Optimization (in Chemistry)

Guido Falk von Rudorff

Frequent problem class

Energy

- Find most stable molecular geometry	Compare conformers
- Find transition state geometries	Identify reaction pathways

Residuals

- Fitting experimental data	Model observations
- Potential fitting	Simplify calculations
- Machine learning	Surrogate models

Definitions

Solution coefficients x

- Molecular geometries
- Fitting coefficients
- Model coefficients

Scalar objective function f

- Energy
- Residual norm
- Here: smooth, i.e. differentiable function

$$f(x_1, x_2, \ldots, x_n) = f(\mathbf{x}) = y$$

Domain X

- Valid parameter range
- Any solution within accepted

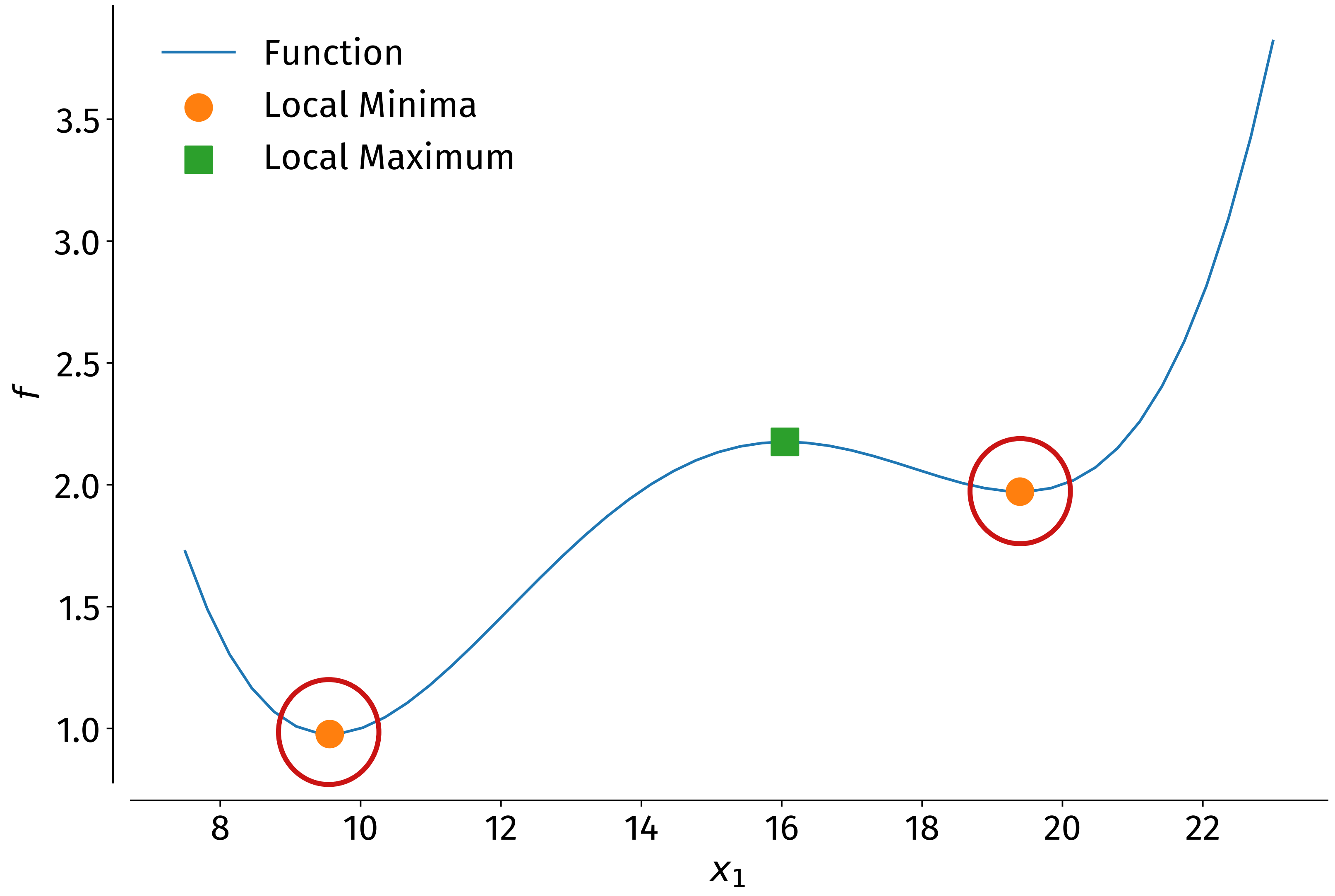
Target x_0

- Maximise or minimise y (over domain)

$$\mathbf{x}_0 \equiv \underset{\mathbf{x}\in X}{\operatorname{argmin}} f(\mathbf{x})$$
$$= \{\mathbf{x} | \mathbf{x}, \mathbf{y} \in X : f(\mathbf{x}) \leq f(\mathbf{y})\}$$

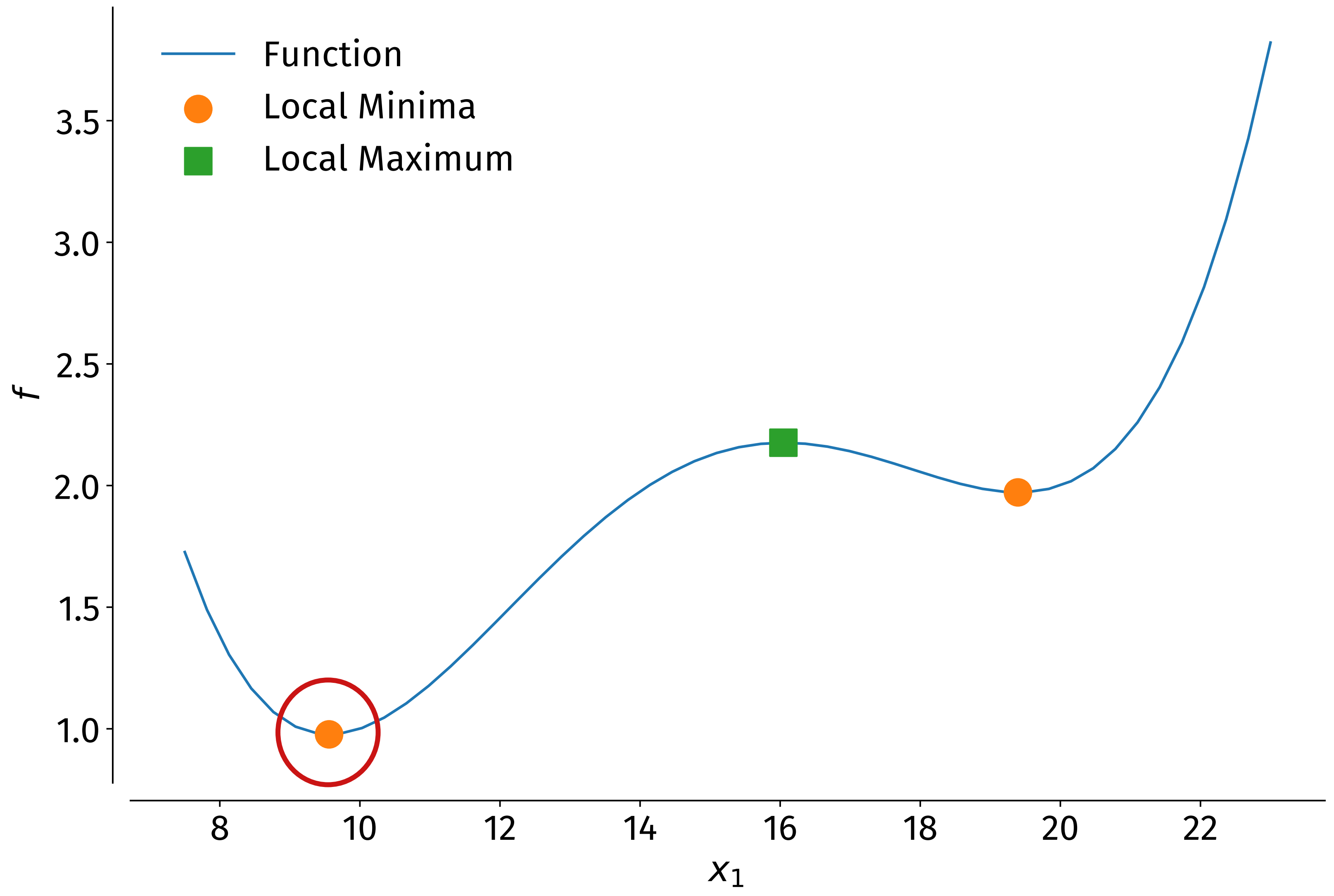
Definition: Local minimum

$$\exists \epsilon > 0 : \forall y \in [x_0 - \epsilon, x_0 + \epsilon] : f(x_0) \leq f(y)$$

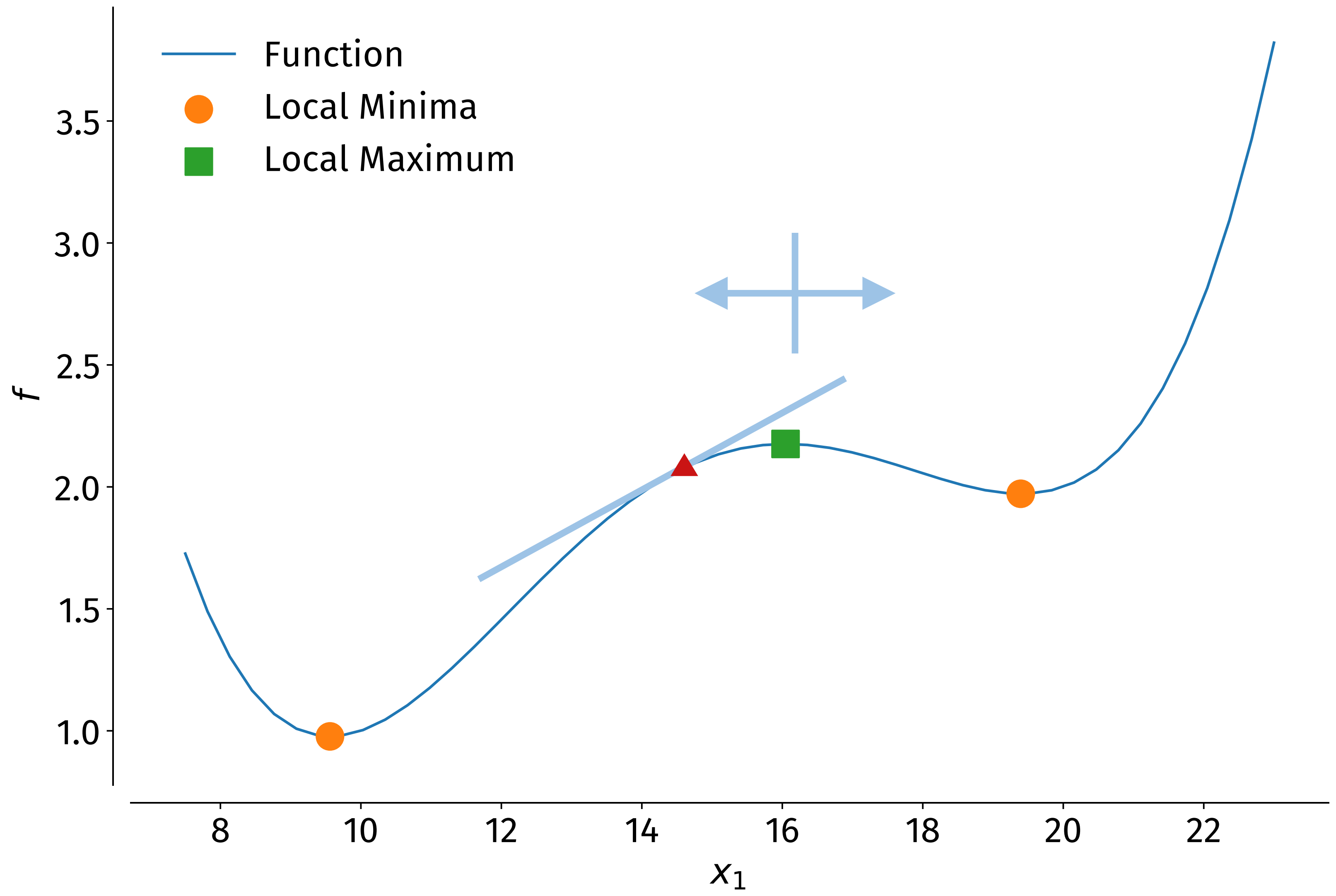


Definition: Global minimum

$$\forall y \in X : f(x_0) \leq f(y)$$

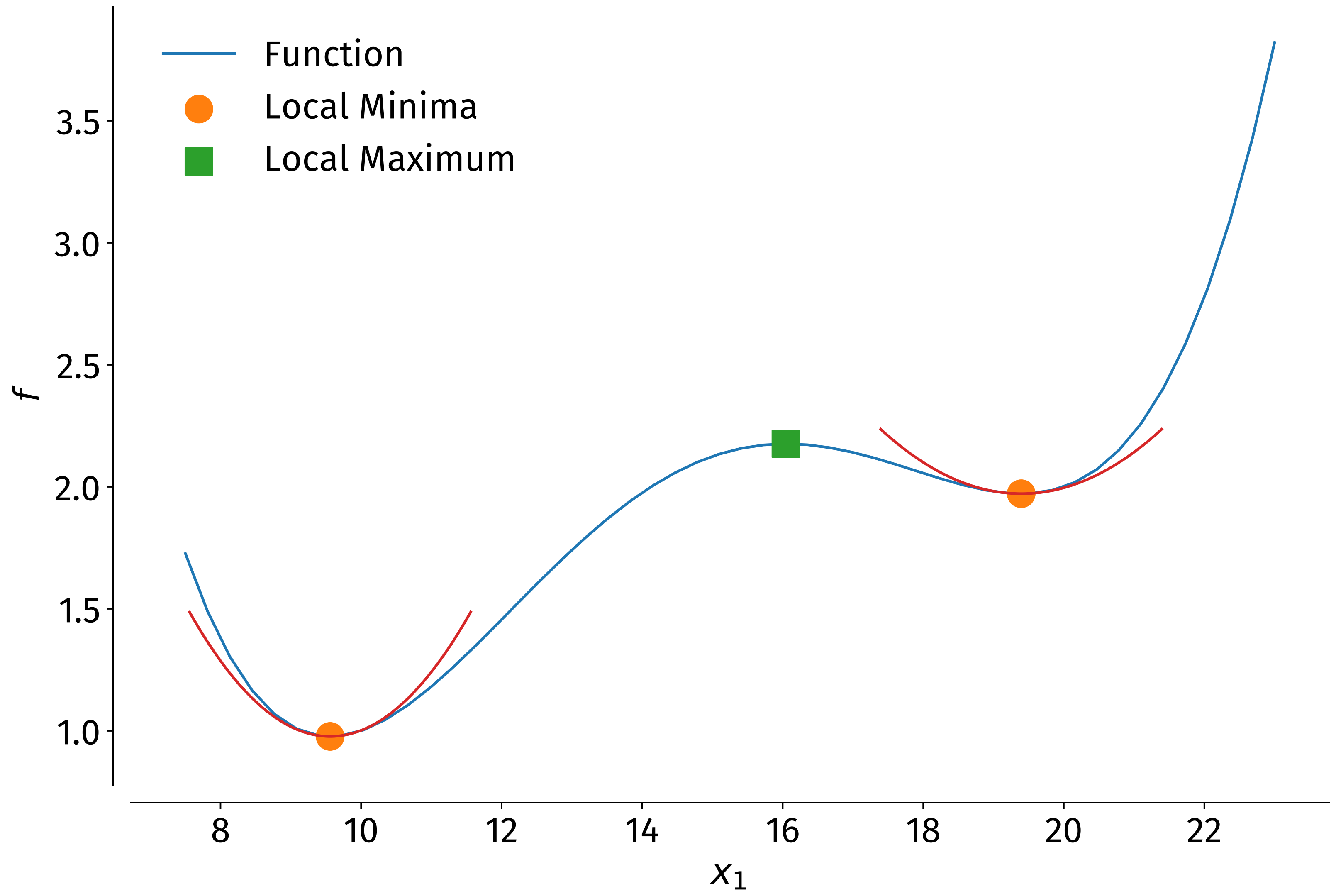


Definition: Attractive basin



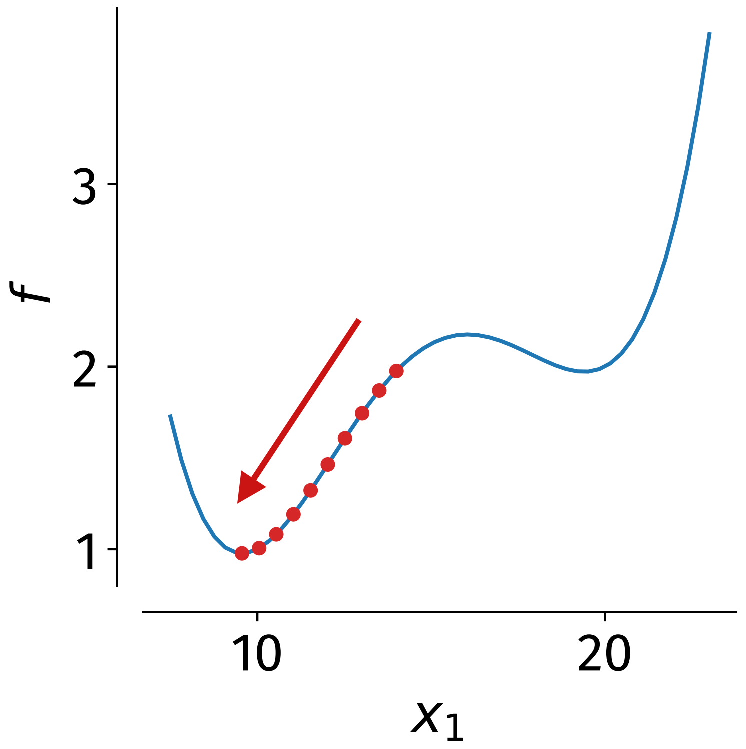
All values that if the gradient is followed reach a given minimum.

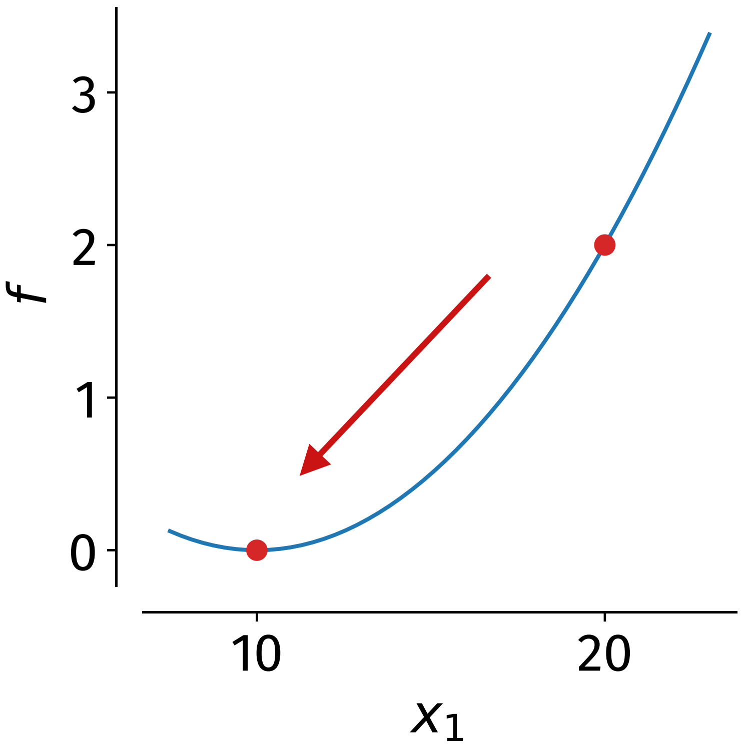
Definition: Quadratic region



All values where a Taylor series expansion up to second order around the minimum is a “good” approximation.

Definition: Iterative vs direct





Iterative

- Edging closer to the minimum
- Continue until close enough

Direct

- One-step optimization
- Analytical expression
- Note: “direct method” = no gradients

Optimization strategies

Pure strategies:

- Follow gradient and/or Hessian	(Quasi-)Newton methods
- Reduce dimensionality	Subspace methods
- (Quasi-)randomly pick points	Stochastic optimisation
- Regularly pick points	Grid refinement
- …	

Hybrid:

- Problem specific
- Typically global optimization
  - E.g. stochastic first, then Newton

Series notation for iterative approaches:

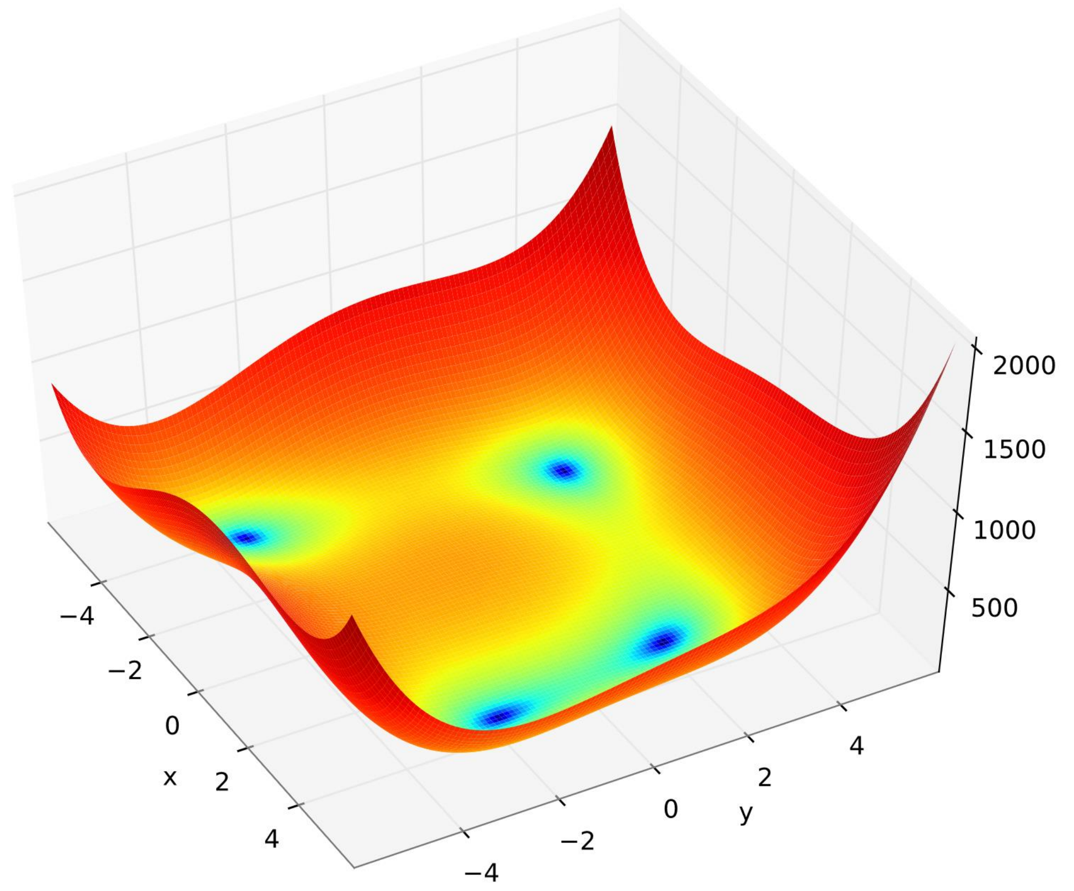
$$\{a_n\} \qquad \lim_{n \to \infty} = x_0$$

Follow gradient and/or Hessian

When to use
- Local minima
- Reasonable initial guess
- Wide attractive basins

When not to use
- Noisy function evaluations
- High dimensionality

Popular representatives
- Newton
- Steepest descent
- BFGS
  `scipy.optimize.minimize(method='BFGS')`
- L-BFGS
  `scipy.optimize.minimize(method='L-BFGS-B')`

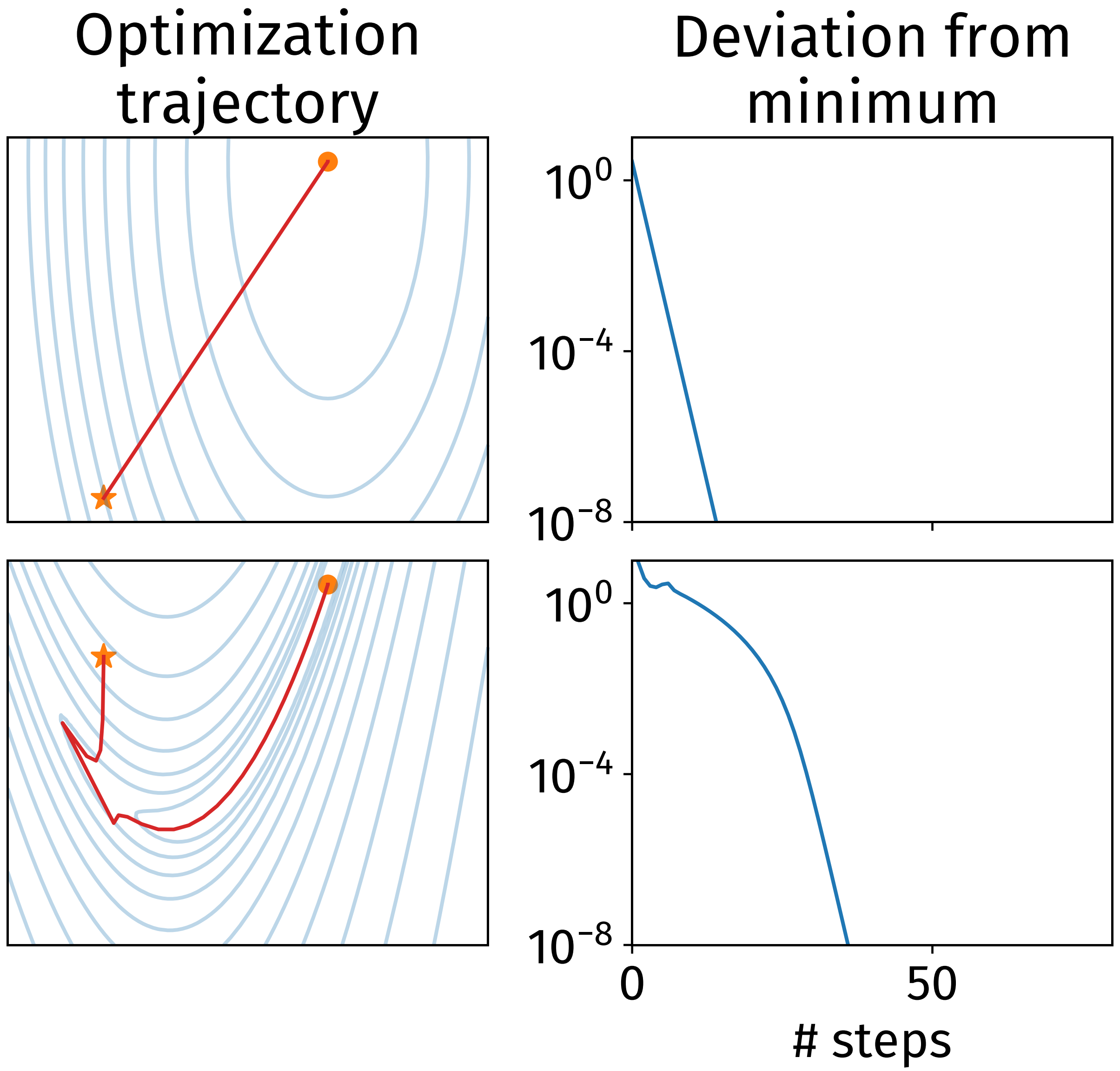


Newton's method

$$a_n - s\left[\nabla^2 f(a_n)\right]^{-1}\nabla f(a_n)$$

Variants
- Scale step size s
- Stochastic Newton

Problems
- Large Hessian and inversion expensive
- Slow with a fixed step



Steepest descent

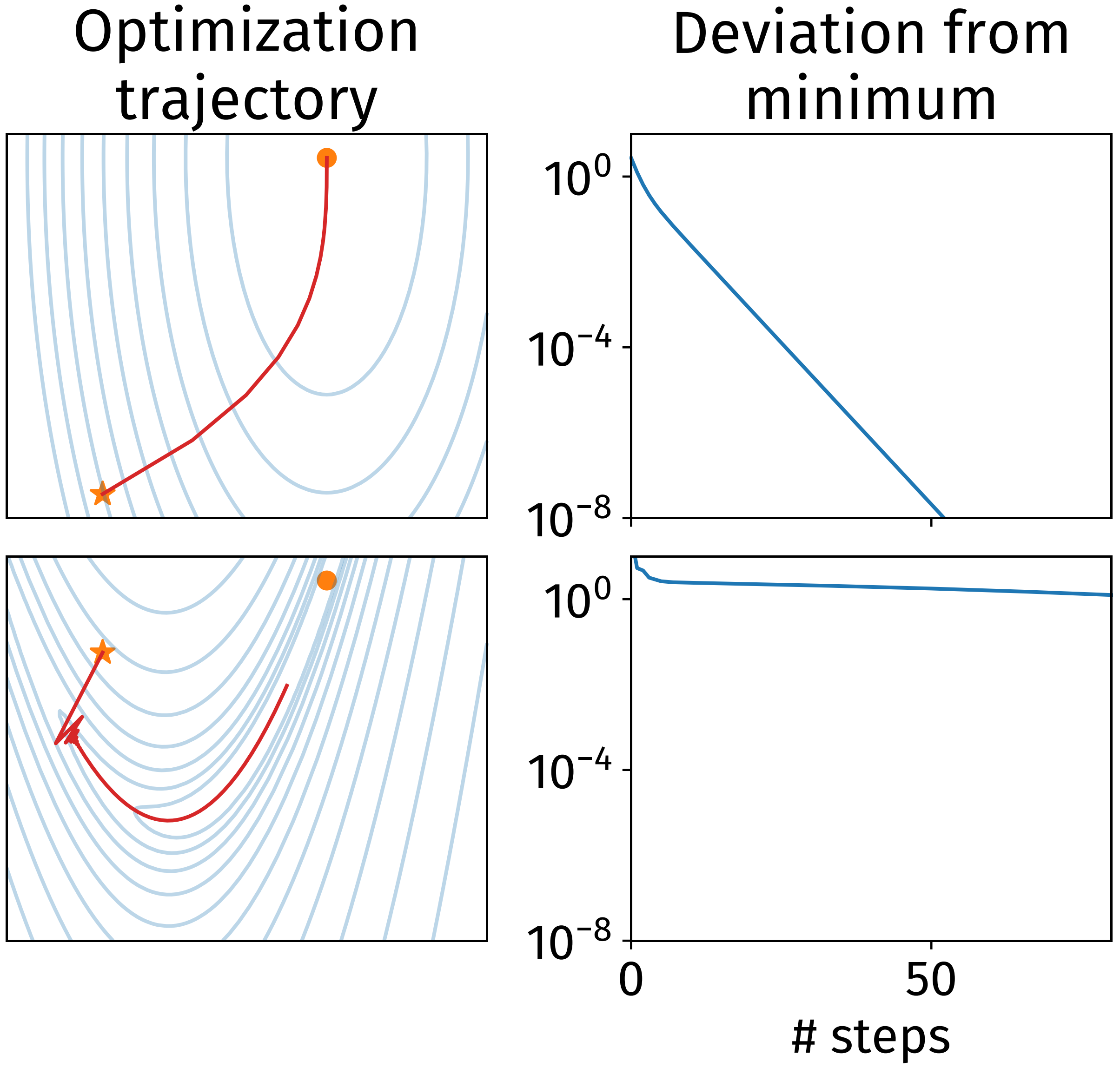
$$a_n - s\nabla f(a_n)$$

Variants

- Adjust step size
- Line search

Problems

- Slow with fixed step
- Oscillations



BFGS

Like Newton's method

$$p_{n+1} = -B_n^{-1} \nabla f(a_n)$$

Line search

$$\alpha_{n+1} = \arg\min f(a_n + \alpha p_{n+1}) \qquad s_{n+1} = \alpha_{n+1} p_{n+1}$$

Update optimisation

$$a_{n+1} = a_n + s_{n+1}$$

Get gradient response

$$y_{n+1} = \nabla f(a_{n+1}) - \nabla f(a_n)$$

Update approximate Hessian

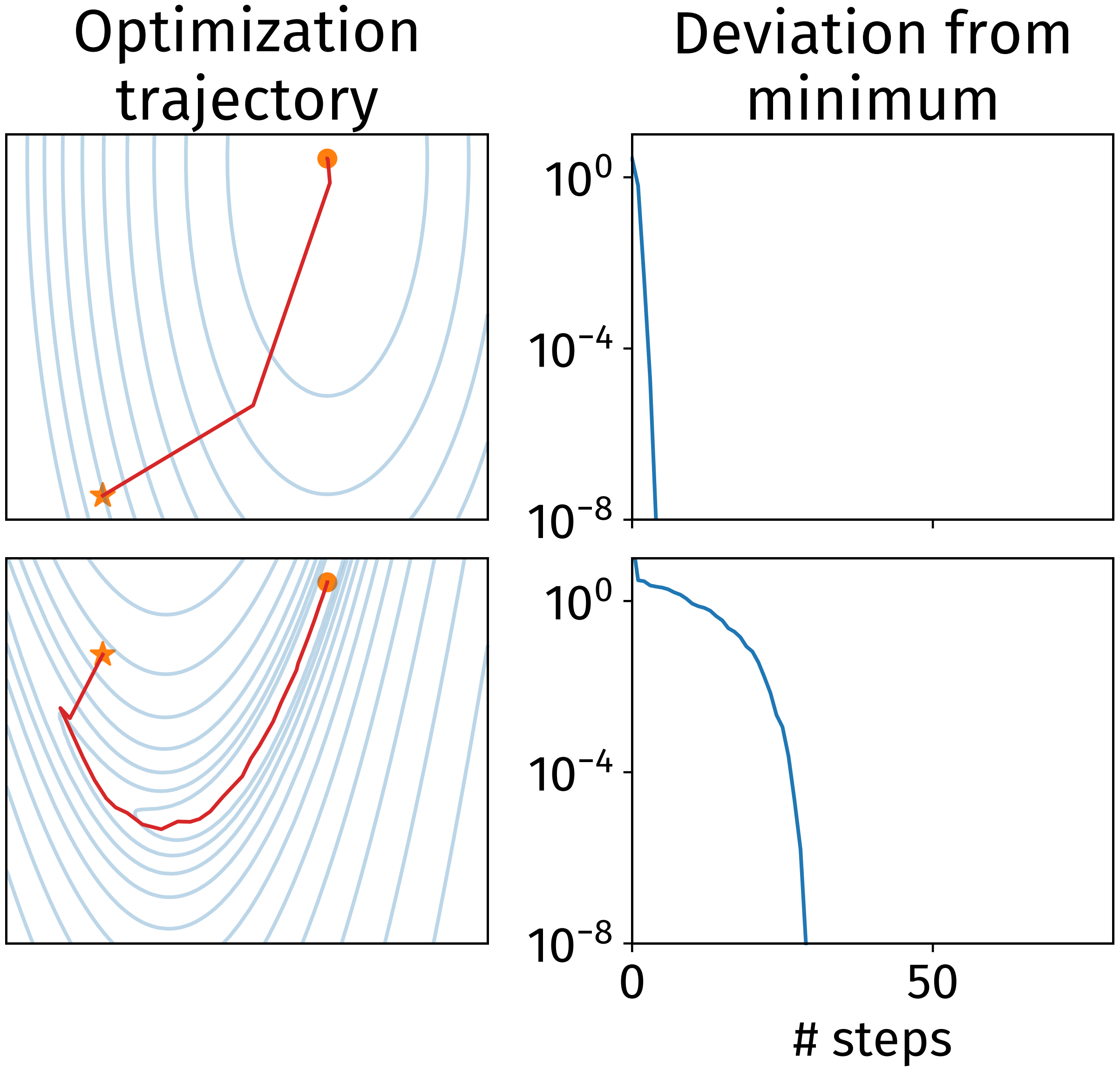
$$B_{n+1} = B_n + \frac{y_{n+1} y_{n+1}^{\mathrm{T}}}{y_{n+1}^{\mathrm{T}} s_{n+1}} - \frac{B_n s_{n+1} s_{n+1}^{\mathrm{T}} B_n^{\mathrm{T}}}{s_{n+1}^{\mathrm{T}} B_n s_{n+1}}$$

BFGS

Newton with approximate Hessian

Variants
- L-BFGS keeping only subset of Hessian

Problems
- Approximate Hessian update expensive
- High memory requirements



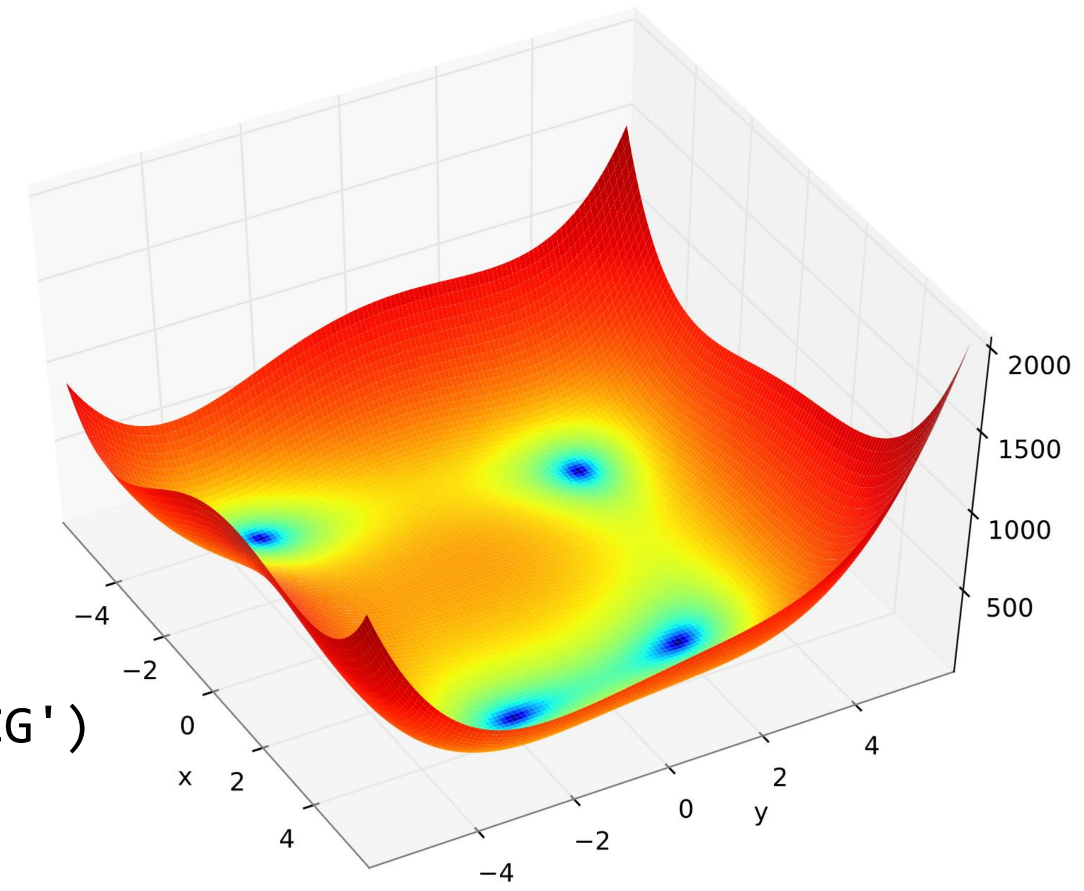
Subspace methods

When to use
- Local minima
- Reasonable initial guess
- High dimensionality

When not to use
- Noisy function evaluations

Popular representatives
- Conjugate Gradients

```
        scipy.optimize.minimize(method='CG')
```



CG (Fletcher-Reeves)

Initialise

$$p_0 = -\nabla f(a_n)$$

Line search

$$\alpha_{n+1} = \arg\min f(a_n + \alpha p_n)$$

Update optimisation

$$a_{n+1} = a_n + \alpha_{n+1} p_n$$

New problem-orthogonal search direction

$$\beta_{n+1} = \frac{||\nabla f(a_{n+1})||^2}{||\nabla f(a_n)||^2}$$

$$p_{n+1} = -\nabla f(a_{n+1}) + \beta_{n+1} p_n$$

CG

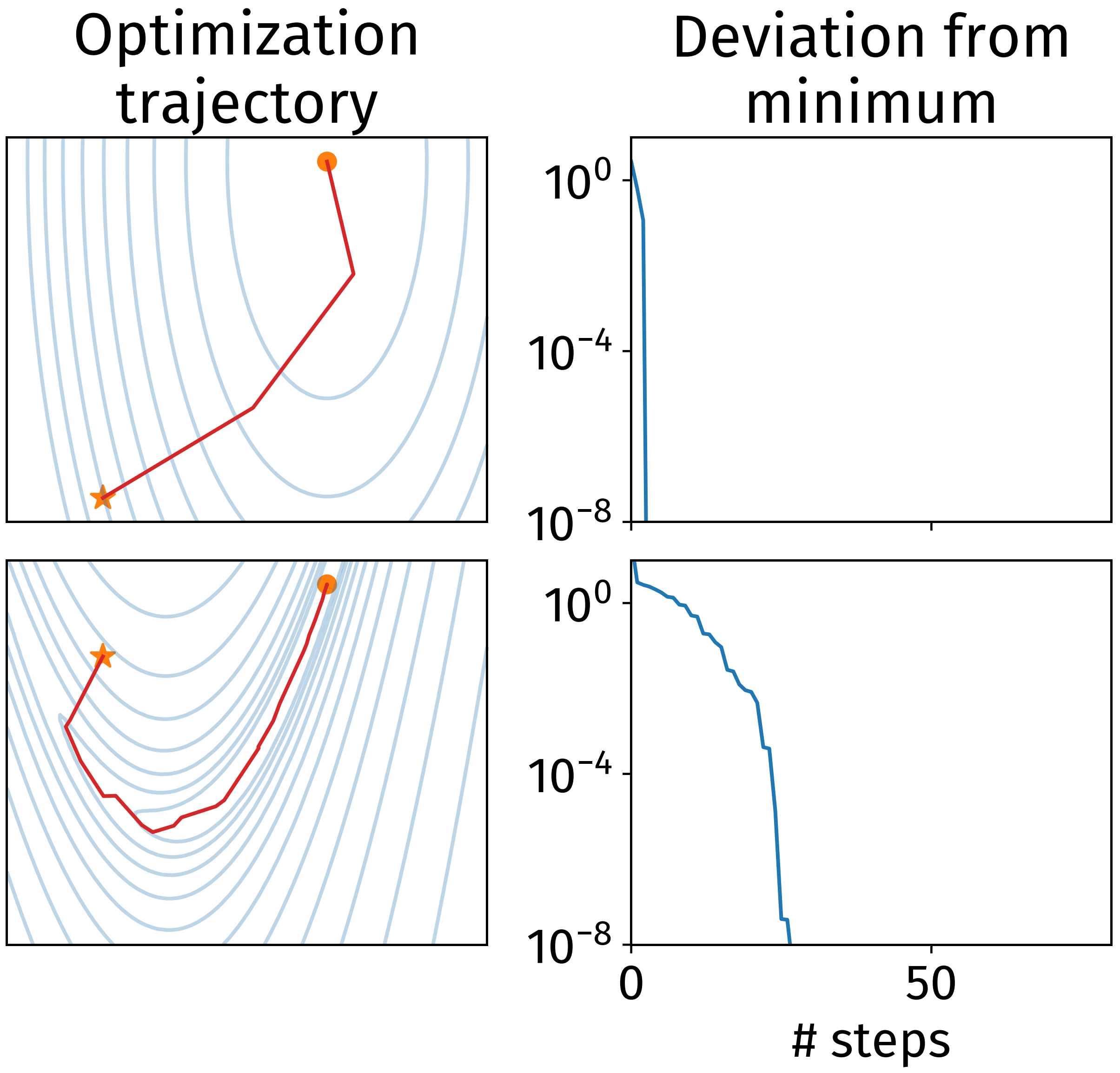
Subsequent residual minimisation

Variants

- Other search directions β

Problems

- Numerical stability: restart



Stochastic optimization family

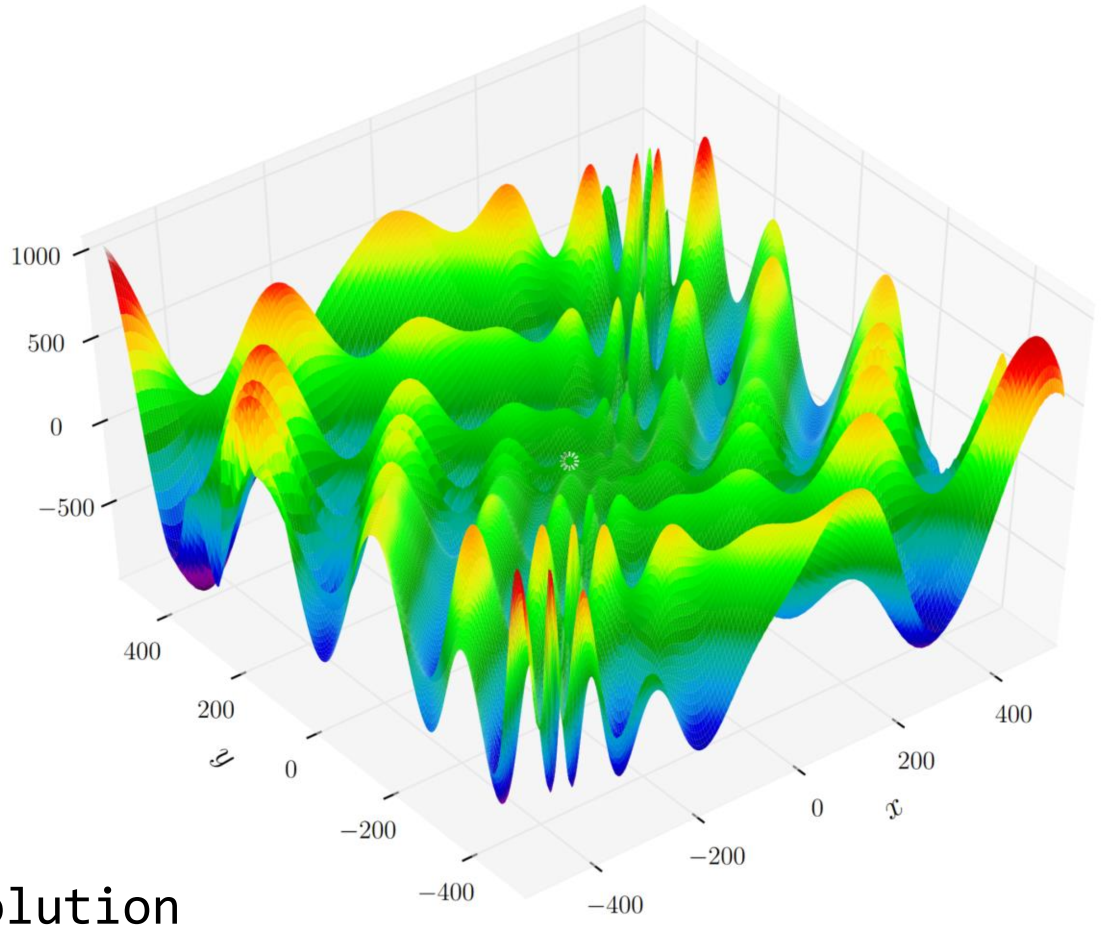
When to use
- Large domain
- Highly non-linear
- Small attractive basins
- Many minima
- High dimensionality

When not to use
- (Cheap) gradients available

Popular representatives
- Simulated annealing

  `scipy.optimize.basinhopping`
- Genetic algorithms

  `scipy.optimize.differential_evolution`

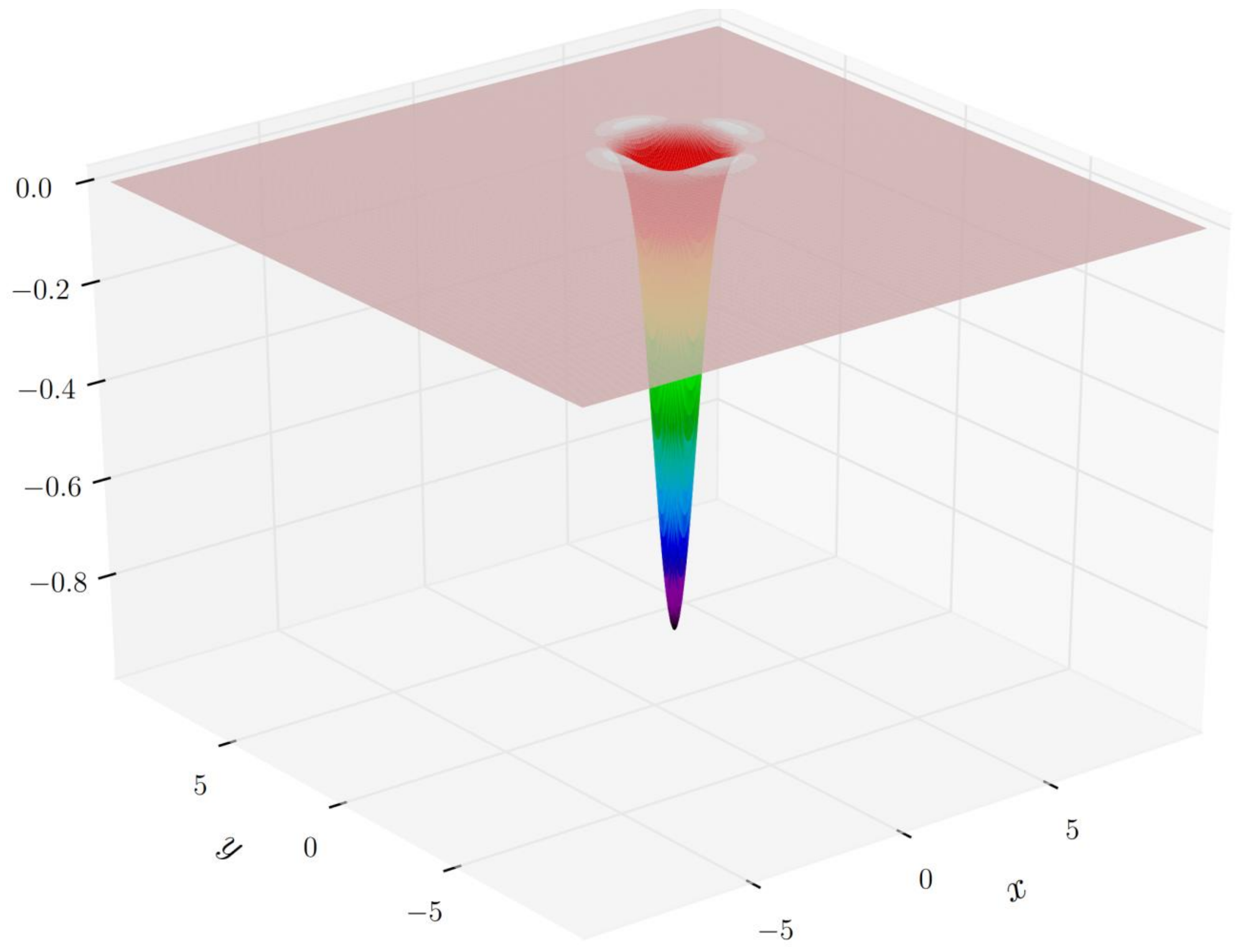


Grid refinement

When to use
- Low dimensionality
- Bounded domain
- Parallel resources

When not to use
- High dimensionality

Popular representatives
- Grid search
  `scipy.optimize.brute`
- Newton-Raphson (on 1D-gradient)
  `scipy.optimize.newton`



Derivatives in Chemistry

Equivalents
- f(x): total energy $E(\mathbf{R}_i, Z_i)$

Gradients
- Molecular forces
  - Commonly implemented
  - Special derivations
- Alchemical derivatives: electronic electrostatic potential

Hessians
- Normal modes
  - Commonly only for spatial derivatives
  - Special derivations (less often)

What if we have no derivatives?

Finite differences (FD)

Approximative derivatives

- Similar to the limit expression

Variants

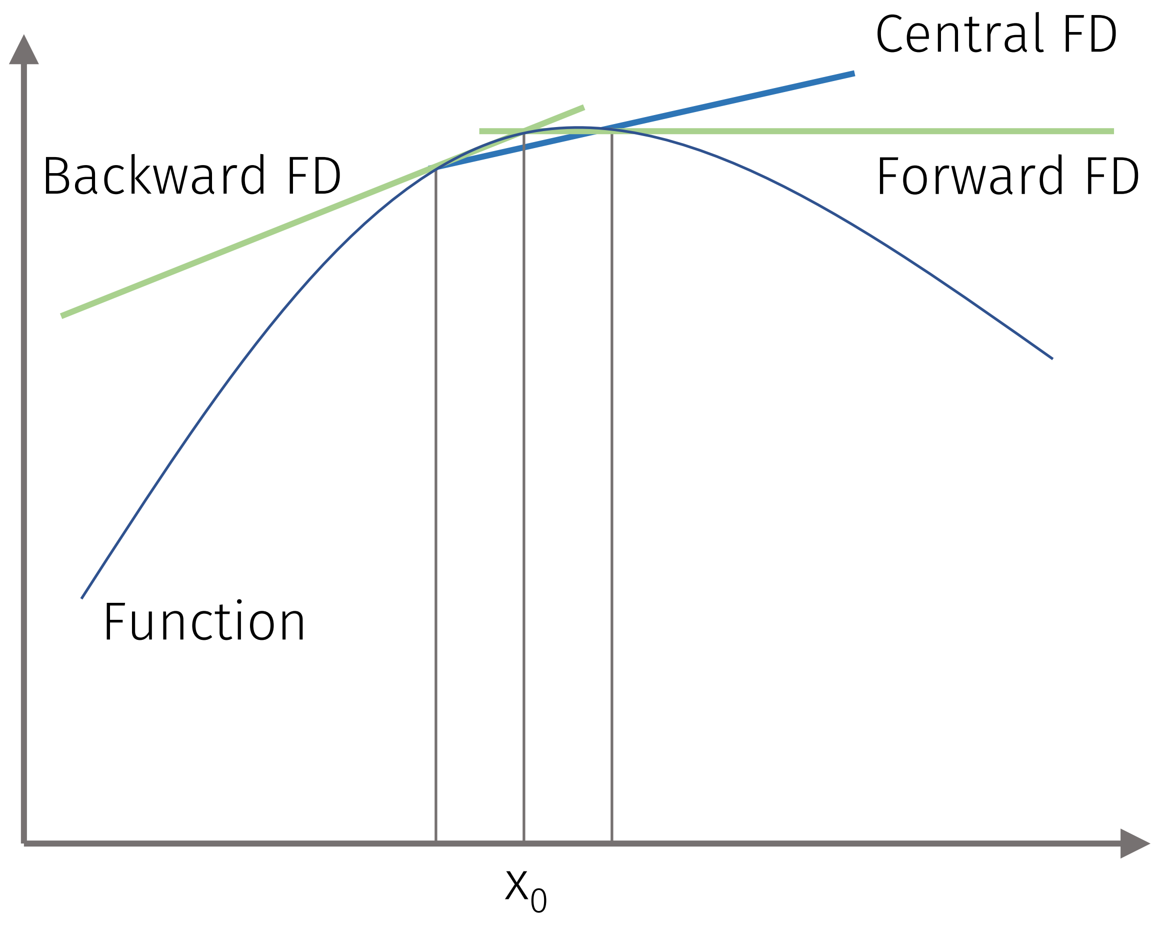
- Forward/backward
- Central
- Higher-order

Issues

- Finite displacement
- Numerical stability / finite precision
- Many calculations

Main advantage

- General applicability
- Just points and weights

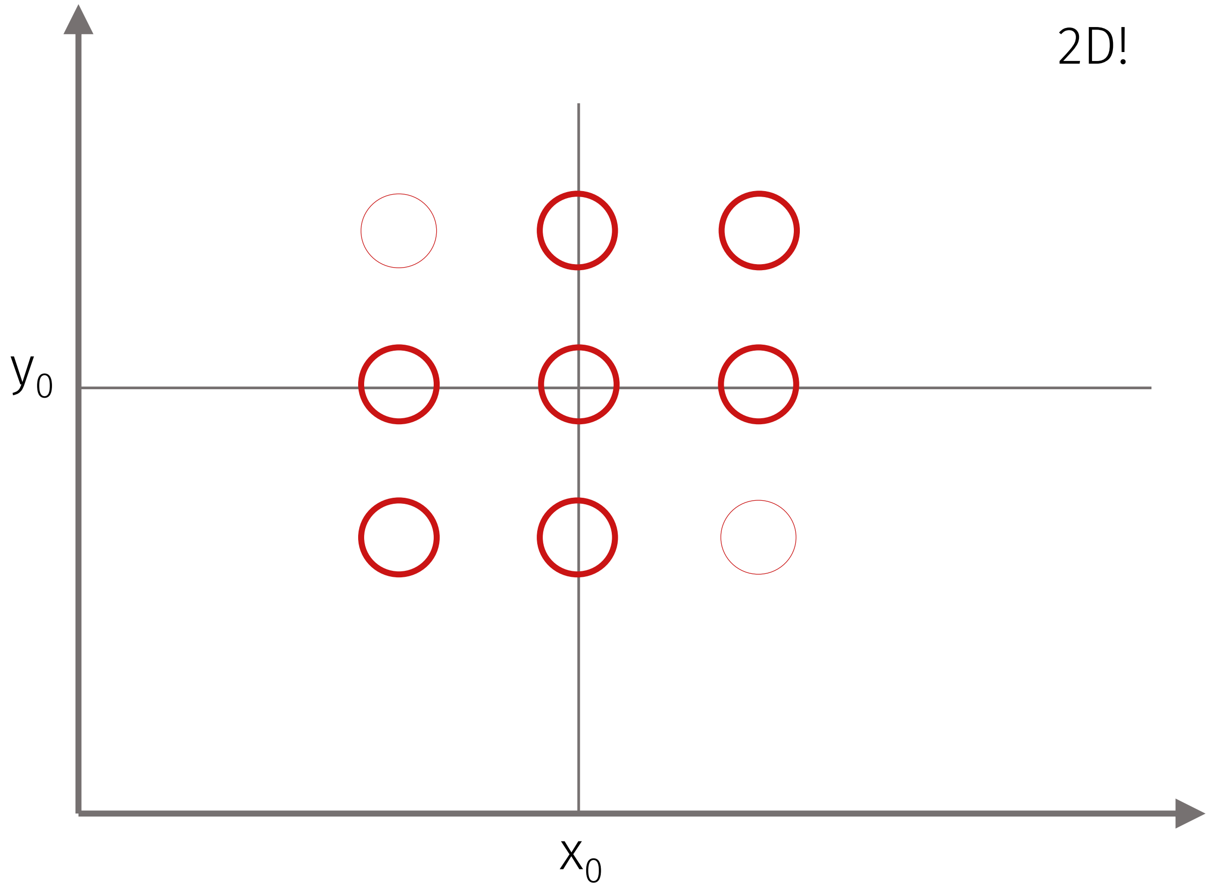


Finite differences (FD)

Approximative derivatives
- Similar to the limit expression

Higher dimensions
- Even more points
- Set of points: stencil
- Balance quality and cost

Weights
- Choose points first
- Taylor expansion around center
- Solve set of linear equations

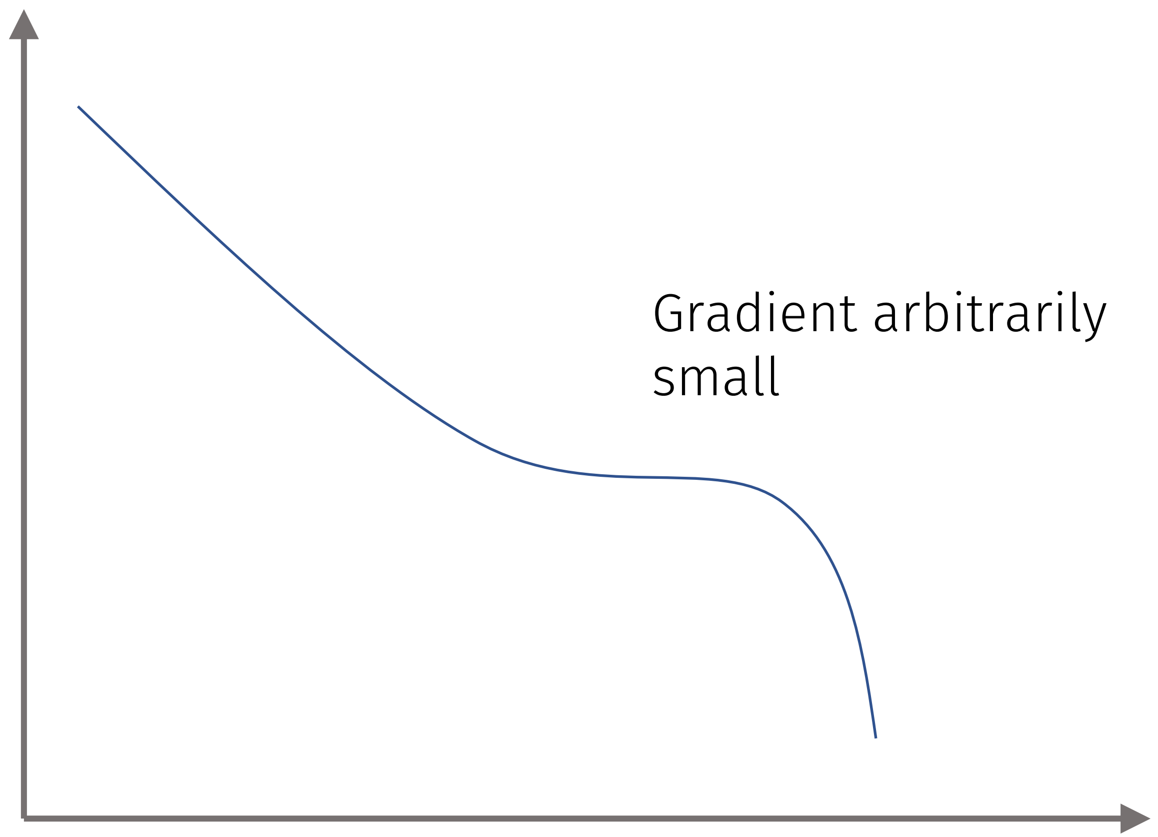


Optimization: Caveats

Convergence
- Hard to establish
- Gradient necessary, but not sufficient
- Hessian expensive
- Local property

Numerical stability
- Finite differences
- Conjugate Gradients
- Shallow minima

Cost of Hessians
- Scales as N^2
  - Water: N=9
  - Caffeine: N=72
- Often only from finite differences



Optimization: Caveats

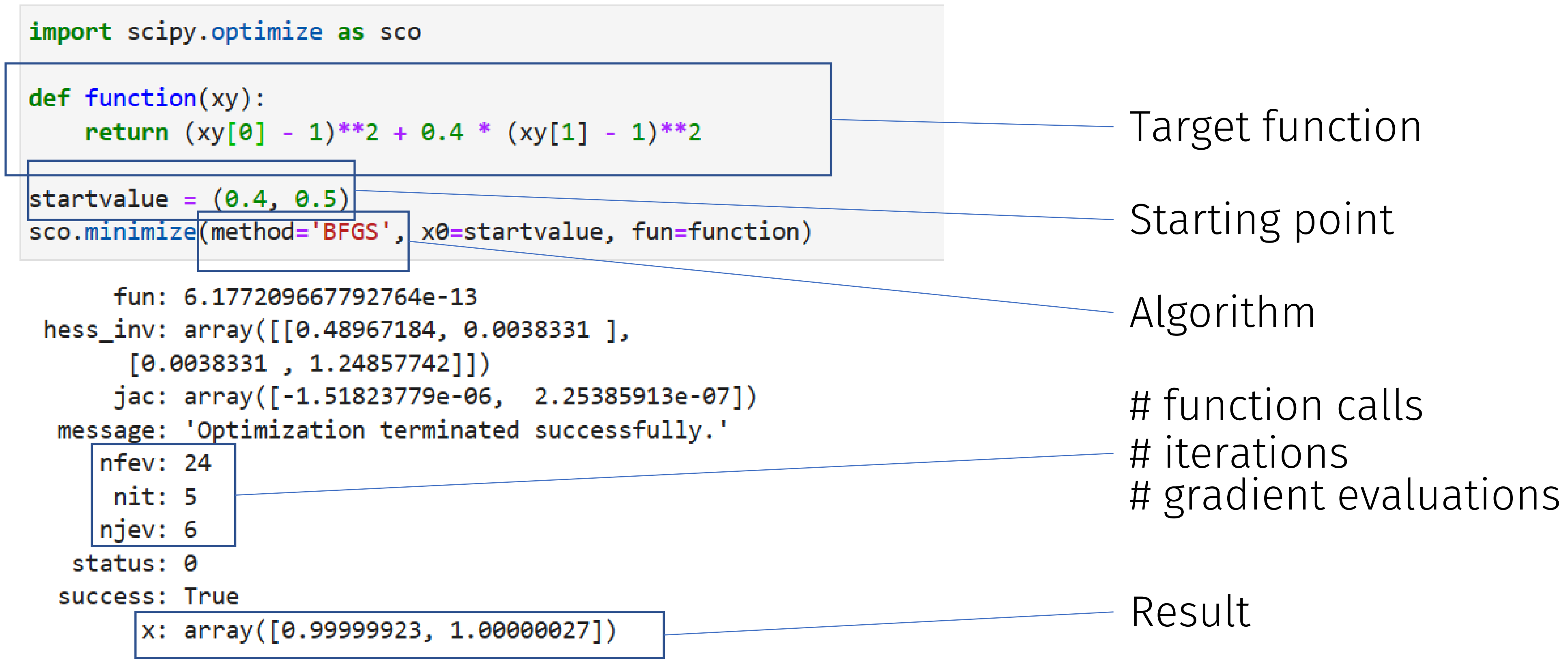
Curse of dimensionality
- Search space quickly increases
- Often forces tiny optimization steps

Preconditioning
- Math not equal to finite-precision implementations
- Transform problem into an equivalent one
- Focus on numerical stability
- Key: use libraries when possible or implement algorithms verbatim

Optimization in python

Numerical libraries in python

- numpy vectorized math
- scipy scientific algorithms
  - scipy.optimize



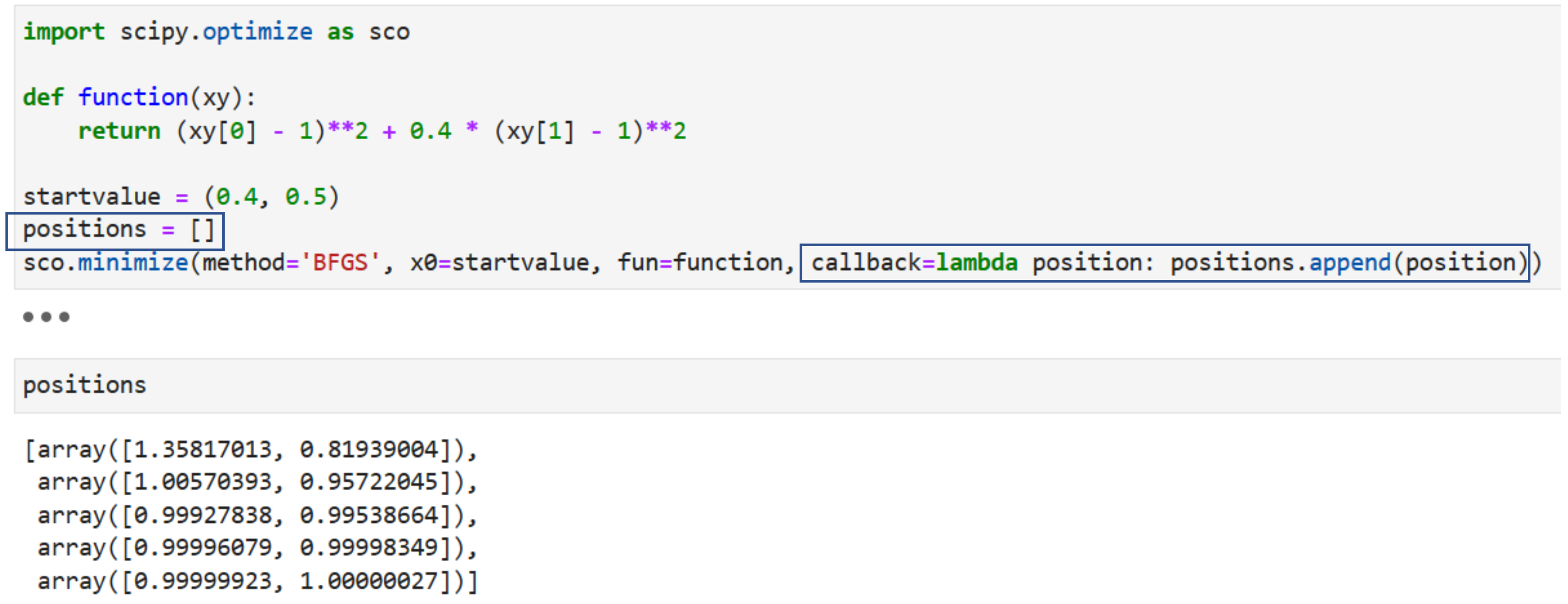
```python
import scipy.optimize as sco

def function(xy):
    return (xy[0] - 1)**2 + 0.4 * (xy[1] - 1)**2

startvalue = (0.4, 0.5)
sco.minimize(method='BFGS', x0=startvalue, fun=function)
```

```
      fun: 6.177209667792764e-13
 hess_inv: array([[0.48967184, 0.0038331 ],
       [0.0038331 , 1.24857742]])
      jac: array([-1.51823779e-06,  2.25385913e-07])
  message: 'Optimization terminated successfully.'
     nfev: 24
      nit: 5
     njev: 6
   status: 0
  success: True
        x: array([0.99999923, 1.00000027])
```

Optimization in python

Recording optimization progress

- One entry *after* each step
- Starting point not included



```
import scipy.optimize as sco

def function(xy):
    return (xy[0] - 1)**2 + 0.4 * (xy[1] - 1)**2

startvalue = (0.4, 0.5)
positions = []
sco.minimize(method='BFGS', x0=startvalue, fun=function, callback=lambda position: positions.append(position))
```

•••

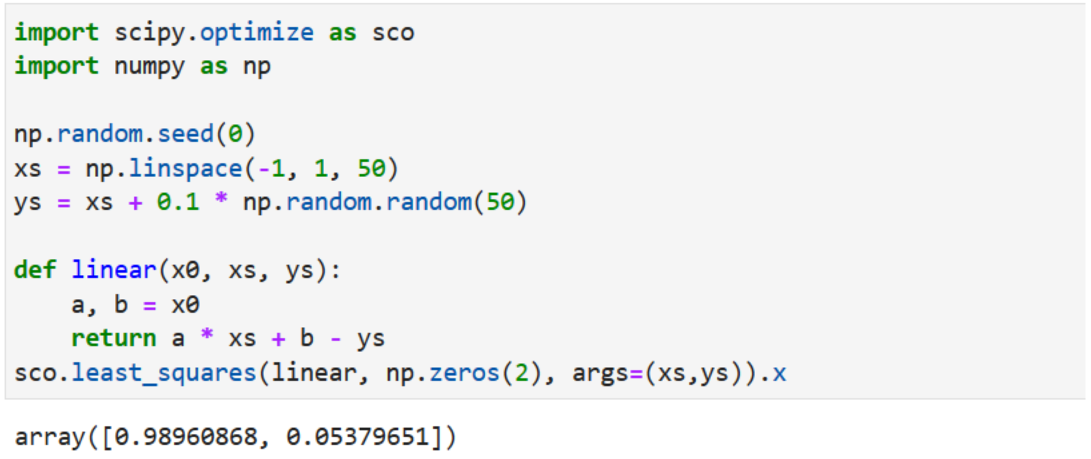
```
positions
```

```
[array([1.35817013, 0.81939004]),
 array([1.00570393, 0.95722045]),
 array([0.99927838, 0.99538664]),
 array([0.99996079, 0.99998349]),
 array([0.99999923, 1.00000027])]
```

Optimization in python

Least-squares fit to a function

- First argument: parameter vector
- Return residuals



```python
import scipy.optimize as sco
import numpy as np

np.random.seed(0)
xs = np.linspace(-1, 1, 50)
ys = xs + 0.1 * np.random.random(50)

def linear(x0, xs, ys):
    a, b = x0
    return a * xs + b - ys
sco.least_squares(linear, np.zeros(2), args=(xs,ys)).x
```

```
array([0.98960868, 0.05379651])
```

Summary

Algorithms
- (Quasi-)Newton methods
- Subspace methods
- Stochastic optimisation
- Grid refinement

Caveats
- Convergence
- Cost of Hessians
- Preconditioning

Python
- Optimization
- Curve fitting

ferchault @ferchault guido.vonrudorff.de