From Gradient-Based to Evolutionary Optimization

Nikolaus Hansen
Inria
CMAP, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris

Outline

- Why is optimization difficult?
- From gradient descent to evolution strategy
- From first order (gradient descent) to second order (variable metric): CMA-ES
- Practical advice and code examples

…feel free to ask questions…

Objective

minimize an objective function

$$f : \mathbb{R}^n \to \mathbb{R}, x \mapsto f(x)$$

- in theory: convergence to the global optimum
- in practice: find a good solution *iteratively* as quickly as possible

Objective: Important Scenarios

minimize an objective function

$$f : \mathbb{R}^n \to \mathbb{R}, x \mapsto f(x)$$

- evaluating f is expensive and/or dominates the costs

  hence quick means small number of f evaluations

- search space dimension n is large

- we can (inexpensively) evaluated the gradient of f

  $f(x) \in \mathbb{R}$, while $\nabla f(x) \in \mathbb{R}^n$

- we can parallelize the evaluations of f

Objective: Important Scenarios

minimize an objective function

$$f : \mathbb{R}^n \to \mathbb{R}, \; x \mapsto f(x)$$

- evaluating f is expensive and/or dominates the costs

  hence quick means small number of f evaluations

- search space dimension n is large
- we can (inexpensively) evaluated the gradient of f

  $f(x) \in \mathbb{R}$, while $\nabla f(x) \in \mathbb{R}^n$

- we can parallelize the evaluations of f

Objective

minimize an objective function

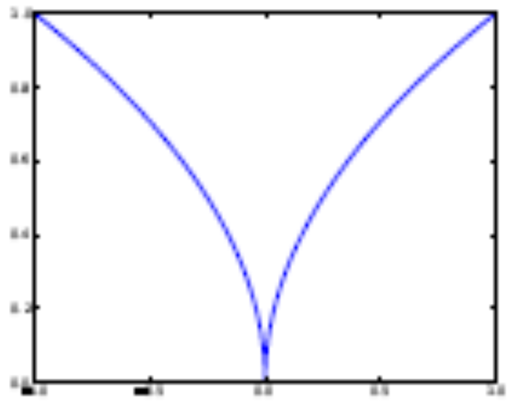
$$f : \mathbb{R}^n \to \mathbb{R},\ x \mapsto f(x)$$

What Makes an Optimization Problem Difficult?

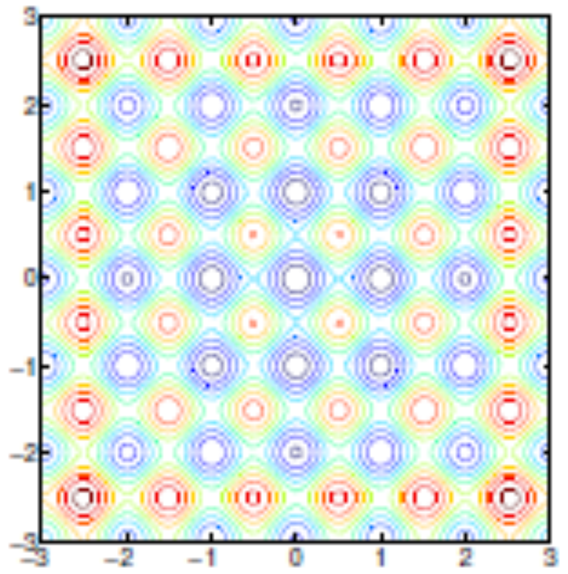
What Makes an Optimization Problem Difficult?

- non-linear, non-quadratic

on linear and quadratic functions
specialized search policies are available



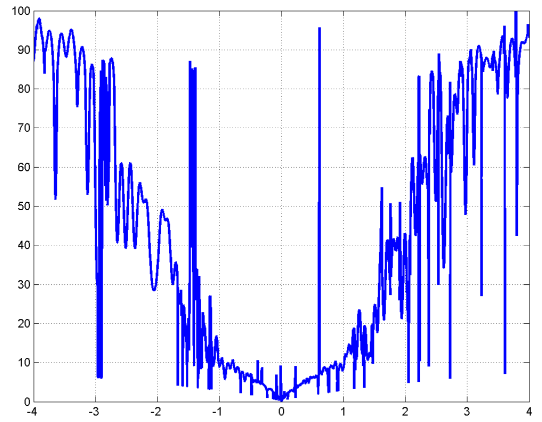
- non-convexity



- dimensionality (size of search space) and non-separability

dimension considerably larger than three with
dependencies between the variables

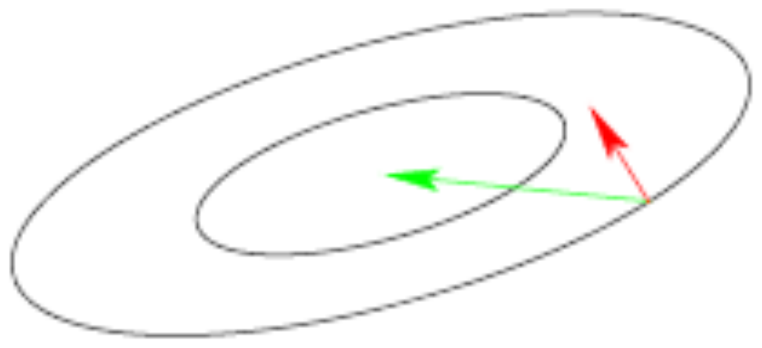
- multimodality



- ruggedness

high frequency modality, non-smooth, discontinuous

- ill-conditioning



varying sensitivities,
worst case: non-smooth concave level sets

gradient direction Newton direction

In any case, the objective function must be highly regular

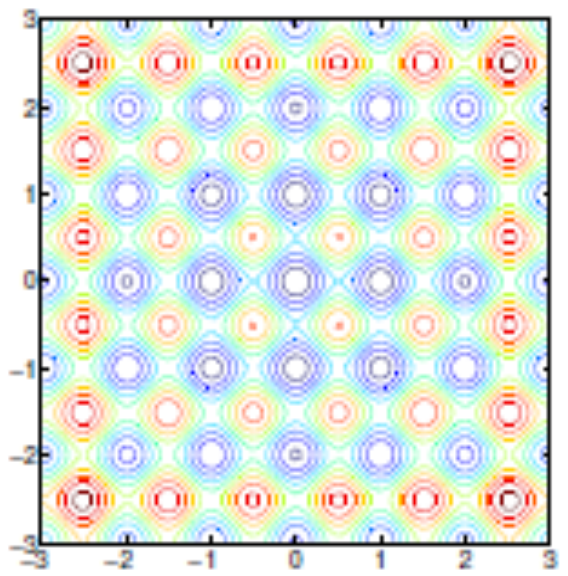
dimensionality: On Separable Functions

- Separable functions: for all $x \in \mathbb{R}^n$, for all i,

$$\arg\min_{x_i} f(x) \text{ is independent of } x$$

- Additively decomposable functions:

$$f(x) = \sum_{i=1}^{n} g_i(x_i), \quad x = (x_1, \ldots, x_n)$$



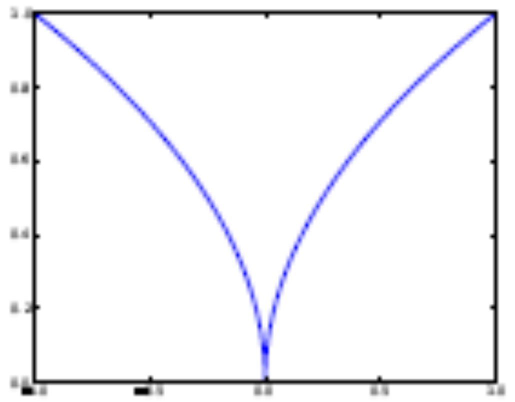
are separable

can be solved with n one-dimensional optimizations

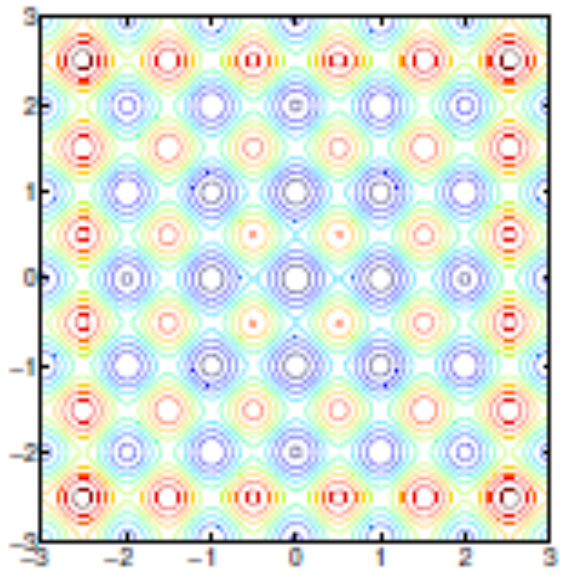
What Makes an Optimization Problem Difficult?

- non-linear, non-quadratic

on linear and quadratic functions
specialized search policies are available



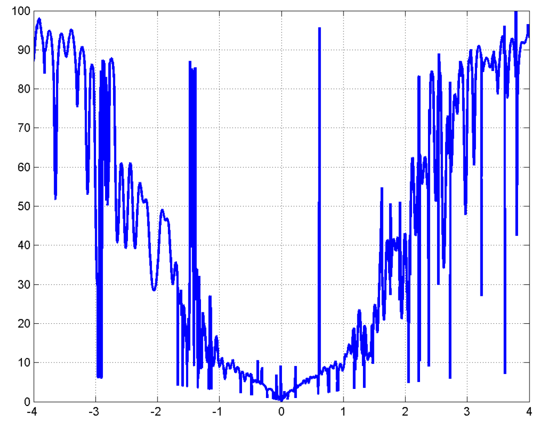
- non-convexity



- dimensionality (size of search space) and non-separability

dimension considerably larger than three with
dependencies between the variables

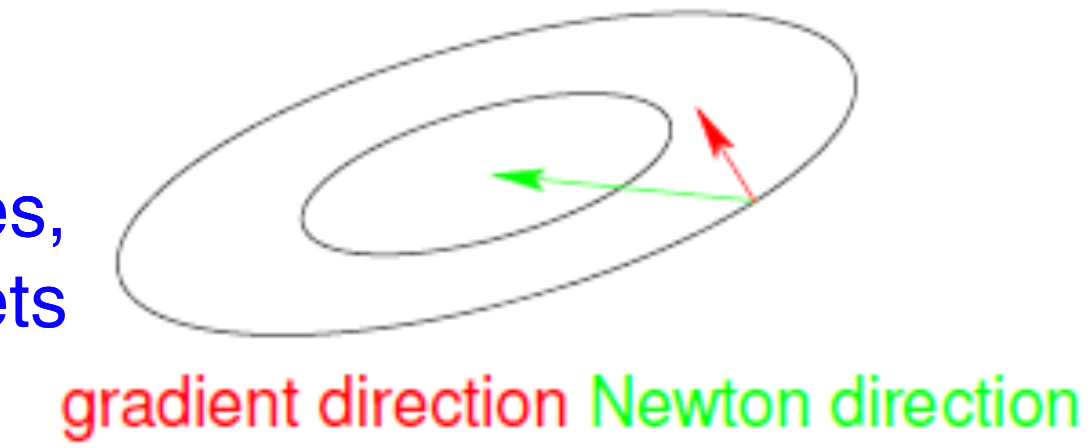
- multimodality



- ruggedness

high frequency modality, non-smooth, discontinuous

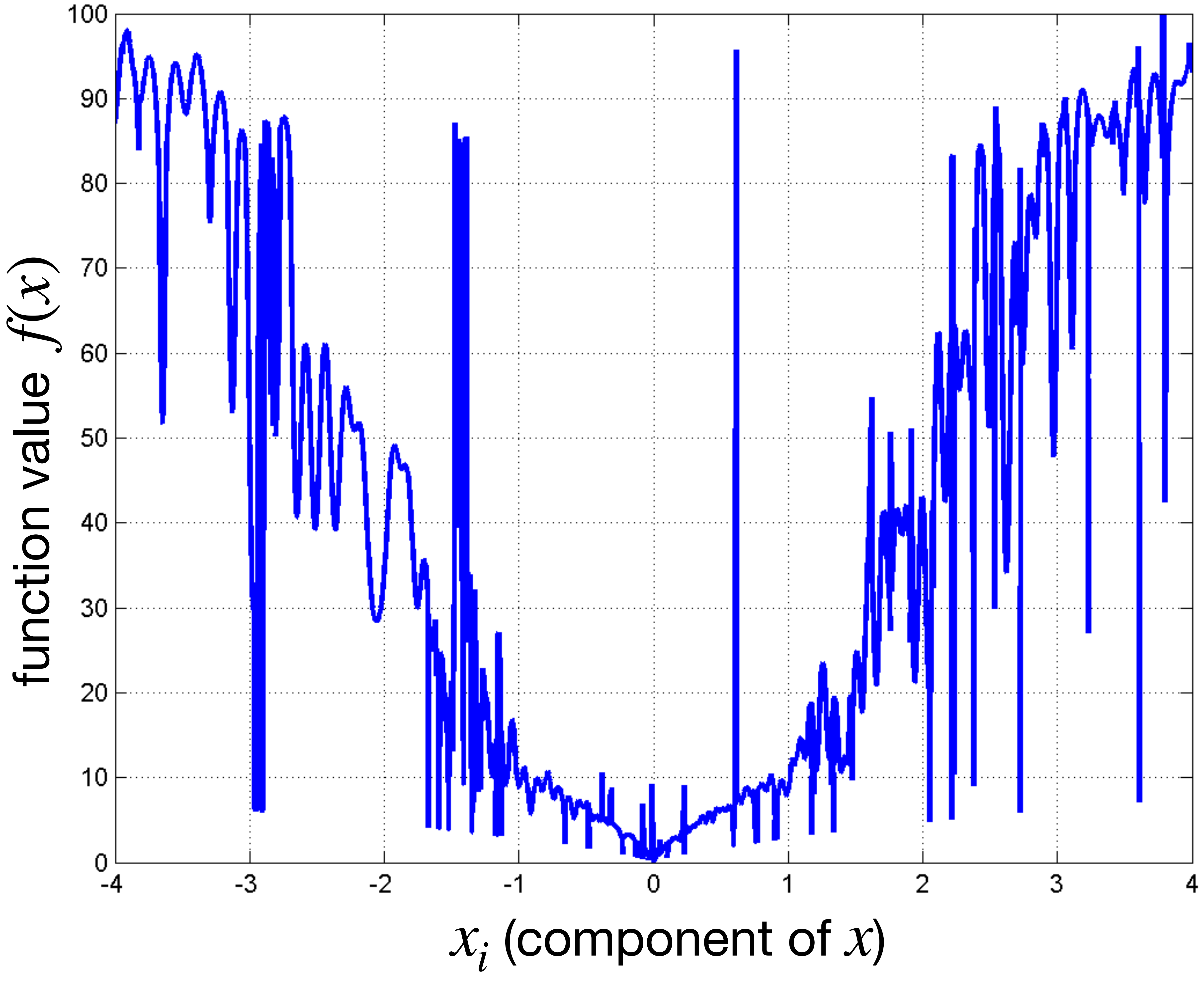
- ill-conditioning



varying sensitivities,
worst case: non-smooth concave level sets

In any case, the objective function must be highly regular

Section Through a 5-Dimensional Rugged Landscape

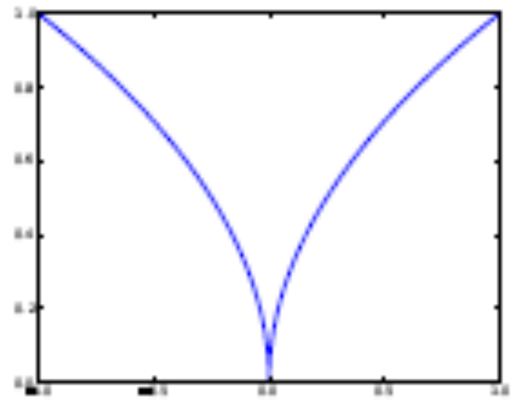


$$f : \mathbb{R}^n \to \mathbb{R}, x \mapsto f(x), n = 5$$

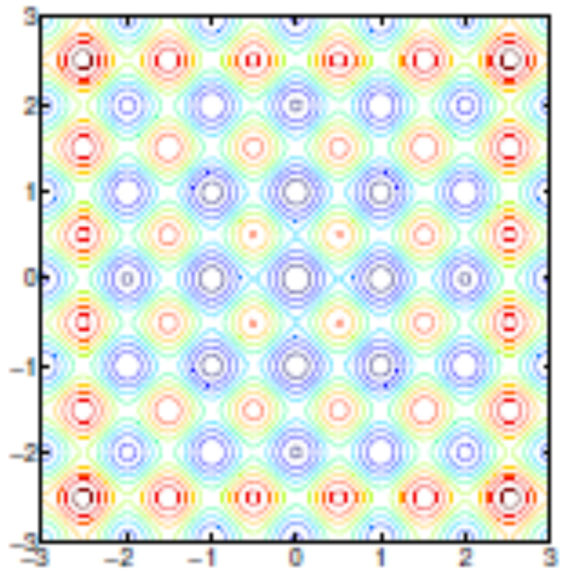
What Makes an Optimization Problem Difficult?

- non-linear, non-quadratic

on linear and quadratic functions
specialized search policies are available



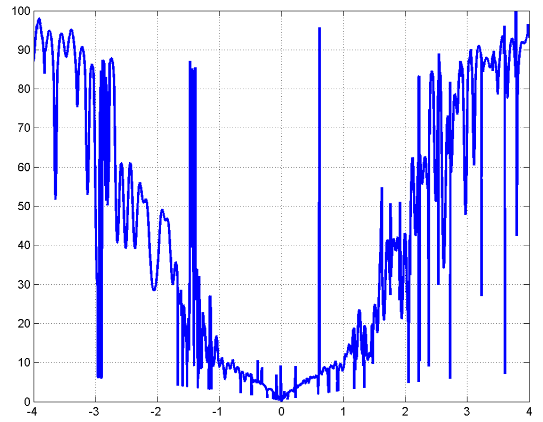
- non-convexity



- dimensionality (size of search space) and non-separability

dimension considerably larger than three with
dependencies between the variables

- multimodality

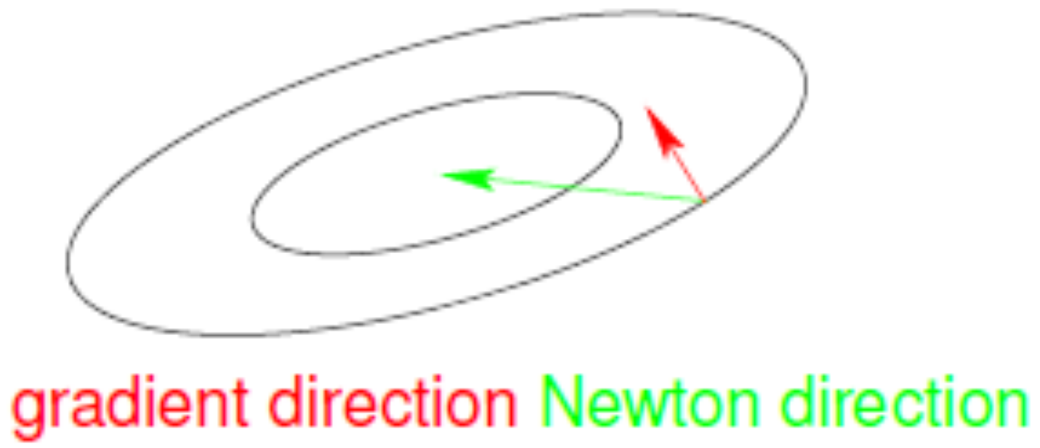


- ruggedness

high frequency modality, non-smooth, discontinuous

- ill-conditioning

varying sensitivities
worst case: non-smooth concave level sets



In any case, the objective function must be highly regular

Flexible Muscle-Based Locomotion for Bipedal Creatures

SIGGRAPH ASIA 2013

Thomas Geijtenbeek
Michiel van de Panne
Frank van der Stappen

Flexible Muscle-Based Locomotion for Bipedal Creatures
T. Geijtenbeek, M van de Panne, F van der Stappen
https://youtu.be/pgaEE27nsQw

Landscape of Continuous Search Methods

Gradient-based (Taylor, local)

- Conjugate gradient methods [Fletcher & Reeves 1964]
- Quasi-Newton methods (BFGS) [Broyden et al 1970]

Derivative-free optimization (DFO)

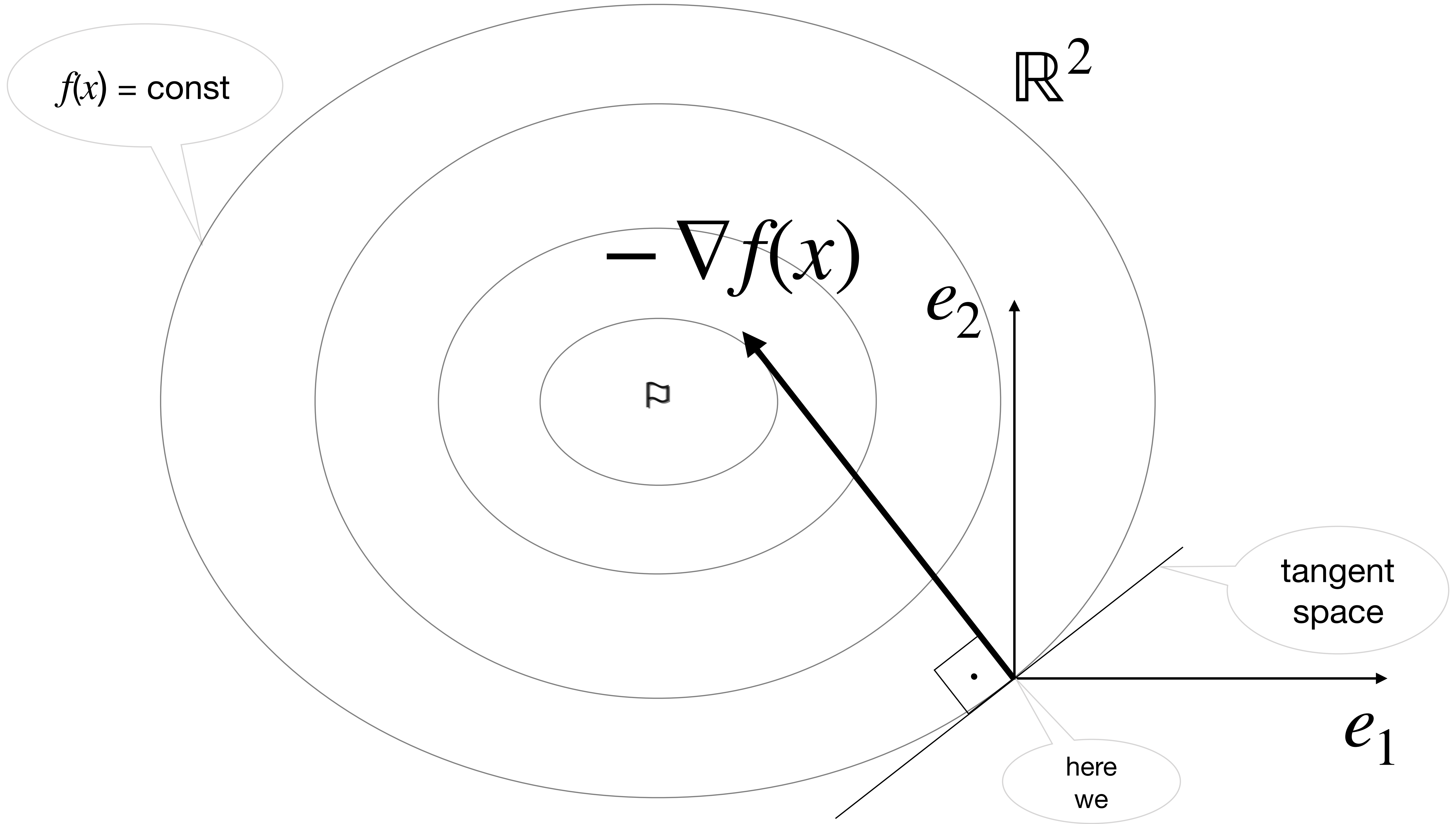
- Trust-region methods (NEWUOA, BOBYQA) [Powell 2006, 2009]
- Simplex downhill [Nelder & Mead 1965]
- Pattern search [Hooke & Jeeves 1961, Audet & Dennis 2006]

Stochastic (randomized) search methods

- Evolutionary algorithms (broader sense, continuous domain)
  - Differential Evolution [Storn & Price 1997]
  - Particle Swarm Optimization [Kennedy & Eberhart 1995]
  - **Evolution Strategies** [Rechenberg 1965, Hansen & Ostermeier 2001]
- Simulated annealing [Kirkpatrick et al 1983]
- Simultaneous perturbation stochastic approximation (SPSA) [Spall 2000]

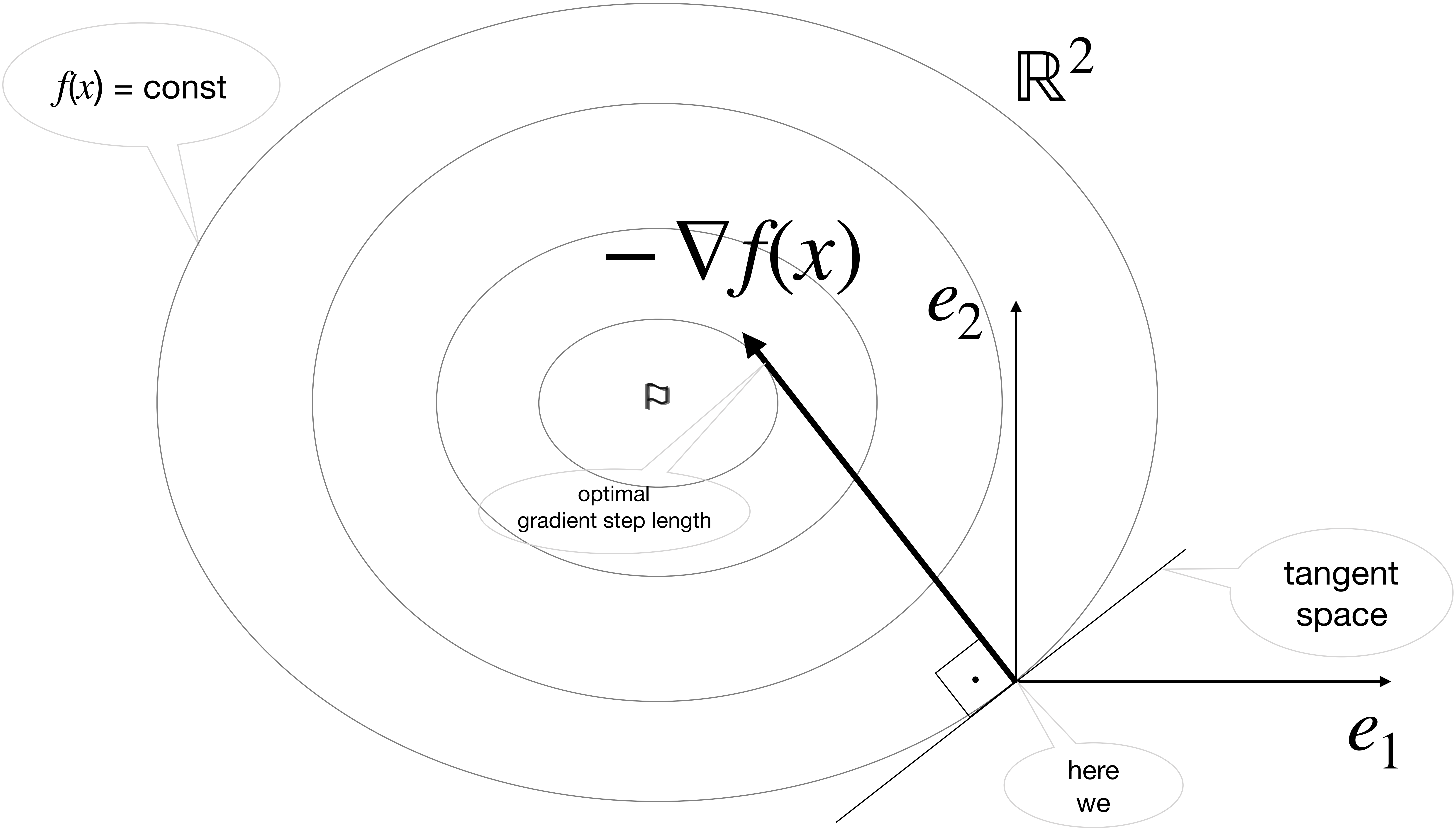
Basic Approach: Gradient Descent

The *gradient* is the local direction of the maximal f increase



Basic Approach: Gradient Descent

The *gradient* is the local direction of the maximal f increase



Basic Approach: Gradient Descent

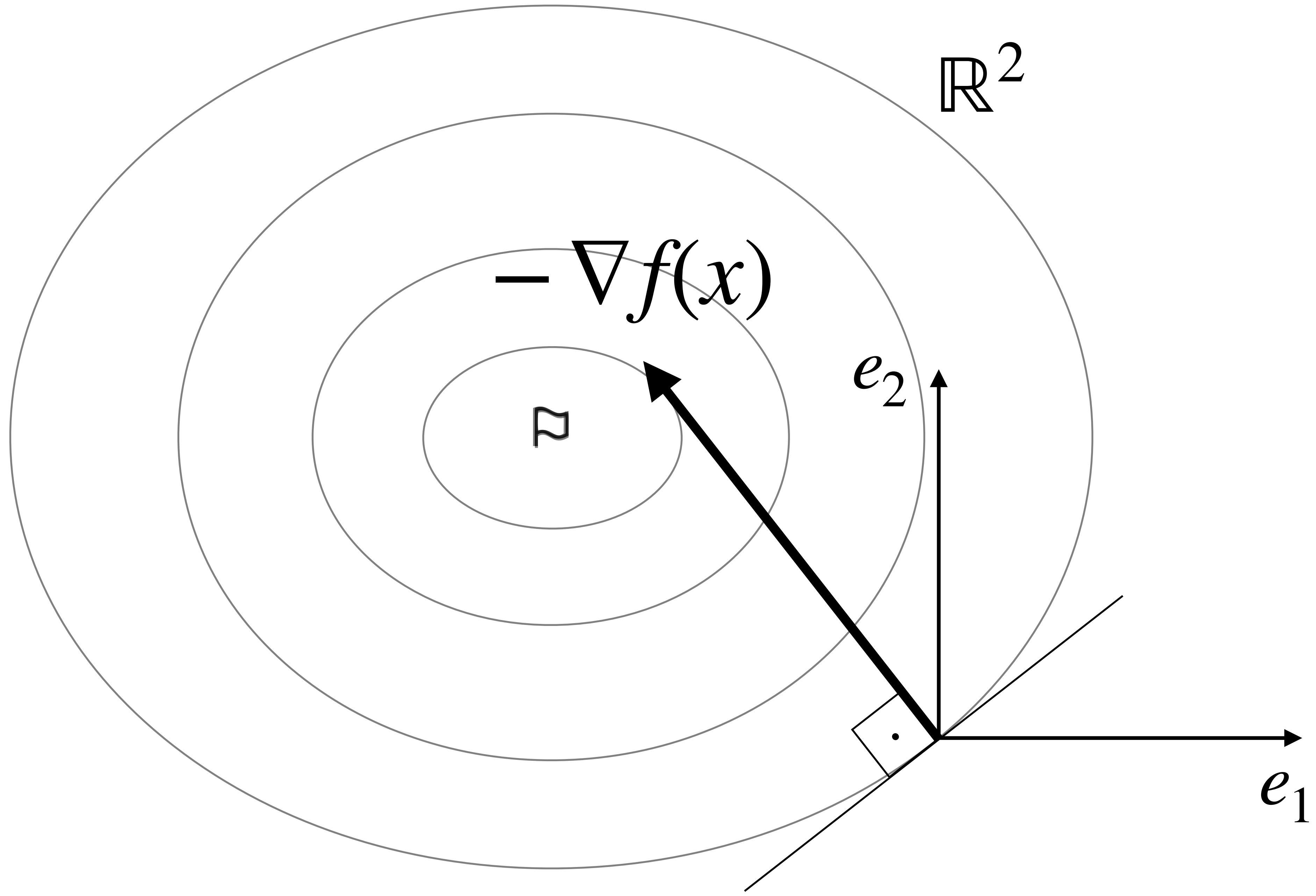
The *gradient* is the local direction of the maximal f increase

small test step

$$\nabla f(x) = -\sum_{i=1}^{n} w_i e_i \qquad -w_i = lim_{\delta\to 0}\frac{f(x+\delta e_i) - f(x)}{\delta}$$

$$x \leftarrow x - \sigma \nabla f(x)$$

$$= x + \sigma \sum_{i=1}^{n} w_i e_i$$



Basic Approach: Gradient Descent

The *gradient* is the local direction of the maximal f increase

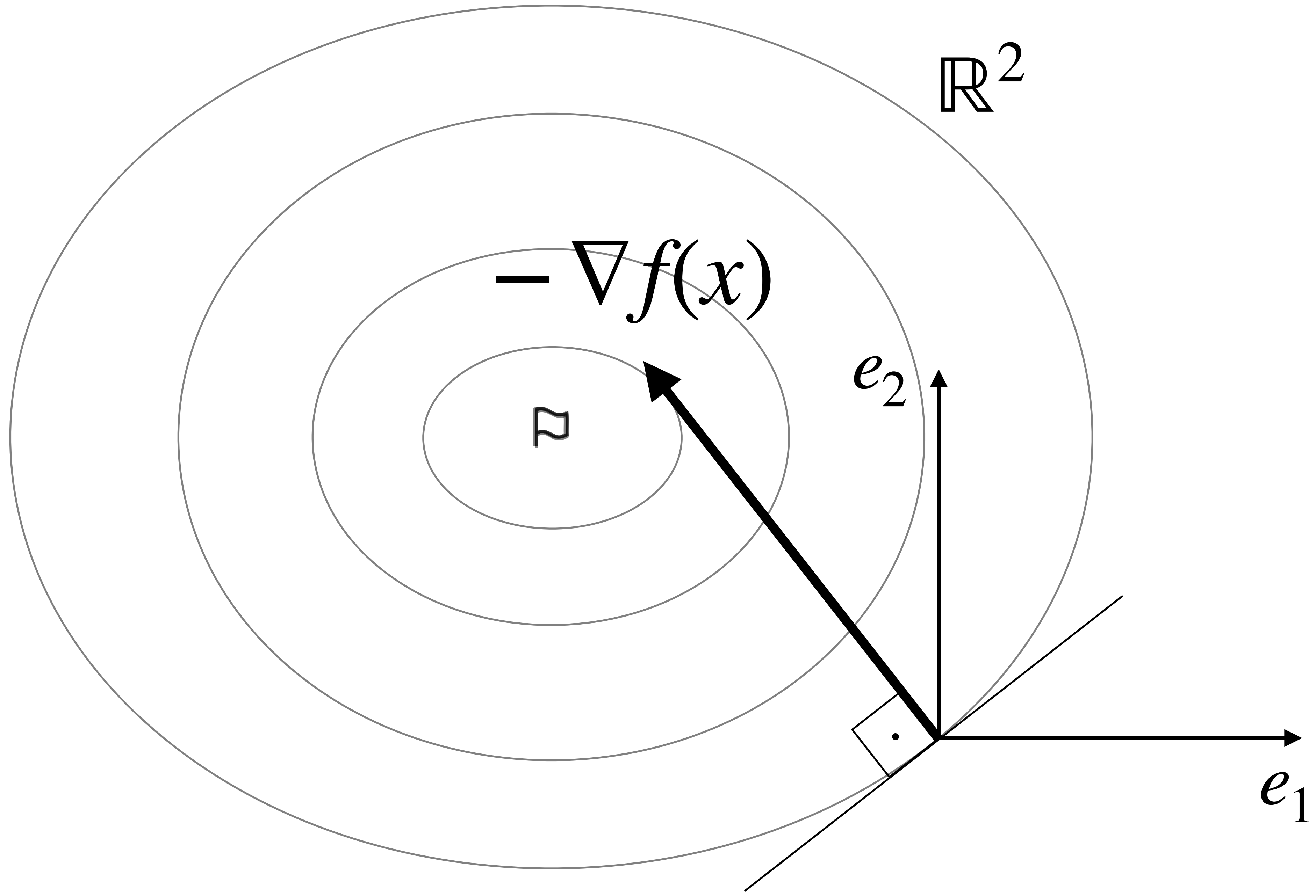
partial derivative $\frac{\partial f}{\partial x_i}(x)$

small test step

$$\nabla f(x) = -\sum_{i=1}^{n} w_i e_i \qquad -w_i = lim_{\delta \to 0} \frac{f(x+\delta e_i) - f(x)}{\delta}$$

$$x \leftarrow x - \sigma \nabla f(x)$$

$$= x + \sigma \sum_{i=1}^{n} w_i e_i$$



Basic Approach: Gradient Descent

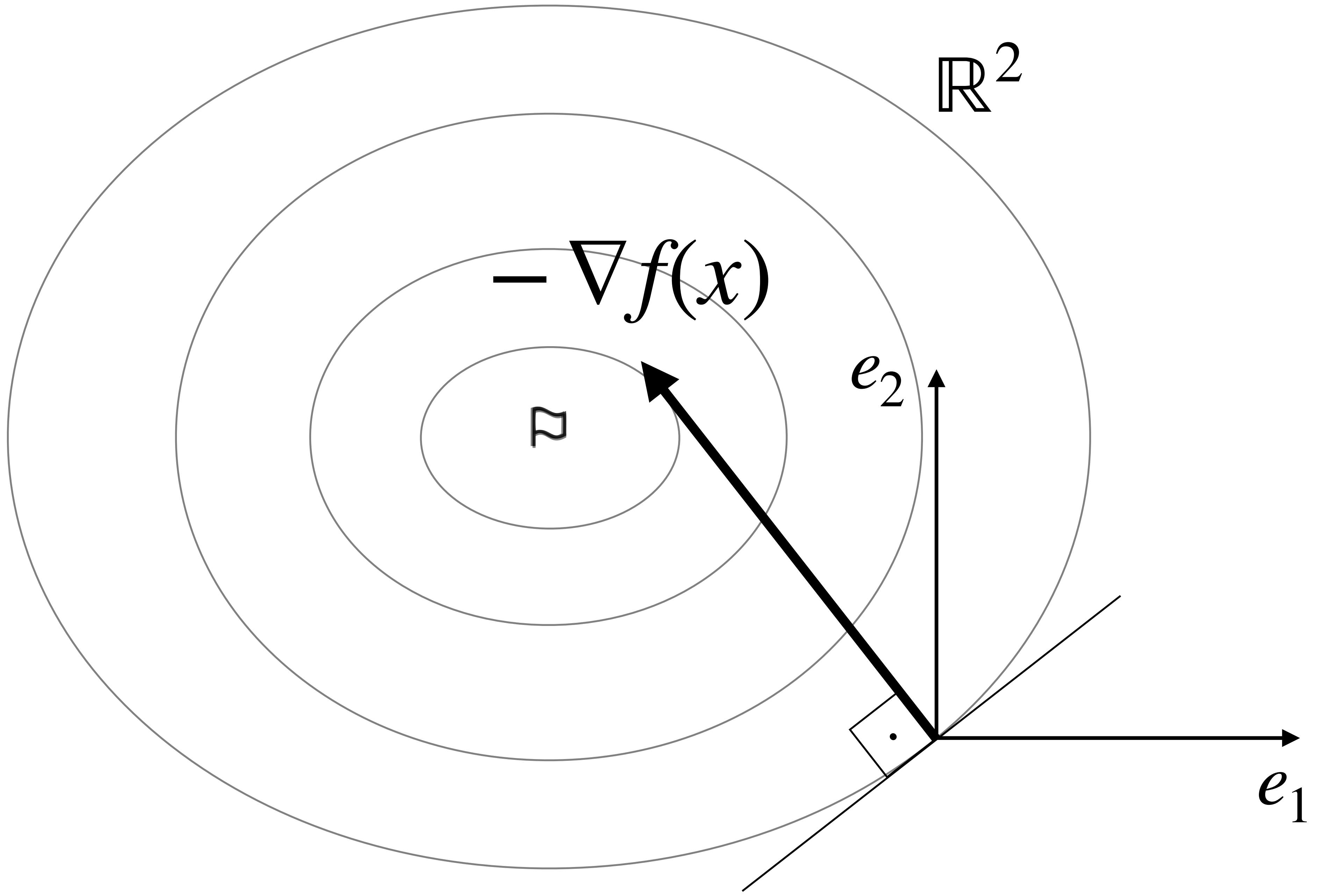
The *gradient* is the local direction of the maximal f increase

small test step

$$\nabla f(x) \approx -\sum_{i=1}^{n} w_i e_i \qquad -w_i = \cancel{\lim_{\delta \to 0}} \frac{f(x + \delta e_i) - f(x)}{\delta}$$

$$x \leftarrow x - \sigma \nabla f(x)$$

$$\approx x + \sigma \sum_{i=1}^{n} w_i e_i$$



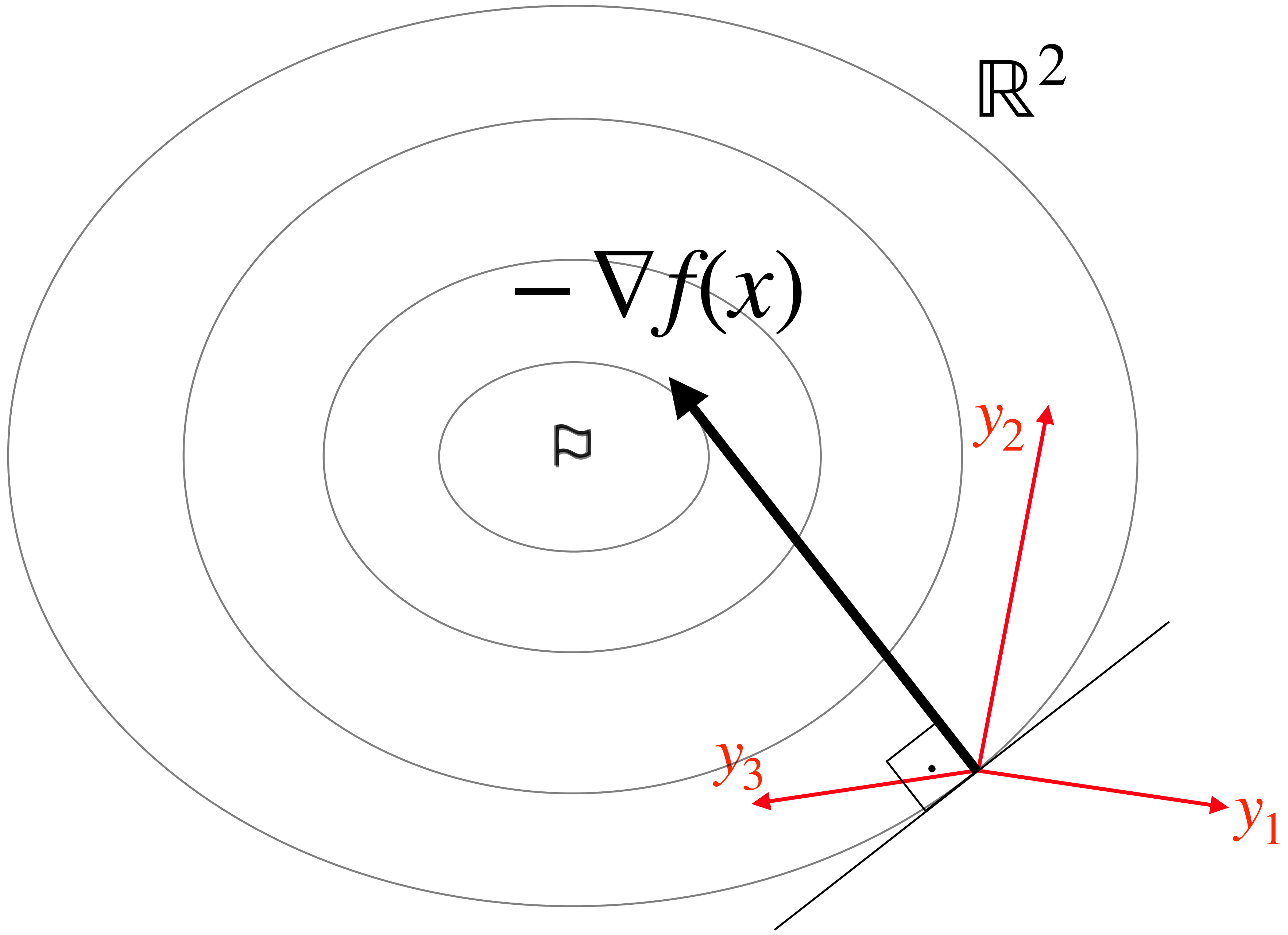
Basic Approach: Approximated Gradient Descent

We modify the gradient equation…

$$\nabla f(x) \approx -\sum_{i=1}^{m} w_i y_i \qquad -w_i = \cancel{\lim_{\delta \to 0}} \frac{f(x + \delta y_i) - f(x)}{\delta}$$

$$x \leftarrow x - \sigma \nabla f(x)$$

$$\approx x + \sigma \sum_{i=1}^{m} w_i y_i$$



Basic Approach: Approximated Gradient Descent

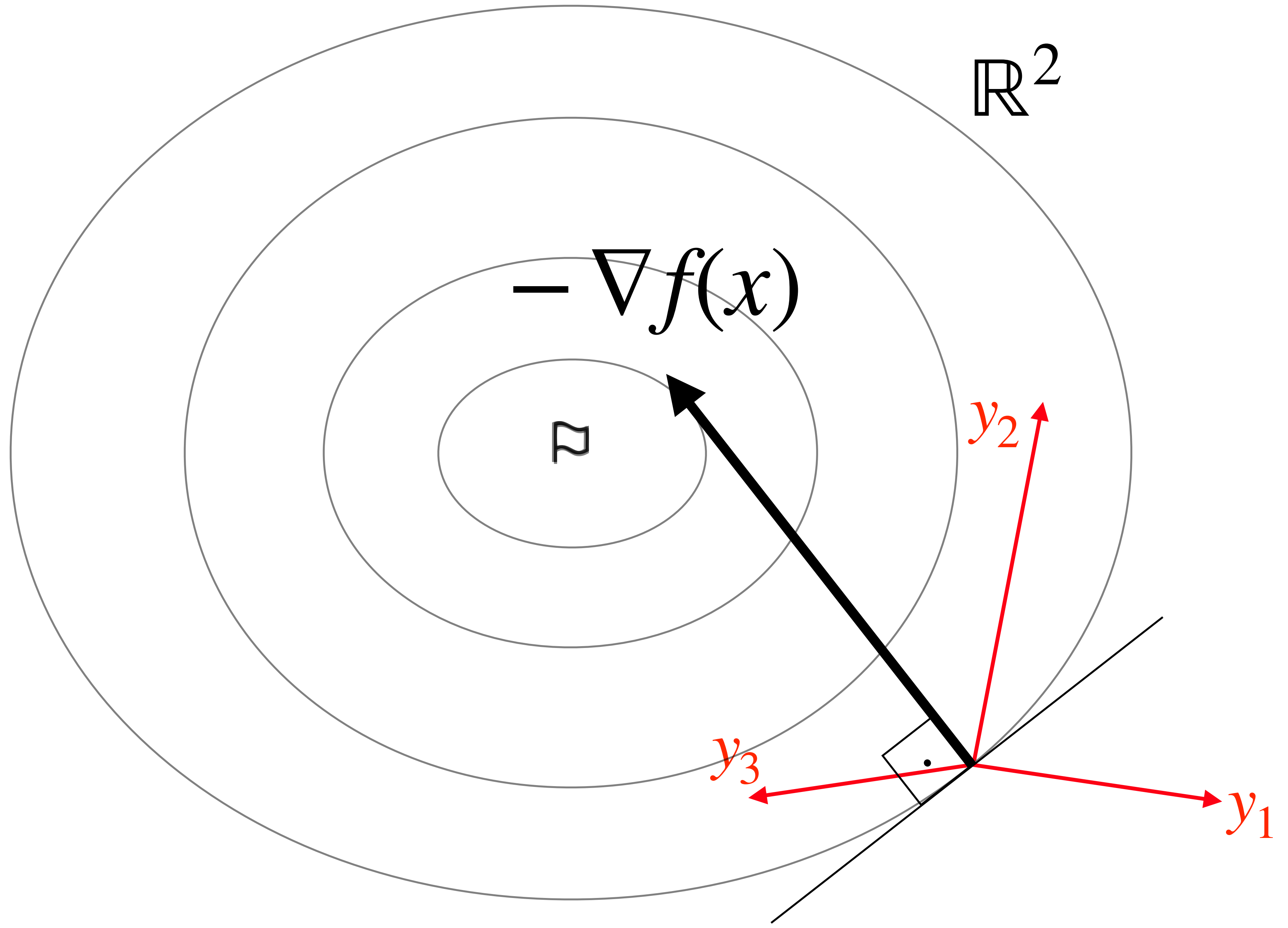
We modify the gradient equation…

$$y_i \sim \mathcal{N}(0, I) \qquad -w_i = \cancel{\lim_{\delta\to 0}} \frac{f(x+\delta y_i) - f(x)}{\delta}$$

~~small~~ test step

$$x \leftarrow x - \sigma \nabla f(x)$$

$$\approx x + \sigma \sum_{i=1}^{m} w_i y_i$$



Evolutionary Gradient Search (EGS) [Salmon 1998, Arnold & Salomon 2007]

Rank-Based Approximated Gradient Descent

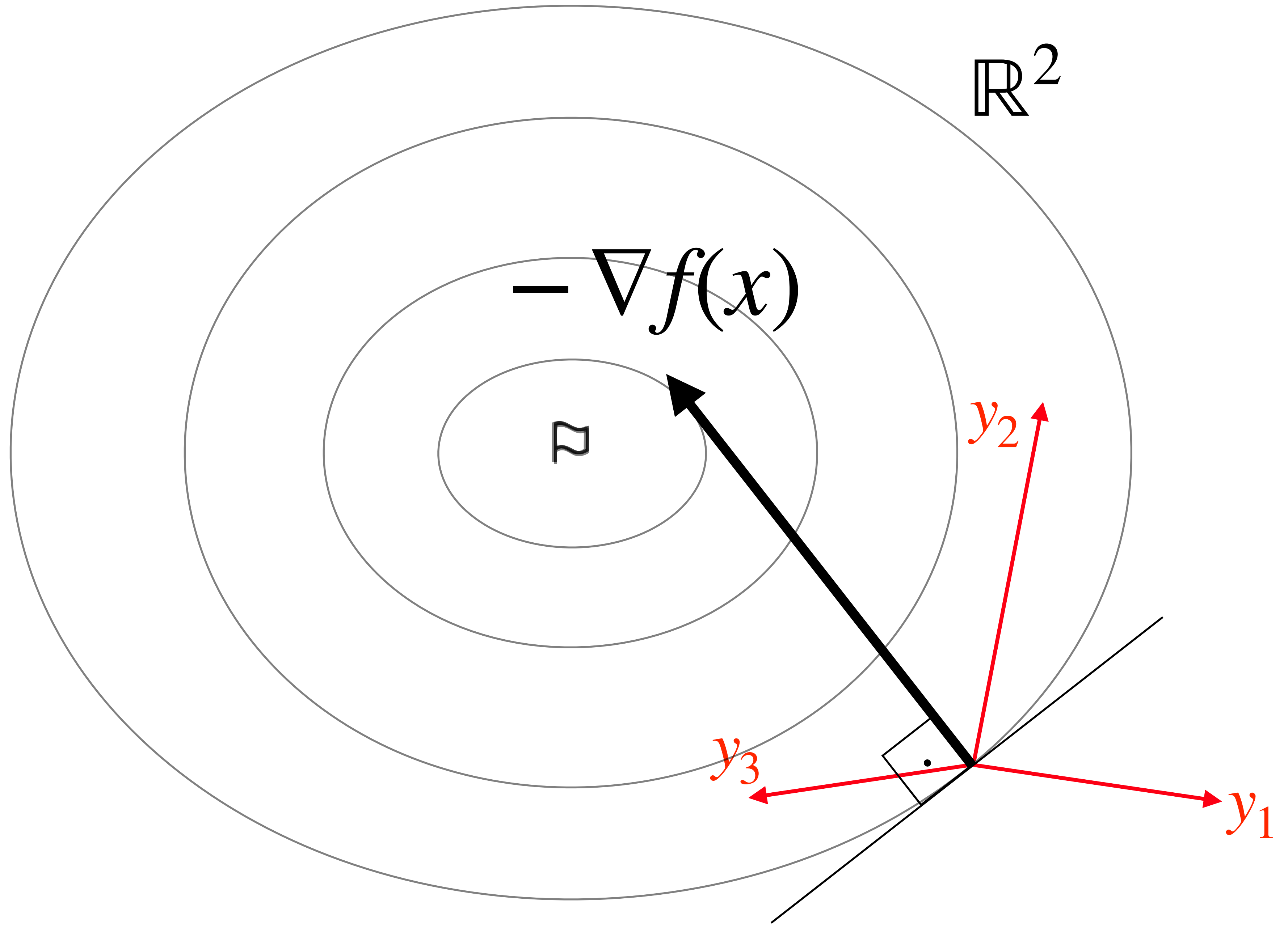
Using ranks introduces invariance to order-preserving f-transformations.

~~small~~ test step

$$y_i \sim \mathcal{N}(0, I) \qquad\qquad -w_i \propto \textbf{rank}_i(f(x + \delta y_i)) - \frac{m+1}{2}$$

$$x \leftarrow x - \sigma \nabla f(x)$$

$$\approx x + \sigma \sum_{i=1}^{m} w_i y_i$$



Evolution Strategy (ES) [Rechenberg 1973, Schwefel 1981, Rudolph 1997]

Rank-Based Approximated Gradient Descent

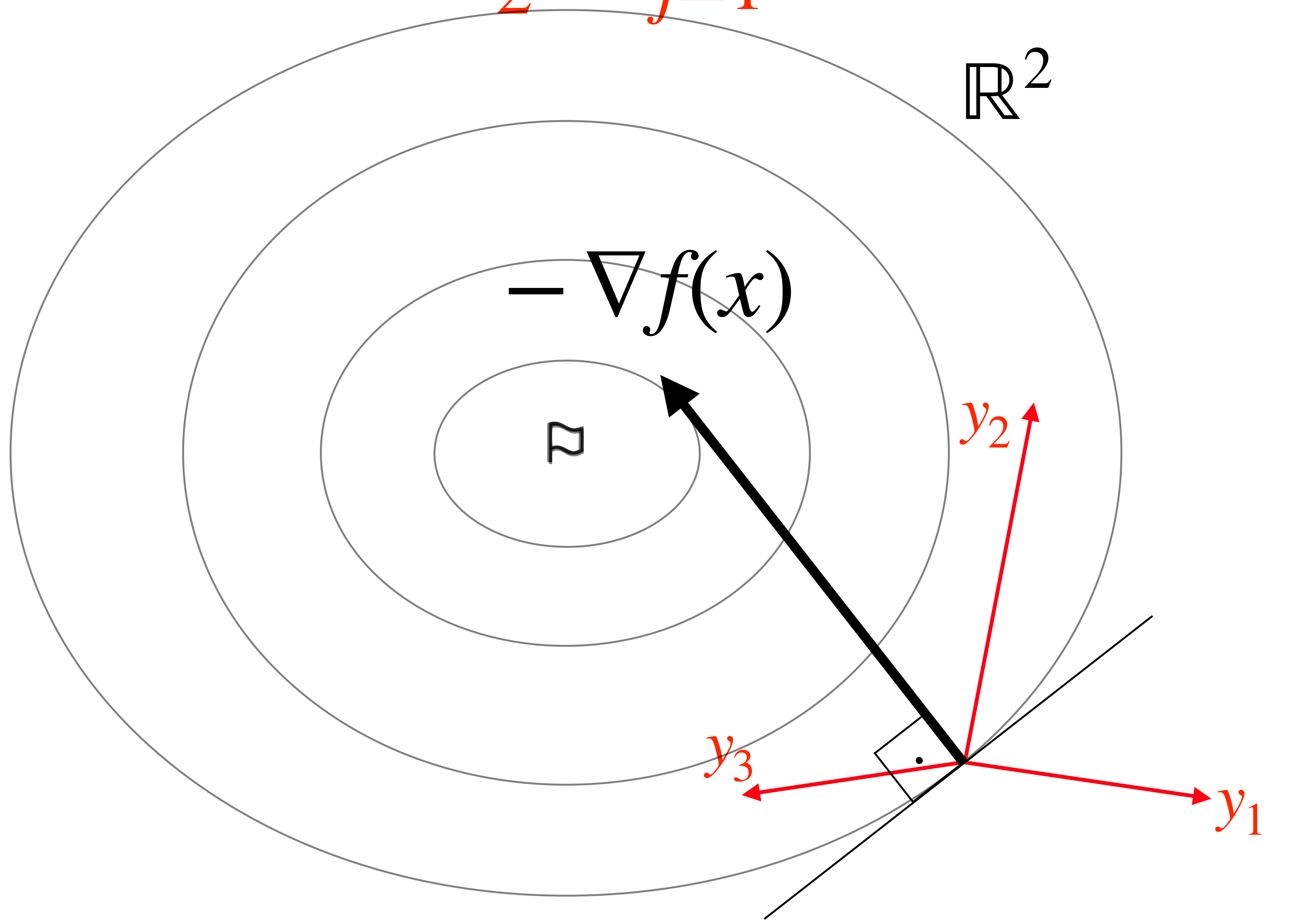
Using ranks introduces invariance to order-preserving f-transformations.

~~small~~ test step

$$y_i \sim \mathcal{N}(0,I) \qquad -w_i = \frac{\ln\left(\textbf{rank}_i(f(x+\delta y_i))\right) - \ln\frac{m+1}{2}}{\frac{1}{2}\sum_{j=1}^{m}\ln(j)}$$

$$x \leftarrow x - \sigma\nabla f(x)$$

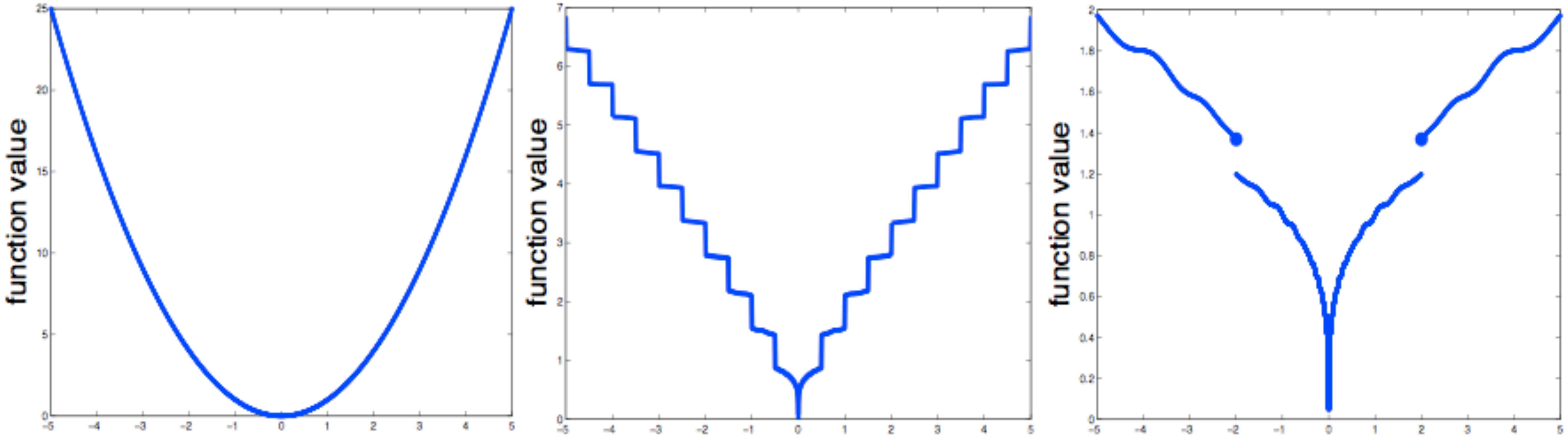
$$\approx x + \sigma\sum_{i=1}^{m} w_i y_i$$



Evolution Strategy (ES) [Rechenberg 1973, Schwefel 1981, Rudolph 1997, Hansen & Ostermeier 2001]

Invariance from Rank-Based Weights

$f = h$ $\qquad$ $f = g_1 \circ h$ $\qquad$ $f = g_2 \circ h$



Three functions belonging to the same equivalence class

A *rank-based search algorithm* is invariant under the transformation with any order preserving (strictly increasing) g.

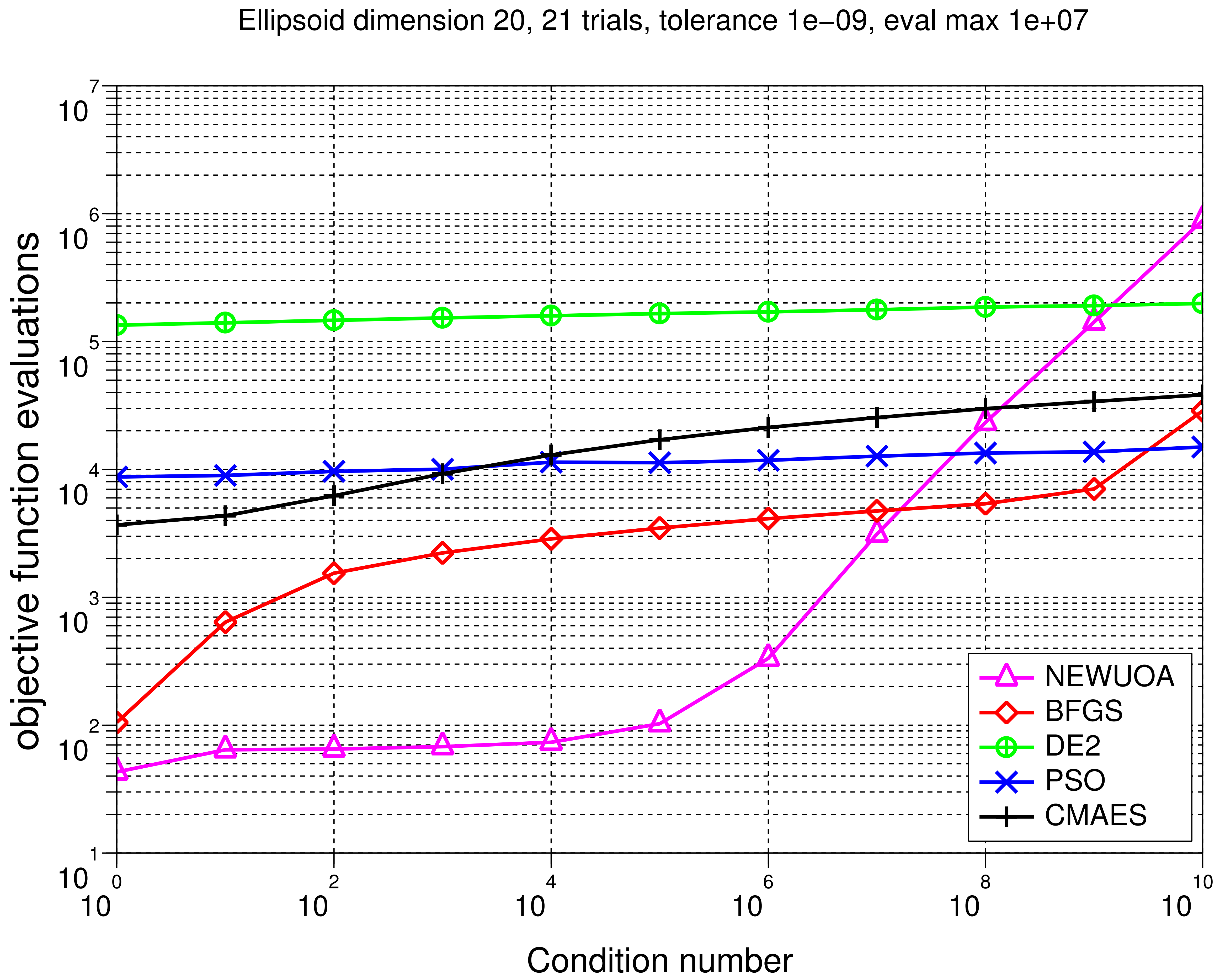
Invariances make

- observations meaningful
- algorithms predictable and/or "robust"

as a rigorous notion of generalization

Comparison to BFGS, NEWUOA, PSO and DE

f convex quadratic, separable with varying condition number α



BFGS (Broyden et al 1970)
NEWUAO (Powell 2004)
DE (Storn & Price 1996)
PSO (Kennedy & Eberhart 1995)
CMA-ES (Hansen & Ostermeier 2001)

$f(\boldsymbol{x}) = g(\boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x})$ with

$\boldsymbol{H}$ diagonal

g identity (for BFGS and NEWUOA)

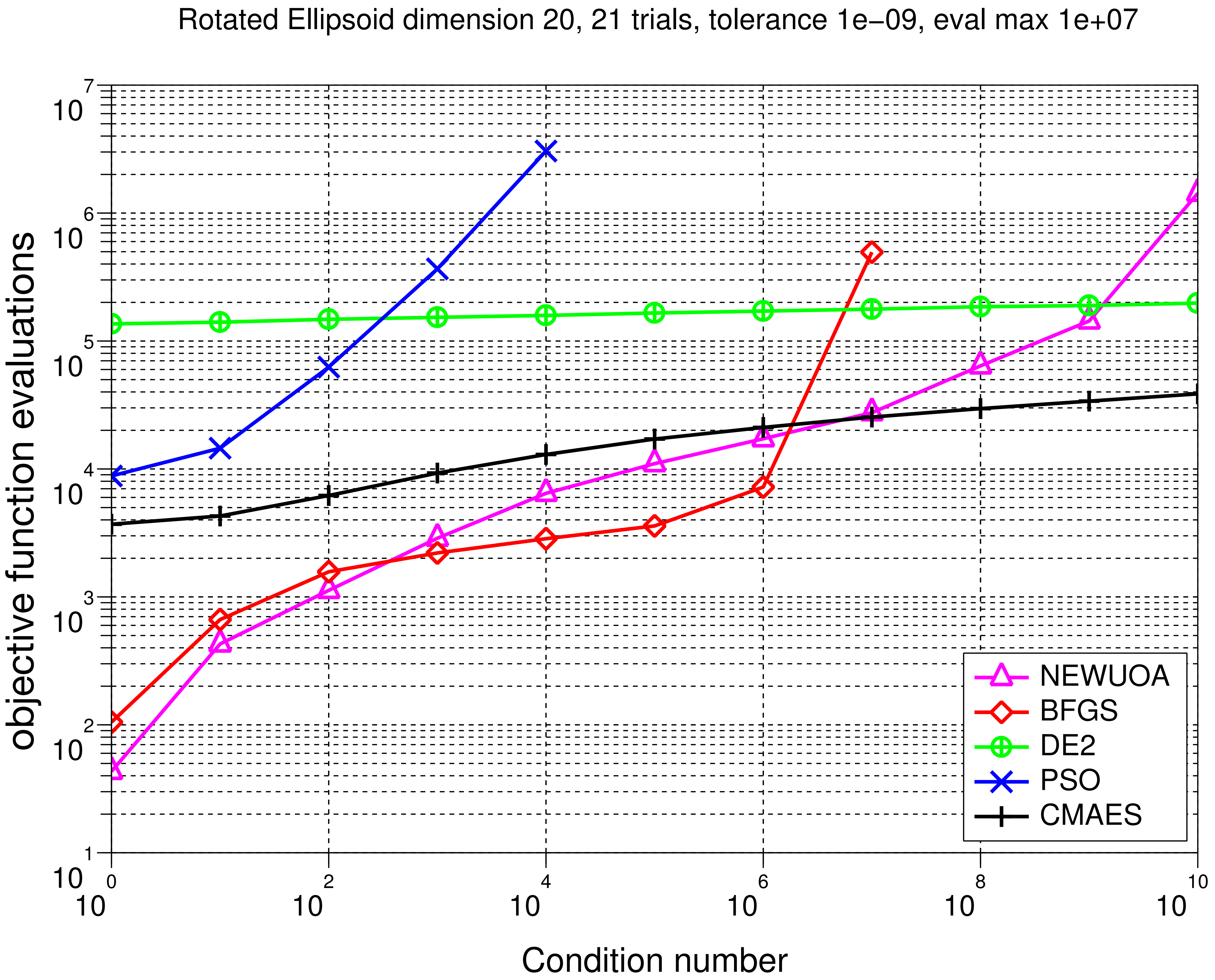
g any order-preserving = strictly increasing function (for all other)

SP1 = average number of objective function evaluations[14] to reach the target function value of $g^{-1}(10^{-9})$

[14] Auger et.al. (2009): Experimental comparisons of derivative free optimization algorithms, SEA

Comparison to BFGS, NEWUOA, PSO and DE

f convex quadratic, non-separable (rotated) with varying condition number α



BFGS (Broyden et al 1970)
NEWUAO (Powell 2004)
DE (Storn & Price 1996)
PSO (Kennedy & Eberhart 1995)
CMA-ES (Hansen & Ostermeier 2001)

$f(\boldsymbol{x}) = g(\boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x})$ with

$\boldsymbol{H}$ full

g identity (for BFGS and NEWUOA)

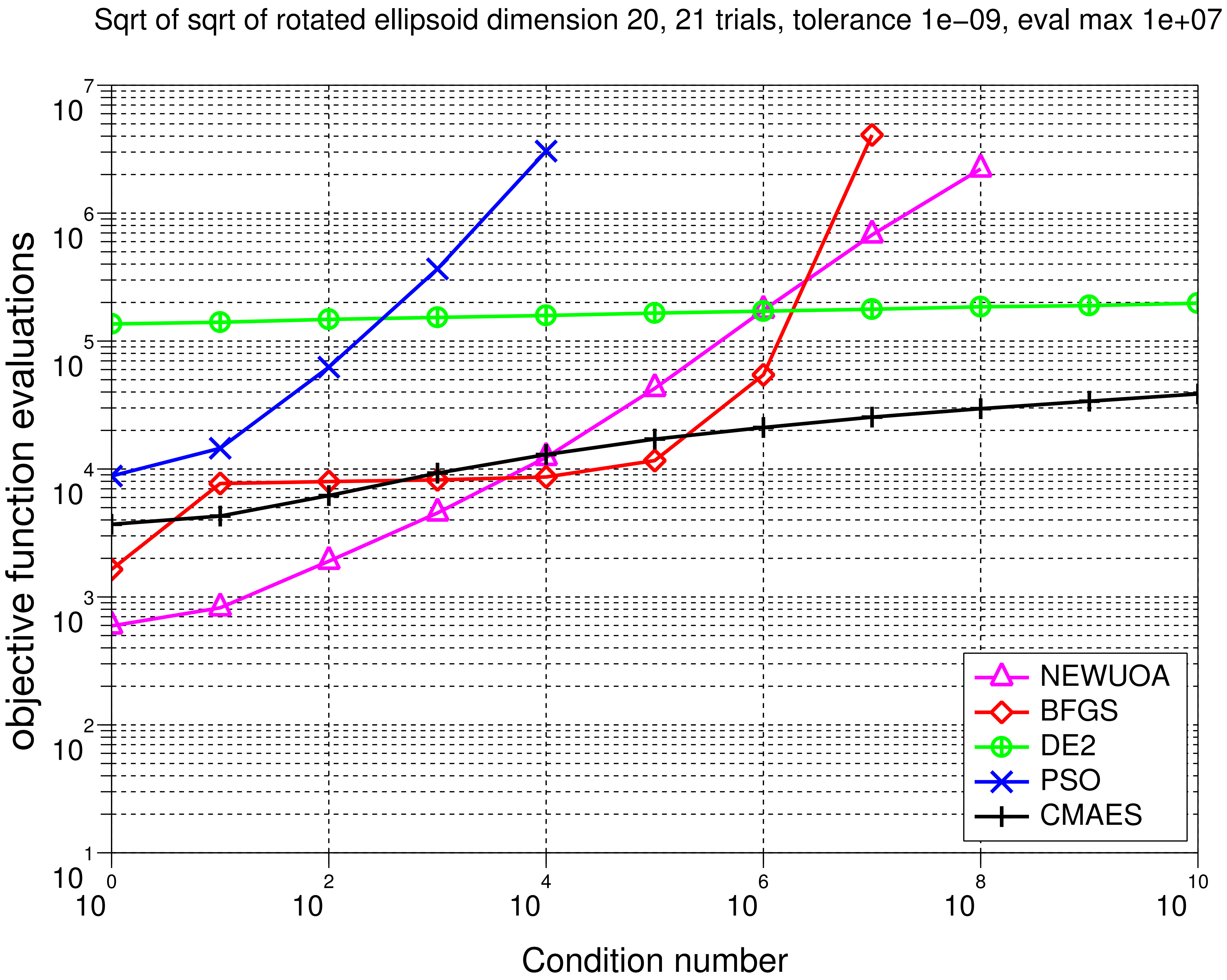
g any order-preserving = strictly increasing function (for all other)

SP1 = average number of objective function evaluations[15] to reach the target function value of $g^{-1}(10^{-9})$

[15] Auger et.al. (2009): Experimental comparisons of derivative free optimization algorithms, SEA

Comparison to BFGS, NEWUOA, PSO and DE

f non-convex, non-separable (rotated) with varying condition number α



BFGS (Broyden et al 1970)
NEWUAO (Powell 2004)
DE (Storn & Price 1996)
PSO (Kennedy & Eberhart 1995)
CMA-ES (Hansen & Ostermeier 2001)

$f(\boldsymbol{x}) = g(\boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x})$ with
$\boldsymbol{H}$ full
$g : x \mapsto x^{1/4}$ (for BFGS and NEWUOA)
g any order-preserving = strictly increasing function (for all other)

SP1 = average number of objective function evaluations[16] to reach the target function value of $g^{-1}(10^{-9})$

[16] Auger et.al. (2009): Experimental comparisons of derivative free optimization algorithms, SEA

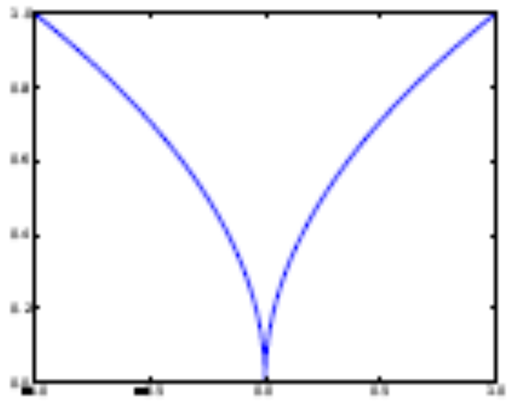
From Gradient Search to Evolution Strategies

	Gradient Search	Evolution Strategy
Test Steps:	unit vectors	random vectors
	dimension n	any number > 1
	small	large
Weights:	partial derivatives	fixed rank-based
Realized Step Length:	line search	step-size control (non-trivial)

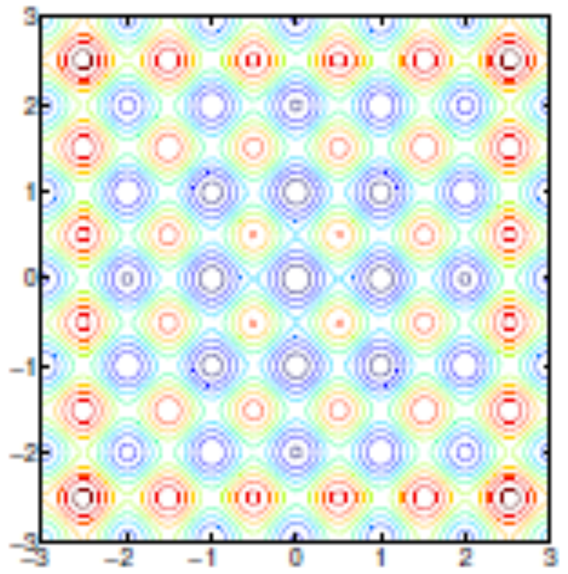
What Makes an Optimization Problem Difficult?

- non-linear, non-quadratic

on linear and quadratic functions
specialized search policies are available



- non-convexity

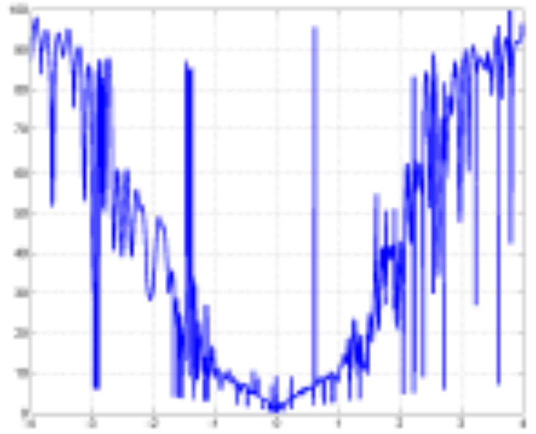


- dimensionality (size of search space) and non-separability

dimension considerably larger than three with
dependencies between the variables

From Gradient-Based to
Evolutionary Optimization

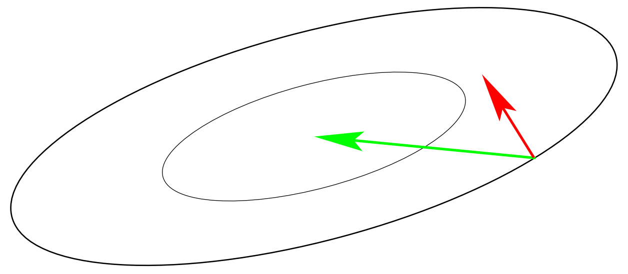
- multimodality



- ruggedness

high frequency modality, non-smooth, discontinuous

- ill-conditioning



varying sensitivities,
worst case: non-smooth concave level sets

In any case, the objective function must be highly regular

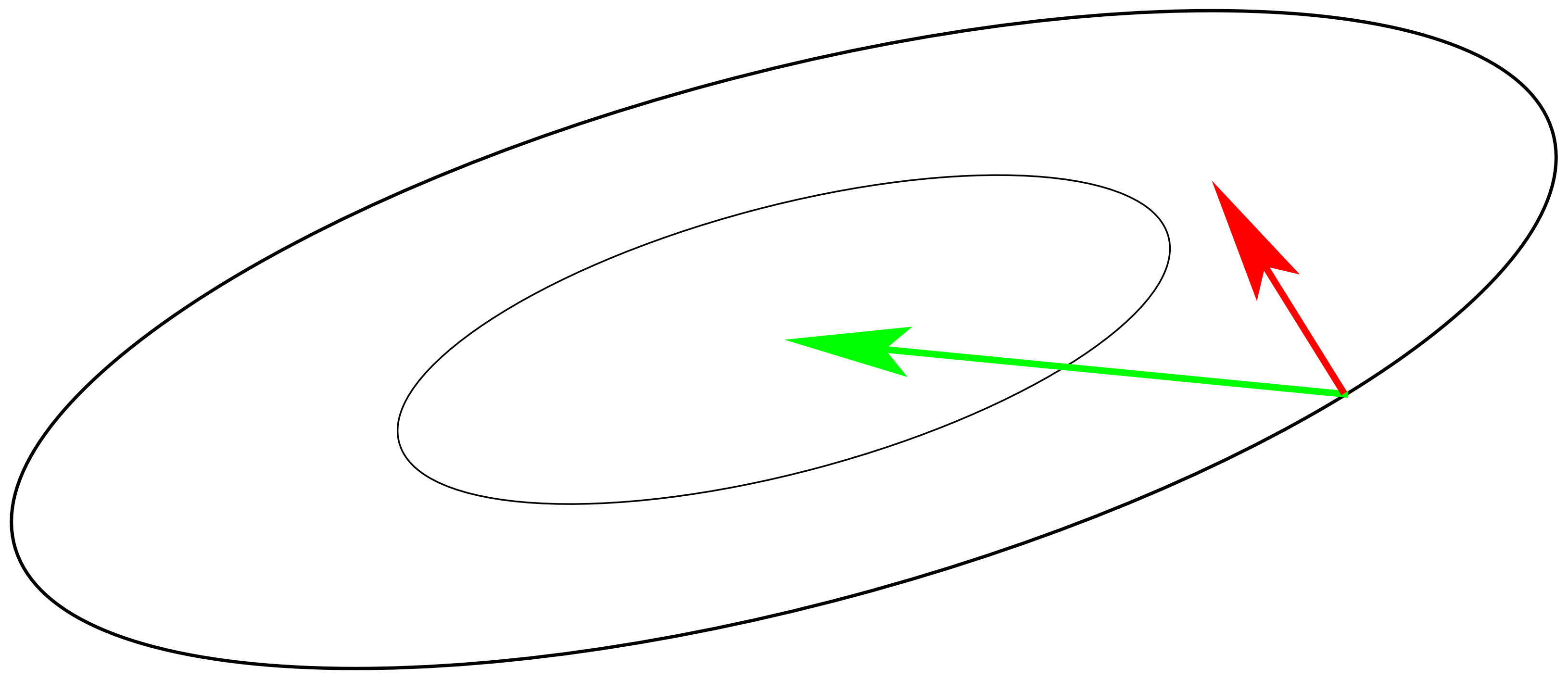
Ill-Conditioned Problems

Curvature of level sets

Consider the convex-quadratic function

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

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



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

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

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

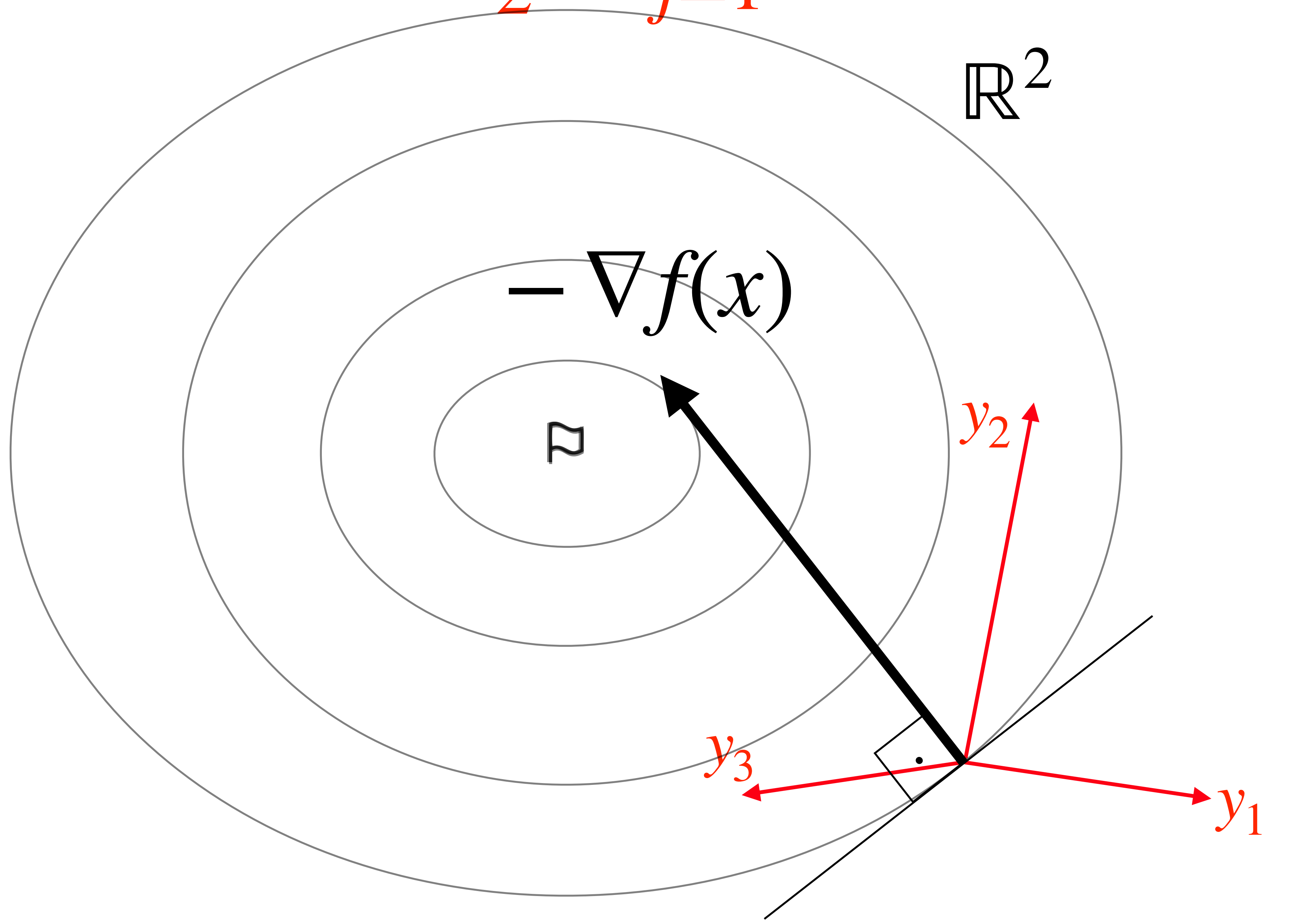
Rank-Based Approximated Gradient Descent

Using ranks introduces invariance to order-preserving f-transformations.

$$y_i \sim \mathcal{N}(0, I) \qquad -w_i = \frac{\ln\left(\textbf{rank}_i(f(x + \delta y_i))\right) - \ln\frac{m+1}{2}}{\frac{1}{2}\sum_{j=1}^{m}\ln(j)}$$

$$x \leftarrow x - \sigma \nabla f(x)$$

$$\approx x + \sigma \sum_{i=1}^{m} w_i y_i$$

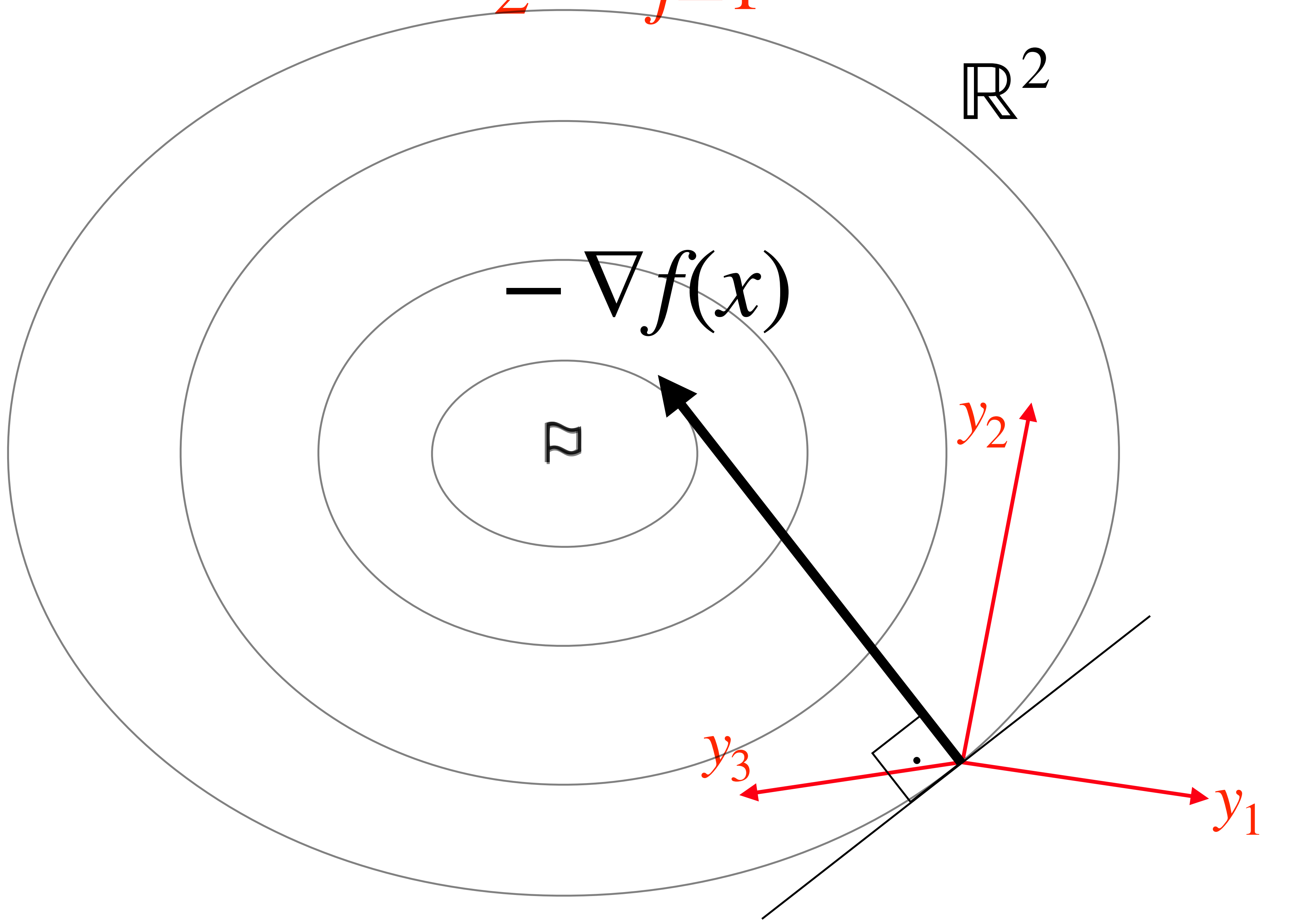


Evolution Strategy (ES) [Rechenberg 1973, Schwefel 1981, Rudolph 1997]

Rank-Based Approximated Gradient Descent

variable metric, updated to estimate H^{-1}

$$y_i \sim \mathcal{N}(0, C) \qquad -w_i = \frac{\ln\left(\mathbf{rank}_i(f(x+\delta y_i))\right) - \ln\frac{m+1}{2}}{\frac{1}{2}\sum_{j=1}^{m}\ln(j)}$$



$$x \leftarrow x - \sigma \nabla f(x)$$

$$\approx x + \sigma \sum_{i=1}^{m} w_i y_i$$

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [Hansen & Ostermeier 2001, Hansen et al 2003]

CMA-ES

Let $m \in \mathbb{R}^n$, $\sigma > 0$, $C = \mathbf{I}_n$, $y_0 = \mathbf{0}$

population size

$$x_k \sim \mathcal{N}(m, \sigma^2 C) = m + \sigma\, \mathcal{N}(0, C) \in \mathbb{R}^n, \quad k = 1 \dots \lambda$$

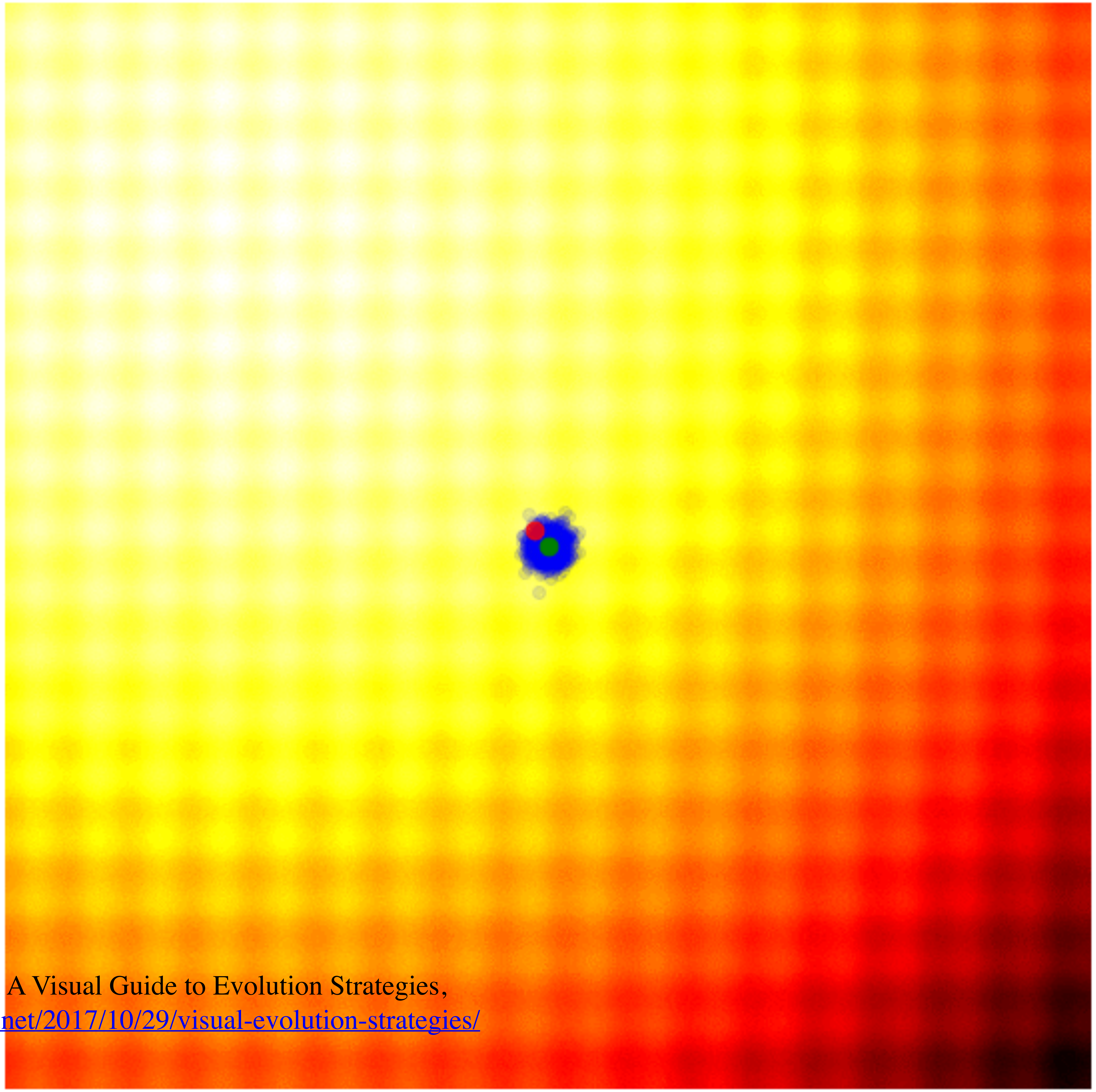
$$y_k = \frac{x_{\text{permute}_\lambda(k)} - m}{\sigma}$$ **sorted by fitness** $y_k \sim \mathcal{N}(0, C)$

$$m \leftarrow m + c_m\, \sigma \sum_{k=1}^{\mu} w_k y_k, \quad c_m \approx \sum_{k=1}^{\mu} w_k \approx 1,\ \mu \approx \lambda/2$$

$$y_0 \leftarrow (1 - c_c)\, y_0 + \sqrt{c_c(2 - c_c)\mu_w} \sum_{k=1}^{\mu} w_k y_k \quad \mu_w = \frac{(\sum_{i=1}^{\mu} w_k)^2}{\sum_{i=1}^{\mu} w_k{}^2}$$

$$C \leftarrow C + c_\mu \sum_{k=0}^{\lambda} w_k (y_k y_k^\top - C), \quad c_\mu \approx \lambda/n^2, \sum_{k=0}^{\lambda} w_k \approx 0$$

$$\sigma \leftarrow \sigma \times \exp(\dots)$$



David Ha (2017). A Visual Guide to Evolution Strategies,
http://blog.otoro.net/2017/10/29/visual-evolution-strategies/

CMA-ES

Covariance Matrix Adaptation Evolution Strategy

- Strive to sample the optimal (multi-variate) Gaussian distribution *at any given iteration*

  - *"optimal" mean (best estimate of the optimum)*

    given the available information

  - *optimal covariance matrix* C

    given the available information

  - optimal step-size σ

    given the covariance matrix

- A natural gradient update of mean and covariance matrix

  provides a theoretical framework/justification [JMLR 18(18), 2017]

$$\theta_{t+1} = \theta_t + \eta \frac{1}{Z(\lambda)} \sum_{k=1}^{\lambda} \big(\lambda/2 - \mathrm{rank}(f(x_k))\big) \widetilde{\nabla}_\theta \ln p(x_k|\theta)\Big|_{\theta=\theta_t}$$

- Convergence speed is almost independent of the number of samples (not $\gg n$)

  property of multi-recombinative Evolution Strategies

Practical Advice

Approaching an Unknown Optimization Problem

- Objective formulation

  for example $\sum_i x_i^2$ and $\sum_i |x_i|$ have the same optimal (minimal) solution but may be very differently "optimizable"

- Problem/variable encoding

  for example log scale vs linear scale vs quadratic transformation

- Create section plots ($f(x)$ vs x on a line)

  one-dimensional grid search is cheap, may reveal ill-conditioning or multi-modality

- Try to locally improve a given (good) solution

- Start local search from different initial solutions

  Ending up always in different solutions? Or always in the same?

- Apply "global search" setting

- see also http://cma.gforge.inria.fr/cmaes_sourcecode_page.html#practical pypi.org/project/cma/

cma 2.7.0

Latest version

```
pip install cma
```

Last released: Apr 25, 2019

CMA-ES, Covariance Matrix Adaptation Evolution Strategy for non-linear numerical optimization in Python

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage

Statistics

GitHub statistics:

- Stars: 310
- Forks: 51
- Open issues/PRs: 25

View statistics for this project via Libraries.io, or by using Google BigQuery

Project description

A stochastic numerical optimization algorithm for difficult (non-convex, ill-conditioned, multi-modal, rugged, noisy) optimization problems in continuous search spaces, implemented in Python.

Typical domain of application are bound-constrained or unconstrained objective functions with:

- search space dimension between, say, 5 and (a few) 100,
- no gradients available,
- at least, say, 100 times dimension function evaluations needed to get satisfactory solutions,
- non-separable, ill-conditioned, or rugged/multi-modal landscapes.

The CMA-ES is quite reliable, however for small budgets (fewer function evaluations than, say, 100 times dimension) or in very small dimensions better (i.e. faster) methods are available.

The `pycma` module provides two independent implementations of the CMA-ES algorithm in the classes `cma.CMAEvolutionStrategy` and `cma.purecma.CMAES`.

Installation

There are several ways of installation:

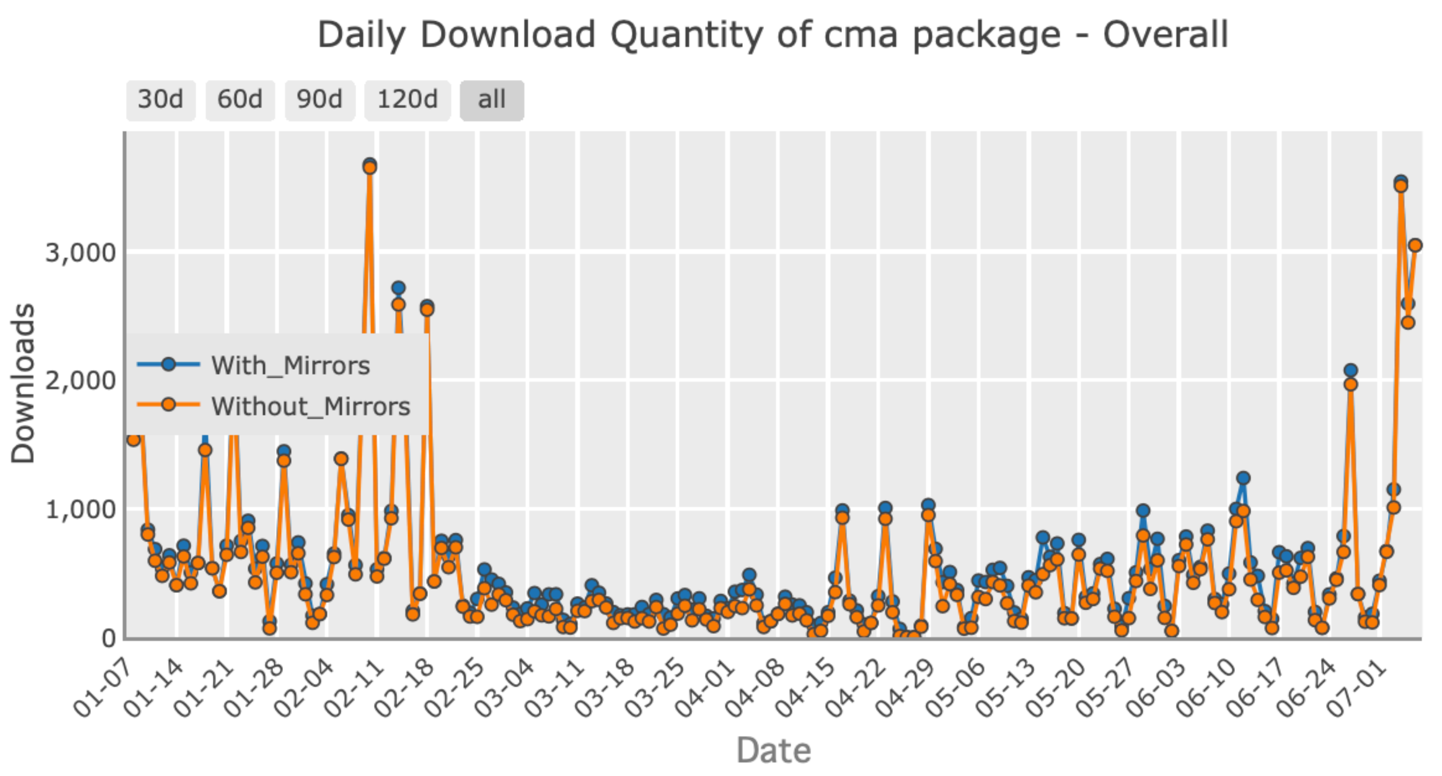
- In the terminal command line type:

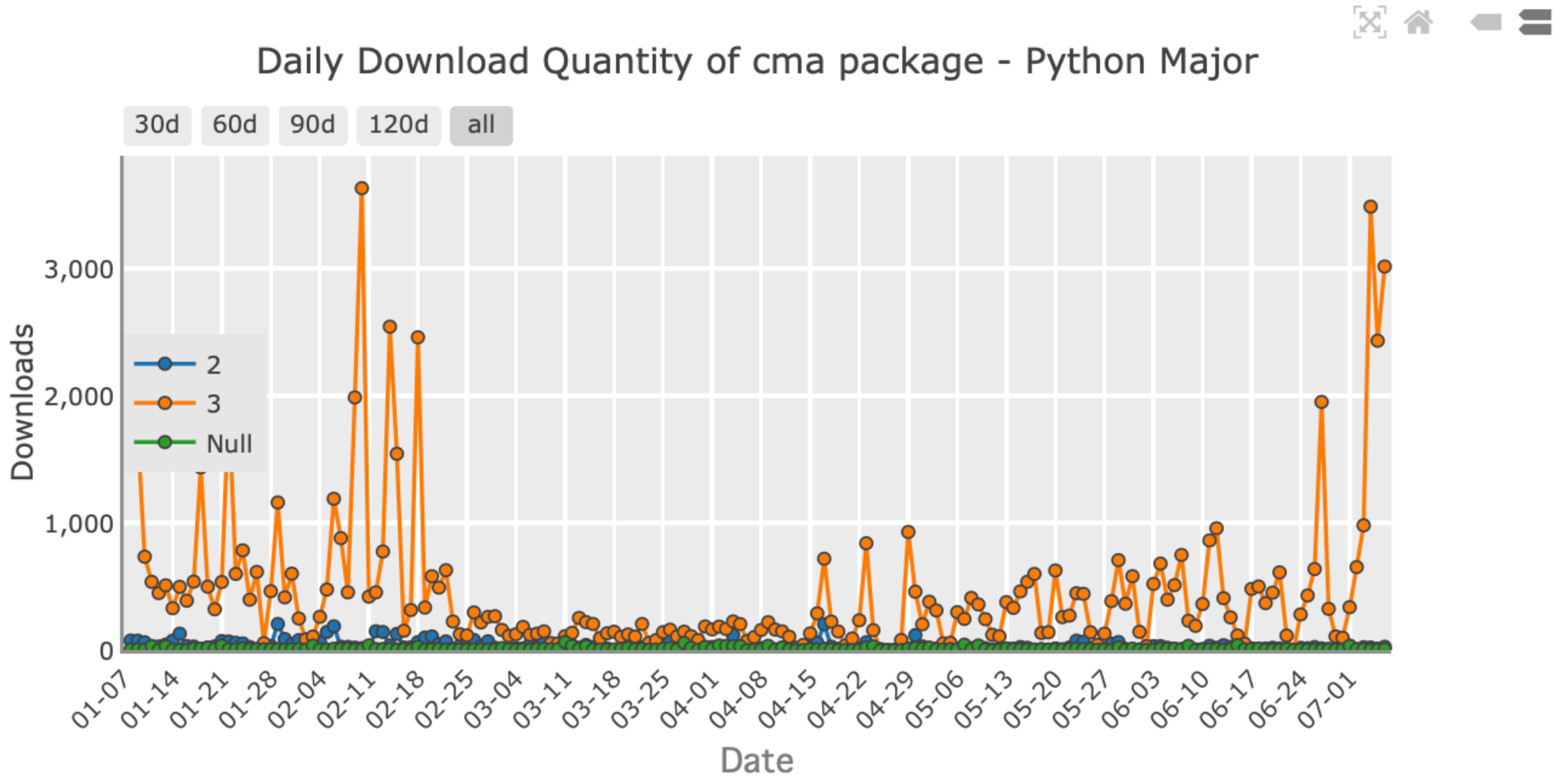
pypistats.org/packages/cma

cma

PyPI page
Home page
Author: Nikolaus Hansen
License: BSD
Summary: CMA-ES, Covariance Matrix Adaptation Evolution Strategy for non-linear numerical optimization in Python
Latest version: 2.7.0

Downloads last day: 3,048
Downloads last week: 11,198
Downloads last month: 22,184





Python Example in Jupyter-Lab

```python
# download & install anaconda python
# optional: "conda create" in case a different Python version is needed
# shell cmd "pip install cma" to install a CMA-ES module (or see github)
# shell cmd "jupyter-notebook" or "jupyter-lab"

%pylab ipympl
import cma
```

```
Populating the interactive namespace from numpy and matplotlib
```

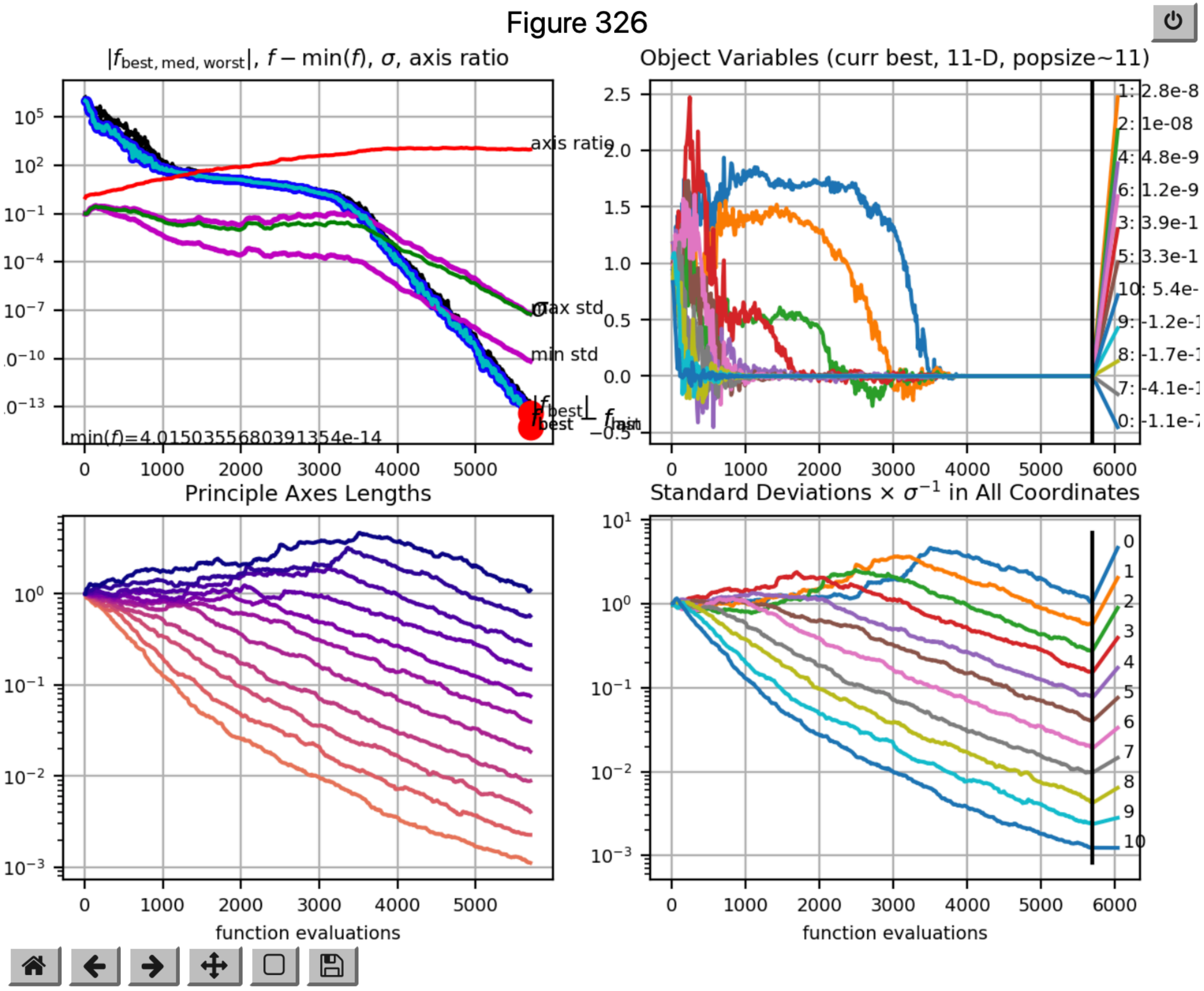
```python
x, es = cma.fmin2(cma.ff.elli, 11 * [1], 0.1)
```

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

```
(5_w,11)-aCMA-ES (mu_w=3.4,w_1=42%) in dimension 11 (seed=822389, Tue Jul  9 16:35:30 2019
)
Iterat #Fevals   function value  axis ratio  sigma  min&max std  t[m:s]
    1     11 1.037523721813126e+06 1.0e+00 9.97e-02  1e-01  1e-01 0:00.0
    2     22 9.633352873252528e+05 1.3e+00 9.77e-02  9e-02  1e-01 0:00.0
    3     33 7.976836387974678e+05 1.3e+00 1.00e-01  1e-01  1e-01 0:00.0
  100   1100 4.807072196140337e+01 1.5e+01 2.34e-02  3e-03  3e-02 0:00.1
  200   2200 8.681869407386893e+00 1.1e+02 2.37e-02  5e-04  5e-02 0:00.2
  300   3300 4.683795983800626e-01 5.1e+02 2.58e-02  2e-04  9e-02 0:00.3
  400   4400 5.684285745919520e-07 1.1e+03 8.11e-05  2e-07  2e-04 0:00.4
  500   5500 3.857152913051700e-13 9.8e+02 1.78e-07  3e-10  2e-07 0:00.5
  518   5698 4.015035568039135e-14 9.8e+02 5.64e-08  7e-11  6e-08 0:00.5
termination on tolfun=1e-11 (Tue Jul  9 16:35:31 2019)
final/bestever f-value = 2.491784e-14 2.491784e-14
incumbent solution: [-8.03092159e-08  2.46363535e-08  6.15113753e-09  2.77515423e-11
  2.17672302e-09  1.15830669e-09  1.54068182e-09 -8.97542512e-11 ...]
std deviations: [6.12538334e-08 3.23572912e-08 1.56460460e-08 8.59198717e-09
 4.26818495e-09 2.25284147e-09 1.06453007e-09 5.47204032e-10 ...]
```

```python
cma.plot()
```

```
cma.plot()
```



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

<cma.logger.CMADataLogger at 0x7f90d09d0630>

function evaluations function evaluations

<cma.logger.CMADataLogger at 0x7f90d09d0630>

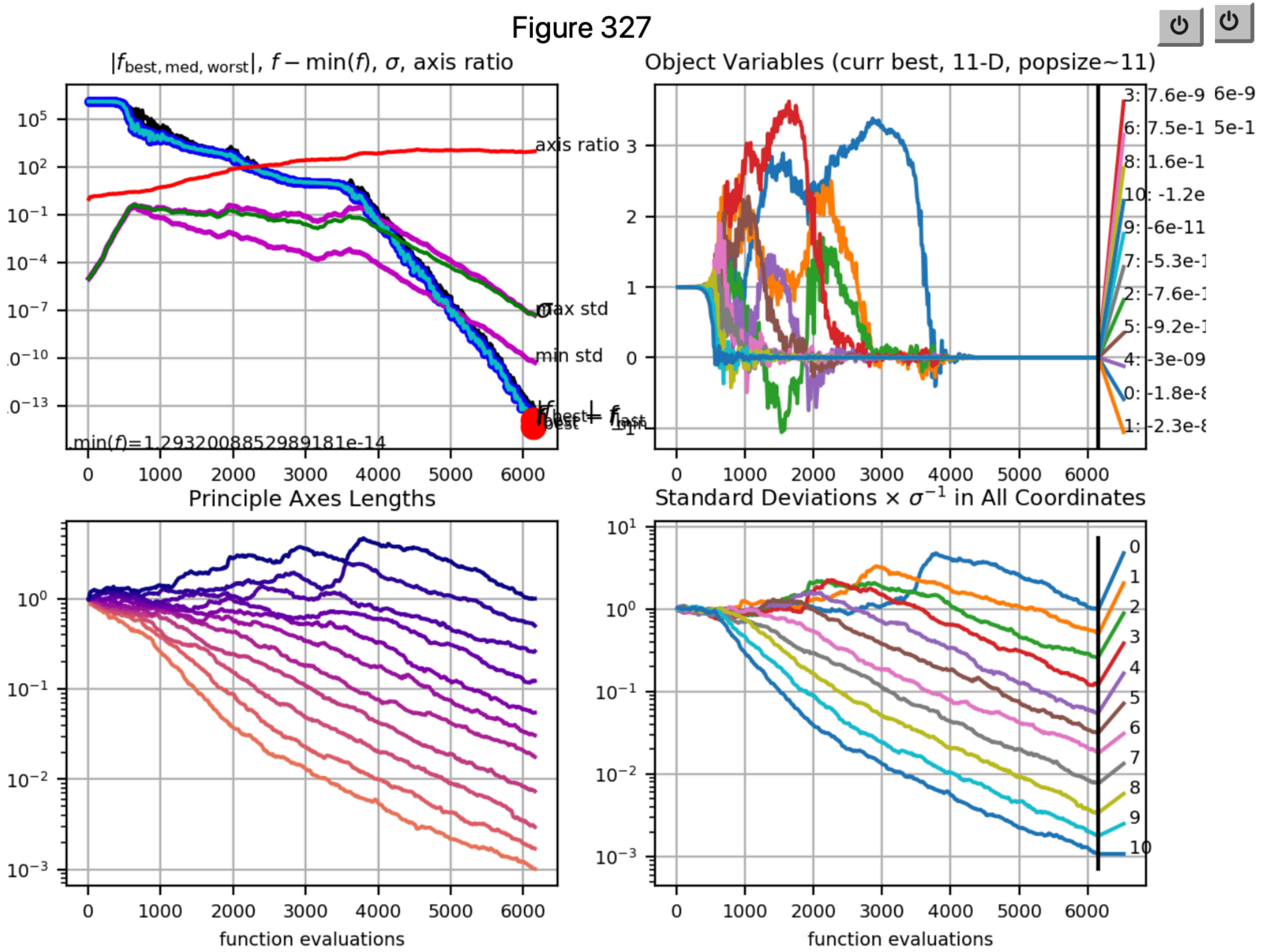
```
1  x, es = cma.fmin2(cma.ff.elli, 11 * [1], 1e-5)
```

```
(5_w,11)-aCMA-ES (mu_w=3.4,w_1=42%) in dimension 11 (seed=909918, Thu Jul 11 10:51:45 2019)
Iterat #Fevals   function value  axis ratio  sigma  min&max std  t[m:s]
    1     11 1.335426651544717e+06 1.0e+00 9.41e-06  9e-06  9e-06 0:00.0
    2     22 1.335403970874783e+06 1.2e+00 1.02e-05  1e-05  1e-05 0:00.0
    3     33 1.335381895674725e+06 1.3e+00 1.13e-05  1e-05  1e-05 0:00.0
   41    451 1.236413461768105e+06 3.4e+00 7.70e-03  6e-03  1e-02 0:00.1
termination on tolfacupx=1000.0 (Thu Jul 11 10:51:45 2019)
final/bestever f-value = 1.239692e+06 1.236413e+06
incumbent solution: [1.01162517 0.99668741 0.98913604 1.01240495 0.99246088 0.98896483
 0.9867979  0.98844654 ...]
std deviations: [0.00708466 0.0063847  0.00740649 0.00944938 0.00763342 0.00751928
 0.00737737 0.00689999 ...]
```

```
1  x, es = cma.fmin2(cma.ff.elli, 11 * [1], 1e-5, {'tolfacupx': 1e6})
```

```
(5_w,11)-aCMA-ES (mu_w=3.4,w_1=42%) in dimension 11 (seed=942830, Thu Jul 11 10:52:23 2019)
Iterat #Fevals   function value  axis ratio  sigma  min&max std  t[m:s]
    1     11 1.335419049661035e+06 1.0e+00 1.03e-05  1e-05  1e-05 0:00.0
    2     22 1.335398380223496e+06 1.2e+00 1.19e-05  1e-05  1e-05 0:00.0
    3     33 1.335379395340697e+06 1.4e+00 1.38e-05  1e-05  1e-05 0:00.0
  100   1100 4.999254215133446e+03 6.5e+00 2.01e-01  5e-02  2e-01 0:00.1
  200   2200 1.030349103924029e+02 9.9e+01 1.24e-01  4e-03  3e-01 0:00.2
  300   3300 9.939855671544342e+00 3.0e+02 3.53e-02  4e-04  9e-02 0:00.3
  400   4400 2.862006949075298e-04 1.1e+03 1.55e-03  6e-06  6e-03 0:00.4
  500   5500 2.373014204647906e-10 1.1e+03 3.30e-06  6e-09  6e-06 0:00.6
  560   6160 1.840635073907070e-14 1.0e+03 5.18e-08  6e-11  5e-08 0:00.7
termination on tolfun=1e-11 (Thu Jul 11 10:52:24 2019)
final/bestever f-value = 9.862934e-15 9.862934e-15
incumbent solution: [-1.83502574e-08 -2.46920059e-08 -2.79112525e-09  1.76731134e-09
 -1.63323394e-09 -3.73512722e-10  4.61670274e-10 -2.78078340e-10 ...]
```

```python
1 cma.plot()
```




```
<cma.logger.CMADataLogger at 0x7f90c02fd828>
```

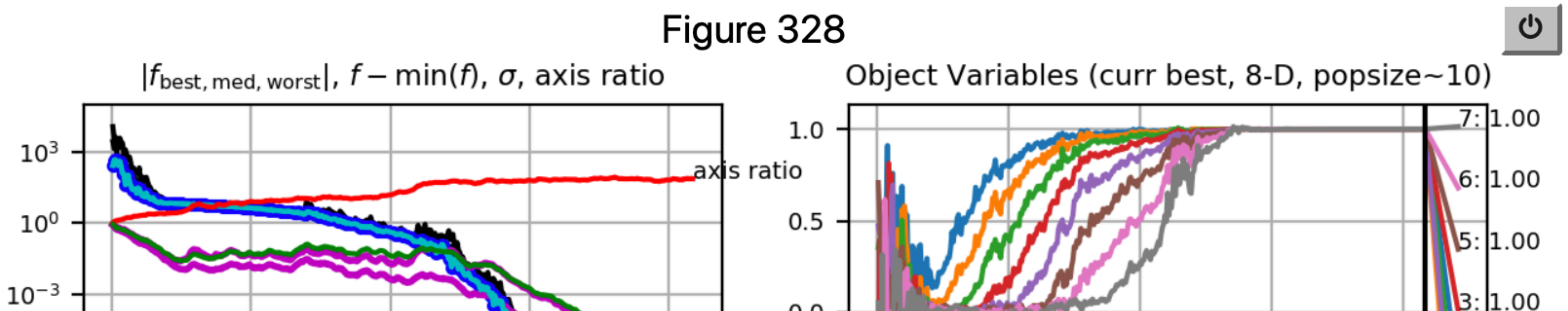
A Transparent Interface

On Object-Oriented Programming of Optimizers [Collette et al 2010]

```
es = cma.CMAEvolutionStrategy(8 * [0], 1.0)
while not es.stop():
    X = es.ask()
    es.tell(X, [cma.ff.rosen(x) for x in X])
    es.logger.add()
    es.disp()
es.result_pretty();
```

```
(5_w,10)-aCMA-ES (mu_w=3.2,w_1=45%) in dimension 8 (seed=610691, Sat Jul  6 20:59:48 2019)
Iterat #Fevals   function value  axis ratio  sigma  min&max std  t[m:s]
    1     10 2.655345399152838e+02 1.0e+00 8.69e-01  8e-01  9e-01 0:00.0
    2     20 4.505842690989847e+02 1.1e+00 7.90e-01  7e-01  8e-01 0:00.0
    3     30 4.555527678670772e+02 1.2e+00 6.54e-01  6e-01  6e-01 0:00.0
  100   1000 4.338593958650433e+00 8.1e+00 5.45e-02  2e-02  5e-02 0:00.1
  200   2000 4.185334229855005e-01 1.5e+01 4.29e-02  4e-03  2e-02 0:00.2
  300   3000 7.700486946188576e-06 6.3e+01 1.78e-03  6e-05  1e-03 0:00.2
  400   4000 5.768459312593045e-13 6.5e+01 2.38e-06  2e-08  7e-07 0:00.3
  417   4170 3.324045009491747e-14 7.4e+01 5.66e-07  4e-09  1e-07 0:00.3
termination on tolfun=1e-11
final/bestever f-value = 3.324045e-14 3.324045e-14
incumbent solution: [1.0000000049914348, 1.0000000000632427, 1.0000000033696235, 1.0000000
043215198, 1.0000000018643387, 1.0000000034016863, 1.0000000062489975, 1.000000017143993]
std deviation: [3.685091313595847e-09, 4.633054743103208e-09, 5.686358351473224e-09, 8.136
372446898455e-09, 1.692834679130827e-08, 3.2897094225652294e-08, 6.40728722918568e-08, 1.2
828702723990136e-07]
```

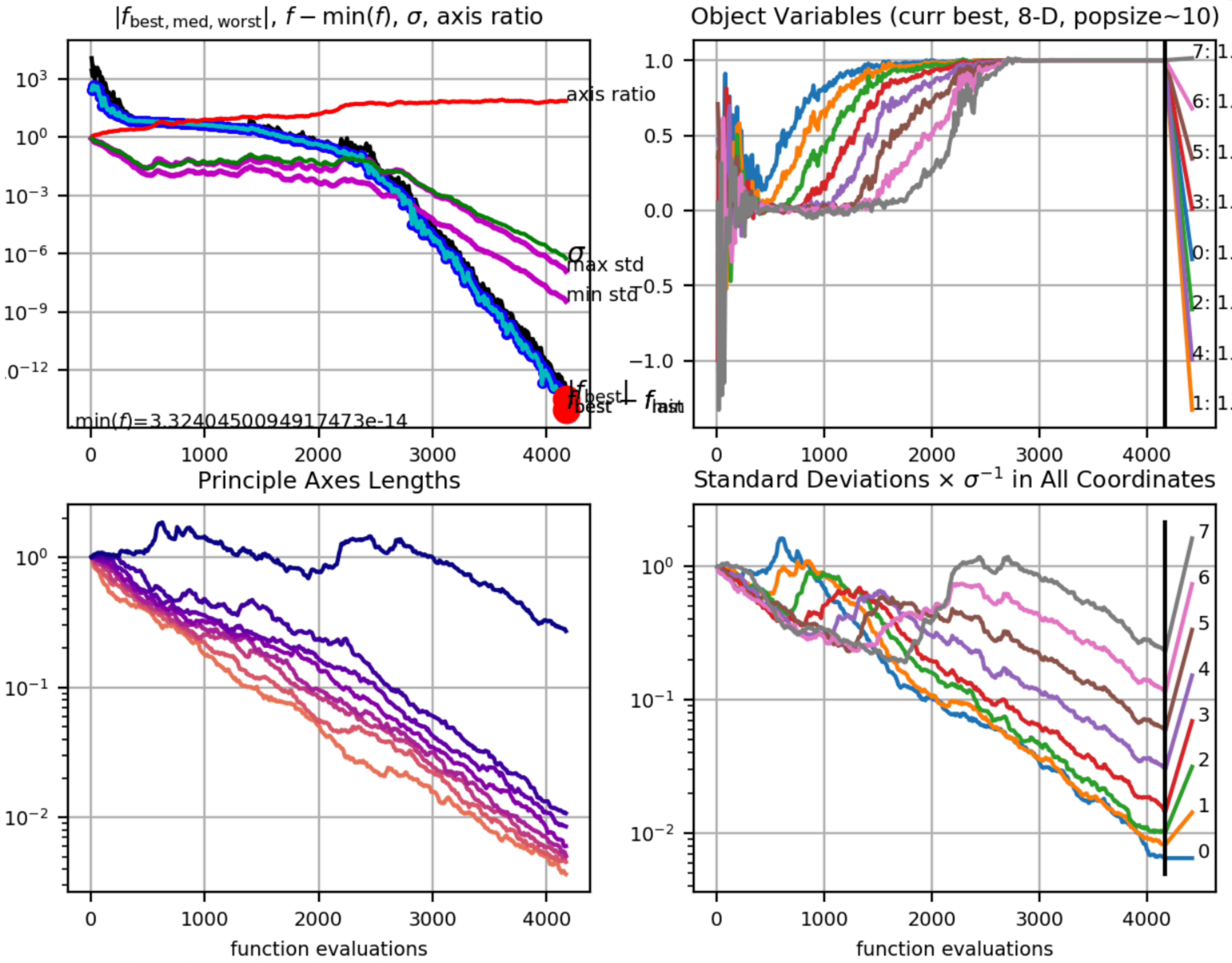
```
cma.plot()
```



828702723990136e-07]

```
cma.plot()
```

Figure 328



<cma.logger.CMADataLogger at 0x7f9e09aafef0>

Thank You