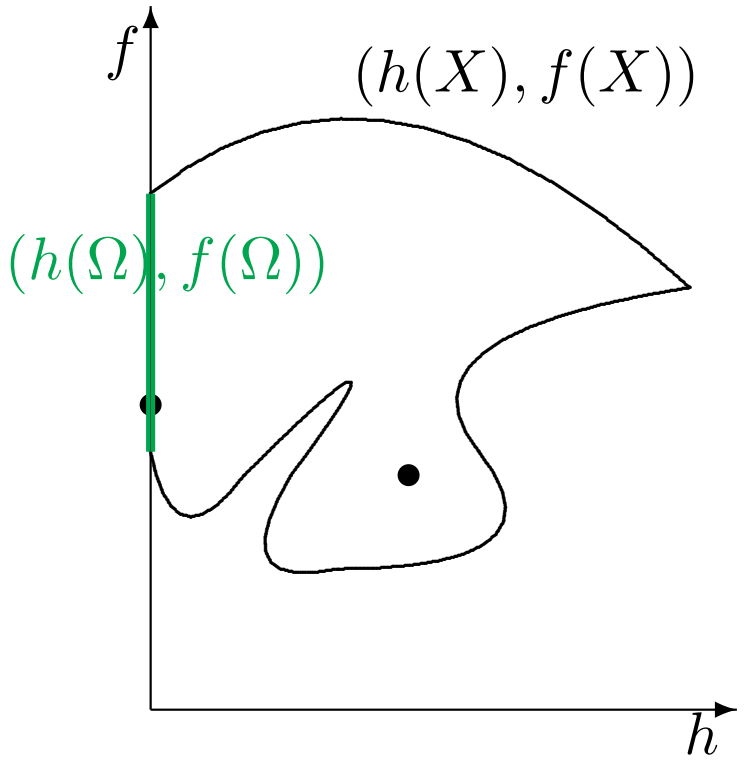
Filter approach to constraints



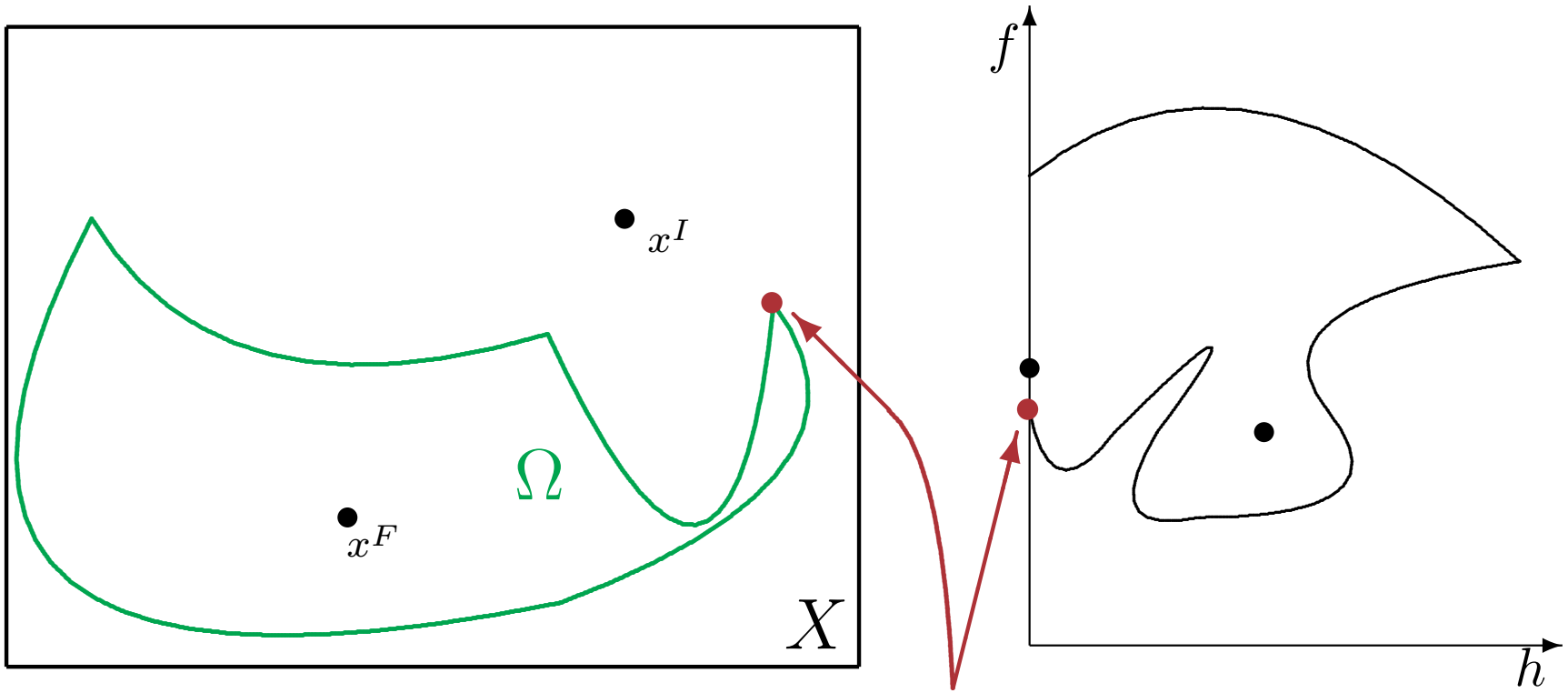


Filter approach to constraints



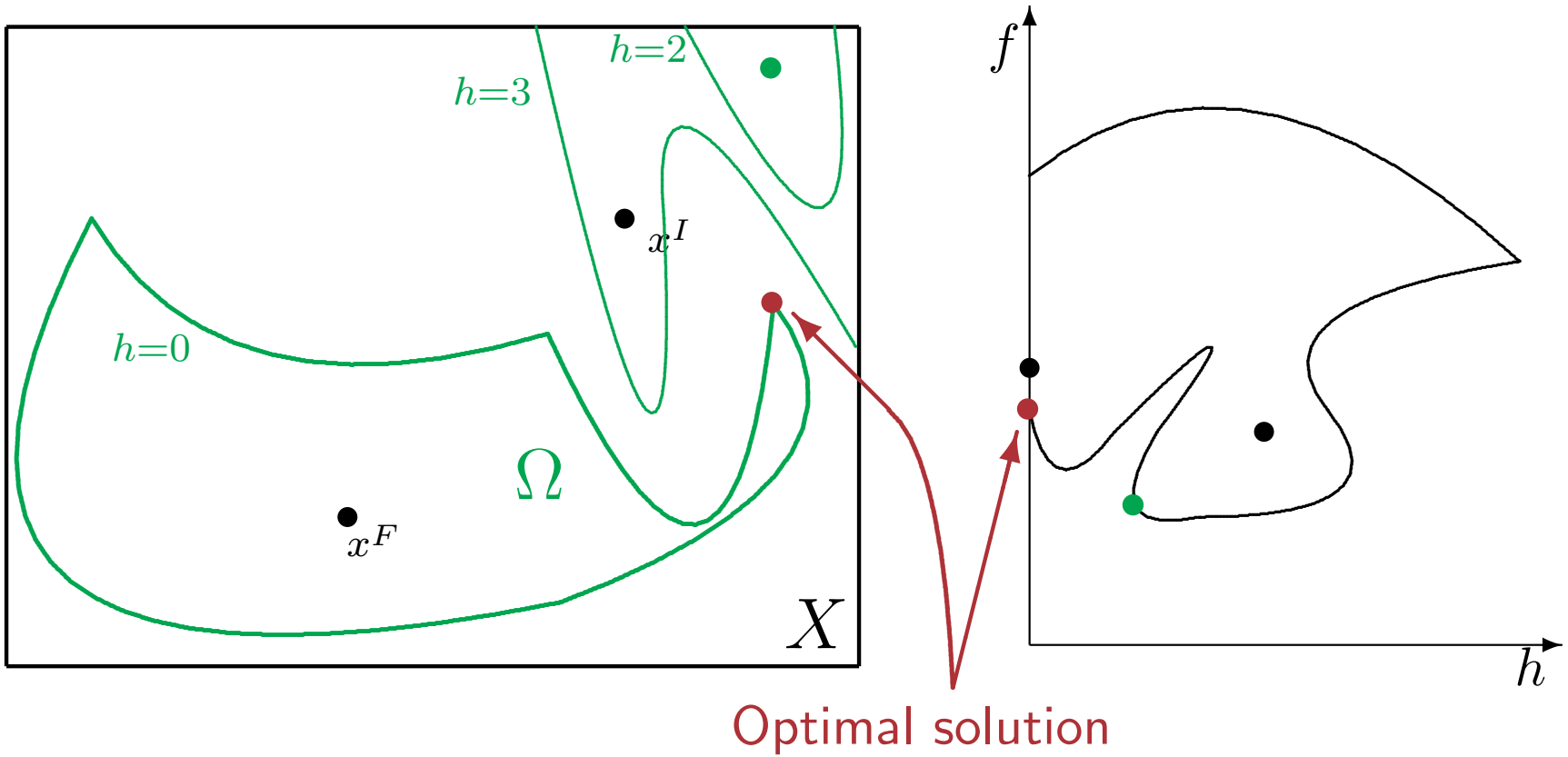


Filter approach to constraints

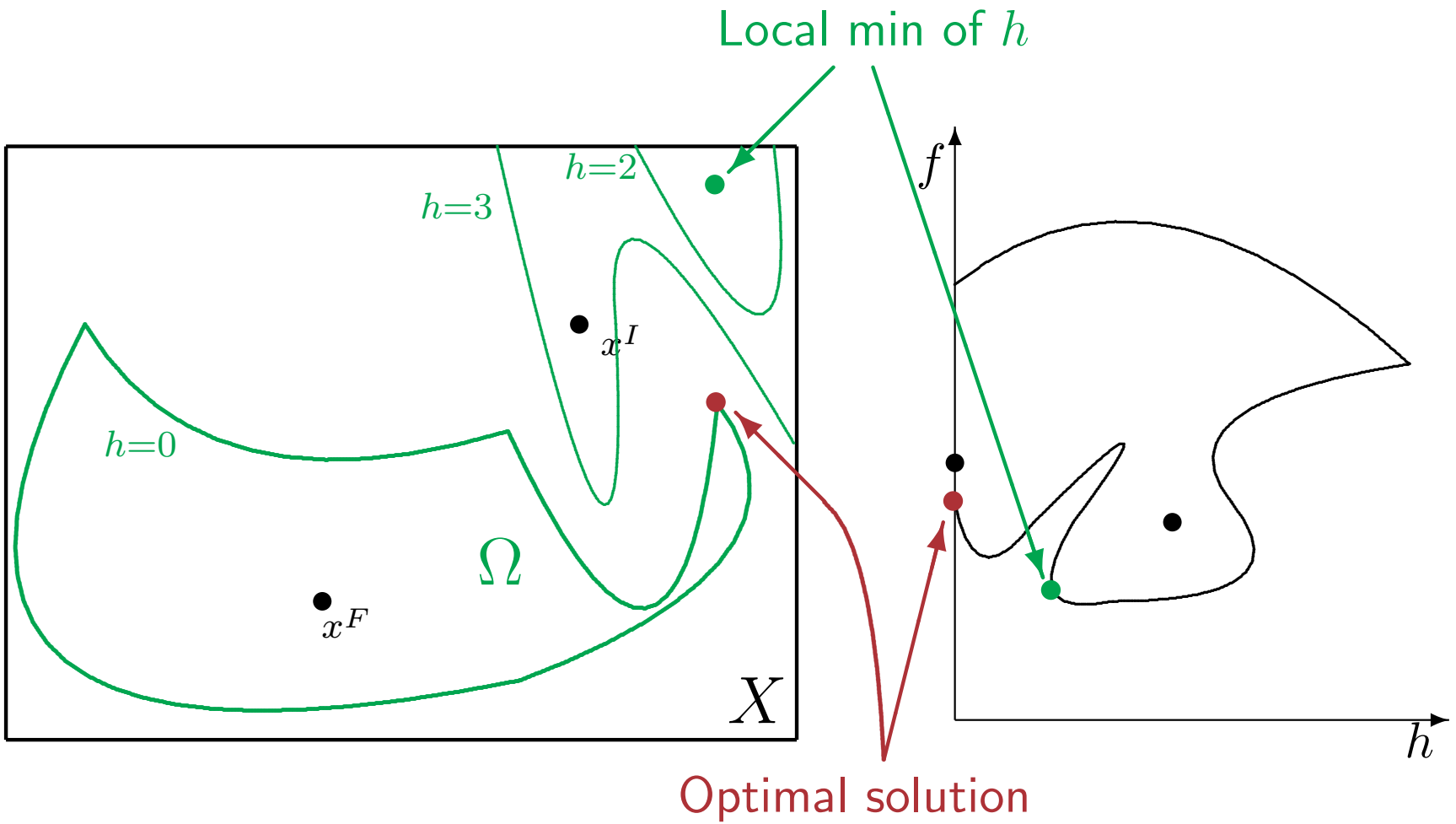


Optimal solution

Filter approach to constraints



Filter approach to constraints

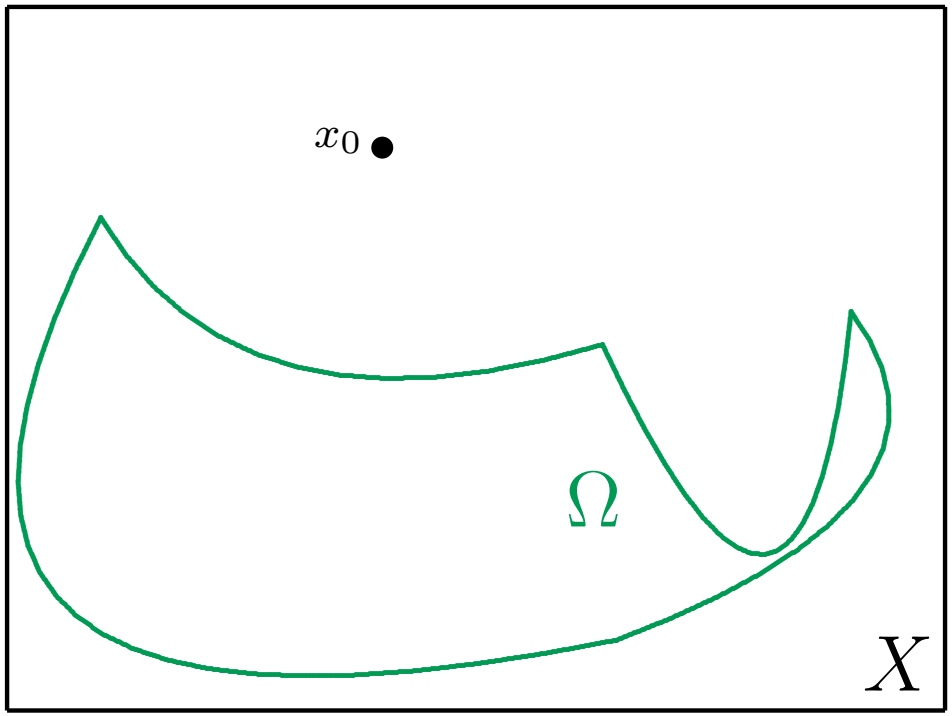


A MADS algorithm with a progressive barrier

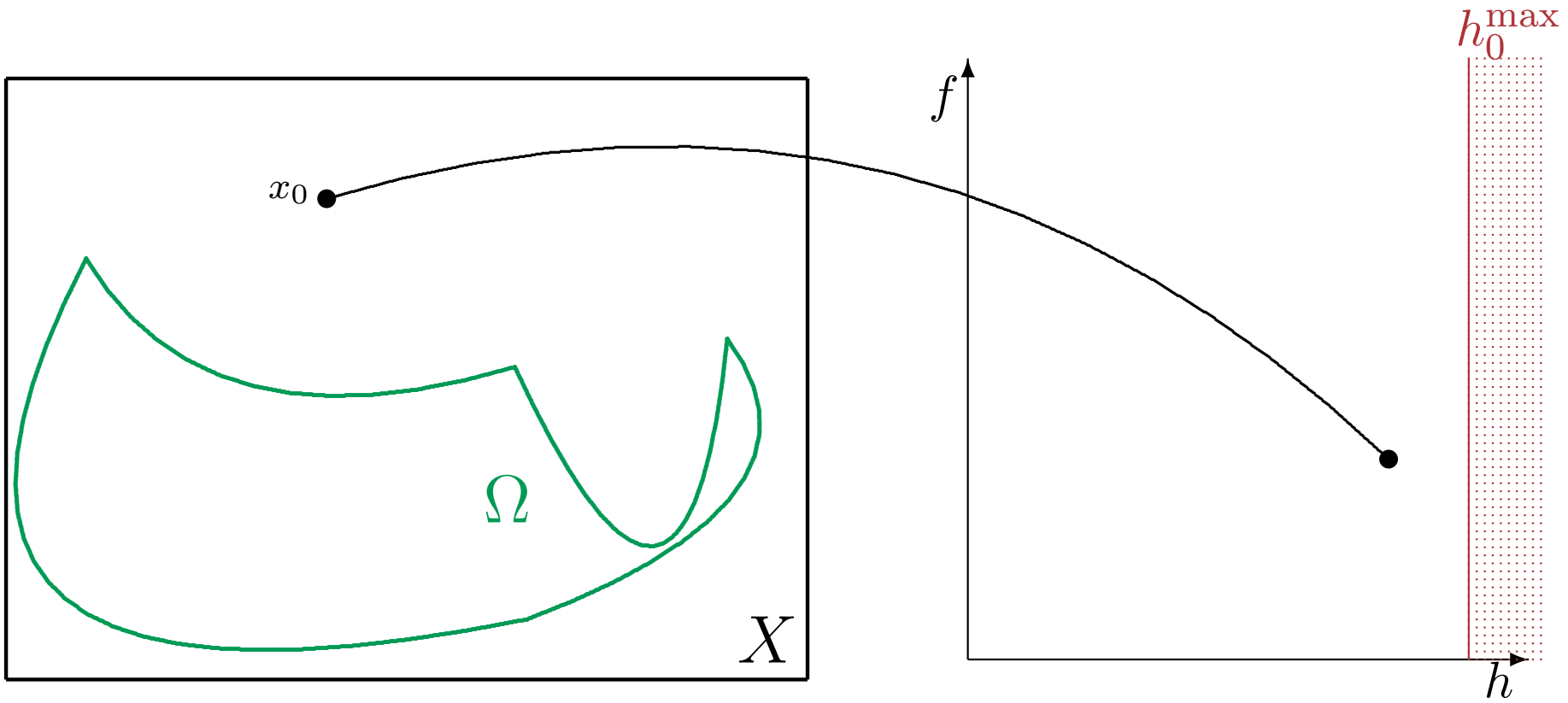
We present an algorithm that

- At iteration k, any trial point whose constraint violation value exceeds the value $h_k^{\max}$ is discarded from consideration.
- As k increases, the threshold $h_k^{\max}$ is reduced.
- The algorithm accepts some trial points that violate the open constraints.

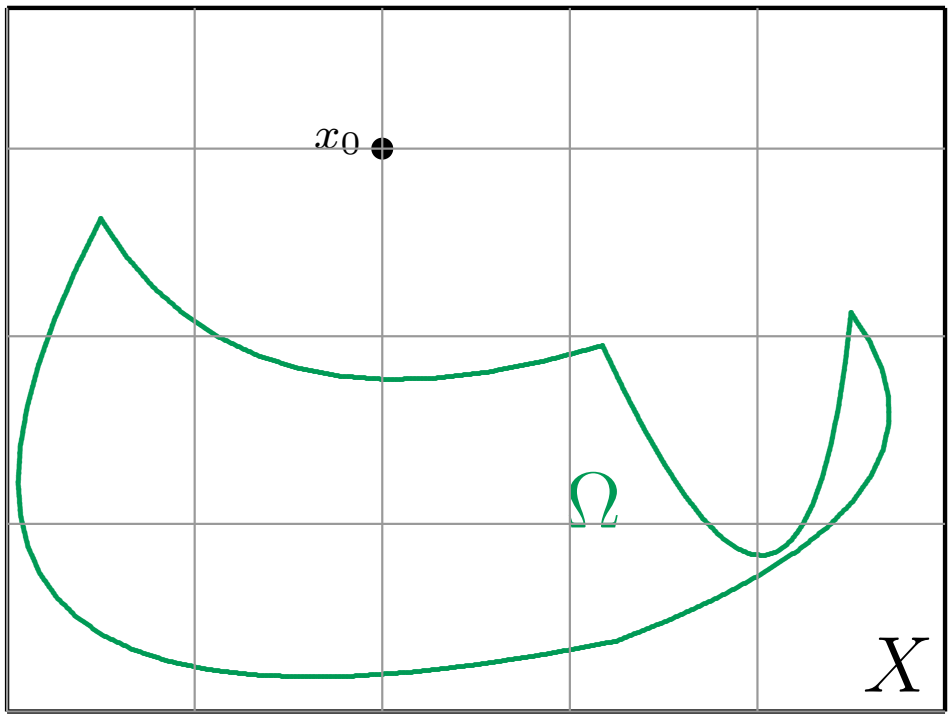
Progressive barrier algorithm : Iteration 0

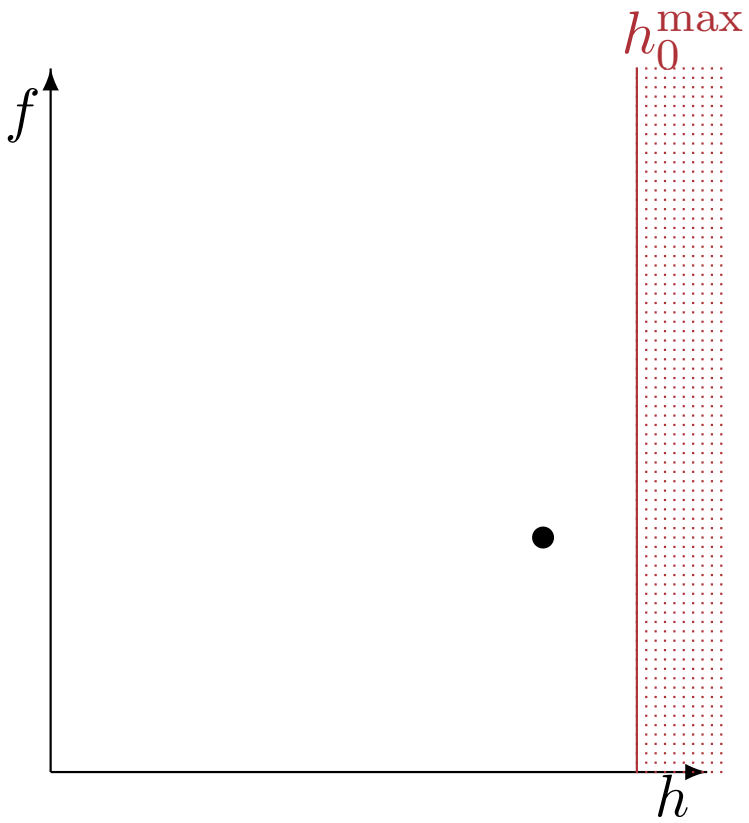


Progressive barrier algorithm : Iteration 0



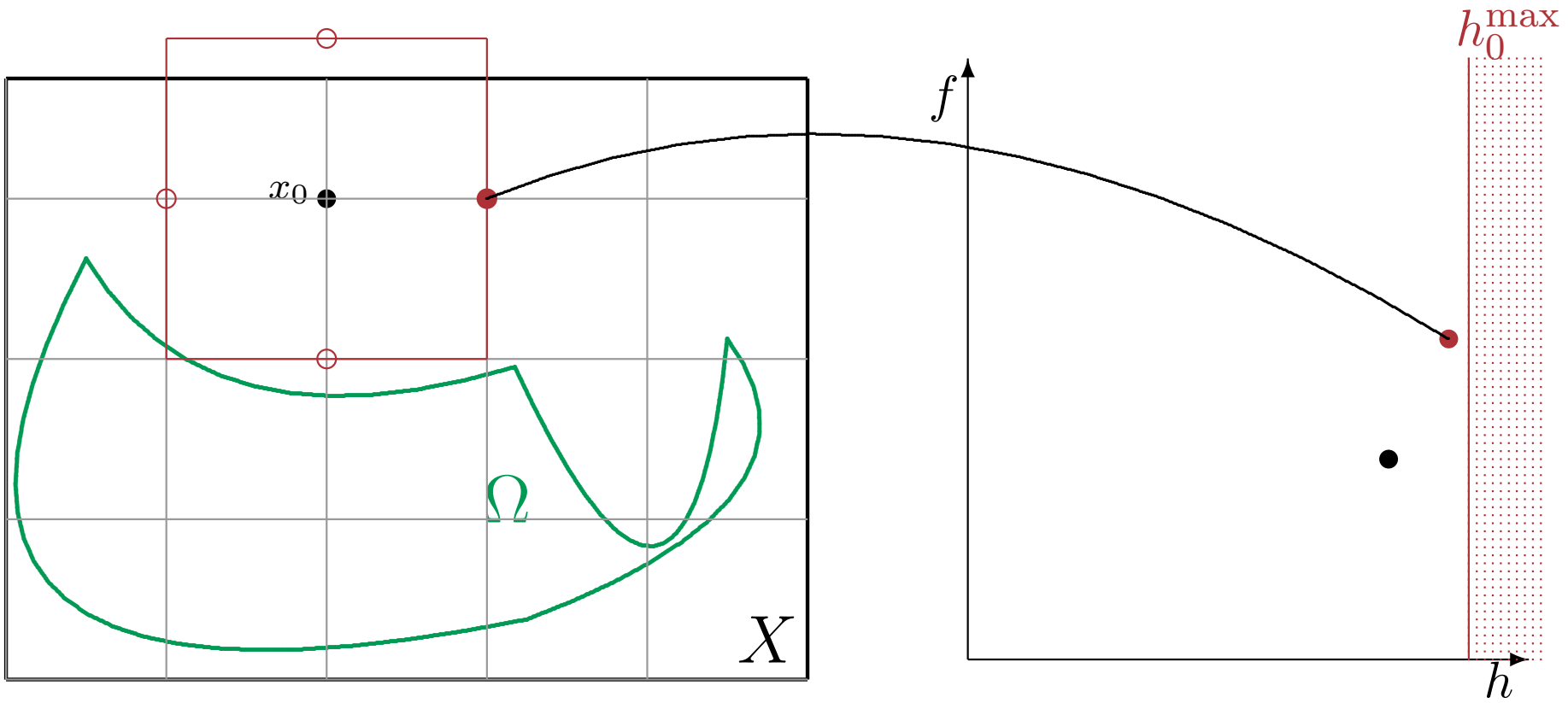
Progressive barrier algorithm : Iteration 0





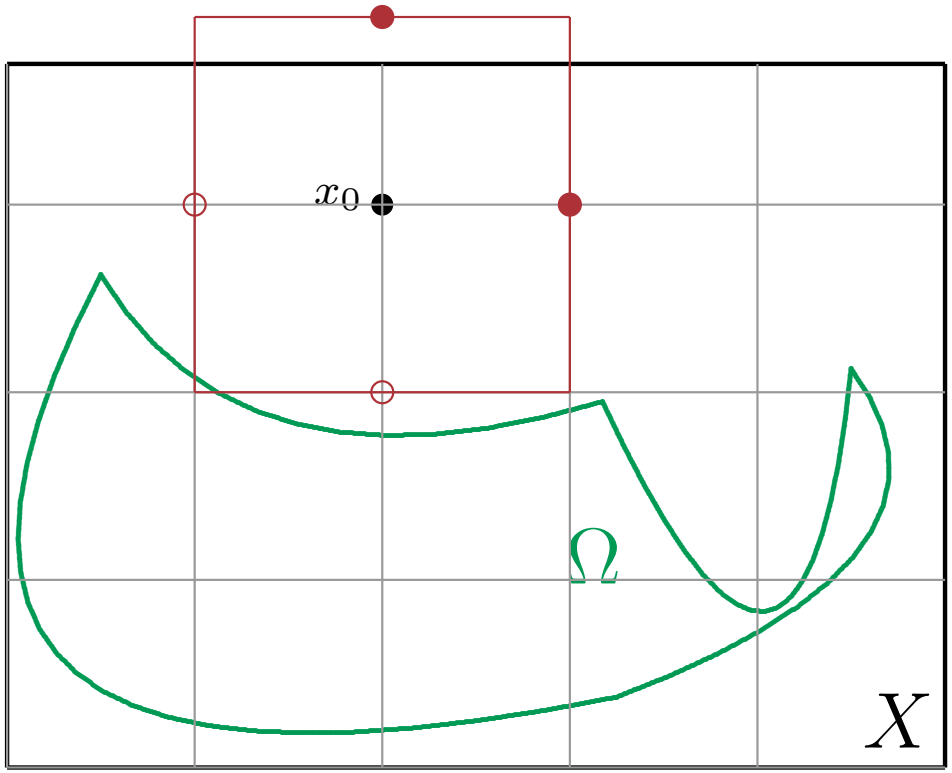
Progressive barrier algorithm : Iteration 0

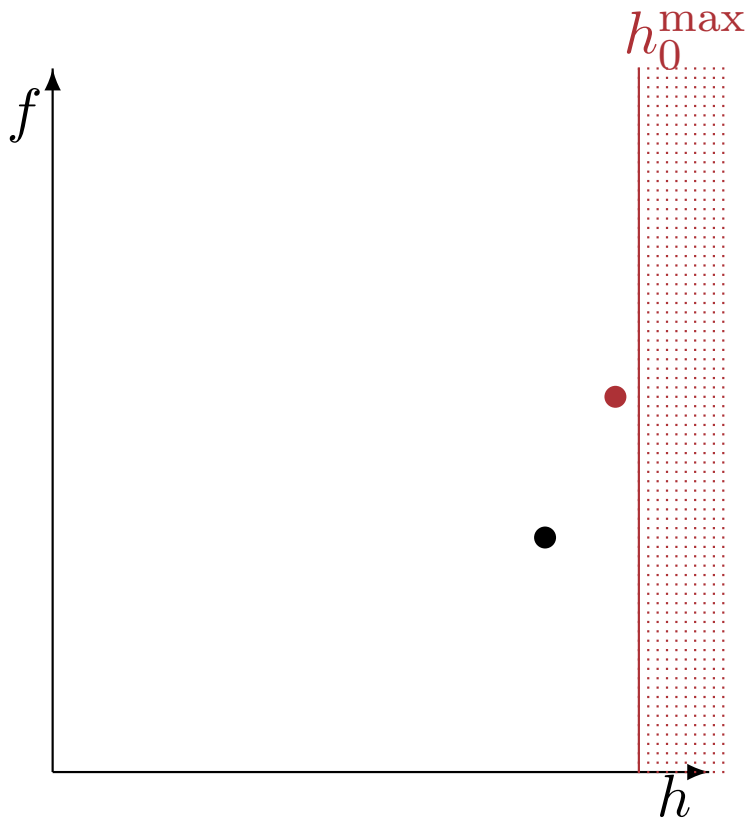
Worst h and worst f



Progressive barrier algorithm : Iteration 0

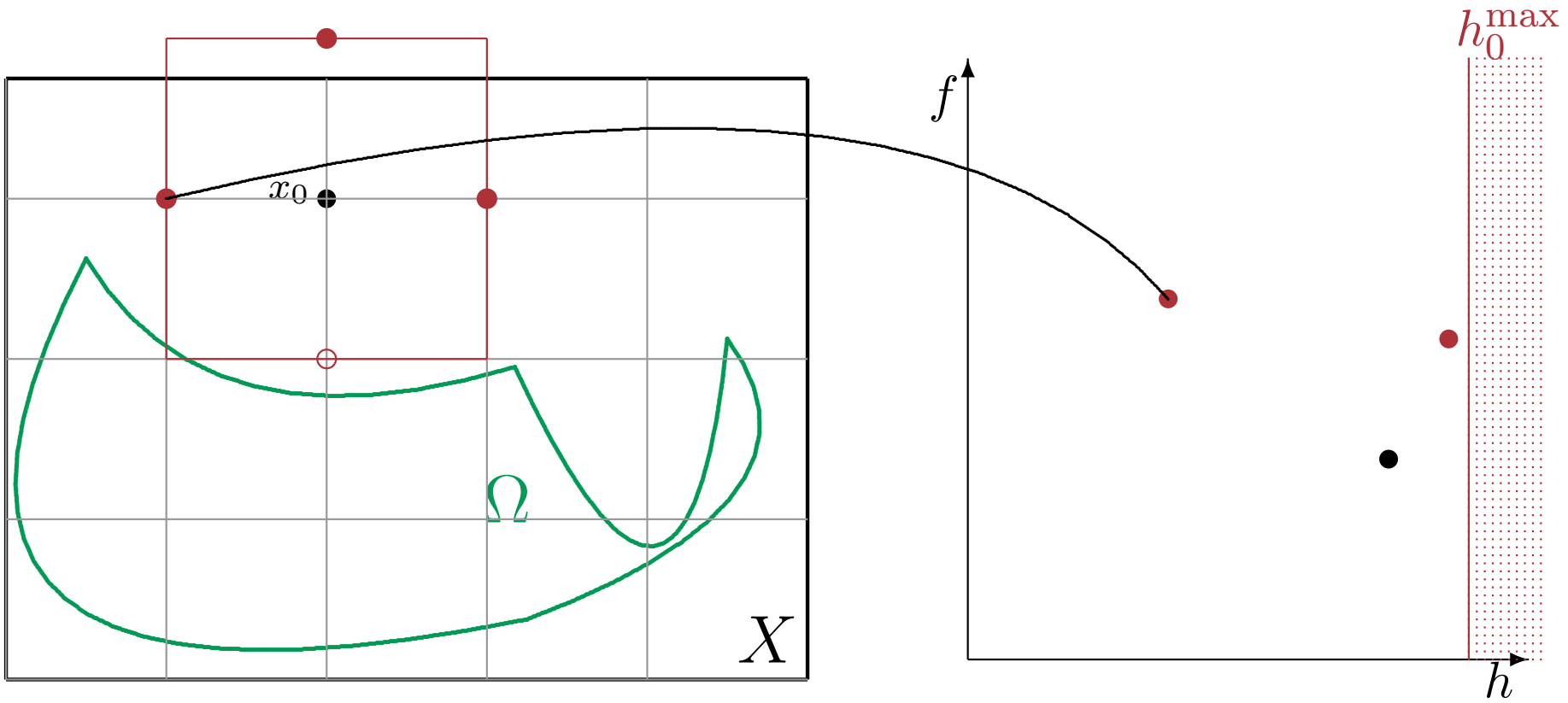
Outside closed constraints X : reject point (barrier approach)





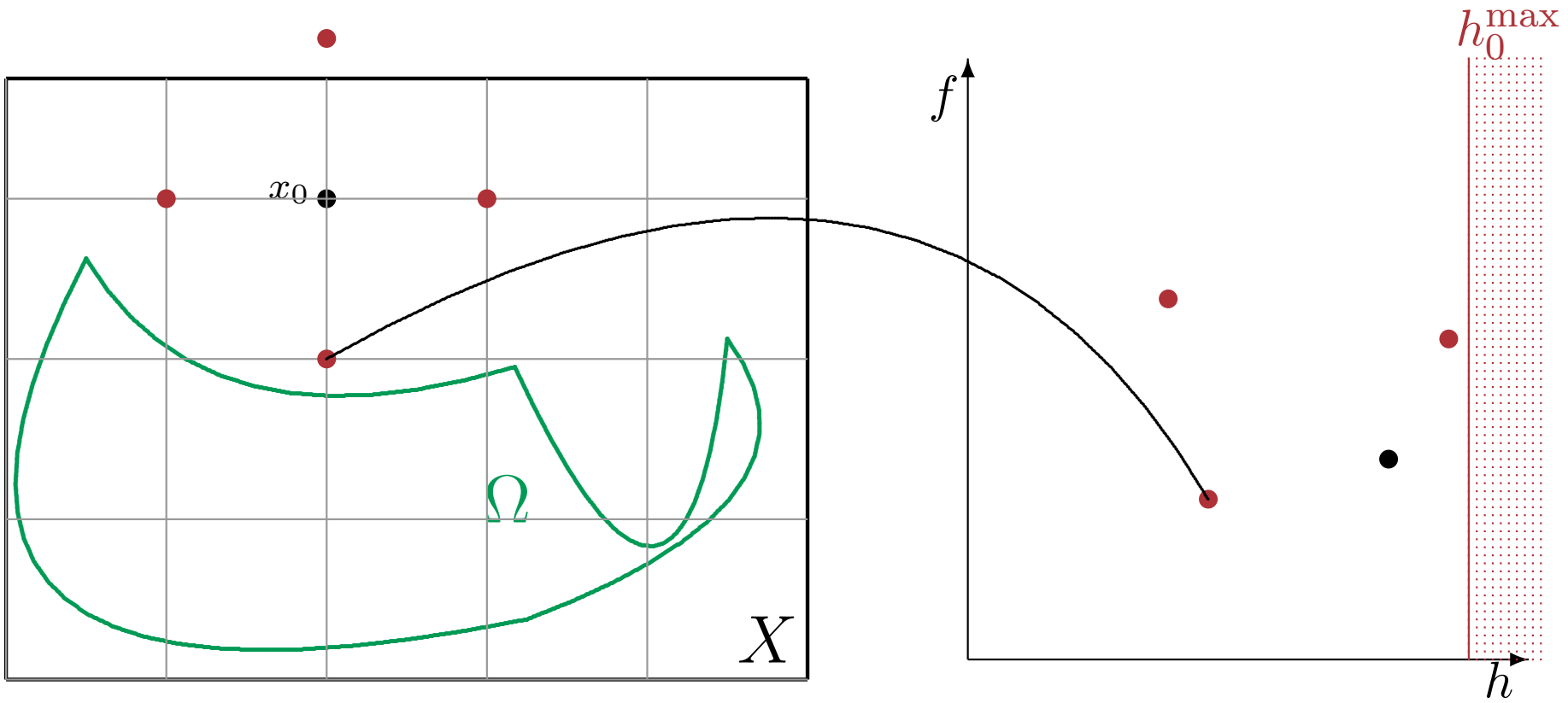
Progressive barrier algorithm : Iteration 0

Better h but worst f



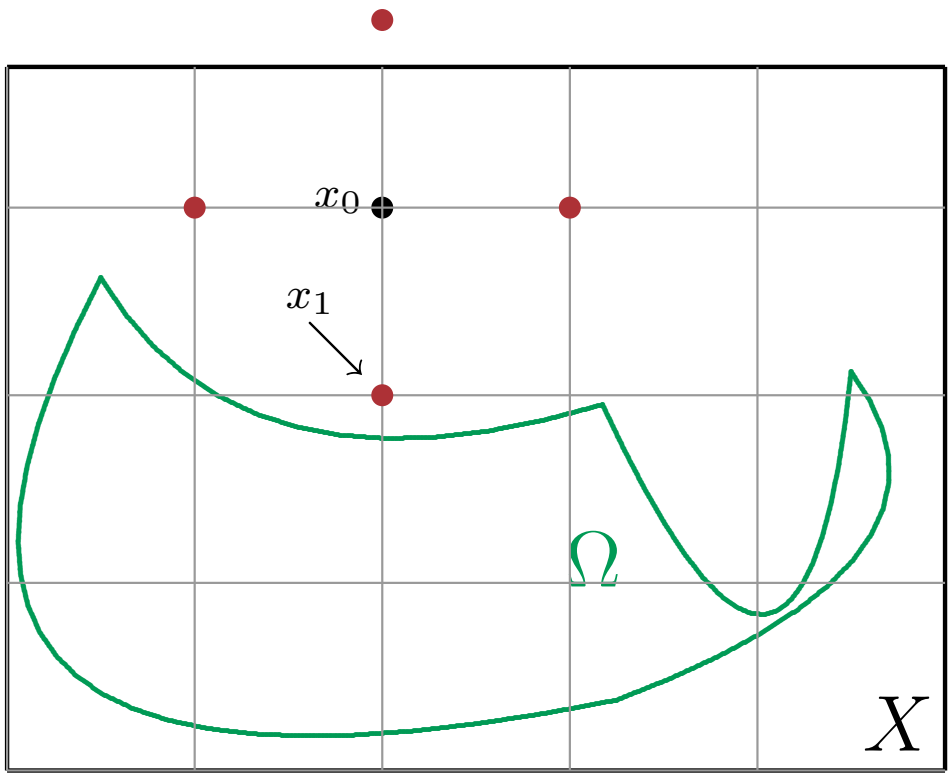
Progressive barrier algorithm : Iteration 0

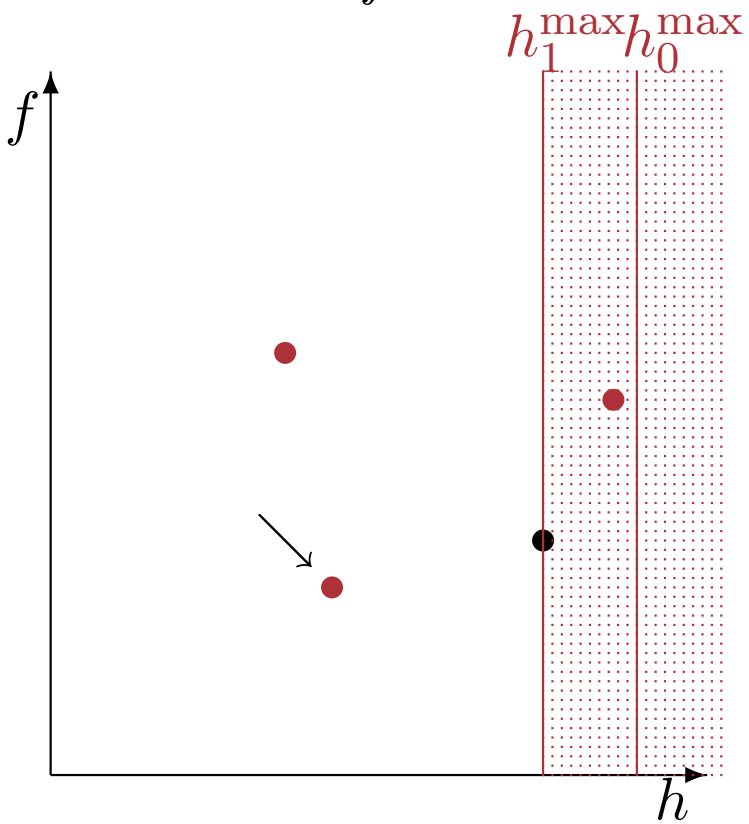
Better h and better f



Progressive barrier algorithm : Iteration 0

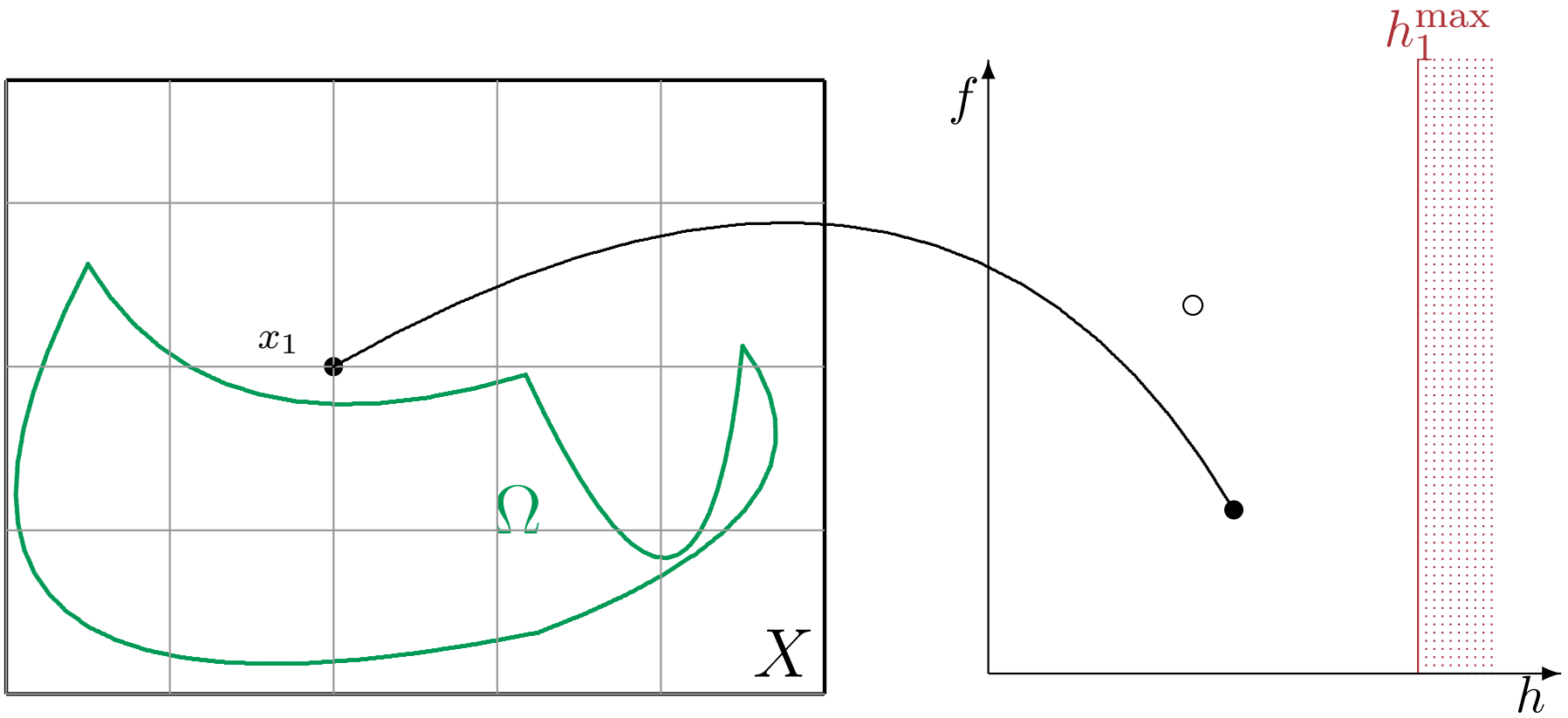
The new infeasible incumbent is the one to the left (in (h, f) plot) of the last one with the best f value



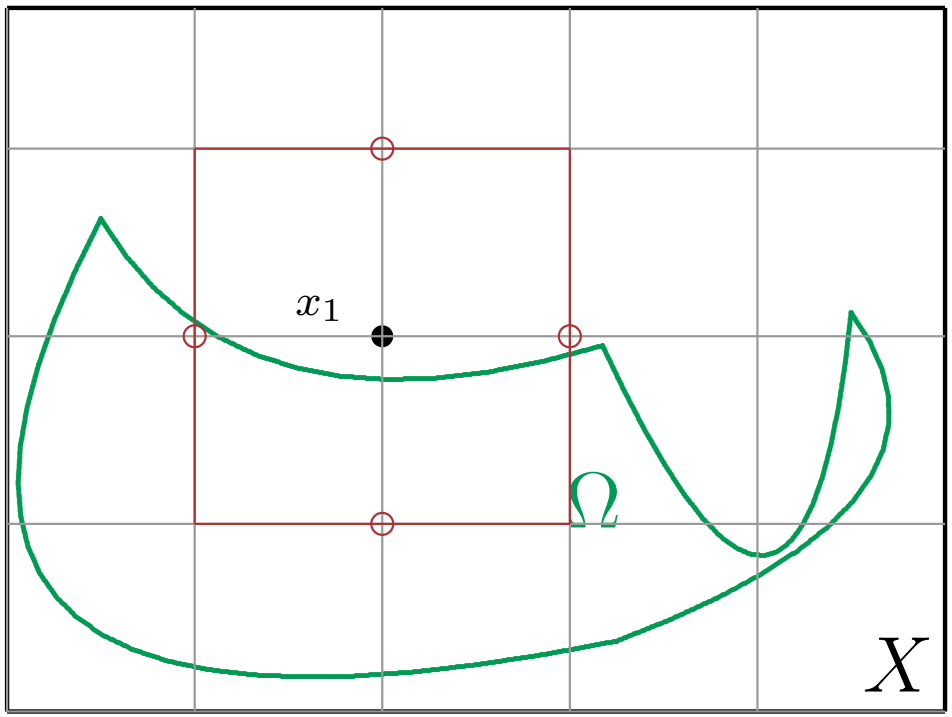


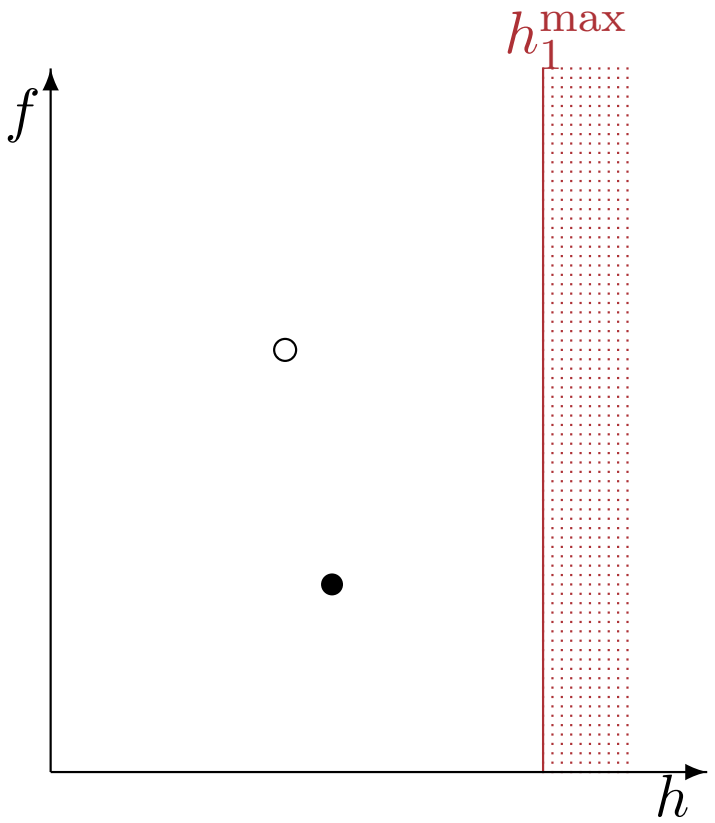
If time $>$ 11h35, then Skip 12 clicks

Progressive barrier algorithm : Iteration 1

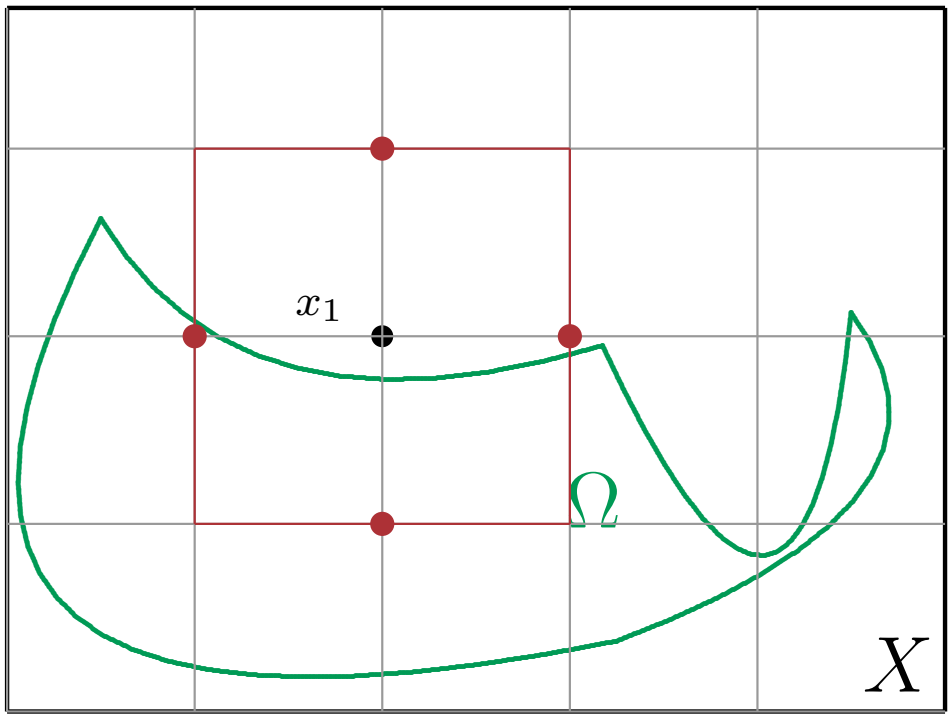


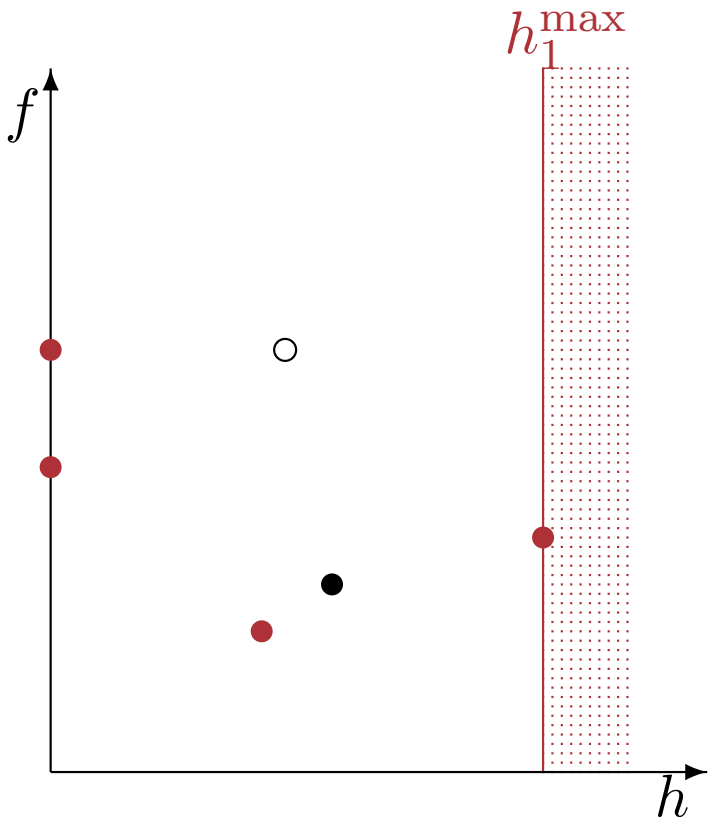
Progressive barrier algorithm : Iteration 1





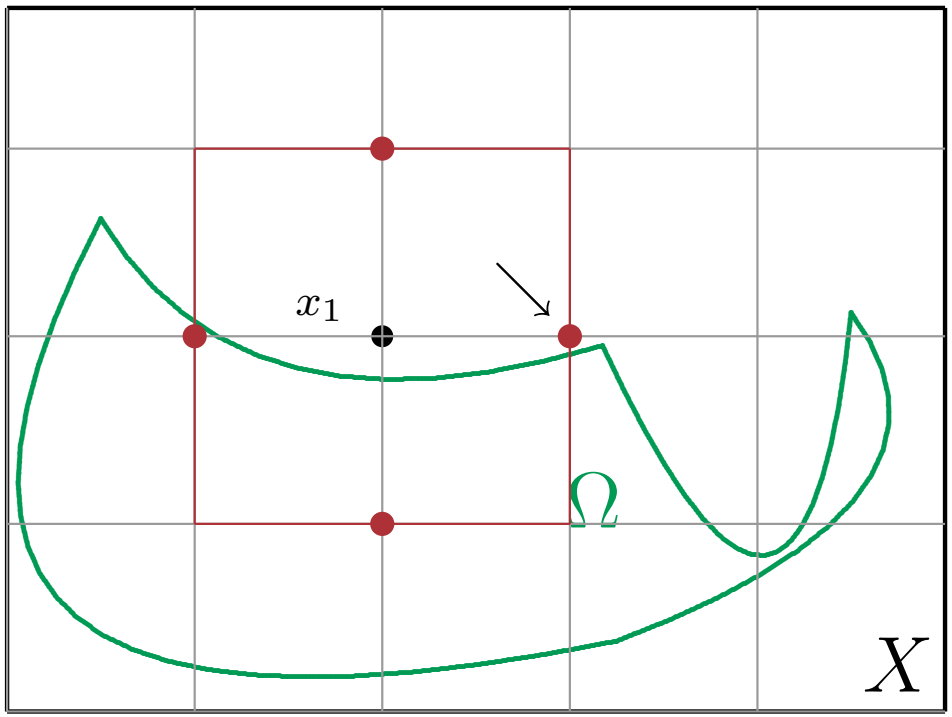
Progressive barrier algorithm : Iteration 1

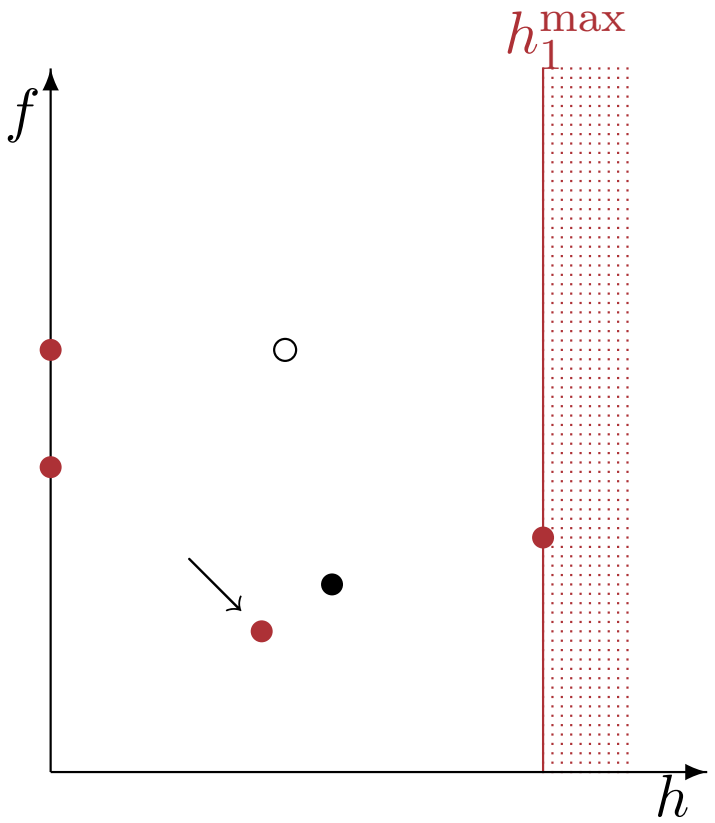




Progressive barrier algorithm : Iteration 1

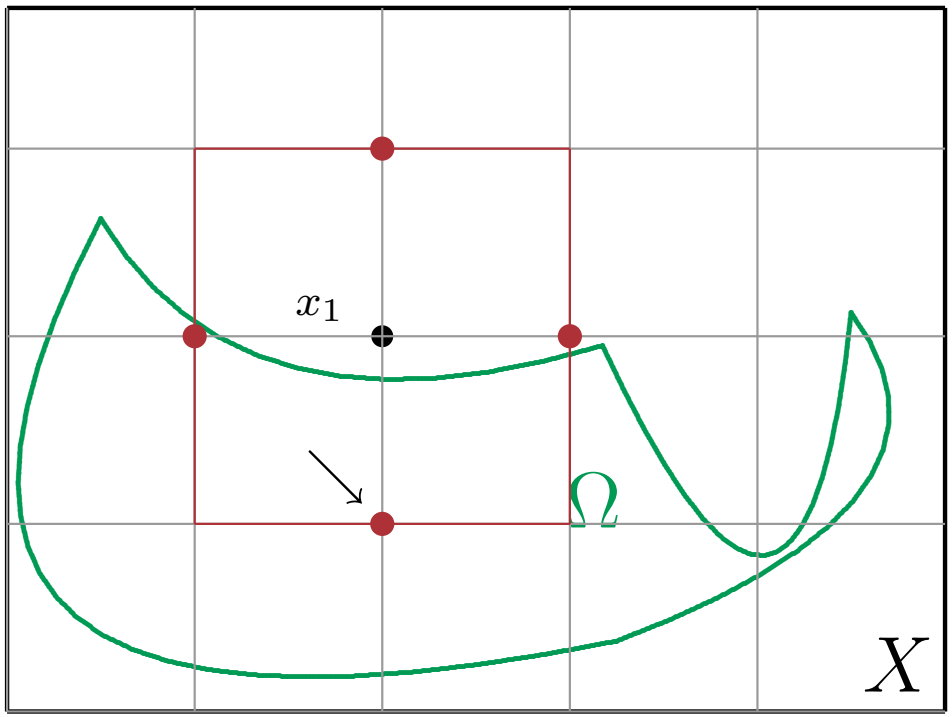
New infeasible incumbent

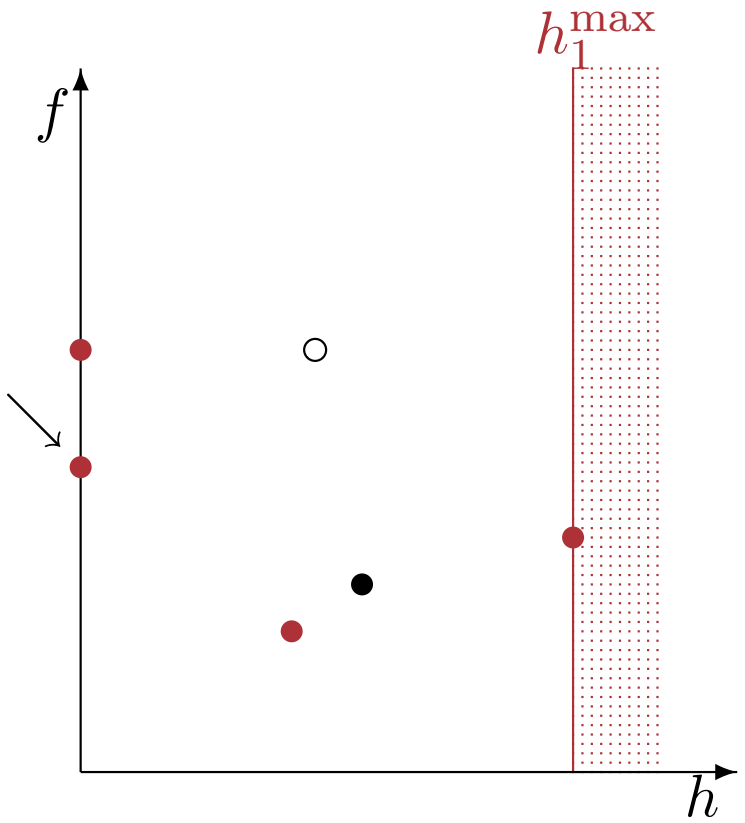




Progressive barrier algorithm : Iteration 1

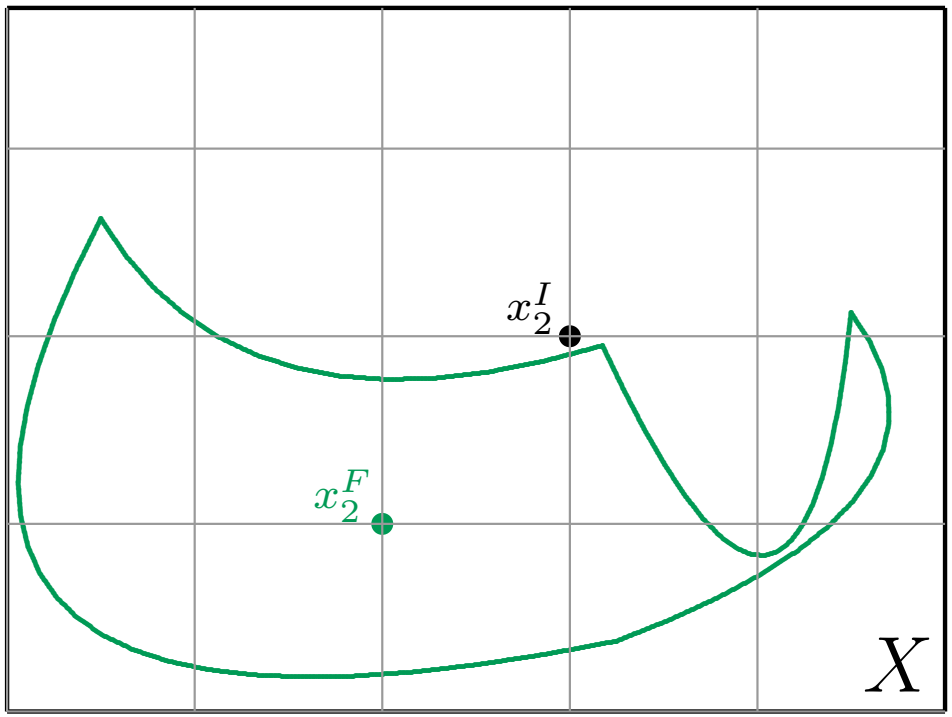
New feasible incumbent

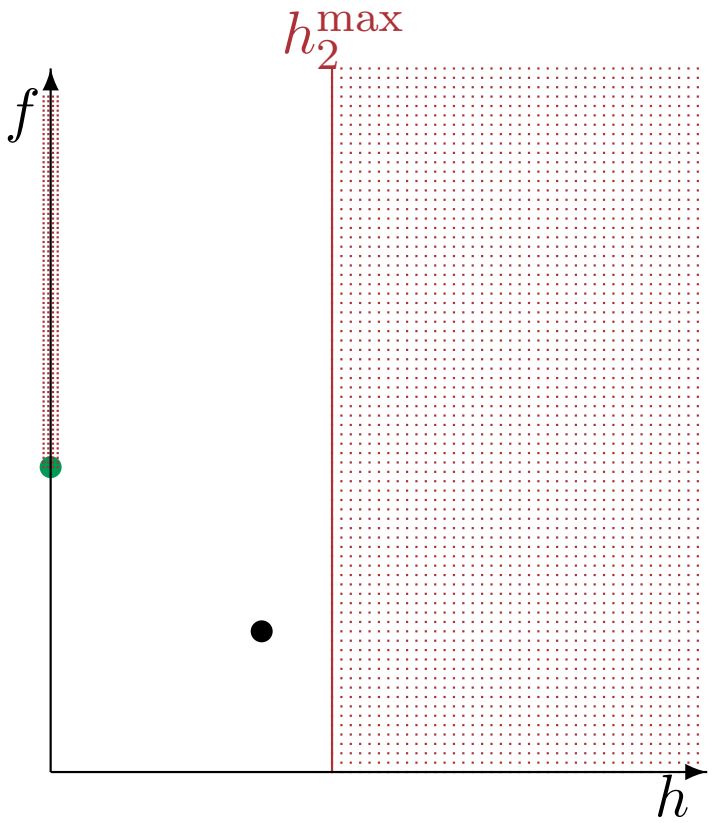




Progressive barrier algorithm : Iteration 2

$h_k^{\max}$ progressively decreases





Convergence analysis of OrthoMads under the progressive barrier

Assumptions

- At least one initial point in X is provided – but not required to be in Ω.
- All iterates belong to some compact set – it is sufficient to assume that level sets of f in X are bounded.

Convergence analysis of OrthoMads under the progressive barrier

Assumptions

- At least one initial point in X is provided – but not required to be in Ω.
- All iterates belong to some compact set – it is sufficient to assume that level sets of f in X are bounded.

Theorem

As $k \to \infty$, OrthoMads's normalized polling directions form a dense set in the unit sphere.

Limit of feasible POLL centers

Theorem

Let $\hat{x} \in \Omega$ be the limit of unsuccessful feasible poll centers $\{x_k^F\}$ on meshes that get infinitely fine. If f is Lipschitz near $\hat{x}$, then $f^\circ(\hat{x}, v) \geq 0$ for all $v \in T_\Omega^H(\hat{x})$.

Limit of feasible POLL centers

Theorem

Let $\hat{x} \in \Omega$ be the limit of unsuccessful feasible poll centers $\{x_k^F\}$ on meshes that get infinitely fine. If f is Lipschitz near $\hat{x}$, then $f^\circ(\hat{x}, v) \geq 0$ for all $v \in T_\Omega^H(\hat{x})$.

Corollary

In addition, if f is strictly differentiable near $\hat{x}$, and if Ω is regular near $\hat{x}$, then $f'(\hat{x}, v) \geq 0$ for all $v \in T_\Omega(\hat{x})$, i.e., $\hat{x}$ is a KKT point for $\min_{x \in \Omega} f(x)$.

Limit of infeasible POLL centers

Theorem

Let $\hat{x} \in X$ be the limit of unsuccessful infeasible poll centers $\{x_k^I\}$ on meshes that get infinitely fine. If h is Lipschitz near $\hat{x}$, then $h^\circ(\hat{x}, v) \geq 0$ for all $v \in T_X^H(\hat{x})$.

Limit of infeasible POLL centers

Theorem

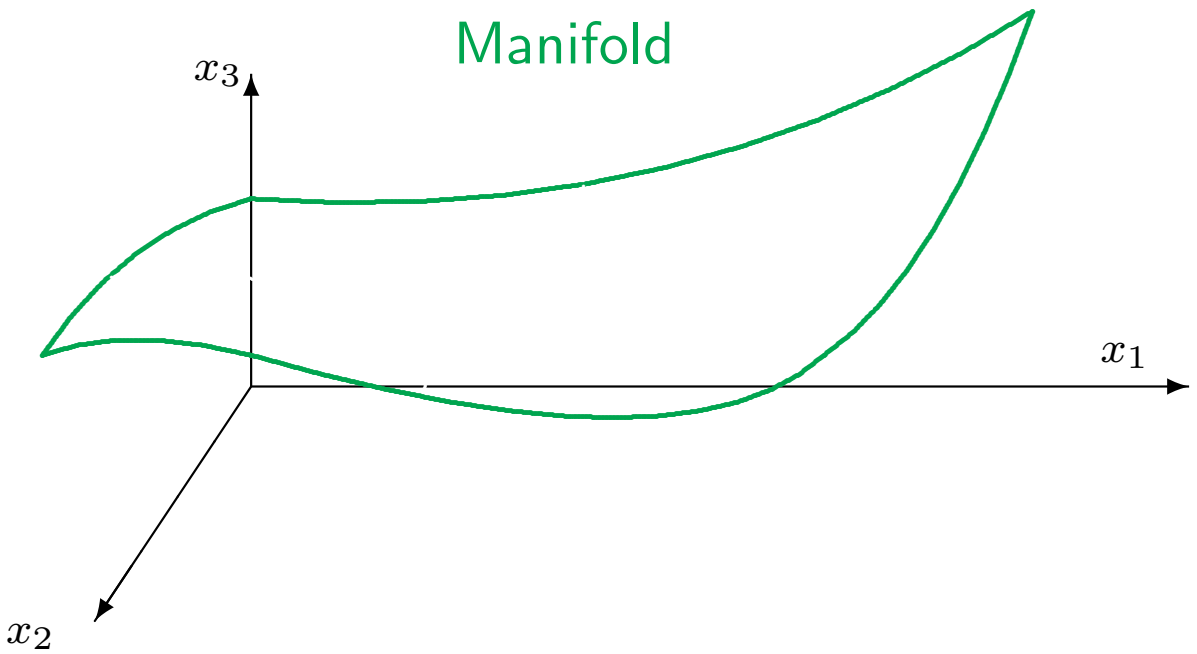
Let $\hat{x} \in X$ be the limit of unsuccessful infeasible poll centers $\{x_k^I\}$ on meshes that get infinitely fine. If h is Lipschitz near $\hat{x}$, then $h^\circ(\hat{x}, v) \geq 0$ for all $v \in T_X^H(\hat{x})$.

Corollary

In addition, if h is strictly differentiable near $\hat{x}$, and if X is regular near $\hat{x}$, then $h'(\hat{x}, v) \geq 0$ for all $v \in T_X(\hat{x})$ i.e., $\hat{x}$ is a KKT point for $\min_{x \in X} h(x)$.

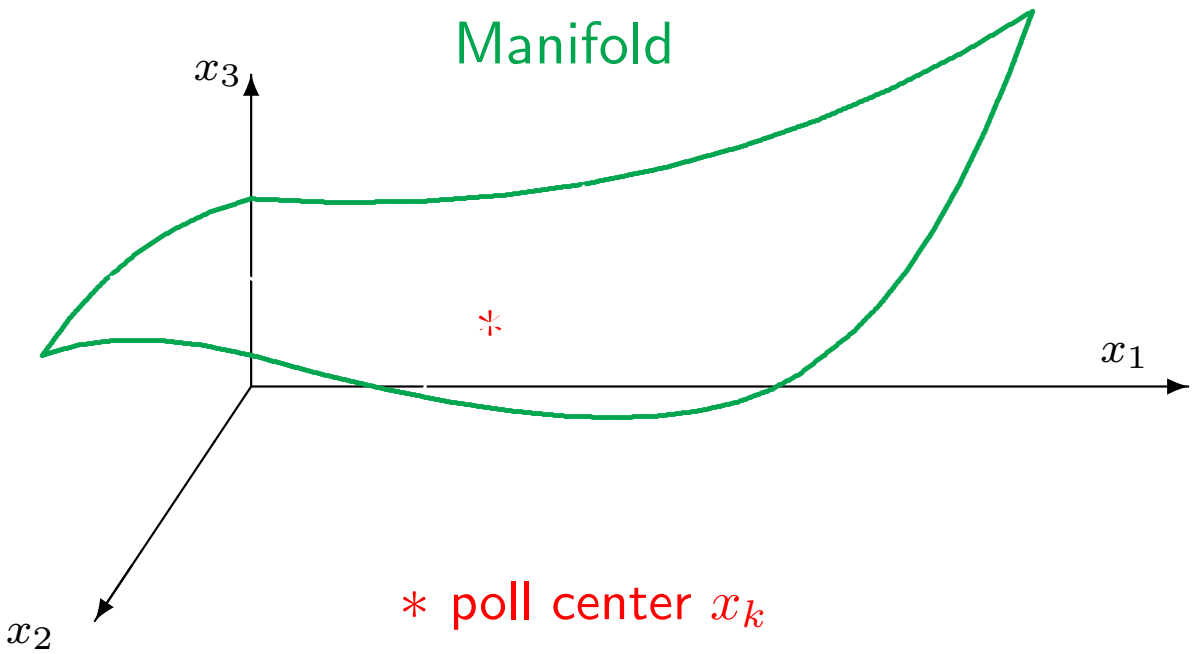
Equality constraints

- All the above ideas dealt with inequality constraints.
- Dreisigmeyer recently proposed to construct a mesh using geodesics of the Riemannian manifold formed by equalities (not necessarily linear). Knowledge of the equalities is necessary.



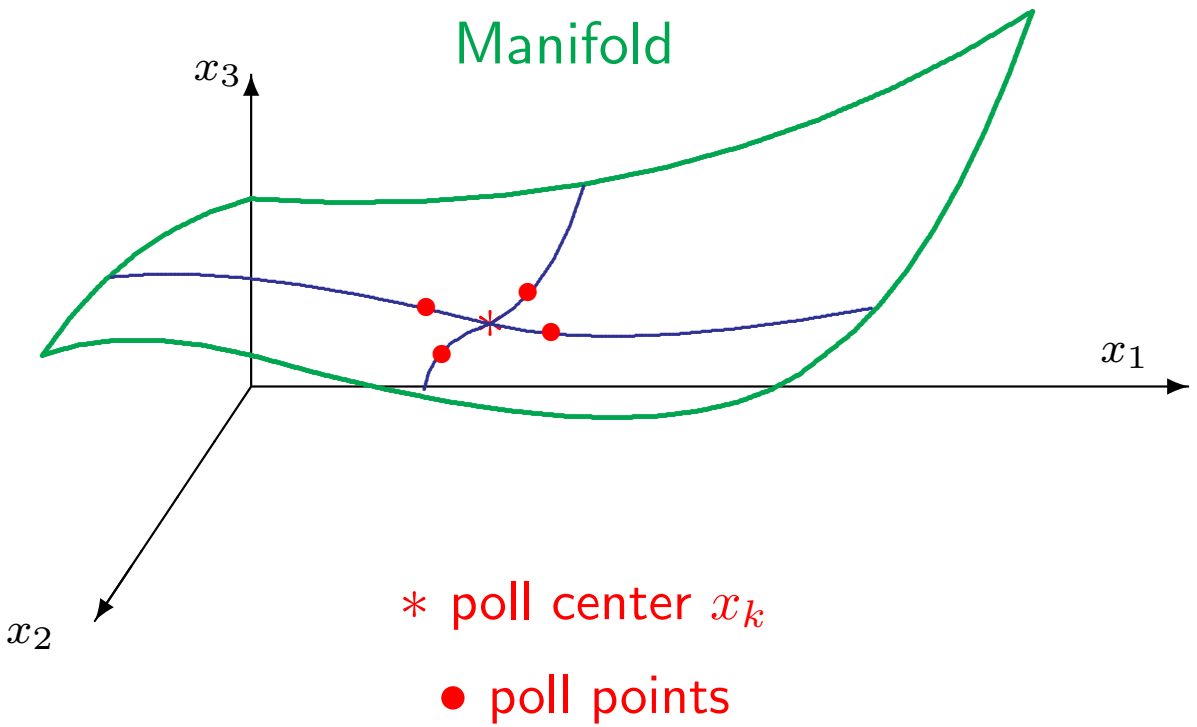
Equality constraints

- All the above ideas dealt with inequality constraints.
- Dreisigmeyer recently proposed to construct a mesh using geodesics of the Riemannian manifold formed by equalities (not necessarily linear). Knowledge of the equalities is necessary.



Equality constraints

- All the above ideas dealt with inequality constraints.
- Dreisigmeyer recently proposed to construct a mesh using geodesics of the Riemannian manifold formed by equalities (not necessarily linear). Knowledge of the equalities is necessary.



Presentation outline

Surrogate functions

Definition

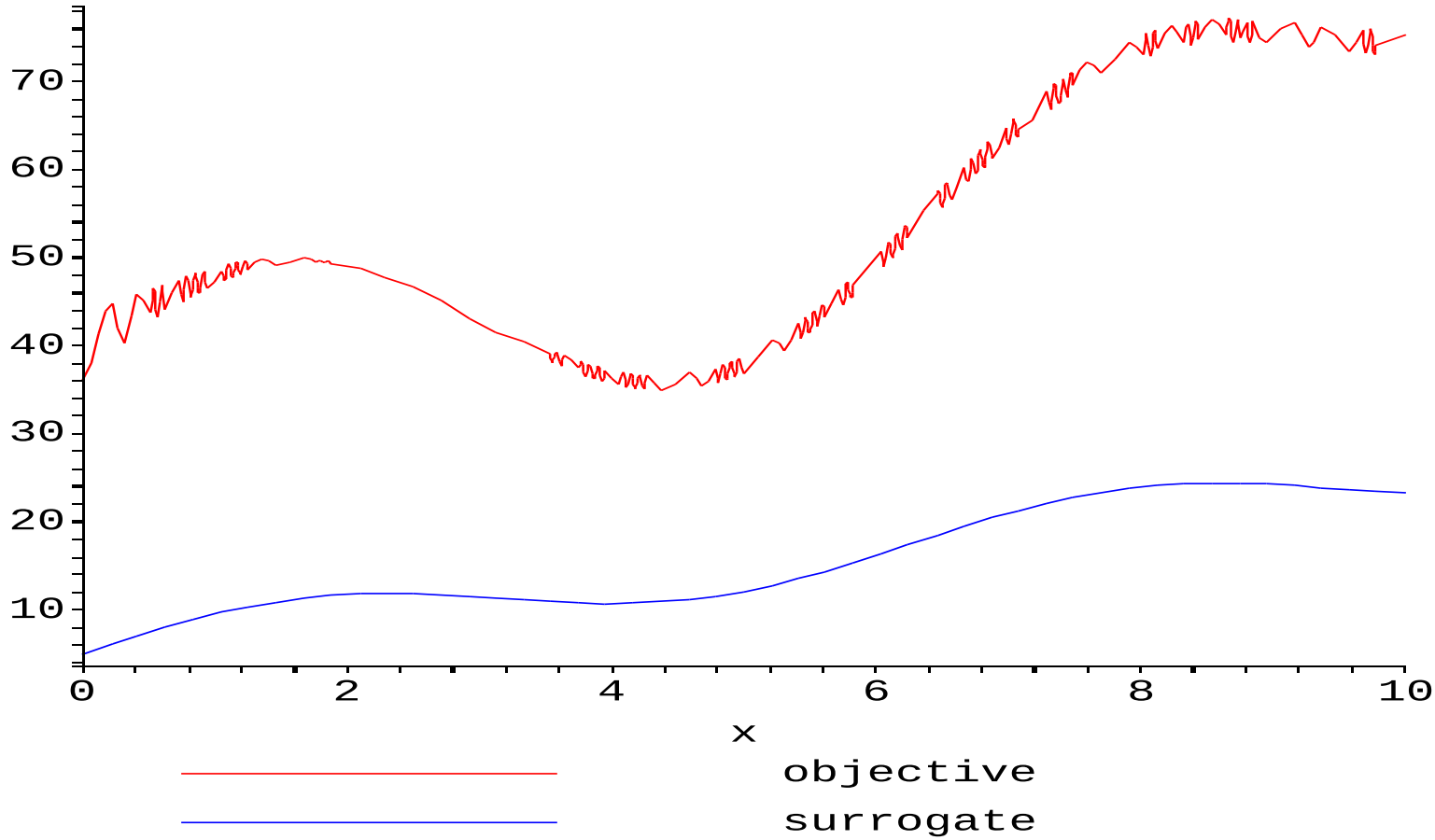
A surrogate s_f of the function f is a function that shares some similarities with f BUT is much cheaper to evaluate.

Examples:

- surfaces obtained from f values at selected sites
- simplified physical models
- lower fidelity models
- a function that evaluates something similar

An example in $\mathbb{R}^1$ of an excellent surrogate

A surrogate s_f is not necessarily a good approximation of the truth f



Example: Minimize f, the total time required to perform 1000 tasks. s_f might be the time required to perform 10 tasks.

A strawman surrogate approach

- Given surrogates s_f and S_Ω of both the objective function and the constraints.
- Minimize $s_f(x)$ for $S_\Omega(x) \leq 0$ to obtain x_s.
  Every user has their favorite approach for this part

A strawman surrogate approach

- Given surrogates s_f and S_Ω of both the objective function and the constraints.
- Minimize $s_f(x)$ for $S_\Omega(x) \leq 0$ to obtain x_s.
  Every user has their favorite approach for this part
- Compute $f(x_s), C(x_s)$ to determine if improvement has been made over the best x found to date.

A strawman surrogate approach

- Given surrogates s_f and S_Ω of both the objective function and the constraints.
- Minimize $s_f(x)$ for $S_\Omega(x) \leq 0$ to obtain x_s.
  Every user has their favorite approach for this part
- Compute $f(x_s), C(x_s)$ to determine if improvement has been made over the best x found to date.
  But, what if no improvement was found?

Response surface global models

- Choose a space filling set of points.
  – say by Latin hypercube or orthogonal array sampling.
- Run the expensive simulations at these sites.
- Interpolate or smooth to f, C on this data.
- May be able to interpolate to gradients as well – Alexandrov

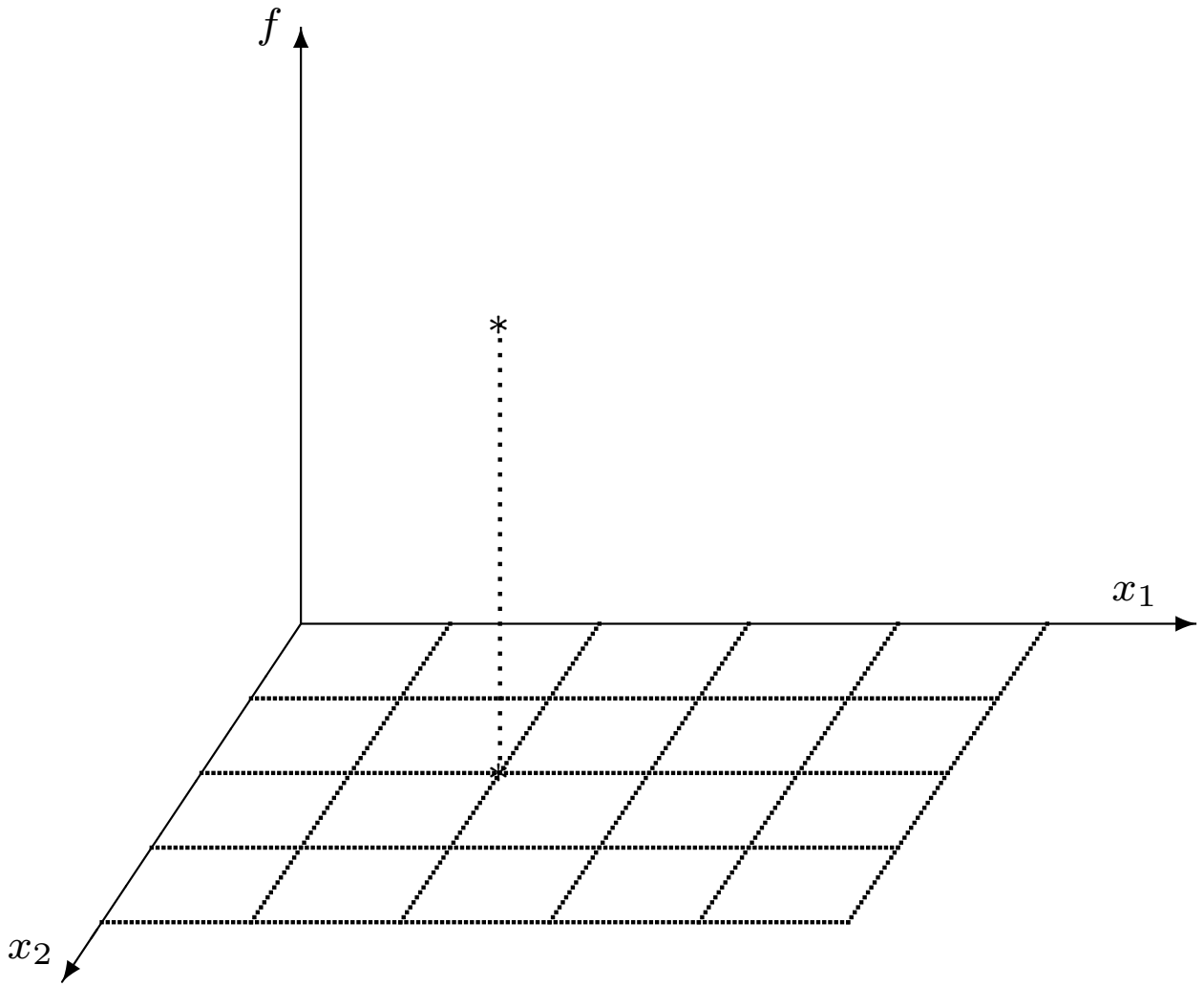
Form of the DACE interpolants

Polynomials introduce extraneous extremes that trap strawman
Spline-like DACE interpolants can be written

$$\hat{s}(x) = \sum_{i=1}^{d} w_i(x) f(x_i) \ .$$

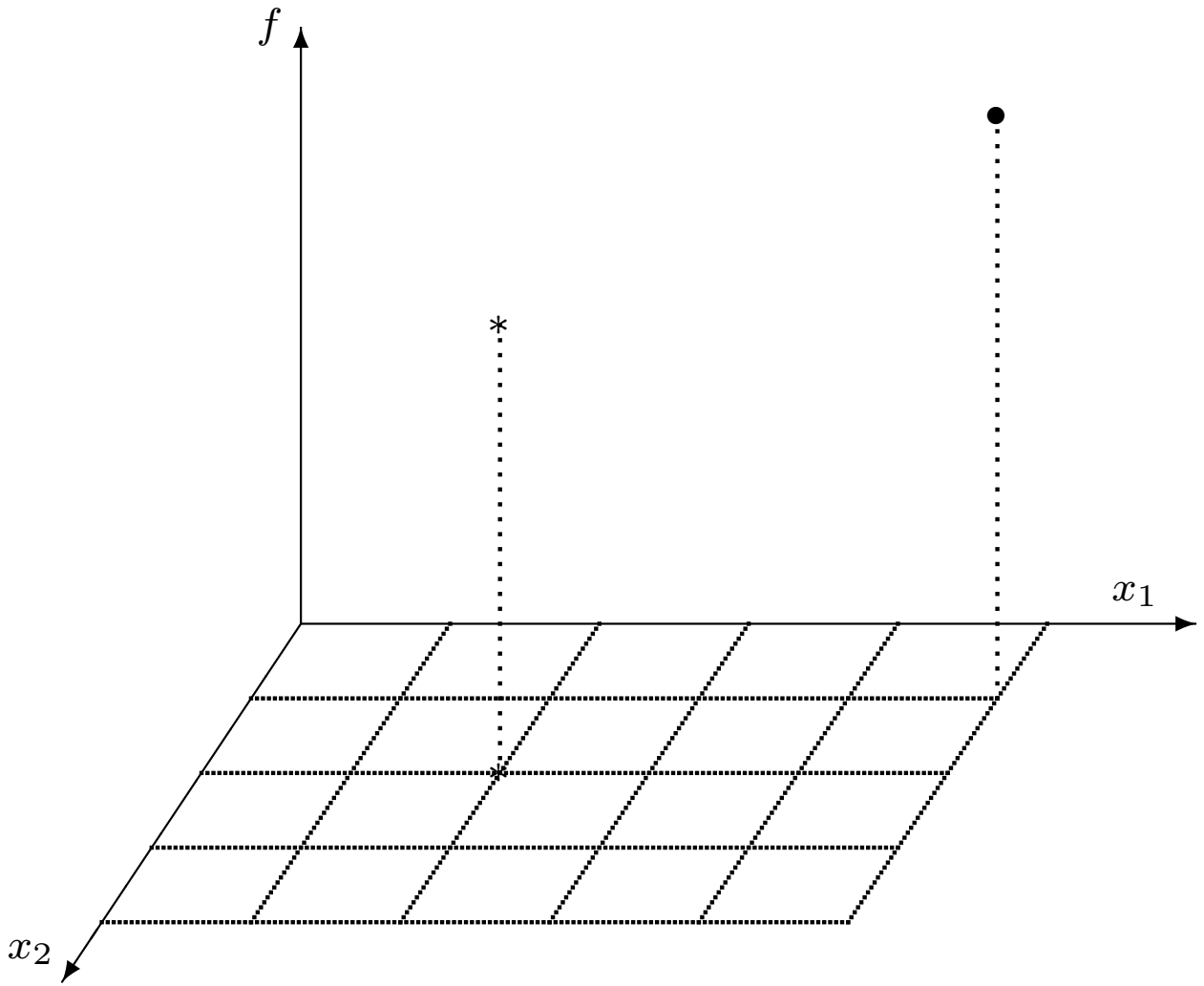
DACE predictions at any x are weighted averages of f at the data sites.
Weight depends on how far site is from x.
Correlation parameters for each site are estimated.

Surrogate Management Framework



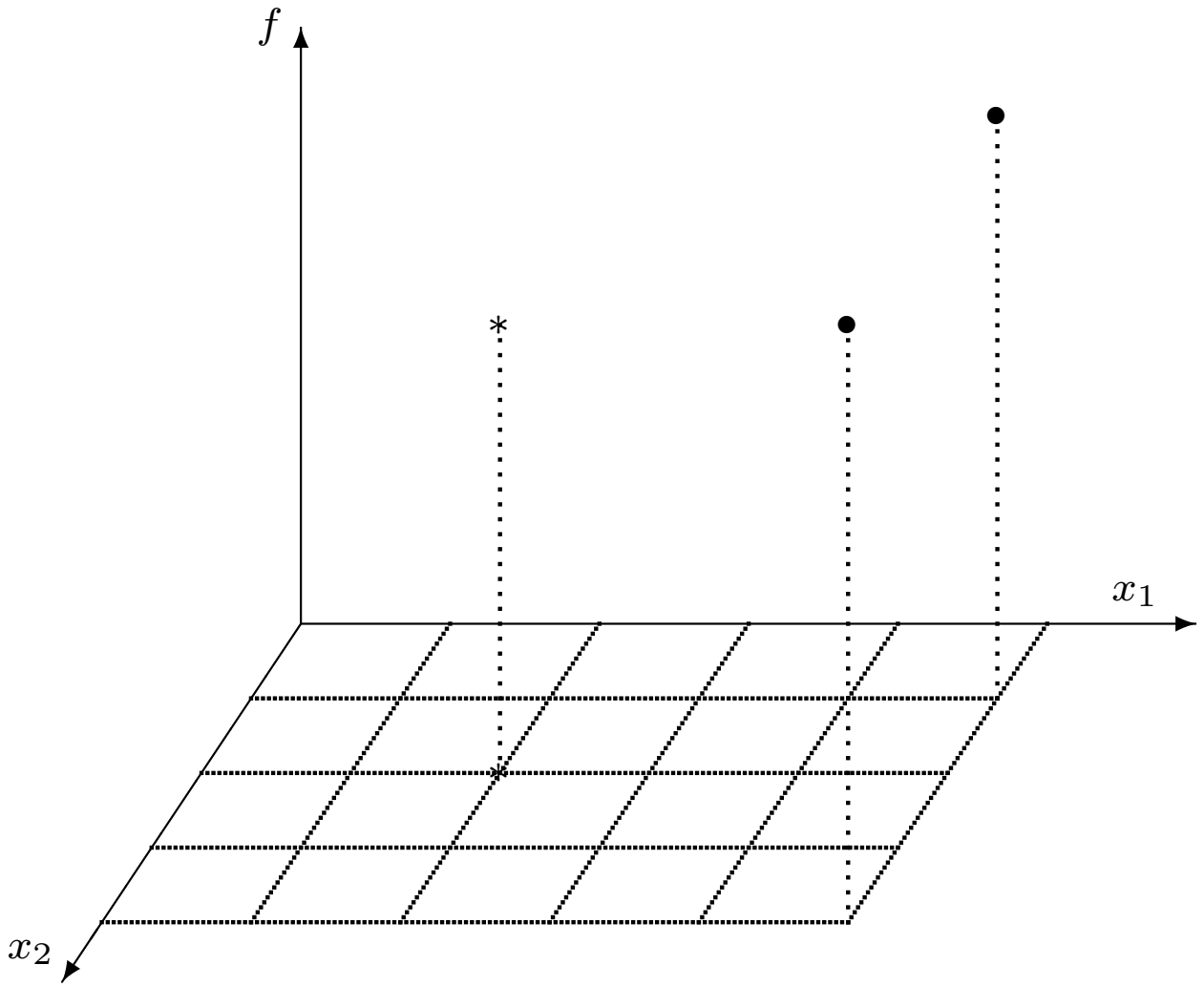
$*$ is the incumbent solution

Surrogate Management Framework



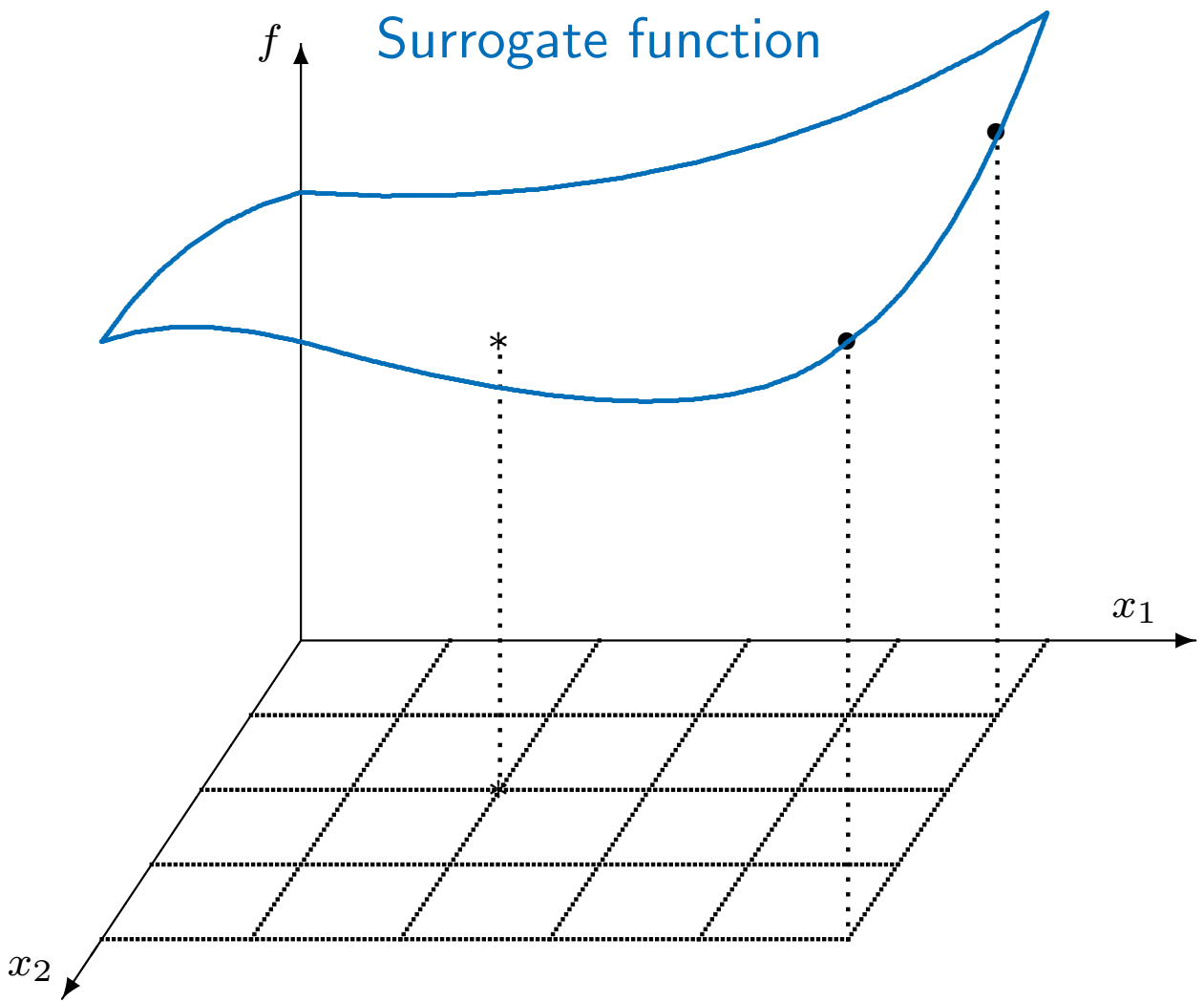
$*$ is the incumbent solution

Surrogate Management Framework



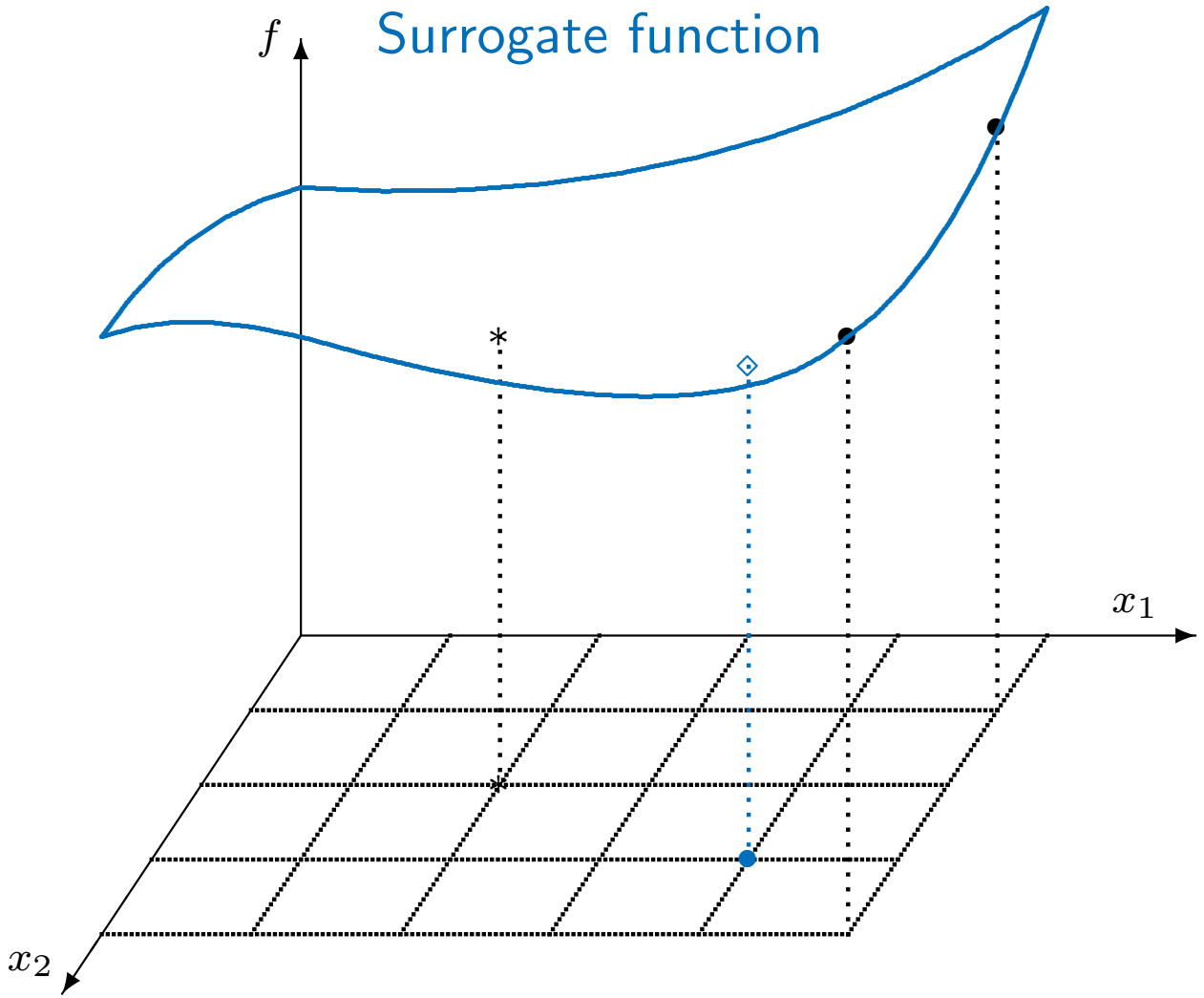
$*$ is the incumbent solution

Surrogate Management Framework



$*$ is the incumbent solution

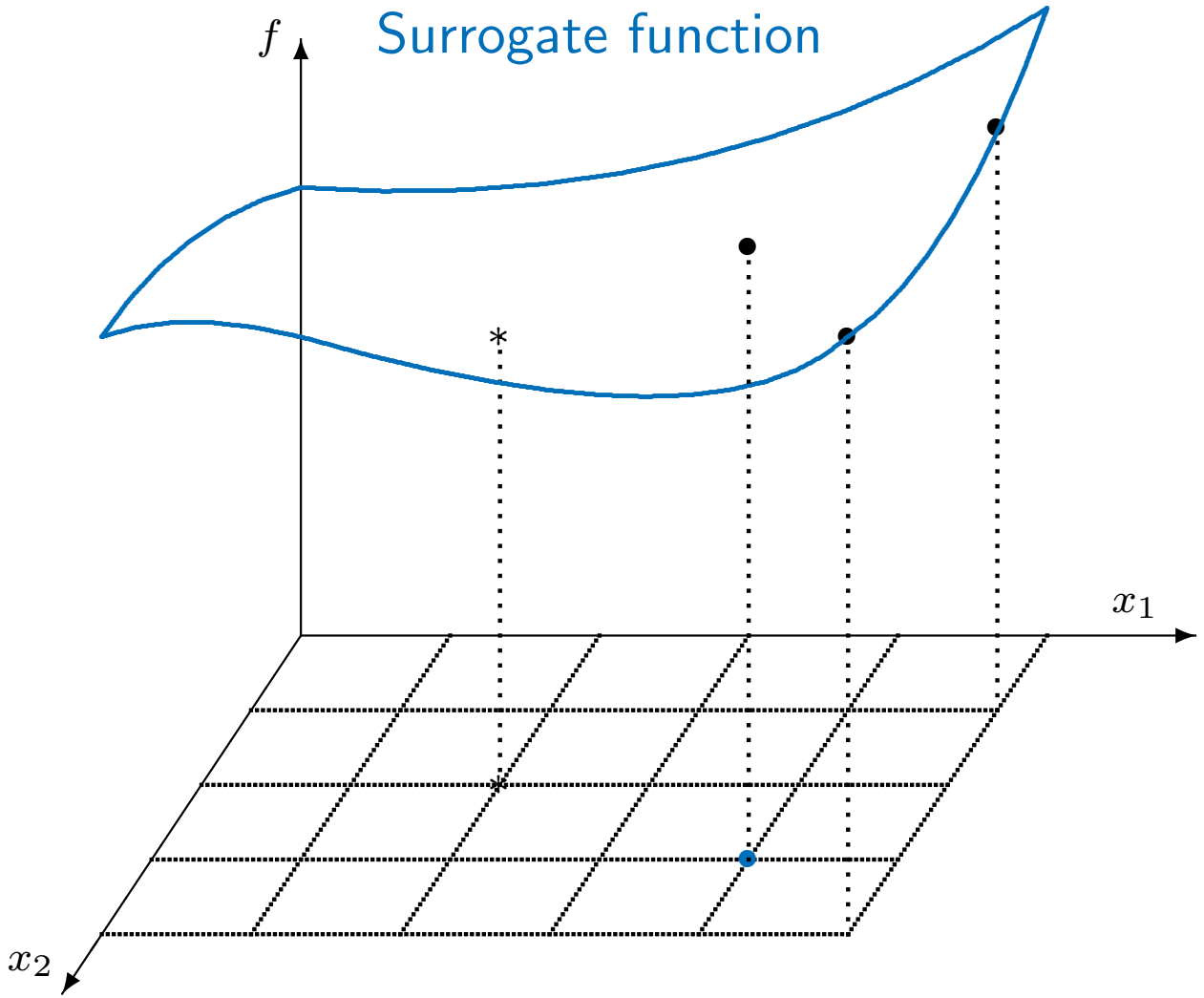
Surrogate Management Framework



$*$ is the incumbent solution

Trial point produced using the surrogate

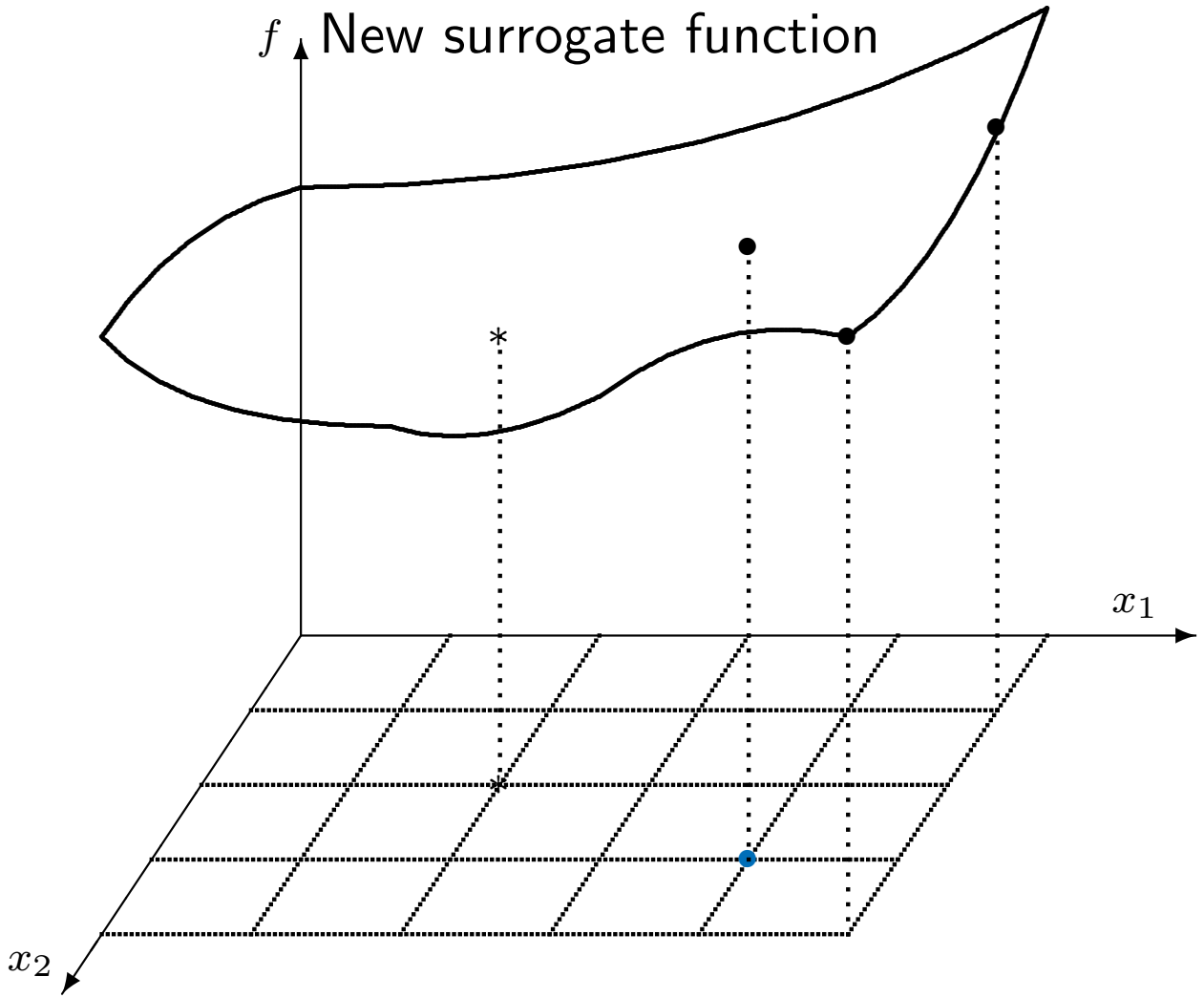
Surrogate Management Framework



$*$ is the incumbent solution

f is evaluated at the trial point

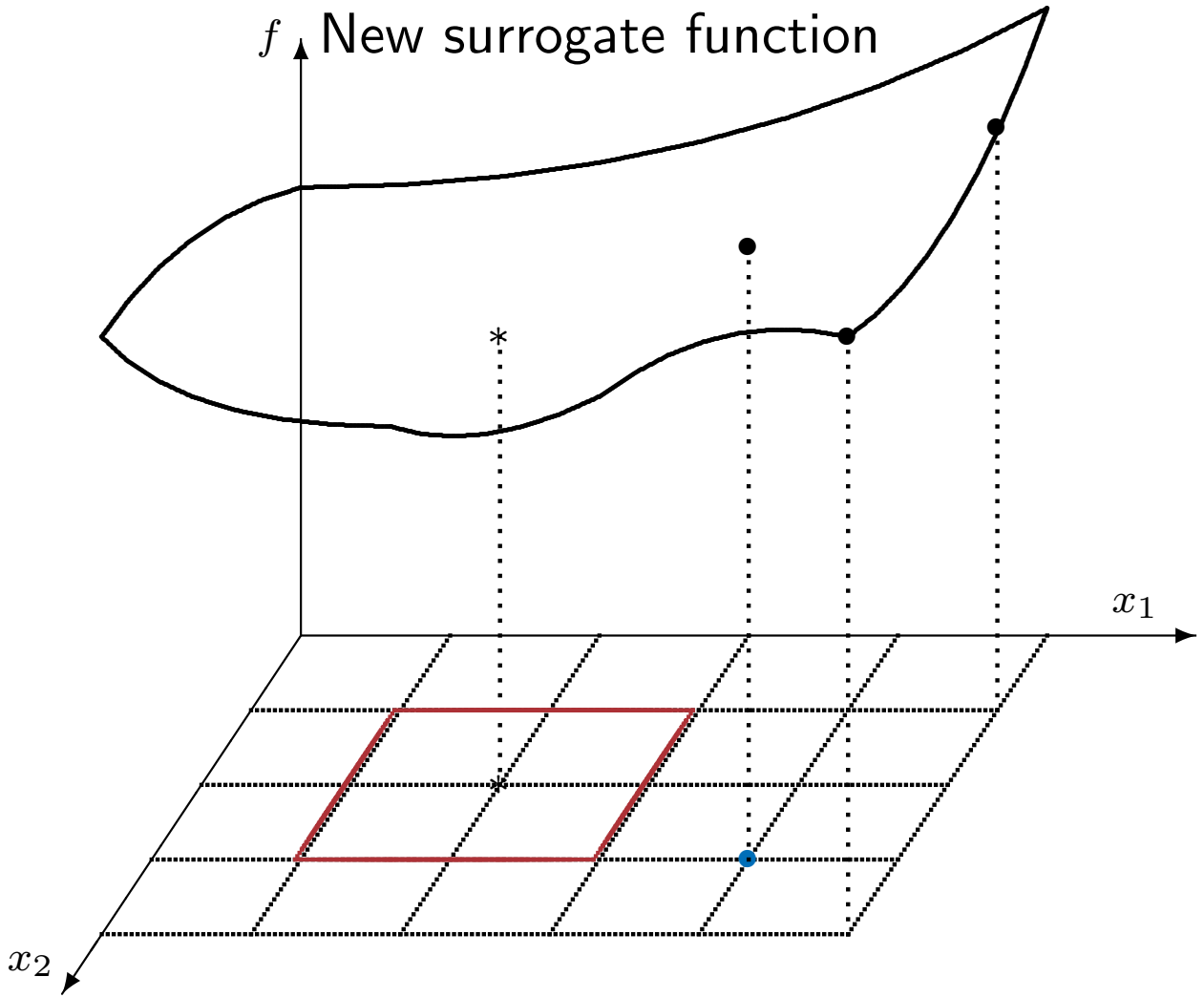
Surrogate Management Framework



$*$ is the incumbent solution

the surrogate function is updated

Surrogate Management Framework

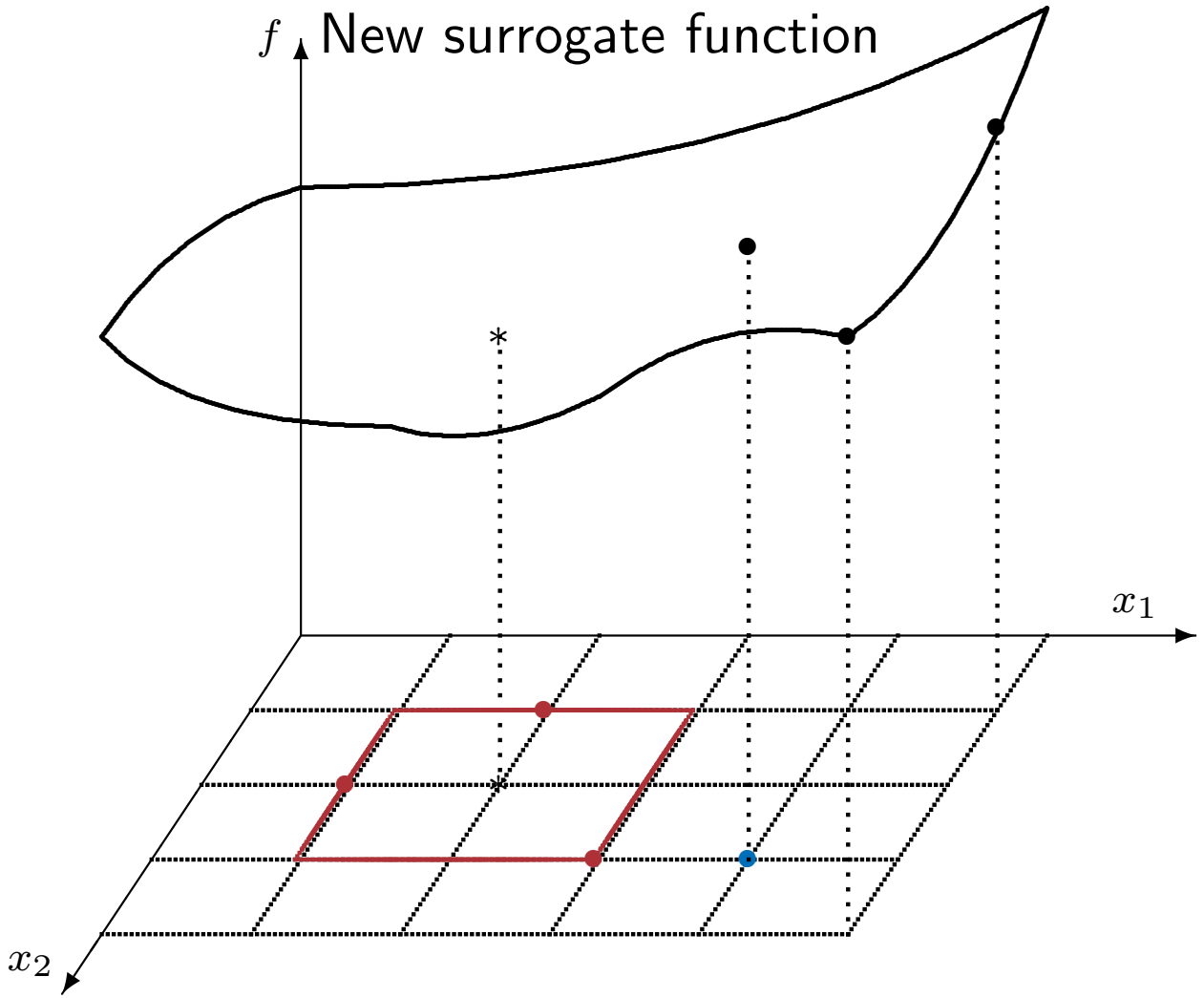


$*$ is the incumbent solution

the surrogate function is updated

Poll around the incumbent (order based on surrogate)

Surrogate Management Framework

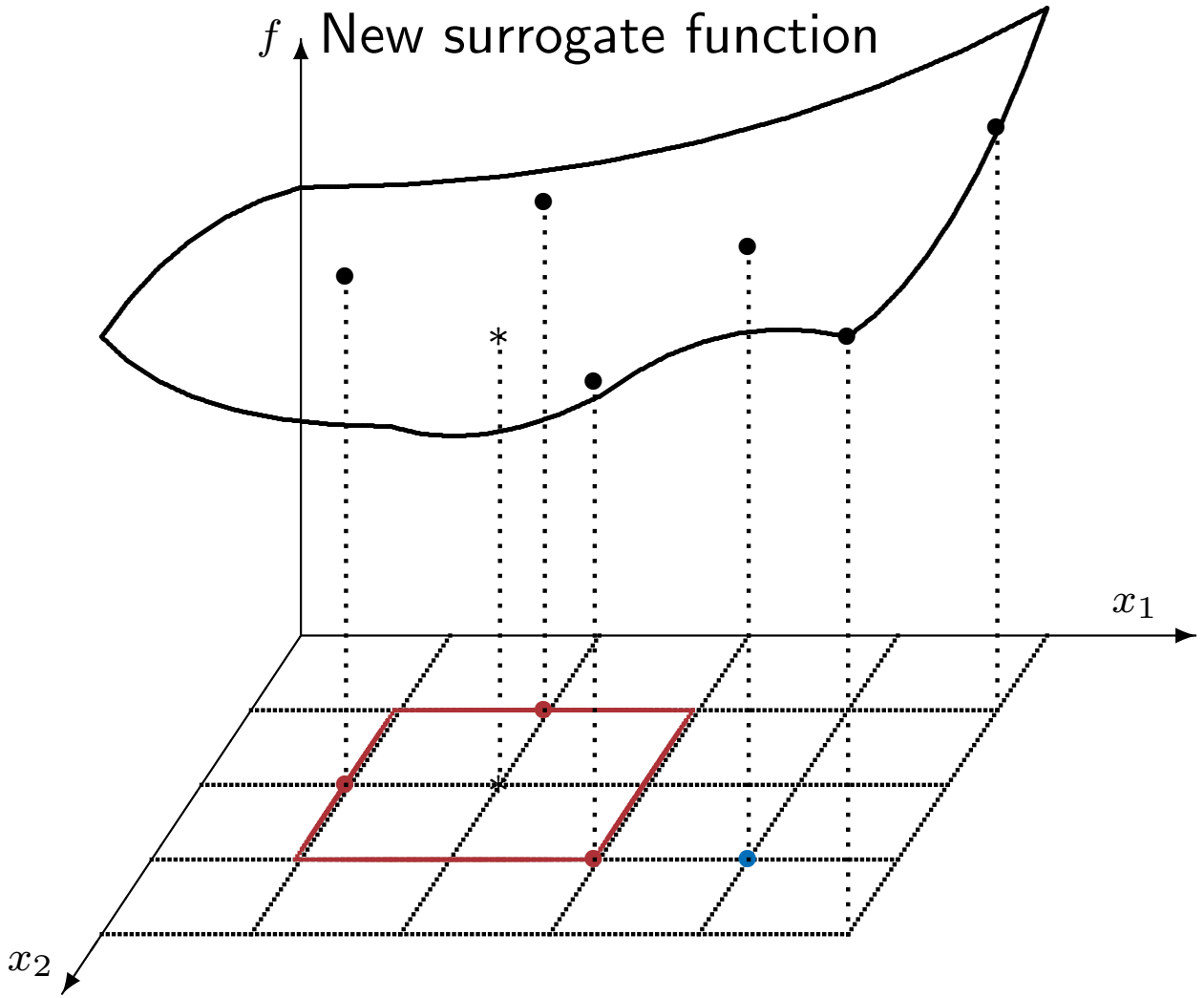


$*$ is the incumbent solution

the surrogate function is updated

Poll around the incumbent (order based on surrogate)

Surrogate Management Framework



$*$ is the incumbent solution

the surrogate function is updated

Poll around the incumbent (order based on surrogate)

Some ways to use the surrogate

- MADS is applied to minimize the surrogate function s_f from x_0, leading to the solution x^s. The functions f and C are then evaluated at x^s. MADS is then applied to minimize s_f from the starting points $\{x_0, x^s\}$.

Some ways to use the surrogate

- MADS is applied to minimize the surrogate function s_f from x_0, leading to the solution x^s. The functions f and C are then evaluated at x^s. MADS is then applied to minimize s_f from the starting points $\{x_0, x^s\}$.
- At each iteration, the search and poll trial points are sorted according to their surrogate function values. Promising trial points are evaluated first.

Some ways to use the surrogate

- MADS is applied to minimize the surrogate function s_f from x_0, leading to the solution x^s. The functions f and C are then evaluated at x^s. MADS is then applied to minimize s_f from the starting points $\{x_0, x^s\}$.
- At each iteration, the search and poll trial points are sorted according to their surrogate function values. Promising trial points are evaluated first.
- At each iteration, the search step returns a list of trial points. The true functions are evaluated only at those having a good surrogate value.

Some ways to use the surrogate

- MADS is applied to minimize the surrogate function s_f from x_0, leading to the solution x^s. The functions f and C are then evaluated at x^s. MADS is then applied to minimize s_f from the starting points $\{x_0, x^s\}$.
- At each iteration, the search and poll trial points are sorted according to their surrogate function values. Promising trial points are evaluated first.
- At each iteration, the search step returns a list of trial points. The true functions are evaluated only at those having a good surrogate value.
- Surrogates are updated when new information about the truth is available. This may be based on an interpolation model of $f - s_f$.

Some ways to use the surrogate

- MADS is applied to minimize the surrogate function s_f from x_0, leading to the solution x^s. The functions f and C are then evaluated at x^s. MADS is then applied to minimize s_f from the starting points $\{x_0, x^s\}$.
- At each iteration, the search and poll trial points are sorted according to their surrogate function values. Promising trial points are evaluated first.
- At each iteration, the search step returns a list of trial points. The true functions are evaluated only at those having a good surrogate value.
- Surrogates are updated when new information about the truth is available. This may be based on an interpolation model of $f - s_f$.
- Convergence analysis is not altered by the use of surrogates.

Presentation outline

Constrained TR interpolation-based methods

A number of pratical trust-region SQP type methods have been proposed (see available software).

The main ideas are the following:

- Use of quadratic models for the Lagrangian function.
- Models for the constraints can be linear, or quadratic (especially when function evaluations are expensive but leading to quadratically constrained TR subproblems).
- Globalization requires a merit function (typically f) or a filter.
- Open constraints have no influence on the poisedness for f.

Constrained TR interpolation-based methods

A number of pratical trust-region SQP type methods have been proposed (see available software).

The main ideas are the following:

- Use of quadratic models for the Lagrangian function.
- Models for the constraints can be linear, or quadratic (especially when function evaluations are expensive but leading to quadratically constrained TR subproblems).
- Globalization requires a merit function (typically f) or a filter.
- Open constraints have no influence on the poisedness for f.

Currently, there is no (non-obvious) convergence theory developed for TR interpolation-based methods (in the constrained case).

For example, for (i) linear or box constraints (closed) and (ii) open constraints without derivatives, the unconstrained theory should be reasonably easy to adapt.

Towards global optimization

(A) One direction is along radial basis functions.

(B) Another is to adapt direct search:

- Use the search step of the search-poll framework to incorporate a dissemination method or heuristic for global optimization purposes.

  $\longrightarrow$ Such schemes provide a wider exploration of the variable domain or feasible region.

  $\longrightarrow$ Examples are particle swarm and variable neighborhood search.

- Global convergence (of the overall algorithm) to a stationary point is still guaranteed.
- Robustness and efficiency of the heuristic (used in the search step) are generally improved.

Software (directional direct search)

APPSPACK: Asynchronous parallel pattern search
http://software.sandia.gov/appspack

NOMAD: Generalized pattern search and mesh adaptive direct search
http://www.gerad.ca/NOMAD
http://www.afit.edu/en/ENC/Faculty/MAbramson/NOMADm.html

Mads: Matlab's implementation of the LtMads method – gads toolbox
http://www.mathworks.com/products/gads

SID-PSM: Generalized pattern search guided by simplex derivatives
http://www.mat.uc.pt/sid-psm

Software (direct search — others in Matlab)

Iterative Methods for Optimization: `Matlab` Codes
Hooke-Jeeves, multidirectional search, and Nelder-Mead methods
`http://www4.ncsu.edu/~ctk/matlab_darts.html`

The Matrix Computation Toolbox
Multidirectional search, alternating directions, and Nelder-Mead methods
`http://www.maths.manchester.ac.uk/~higham/mctoolbox`

`fminsearch`: `Matlab`'s implementation of the Nelder-Mead method
`http://www.mathworks.com/access/helpdesk/help/techdoc/ref/fminsearch.html`

Software (trust-region interpolation based methods)

DFO: Trust-region interpolation-based method
http://www.coin-or.org/projects.html

UOBYQA, NEWUOA: Trust-region interpolation-based methods
mjdp@cam.ac.uk

WEDGE: Trust-region interpolation-based method
http://www.ece.northwestern.edu/~nocedal/wedge.html

Software (others)

BOOSTERS: Trust-region interpolation-based method (based on radial basis functions)
http://roso.epfl.ch/rodrigue/boosters.htm

CONDOR: Trust-region interpolation-based method (version of UOBYQA in parallel)
http://www.applied-mathematics.net/optimization/CONDORdownload.html

Implicit Filtering: implicit filtering method
http://www4.ncsu.edu/~ctk/iffco.html

Software (Global Optimization)

`PSwarm`: Coordinate search and particle swarm for global optimization
`http://www.norg.uminho.pt/aivaz/pswarm`

Benchmarking

- K. R. Fowler, J. P. Reese, C. E. Kees, J. E. Dennis Jr., C. T. Kelley, C. T. Miller, C. Audet, A. J. Booker, G. Couture, R. W. Darwin, M. W. Farthing, D. E. Finkel, J. M. Gablonsky, G. Gray, and T. G. Kolda, *A comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems*, Advances in Water Resources, 31(2): 743-757, 2008.

- J. J. Moré and S. M. Wild, *Benchmarking derivative-free optimization algorithms*, Tech. Report ANL/MCS-P1471-1207, Mathematics and Computer Science Division, Argonne National Laboratory, USA, 2007.

Main references

- C. Audet and J. E. Dennis, Jr., *Mesh adaptive direct search algorithms for constrained optimization*, SIAM Journal on Optimization, 17 (2006) 188–217.

- A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*, MPS-SIAM Series on Optimization, SIAM, Philadelphia, to appear in 2008.

- 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) 385–482.