

CMA-ES and Advanced Adaptation Mechanisms

Youhei Akimoto¹ & Nikolaus Hansen²

1. University of Tsukuba, Japan

2. Inria & Ecole Polytechnique, France

akimoto@cs.tsukuba.ac.jp
forename.lastname@inria.fr

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA © 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-9268-6/22/07...\$15.00 <https://doi.org/10.1145/3520304.3533648>



We are happy to answer questions at any time.

Topics

1. What makes an optimization problem difficult to solve?

2. How does the CMA-ES work?

- Normal Distribution, Rank-Based Recombination
- Step-Size Adaptation
- Covariance Matrix Adaptation

3. What can/should the users do for the CMA-ES to work effectively on their problem?

- Choice of problem formulation and encoding (not covered)
- Choice of initial solution and initial step-size
- Restarts, Increasing Population Size
- Restricted Covariance Matrix

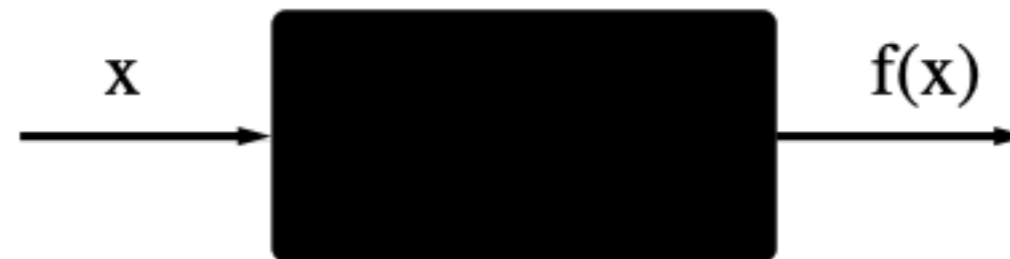
Problem Statement

Continuous Domain Search/Optimization

- Task: **minimize** an **objective function** (*fitness function, loss function*) in continuous domain

$$f : \mathcal{X} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}, \quad \mathbf{x} \mapsto f(\mathbf{x})$$

- **Black Box** scenario (direct search scenario)



- ▶ gradients are not available or not useful
- ▶ problem domain specific knowledge is used only within the black box, e.g. within an appropriate encoding
- Search **costs**: number of function evaluations

Problem Statement

Continuous Domain Search/Optimization

- Goal

- ▶ fast convergence to the global optimum
- ▶ solution x with **small function value** $f(x)$ with **least search cost** ... or to a robust solution x
there are two conflicting objectives

- Typical Examples

- ▶ shape optimization (e.g. using CFD)
 - ▶ model calibration
 - ▶ parameter calibration
- curve fitting, airfoils
biological, physical
controller, plants, images

- Difficulties

- ▶ exhaustive search is infeasible
- ▶ naive random search takes too long
- ▶ deterministic search is not successful / takes too long

Approach: stochastic search, Evolutionary Algorithms

Topics

1. What makes an optimization problem difficult to solve?

2. How does the CMA-ES work?

- Normal Distribution, Rank-Based Recombination
- Step-Size Adaptation
- Covariance Matrix Adaptation

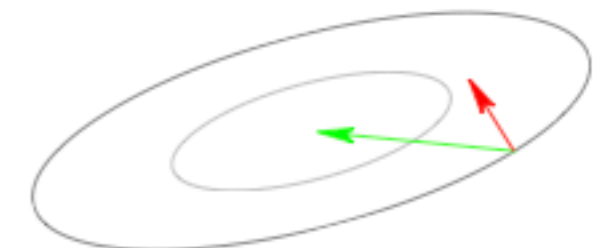
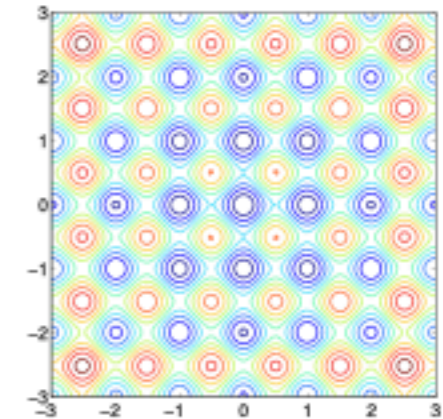
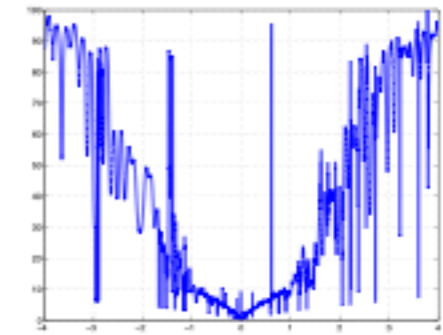
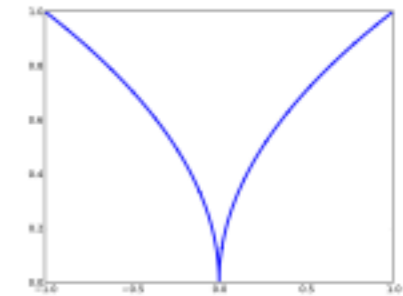
3. What can/should the users do for the CMA-ES to work effectively on their problem?

- Choice of problem formulation and encoding (not covered)
- Choice of initial solution and initial step-size
- Restarts, Increasing Population Size
- Restricted Covariance Matrix

What Makes a Function Difficult to Solve?

Why stochastic search?

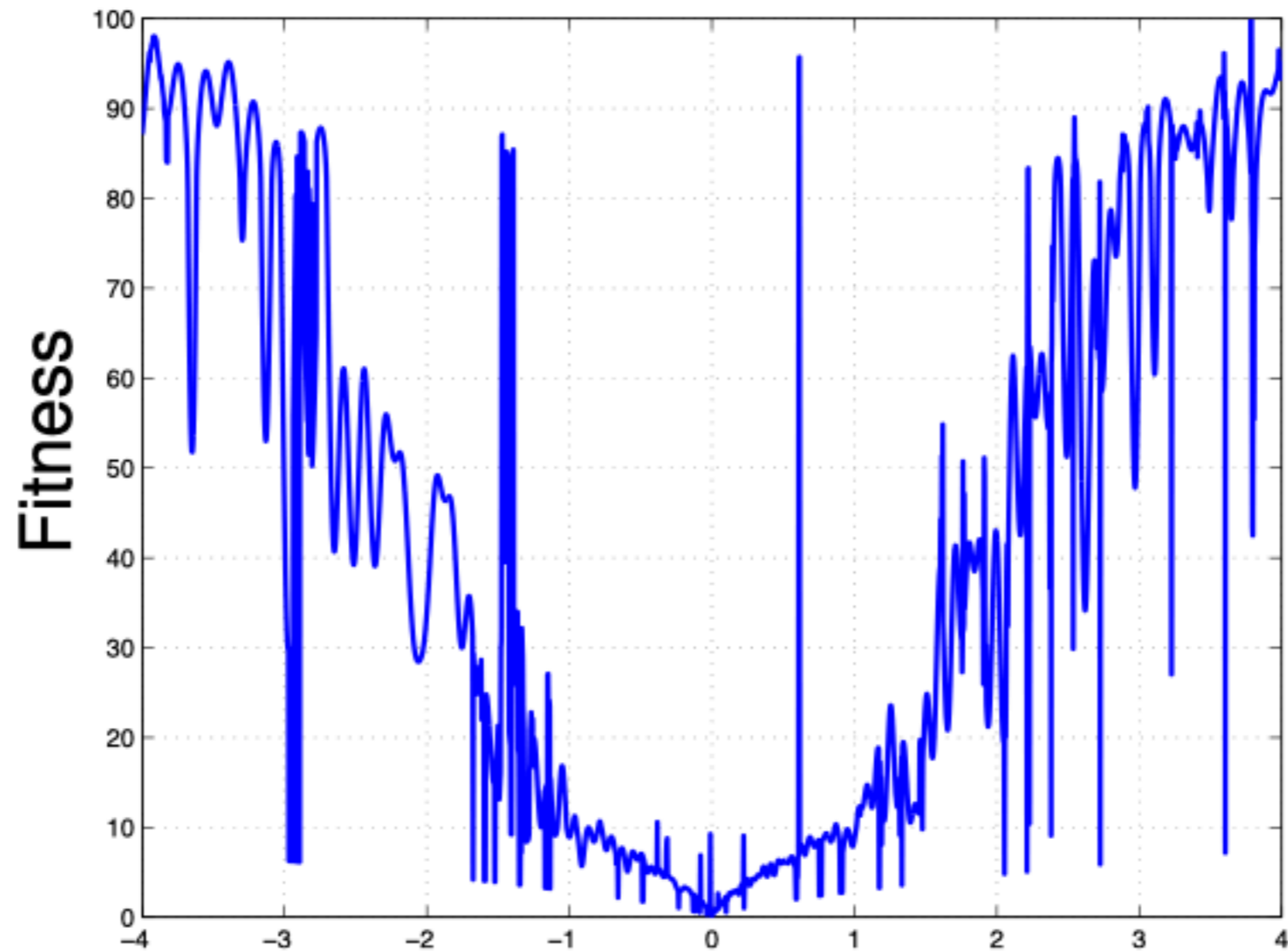
- non-linear, non-quadratic, non-convex
on linear and quadratic functions much better search policies are available
- ruggedness
non-smooth, discontinuous, multimodal, and/or noisy function
- dimensionality (size of search space)
(considerably) larger than three
- non-separability
dependencies between the objective variables
- ill-conditioning
- non-smooth level sets



gradient direction Newton direction

Ruggedness

non-smooth, discontinuous, multimodal, and/or noisy



cut from a 5-D example, (easily) solvable with evolution strategies

Curse of Dimensionality

The term *Curse of dimensionality* (Richard Bellman) refers to problems caused by the **rapid increase in volume** associated with adding extra dimensions to a (mathematical) space.

Example: Consider placing 20 points equally spaced onto the interval $[0, 1]$. Now consider the 10-dimensional space $[0, 1]^{10}$. To get **similar coverage** in terms of distance between adjacent points requires $20^{10} \approx 10^{13}$ points. 20 points appear now as isolated points in a vast empty space.

Remark: **distance measures** break down in higher dimensionalities (the central limit theorem kicks in)

Consequence: a **search policy** that is valuable in small dimensions **might be useless** in moderate or large dimensional search spaces.
Example: exhaustive search.

Separable Problems

Definition (Separable Problem)

A function f is separable if

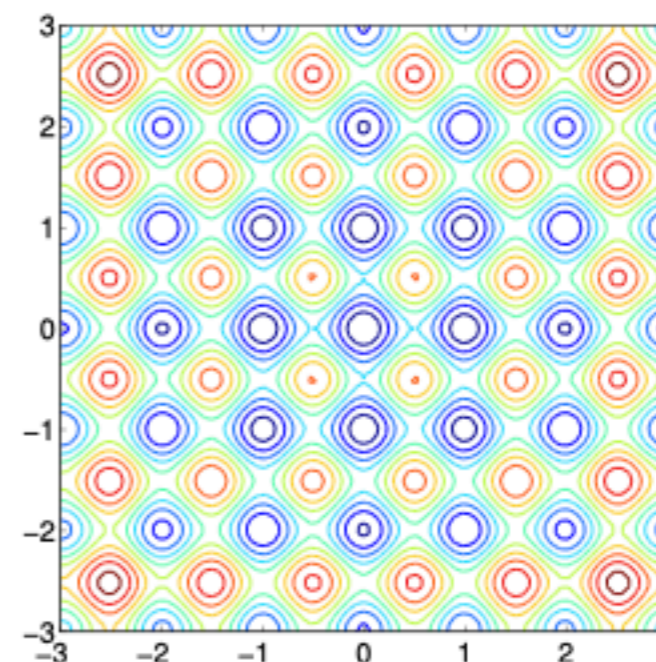
$$\arg \min_{(x_1, \dots, x_n)} f(x_1, \dots, x_n) = \left(\arg \min_{x_1} f(x_1, \dots), \dots, \arg \min_{x_n} f(\dots, x_n) \right)$$

\Rightarrow it follows that f can be optimized in a sequence of n independent 1-D optimization processes

Example: Additively decomposable functions

$$f(x_1, \dots, x_n) = \sum_{i=1}^n f_i(x_i)$$

Rastrigin function



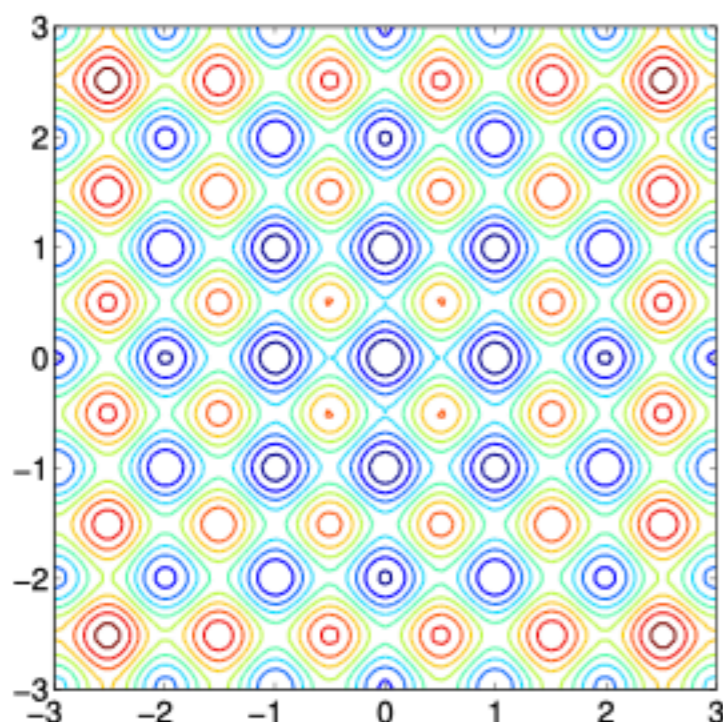
Non-Separable Problems

Building a non-separable problem from a separable one ^(1,2)

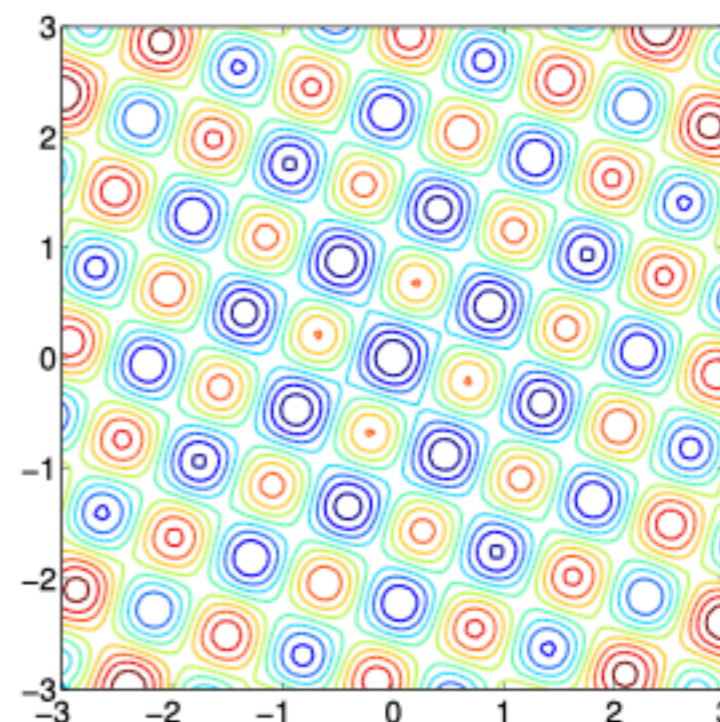
Rotating the coordinate system

- $f : \mathbf{x} \mapsto f(\mathbf{x})$ separable
- $f : \mathbf{x} \mapsto f(\mathbf{R}\mathbf{x})$ non-separable

R rotation matrix



R
→



¹ Hansen, Ostermeier, Gawelczyk (1995). On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. Sixth ICGA, pp. 57-64, Morgan Kaufmann

² Salomon (1996). "Reevaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions; A survey of some theoretical and practical aspects of genetic algorithms." BioSystems, 39(3):263-278

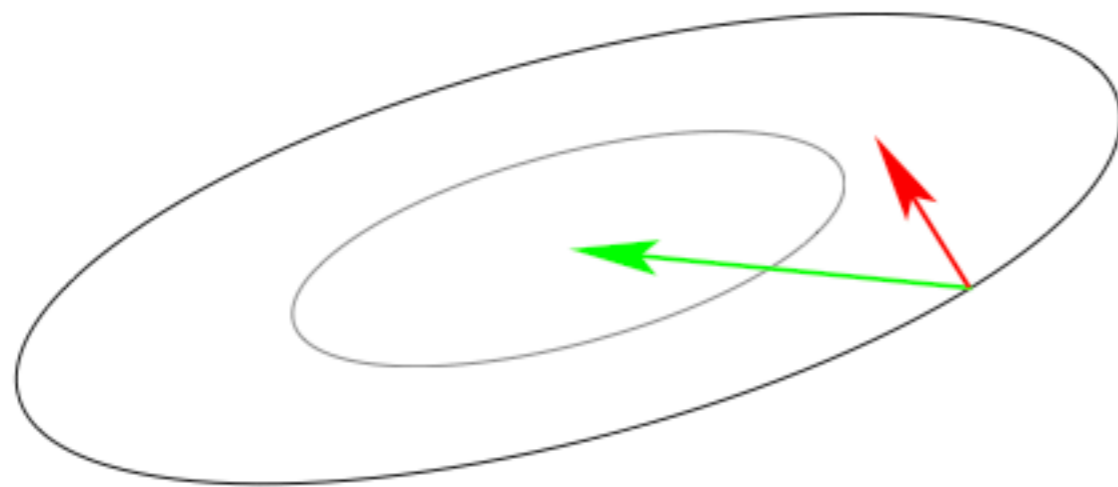
III-Conditioned Problems

Curvature of level sets

Consider the convex-quadratic function

$$f(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T \mathbf{H}(\mathbf{x} - \mathbf{x}^*) = \frac{1}{2} \sum_i h_{i,i} (x_i - x_i^*)^2 + \frac{1}{2} \sum_{i \neq j} h_{i,j} (x_i - x_i^*)(x_j - x_j^*)$$

\mathbf{H} is Hessian matrix of f and symmetric positive definite



gradient direction $-f'(\mathbf{x})^T$

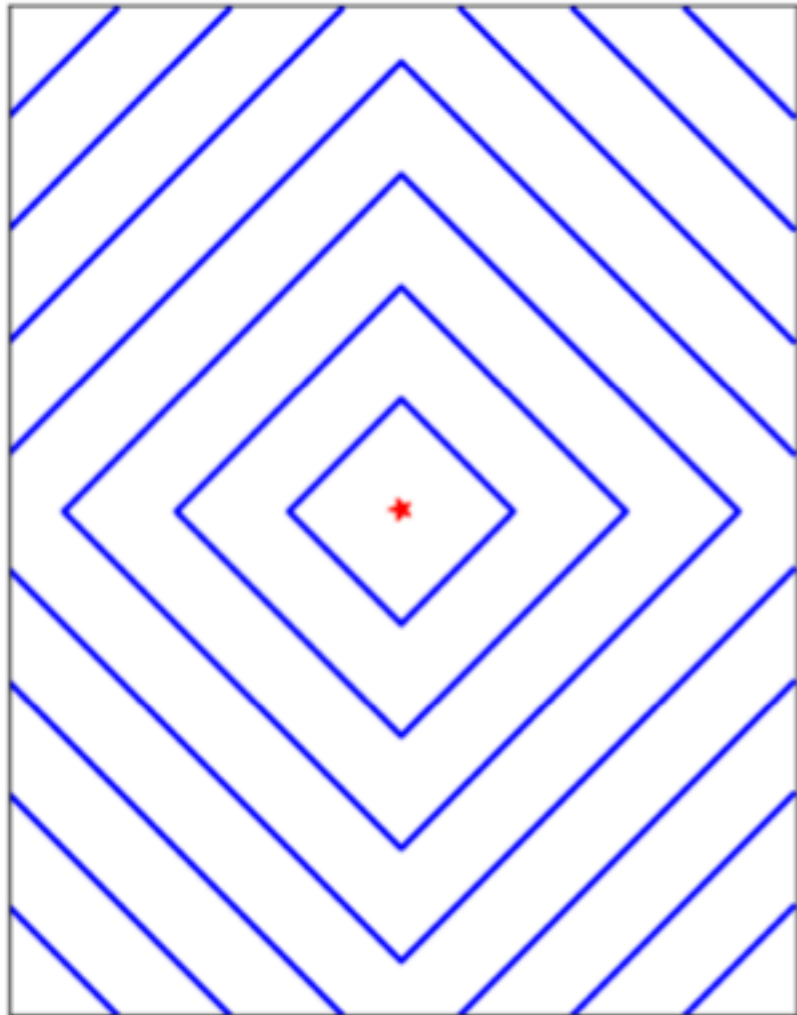
Newton direction $-\mathbf{H}^{-1}f'(\mathbf{x})^T$

III-conditioning means **squeezed level sets** (high curvature).
Condition number equals nine here. Condition numbers up to 10^{10}
are not unusual in real world problems.

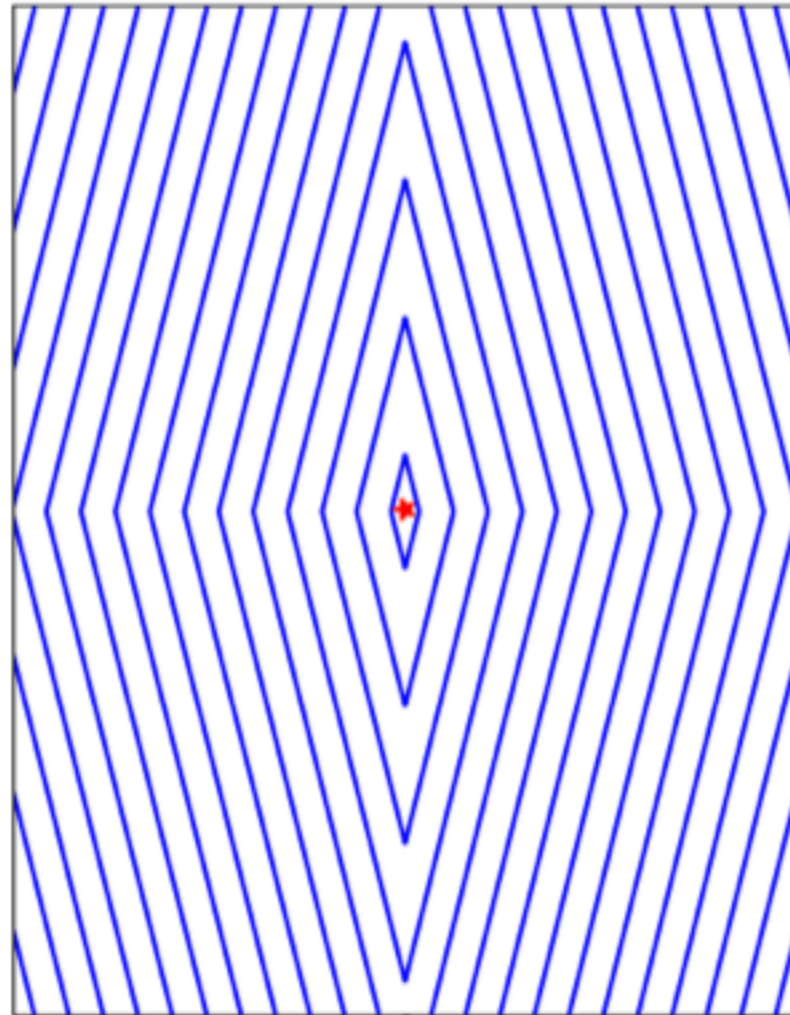
If $\mathbf{H} \approx \mathbf{I}$ (small condition number of \mathbf{H}) first order information (e.g. the gradient) is sufficient. Otherwise **second order information** (estimation of \mathbf{H}^{-1}) **is necessary**.

Non-smooth level sets (sharp ridges)

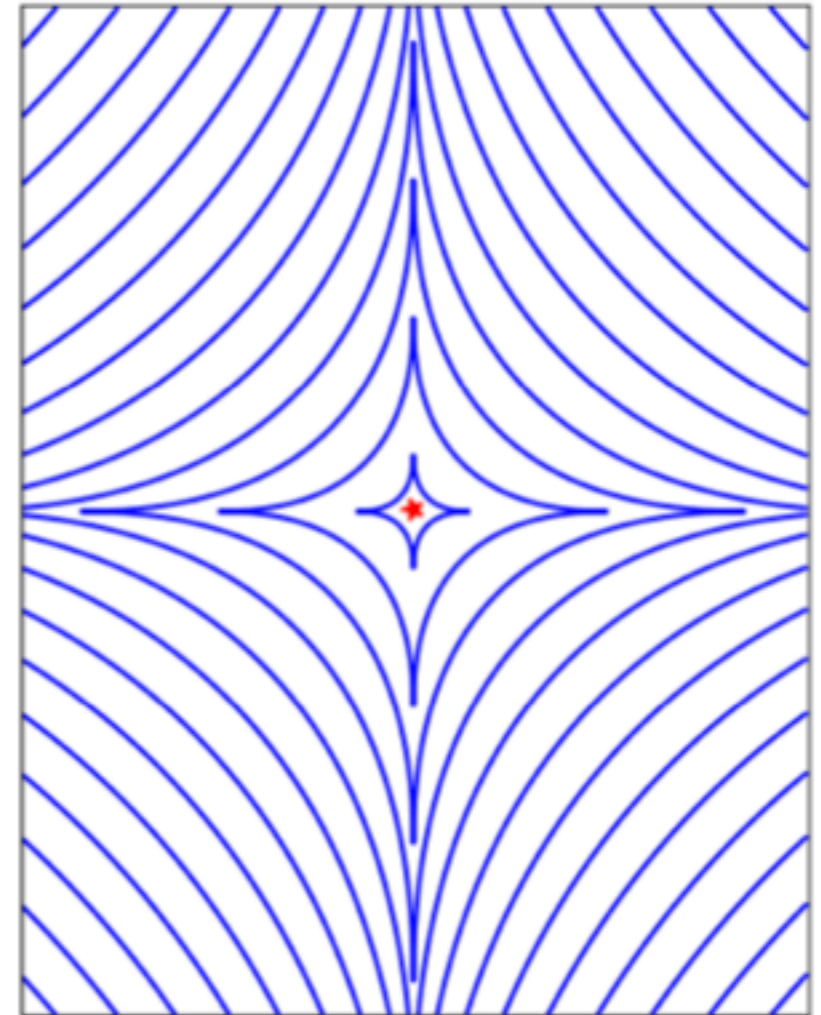
Similar difficulty **but worse** than ill-conditioning



1-norm



scaled 1-norm



1/2-norm

opening angle is the crucial parameter

What Makes a Function Difficult to Solve?

... and what can be done

The Problem	Possible Approaches
Dimensionality	exploiting the problem structure separability, locality/neighborhood, encoding
Ill-conditioning	second order approach changes the neighborhood metric
Ruggedness and non-smooth level sets	<p>non-local policy, large sampling width (step-size) as large as possible while preserving a reasonable convergence speed</p> <p>population-based method, stochastic, non-elitistic</p> <p>recombination operator serves as repair mechanism</p> <p>restarts</p>

... metaphors

Topics

1. What makes the problem difficult to solve?

2. How does the CMA-ES work?

- Normal Distribution, Rank-Based Recombination
- Step-Size Adaptation
- Covariance Matrix Adaptation

3. What can/should the users do for the CMA-ES to work effectively on their problem?

- Choice of problem formulation and encoding (not covered)
- Choice of initial solution and initial step-size
- Restarts, Increasing Population Size
- Restricted Covariance Matrix

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via **operators on a population**, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via **operators on a population**, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via **operators on a population**, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via **operators on a population**, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via **operators on a population**, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via **operators on a population**, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via **operators on a population**, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via **operators on a population**, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms* ↻

The CMA-ES

Input: $\mathbf{m} \in \mathbb{R}^n$; $\sigma \in \mathbb{R}_+$; $\lambda \in \mathbb{N}_{\geq 2}$, usually $\lambda \geq 5$, default $4 + \lfloor 3 \log n \rfloor$

Set $c_m = 1$; $c_1 \approx 2/n^2$; $c_\mu \approx \mu_w/n^2$; $c_c \approx 4/n$; $c_\sigma \approx 1/\sqrt{n}$; $d_\sigma \approx 1$; $w_{i=1\dots\lambda}$ decreasing in i and $\sum_{i=1}^\mu w_i = 1$, $w_\mu > 0 \geq w_{\mu+1}$, $\mu_w^{-1} := \sum_{i=1}^\mu w_i^2 \approx 3/\lambda$

Initialize $\mathbf{C} = \mathbf{I}$, and $\mathbf{p}_c = \mathbf{0}$, $\mathbf{p}_\sigma = \mathbf{0}$

While not terminate

$\mathbf{x}_i = \mathbf{m} + \sigma \mathbf{y}_i$, where $\mathbf{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$ for $i = 1, \dots, \lambda$ sampling

$\mathbf{m} \leftarrow \mathbf{m} + c_m \sigma \mathbf{y}_w$, where $\mathbf{y}_w = \sum_{i=1}^\mu w_{\text{rk}(i)} \mathbf{y}_i$ update mean

$\mathbf{p}_\sigma \leftarrow (1 - c_\sigma) \mathbf{p}_\sigma + \sqrt{1 - (1 - c_\sigma)^2} \sqrt{\mu_w} \mathbf{C}^{-\frac{1}{2}} \mathbf{y}_w$ path for σ

$\mathbf{p}_c \leftarrow (1 - c_c) \mathbf{p}_c + \mathbf{1}_{[0, 2n]} \left\{ \|\mathbf{p}_\sigma\|^2 \right\} \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w} \mathbf{y}_w$ path for \mathbf{C}

$\sigma \leftarrow \sigma \times \exp \left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|\mathbf{p}_\sigma\|}{\mathbb{E} \|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1 \right) \right)$ update of σ

$\mathbf{C} \leftarrow \mathbf{C} + c_\mu \sum_{i=1}^\lambda w_{\text{rk}(i)} (\mathbf{y}_i \mathbf{y}_i^\top - \mathbf{C}) + c_1 (\mathbf{p}_c \mathbf{p}_c^\top - \mathbf{C})$ update \mathbf{C}

Not covered: termination, restarts, useful output, search boundaries and encoding, corrections for: positive definiteness guaranty, \mathbf{p}_c variance loss, c_σ and d_σ for large λ

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via **operators on a population**, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Evolution Strategies

New search points are sampled normally distributed

$$\mathbf{x}_i \sim \mathbf{m} + \sigma \mathcal{N}_i(\mathbf{0}, \mathbf{C}) \quad \text{for } i = 1, \dots, \lambda$$

as perturbations of \mathbf{m} , where $\mathbf{x}_i, \mathbf{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\mathbf{C} \in \mathbb{R}^{n \times n}$

where

- the **mean** vector $\mathbf{m} \in \mathbb{R}^n$ represents the favorite solution
- the so-called **step-size** $\sigma \in \mathbb{R}_+$ controls the *step length*
- the **covariance matrix** $\mathbf{C} \in \mathbb{R}^{n \times n}$ determines the **shape** of the distribution ellipsoid

here, all new points are sampled with the same parameters

The question remains how to update \mathbf{m} , \mathbf{C} , and σ .



Why Normal Distributions?

① widely observed in nature, for example as phenotypic traits

② only stable distribution with finite variance

stable means that the sum of normal variates is again normal:

$$\mathcal{N}(\mathbf{x}, \mathbf{A}) + \mathcal{N}(\mathbf{y}, \mathbf{B}) \sim \mathcal{N}(\mathbf{x} + \mathbf{y}, \mathbf{A} + \mathbf{B})$$

helpful in **design and analysis** of algorithms
related to the *central limit theorem*

③ most convenient way to generate **isotropic** search points

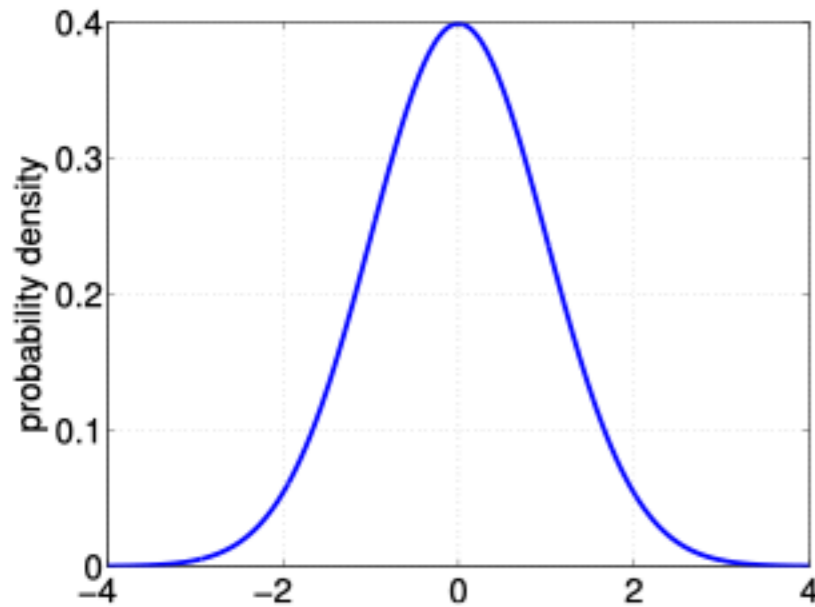
the isotropic distribution does **not favor any direction**, rotational invariant

④ maximum entropy distribution with finite variance

the least possible assumptions on f in the distribution shape

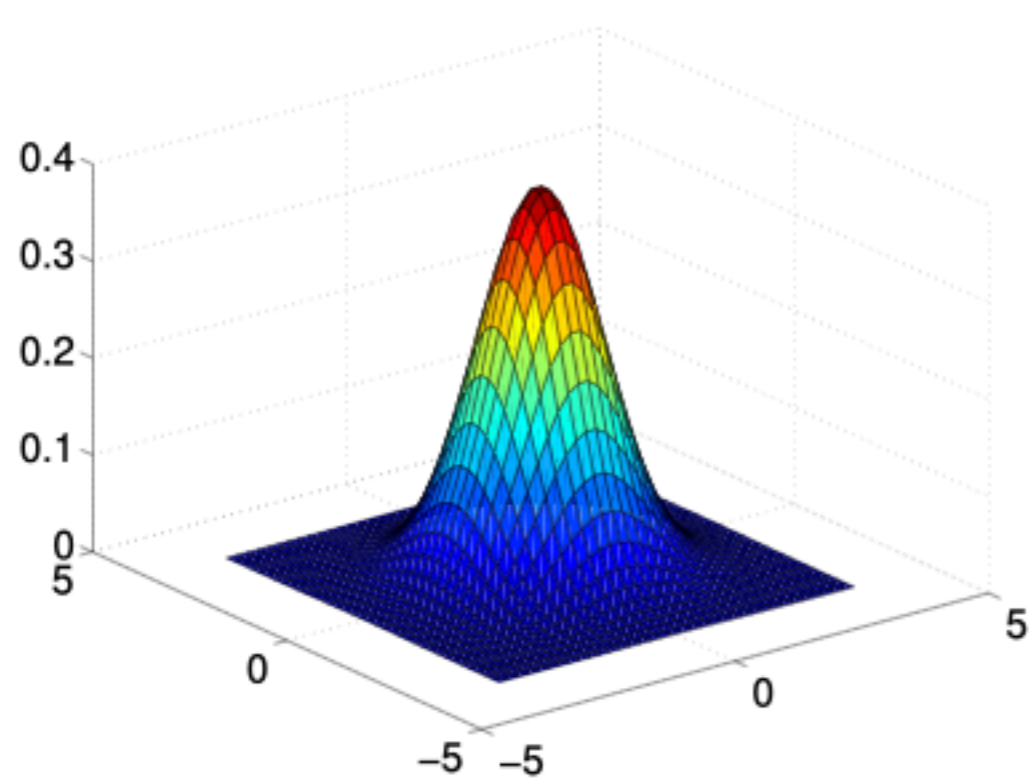
Normal Distribution

Standard Normal Distribution

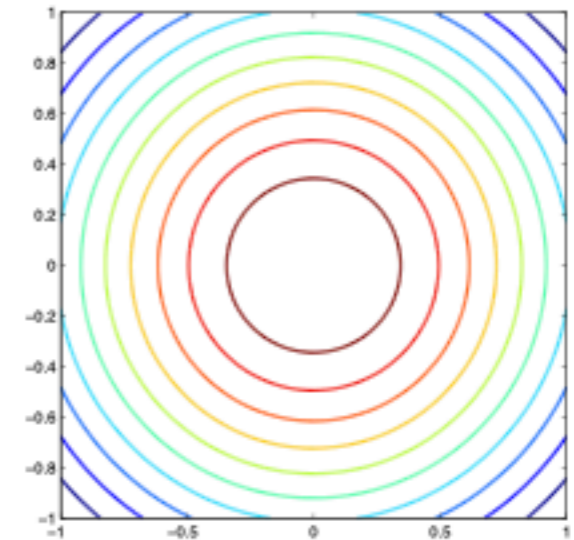


probability density of the 1-D standard normal distribution

2-D Normal Distribution



probability density of a 2-D normal distribution

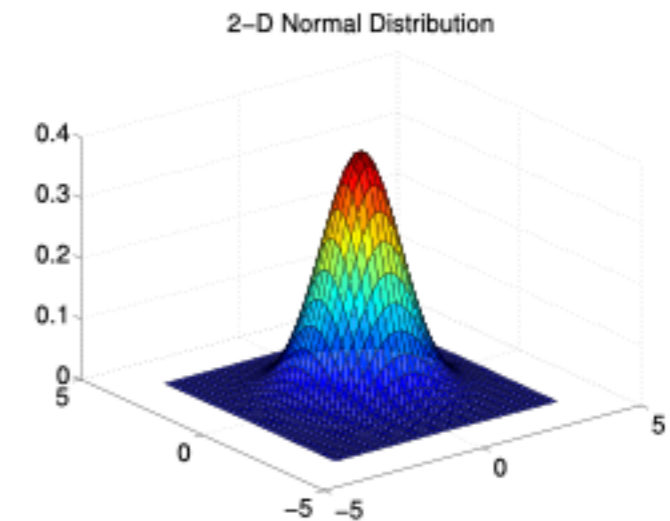


The Multi-Variate (n -Dimensional) Normal Distribution

Any multi-variate normal distribution $\mathcal{N}(\mathbf{m}, \mathbf{C})$ is uniquely determined by its mean value $\mathbf{m} \in \mathbb{R}^n$ and its symmetric positive definite $n \times n$ covariance matrix \mathbf{C} .

The **mean** value \mathbf{m}

- determines the displacement (translation)
- value with the largest density (modal value)
- the distribution is symmetric about the distribution mean

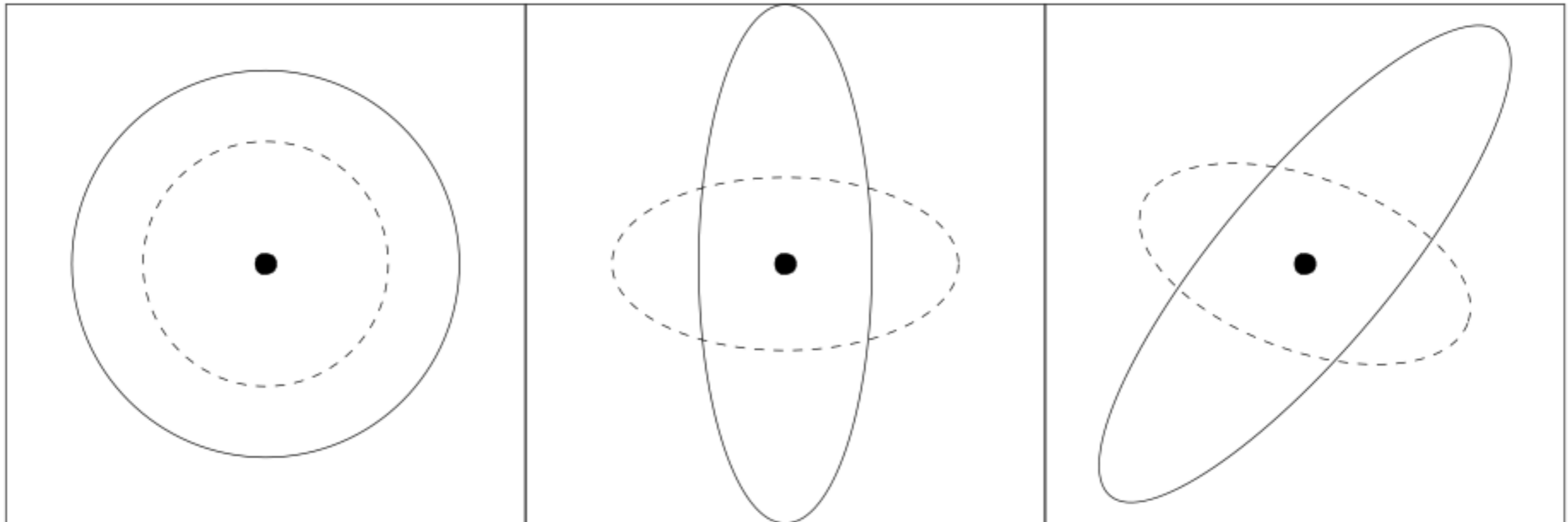


The **covariance matrix** \mathbf{C}

- determines the shape
- **geometrical interpretation**: any covariance matrix can be uniquely identified with the iso-density ellipsoid $\{x \in \mathbb{R}^n \mid (x - \mathbf{m})^T \mathbf{C}^{-1} (x - \mathbf{m}) = n\}$

... any **covariance matrix** can be uniquely identified with the iso-density ellipsoid
 $\{\mathbf{x} \in \mathbb{R}^n \mid (\mathbf{x} - \mathbf{m})^T \mathbf{C}^{-1} (\mathbf{x} - \mathbf{m}) = n\}$

Lines of Equal Density



$\mathcal{N}(\mathbf{m}, \sigma^2 \mathbf{I}) \sim \mathbf{m} + \sigma \mathcal{N}(\mathbf{0}, \mathbf{I})$
one degree of freedom σ
 components are
 independent standard
 normally distributed

$\mathcal{N}(\mathbf{m}, \mathbf{D}^2) \sim \mathbf{m} + \mathbf{D} \mathcal{N}(\mathbf{0}, \mathbf{I})$
 n degrees of freedom
 components are
 independent, scaled

$\mathcal{N}(\mathbf{m}, \mathbf{C}) \sim \mathbf{m} + \mathbf{C}^{\frac{1}{2}} \mathcal{N}(\mathbf{0}, \mathbf{I})$
 $(n^2 + n)/2$ degrees of freedom
 components are
 correlated

where \mathbf{I} is the identity matrix (isotropic case) and \mathbf{D} is a diagonal matrix (reasonable for separable problems) and $\mathbf{A} \times \mathcal{N}(\mathbf{0}, \mathbf{I}) \sim \mathcal{N}(\mathbf{0}, \mathbf{A}\mathbf{A}^T)$ holds for all \mathbf{A} .

Multivariate Normal Distribution and Eigenvalues

For any positive definite symmetric \mathbf{C} ,

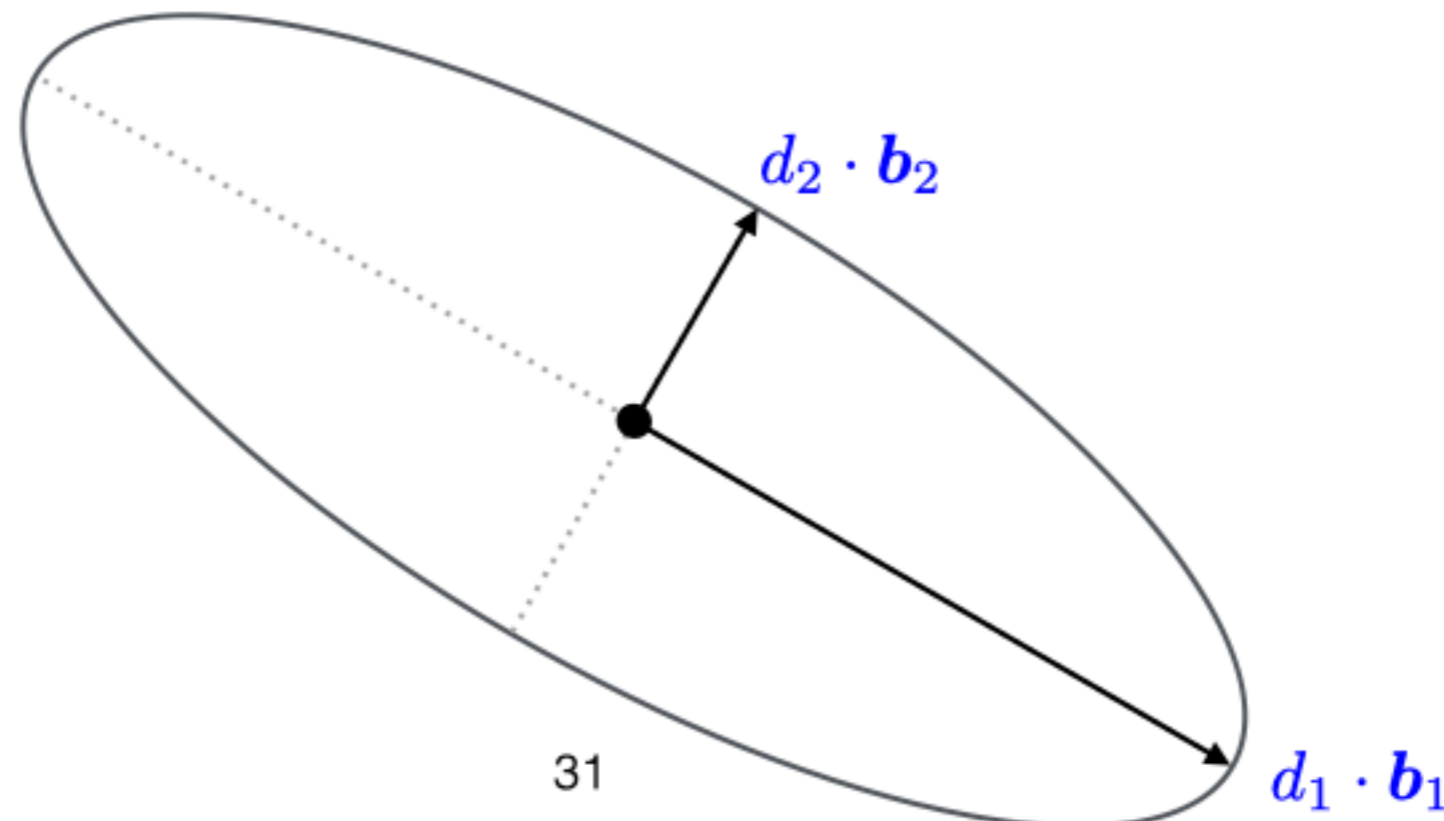
$$\mathbf{C} = d_1^2 \mathbf{b}_1 \mathbf{b}_1^T + \dots + d_N^2 \mathbf{b}_N \mathbf{b}_N^T$$

d_i : square root of the eigenvalue of \mathbf{C}

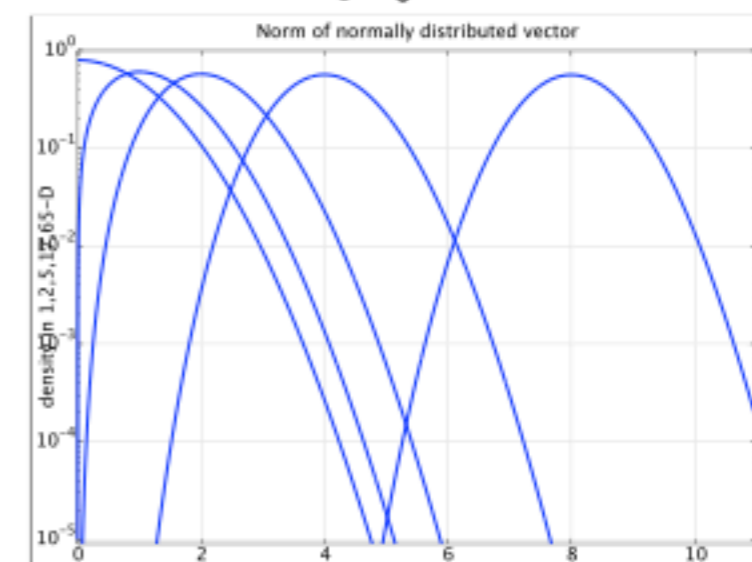
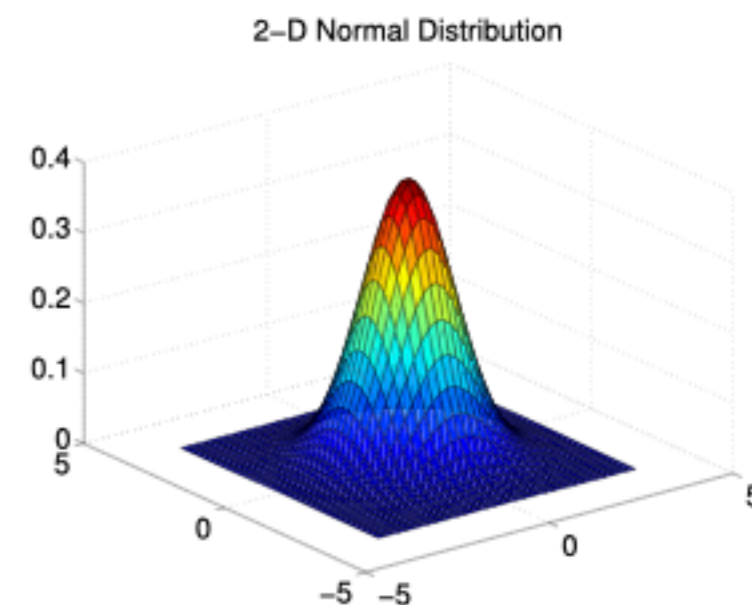
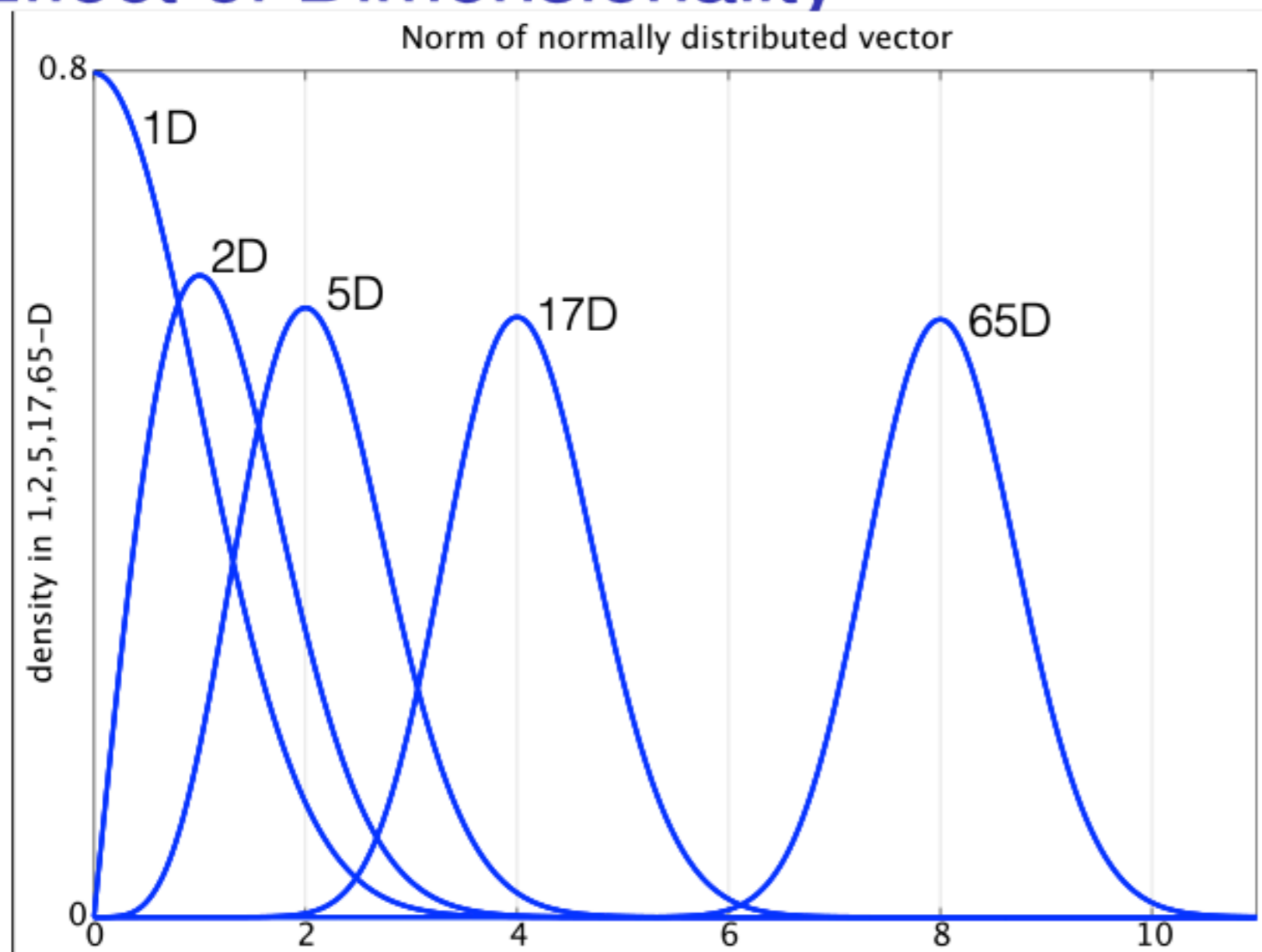
\mathbf{b}_i : eigenvector of \mathbf{C} , corresponding to d_i

The multivariate normal distribution $\mathcal{N}(\mathbf{m}, \mathbf{C})$

$$\mathcal{N}(\mathbf{m}, \mathbf{C}) \sim \mathbf{m} + \mathcal{N}(0, d_1^2) \mathbf{b}_1 + \dots + \mathcal{N}(0, d_N^2) \mathbf{b}_N$$



Effect of Dimensionality



$\|\mathcal{N}(\mathbf{0}, \mathbf{I})\| \longrightarrow \mathcal{N}\left(\sqrt{n-1/2}, 1/2\right)$ with modal value $\sqrt{n-1}$

yet: maximum entropy distribution

also consider a difference between two vectors:

$$\|\mathcal{N}(\mathbf{0}, \mathbf{I}) - \mathcal{N}(\mathbf{0}, \mathbf{I})\| \sim \|\mathcal{N}(\mathbf{0}, \mathbf{I}) + \mathcal{N}(\mathbf{0}, \mathbf{I})\| \sim \sqrt{2}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$$

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via **operators on a population**, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

The $(\mu/\mu, \lambda)$ -ES, Update of the Distribution Mean

Non-elitist selection and intermediate (weighted) recombination

Given the i -th solution point $\mathbf{x}_i = \mathbf{m} + \sigma \underbrace{\mathcal{N}_i(\mathbf{0}, \mathbf{C})}_{=: \mathbf{y}_i} = \mathbf{m} + \sigma \mathbf{y}_i$

Let $\mathbf{x}_{i:\lambda}$ the i -th ranked solution point, such that $f(\mathbf{x}_{1:\lambda}) \leq \dots \leq f(\mathbf{x}_{\lambda:\lambda})$.

The new mean reads

$$\mathbf{m} \leftarrow \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda} = \mathbf{m} + \sigma \underbrace{\sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}}_{=: \mathbf{y}_w}$$

where

$$w_1 \geq \dots \geq w_{\mu} > 0, \quad \sum_{i=1}^{\mu} w_i = 1, \quad \frac{1}{\sum_{i=1}^{\mu} w_i^2} =: \mu_w \approx \frac{\lambda}{4}$$

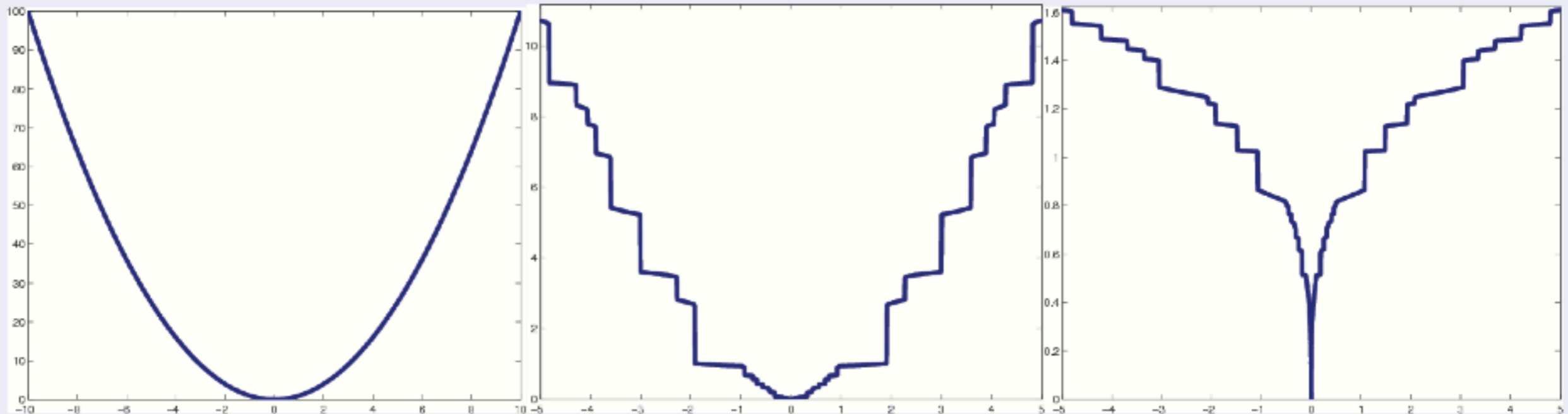
The best μ points are selected from the new solutions (non-elitistic) and weighted intermediate recombination is applied.

Invariance Under Monotonically Increasing Functions

Rank-based algorithms

Update of all parameters uses only the ranks

$$f(x_{1:\lambda}) \leq f(x_{2:\lambda}) \leq \dots \leq f(x_{\lambda:\lambda})$$



$$g(f(x_{1:\lambda})) \leq g(f(x_{2:\lambda})) \leq \dots \leq g(f(x_{\lambda:\lambda})) \quad \forall g$$

g is strictly monotonically increasing
 g preserves ranks

3

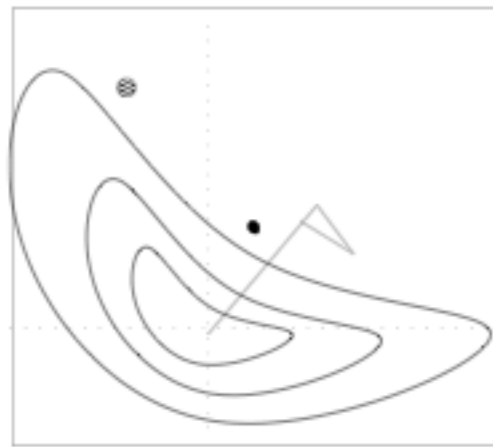
³ Whitley 1989. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best,

ICGA

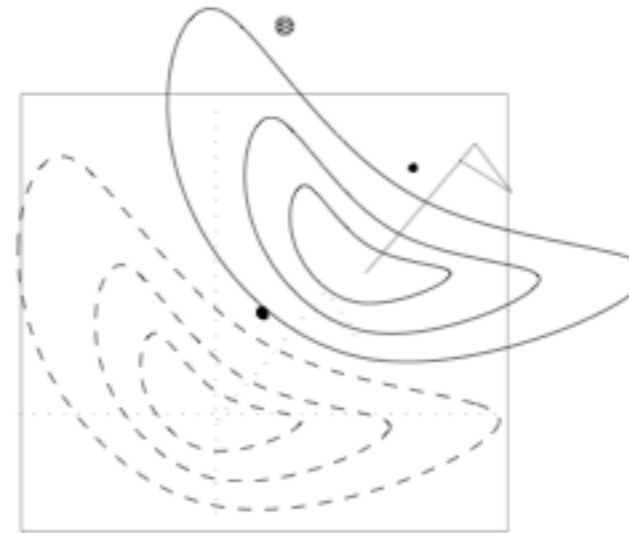
Basic Invariance in Search Space

- translation invariance

is true for most optimization algorithms



$$f(\mathbf{x}) \leftrightarrow f(\mathbf{x} - \mathbf{a})$$

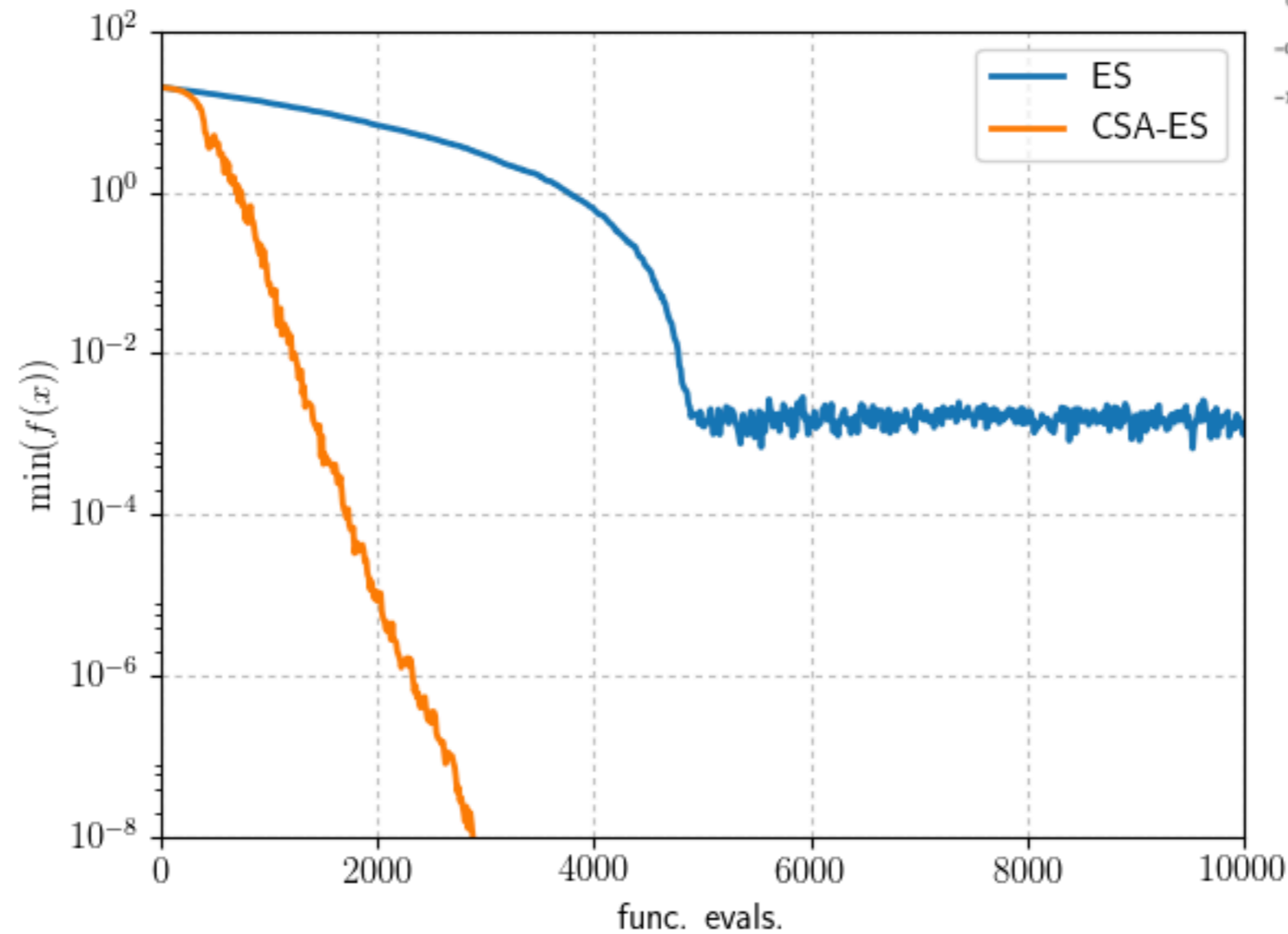


Identical behavior on f and f_a

$$\begin{aligned} f &: \mathbf{x} \mapsto f(\mathbf{x}), & \mathbf{x}^{(t=0)} &= \mathbf{x}_0 \\ f_a &: \mathbf{x} \mapsto f(\mathbf{x} - \mathbf{a}), & \mathbf{x}^{(t=0)} &= \mathbf{x}_0 + \mathbf{a} \end{aligned}$$

No difference can be observed w.r.t. the argument of f

Summary



On 20D Sphere Function: $f(\mathbf{x}) = \sum_{i=1}^N [\mathbf{x}]_i^2$

- ES without adaptation can't approach the optimum \Rightarrow adaptation required

Evolution Strategies

Recalling

New search points are sampled normally distributed

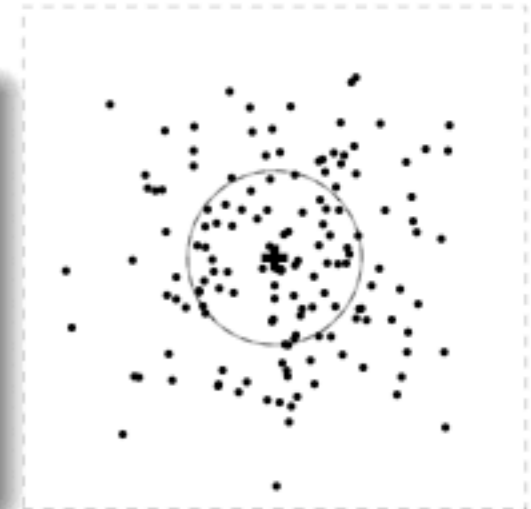
$$\mathbf{x}_i \sim \mathbf{m} + \sigma \mathcal{N}_i(\mathbf{0}, \mathbf{C}) \quad \text{for } i = 1, \dots, \lambda$$

as perturbations of \mathbf{m} , where $\mathbf{x}_i, \mathbf{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\mathbf{C} \in \mathbb{R}^{n \times n}$

where

- the **mean** vector $\mathbf{m} \in \mathbb{R}^n$ represents the favorite solution and $\mathbf{m} \leftarrow \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}$
- the so-called **step-size** $\sigma \in \mathbb{R}_+$ controls the *step length*
- the **covariance matrix** $\mathbf{C} \in \mathbb{R}^{n \times n}$ determines the **shape** of the distribution ellipsoid

The remaining question is how to update σ and \mathbf{C} .



Methods for Step-Size Control

- **1/5-th success rule^{ab}**, often applied with “+”-selection
 - increase step-size if more than 20% of the new solutions are successful, decrease otherwise
- **σ -self-adaptation^c**, applied with “,”-selection
 - mutation is applied to the step-size and the better, according to the objective function value, is selected
 - simplified “global” self-adaptation
- **path length control^d** (Cumulative Step-size Adaptation, CSA)^e
 - self-adaptation derandomized and non-localized
- **two-point adaptation^f** (TPA)
 - akin to a rudimentary line search along the mean shift

^aRechenberg 1973, *Evolutionsstrategie, Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog

^bSchumer and Steiglitz 1968. Adaptive step size random search. *IEEE TAC*

^cSchwefel 1981, *Numerical Optimization of Computer Models*, Wiley

^dHansen & Ostermeier 2001, Completely Derandomized Self-Adaptation in Evolution Strategies, *Evol. Comput.* 9 (2)

^eOstermeier *et al* 1994, Step-size adaptation based on non-local use of selection information, *PPSN IV*

^fHansen *et al* 2014, How to assess step-size adaptation mechanisms in a derandomised search, *PPSN IX*

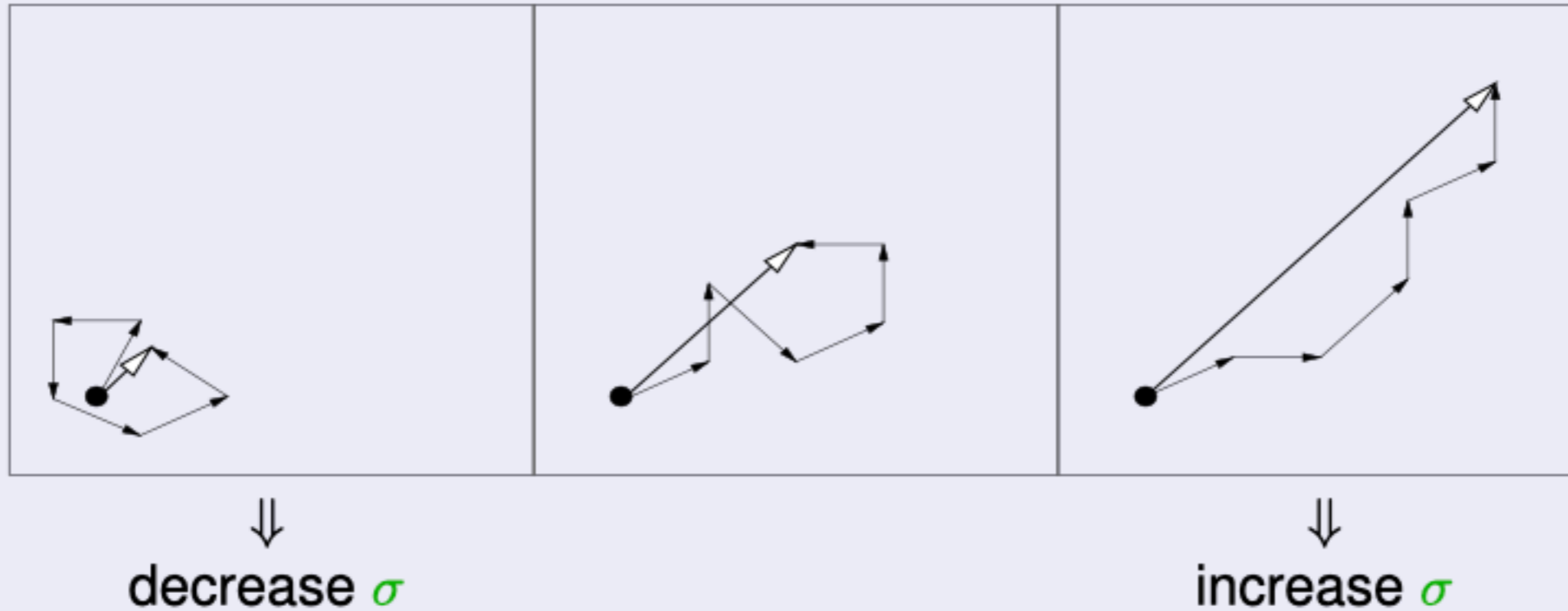
Path Length Control (CSA)

The Concept of Cumulative Step-Size Adaptation

$$\begin{aligned} \mathbf{x}_i &= \mathbf{m} + \sigma \mathbf{y}_i \\ \mathbf{m} &\leftarrow \mathbf{m} + \sigma \mathbf{y}_w \end{aligned}$$

Measure the length of the *evolution path*

the pathway of the mean vector \mathbf{m} in the generation sequence



loosely speaking steps are

- perpendicular under random selection (in expectation)
- perpendicular in the desired situation (to be most efficient)

Path Length Control (CSA)

The Equations

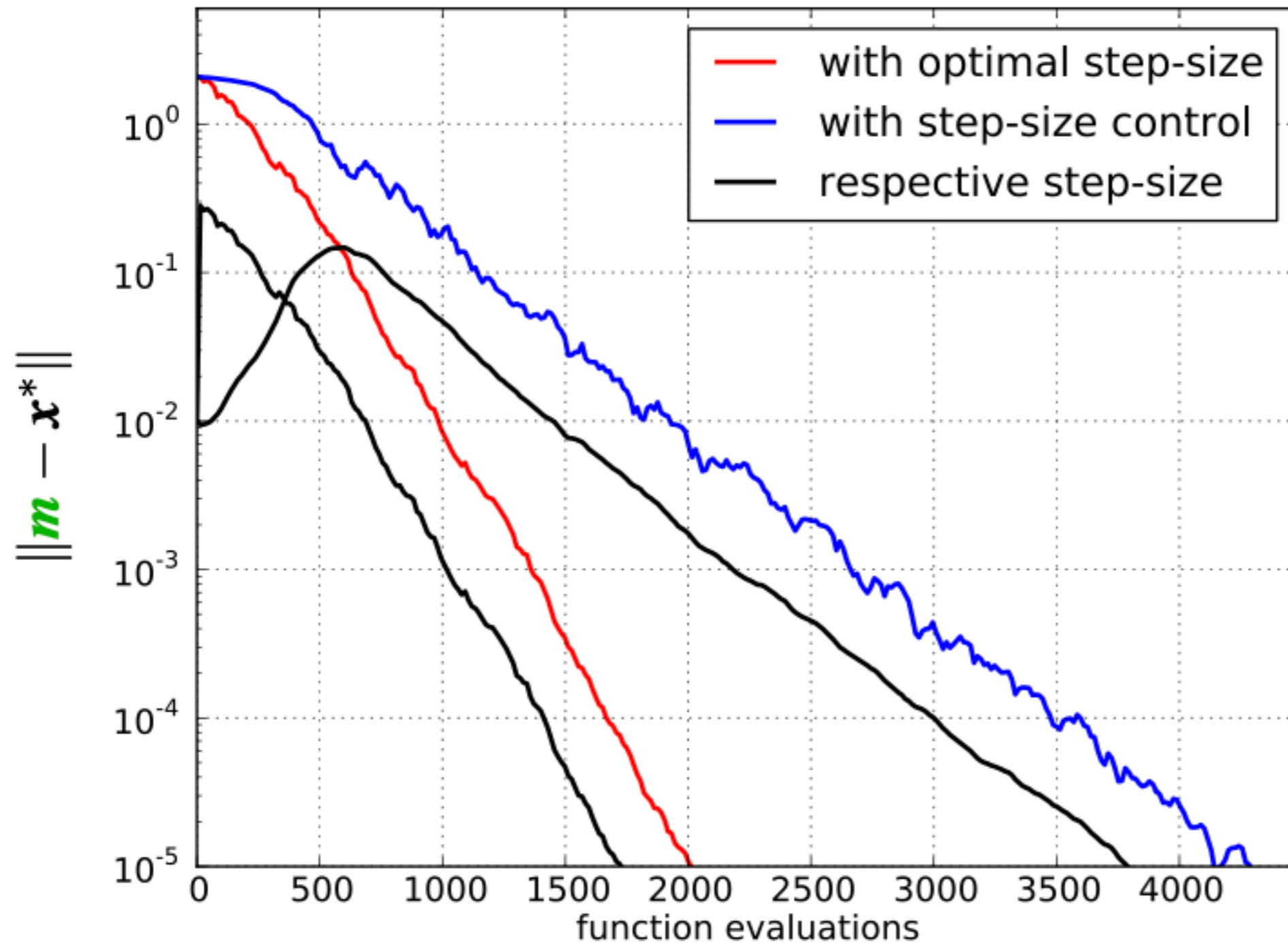
Initialize $\mathbf{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, evolution path $\mathbf{p}_\sigma = \mathbf{0}$,
set $c_\sigma \approx 4/n$, $d_\sigma \approx 1$.

$$\mathbf{m} \leftarrow \mathbf{m} + \sigma \mathbf{y}_w \quad \text{where } \mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda} \quad \text{update mean}$$

$$\mathbf{p}_\sigma \leftarrow (1 - c_\sigma) \mathbf{p}_\sigma + \underbrace{\sqrt{1 - (1 - c_\sigma)^2}}_{\text{accounts for } 1 - c_\sigma} \underbrace{\sqrt{\mu_w}}_{\text{accounts for } w_i} \mathbf{y}_w$$

$$\sigma \leftarrow \sigma \times \underbrace{\exp \left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|\mathbf{p}_\sigma\|}{\mathbb{E} \|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1 \right) \right)}_{>1 \iff \|\mathbf{p}_\sigma\| \text{ is greater than its expectation}} \quad \text{update step-size}$$

(5/5, 10)-CSA-ES, default parameters



$$f(\mathbf{x}) = \sum_{i=1}^n x_i^2$$

in $[-0.2, 0.8]^n$
for $n = 30$

Two-Point Step-Size Adaptation (TPA)

- Sample a pair of symmetric points along the previous mean shift

$$\mathbf{x}_{1/2} = \mathbf{m}^{(g)} \pm \sigma^{(g)} \frac{\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|}{\|\mathbf{m}^{(g)} - \mathbf{m}^{(g-1)}\|_{\mathbf{C}^{(g)}}} (\mathbf{m}^{(g)} - \mathbf{m}^{(g-1)})$$

$$\|\mathbf{x}\|_{\mathbf{C}} := \mathbf{x}^T \mathbf{C}^{-1} \mathbf{x}$$

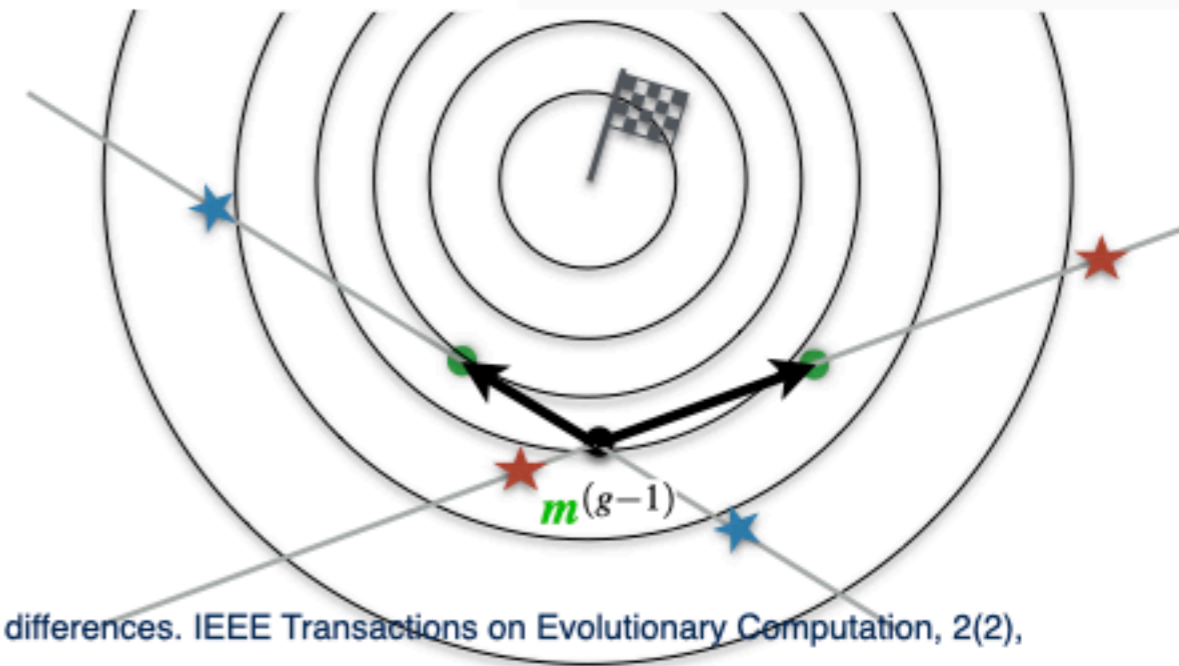
- Compare the ranking of \mathbf{x}_1 and \mathbf{x}_2 among λ current populations

$$s^{(g+1)} = (1 - c_s) s^{(g)} + c_s \underbrace{\frac{\text{rank}(\mathbf{x}_2) - \text{rank}(\mathbf{x}_1)}{\lambda - 1}}_{> 0 \text{ if forward direction ranks better}}$$

> 0 if forward direction ranks better

- Update the step-size

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{s^{(g+1)}}{d_\sigma}\right)$$



[Salomon, 1998] Salomon, R. (1998). Evolutionary algorithms and gradient search: similarities and differences. *IEEE Transactions on Evolutionary Computation*, 2(2), pages 45–55.

[Hansen, 2008] Hansen, N. (2008). CMA-ES with two-point step-size adaptation. [research report] rr-6527, 2008. Inria-00276854v5.

[Hansen et al, 2014] Hansen, N., Atamna, A., and Auger, A. (2014). How to assess step-size adaptation mechanisms in randomised search. In *Parallel Problem Solving from Nature—PPSN XIII*, pages 60–69. Springer.

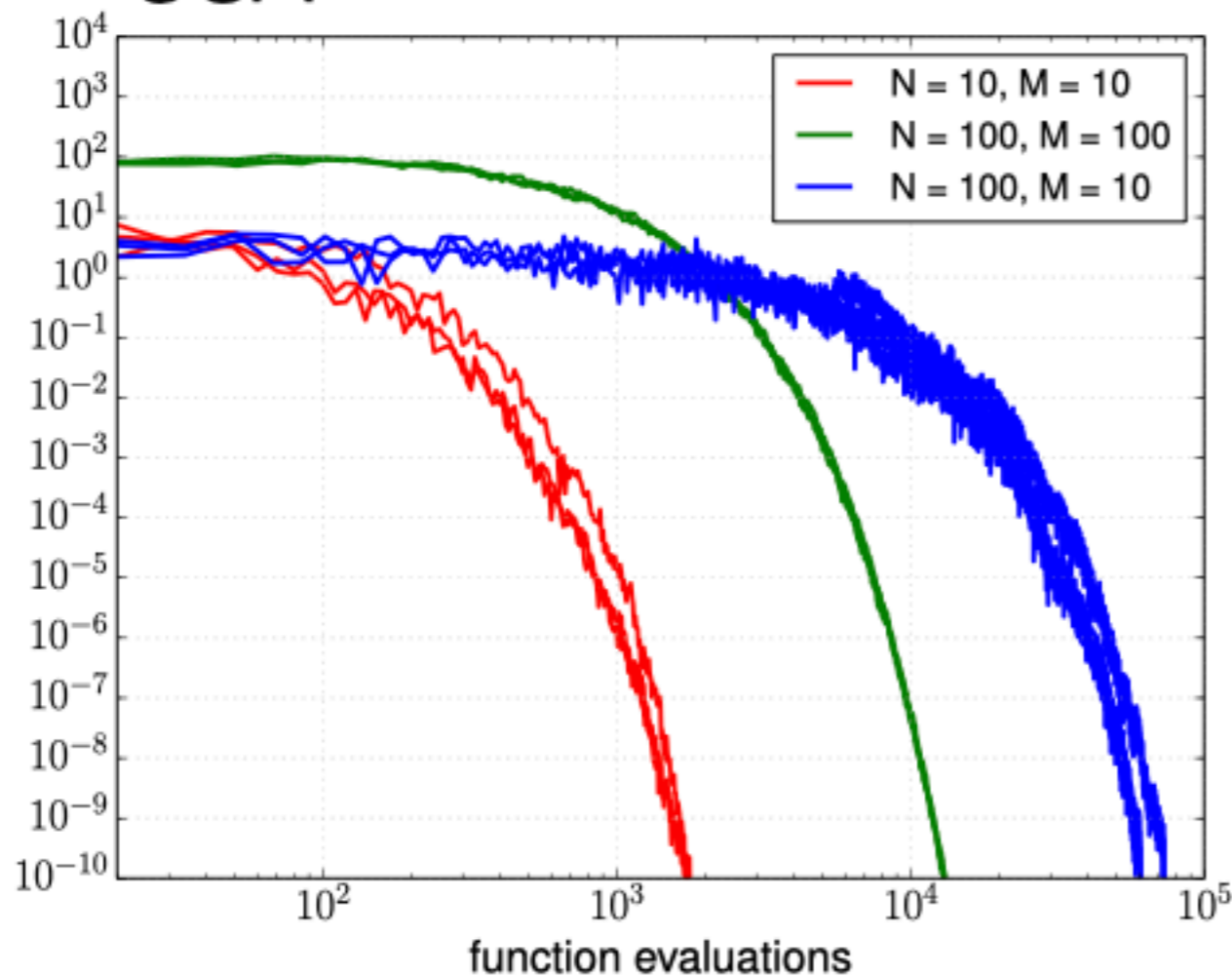
[Akimoto and Hansen, 2020] Akimoto, Y. and Hansen, N. (2020). Diagonal acceleration for covariance matrix adaptation evolution strategies. *Evolutionary computation*, 28(3), pages 405–435.

On Sphere with Low Effective Dimension

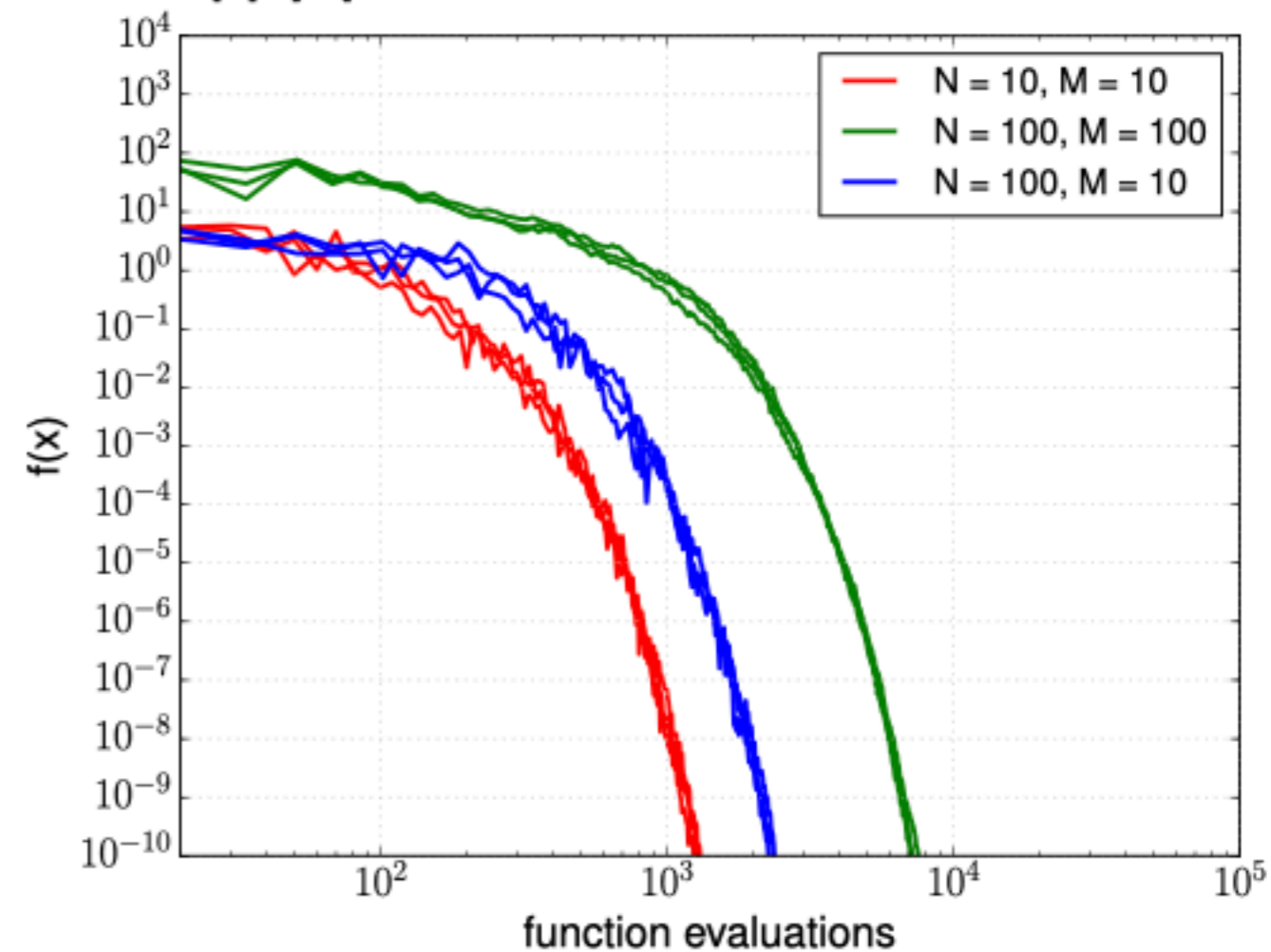
On a function with low effective dimension

- $f(\mathbf{x}) = \sum_{i=1}^M [\mathbf{x}]_i^2$, $\mathbf{x} \in \mathbb{R}^N$, $M \leq N$.
- $N - M$ variables do not affect the function value

CSA



TPA



Alternatives: Success-Based Step-Size Control

comparing the fitness distributions of current and previous iterations

Generalizations of 1/5th-success-rule for non-elitist and multi-recombinant ES

- **Median Success Rule** [Ait Elhara et al., 2013]
- **Population Success Rule** [Loshchilov, 2014]

controls a *success probability*

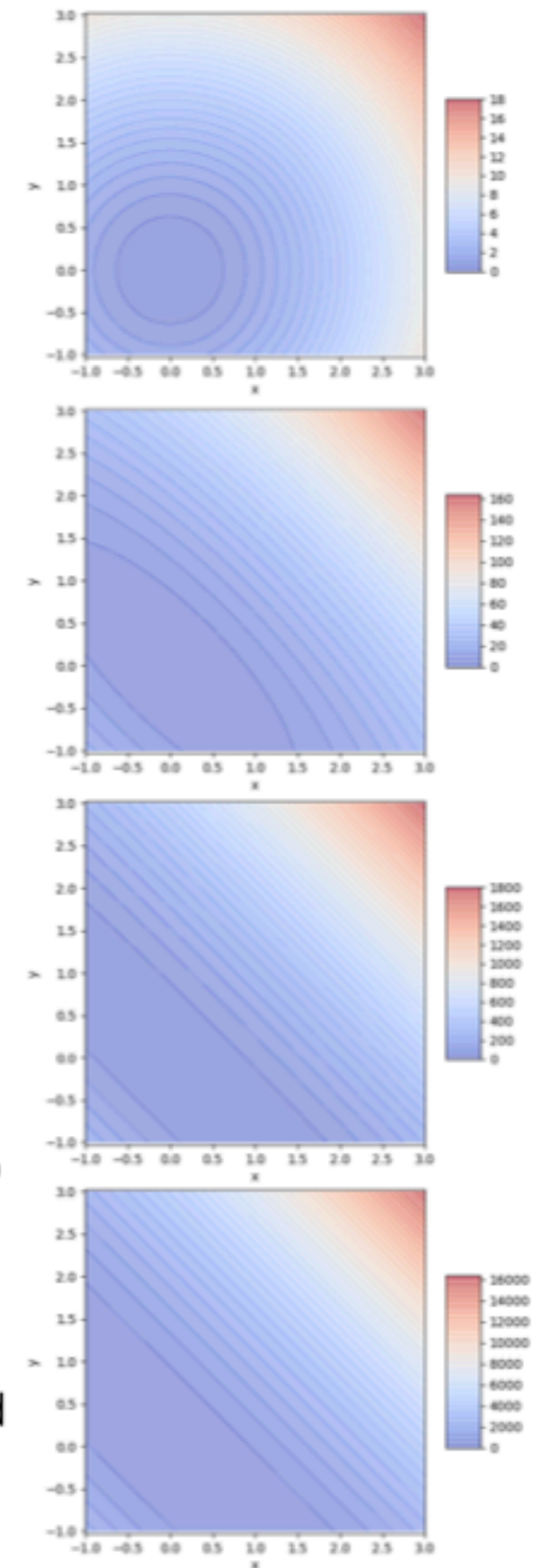
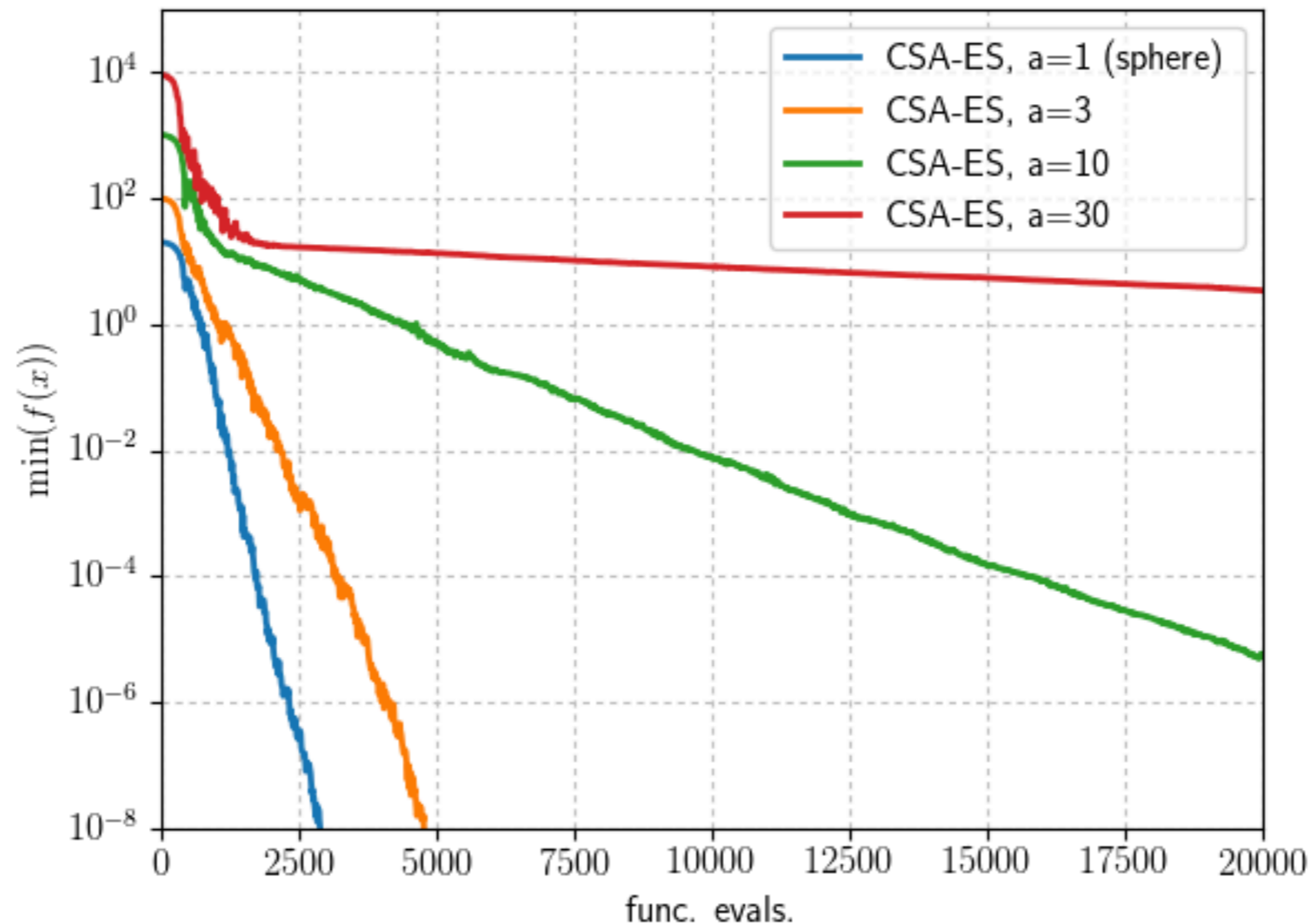
An advantage over CSA and TPA: Cheap Computation

- It depends only on λ .
- cf. CSA and TPA require a computation of $\mathbf{C}^{-1/2}\mathbf{x}$ and $\mathbf{C}^{-1}\mathbf{x}$, respectively.

[Ait Elhara et al., 2013] Ait Elhara, O., Auger, A., and Hansen, N. (2013). A median success rule for non-elitist evolution strategies: Study of feasibility. In Proc. of the GECCO, pages 415–422.

[Loshchilov, 2014] Loshchilov, I. (2014). A computationally efficient limited memory cma-es for large scale optimization. In Proc. of the GECCO, pages 397–404.

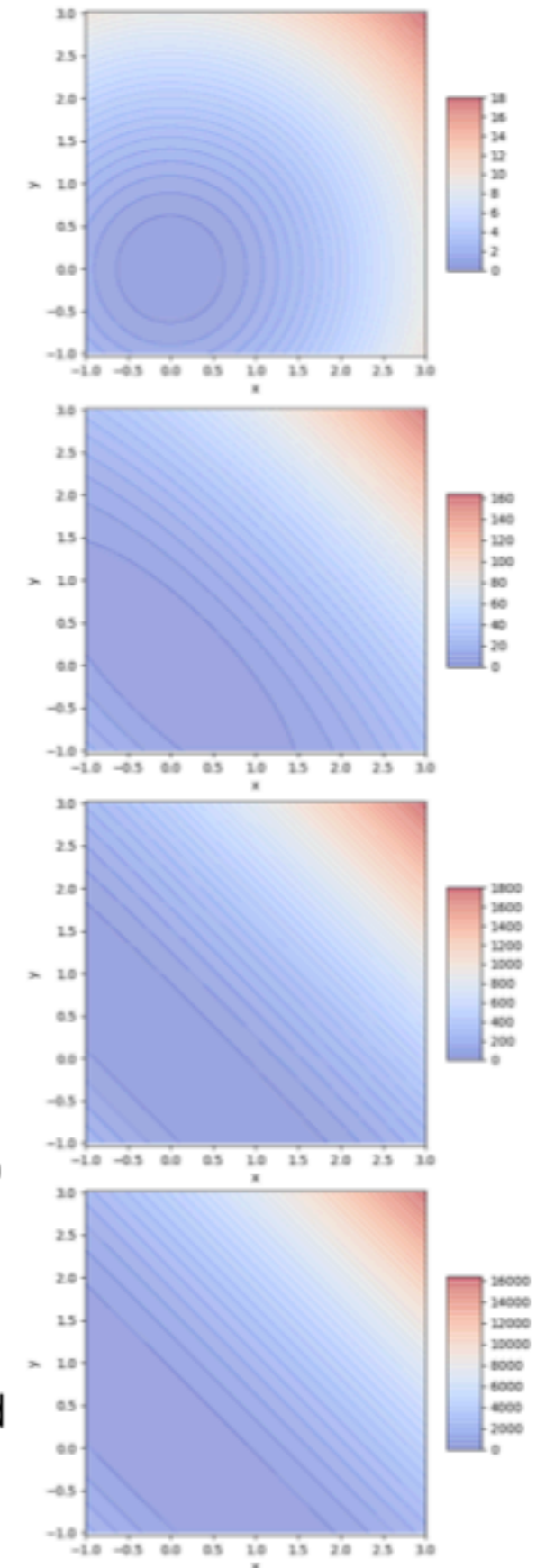
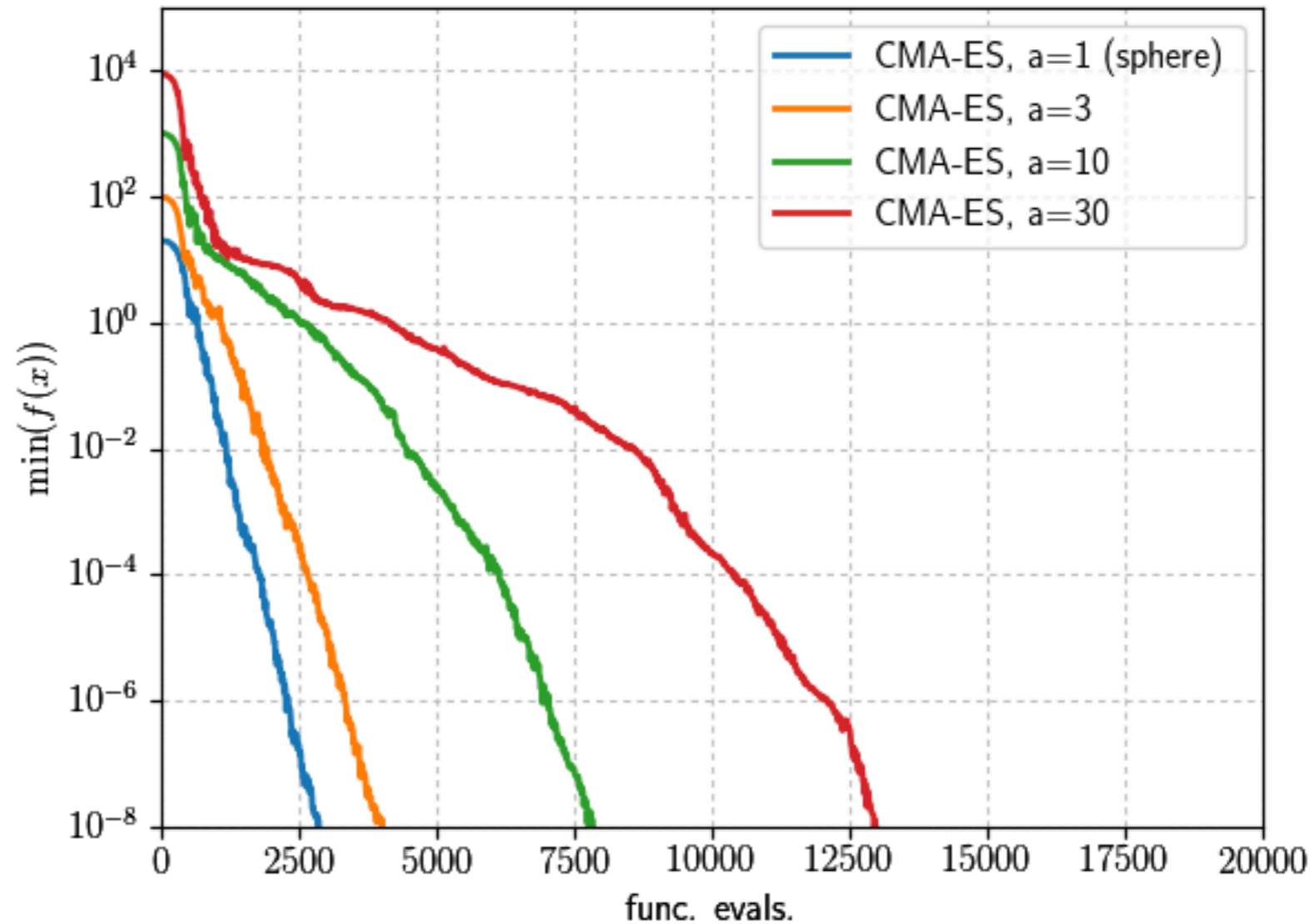
Step-Size Control Is Not Enough



On 20D TwoAxes Function: $f(\mathbf{x}) = \sum_{i=1}^{N/2} [\mathbf{R}\mathbf{x}]_i^2 + a^2 \sum_{i=N/2+1}^N [\mathbf{R}\mathbf{x}]_i^2$, \mathbf{R} : orthogonal

- convergence speed of CSA-ES becomes lower as the function becomes ill conditioned (a^2 becomes greater) \Rightarrow **covariance matrix adaptation required**

Step-Size Control Is Not Enough



On 20D TwoAxes Function: $f(\mathbf{x}) = \sum_{i=1}^{N/2} [\mathbf{R}\mathbf{x}]_i^2 + a^2 \sum_{i=N/2+1}^N [\mathbf{R}\mathbf{x}]_i^2$, \mathbf{R} : orthogonal

- convergence speed of CSA-ES becomes lower as the function becomes ill conditioned (a^2 becomes greater) \Rightarrow **covariance matrix adaptation required**

Evolution Strategies

Recalling

New search points are sampled normally distributed

$$\mathbf{x}_i \sim \mathbf{m} + \sigma \mathcal{N}_i(\mathbf{0}, \mathbf{C}) \quad \text{for } i = 1, \dots, \lambda$$

as perturbations of \mathbf{m} , where $\mathbf{x}_i, \mathbf{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\mathbf{C} \in \mathbb{R}^{n \times n}$

where

- the **mean** vector $\mathbf{m} \in \mathbb{R}^n$ represents the favorite solution
- the so-called **step-size** $\sigma \in \mathbb{R}_+$ controls the *step length*
- the **covariance matrix** $\mathbf{C} \in \mathbb{R}^{n \times n}$ determines the **shape** of the distribution ellipsoid

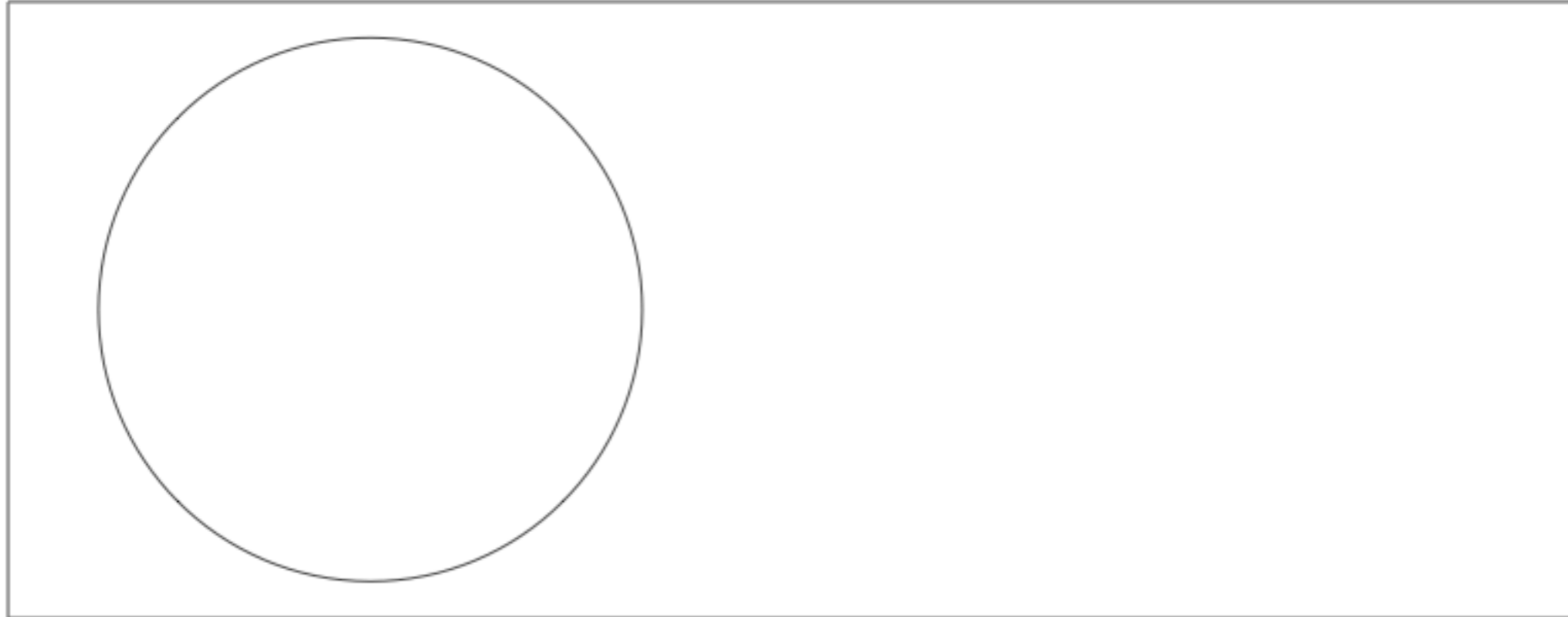
The remaining question is how to update \mathbf{C} .



Covariance Matrix Adaptation

Rank-One Update

$$\mathbf{m} \leftarrow \mathbf{m} + \sigma \mathbf{y}_w, \quad \mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}, \quad \mathbf{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



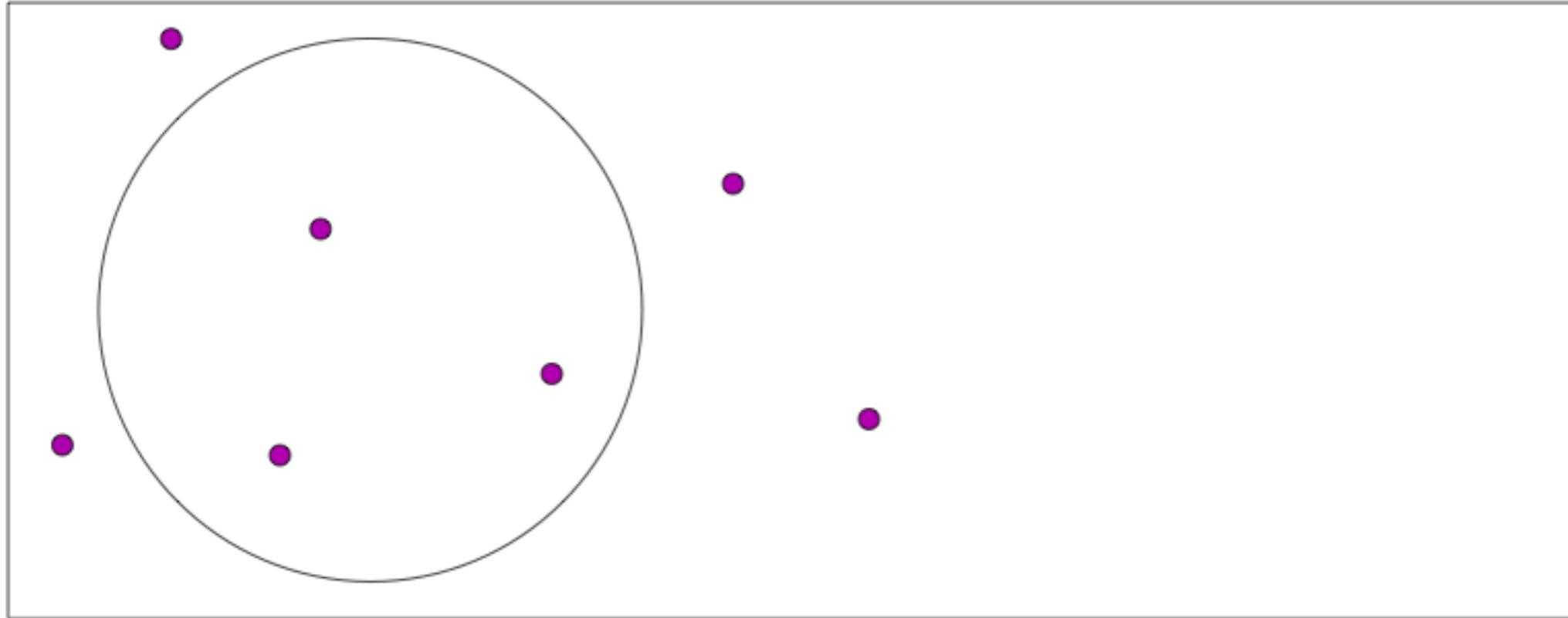
initial distribution, $\mathbf{C} = \mathbf{I}$

... equations

Covariance Matrix Adaptation

Rank-One Update

$$\mathbf{m} \leftarrow \mathbf{m} + \sigma \mathbf{y}_w, \quad \mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}, \quad \mathbf{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



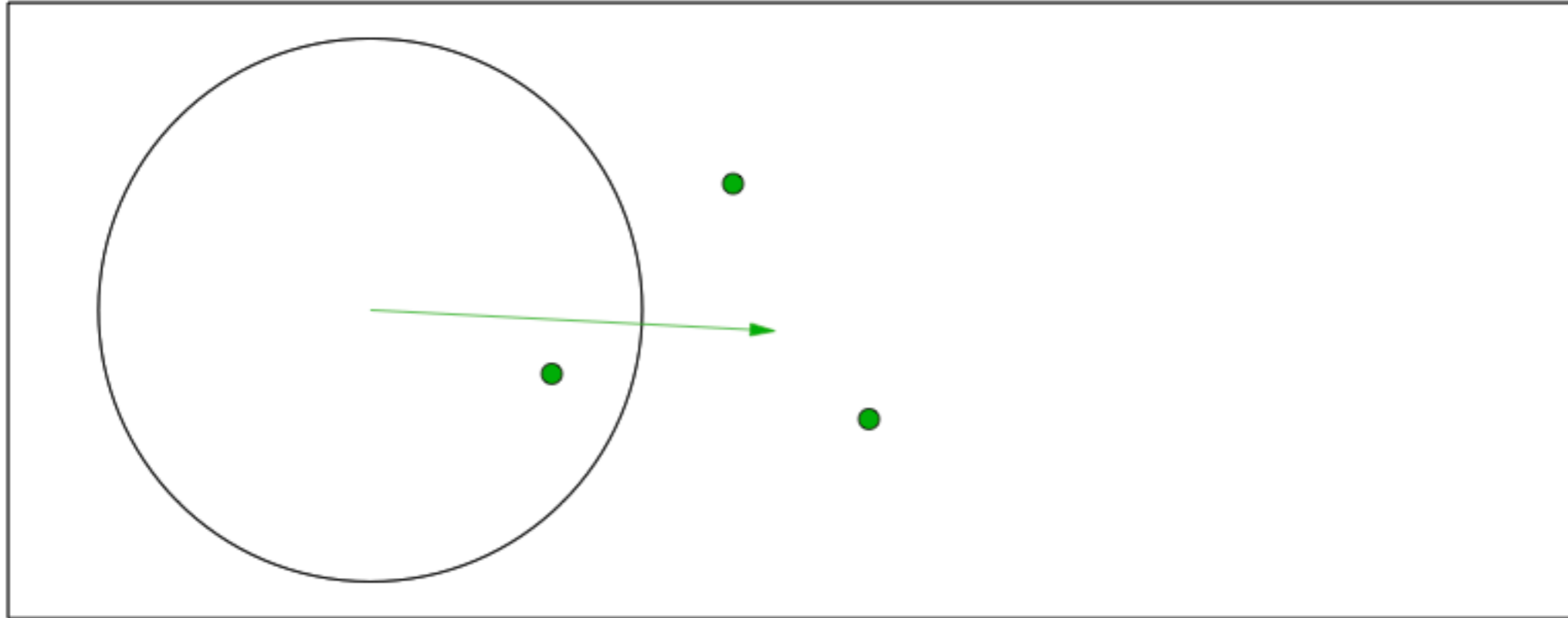
initial distribution, $\mathbf{C} = \mathbf{I}$

... equations

Covariance Matrix Adaptation

Rank-One Update

$$\mathbf{m} \leftarrow \mathbf{m} + \sigma \mathbf{y}_w, \quad \mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}, \quad \mathbf{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



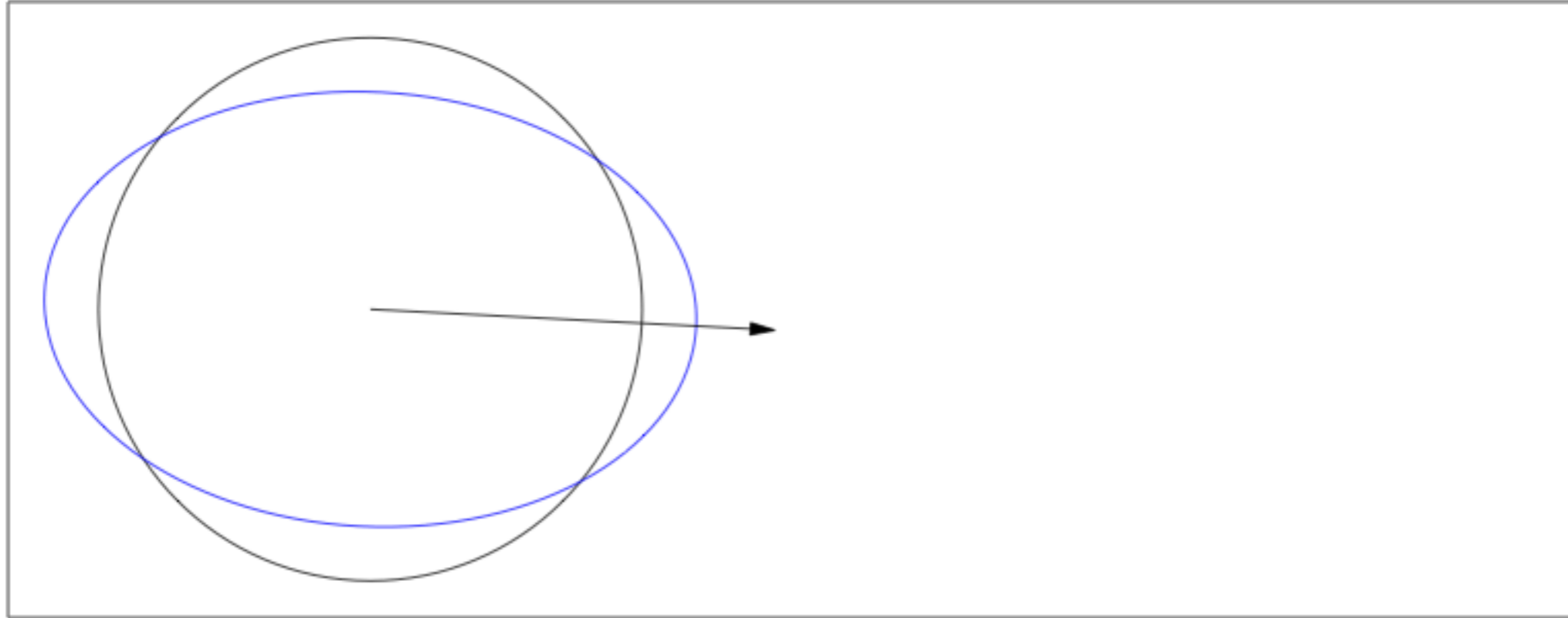
\mathbf{y}_w , movement of the population mean \mathbf{m} (disregarding σ)

... equations

Covariance Matrix Adaptation

Rank-One Update

$$\mathbf{m} \leftarrow \mathbf{m} + \sigma \mathbf{y}_w, \quad \mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}, \quad \mathbf{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



mixture of distribution \mathbf{C} and step \mathbf{y}_w ,

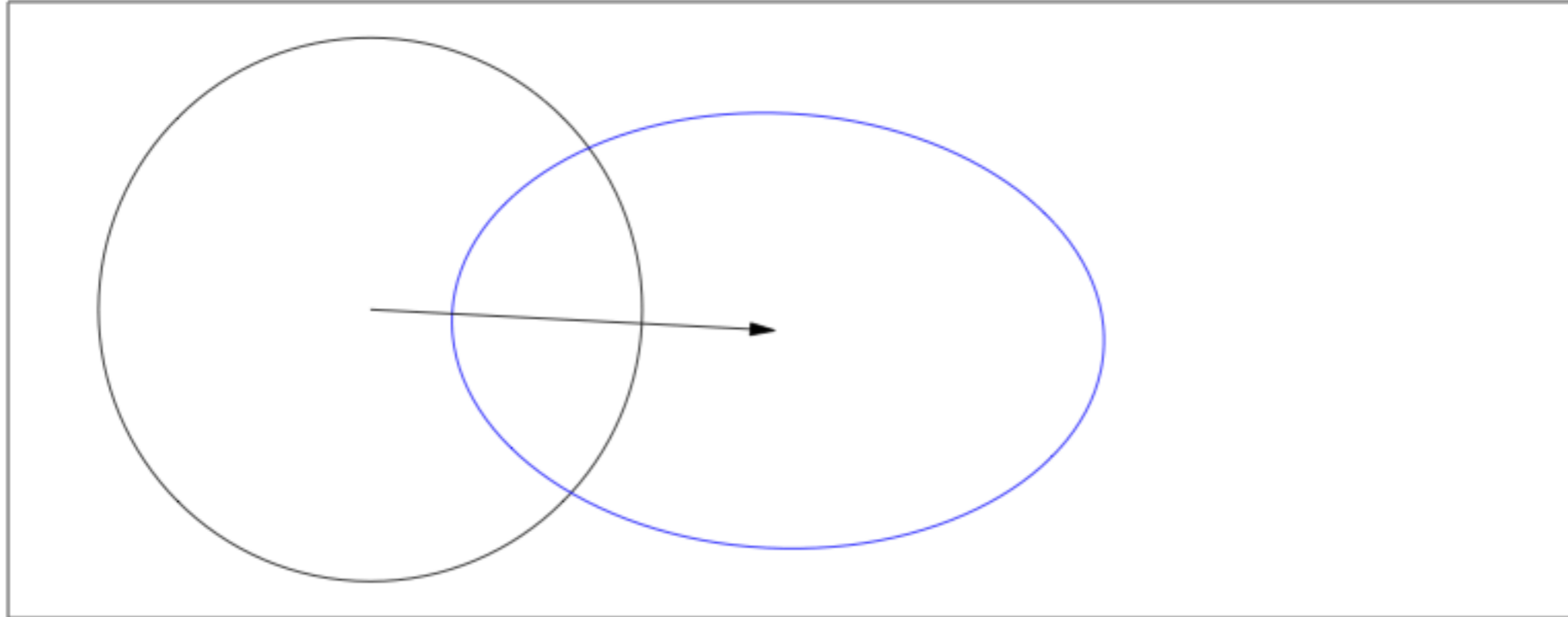
$$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \mathbf{y}_w \mathbf{y}_w^T$$

... equations

Covariance Matrix Adaptation

Rank-One Update

$$\mathbf{m} \leftarrow \mathbf{m} + \sigma \mathbf{y}_w, \quad \mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}, \quad \mathbf{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



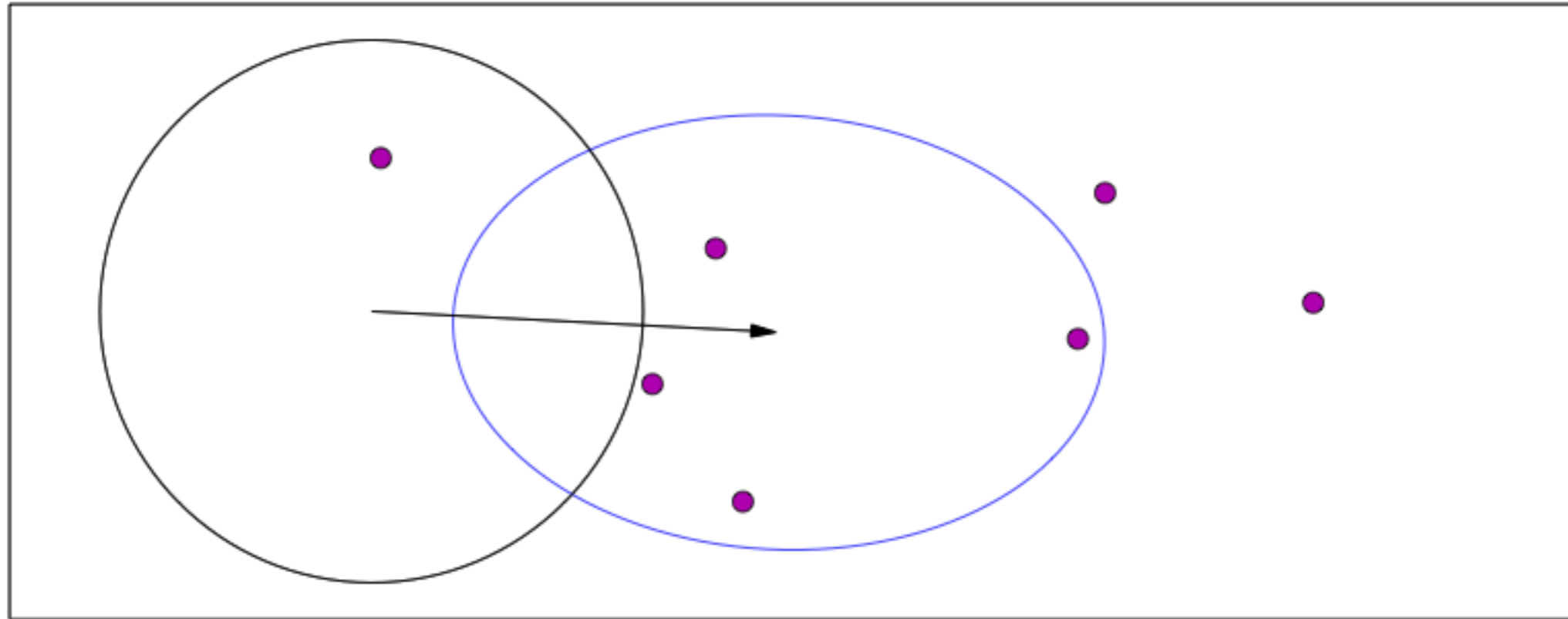
new distribution (disregarding σ)

... equations

Covariance Matrix Adaptation

Rank-One Update

$$\mathbf{m} \leftarrow \mathbf{m} + \sigma \mathbf{y}_w, \quad \mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}, \quad \mathbf{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



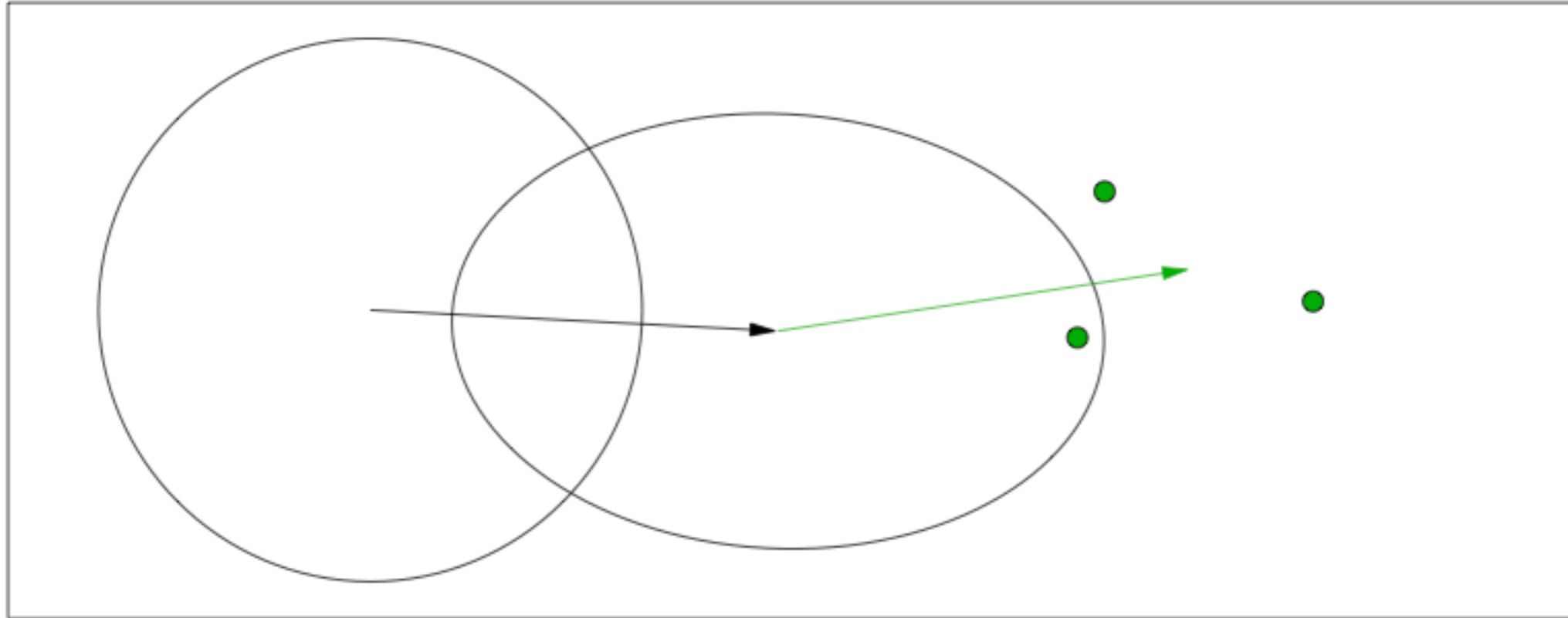
new distribution (disregarding σ)

... equations

Covariance Matrix Adaptation

Rank-One Update

$$\mathbf{m} \leftarrow \mathbf{m} + \sigma \mathbf{y}_w, \quad \mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}, \quad \mathbf{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



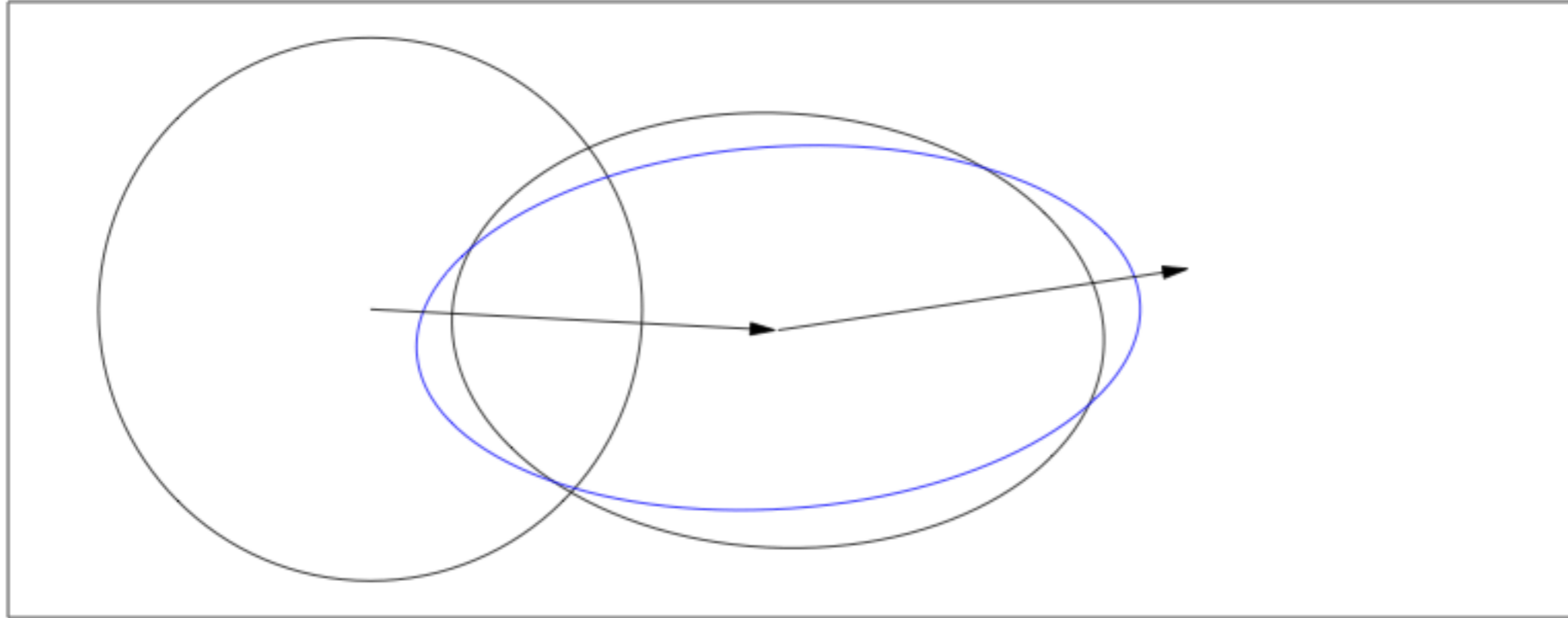
movement of the population mean \mathbf{m}

... equations

Covariance Matrix Adaptation

Rank-One Update

$$\mathbf{m} \leftarrow \mathbf{m} + \sigma \mathbf{y}_w, \quad \mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}, \quad \mathbf{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



mixture of distribution \mathbf{C} and step \mathbf{y}_w ,

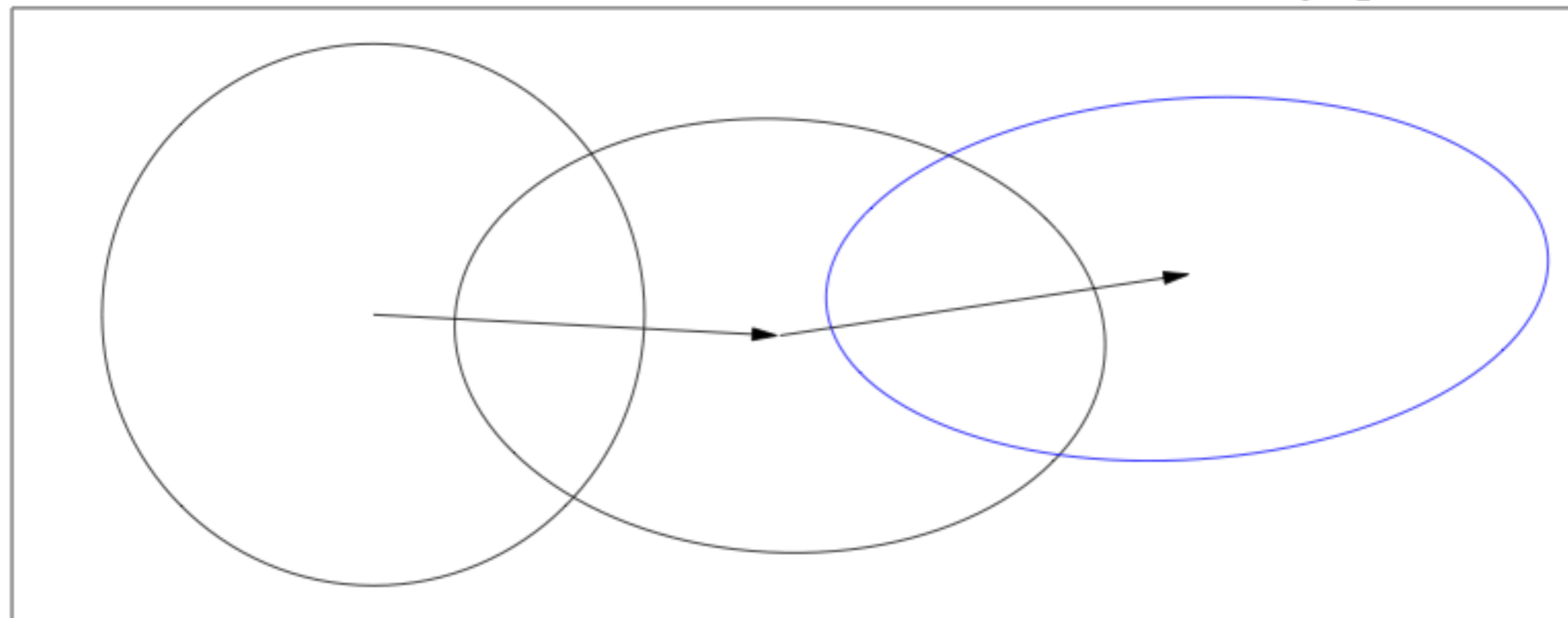
$$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \mathbf{y}_w \mathbf{y}_w^T$$

... equations

Covariance Matrix Adaptation

Rank-One Update

$$\mathbf{m} \leftarrow \mathbf{m} + \sigma \mathbf{y}_w, \quad \mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}, \quad \mathbf{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



new distribution,

$$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \mathbf{y}_w \mathbf{y}_w^T$$

the ruling principle: the adaptation **increases the likelihood of successful steps**, \mathbf{y}_w , to appear again

another viewpoint: the adaptation **follows a natural gradient**

approximation of the expected fitness

... equations

Covariance Matrix Adaptation

Rank-One Update

Initialize $\mathbf{m} \in \mathbb{R}^n$, and $\mathbf{C} = \mathbf{I}$, set $\sigma = 1$, learning rate $c_{\text{cov}} \approx 2/n^2$

While not terminate

$$\mathbf{x}_i = \mathbf{m} + \sigma \mathbf{y}_i, \quad \mathbf{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C}),$$

$$\mathbf{m} \leftarrow \mathbf{m} + \sigma \mathbf{y}_w \quad \text{where } \mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}$$

$$\mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}} \underbrace{\mu_w}_{\text{rank-one}} \mathbf{y}_w \mathbf{y}_w^T \quad \text{where } \mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \geq 1$$

The rank-one update has been found independently in several domains^{6 7 8 9}

⁶ Kjellström&Taxén 1981. Stochastic Optimization in System Design, IEEE TCS

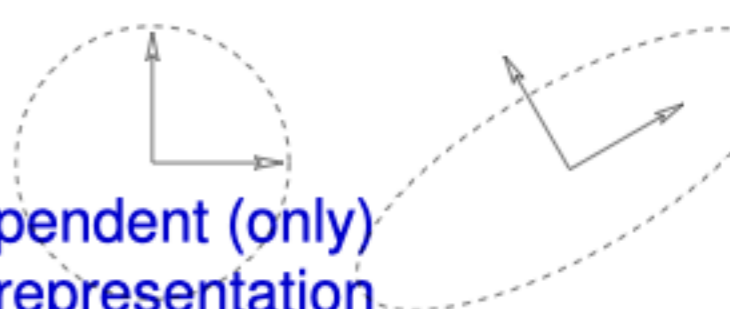
⁷ Hansen&Ostermeier 1996. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation, ICEC

⁸ Ljung 1999. System Identification: Theory for the User

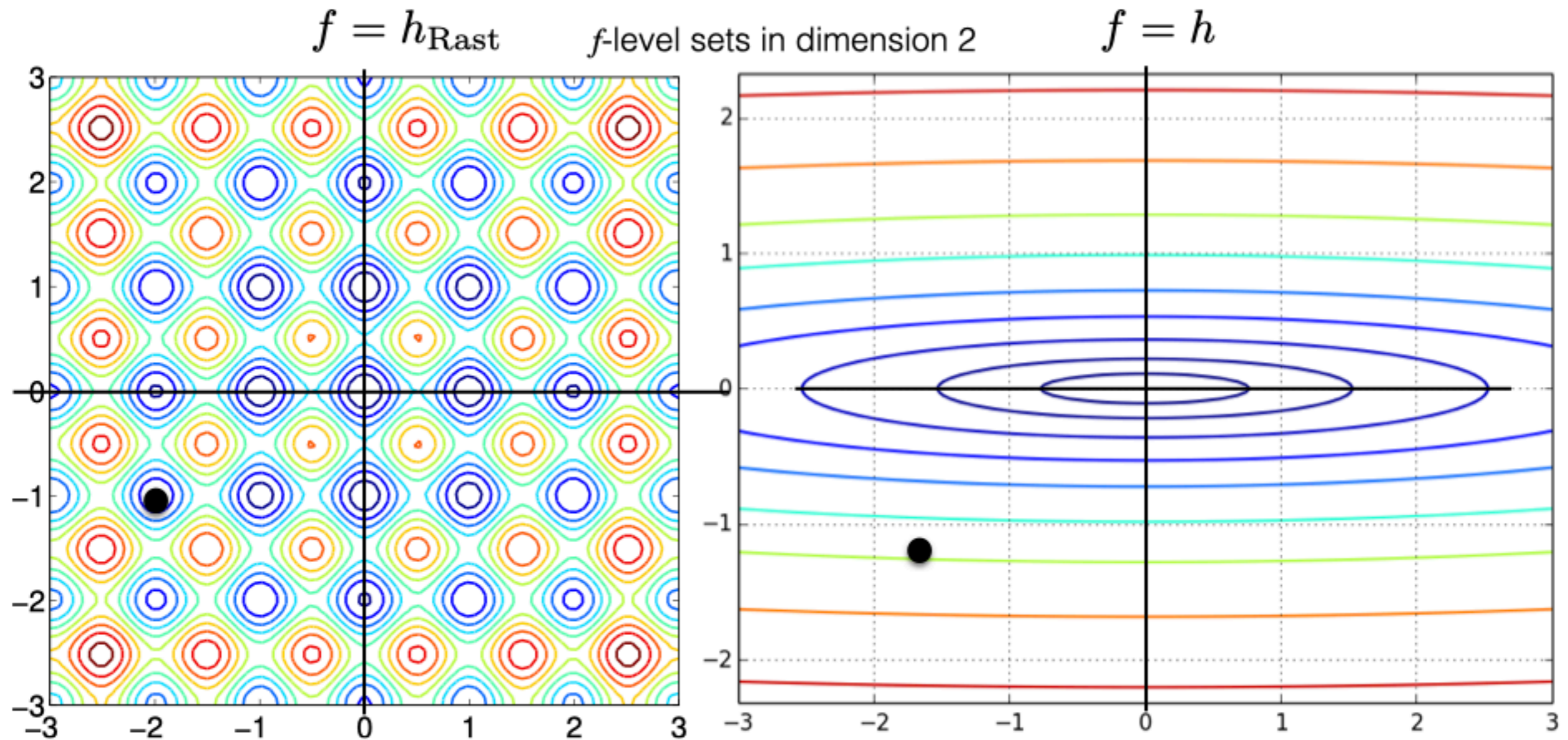
⁹ Haario et al 2001. An adaptive Metropolis algorithm, JSTOR

$$\mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}}\mu_w \mathbf{y}_w \mathbf{y}_w^T$$

covariance matrix adaptation

- learns all **pairwise dependencies** between variables
off-diagonal entries in the covariance matrix reflect the dependencies
- conducts a **principle component analysis** (PCA) of steps \mathbf{y}_w ,
sequentially in time and space
eigenvectors of the covariance matrix \mathbf{C} are the principle components / the principle axes of the mutation ellipsoid
- learns a new **rotated problem representation**
components are independent (only)
in the new representation 
- learns a **new** (Mahalanobis) **metric**
variable metric method
- approximates the **inverse Hessian** on quadratic functions
transformation into the sphere function
- for $\mu = 1$: conducts a **natural gradient ascent** on the distribution \mathcal{N}
entirely independent of the given coordinate system

Invariance Under Rigid Search Space Transformation



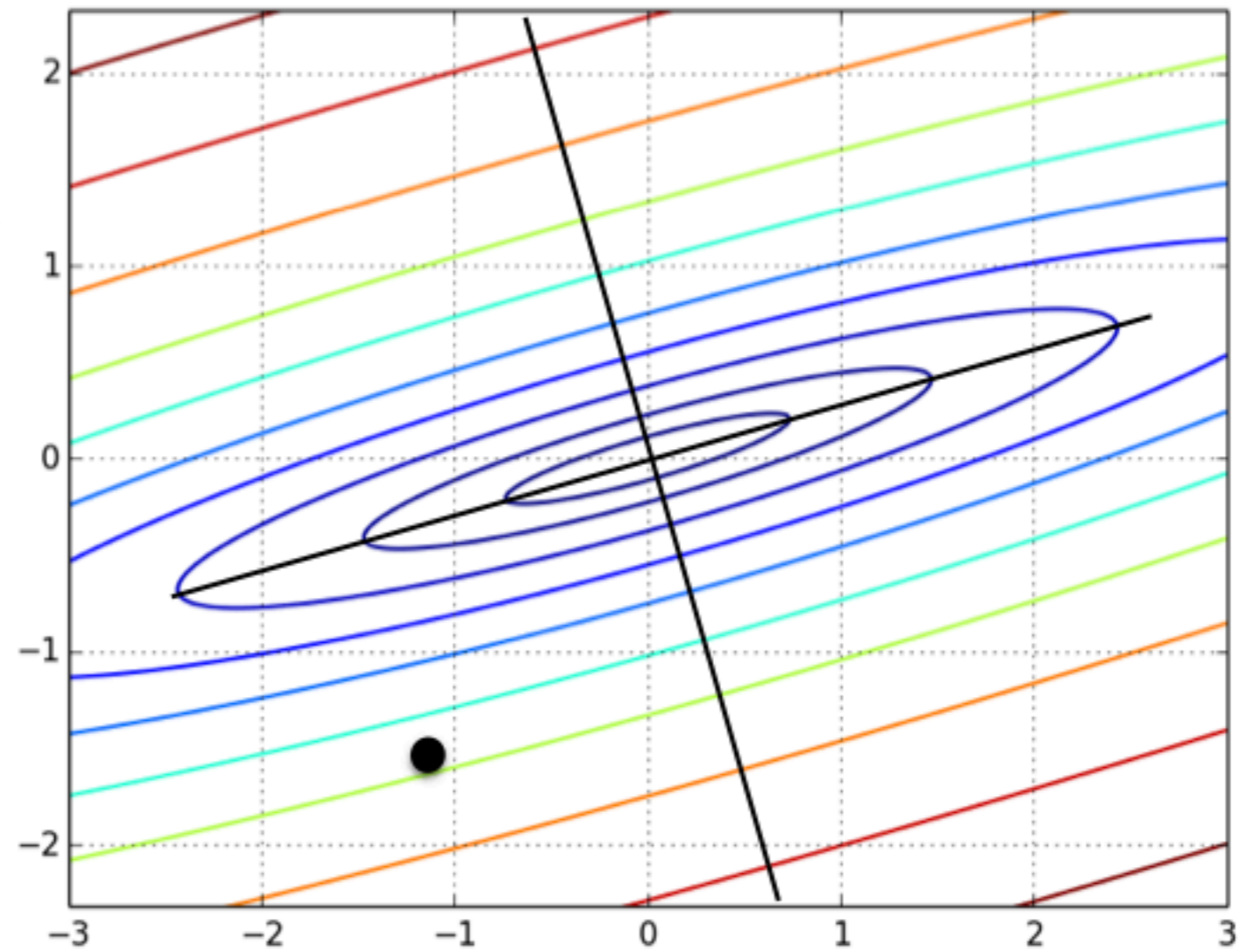
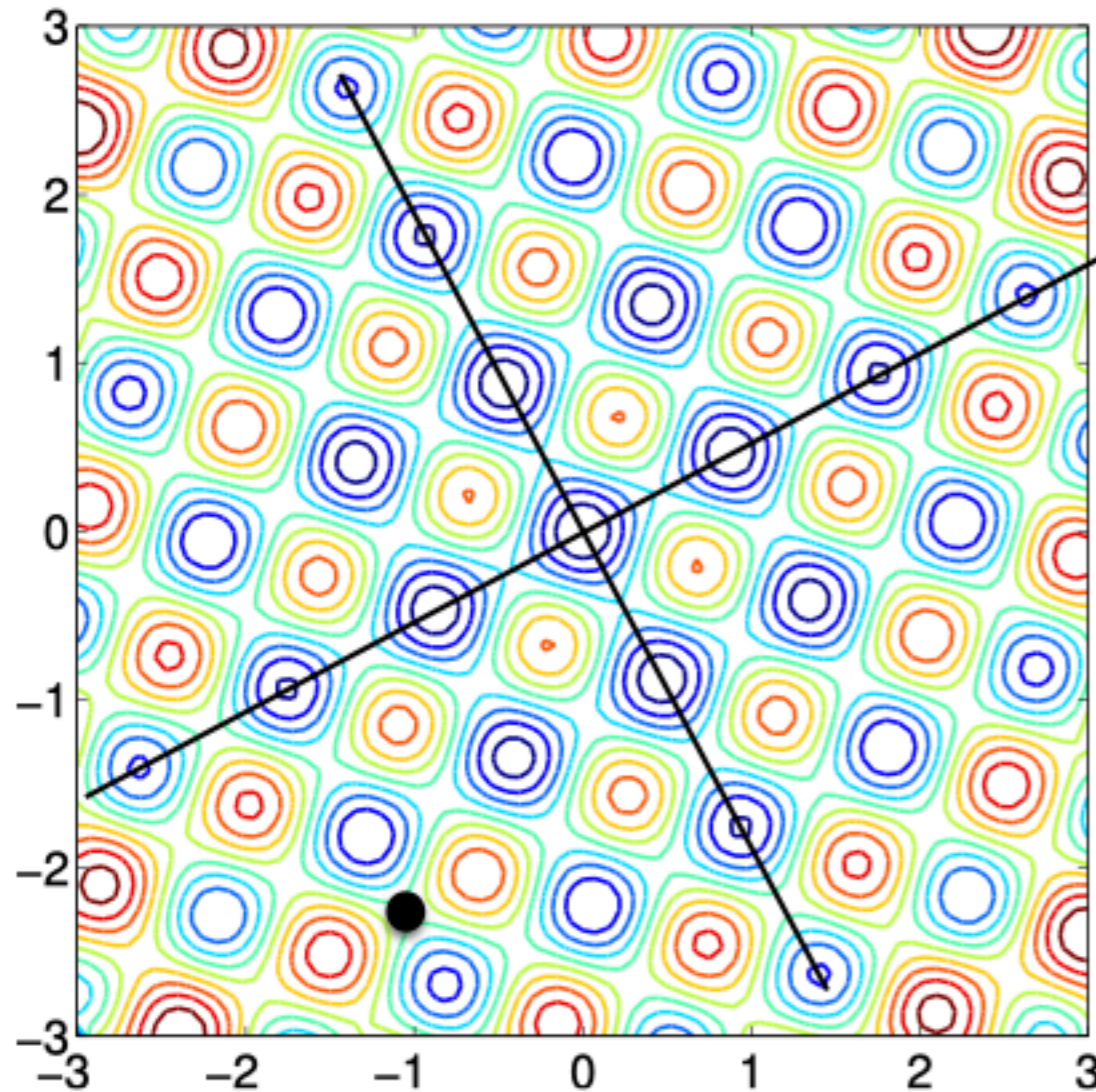
for example, invariance under search space rotation
 (separable \Leftrightarrow non-separable)

Invariance Under Rigid Search Space Transformation

$$f = h_{\text{Rast}} \circ R$$

f -level sets in dimension 2

$$f = h \circ R$$



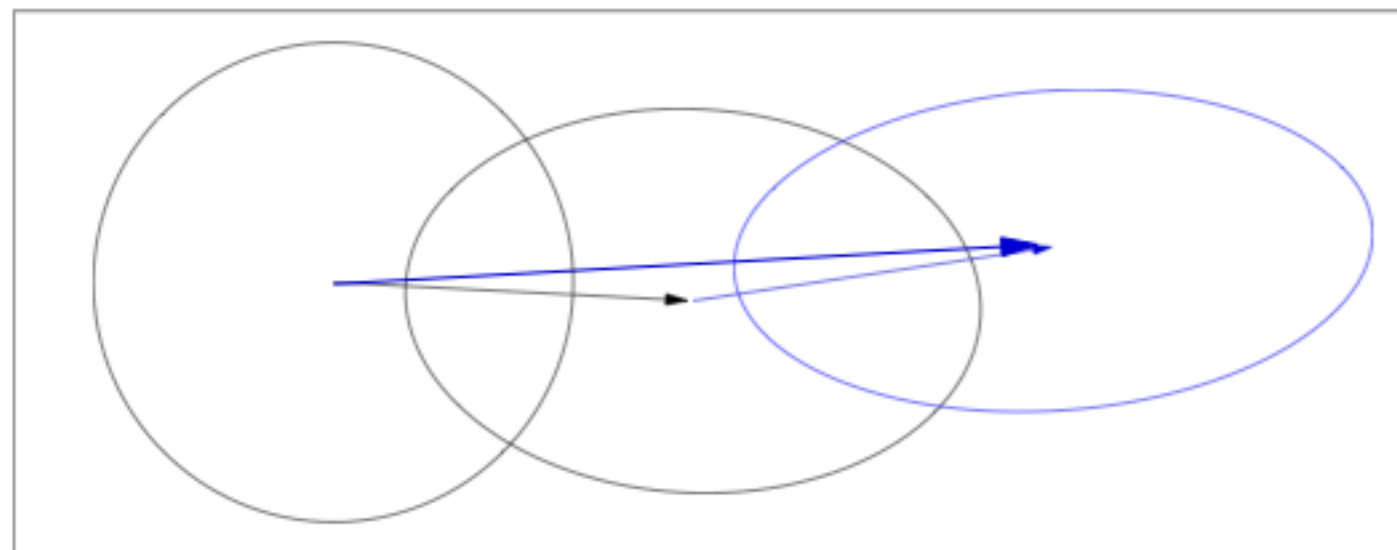
for example, invariance under search space rotation
(separable \Leftrightarrow **non-separable**)

Cumulation

The Evolution Path

Evolution Path

Conceptually, the evolution path is the **search path** the strategy takes **over a number of generation steps**. It can be expressed as a sum of consecutive *steps* of the mean m .



An exponentially weighted sum of steps y_w is used

$$p_c \propto \sum_{i=0}^g \underbrace{(1 - c_c)^{g-i}}_{\text{exponentially fading weights}} y_w^{(i)}$$

The recursive construction of the evolution path (cumulation):

$$p_c \leftarrow \underbrace{(1 - c_c)}_{\text{decay factor}} p_c + \underbrace{\sqrt{1 - (1 - c_c)^2}}_{\text{normalization factor}} \sqrt{\mu_w} \underbrace{y_w}_{\text{input} = \frac{m - m_{\text{old}}}{\sigma}}$$

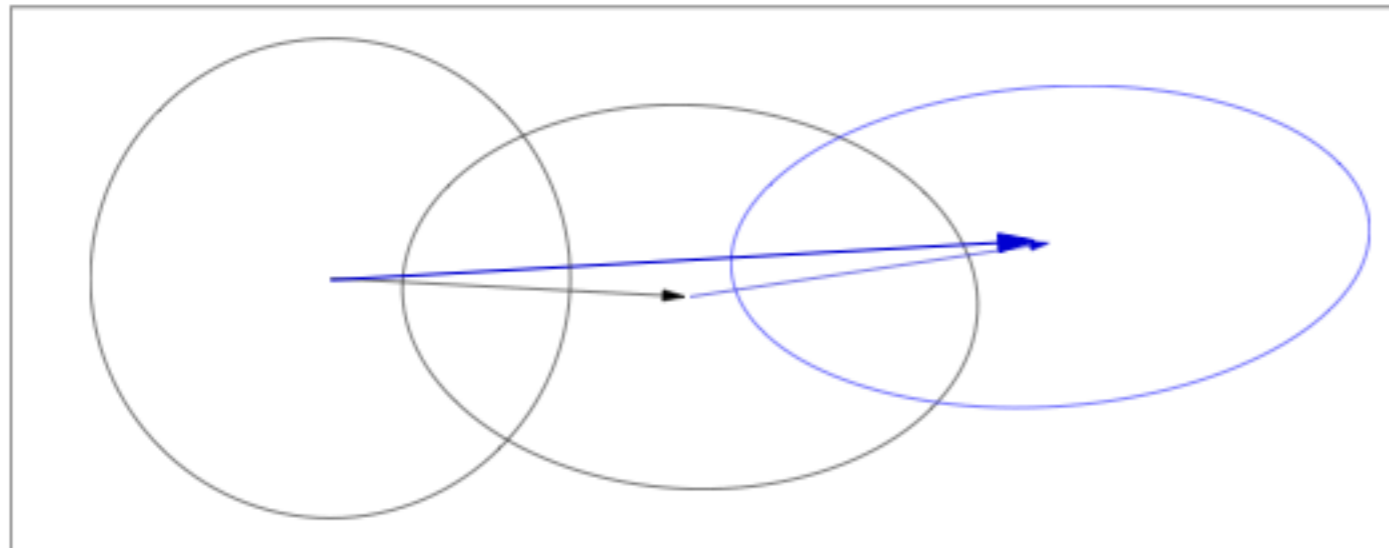
where $\mu_w = \frac{1}{\sum w_i^2}$, $c_c \ll 1$. **History information** is accumulated in the evolution path.

Cumulation

The Evolution Path

Evolution Path

Conceptually, the evolution path is the **search path** the strategy takes **over a number of generation steps**. It can be expressed as a sum of consecutive *steps* of the mean m .



An exponentially weighted sum of steps y_w is used

$$p_c \propto \sum_{i=0}^g \underbrace{(1 - c_c)^{g-i}}_{\text{exponentially fading weights}} y_w^{(i)}$$

The recursive construction of the evolution path (cumulation):

$$p_c \leftarrow \underbrace{(1 - c_c)}_{\text{decay factor}} p_c + \underbrace{\sqrt{1 - (1 - c_c)^2}}_{\text{normalization factor}} \sqrt{\mu_w} \underbrace{y_w}_{\text{input} = \frac{m - m_{\text{old}}}{\sigma}}$$

where $\mu_w = \frac{1}{\sum w_i^2}$, $c_c \ll 1$. **History information** is accumulated in the evolution path.

“Cumulation” is a widely used technique and also know as

- *exponential smoothing* in time series, forecasting
- exponentially weighted *moving average*
- *iterate averaging* in stochastic approximation
- *momentum* in the back-propagation algorithm for ANNs
- ...

“Cumulation” conducts a *low-pass* filtering, but there is more to it. . .

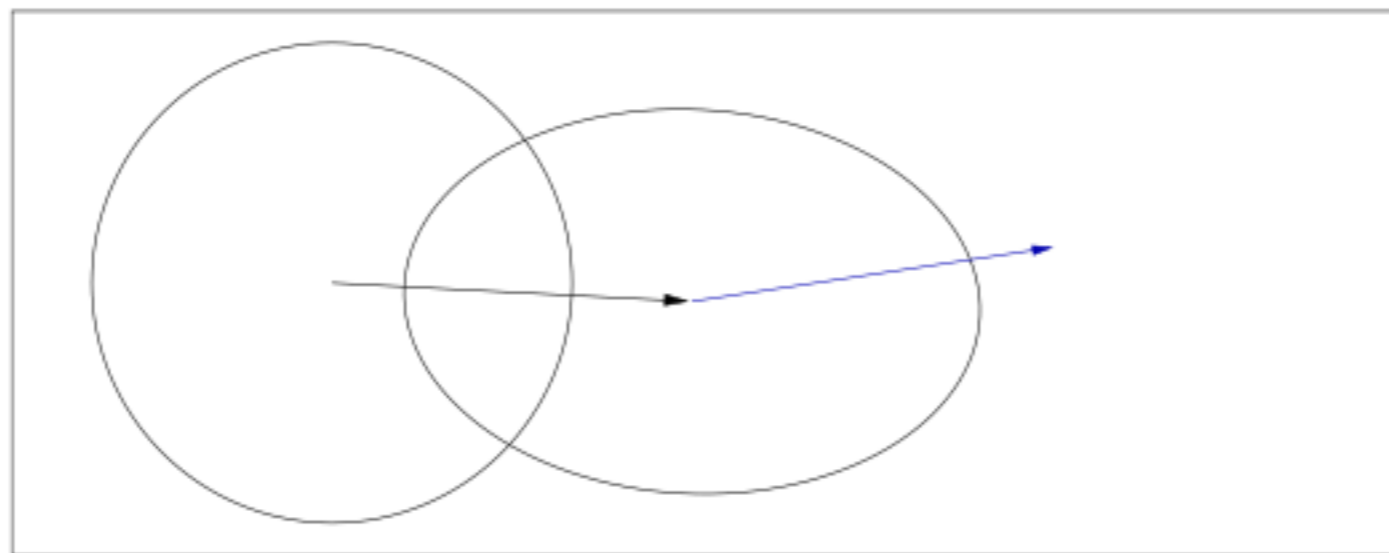
...why?

Cumulation

$$\mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}}\mu_w\mathbf{y}_w\mathbf{y}_w^T$$

Utilizing the Evolution Path

We used $\mathbf{y}_w\mathbf{y}_w^T$ for updating \mathbf{C} . Because $\mathbf{y}_w\mathbf{y}_w^T = -\mathbf{y}_w(-\mathbf{y}_w)^T$ the sign of \mathbf{y}_w is lost.



The **sign information** (signifying correlation *between* steps) is (re-)introduced by using the *evolution path*.

$$\mathbf{p}_c \leftarrow \underbrace{(1 - c_c)}_{\text{decay factor}} \mathbf{p}_c + \underbrace{\sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w}}_{\text{normalization factor}} \mathbf{y}_w$$

$$\mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}} \underbrace{\mathbf{p}_c\mathbf{p}_c^T}_{\text{rank-one}}$$

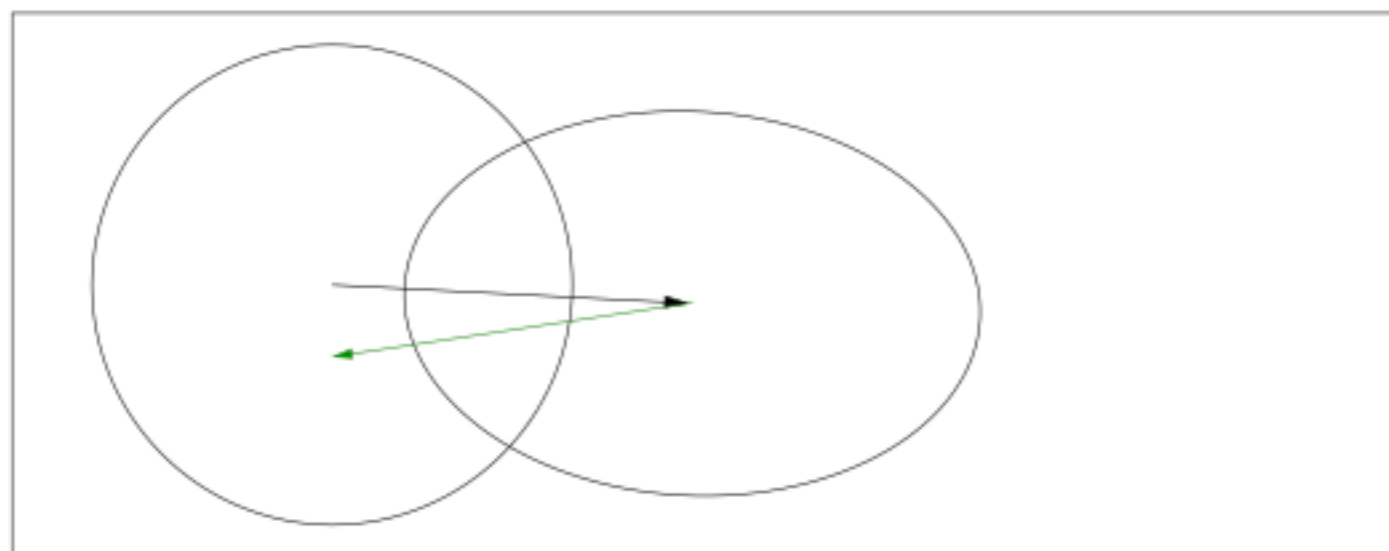
where $\mu_w = \frac{1}{\sum w_i^2}$, $c_{\text{cov}} \ll c_c \ll 1$ such that $1/c_c$ is the “backward time horizon”.

Cumulation

$$\mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}}\mu_w\mathbf{y}_w\mathbf{y}_w^T$$

Utilizing the Evolution Path

We used $\mathbf{y}_w\mathbf{y}_w^T$ for updating \mathbf{C} . Because $\mathbf{y}_w\mathbf{y}_w^T = -\mathbf{y}_w(-\mathbf{y}_w)^T$ the sign of \mathbf{y}_w is lost.



The **sign information** (signifying correlation *between* steps) is (re-)introduced by using the *evolution path*.

$$\mathbf{p}_c \leftarrow \underbrace{(1 - c_c)}_{\text{decay factor}} \mathbf{p}_c + \underbrace{\sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w}}_{\text{normalization factor}} \mathbf{y}_w$$

$$\mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}} \underbrace{\mathbf{p}_c\mathbf{p}_c^T}_{\text{rank-one}}$$

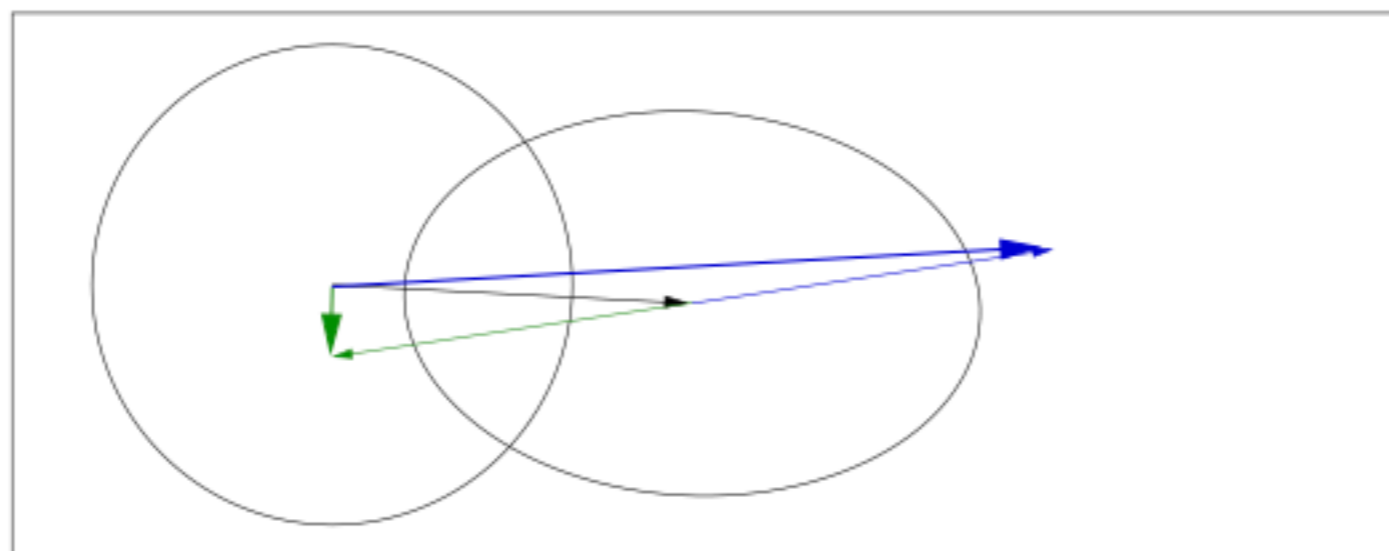
where $\mu_w = \frac{1}{\sum w_i^2}$, $c_{\text{cov}} \ll c_c \ll 1$ such that $1/c_c$ is the “backward time horizon”.

Cumulation

$$\mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}}\mu_w\mathbf{y}_w\mathbf{y}_w^T$$

Utilizing the Evolution Path

We used $\mathbf{y}_w\mathbf{y}_w^T$ for updating \mathbf{C} . Because $\mathbf{y}_w\mathbf{y}_w^T = -\mathbf{y}_w(-\mathbf{y}_w)^T$ the sign of \mathbf{y}_w is lost.



The **sign information** (signifying correlation *between* steps) is (re-)introduced by using the *evolution path*.

$$\mathbf{p}_c \leftarrow \underbrace{(1 - c_c)}_{\text{decay factor}} \mathbf{p}_c + \underbrace{\sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w}}_{\text{normalization factor}} \mathbf{y}_w$$

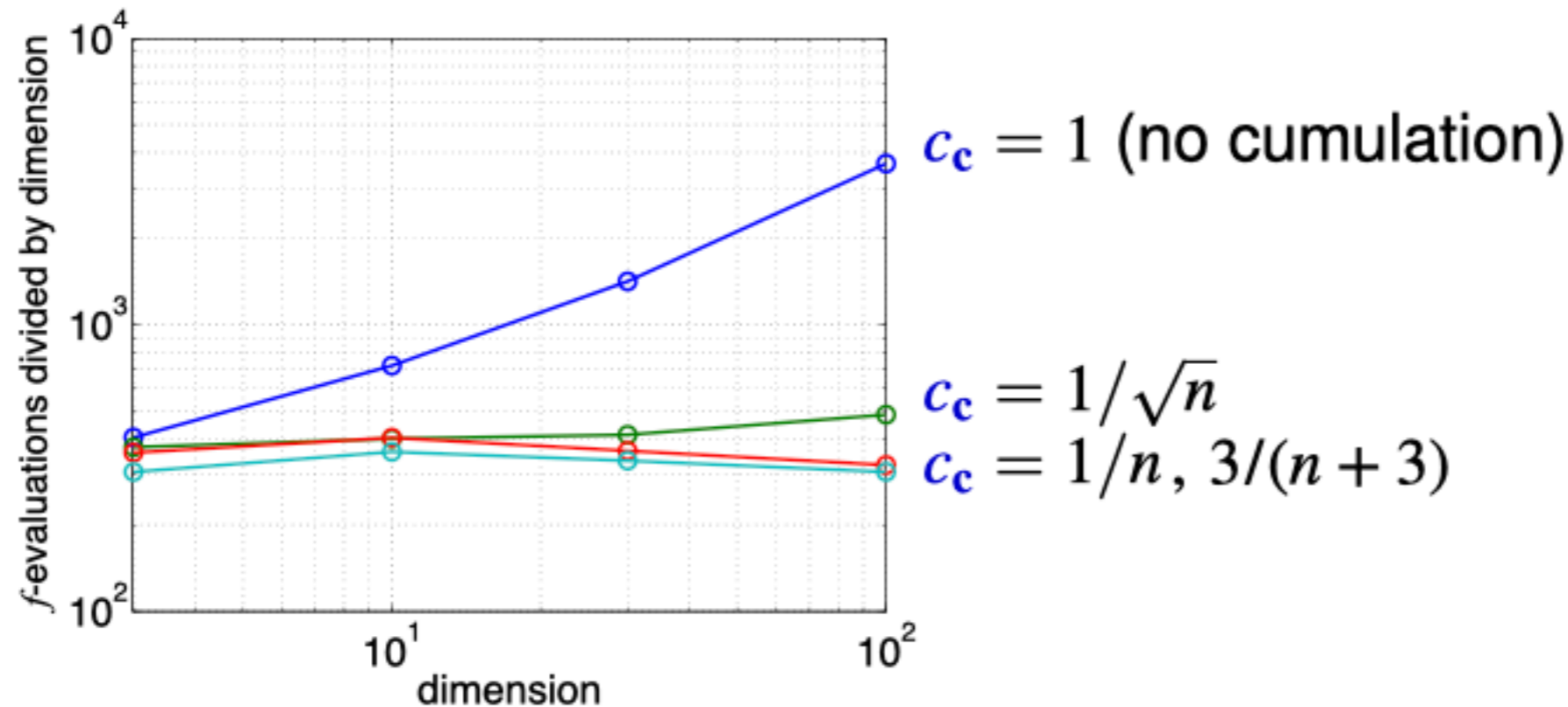
$$\mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}} \underbrace{\mathbf{p}_c \mathbf{p}_c^T}_{\text{rank-one}}$$

where $\mu_w = \frac{1}{\sum w_i^2}$, $c_{\text{cov}} \ll c_c \ll 1$ such that $1/c_c$ is the “backward time horizon”.

Using an **evolution path** for the **rank-one update** of the covariance matrix reduces the number of function evaluations to adapt to a straight ridge **from about $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$** .^(a)

^aHansen & Auger 2013. Principled design of continuous stochastic search: From theory to practice.

Number of f -evaluations divided by dimension on the cigar function $f(\mathbf{x}) = x_1^2 + 10^6 \sum_{i=2}^n x_i^2$



The overall model complexity is n^2 but important parts of the model can be learned in time of order n

Rank- μ Update

$$\begin{aligned} \mathbf{x}_i &= \mathbf{m} + \sigma \mathbf{y}_i, & \mathbf{y}_i &\sim \mathcal{N}_i(\mathbf{0}, \mathbf{C}), \\ \mathbf{m} &\leftarrow \mathbf{m} + \sigma \mathbf{y}_w, & \mathbf{y}_w &= \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda} \end{aligned}$$

The rank- μ update extends the update rule for **large population sizes** λ using $\mu > 1$ vectors to update \mathbf{C} at each generation step.

The weighted empirical covariance matrix

$$\mathbf{C}_\mu = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda} \mathbf{y}_{i:\lambda}^T$$


computes a weighted mean of the outer products of the best μ steps and has rank $\min(\mu, n)$ with probability one.

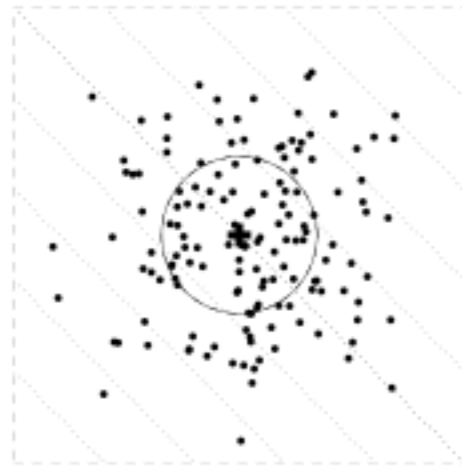
with $\mu = \lambda$ weights can be negative ¹⁰

The rank- μ update then reads

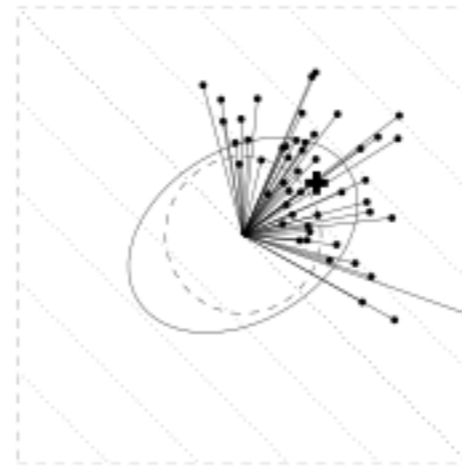
$$\mathbf{C} \leftarrow (1 - c_{\text{cov}}) \mathbf{C} + c_{\text{cov}} \mathbf{C}_\mu$$

where $c_{\text{cov}} \approx \mu_w / n^2$ and $c_{\text{cov}} \leq 1$.

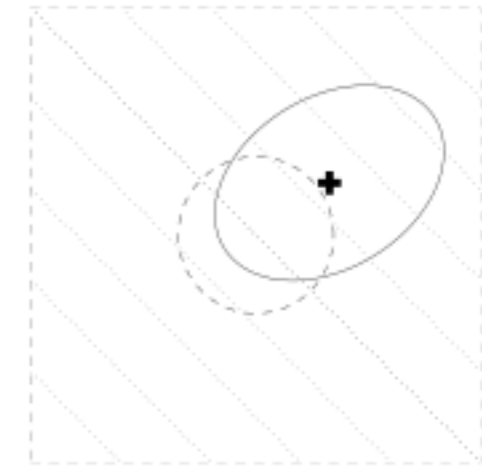
¹⁰Jastrebski and Arnold (2006). Improving evolution strategies through active covariance matrix adaptation. CEC. 



$$x_i = m + \sigma y_i, \quad y_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$$



$$\begin{aligned} \mathbf{C}_\mu &= \frac{1}{\mu} \sum y_{i:\lambda} y_{i:\lambda}^\top \\ \mathbf{C} &\leftarrow (1 - 1) \times \mathbf{C} + 1 \times \mathbf{C}_\mu \end{aligned}$$

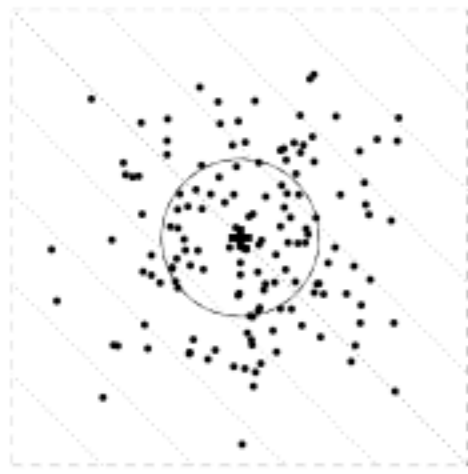


$$m_{\text{new}} \leftarrow m + \frac{1}{\mu} \sum y_{i:\lambda}$$

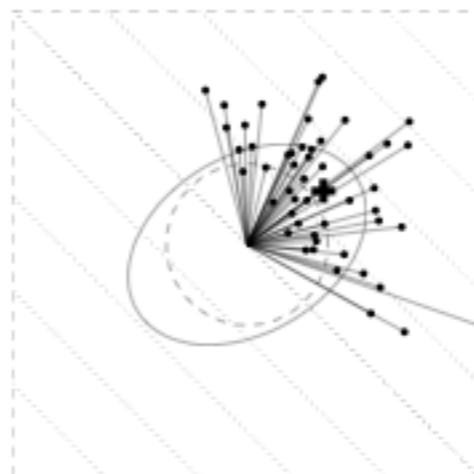
new distribution

sampling of $\lambda = 150$
solutions where
 $\mathbf{C} = \mathbf{I}$ and $\sigma = 1$

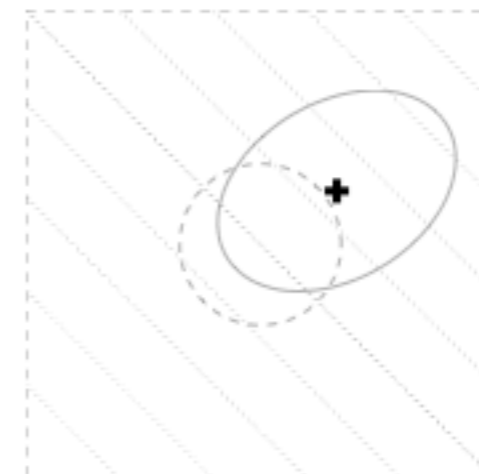
calculating \mathbf{C} where
 $\mu = 50$,
 $w_1 = \dots = w_\mu = \frac{1}{\mu}$,
and $c_{\text{cov}} = 1$

Rank- μ CMA versus Estimation of Multivariate Normal Algorithm EMNA_{global}¹¹

$$x_i = m_{\text{old}} + y_i, \quad y_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$$

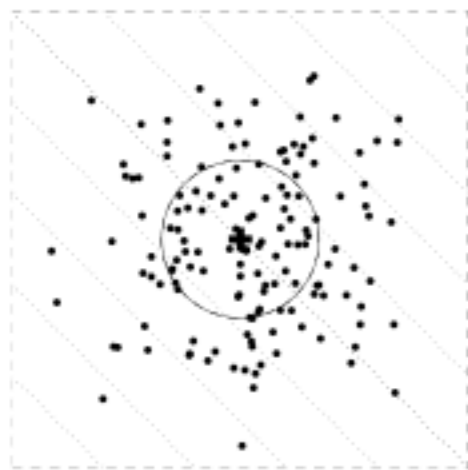


$$\mathbf{C} \leftarrow \frac{1}{\mu} \sum (x_{i:\lambda} - m_{\text{old}})(x_{i:\lambda} - m_{\text{old}})^T$$

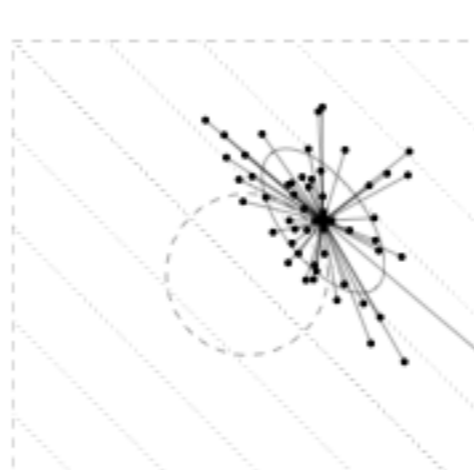


$$m_{\text{new}} = m_{\text{old}} + \frac{1}{\mu} \sum y_{i:\lambda}$$

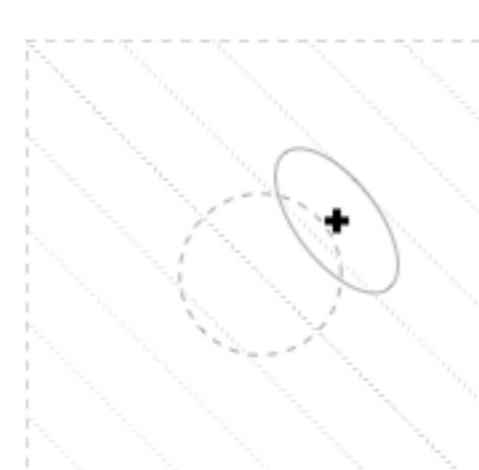
rank- μ CMA
conducts a
PCA of
steps



$$x_i = m_{\text{old}} + y_i, \quad y_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$$



$$\mathbf{C} \leftarrow \frac{1}{\mu} \sum (x_{i:\lambda} - m_{\text{new}})(x_{i:\lambda} - m_{\text{new}})^T$$



$$m_{\text{new}} = m_{\text{old}} + \frac{1}{\mu} \sum y_{i:\lambda}$$

EMNA_{global}
conducts a
PCA of
points

sampling of $\lambda = 150$
solutions (dots)

calculating \mathbf{C} from $\mu = 50$
solutions

new distribution

m_{new} is the minimizer for the variances when calculating \mathbf{C}

¹¹ Hansen, N. (2006). The CMA Evolution Strategy: A Comparing Review. In J.A. Lozano, P. Larranga, I. Inza and E. Bengoetxea (Eds.). Towards a new evolutionary computation. Advances in estimation of distribution algorithms. pp. 75-102

The rank- μ update

- increases the possible learning rate in large populations
roughly from $2/n^2$ to μ_w/n^2
- can reduce the number of necessary **generations** roughly from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ ⁽¹²⁾
given $\mu_w \propto \lambda \propto n$

Therefore the rank- μ update is the primary mechanism whenever a large population size is used

say $\lambda \geq 3n + 10$

The rank-one update

- uses the evolution path and reduces the number of necessary **function evaluations** to learn straight ridges from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.

Rank-one update and rank- μ update can be combined

... all equations

¹²Hansen, Müller, and Koumoutsakos 2003. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11(1), pp. 1-18

The rank- μ update

- increases the possible learning rate in large populations
roughly from $2/n^2$ to μ_w/n^2
- can reduce the number of necessary **generations** roughly from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ ⁽¹²⁾
given $\mu_w \propto \lambda \propto n$

Therefore the rank- μ update is the primary mechanism whenever a large population size is used

say $\lambda \geq 3n + 10$

The rank-one update

- uses the evolution path and reduces the number of necessary **function evaluations** to learn straight ridges from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.

Rank-one update and rank- μ update can be combined

... all equations

¹²Hansen, Müller, and Koumoutsakos 2003. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11(1), pp. 1-18

The rank- μ update

- increases the possible learning rate in large populations
roughly from $2/n^2$ to μ_w/n^2
- can reduce the number of necessary **generations** roughly from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ ⁽¹²⁾
given $\mu_w \propto \lambda \propto n$

Therefore the rank- μ update is the primary mechanism whenever a large population size is used

say $\lambda \geq 3n + 10$

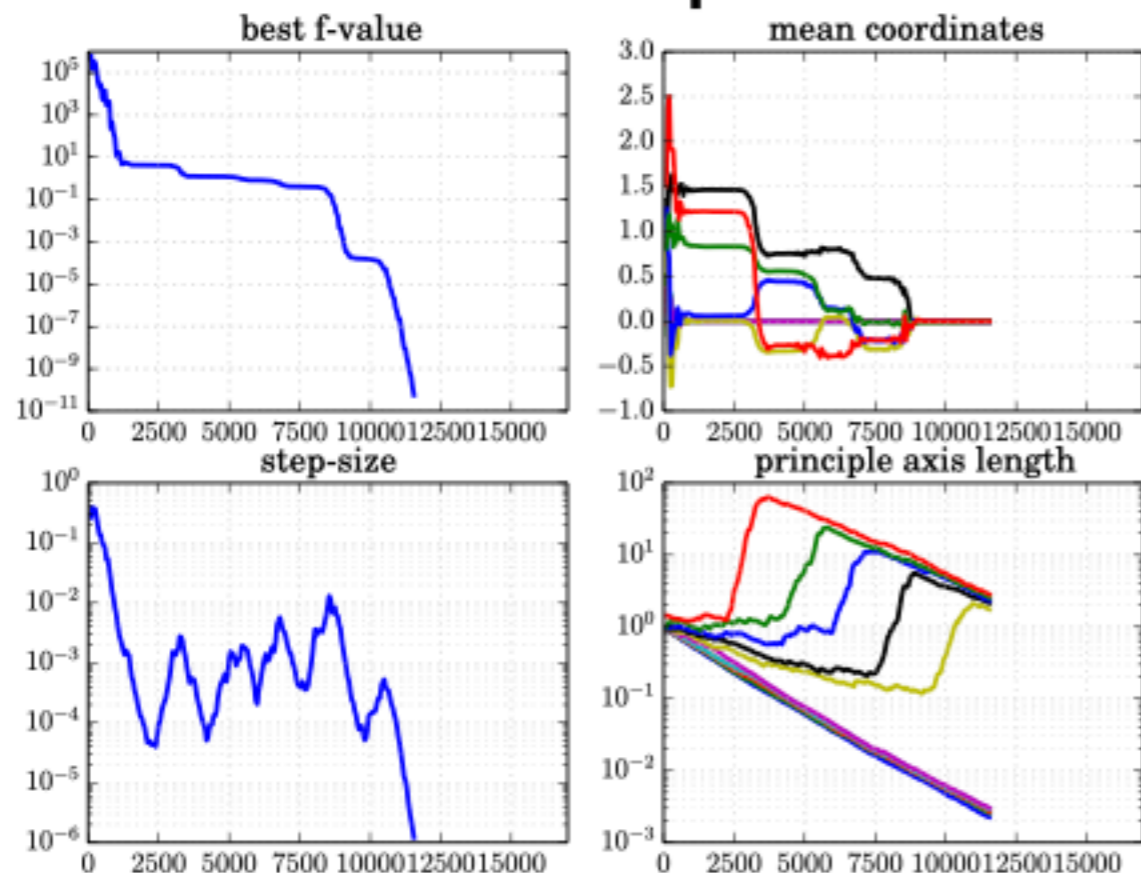
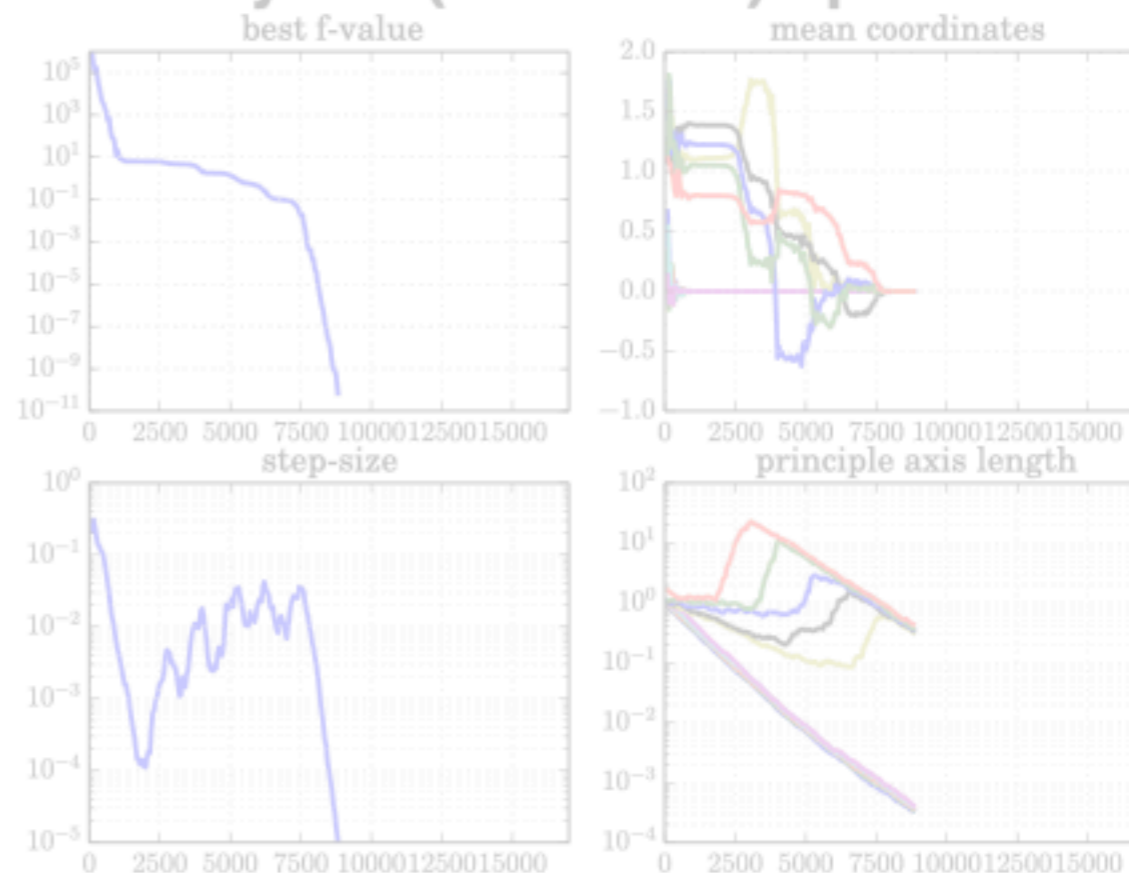
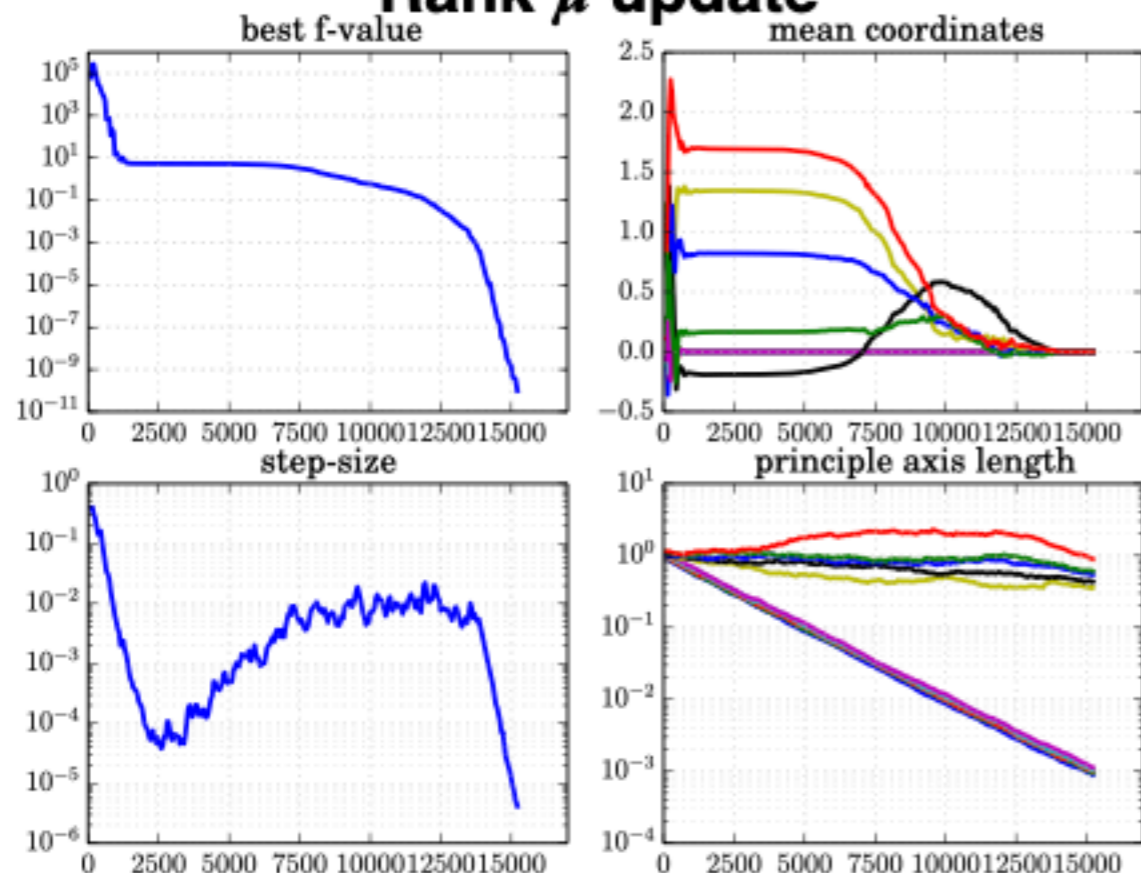
The rank-one update

- uses the evolution path and reduces the number of necessary **function evaluations** to learn straight ridges from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.

Rank-one update and rank- μ update can be combined

... all equations

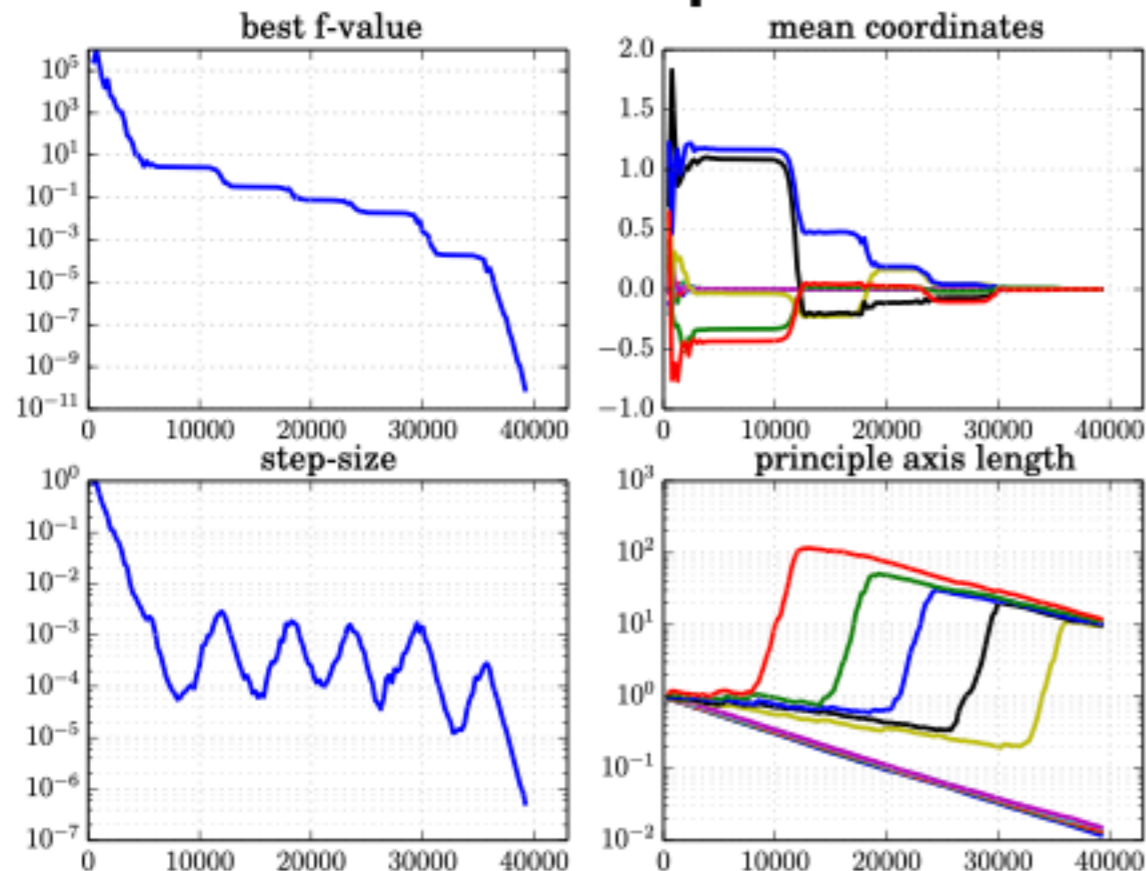
¹²Hansen, Müller, and Koumoutsakos 2003. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11(1), pp. 1-18

Rank-one update**Hybrid (combined) update****Rank- μ update**

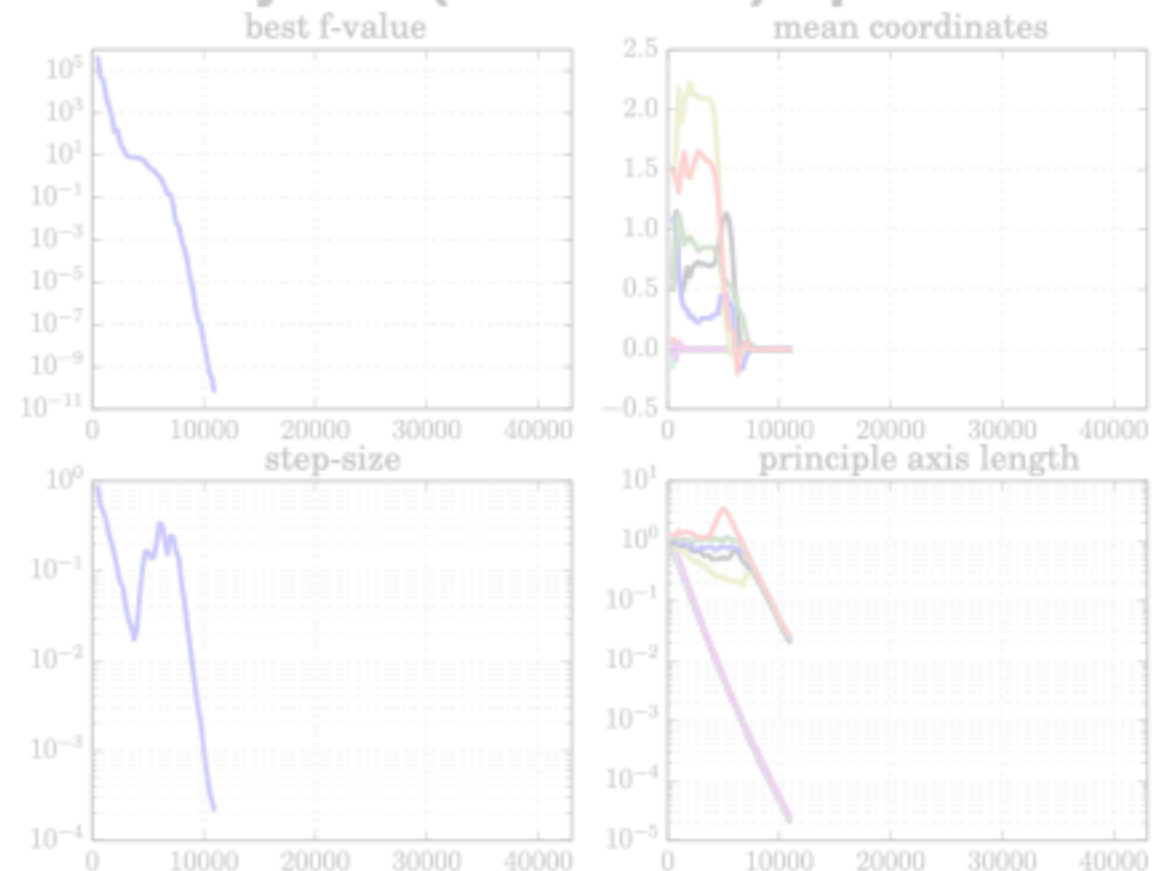
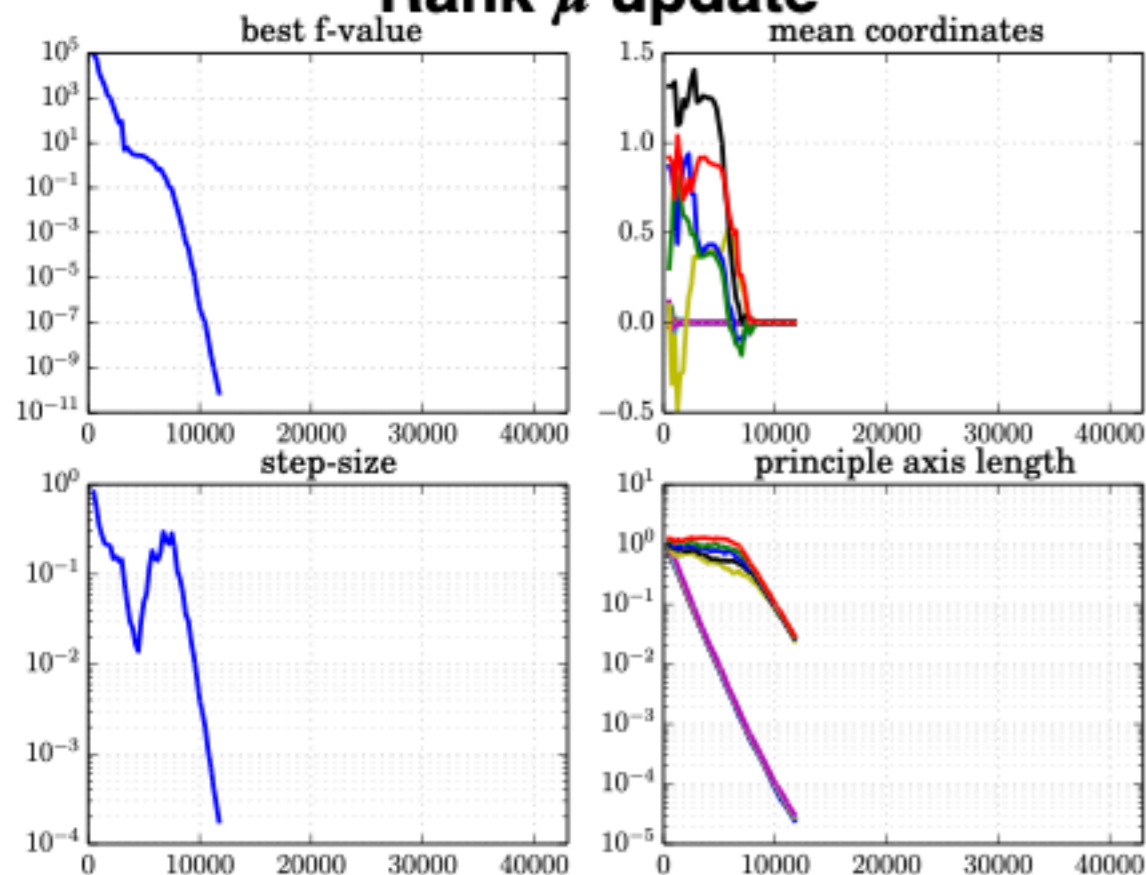
$$f_{\text{TwoAxes}}(x) = \sum_{i=1}^5 x_i^2 + 10^6 \sum_{i=6}^{10} x_i^2$$

$\lambda = 10$ (default for $N = 10$)

Rank-one update



Hybrid (combined) update

Rank- μ update

$$f_{\text{TwoAxes}}(x) = \sum_{i=1}^5 x_i^2 + 10^6 \sum_{i=6}^{10} x_i^2$$

$$\lambda = 50$$

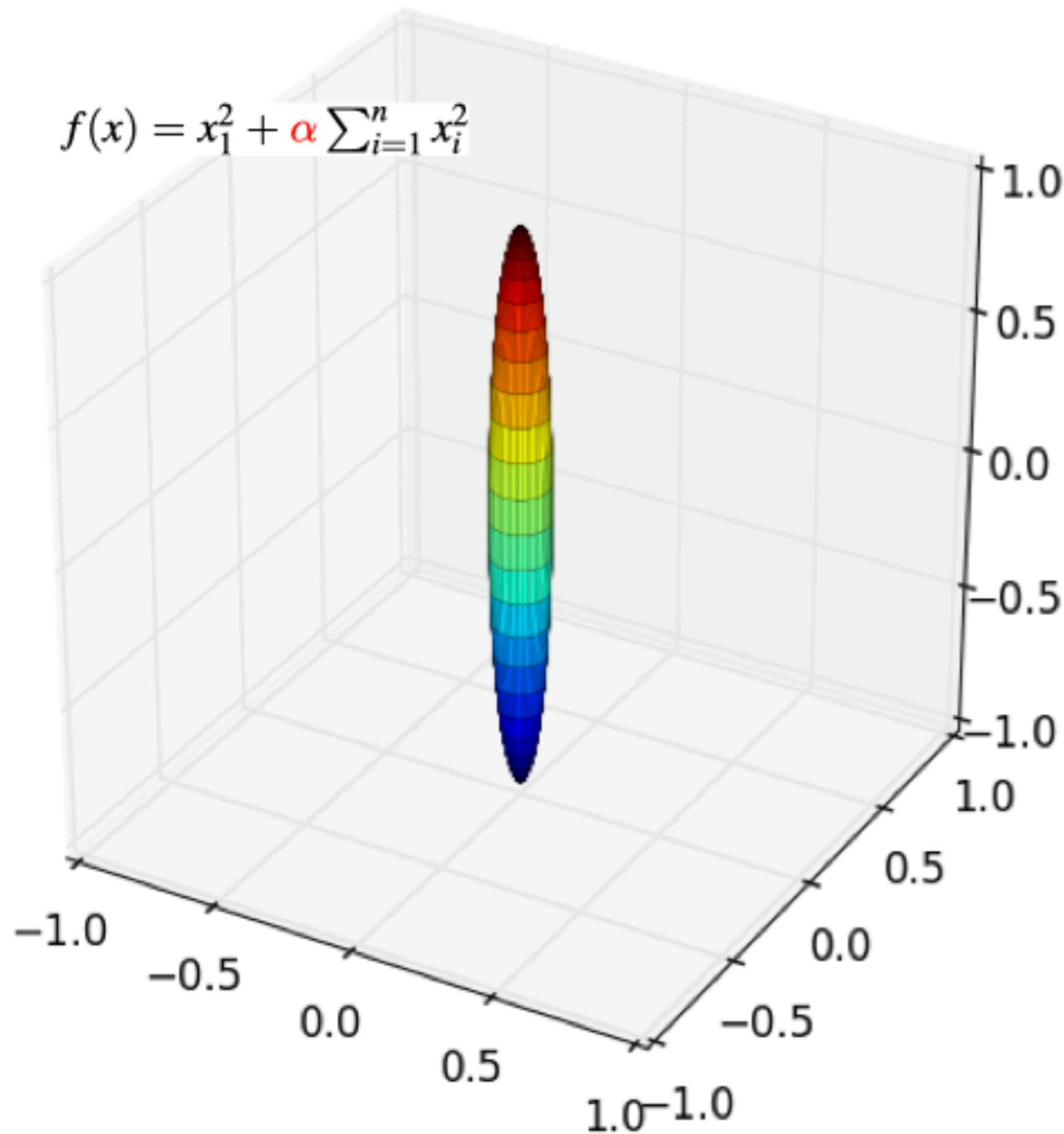
Different Types of Ill-Conditioning

(α : Axes Ratio = 10)

Cigar Type:

1 long axis = n-1 short axes

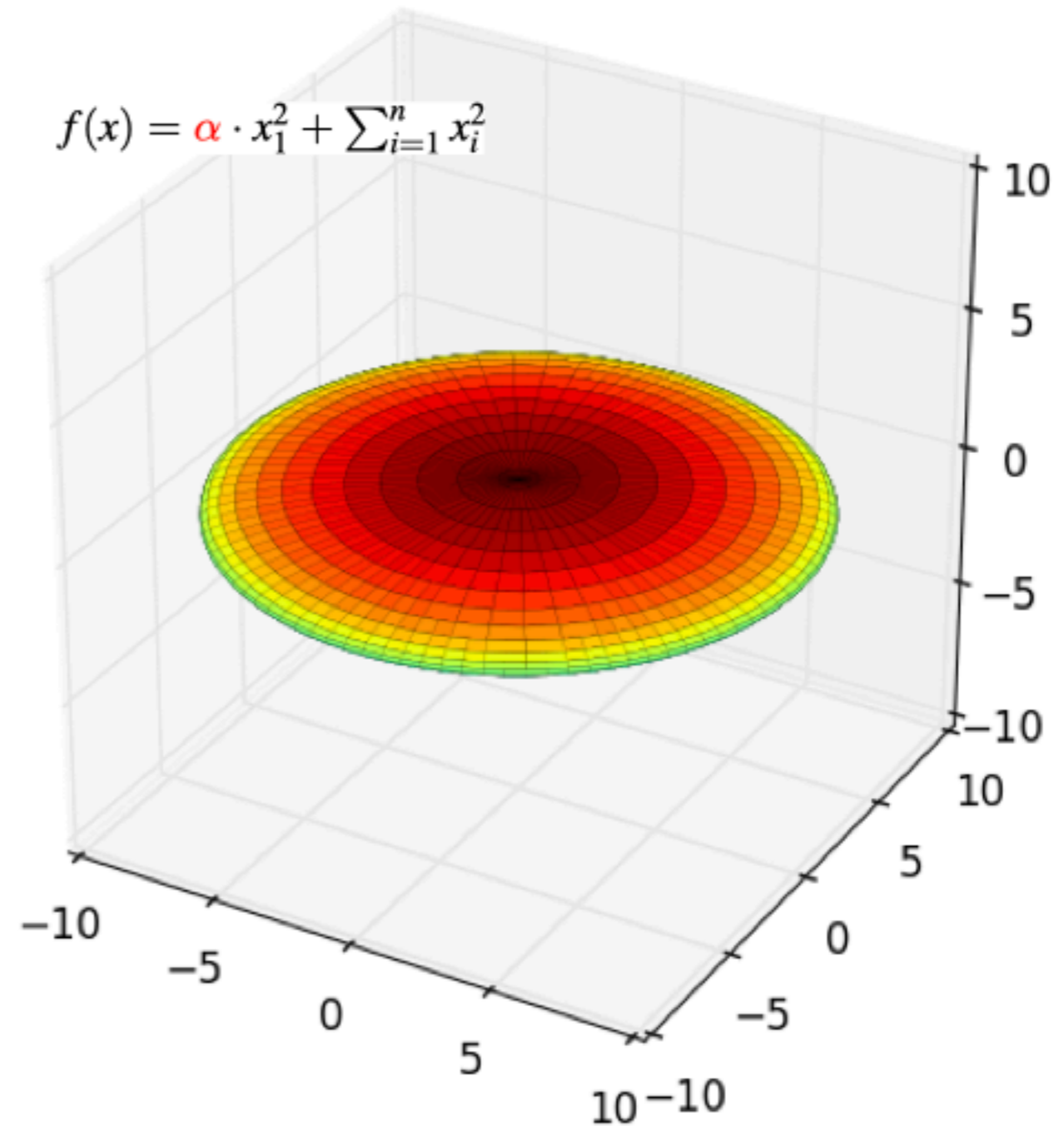
$$f(x) = x_1^2 + \alpha \sum_{i=1}^n x_i^2$$



Discus Type:

1 short axis = n-1 long axes

$$f(x) = \alpha \cdot x_1^2 + \sum_{i=1}^n x_i^2$$



Active Update

utilize negative weights [Jastrebski and Arnold, 2006]

Active Update (rewriting)

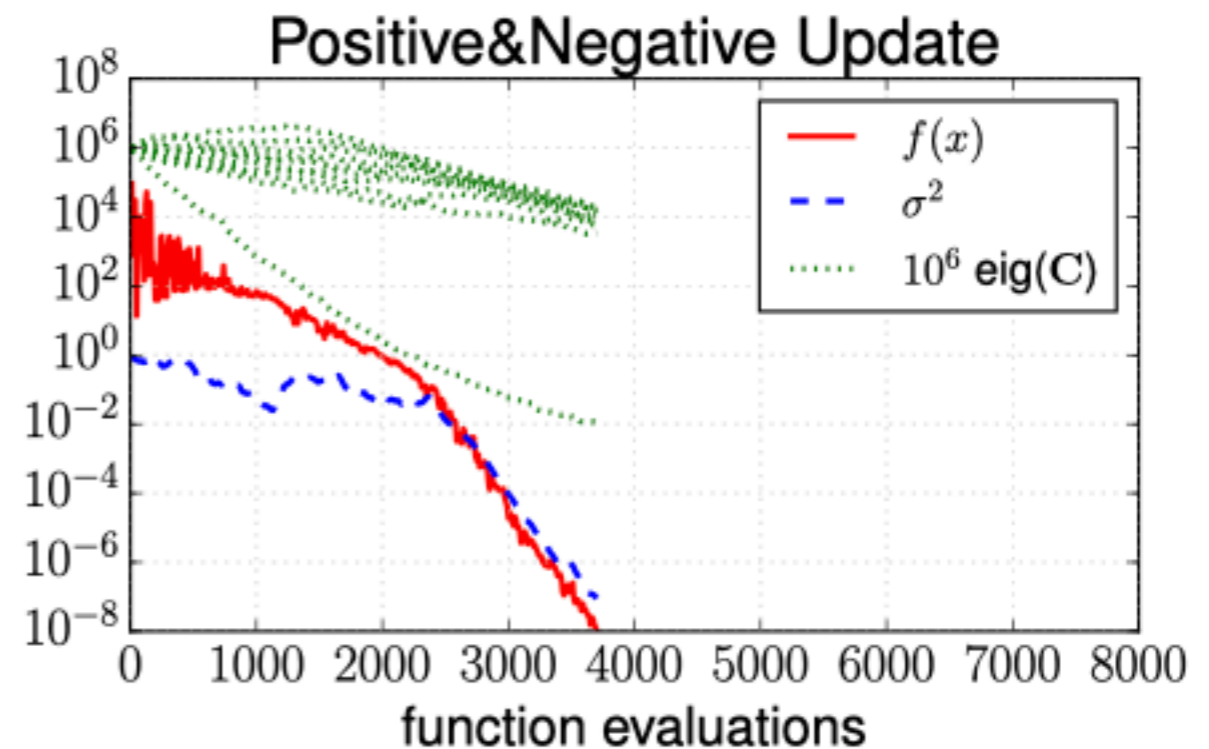
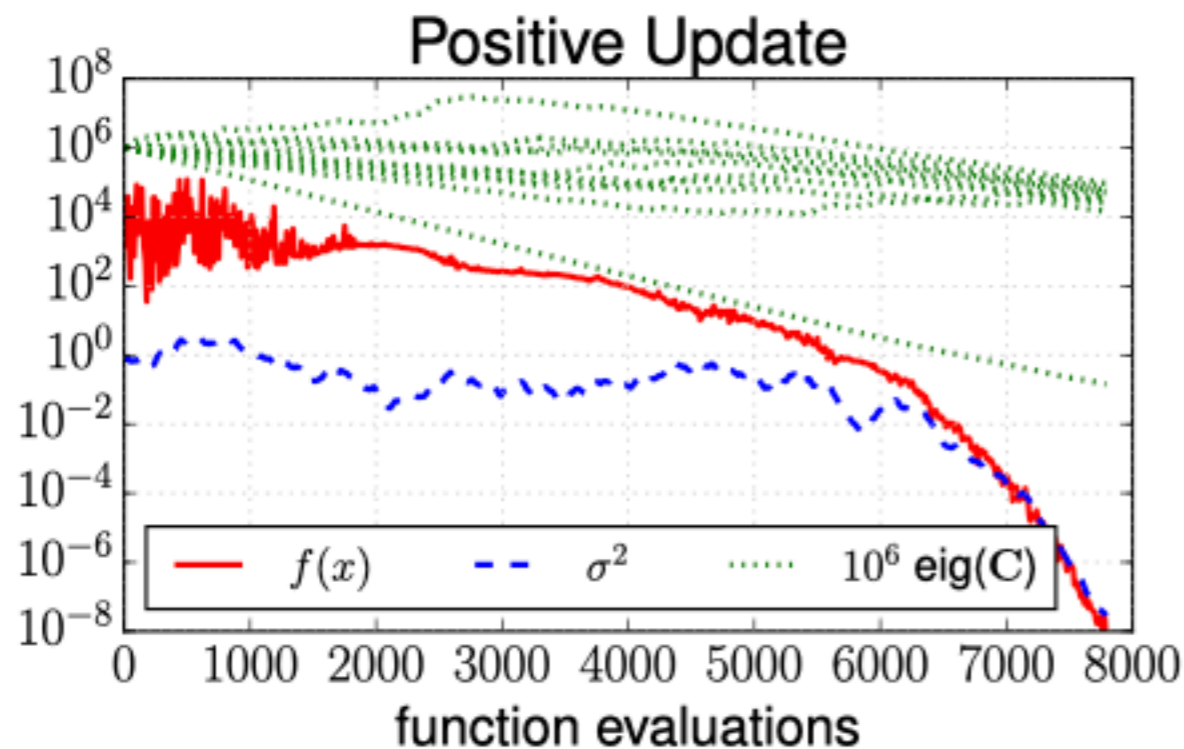
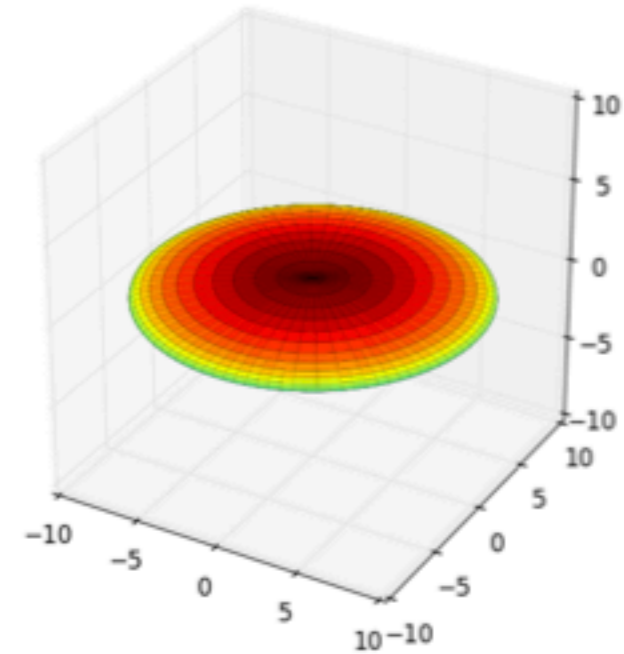
$$\mathbf{C} \leftarrow \mathbf{C} + \underbrace{c_1 \mathbf{p}_c \mathbf{p}_c^T + c_\mu \sum_{i=1}^{\lfloor \lambda/2 \rfloor} w_i \mathbf{y}_{i:\lambda} \mathbf{y}_{i:\lambda}^T}_{\text{increasing the variances in promising directions}} - \underbrace{c_\mu \sum_{i=\lambda - \lfloor \lambda/2 \rfloor + 1}^{\lambda} |w_i| \mathbf{y}_{i:\lambda} \mathbf{y}_{i:\lambda}^T}_{\text{decreasing the variances in unpromising directions}}$$

- increases the variance in the directions of \mathbf{p}_c and promising steps $\mathbf{y}_{i:\lambda}$ ($i \leq \lfloor \lambda/2 \rfloor$)
- decrease the variance in the directions of unpromising steps $\mathbf{y}_{i:\lambda}$ ($i \geq \lambda - \lfloor \lambda/2 \rfloor + 1$)
- keep the variance in the subspace orthogonal to the above

On 10D Discus Function

10D Discus Function (axis ratio: $\alpha = 10^3$)

$$f(x) = \alpha^2 \cdot x_1^2 + \sum_{i=1}^n x_i^2$$



- Positive: wait for the smallest $\text{eig}(C)$ decreasing
- Active: decrease the smallest $\text{eig}(C)$ actively

Summary

Active Covariance Matrix Adaptation + Cumulation

$$\mathbf{C} \leftarrow (1 - c_1 - c_\mu + c_\mu^-) \mathbf{C} + c_1 \mathbf{p}_c \mathbf{p}_c^T + c_\mu \sum_{i=1}^{\lfloor \lambda/2 \rfloor} w_i \mathbf{y}_{i:\lambda} \mathbf{y}_{i:\lambda}^T - c_\mu^- \sum_{i=\lambda - \lfloor \lambda/2 \rfloor + 1}^{\lambda} |w_i| \mathbf{y}_{i:\lambda} \mathbf{y}_{i:\lambda}^T$$

- $-|w_i| < 0$ (for $i \geq \lambda - \mu$): negative weight assigned to $\mathbf{y}_{i:\lambda}$, $\sum_{i=\lambda-\mu}^{\lambda} |w_i| = 1$.
- $c_\mu^- > 0$: learning rate for the active update

These components compensate each other

- cumulation: excels to learn a long axis, but inefficient for a large λ
- rank- μ update: efficient for a large λ
- active update: effective to learn short axes

An important yet solvable issue of active update

- The positive definiteness of \mathbf{C} will be violated if c_μ^- is not small enough
- The positive definiteness can be guaranteed w.p.1 by controlling $c_\mu^- w_i$

Input: $\mathbf{m} \in \mathbb{R}^n$; $\sigma \in \mathbb{R}_+$; $\lambda \in \mathbb{N}_{\geq 2}$, usually $\lambda \geq 5$, default $4 + \lfloor 3 \log n \rfloor$

Set $c_m = 1$; $c_1 \approx 2/n^2$; $c_\mu \approx \mu_w/n^2$; $c_c \approx 4/n$; $c_\sigma \approx 1/\sqrt{n}$; $d_\sigma \approx 1$; $w_{i=1\dots\lambda}$ decreasing in i and $\sum_{i=1}^{\mu} w_i = 1$, $w_\mu > 0 \geq w_{\mu+1}$, $\mu_w^{-1} := \sum_{i=1}^{\mu} w_i^2 \approx 3/\lambda$

Initialize $\mathbf{C} = \mathbf{I}$, and $\mathbf{p}_c = \mathbf{0}$, $\mathbf{p}_\sigma = \mathbf{0}$

While not terminate

$\mathbf{x}_i = \mathbf{m} + \sigma \mathbf{y}_i$, where $\mathbf{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$ for $i = 1, \dots, \lambda$ sampling

$\mathbf{m} \leftarrow \mathbf{m} + c_m \sigma \mathbf{y}_w$, where $\mathbf{y}_w = \sum_{i=1}^{\mu} w_{\text{rk}(i)} \mathbf{y}_i$ update mean

$\mathbf{p}_\sigma \leftarrow (1 - c_\sigma) \mathbf{p}_\sigma + \sqrt{1 - (1 - c_\sigma)^2} \sqrt{\mu_w} \mathbf{C}^{-\frac{1}{2}} \mathbf{y}_w$ path for σ

$\mathbf{p}_c \leftarrow (1 - c_c) \mathbf{p}_c + \mathbf{1}_{[0, 2n]} \{ \|\mathbf{p}_\sigma\|^2 \} \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w} \mathbf{y}_w$ path for \mathbf{C}

$\sigma \leftarrow \sigma \times \exp \left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|\mathbf{p}_\sigma\|}{\mathbb{E} \|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1 \right) \right)$ update of σ

$\mathbf{C} \leftarrow \mathbf{C} + c_\mu \sum_{i=1}^{\lambda} w_{\text{rk}(i)} (\mathbf{y}_i \mathbf{y}_i^\top - \mathbf{C}) + c_1 (\mathbf{p}_c \mathbf{p}_c^\top - \mathbf{C})$ update \mathbf{C}

Not covered: termination, restarts, useful output, search boundaries and encoding, corrections for: positive definiteness guaranty, \mathbf{p}_c variance loss, c_σ and d_σ for large λ

Topics

1. What makes the problem difficult to solve?

2. How does the CMA-ES work?

- Normal Distribution, Rank-Based Recombination
- Step-Size Adaptation
- Covariance Matrix Adaptation

3. What can/should the users do for the CMA-ES to work effectively on their problem?

- Choice of problem formulation and encoding (not covered)
- Choice of initial solution and initial step-size
- Restarts, Increasing Population Size
- Restricted Covariance Matrix

Default Parameter Values

CMA-ES + (B)IPOP Restart Strategy = Quasi-Parameter Free Optimizer

The following parameters were identified in carefully chosen experimental set ups.

- related to selection and recombination
 - λ : offspring number, new solutions sampled, population size
 - μ : parent number, solutions involved in mean update
 - w_i : recombination weights
- related to \mathbf{C} -update
 - $1 - c_c$: decay rate for the evolution path, cumulation factor
 - c_1 : learning rate for rank-one update of \mathbf{C}
 - c_μ : learning rate for rank- μ update of \mathbf{C}
- related to σ -update
 - $1 - c_\sigma$: decay rate of the evolution path
 - d_σ : damping for σ -change

The default values depends only on the **dimension**. They do in the first place **not depend on the objective function**.

Parameters to be set depending on the problem

Initialization and termination conditions

The following should be set or implemented depending on the problem.

- related to the initial search distribution
 - $\mathbf{m}^{(0)}$: initial mean vector
 - $\sigma^{(0)}$ (or $\sqrt{\mathbf{C}_{i,i}^{(0)}}$): initial (coordinate-wise) standard deviation
- related to stopping conditions
 - max. func. evals.
 - max. iterations
 - function value tolerance
 - min. axis length
 - stagnation

Practical Hints:

- start with an initial guess $\mathbf{m}^{(0)}$ with a relatively small step-size $\sigma^{(0)}$ to *locally* improve the current guess;
- then increase the step-size, e.g., by factor of 10, to *globally* search for a better solution.

Python CMA-ES Implementation

<https://github.com/CMA-ES/pycma>

pycma

A Python implementation of CMA-ES and a few related numerical optimization tools.

The [Covariance Matrix Adaptation Evolution Strategy \(CMA-ES\)](#) is a stochastic derivative-free numerical optimization algorithm for difficult (non-convex, ill-conditioned, multi-modal, rugged, noisy) optimization problems in continuous search spaces.

Useful links:

- [A quick start guide with a few usage examples](#)
- [The API Documentation](#)
- [Hints for how to use this \(kind of\) optimization module in practice](#)

Installation of the [latest release](#)

Type

```
python -m pip install cma
```

in a system shell to install the [latest release](#) from the [Python Package Index \(PyPI\)](#). The release link also provides more installation hints and a quick start guide.

Python CMA-ES Demo

<https://github.com/CMA-ES/pycma>

Optimizing 10D Rosenbrock Function

```
In [1]: import cma # import
opts = cma.CMAOptions() # CMA Options
opts['ftarget'] = 1e-4 # - function value target
opts['maxfevals'] = 1e6 # - max. function evaluations
cma.fmin(cma.ff.rosen, # Minimize Rosenbrock function
        x0=[0.0] * 10, # - x0 = [0,..., 0]
        sigma0=0.1, # - sigma0 = 0.1
        options=opts) # - other options
```

```
(5_w,10)-aCMA-ES (mu_w=3.2,w_1=45%) in dimension 10 (seed=909490, Mon Apr 16 13:39:57 2018)
```

Iterat	#Fevals	function value	axis ratio	sigma	min&max	std	t[m:s]
1	10	1.169928472214858e+01	1.0e+00	9.12e-02	9e-02	9e-02	0:00.0
2	20	1.363303277917634e+01	1.1e+00	8.33e-02	8e-02	8e-02	0:00.0
3	30	1.232089008099892e+01	1.2e+00	7.55e-02	7e-02	8e-02	0:00.0
100	1000	5.724977739870999e+00	9.1e+00	1.65e-02	7e-03	2e-02	0:00.1
200	2000	2.550841127554589e+00	1.5e+01	3.97e-02	1e-02	4e-02	0:00.2
300	3000	3.674986141687857e-01	1.5e+01	2.76e-02	3e-03	2e-02	0:00.4
400	4000	1.266345464781239e-03	5.0e+01	1.18e-02	8e-04	2e-02	0:00.5
420	4200	7.039461687999381e-05	5.5e+01	4.04e-03	2e-04	5e-03	0:00.5

```
termination on ftarget=0.0001 (Mon Apr 16 13:39:58 2018)
```

```
final/bestever f-value = 2.804423e-05 2.804423e-05
```

```
incumbent solution: [ 0.9998542  0.99996219  0.9999681  1.00000445  0.99998977  0.99968537
```

```
0.99954974  0.99918266 ...]
```

```
std deviations: [ 0.00023937  0.00022203  0.00024836  0.00024782  0.00031258  0.00043481
```

```
0.00078261  0.0014964 ...]
```

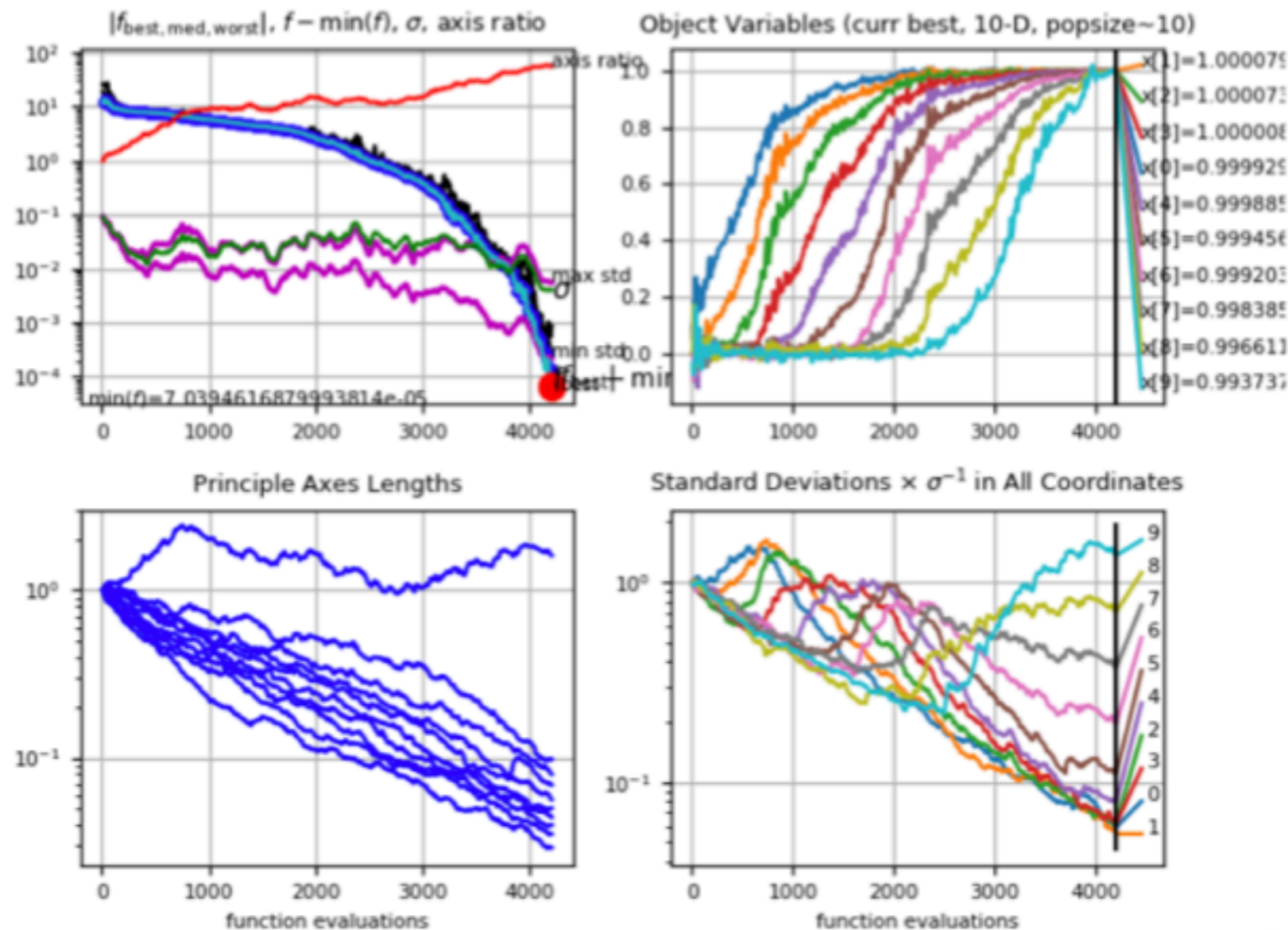
Python CMA-ES Demo

<https://github.com/CMA-ES/pycma>

Optimizing 10D Rosenbrock Function

```
cma.plot()
```

Figure 325



Multimodality

Two approaches for multimodal functions: Try again with

- a larger population size
- a smaller initial step-size (and random initial mean vector)

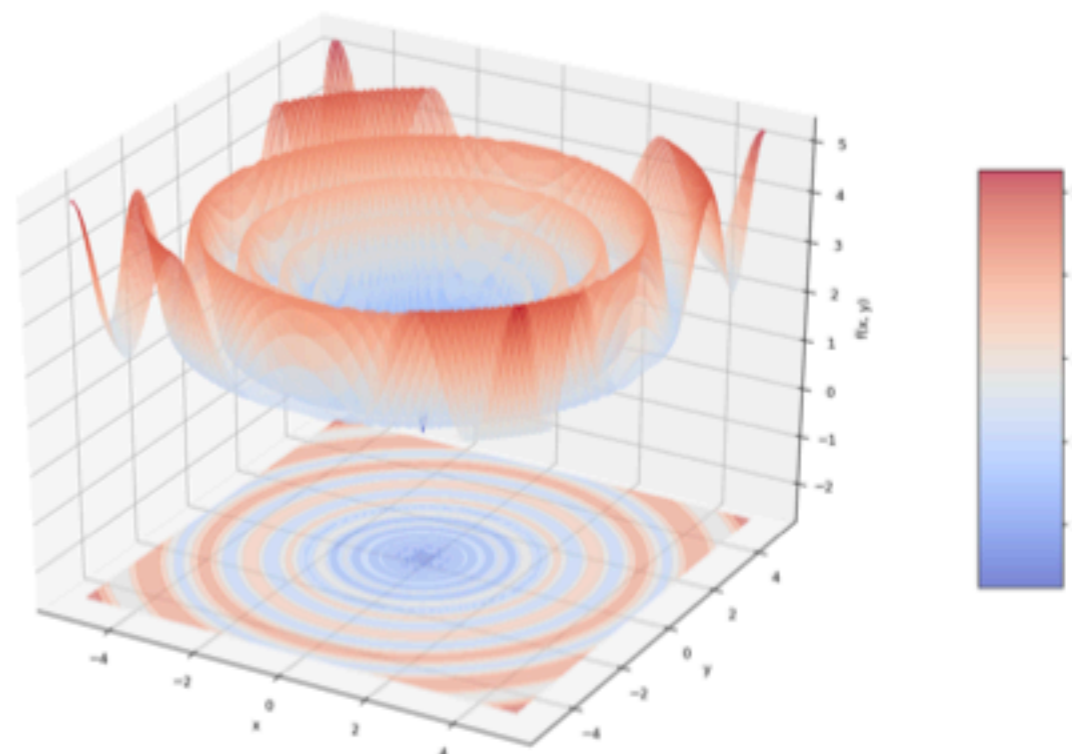
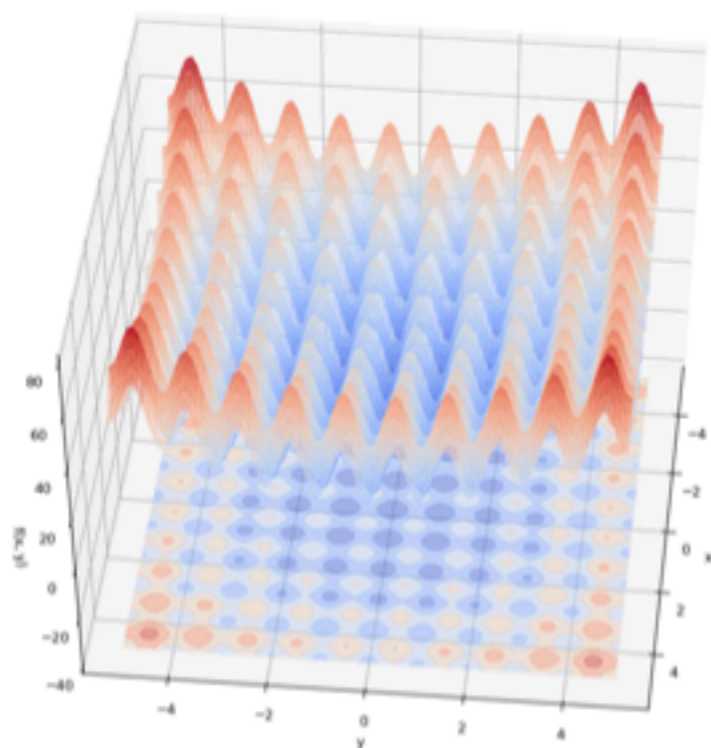
Multimodality

Approaches for multimodal functions: Try again with

- the final solution as initial solution (non-elitist) and small step-size
- a larger population size
- a different initial mean vector (and a smaller initial step-size)

A restart with a **large population size** helps if the objective function has a **well global structure**

- functions such as Schaffer, Rastrigin, BBOB function 15~19
- loosely, unimodal global structure + deterministic noise



Multimodality

Hansen and Kern. Evaluating the CMA Evolution Strategy on Multimodal Test Functions, PPSN 2004.

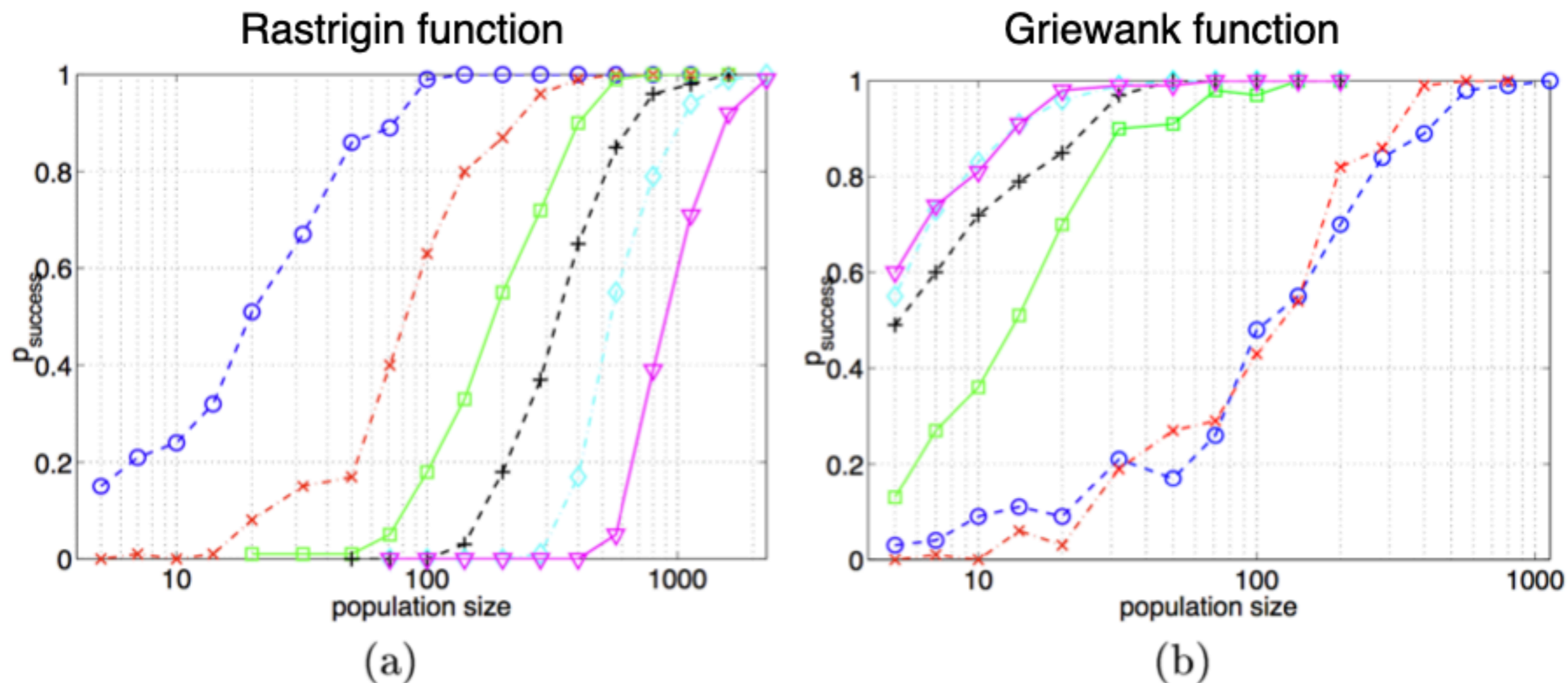


Fig. 1. Success rate to reach $f_{\text{stop}} = 10^{-10}$ versus population size for (a) Rastrigin function (b) Griewank function for dimensions $n = 2$ ('---○---'), $n = 5$ ('---×---'), $n = 10$ ('---□---'), $n = 20$ ('---+---'), $n = 40$ ('---◇---'), and $n = 80$ ('---▽---').

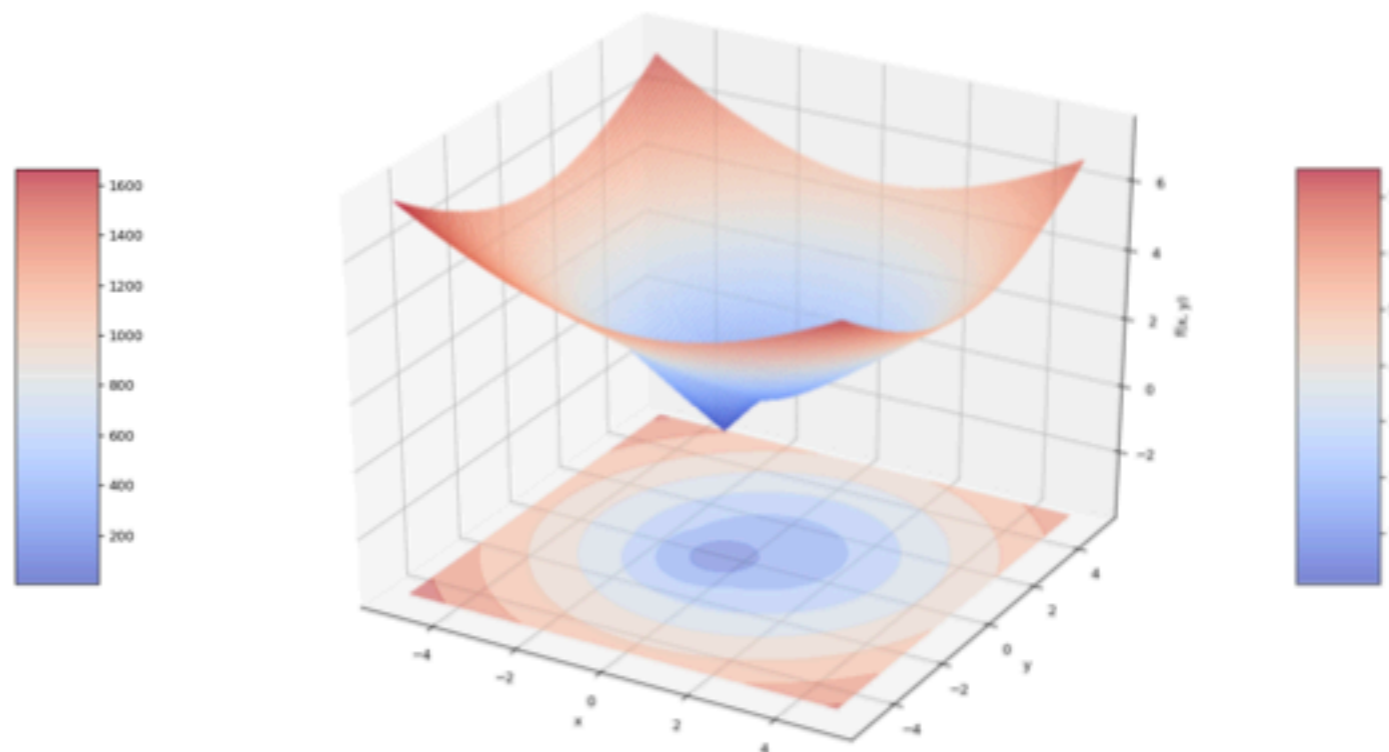
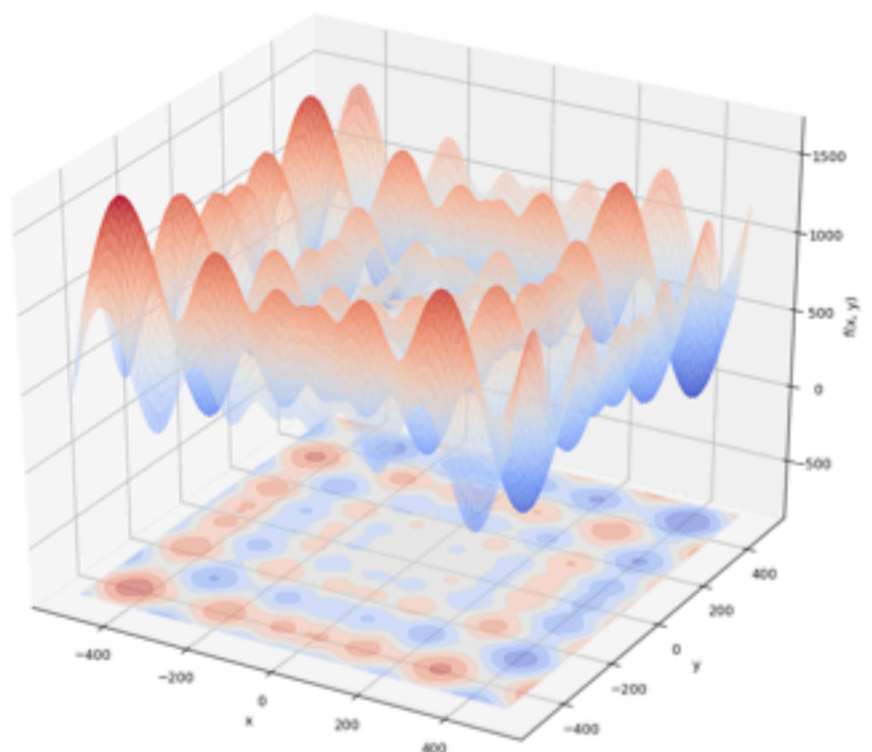
Multimodality

Approaches for multimodal functions: Try again with

- the final solution as initial solution (non-elitist) and small step-size
- a larger population size
- a different initial mean vector (and a smaller initial step-size)

A restart with a **small initial step-size** helps if the objective function has a **weak global structure**

- functions such as Schwefel, Bi-Sphere, BBOB function 20~24



a large population size can have a negative effect

Restart Strategy

It makes the CMA-ES parameter free

IPOP: Restart with increasing the population size

- start with the default population size
- double the population size after each trial (parameter sweep)
- may be considered as gold standard for automated restarts

BIPOP: IPOP regime + Local search regime

- IPOP regime: restart with increasing population size
- Local search regime: restart with a smaller step-size and a smaller population size than the IPOP regime

Topics

1. What makes the problem difficult to solve?

2. How does the CMA-ES work?

- Normal Distribution, Rank-Based Recombination
- Step-Size Adaptation
- Covariance Matrix Adaptation

3. What can/should the users do for the CMA-ES to work effectively on their problem?

- Choice of problem formulation and encoding (not covered)
- Choice of initial solution and initial step-size
- Restarts, Increasing Population Size
- Restricted Covariance Matrix

Motivation of the Restricted Covariance Matrix

Bottlenecks of the CMA-ES on high dimensional problems

- 1 $\mathcal{O}(N^2)$ Time and Space Complexities
 - ▶ to store and update $\mathbf{C} \in \mathbb{R}^{N \times N}$
 - ▶ to compute the eigen decomposition of \mathbf{C}
- 2 $\mathcal{O}(1/N^2)$ Learning Rates for \mathbf{C} -Update
 - ▶ $c_\mu \approx \mu_w / N^2$
 - ▶ $c_1 \approx 2 / N^2$

Exploit prior knowledge on the problem structure such as separability

- ⇒ decrease the degrees of freedom of the covariance matrix for
- less time and space complexities
 - a higher learning rates that potentially accelerate the adaptation

Variants with Restricted Covariance Matrix

CMA-ES Variants with Restricted Covariance Matrices

- Sep-CMA [Ros and Hansen, 2008]
 - ▶ $\mathbf{C} = \mathbf{D}$. \mathbf{D} : Diagonal
- VD-CMA [Akimoto et al., 2014]
 - ▶ $\mathbf{C} = \mathbf{D}(\mathbf{I} + \mathbf{v}\mathbf{v}^T)\mathbf{D}$. \mathbf{D} : Diagonal, $\mathbf{v} \in \mathbb{R}^N$.
- LM-CMA [Loshchilov, 2014]
 - ▶ $\mathbf{C} = \mathbf{I} + \sum_{i=1}^k \mathbf{v}_i\mathbf{v}_i^T$. $\mathbf{v}_i \in \mathbb{R}^N$.
- Vkd-CMA [Akimoto and Hansen, 2016]
 - ▶ $\mathbf{C} = \mathbf{D}(\mathbf{I} + \sum_{i=1}^k \mathbf{v}_i\mathbf{v}_i^T)\mathbf{D}$. $\mathbf{v}_i \in \mathbb{R}^N$.

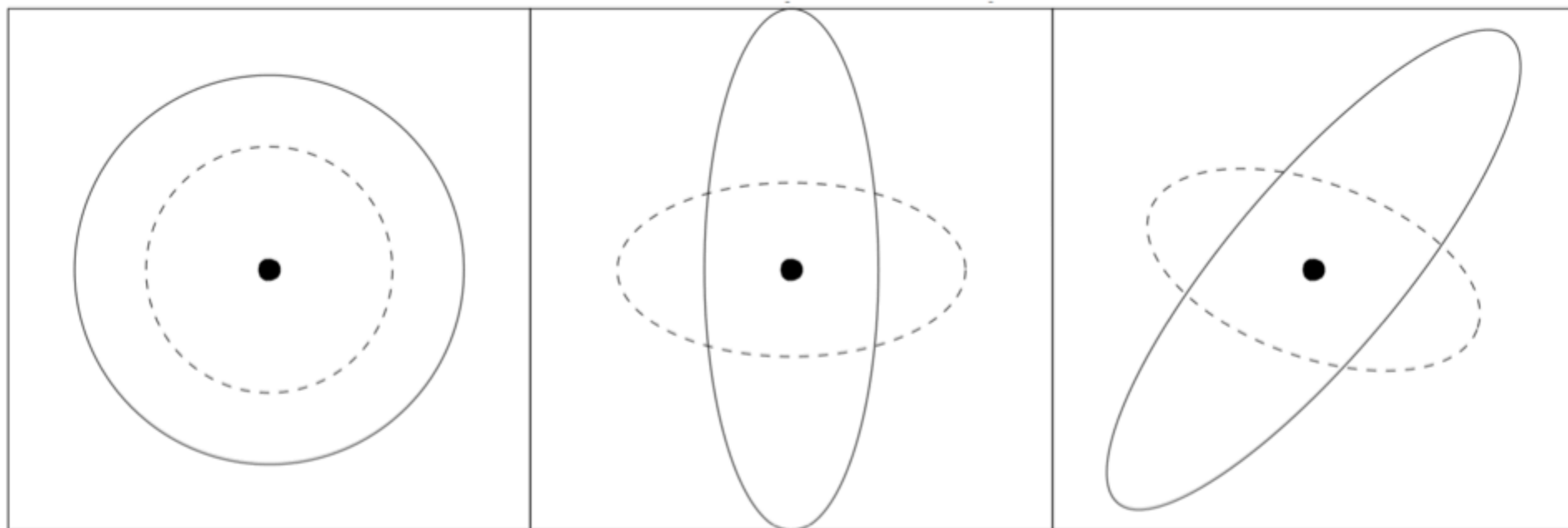
[Ros and Hansen, 2008] Ros, R. and Hansen, N. (2008). A simple modification in CMA-ES achieving linear time and space complexity. In *Parallel Problem Solving from Nature - PPSN X*, pages 296–305. Springer.

[Akimoto et al., 2014] Akimoto, Y., Auger, A., and Hansen, N. (2014). Comparison-based natural gradient optimization in high dimension. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 373–380, Vancouver, BC, Canada.

[Loshchilov, 2014] Loshchilov, I. (2014). A computationally efficient limited memory cma-es for large scale optimization. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 397–404.

[Akimoto and Hansen, 2016] Akimoto, Y. and Hansen, N. (2016). Projection-based restricted covariance matrix adaptation for high dimension. In *Genetic and Evolutionary Computation Conference, GECCO 2016, Denver, Colorado, USA, July 20-24, 2016*, page (accepted). ACM.

Separable CMA (Sep-CMA)



$\mathcal{N}(\mathbf{m}, \sigma^2 \mathbf{I}) \sim \mathbf{m} + \sigma \mathcal{N}(\mathbf{0}, \mathbf{I})$
 one degree of freedom σ

$\mathcal{N}(\mathbf{m}, \mathbf{D}^2) \sim \mathbf{m} + \mathbf{D} \mathcal{N}(\mathbf{0}, \mathbf{I})$
 n degrees of freedom

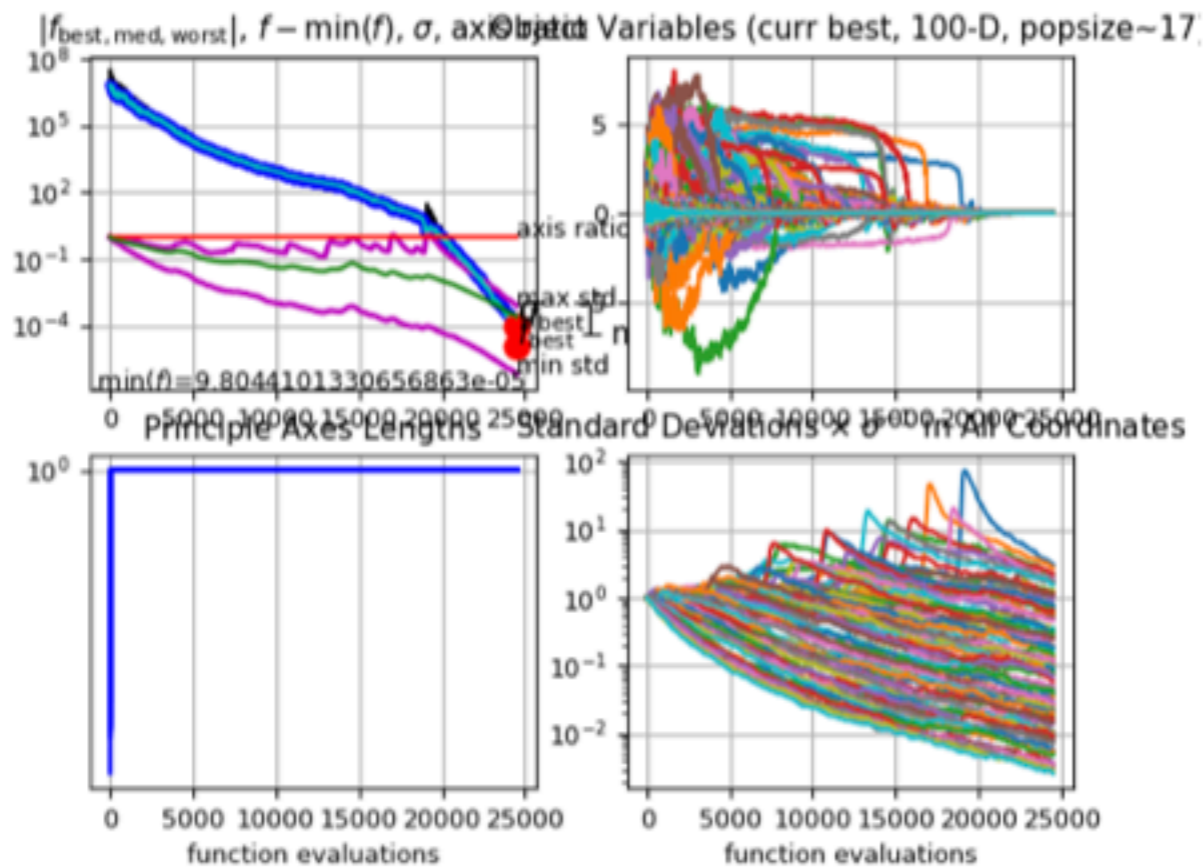
$\mathcal{N}(\mathbf{m}, \mathbf{C}) \sim \mathbf{m} + \mathbf{C}^{\frac{1}{2}} \mathcal{N}(\mathbf{0}, \mathbf{I})$
 $(n^2 + n)/2$ degrees of freedom

$$\text{CMA} \quad \mathbf{C}_{\text{cma}}^{(t+1)} = \mathbf{C}^{(t)} + c_1 (\mathbf{p}_c \mathbf{p}_c^T - \mathbf{C}^{(t)}) + c_\mu \sum_{i=1}^{\mu} w_i \left((\mathbf{x}_i - \mathbf{m}^{(t)}) (\mathbf{x}_i - \mathbf{m}^{(t)})^T - \mathbf{C}^{(t)} \right)$$

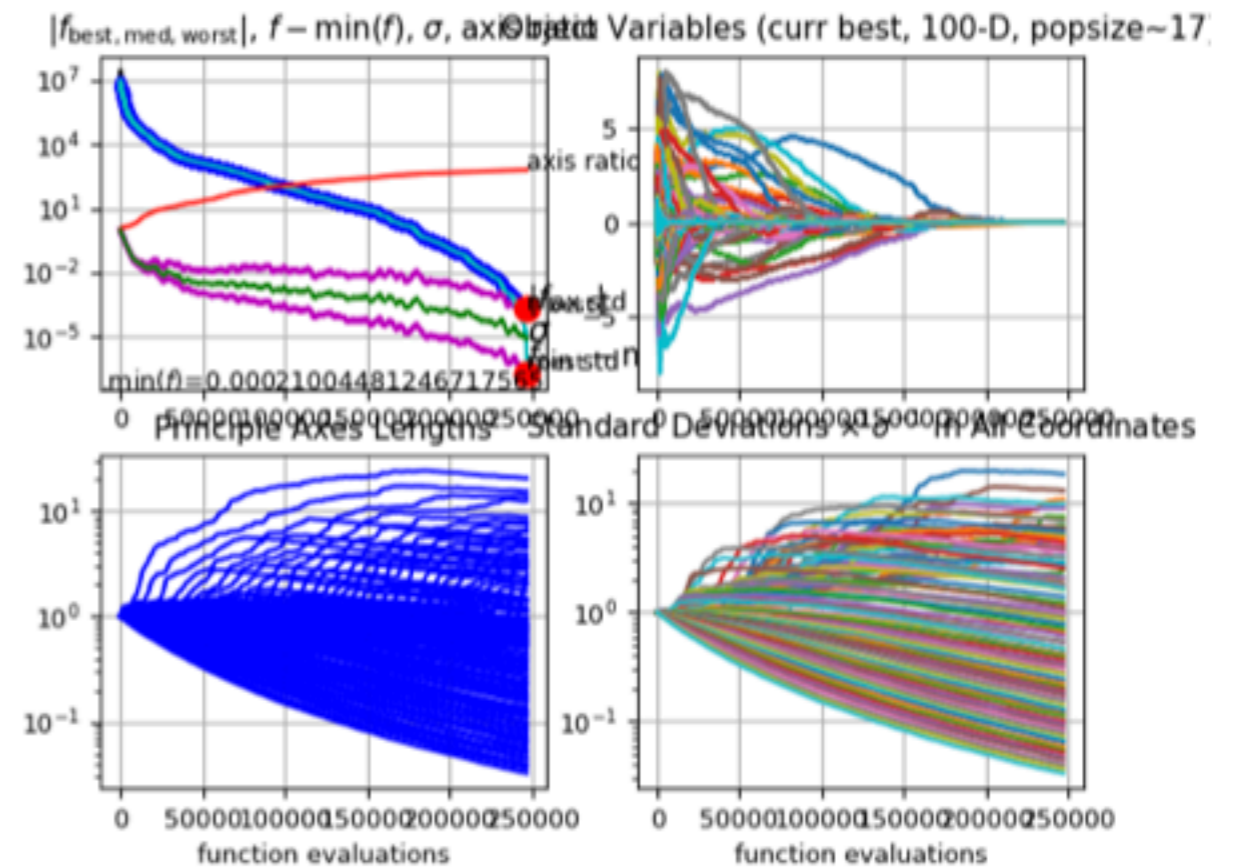
$$\text{SEP} \quad [\mathbf{C}_{\text{sep}}^{(t+1)}]_{k,k} = [\mathbf{C}^{(t)}]_{k,k} + c_1 \left([p_c]_k^2 - [\mathbf{C}^{(t)}]_{k,k} \right) + c_\mu \sum_{i=1}^{\mu} w_i \left([\mathbf{x}_i - \mathbf{m}^{(t)}]_k^2 - [\mathbf{C}^{(t)}]_{k,k} \right)$$

$(N + 2)/3$ times greater than CMA

Demo: On 100D Separable Ellipsoid Function



Separable-CMA



CMA

- CMA needed 10 times more FEs + more CPU time
- However, Sep-CMA won't be able to solve rotated ellipsoid function as efficiently as it solves separable ellipsoid

Summary and Final Remarks

Main Characteristics of (CMA) Evolution Strategies

- 1 Multivariate normal distribution to generate new search points
follows the maximum entropy principle
- 2 Rank-based selection
implies invariance, same performance on $g(f(x))$ for any increasing g
more invariance properties are featured
- 3 Step-size control facilitates fast (log-linear) convergence and possibly linear scaling with the dimension
in CMA-ES based on an **evolution path** (a non-local trajectory)
- 4 *Covariance matrix adaptation (CMA)* **increases the likelihood of previously successful steps** and can improve performance by orders of magnitude

the update follows the natural gradient
 $\mathbf{C} \propto \mathbf{H}^{-1} \iff$ adapts a variable metric
 \iff new (rotated) problem representation
 $\implies f : \mathbf{x} \mapsto g(\mathbf{x}^T \mathbf{H} \mathbf{x})$ reduces to $\mathbf{x} \mapsto \mathbf{x}^T \mathbf{x}$

Limitations

of CMA Evolution Strategies

- **internal CPU-time:** $10^{-8}n^2$ seconds per function evaluation on a 2GHz PC, tweaks are available
 - 1 000 000 f -evaluations in 100-D take 100 seconds *internal* CPU-time
 - variants with restricted covariance matrix such as Sep-CMA
- better methods are presumably available in case of
 - ▶ partly separable problems
 - ▶ specific problems, for example with cheap gradients
 - specific methods
 - ▶ small dimension ($n \ll 10$)
 - for example Nelder-Mead
 - ▶ small running times (number of f -evaluations $< 100n$)
 - model-based methods

Thank you

Source code for CMA-ES in C, C++, Java, Matlab, Octave, Python, R, Scilab
and

Practical hints for problem formulation, variable encoding, parameter setting
are available (or linked to) at

<https://cma-es.github.io/>