Model robustness and error metrics

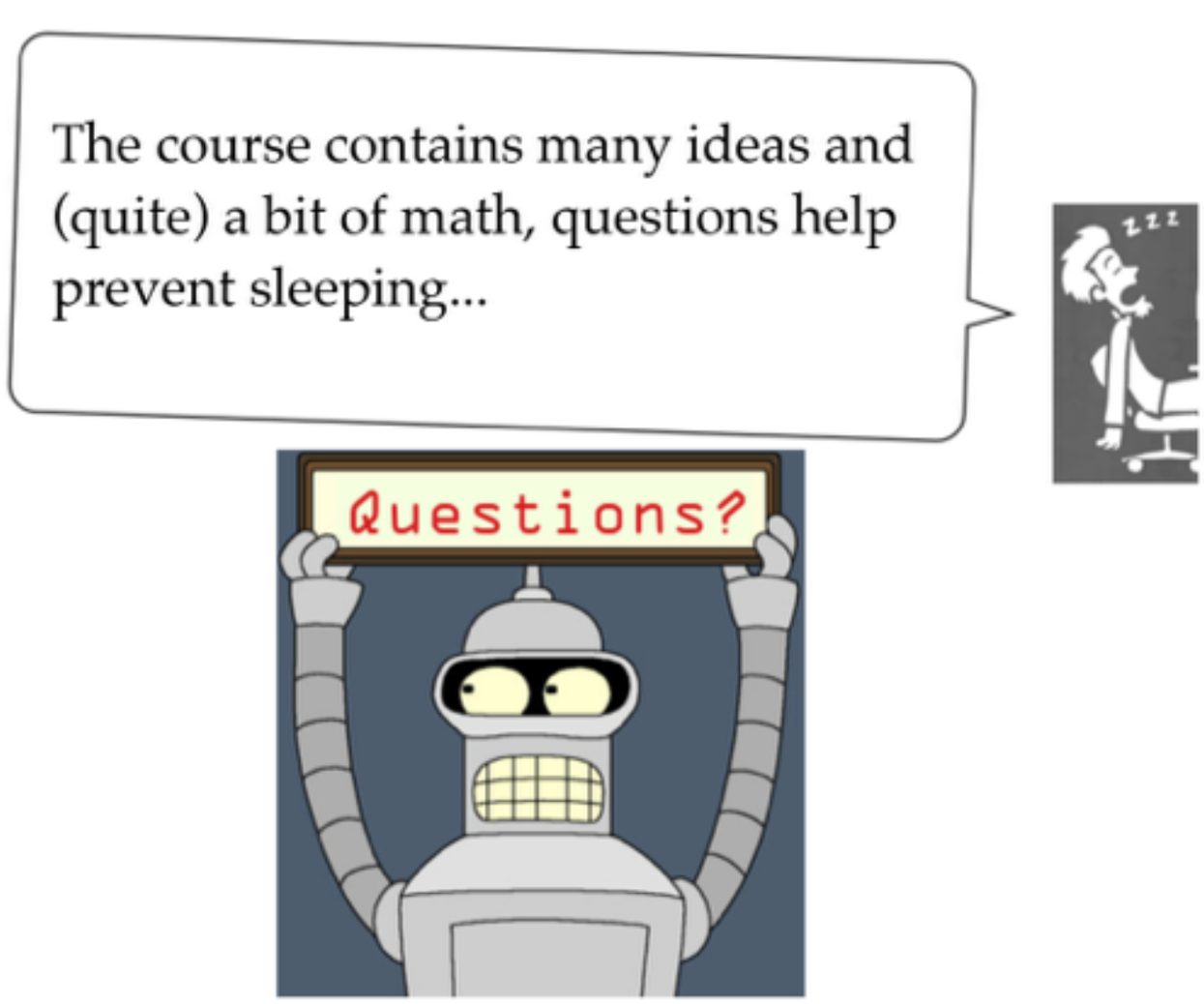
Valeriya Naumova

Simula Research Laboratory AS

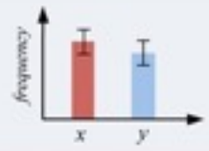
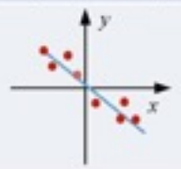
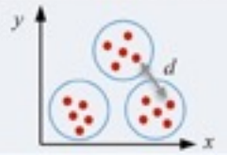
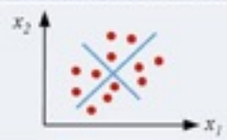
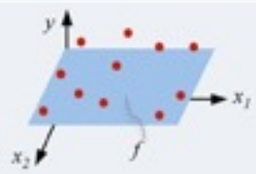
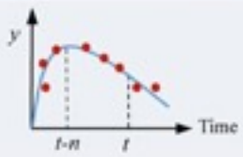
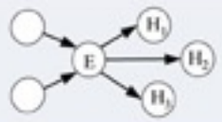
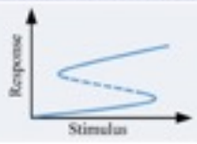
2018 Summer School in Computational Cardiac Physiology

Outline

1. Why mathematical models?
2. What is a parameter estimation problem?
3. How to estimate model parameters?
4. How well the parameters are determined?
5. How large are uncertainties?



Why model?

Testing for differences	**Significance Analysis**	Is there a difference between the two groups?
"before/after" "with/without"	$95\%\ \text{CI} = \overline{X} \pm \left(t_{(n-1)}(5\%) \times \overline{\text{SE}}\right)$	
Analysing covariaton	**Correlation Analysis**	Does a small/large value of x, coincide with a small/large value of y?
"coincides with"	$r_{x,y} = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2 \sum y_i^2}}$	
Identifying groups	**Cluster/Discriminant Analysis**	Can the data be grouped on the basis of features x and y?
"is similar to"	$D = \min_i \left\{ \min_j \left\{ \frac{d(i,j)}{\max_k d'(k)} \right\} \right\}$	
Condensing data	**Component/Factor Analysis**	Explain the variability through a weighted linear combination of principal components.
"reduces to"	$X = U \Sigma W^T$	
Fitting data	**Regression Analysis**	What is the predicted value of y, given x_1 and x_2?
"relates to"	$y = f(x_1, \ldots, x_n)$	
Numerical predictions	**Time Series Analysis**	What is the predicted value of y, knowing past values of y?
"follows"	$y(t) = f(y(t-1), y(t-2), \ldots)$	
Analyzing influences	**Bayesian (network) Analysis**	From the observed evidence, H_2 is most probable.
"Given E, the probability of H is"	$P(H_j \mid E) = \frac{\Pr(E \mid H_j)\Pr(H_j)}{\sum \Pr(E \mid H_j)\Pr(H_j)}$	
Investigating mechanisms	**Dynamical Systems Theory**	The response can be explained by a (bistable) switching mechanism.
"causally entails" "If ... Then ..."	$\frac{d}{dt} x_i = f_i(x_1(t), \ldots, x_n(t), u(t))$	

Ease of experiments

Power of explanation

Wolkenhauer, Frontier in Physiology, 2014.

Systems biology loop

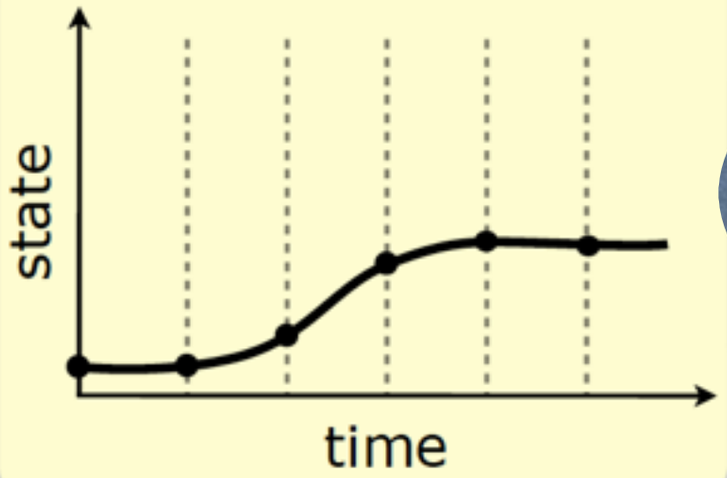
prior knowledge

e.g., ECG measurements

- *parameter optimisation*
- *uncertainty analysis*
- *model selection*

model calibration

population average data



Biological / physiological question

differential equation models

$$\dot{x} = Sv(x, \theta)$$

$$y = h(x, \theta)$$

prediction/ validation

What is a parameter estimation problem?

Parameter estimation concept

Scientists and modellers frequently wish to relate **physical/biological parameters** θ characterising a model to collected **observations** making up some data sets, y

We will assume that the fundamental physics/biology are adequately understood, may be specified

$$G(\theta) = y$$

Computing $G(\theta)$ might involve solving an ordinary differential equation or partial differential equation.

Parameter estimation concept

Example: Drug concentration dynamics

$$\dot{x}(t) = -p_1 x(t) + p_2 u(t), \quad x(0) = 0$$
$$y(t) = p_3 x(t)$$

$x(t)$ – drug concentration

$u(t)$ – test-input drug injection

$y(t)$ – drug concentration measurement

$\theta = (p_1, p_2, p_3)$ – model parameters

For given u(t), the solution is given as

$$y(t) = p_3 p_2 \int_0^1 e^{-p_1(t-s)} u(s) ds$$

For u(t) = $\delta(t)$

$$y(t) = \underbrace{p_3 p_2 e^{-p_1 t}}_{G(\theta)}$$

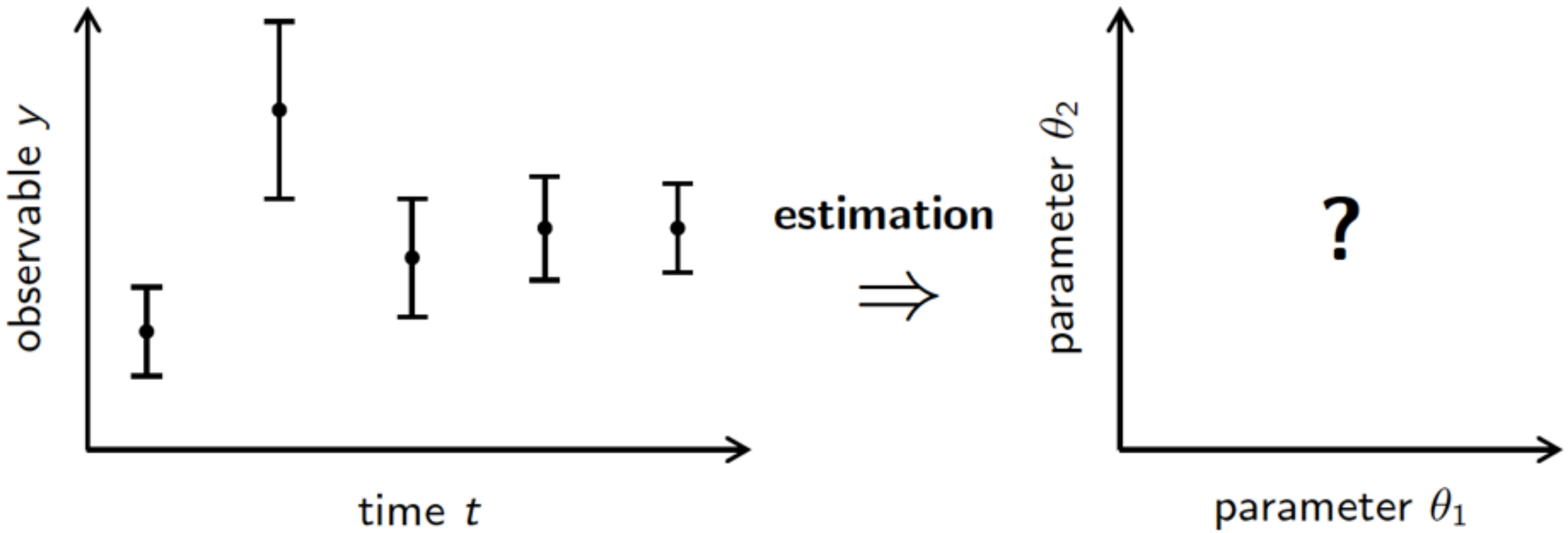
Parameter estimation concept

Forward problem

Given parameters θ find solution x and observable y.

Inverse problem

Given observable y find parameters θ.



Parameter estimation concept

Generic ODE model

Solution: $$\dot{x}(t) = f(x(t,\theta), \theta, u(t)), \quad x(t_0,\theta) = x_0$$

Observables: $$y(t,\theta) = h(x(t,\theta);\theta)$$

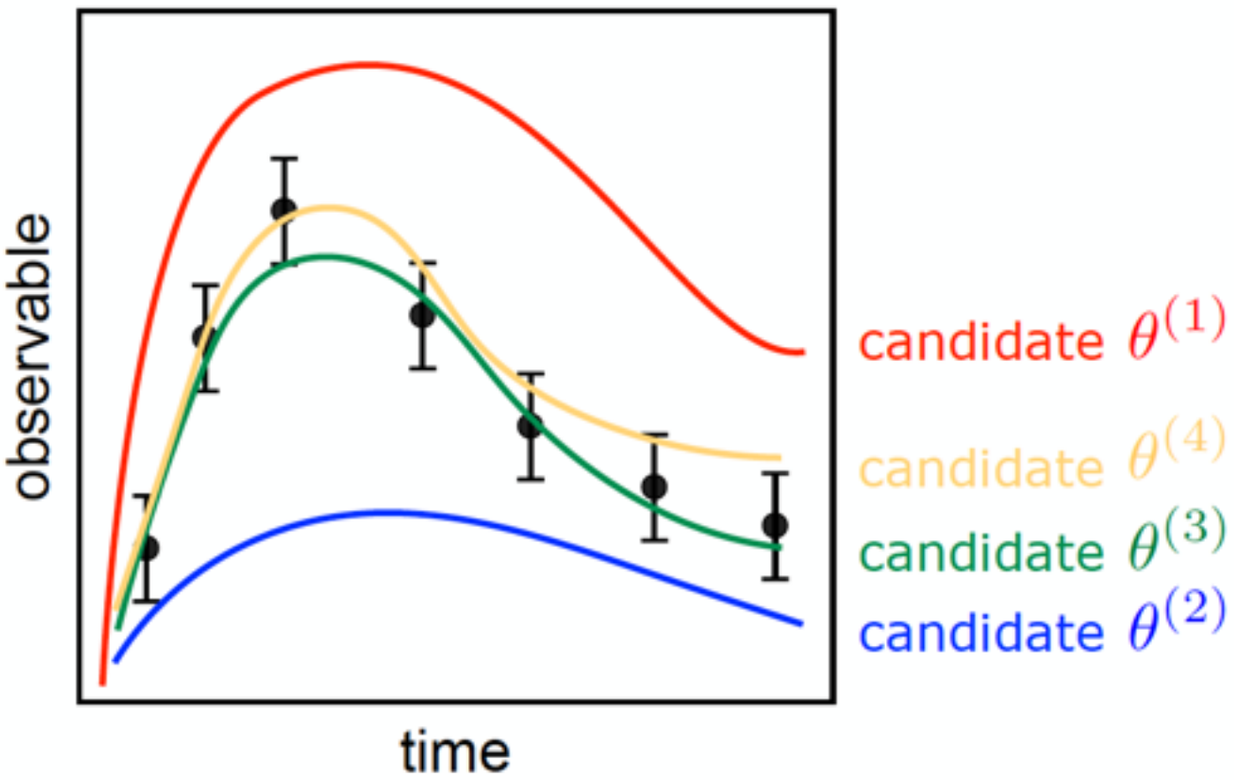
Issues in inverse problems:
- Solution existence
- Solution uniqueness
- Solution instability

Measurements: $$\bar{y}_j(t_i) = y_j(t_i,\theta) + \epsilon_j(t_i), \quad \epsilon_j(t_i) \sim \mathcal{N}(0, \sigma^2_{i,j})$$

Training set: $$\mathcal{D} = \{(\bar{y}_j)_{j=1}^n, t_i\}_{i=0}^T$$

Parameter estimation concept

Training set: $\mathcal{D} = \{(\bar{y}_j)_{j=1}^{n}, t_i\}_{i=0}^{T}$



How to distinguish qualitatively?

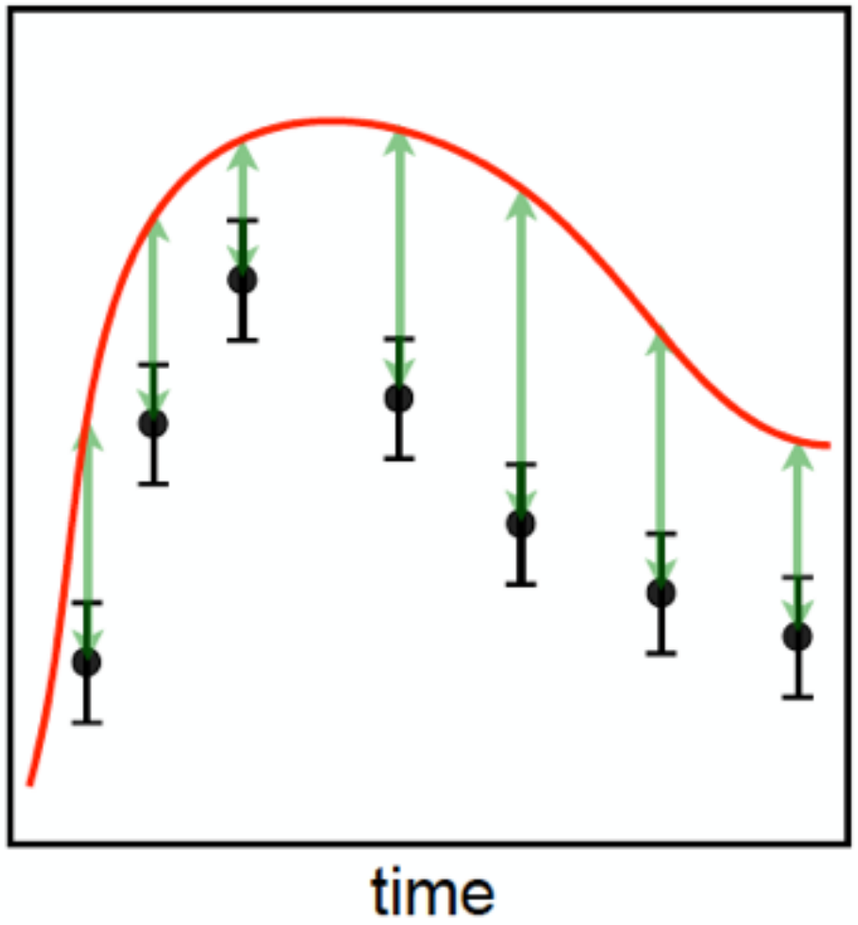
Least squares approach for parameter estimation:

$$J(\theta) = \int_{t_0}^{T} [\bar{y}(t) - y(t,\theta)]^2 dt = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} \left(\frac{\bar{y}_j(t_i) - y_j(t_i,\theta)}{\sigma_{ij}} \right)^2$$

Parameter estimation concept

Least squares approach for parameter estimation:

$$J(\theta) = \int_{t_0}^{T} [\bar{y}(t) - y(t,\theta)]^2 dt = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=0}^{T} \left(\frac{\bar{y}_j(t_i) - y_j(t_i,\theta)}{\sigma_{ij}} \right)^2$$



1. Simulation of mechanistic model
2. Calculation of distance between simulation results and data
3. Evaluation of sum-of-squares

$$J(\theta) = \quad^{2}\quad^{2}\quad^{2}\quad^{2}\quad^{2}\quad^{2}\quad^{2}$$
$$\qquad\quad + \quad + \quad + \quad + \quad + \quad +$$

Goal: Construct a sequence of points along which the fit improves

Parameter estimation concept

Least squares approach:

$$J(\theta) = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=0}^{T} \left(\frac{\bar{y}_j(t_i) - y_j(t_i, \theta)}{\sigma_{ij}} \right)^2$$

Optimisation problem:

$$\min_{\theta} J(\theta)$$

- nonlinear
- non convex

  => multiple local minima….

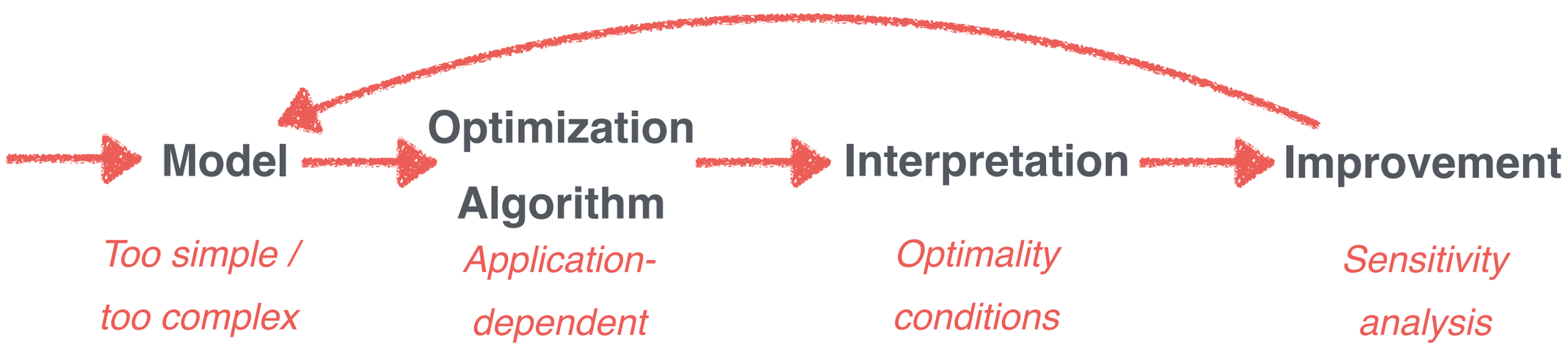
  => how to evaluate function gradients?

10-min introduction to fundamentals in optimisation

Introduction

"fascinating blend of theory and computation, heuristics and rigour"
R. Fletcher, 2000

- **Optimization** is an important tool in decision science and in the analysis of physical systems.
- **Key ingredients**:
  - **objective** is a quantitative measure of the performance of the system under study (time, potential energy, etc.) that we want to minimize or maximize;
  - **variables** or **unknowns** are the components of the system for which we want to find values;
  - **constraints** are the functions that describe the relationships among the variables and that define the allowable values for the variables.



Notation and Definition

Optimization Problem

$$\min_{x \in \mathbb{R}^n} f_0(x) \quad \text{subject to} \quad \begin{aligned} f_i(x) &= 0, i \in \mathcal{E} \\ f_i(x) &\geq 0, i \in \mathcal{I} \end{aligned}$$

$x = (x_1, \ldots, x_n)$: optimization variables;

$f_0 : \mathbb{R}^n \to \mathbb{R}$: objective function;

$f_i : \mathbb{R}^n \to \mathbb{R}, \quad i = 1, \ldots, m$: constraint functions

$\mathcal{I}$ and $\mathcal{E}$ are sets of indices for equality and inequality constraints, respectively

$$\max f = \quad \min -f$$

Feasible region:

set of points satisfying all constraints

Optimal solution:

x^* has smallest value of f_0 among all vectors that satisfy the constraint

Notation and Definition: Examples

Data Fitting

- **variables**: model parameters
- **constraints**: prior information, parameter limits
- **objective**: measure of misfit or prediction error

Portfolio Optimization

- **variables**: amount invested in different assets
- **constraints**: budget, max/min investment per asset, minimum return
- **objective**: overall risk or return variance

Notation and Definition: Examples

The Transportation Problem

A pharma company has *2 factories* F_1 *and* F_2 and a dozen retail outlets (pharmacies) $R_1, R_2, \ldots R_{12}$.

Each factory F_i can produce a_i quantity of certain antiarrhythmic pills each week; a_i called the capacity of the plant.

Each retail outlet R_j has a known *weekly demand* of b_j quantity of the product. The cost of shipping of one quantity of the pill from factory F_i to retail outlet R_j is c_{ij}.

Problem: determine how much of the product to ship from each factory to each outlet so as to satisfy all the requirements and minimize costs.

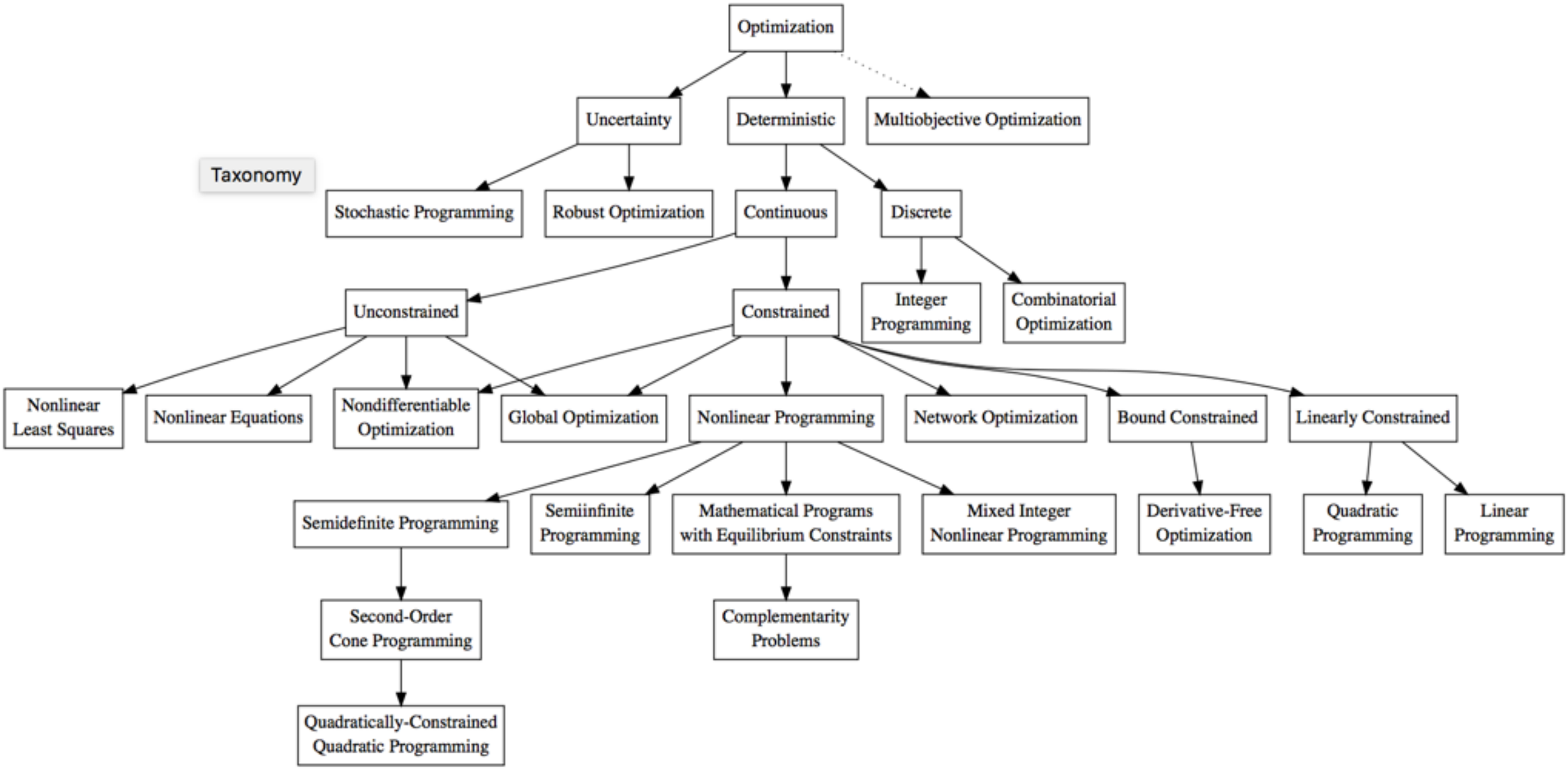
→ **Linear programming problem**
(objective function and the constraints are all linear functions)

Notation and Definition

Optimization Types and Main Concepts

- **Continuous versus Discrete Optimization (**integer programming problems**)**
- **Constrained and Unconstrained Optimization**
  - **Unconstrained:** $\mathcal{I} = \mathcal{E} = \emptyset$
  - **Constrained:** Linear programming / nonlinear programming problems
- **Global and Local Optimization**
- **Stochastic and Deterministic Optimization**
- **Convexity**
  - **Convex programming:** Objective function is convex / equality constraints are linear / inequality constraints are concave
- **Optimization Algorithms (iterative)**
  - Robustness / Efficiency / Accuracy

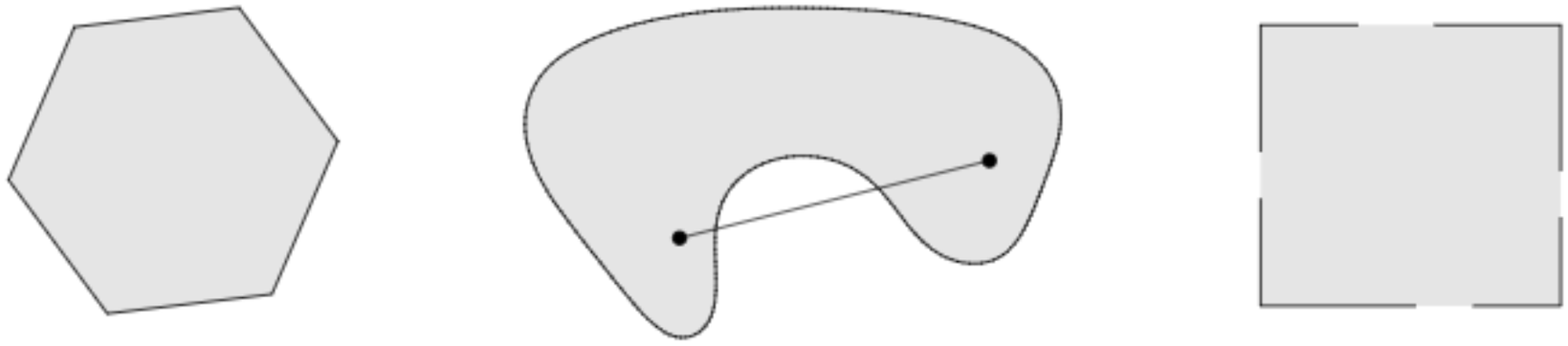
Optimization Tree



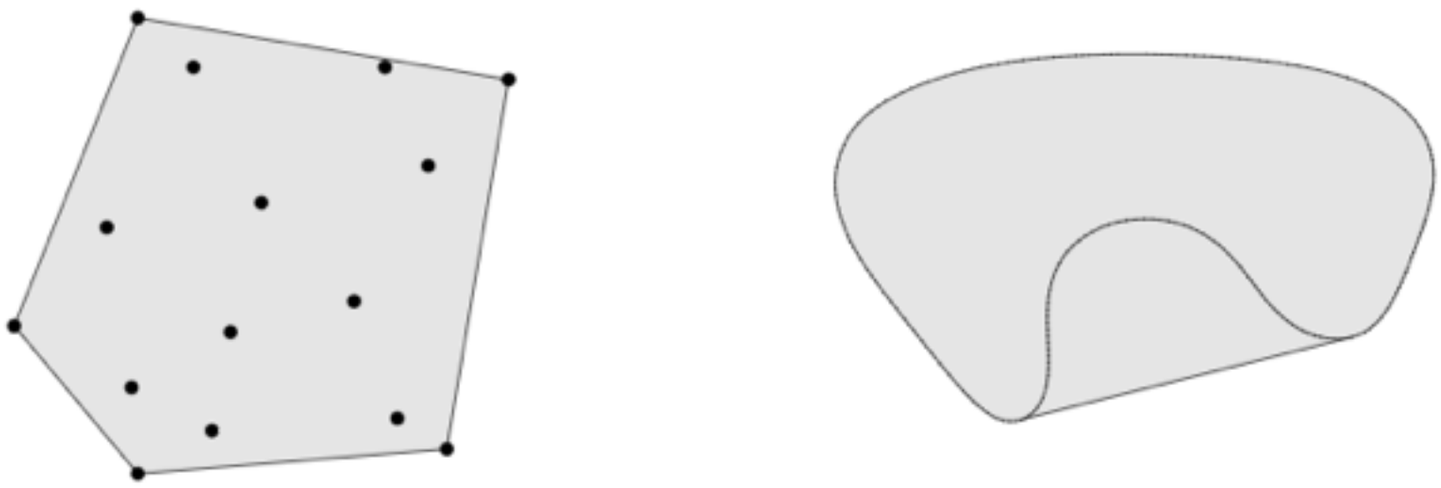
Notation and Definition

Convexity: Sets

$$x, y \in S, 0 \leq \alpha \leq 1 \Longrightarrow \alpha x + (1 - \alpha) y \in S$$



one convex, two nonconvex sets



Convex hull: set of all convex combinations of points in the set

Notation and Definition

Convexity: Functions

$f : \mathbb{R}^n \to \mathbb{R}$ is convex if its domain is a convex set and

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y),\ 0 \leq \alpha \leq 1$$

f is concave if $-f$ is convex

f is strictly convex if its domain is convex and

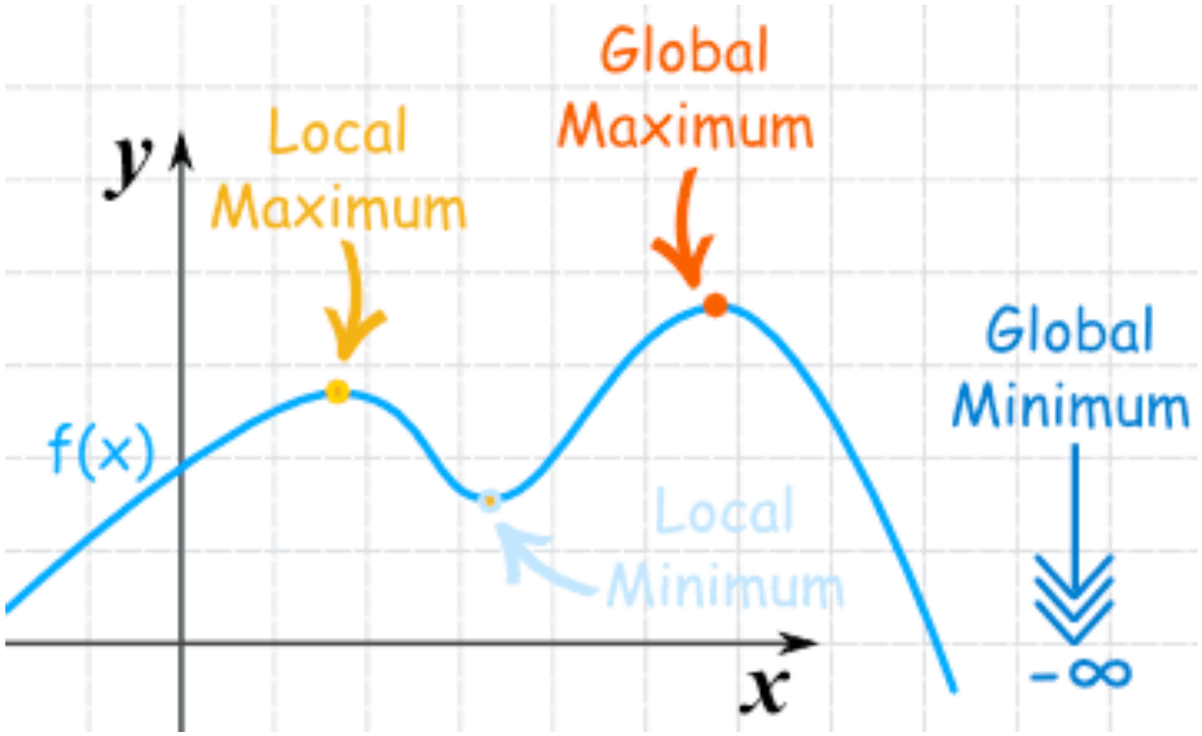
$$f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y),\ x \neq y,\ 0 < \alpha < 1$$

Notation and Definition

Unconstrained Optimization. What is a solution?

$$\min_x f(x)$$

where $x \in \mathbb{R}^n,\ n \geq 1$ is a real vector and $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth function.



Global minimizer:

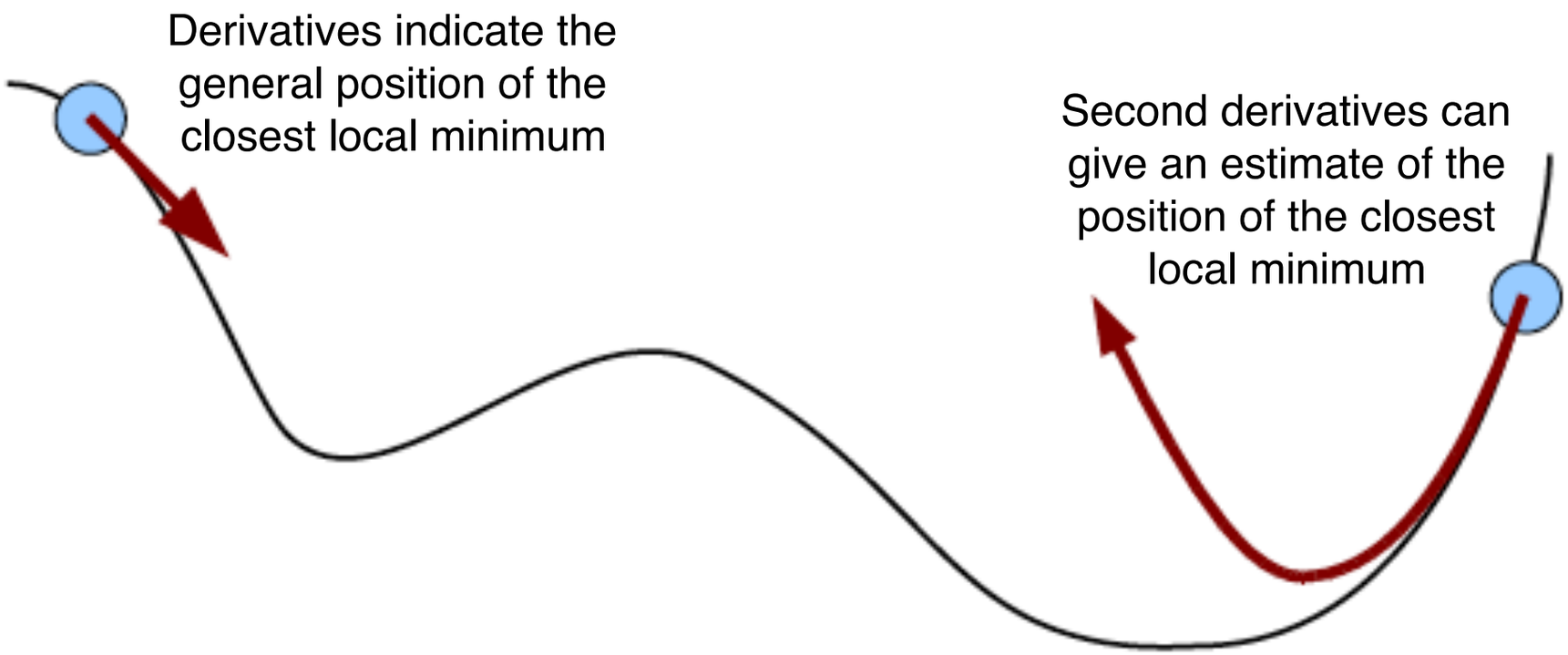
$$f(x^*) \leq f(x) \quad \forall x \in \mathbf{R}^n$$

Local minimizer:

$$f(x^*) \leq f(x^* + \epsilon)$$
$$-\delta \leq \epsilon \leq \delta, \quad \delta > 0$$

Notation and Definition

Differentiability



No such local cues without derivatives

- Derivatives may not exist.
- Derivatives may be too costly to compute.

gradient:
change of objective function with respect to the parameters

Notation and Definition. What is a solution?

Recognizing a Local Minimum

Taylor's Theorem

$$f(x^*) = f(x) + \nabla f(x)^T(x - x^*) + \tfrac{1}{2}(x - x^*)^T \nabla^2 f(x)(x - x^*)$$

First-Order Necessary Conditions

x^* is a local minimizer and f is cont. diff. in an open neighbourhood of x^*

$$\Longrightarrow \nabla f(x^*) = 0.$$

Stationary point: $\nabla f(x^*) = 0.$

Second-Order Necessary Conditions

x^* is a local minimizer, f is twice cont.diff. in an open neighbourhood of x^*

$$\Longrightarrow \nabla f(x^*) = 0 \text{ and } \nabla^2 f(x^*) \text{ is psd}$$

Reminder: B is pd if $p^T B p > 0 \quad \forall p \neq 0$
B is psd if $p^T B p \geq 0 \quad \forall p$

Notation and Definition. What is a solution?

Characterizing a Local Minimum

Second-Order Sufficient Conditions

$\nabla^2 f$ is cont. in an open neighbourhood of x^*
$\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is pd

$\Longrightarrow x^*$ is a strict local minimizer of f

Example: $f(x) = x^4$ has a strict local minimum at $x^* = 0$.

Notation and Definition. What is a solution?

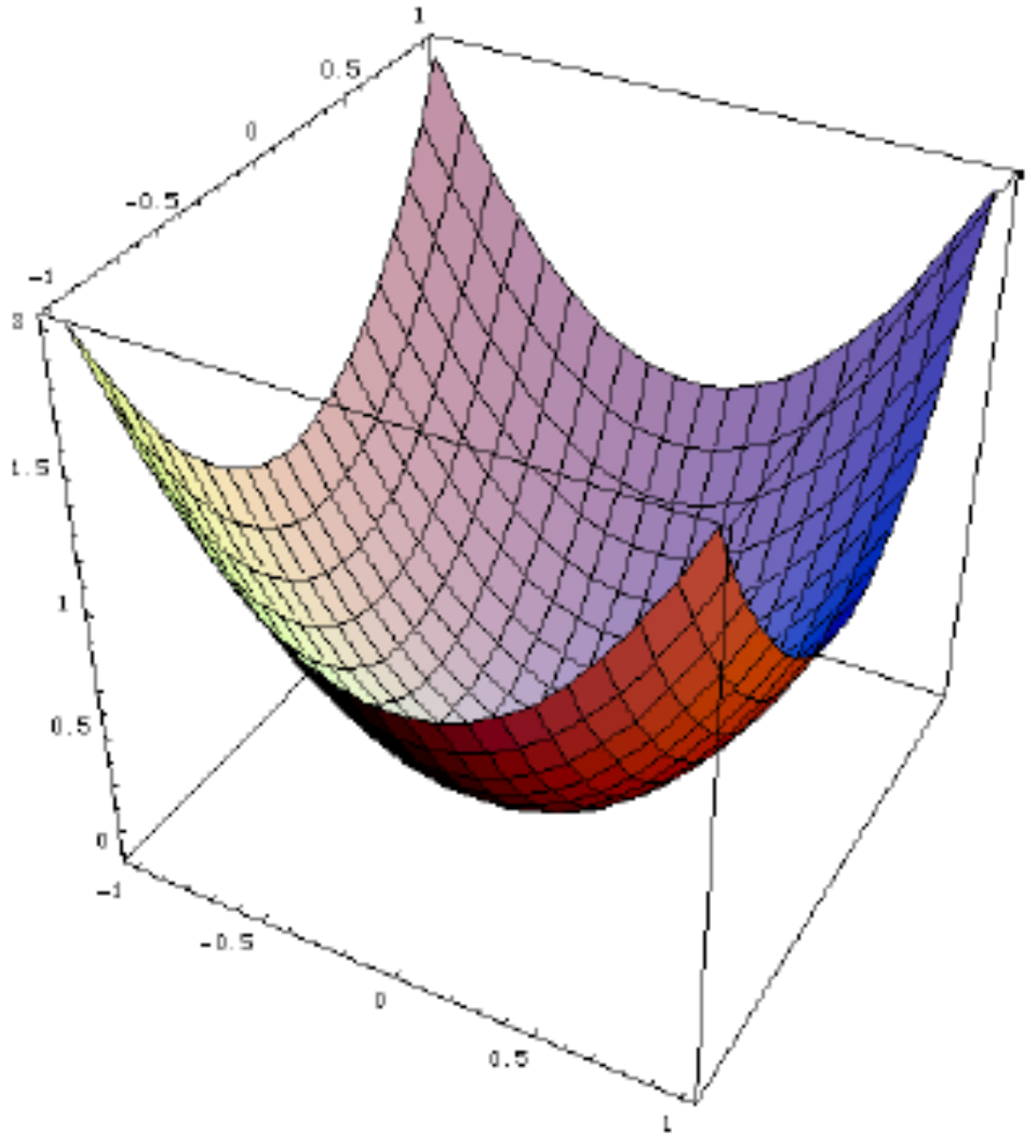
Characterizing a Local Minimum

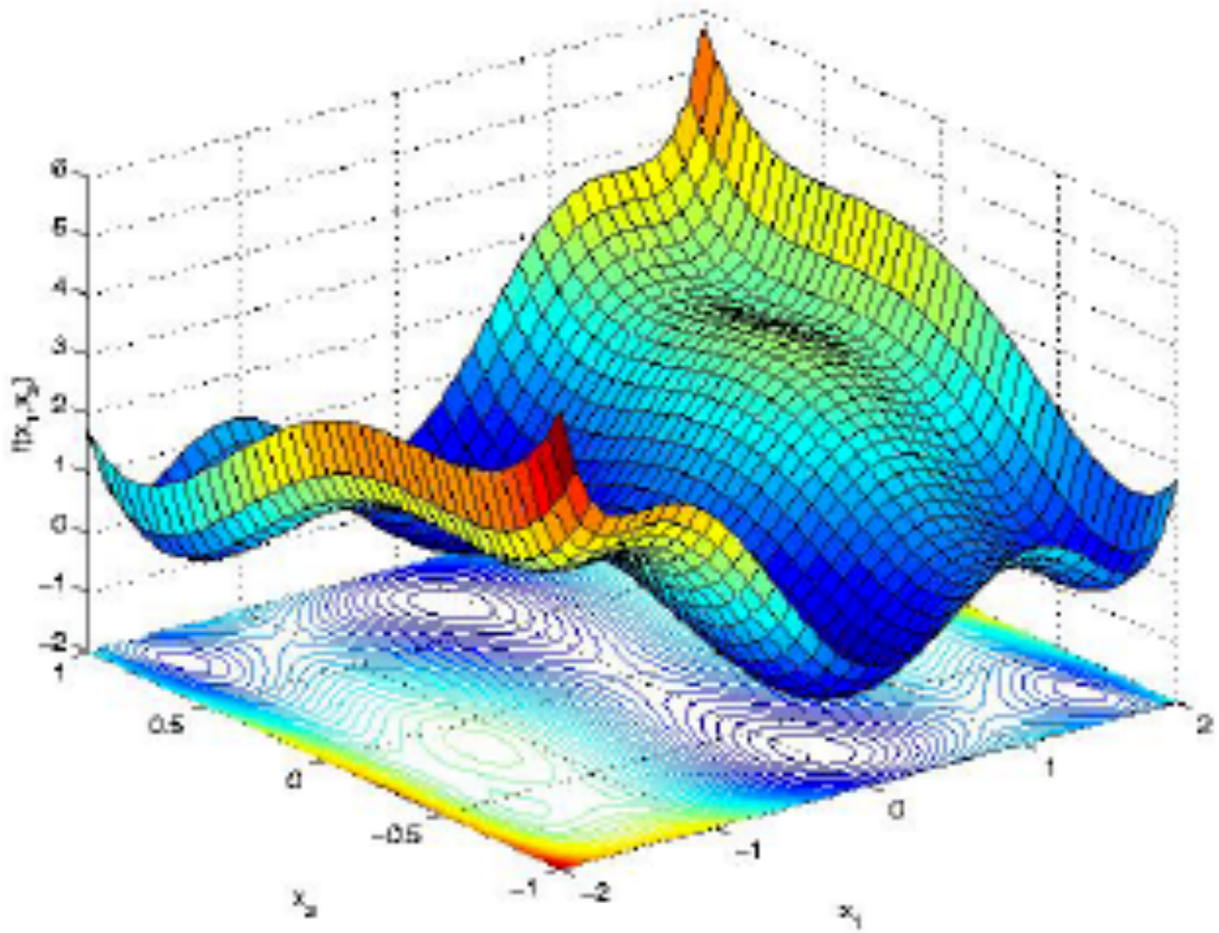
Convex functions

When the function is convex, any local minimizer is a global minimizer. If in addition the function is differentiable, then any stationary point is a global minimizer.

Notation and Definition

Convexity: Functions





Local = **Global**

Stationary Point differentiable **Global**

Local minima, saddle points, plateaux, etc

Optimization algorithms are easy to use. They always return the same solution.

Optimization algorithms usually finds local minima. Result depends on subtle details.

Notation and Definition: Examples

The Transportation Problem

A pharma company has *2 factories* F_1 *and* F_2 and a dozen retail outlets (pharmacies) $R_1, R_2, \ldots R_{12}$.

Each factory F_i can produce a_i quantity of certain antiarrhythmic pills each week; a_i called the capacity of the plant.

Each retail outlet R_j has a known *weekly demand* of b_j quantity of the product. The cost of shipping of one quantity of the pill from factory F_i to retail outlet R_j is c_{ij}.

Problem: determine how much of the product to ship from each factory to each outlet so as to satisfy all the requirements and minimize costs.

Have a break and fun in funding a solution!

Notation and Definition: Examples

The Transportation Problem: Solution

The Decision Variables:

The Objective function:

The Constraints:

Formulation:

Notation and Definition: Examples

The Transportation Problem: Solution

$$\min \quad \sum_{ij} c_{ij} x_{ij}$$

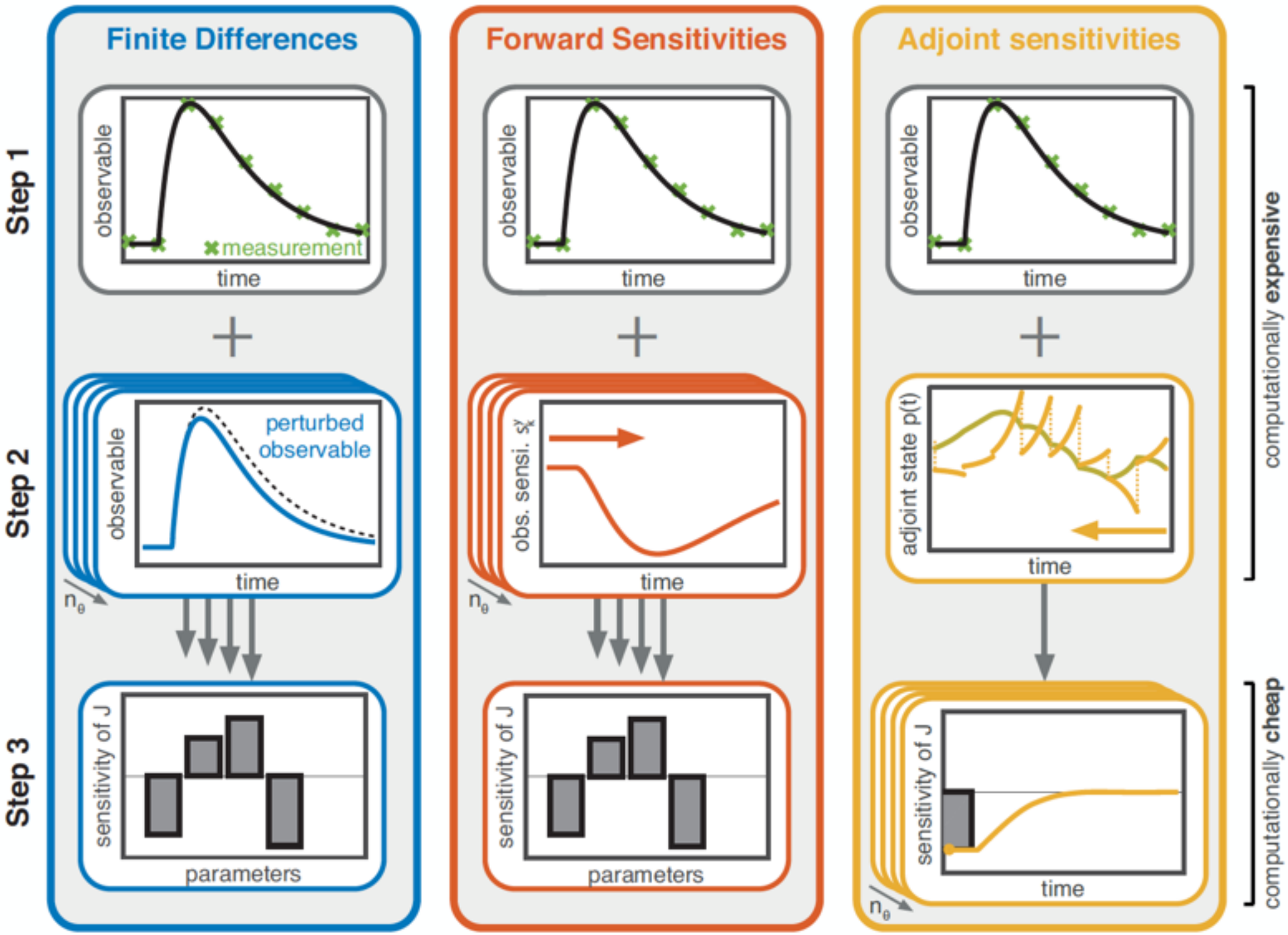
$$\text{s.t.} \quad \sum_{j=1} x_{ij} \leq a_i, \quad i = 1, 2$$

$$\sum_{i=1}^{2} x_{ij} \geq b_j, \quad j = 1, \ldots, 12$$

$$x_{ij} \geq 0, \quad i = 1, 2, \quad j = 1, \ldots, 12$$

How can we compute sensitivities and gradients?

gradient:
change of objective function
with respect to the parameters

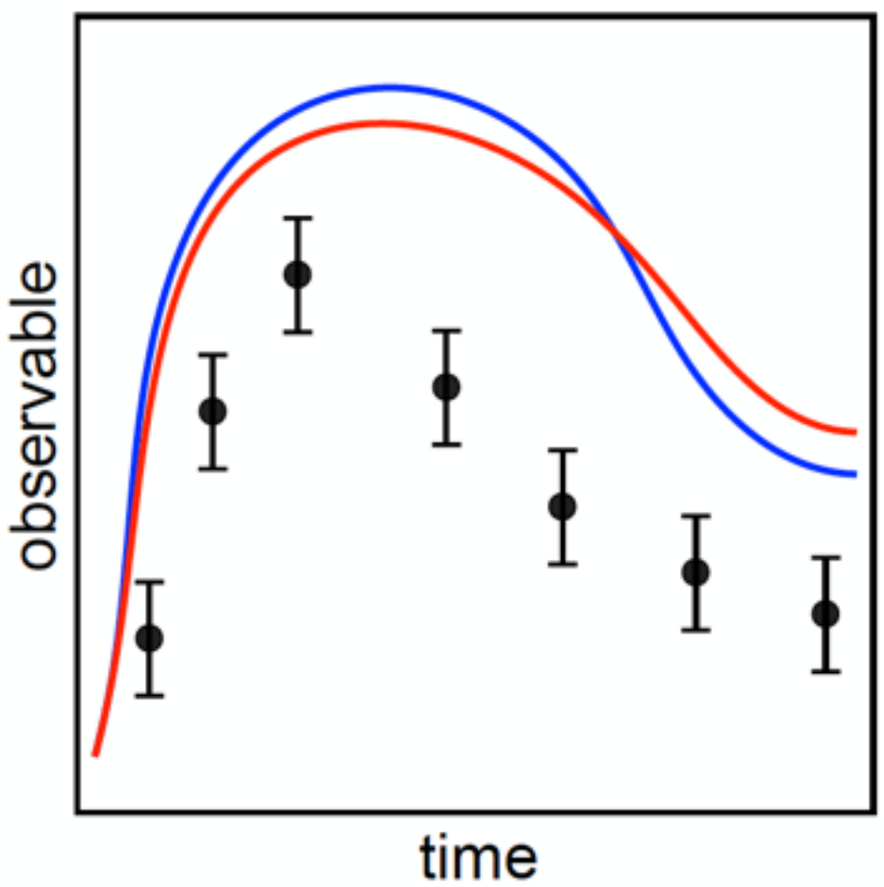


F. Fröhlich et al., PLoS Comp. Biol., 2017

Evaluation of gradient

Finite differences

1. Calculation of objective function for parameter θ
2. Calculation of objective function for parameter $\theta + \Delta e_i$
3. Calculation of gradient as the normalised difference $g_i(\theta)$



$$g_i(\theta) = \frac{\partial J}{\partial \theta_i} = \frac{J(\theta + \Delta e_i) - J(\theta)}{\Delta}$$

What to do?

$g_i(\theta) > 0$: decrease parameter value

$g_i(\theta) < 0$: increase parameter value

$g_i(\theta) = 0$: stop

Evaluation of gradient

Forward sensitivities

$$\frac{\partial J}{\partial \theta_k} = \sum_{j=1}^{n} \sum_{i=0}^{T} \left(\frac{\bar{y}_j(t_i) - y_j(t_i, \theta)}{\sigma_{ij}} \right) \frac{\partial y_j}{\partial \theta_k} \Big|_{(t_i, \theta_k)}$$

output sensitivities

1. Compute state and output trajectory via simulation

state trajectory

state sensitivity

$$x(t, \theta + \Delta e_k) \approx x(t, \theta) + s_k^x(t, \theta)\Delta$$
$$y(t, \theta + \Delta e_k) \approx y(t, \theta) + s_k^y(t, \theta)\Delta$$

output trajectory

output sensitivity

$$s_k^x(t,\theta) = \begin{pmatrix} \frac{\partial x_1(t,\theta)}{\partial \theta_k} \\ \cdots \\ \cdots \\ \frac{\partial x_n(t,\theta)}{\partial \theta_k} \end{pmatrix} \qquad s_k^y(t,\theta) = \begin{pmatrix} \frac{\partial y_1(t,\theta)}{\partial \theta_k} \\ \cdots \\ \cdots \\ \frac{\partial y_n(t,\theta)}{\partial \theta_k} \end{pmatrix}$$

2. Compute state and output sensitivities using the state trajectory

$$\dot{s}_k^x = \frac{\partial f}{\partial x} s_k^x + \frac{\partial f}{\partial \theta_k}, \qquad s_k^x(0) = \frac{\partial x_0}{\partial \theta_k}$$
$$s_{i,k}^y = \frac{\partial h_i}{\partial x} s_k^x + \frac{\partial h_i}{\partial \theta_k}$$

Evaluation of gradient

Forward sensitivities

3. Gradient elements of the objective function $\frac{\partial J}{\partial \theta_k}$ from the output sensitivities and the output trajectory

$$\frac{\partial J}{\partial \theta_k} = \sum_{j=1}^{n} \sum_{i=0}^{T} \left(\frac{\bar{y}_j(t_i) - y_j(t_i, \theta)}{\sigma_{ij}} \right) \frac{\partial y_j}{\partial \theta_k} \Big|_{(t_i, \theta_k)}$$

output sensitivities

Remarks

- Numerically more robust than finite differences
- Computational cost grows linearly with the number of parameters

Evaluation of gradient

Adjoint methods

1. Compute state an output trajectory via simulation

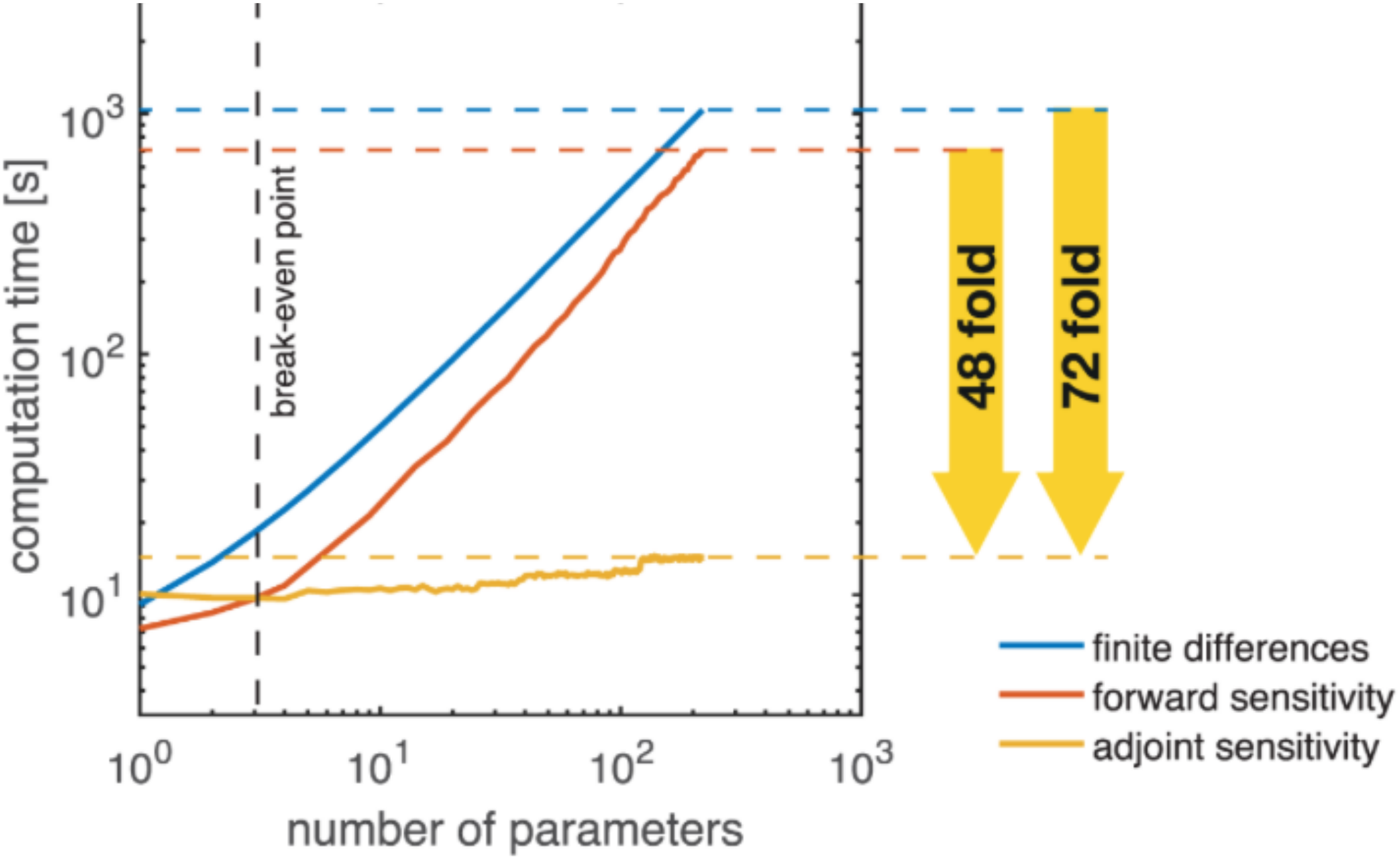
$$\dot{x}(t) = f(x(t,\theta),\theta,u(t)), \quad x(t_0,\theta) = x_0$$
$$y(t,\theta) = h(x(t,\theta);\theta)$$

2. Compute the adjoint state as solution to backward differential equation

$$p(t) = 0, \quad t \in (t_N, t_{N+1})$$
$$\text{for } k = N:-1:1$$
$$\dot{p}(t) = -\left.\frac{\partial f}{\partial x}\right|_{x(t),\theta}^{T} p(t), \quad t \in (t_{k-1}, t_k)$$
$$\text{with } p(t_k) = \lim_{t \to t_k^+} p(t) + \sum_{j=1}^{m} \frac{1}{\sigma_{j,k}^2} \frac{\partial h_j}{\partial x}(x(t_k),\theta)^T (\bar{y}_{j,k} - h_j(x(t_k),\theta))$$

3. Compute gradient using one-dimensional integral

$$\frac{\partial J}{\partial \theta_i} = -\int_0^T p(t)^T \left.\frac{\partial f}{\partial \theta_i}\right|_{x(t),\theta} dt - \sum_{k=1}^{N}\sum_{j=1}^{m} 1 \left.\frac{\partial h_j}{\partial \theta_i}\right|_{x(t_k),\theta}^{T} \left(\frac{\bar{y}_{j,k} - h_j(x(t_k),\theta)}{\sigma_{j,k}^2}\right) - p(0)^T \frac{\partial x_0}{\partial \theta_i}$$



F. Fröhlich et al., PLoS Comp Biology, 2017

How does local optimisation work?

Local optimisation

Optimisation problem:

$$\min_{\theta} J(\theta)$$

Optimisation methods:

- Gradient-free methods (sampling methods,…)
- Gradient-based methods (gradient descent, Newton method,…)

Variety of a powerful collection of derivative-based algorithms for unconstrained optimisation of smooth functions.

Local optimisation

Goal of local optimisation:

Construction of sequence of parameters

$$\theta^0, \theta^1, \theta^2, \ldots$$

along which objective function decreases

$$J(\theta^0) > J(\theta^1) > J(\theta^2) \ldots$$

using local properties of the objective function.

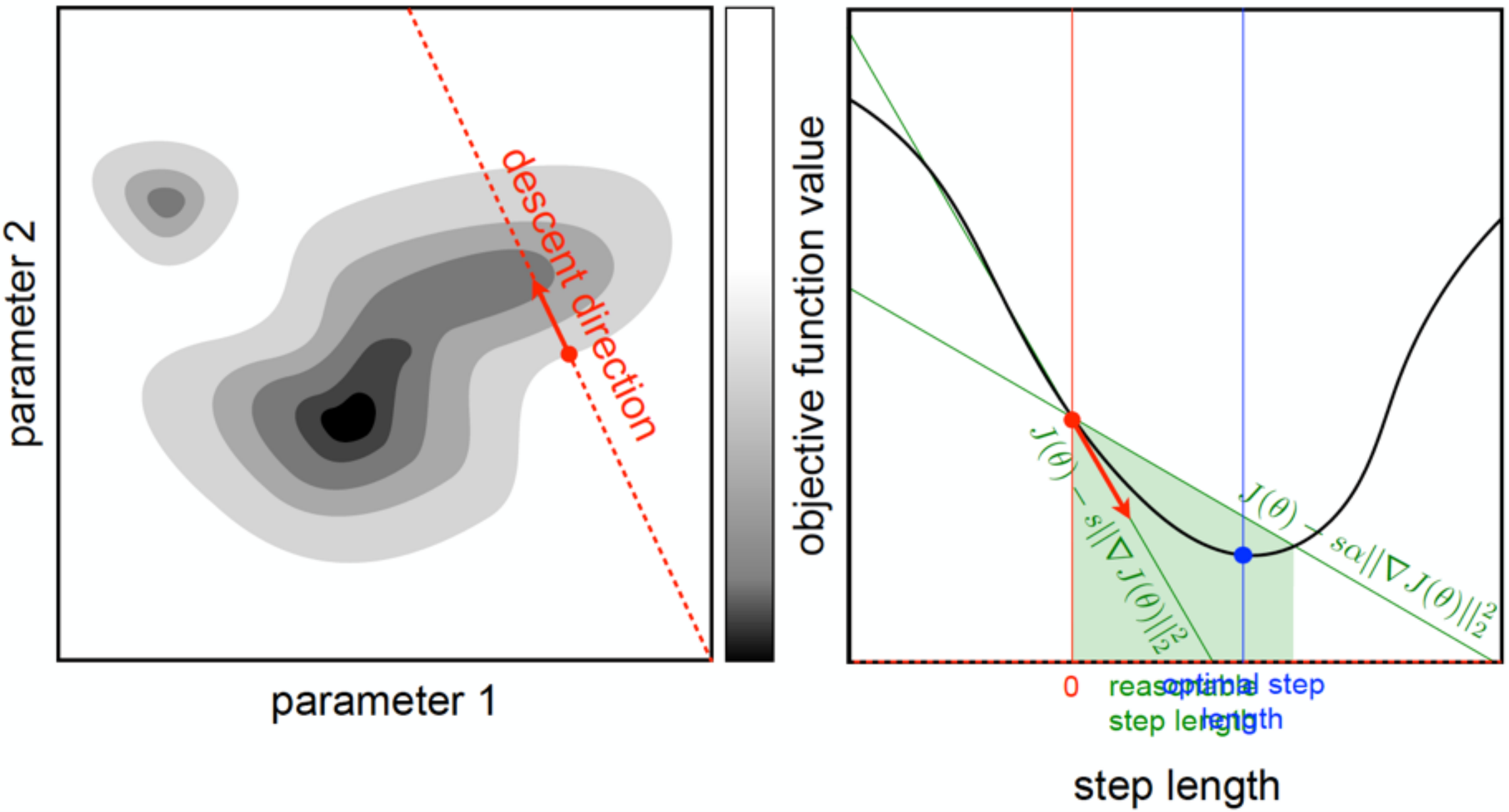
Optimiser path construction

$$\theta^{k+1} = \theta^k + t^k \Delta^k$$

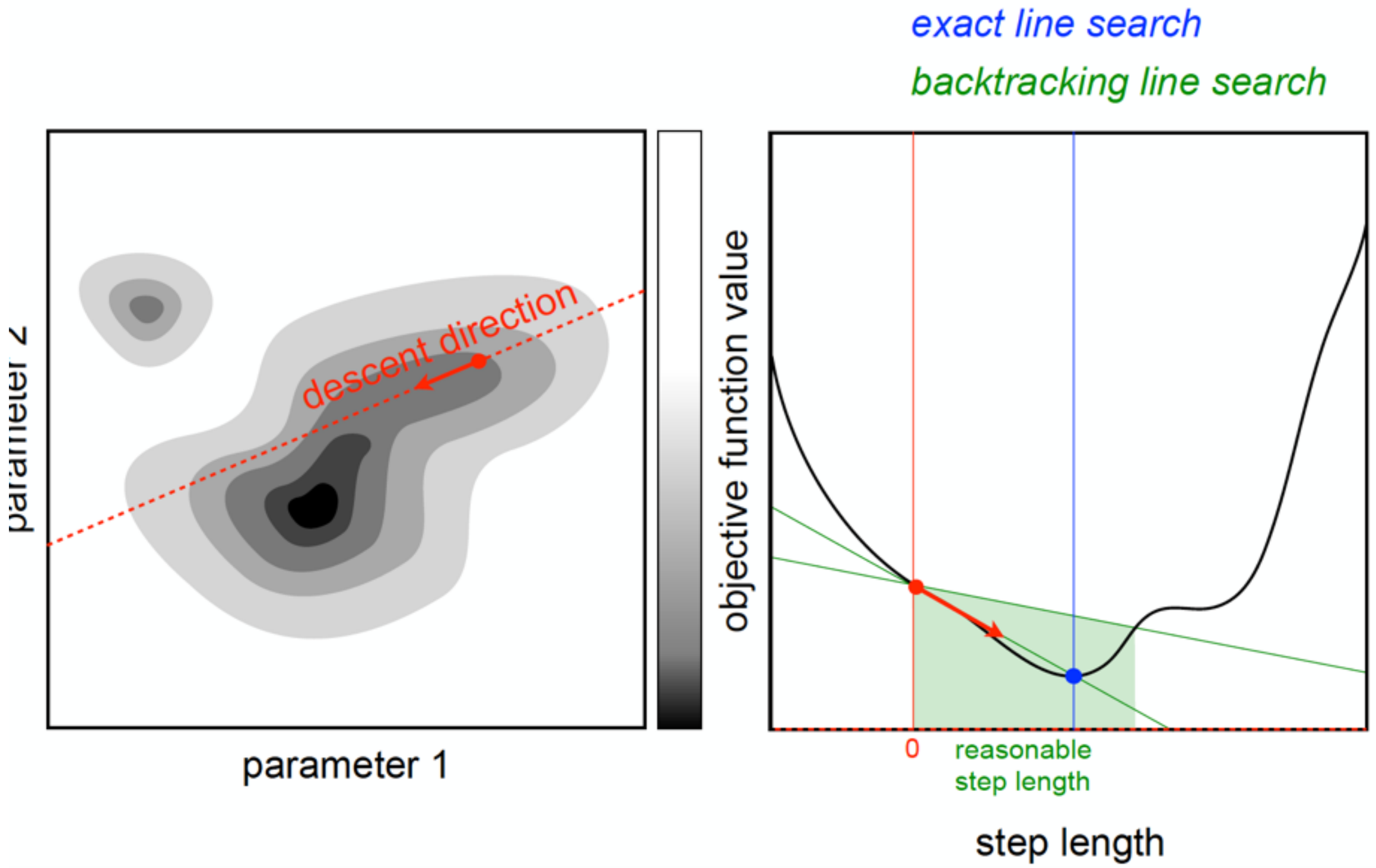
$\Delta^k \in \mathbb{R}^n$ search direction

$t^k \in \mathbb{R}_+$ step length

Line search methods



Line search methods



Steepest descent method

Search direction

Direction with strongest (local) decrease for given vector norm

$$\Delta^k = \arg \min_{v, \|v\|=1} \nabla_\theta J(\theta^k)^T v$$

Special cases:

Euclidean norm $\|v\| := v^T v$

$$\Delta^k = -\nabla_\theta J(\theta^k)$$ **Gradient descent**

General quadratic norm $\|v\| := v^T P v$ with positive definite matrix P

$$\Delta^k = -P^{-1} \nabla_\theta J(\theta^k)$$

Step length

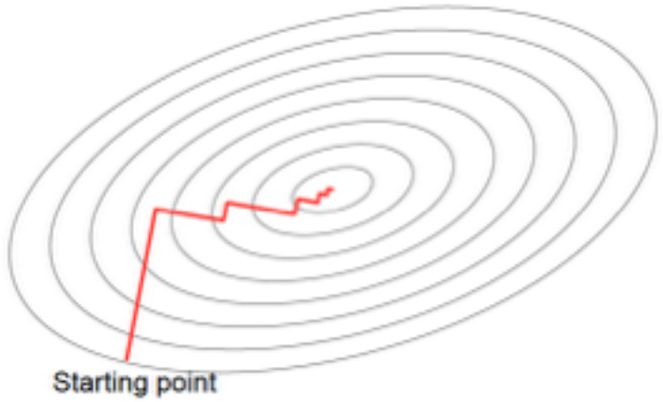
Minimisation of the functional along $\theta = \theta^k + t^k \Delta^k$

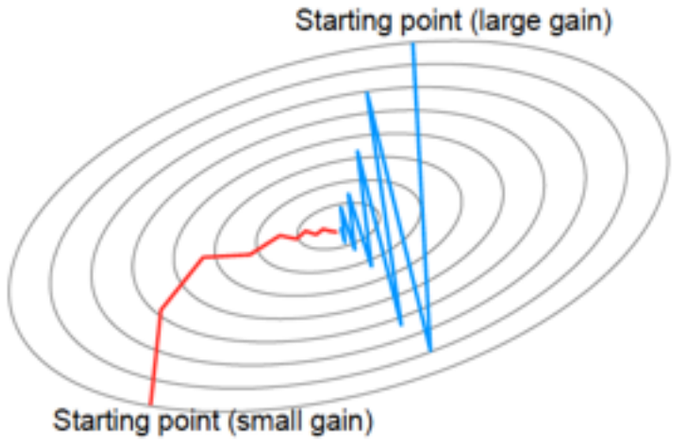
exact line search $t^k := \arg \min_{t>0} J(\theta^k + t\Delta^k)$

backtracking line search (= successive geometric reduction of step length)

Steepest descent method: Implementation

Input: starting point $\theta^{(0)}$
set $k \leftarrow 0$
repeat
  calculate steepest descent direction $\Delta^{(k)}$
  calculate step size $t^{(k)}$ by line search
  set $\theta^{(k+1)} \leftarrow \theta^{(k)} + t^{(k)}\Delta^{(k)}$
  set $k \leftarrow k + 1$
until *stopping criterion holds*;





Common stopping criteria

Small change of objective functional: $\|J(\theta^{k+1}) - J(\theta^{k})\| < \epsilon_J$

Small change of parameters: $\|\theta^{k+1} - \theta^{k}\| < \epsilon_\theta$

Number of function evaluations: $k > N$

Remarks

- The calculation of the descent direction and the step size requires the numerical simulation of the ODE.
- Beneficial if the calculation of gradient if cheap.
- Slow on difficult problems

Newton method

Idea: Next point as optimum of local approximation (**Taylor explansion**)

$$J(\theta) \approx J(\theta^k) + \nabla_\theta J(\theta^k)(\theta - \theta^k) + \frac{1}{2}(\theta - \theta^k)\nabla^2_\theta J(\theta^k)(\theta - \theta^k)^T$$

Newton method

$$\theta^{k+1} = \theta^k - (\nabla^2_\theta J(\theta^k))^{-1}\nabla_\theta J(\theta^k)$$

Remarks:

- (Local) quadratic convergence.
- Positive definite Hessian required (very few iterations needed).
- Local approximation might not be satisfactory.
- Computation and storage of Hessian can be **quiet costly**!

$$H(\omega) = \begin{pmatrix} \frac{\partial^2 f}{\partial \omega_1 \partial \omega_1} & \cdots & \frac{\partial^2 f}{\partial \omega_1 \partial \omega_d} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial \omega_d \partial \omega_1} & \cdots & \frac{\partial^2 f}{\partial \omega_d \partial \omega_d} \end{pmatrix}$$

Newton method

$$\theta^{k+1} = \theta^k - B_k^{-1}\nabla_\theta J(\theta^k)$$

Conjugate gradient method

Idea: Adaptation of the method for solving systems of linear equations

$$Ax = b \quad \Longleftrightarrow \quad \min \phi(x) = \tfrac{1}{2} x^T A x - b^T x.$$

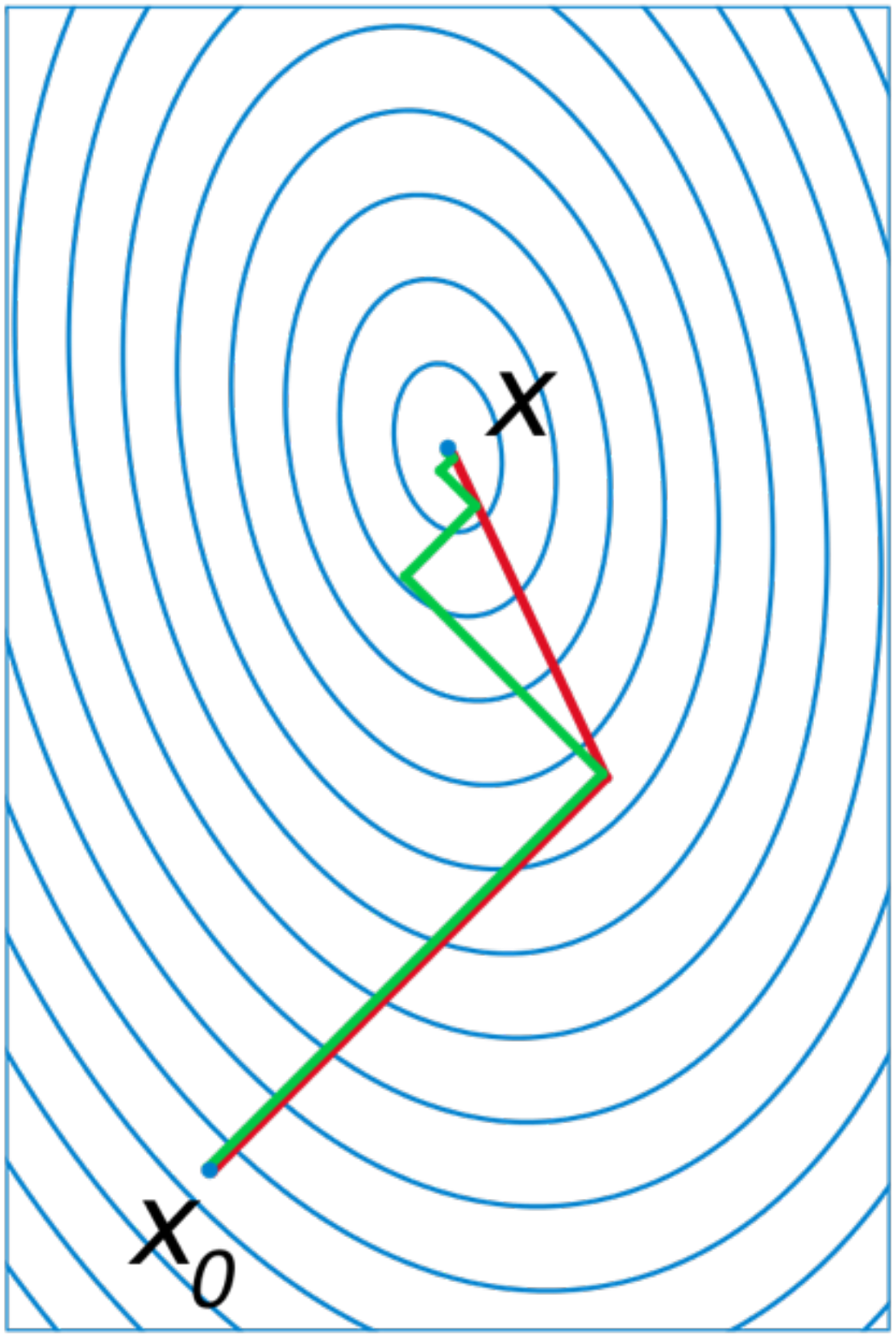
Conjugate gradient method

$$\Delta^k = -\nabla_\theta J(\theta^k) + \beta_k \Delta^{k-1}$$

Remarks:

- More effective than the steepest descent direction.
- No matrix storage. Almost as simple as the steepest descent methods.
- Adapted to solve nonlinear optimisation problems.
- Do not attain fast convergence rates as Newton or Quasi-Newton.
- Sensitive to roundoff errors.

Reminder: two non-zero vectors u and v are conjugate (wrt A) if $u^T A v = 0$.



A comparison of the convergence of steepest descent with optimal step size (in green) and conjugate vector (in red) for minimizing a quadratic function associated with a given linear system.

Conjugate gradient, assuming exact arithmetic, converges in at most n steps where n is the size of the matrix of the system (here n=2).

From Wikipedia

Trust Region

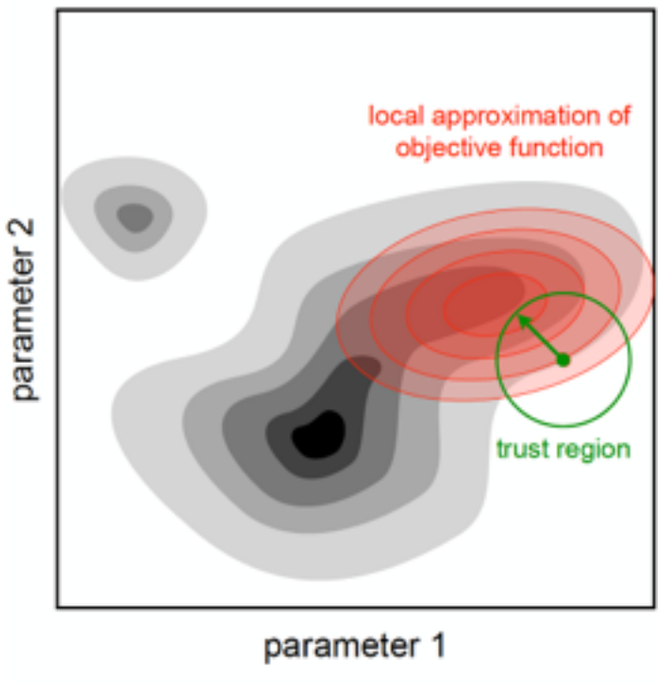
Idea: Construct a model function whose behaviour near the current point is similar to that of the actual objective function.

Trust region

In trust-region methods the next point is the minimiser of the quadratic approximation of the objective function with the trust region,

$$\theta^{k+1} = \theta^k + \Delta \text{ with } \Delta = \arg\min_{\|\Delta\|_2 \leq r^k} \nabla_\theta J(\theta^k)\Delta + \frac{1}{2}\Delta^T(\nabla^2_\theta J(\theta^k))\Delta$$

with the trust-region radius r^k being updated based on the prediction accuracy.



Trust Region

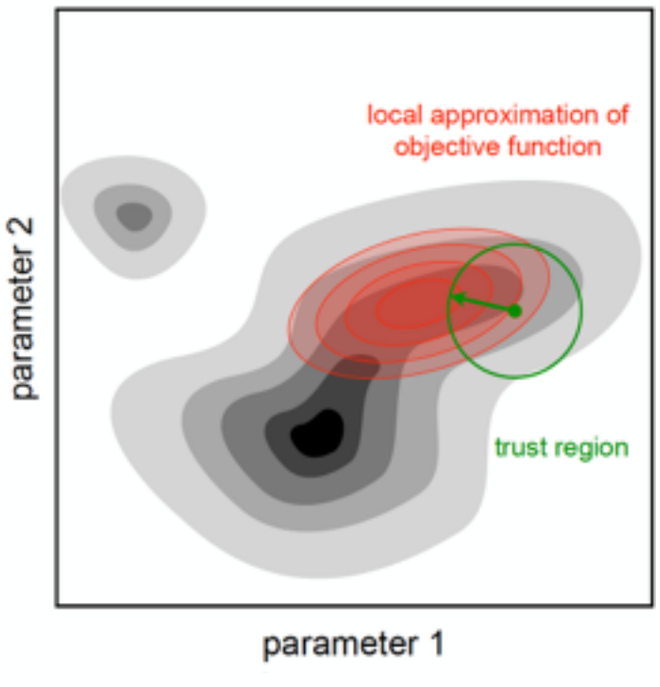
Idea: Construct a model function whose behaviour near the current point is similar to that of the actual objective function.

Trust region

In trust-region methods the next point is the minimiser of the quadratic approximation of the objective function with the trust region,

$$\theta^{k+1} = \theta^k + \Delta \text{ with } \Delta = \arg\min_{\|\Delta\|_2 \leq r^k} \nabla_\theta J(\theta^k)\Delta + \frac{1}{2}\Delta^T(\nabla^2_\theta J(\theta^k))\Delta$$

with the trust-region radius r^k being updated based on the prediction accuracy.



Trust Region

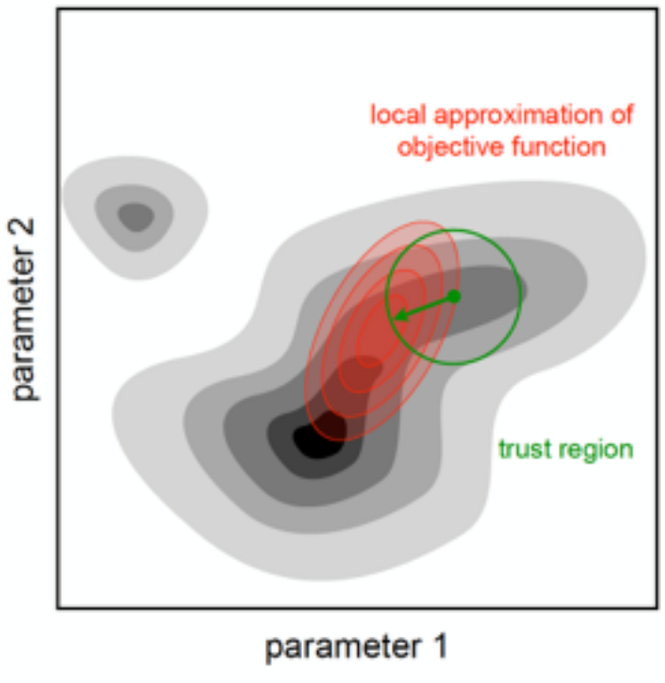
Idea: Construct a model function whose behaviour near the current point is similar to that of the actual objective function.

Trust region

In trust-region methods the next point is the minimiser of the quadratic approximation of the objective function with the trust region,

$$\theta^{k+1} = \theta^k + \Delta \text{ with } \Delta = \arg \min_{\|\Delta\|_2 \leq r^k} \nabla_\theta J(\theta^k)\Delta + \frac{1}{2}\Delta^T(\nabla^2_\theta J(\theta^k))\Delta$$

with the trust-region radius r^k being updated based on the prediction accuracy.



Trust Region

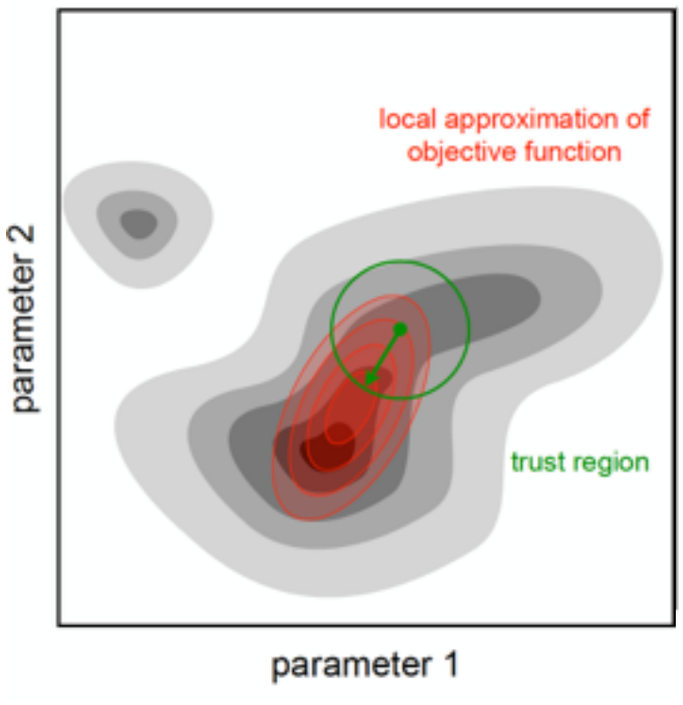
Idea: Construct a model function whose behaviour near the current point is similar to that of the actual objective function.

Trust region

In trust-region methods the next point is the minimiser of the quadratic approximation of the objective function with the trust region,

$$\theta^{k+1} = \theta^k + \Delta \text{ with } \Delta = \arg \min_{\|\Delta\|_2 \leq r^k} \nabla_\theta J(\theta^k)\Delta + \frac{1}{2}\Delta^T(\nabla^2_\theta J(\theta^k))\Delta$$

with the trust-region radius r^k being updated based on the prediction accuracy.



Trust Region

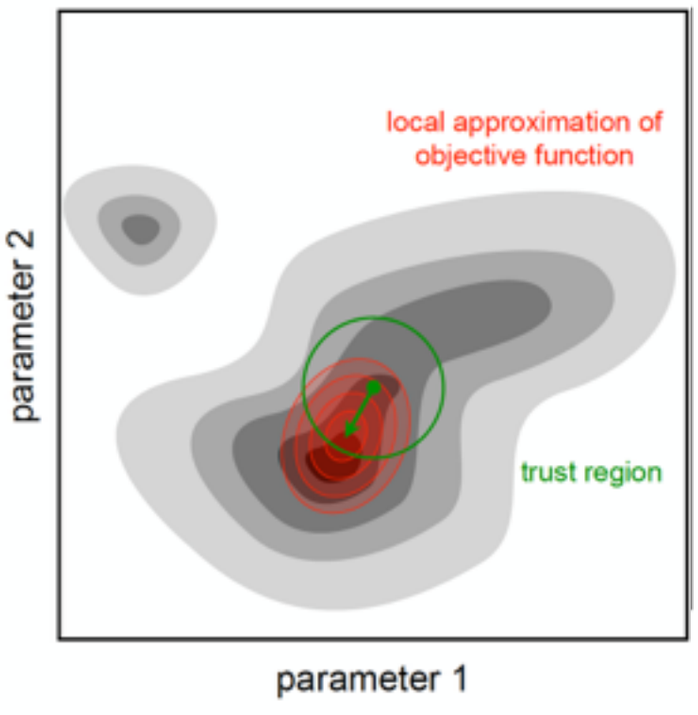
Idea: Construct a model function whose behaviour near the current point is similar to that of the actual objective function.

Trust region

In trust-region methods the next point is the minimiser of the quadratic approximation of the objective function with the trust region,

$$\theta^{k+1} = \theta^k + \Delta \text{ with } \Delta = \arg \min_{\|\Delta\|_2 \leq r^k} \nabla_\theta J(\theta^k)\Delta + \frac{1}{2}\Delta^T(\nabla^2_\theta J(\theta^k))\Delta$$

with the trust-region radius r^k being updated based on the prediction accuracy.



Trust Region

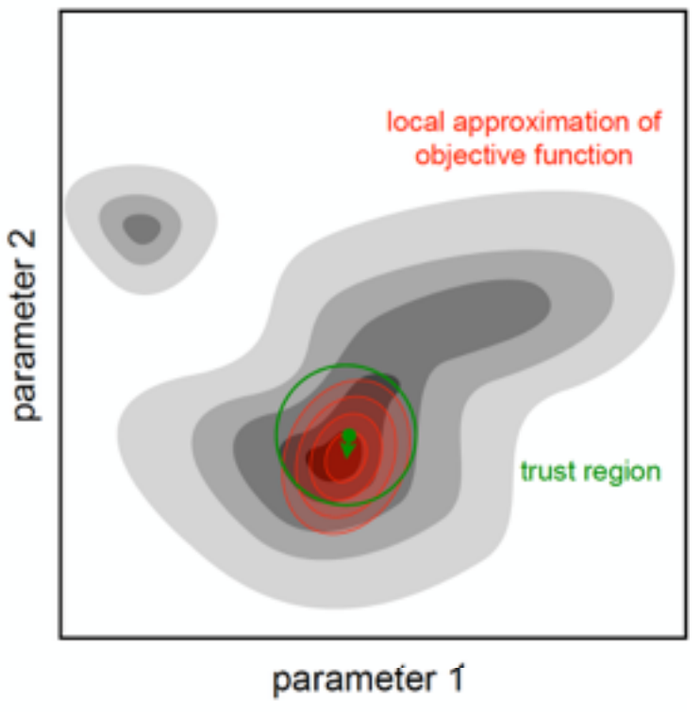
Idea: Construct a model function whose behaviour near the current point is similar to that of the actual objective function.

Trust region

In trust-region methods the next point is the minimiser of the quadratic approximation of the objective function with the trust region,

$$\theta^{k+1} = \theta^k + \Delta \text{ with } \Delta = \arg\min_{\|\Delta\|_2 \leq r^k} \nabla_\theta J(\theta^k)\Delta + \frac{1}{2}\Delta^T(\nabla^2_\theta J(\theta^k))\Delta$$

with the trust-region radius r^k being updated based on the prediction accuracy.



Remark:

• Approach allows conceptually for negative and indefinite Hessians.

Selection of trust region?

Trust Region: Implementation

Input: starting point $\theta^{(0)}$, initial trust-region radius $r^{(0)}$

Input: tuning parameters $\gamma_2 > 1 > \gamma_1 > 0$ and $1 > \nu > 0$

set $k \leftarrow 0$

repeat

calculate gradient and Hessian

calculate update step $\Delta \leftarrow \underset{||\Delta||_2 \leq r^{(k)}}{\arg\min} \nabla_\theta J|_{\theta^{(k)}} \Delta + \frac{1}{2}\Delta^T \left(\nabla^2_\theta J|_{\theta^{(k)}}\right) \Delta$

calculate prediction accuracy $\rho \leftarrow \dfrac{J(\theta^{(k)} + \Delta) - J(\theta^{(k)})}{\nabla_\theta J|_{\theta^{(k)}} \Delta + \frac{1}{2}\Delta^T \left(\nabla^2_\theta J|_{\theta^{(k)}}\right) \Delta}$

if $\rho \geq \nu$ **then**

set $\theta^{(k+1)} \leftarrow \theta^{(k)} + \Delta$

set $r^{(k+1)} \leftarrow \gamma_2 r^{(k)}$

else

set $\theta^{(k+1)} \leftarrow \theta^{(k)}$

set $r^{(k+1)} \leftarrow \gamma_1 r^{(k)}$

end

set $k \leftarrow k + 1$

adaptation of trust region radius

until *stopping criterion holds*;

Trust Region: Implementation

Input: starting point $\theta^{(0)}$, initial trust-region radius $r^{(0)}$

Input: tuning parameters $\gamma_2 > 1 > \gamma_1 > 0$ and $1 > \nu > 0$

set $k \leftarrow 0$

repeat

calculate gradient and Hessian

calculate update step $\Delta \leftarrow \underset{||\Delta||_2 \leq r^{(k)}}{\arg\min} \nabla_\theta J|_{\theta^{(k)}} \Delta + \frac{1}{2}\Delta^T \left(\nabla^2_\theta J|_{\theta^{(k)}}\right) \Delta$

calculate prediction accuracy $\rho \leftarrow \dfrac{J(\theta^{(k)} + \Delta) - J(\theta^{(k)})}{\nabla_\theta J|_{\theta^{(k)}} \Delta + \frac{1}{2}\Delta^T (\nabla^2_\theta J|_{\theta^{(k)}}) \Delta}$

if $\rho \geq \nu$ **then**

set $\theta^{(k+1)} \leftarrow \theta^{(k)} + \Delta$

set $r^{(k+1)} \leftarrow \gamma_2 r^{(k)}$

Expand trust region

else

set $\theta^{(k+1)} \leftarrow \theta^{(k)}$

set $r^{(k+1)} \leftarrow \gamma_1 r^{(k)}$

Shrink trust region

end

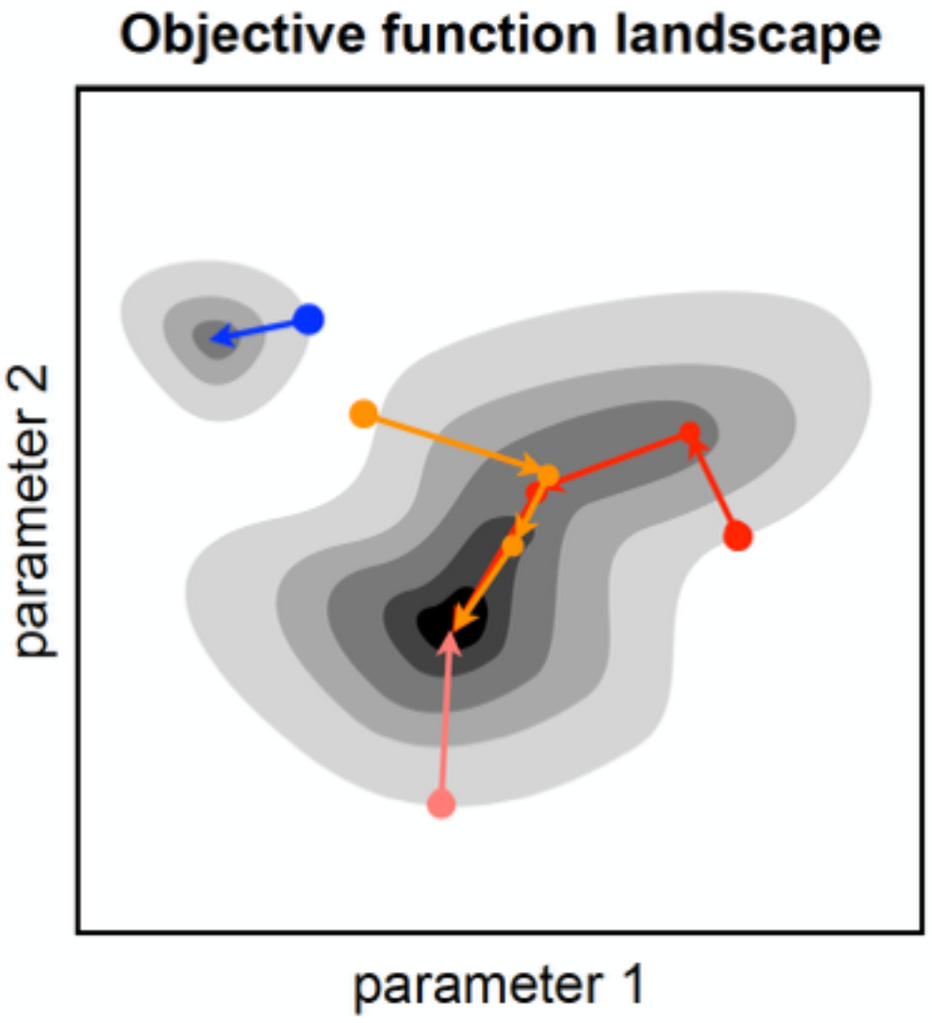
set $k \leftarrow k + 1$

adaptation of trust region radius

until *stopping criterion holds*;

What if multiple local minima?

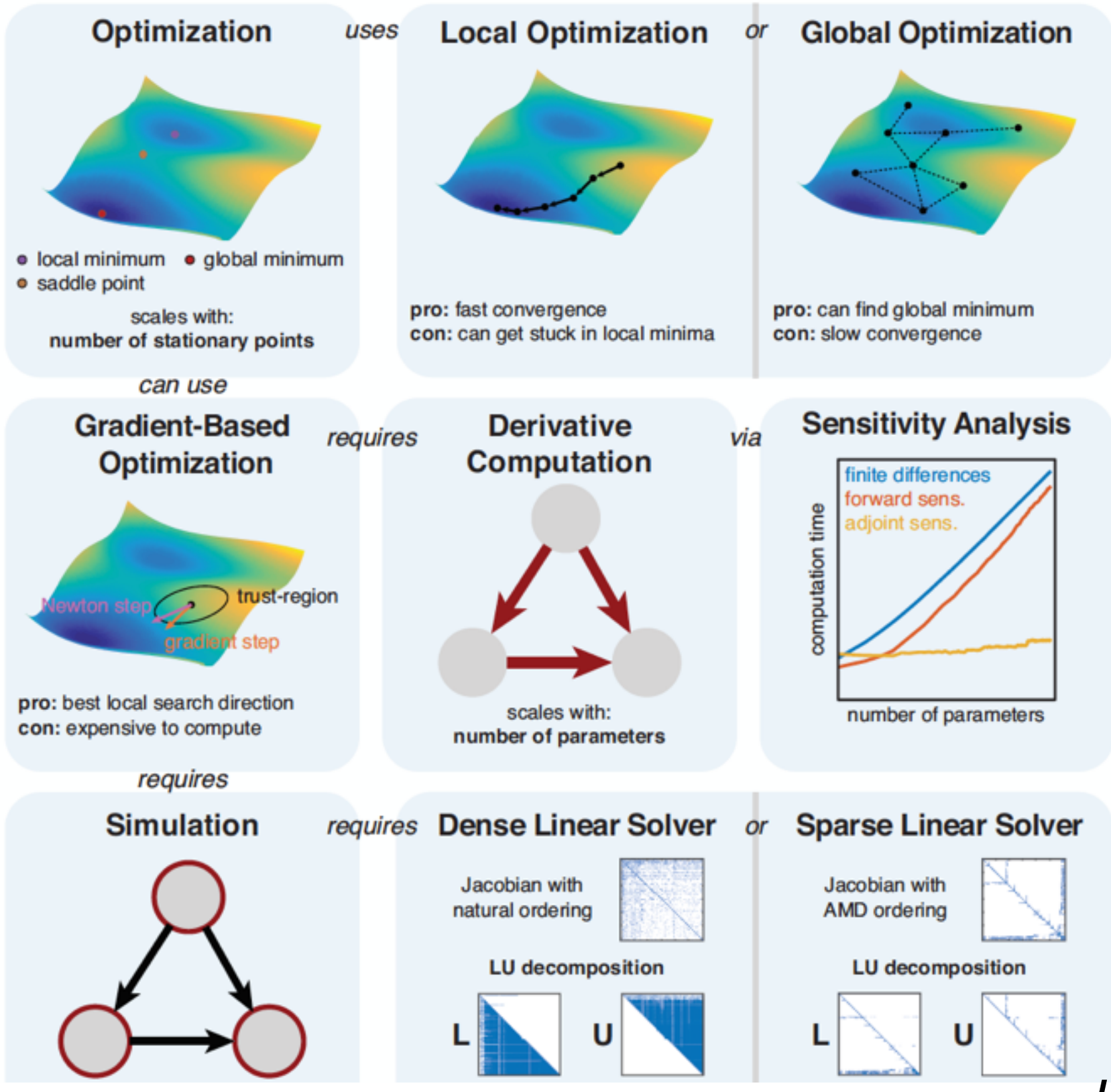
Multi-start local optimisation



Properties of good local optimiser:
Robust and efficient convergence to the local optimum in those basin of attraction the local optimiser is initialised.

Remarks:
• Large number of samples requires if many local optima or if the global optimum has a small basin of attraction.

Ingredients of good derivative-based optimisation



J. Hasenauer, 2017

Derivative-free optimization

Overview. Why?

- Evaluating $\nabla J(\theta)$ in practice is sometimes impossible:
  - the function can be the results of a simulation / experiment;
  - coding of gradient may be time-consuming or impractical.

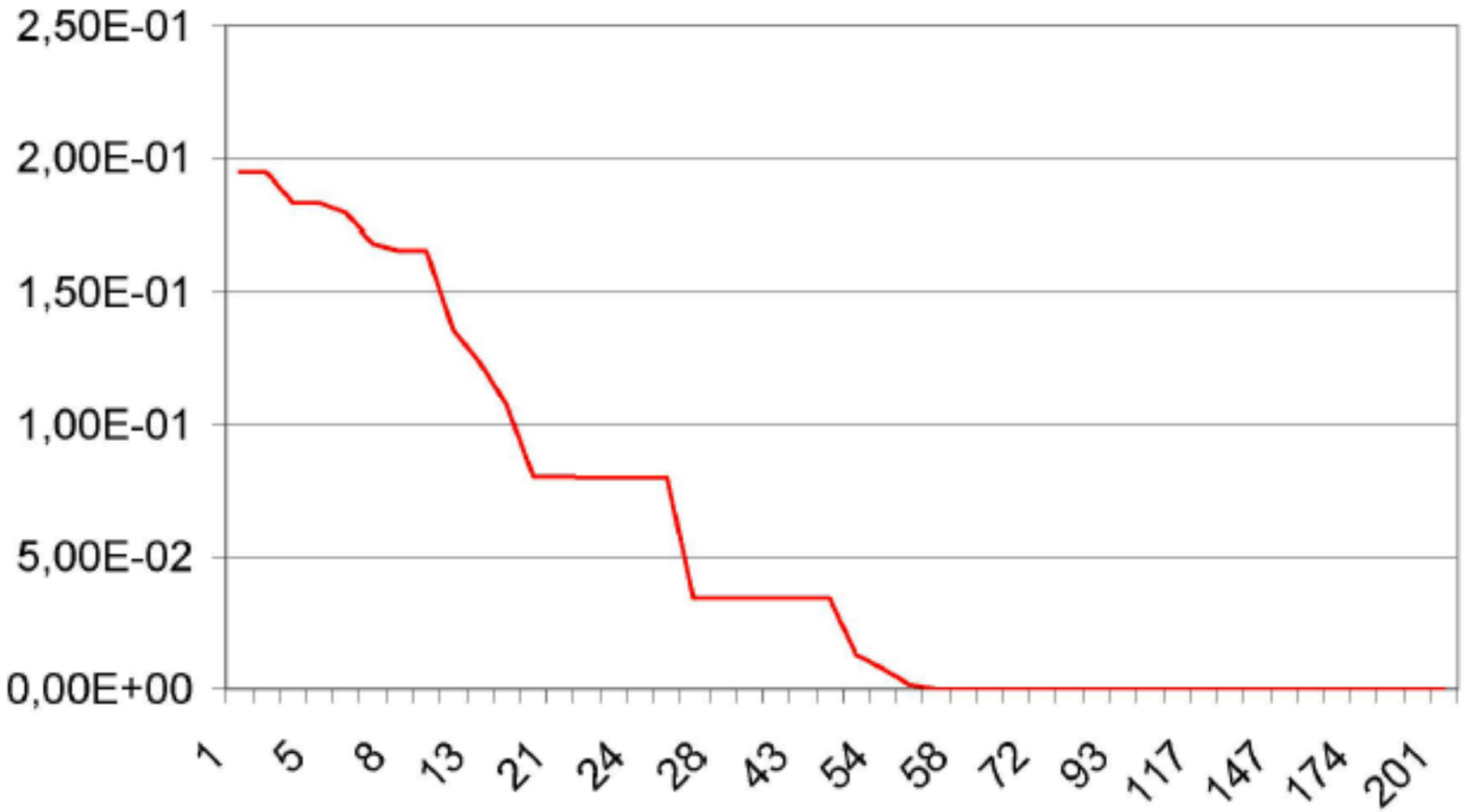
- **Approaches:**
  - approximate gradient and possibly Hessian using finite differences, then apply derivative-based method. **BUT**:
    - Number of function evaluations may be excessive.
    - Unreliable with noise.
  - use function values at a set of sample points, instead of the gradient approximation, and determine a new iterate by a different means. **BUT**:
    - less developed / less efficient.
    - effective only for small problems.
    - difficult to use with general constraints.

Recommendation: derivative-based **>** finite-difference based **>** derivative-free

Derivative-free optimization

Limitations

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



Newton Method:
Quadratic convergence
First + Second order derivatives

Quasi-Newton Method:
Superlinear convergence
First order derivatives

Derivative-Free Optimization

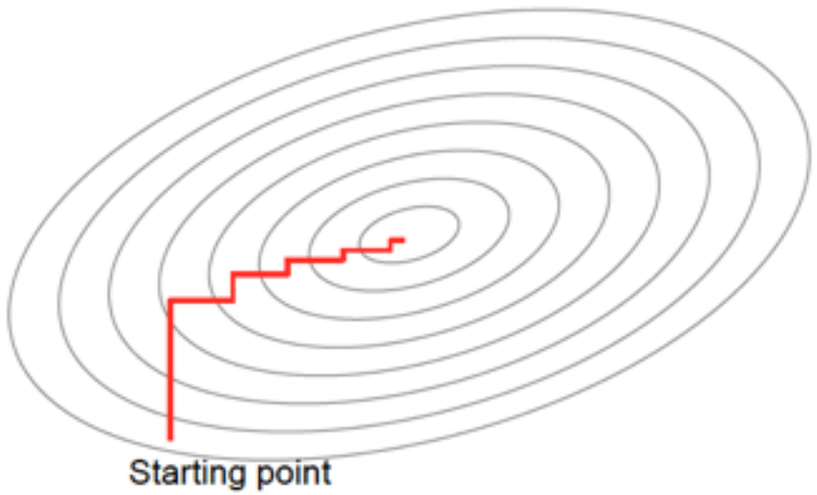
Overview

- Derivative-free optimization (DFO) algorithms sample function values to determine the new iterate.
  - *Model-based methods:* build a linear or quadratic model of the function by interpolating the function at a set of samples and use it with trust-region. Slow convergence rate and very costly steps.
  - *Coordinate and pattern-search methods*: look along certain specified directions from the current iterate for a point with a lower function value.
- *Conjugate direction methods:* construct gradient descent directions using only function values.
- *Finite-difference approximations* to the gradient degrade significantly with noise in f.
- *Downhill simplex method (Nelder-Mead method):* keep track of n+1 points of interest, whose convex hull forms a simplex.

Coordinate search method

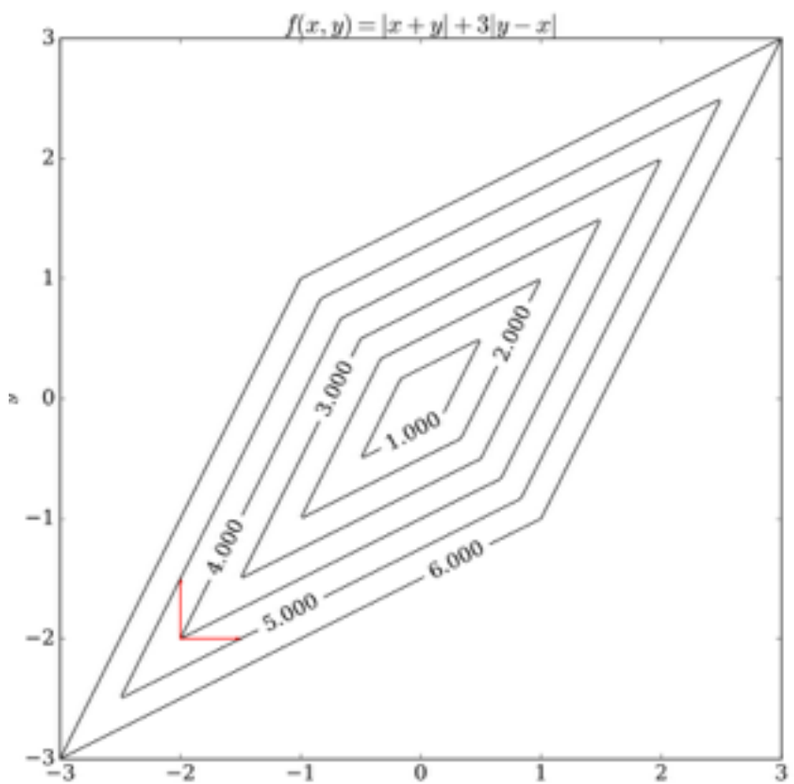
Coordinate search method

Perform successive line searches along the axes.



Pros:

- Simple;
- Competitive to other methods (esp. in machine learning);
- No derivative needed.



Limitations:

- Does not work for non-smooth functions.
- Difficult to parallelise.

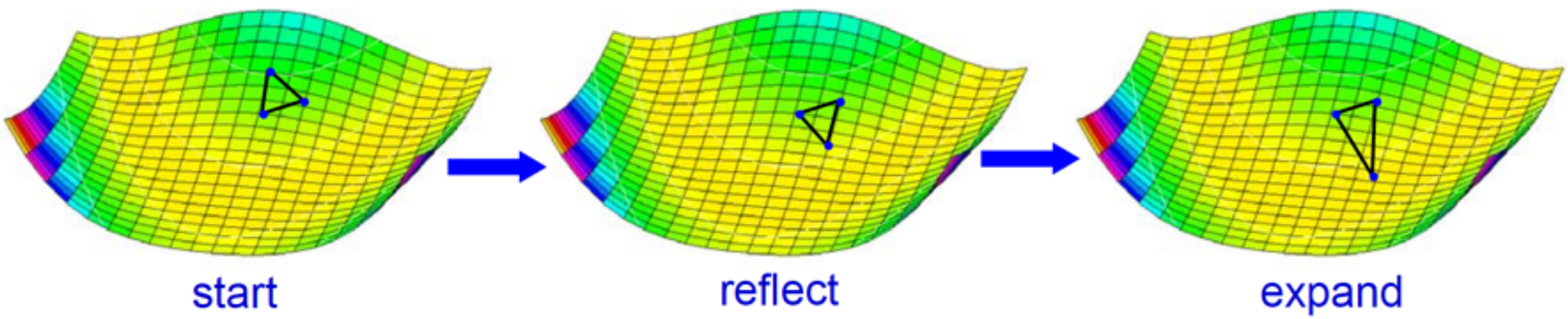
May get stuck at a non-stationary point

Nelder-Mead Method

Nelder-Mead Method

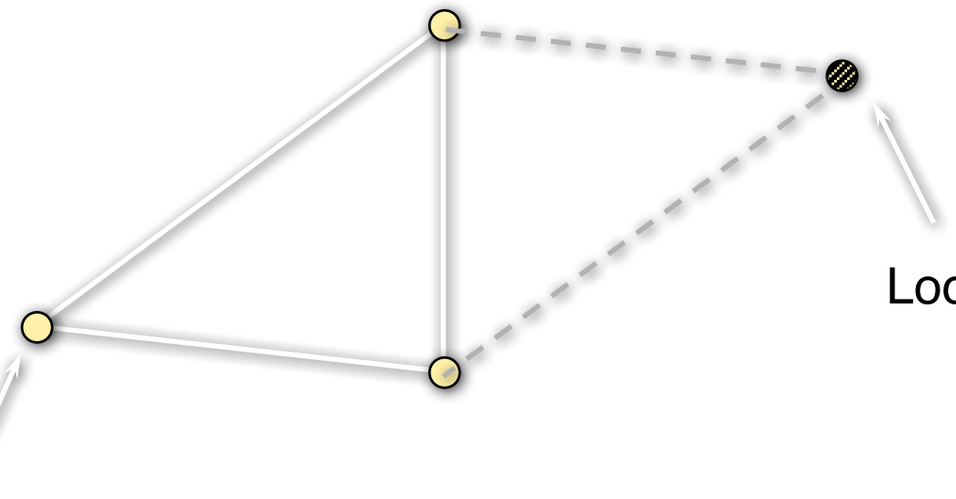
The method uses the concept of a simplex, which is a polytope of n+1 vertices in n dimension (a line segment in 1D space, a triangle in 2D, a tetrahedron in 3D space, etc.)

Update simplex at each iteration by removing vertex with the worst functional value by moving, expanding, or contracting simplex.



Nelder-Mead Method

- Basic operations: **reflection**

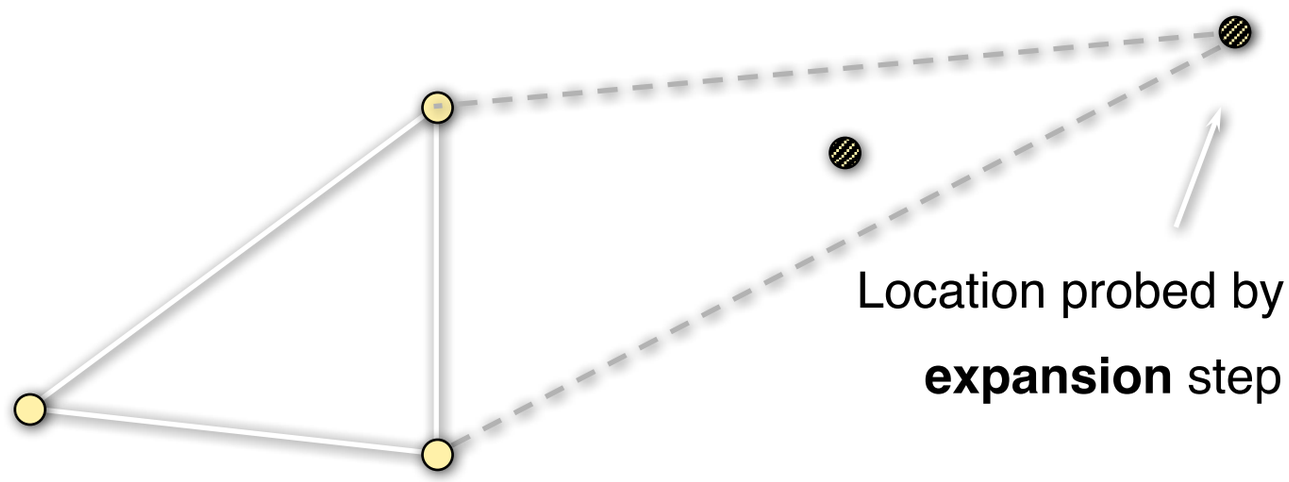


Location probed by an **reflection** step

Worst point (highest function values)

- Basic operations: **expansion**

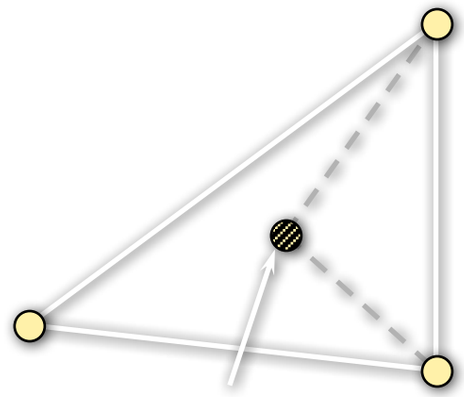
If reflection resulted in best (lowest) value so far, try an expansion



Location probed by an **expansion** step

Nelder-Mead Method

- Basic operations: **contraction**



Location probed by an **contraction** step

Pros:

- Fairly efficient at each iteration;
- Simple to implement;
- No derivatives required;
- Deals well with noise in the cost function.

Cons:

- Can take a lot of iterations;
- Sometimes need restart.

How well are the parameters determined?

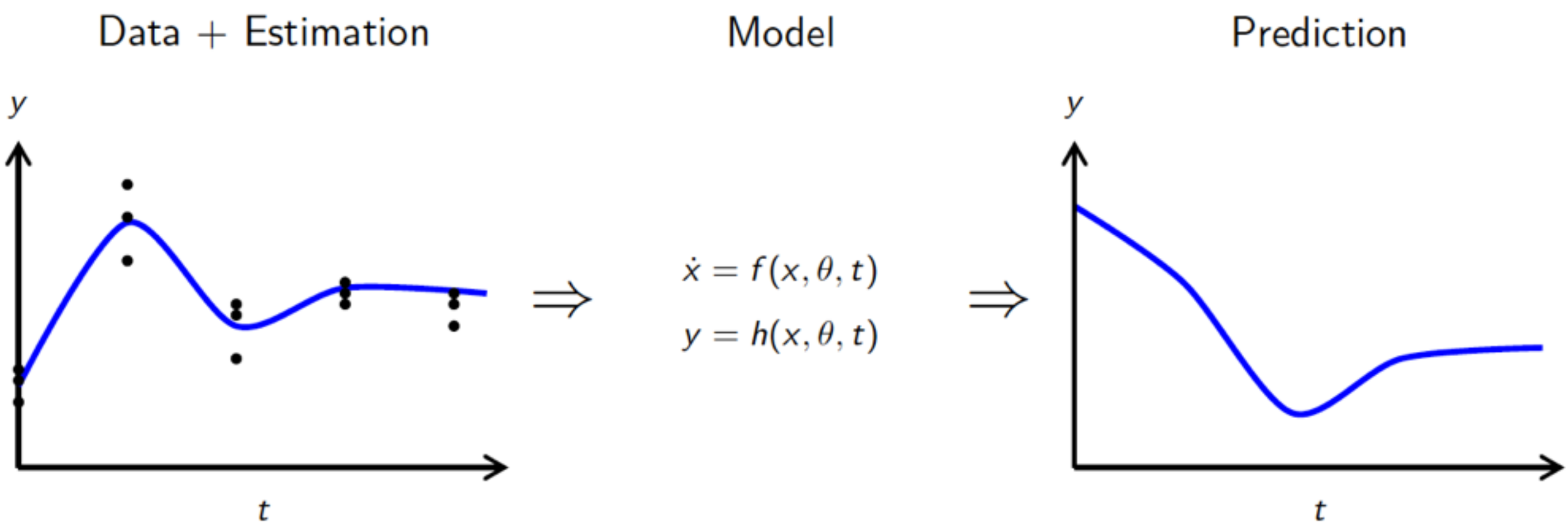
Model and prediction uncertainties

Optimisation-based approach

- Calculation of optimal parameter (e.g. least-square, or maximum-likelihood)

  => optimal model of the process

- Model rejection and prediction using optimal model



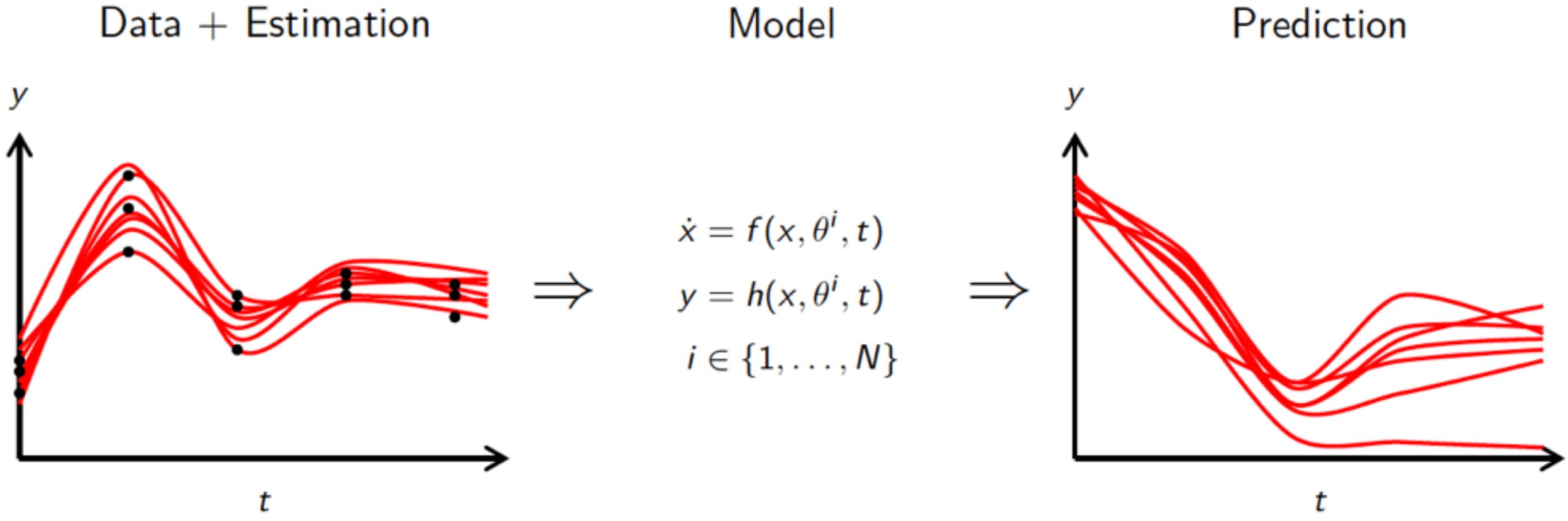
Model and prediction uncertainties

Sampling-based approach

- Calculation of a sample of good parameters (e.g. Bayesian, Bootstraping)

  => sample of good models of the process

- Model rejection and prediction using optimal model



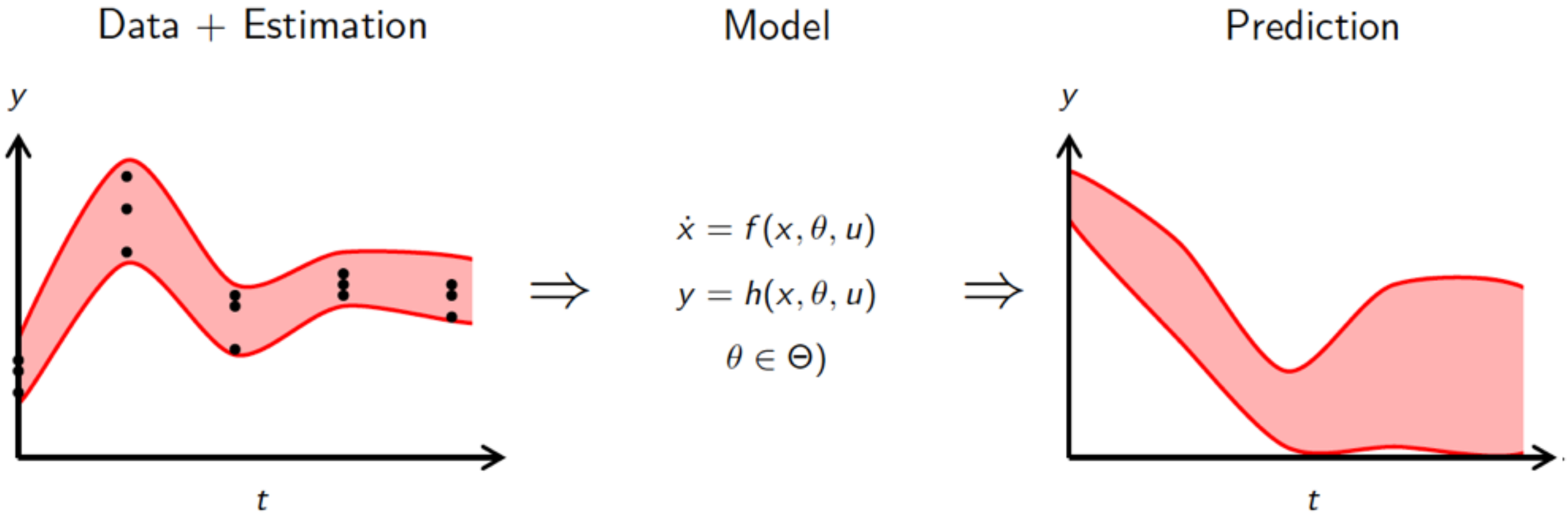
Model and prediction uncertainties

Set-based approach

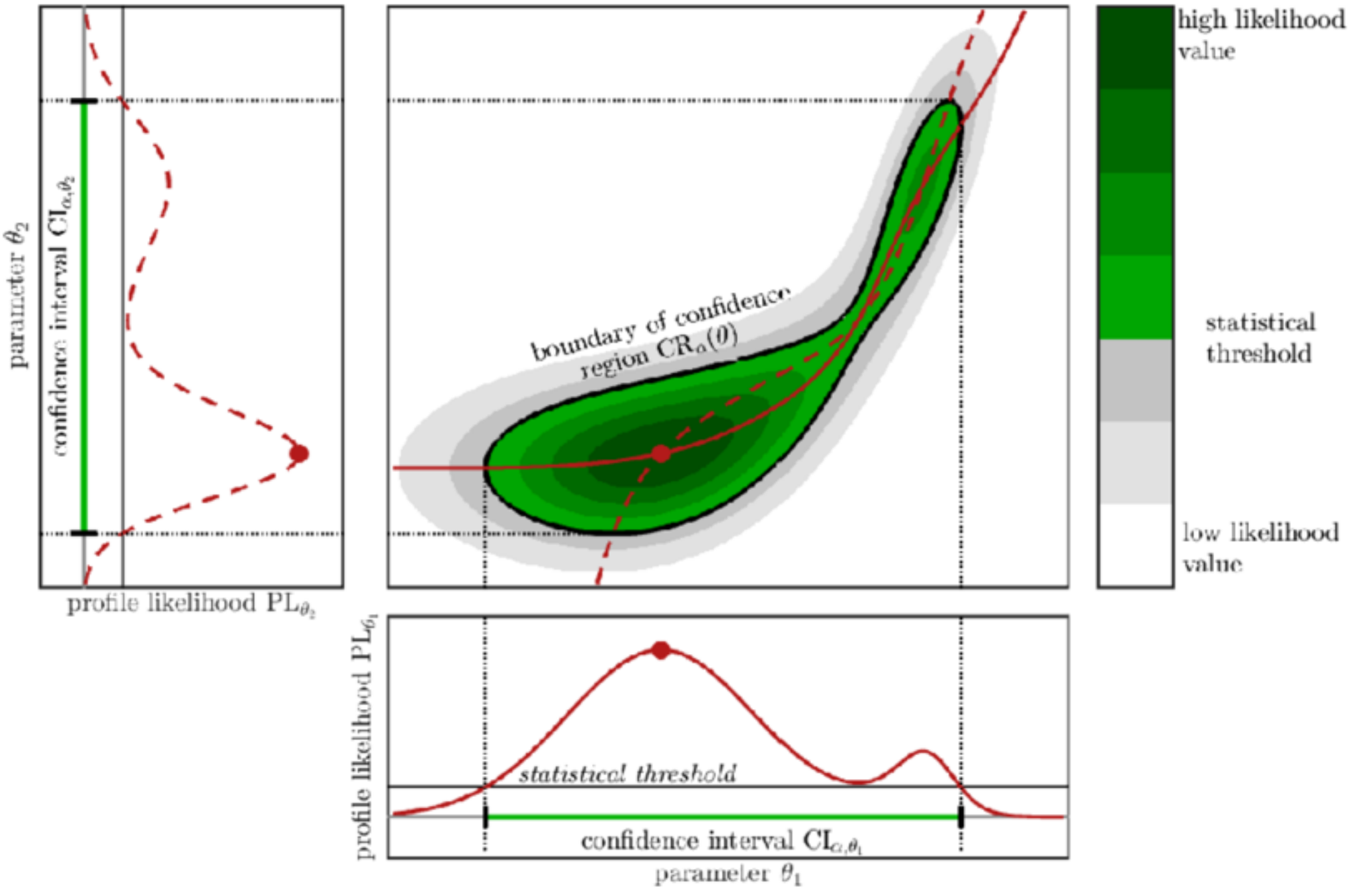
- Calculation of the set of consistent parameters (e.g.interval arithmetics, convex optimisation)

  => all consistent models of the process

- Model rejection and prediction using optimal model



Confidence interval and confidence region



Practical identifiability

Identifiability

A model property $g(\theta)$ is called *identifiable* if its confidence interval $CI_{\alpha,g(\theta)}$ is bounded. Otherwise, it is called *practically non-identifiable.*

Structural identifiability

Does not depend on data quality / quantity!

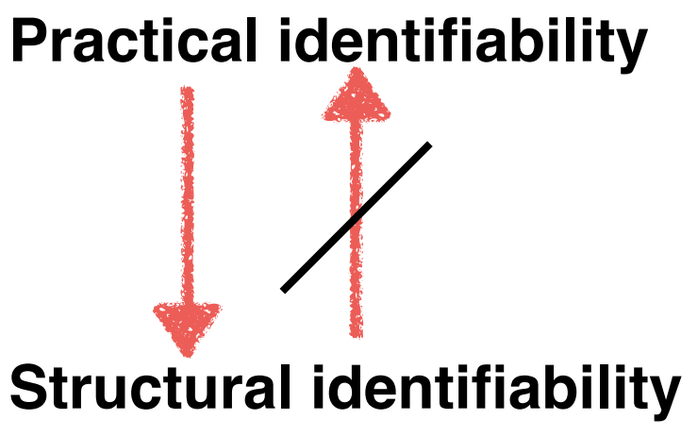
A model is structurally identifiable if it is possible to determine the values of its parameters from observations of its outputs and knowledge of its dynamic equations <=> if $J(\theta)$ has a (local) minimum at θ_0

Practical identifiability

Practical identifiability quantifies the uncertainty in parameter values when estimates from noisy measurements.

Available approaches and methods:

- Asymptotic confidence intervals
- Profile likelihood
- Sampling approaches



Literature

Sensitivities and optimisation

1. J. Nocedal. *Numerical Optimization.* Springer, Second Ed, 2006.
2. F. Fröhlich, B. Kaltenbacher, F. J. Theis, and J. Hasenauer. Scalable Parameter Estimation for Genome-Scale Biochemical Reaction Networks. PLoS Computational Biology, 13(1):e1005331, 2017.
3. R. Fletcher. *Practical Methods of Optimization*. Wiley, 2000.

Confidence intervals and model selection

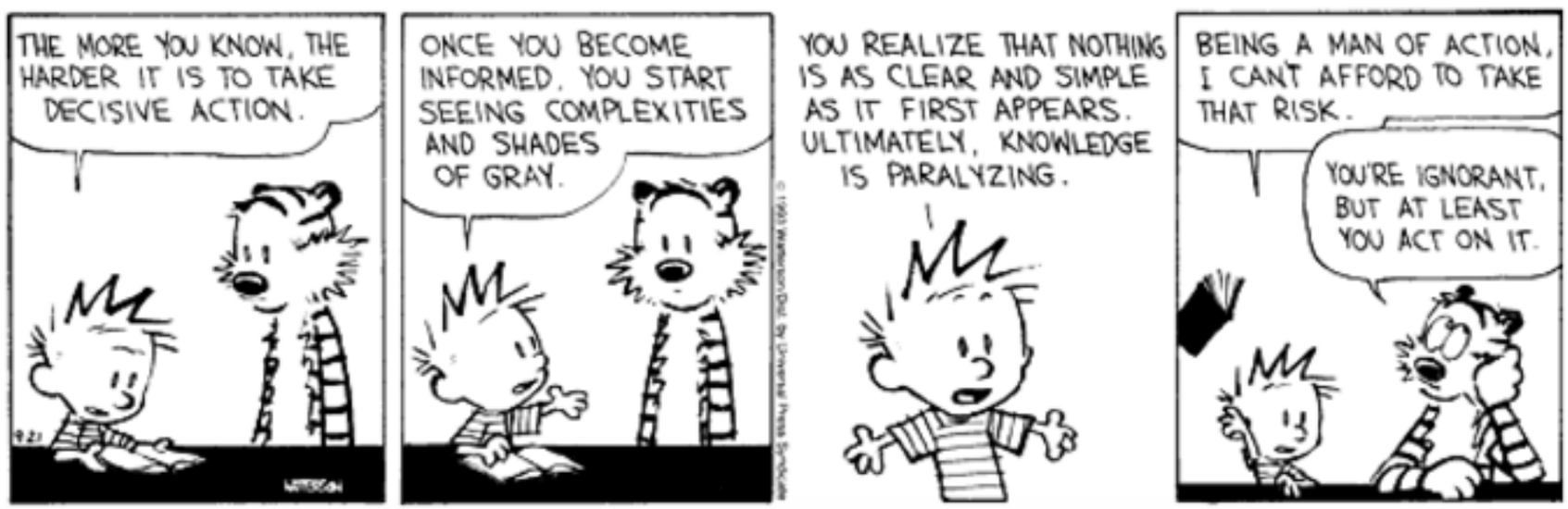
1. A. Raue, C. Kreutz, T. Maiwald, J. Bachmann, M. Schilling, U. Klingmüller, and J. Timmer. Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. Bioinformatics, 25(25):1923-1929, 2009.
2. K. P. Burnham, and D. R. Anderson. Model selection and multimodel inference: A practical information-theoretic approach, Springer, New York, NY. 2002.

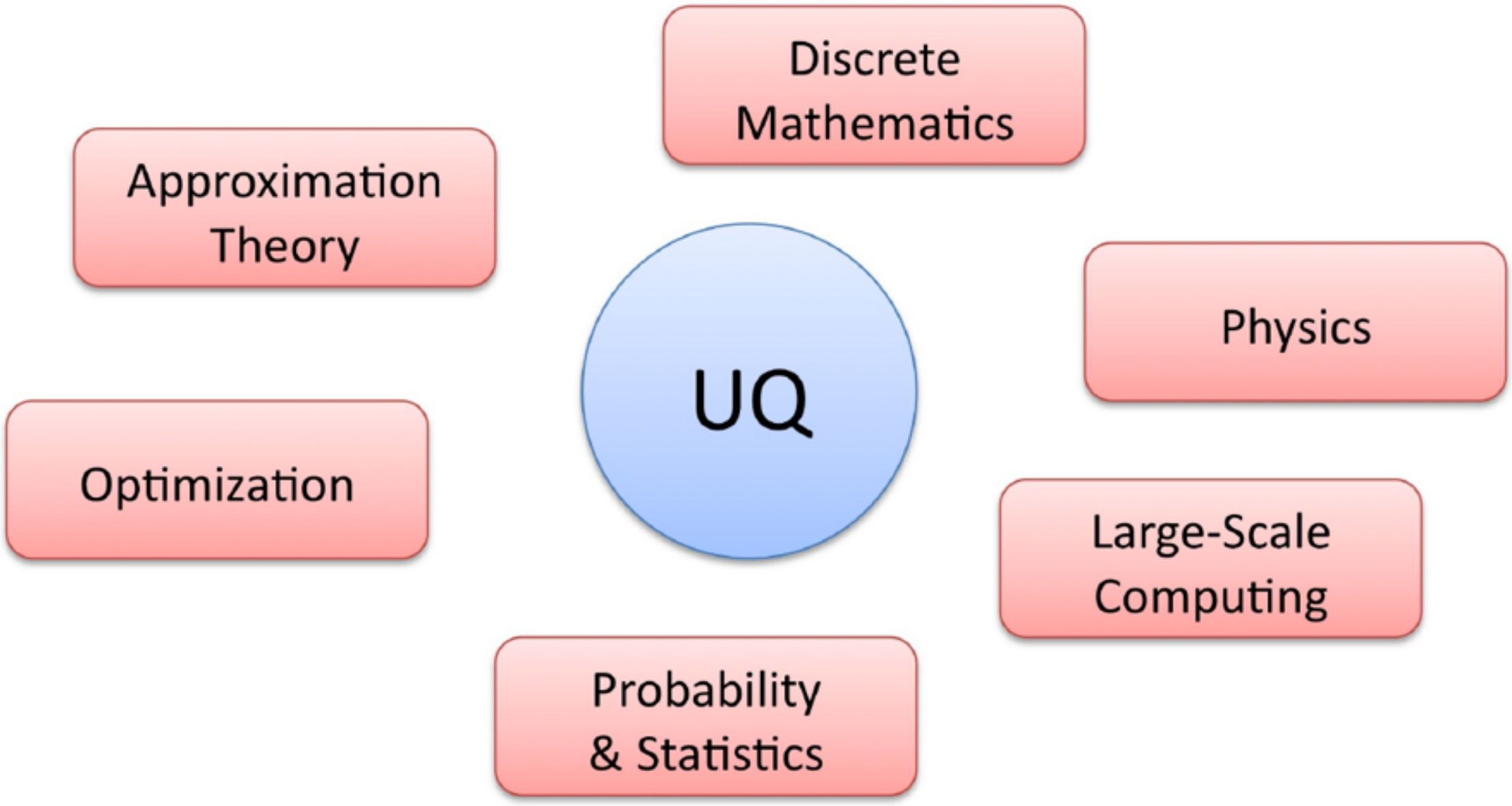
Uncertainty Quantification (UQ)

Definition from Wikipedia

Uncertainty quantification (UQ) is the science of quantitative characterization and reduction of uncertainties in applications. It tries to determine how likely certain outcomes are if some aspects of the system are not exactly known.

An example would be to predict the acceleration of a human body in a head-on crash with another car: even if we exactly knew the speed, small differences in the manufacturing of individual cars, how tightly every bolt has been tightened, etc, will lead to different results that can only be predicted in a statistical sense. [...]




Discrete Mathematics
Approximation Theory
UQ
Physics
Optimization
Large-Scale Computing
Probability & Statistics

Uncertainty Quantification

Why UQ? Decision Making

UQ is critical in identifying the confidence in an outcome.

Provides basis for certification in high-consequence decisions.

UQ is a fundamental component of model validation.

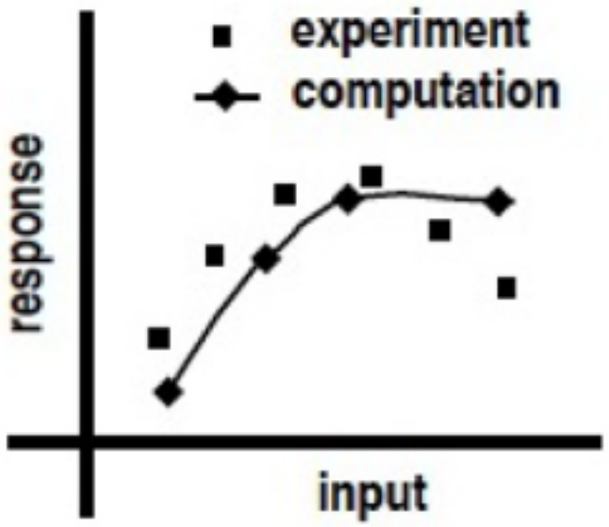
Required to identify the effect limited knowledge in inputs of the simulations

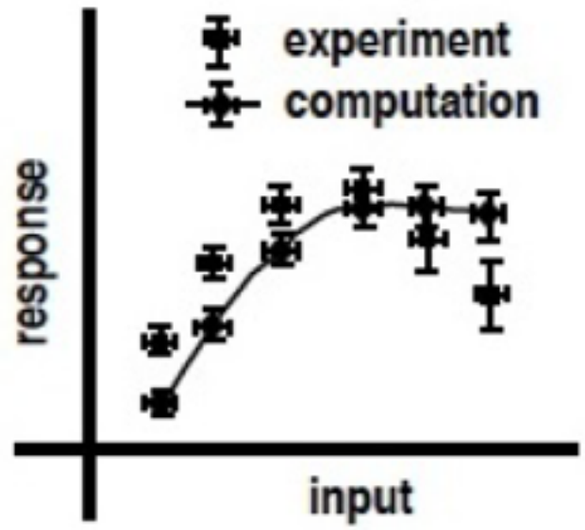
Uncertainty Quantification

Why UQ?

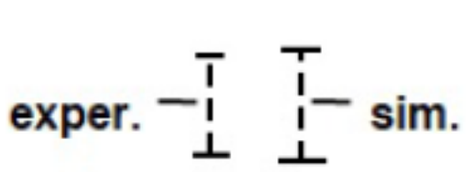
In spite of the wide spread use of simulations it remains difficult to provide objective confidence levels

One of the objective of UQ is to add error bars



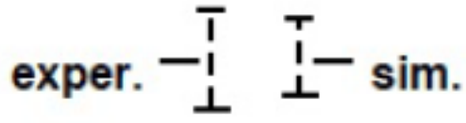


… But also the precise notion of validated model



"optimal"

less "optimal"



VIOLATION

Definitions and notations

"As we know there are known knowns.There are things we know we know. We also know there are known unknowns.That is to say, we know there are some things we do not know. But there are also unknown unknowns,The ones we don't know we don't know."

D. Rumsfeld, 2002, Department of Defense news briefing

The American Institute for Aeronautics and Astronautics (AIAA) has developed the *"Guide for the Verification and Validation (V&V) of Computational Fluid Dynamics Simulations"* (1998)

- **Verification**: The process of determining that a model implementation accurately represents the developer's conceptual description of the model.

  "are we solving the equations correctly?" – it is an exercise in mathematics

- **Validation**: The process of determining the degree to which a model is an accurate representation of the real world for the intended uses of the model.

  "are we solving the correct equations?" – it is an exercise in physics

Definitions and notations

According to the *"Guide for the Verification and Validation (V&V) of Computational Fluid Dynamics Simulations"* (1998)

- **Errors** as recognisable deficiencies of the models or the algorithms employed
- **Uncertainties**: as a potential deficiency that is due to lack of knowledge.

  - ***The definitions are not very precise***
  - ***Do not clearly distinguish between the mathematics and the physics.***
  - ***What is the relation to V&V?***

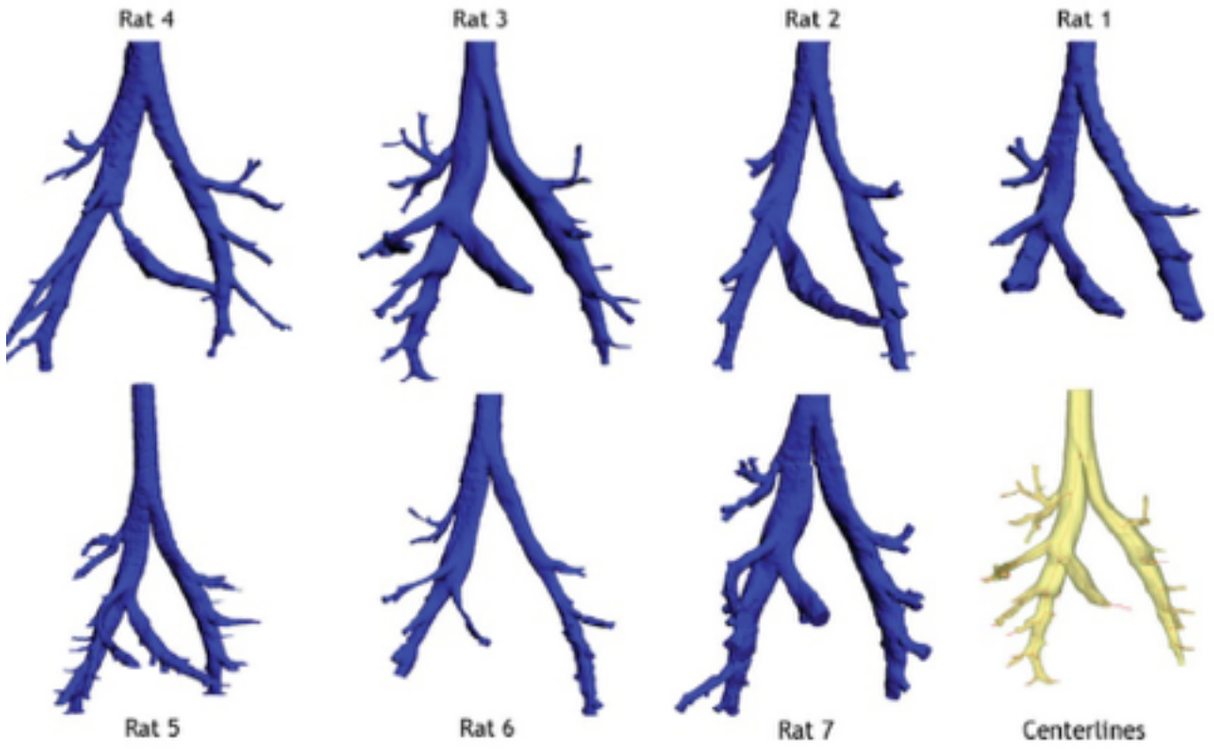
Definitions and notations

- **What are errors?** errors are associated to the translation of a mathematical formulation into a numerical algorithm and a computational code.
  - roundoff, limited convergence of iterative algorithms;
  - implementation mistakes (bugs);
  - is the mathematics…
- **What are uncertainties**? uncertainties are associated to the specification of the input physical parameters required for performing the analysis.
  - is the physics….

Definitions and notations

Uncertainties

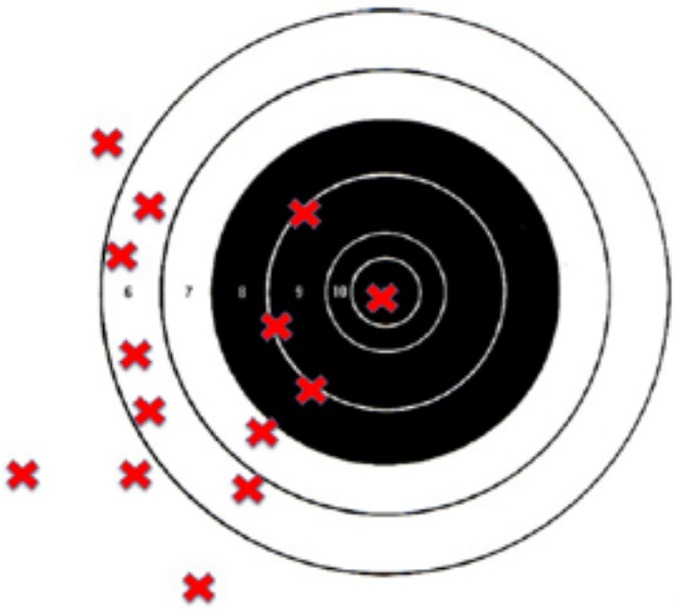
- **Aleatory:** it is the physical variability present in the system or its environment.
  - It is not strictly due to a lack of knowledge and cannot be reduced (also referred to as variability, stochastic uncertainty or **irreducible uncertainty**);
  - **It is naturally defined in a probabilistic framework;**
  - Examples are: material properties, operating conditions manufacturing tolerances, etc.
- In mathematical modelling it is also studied as **noise**.

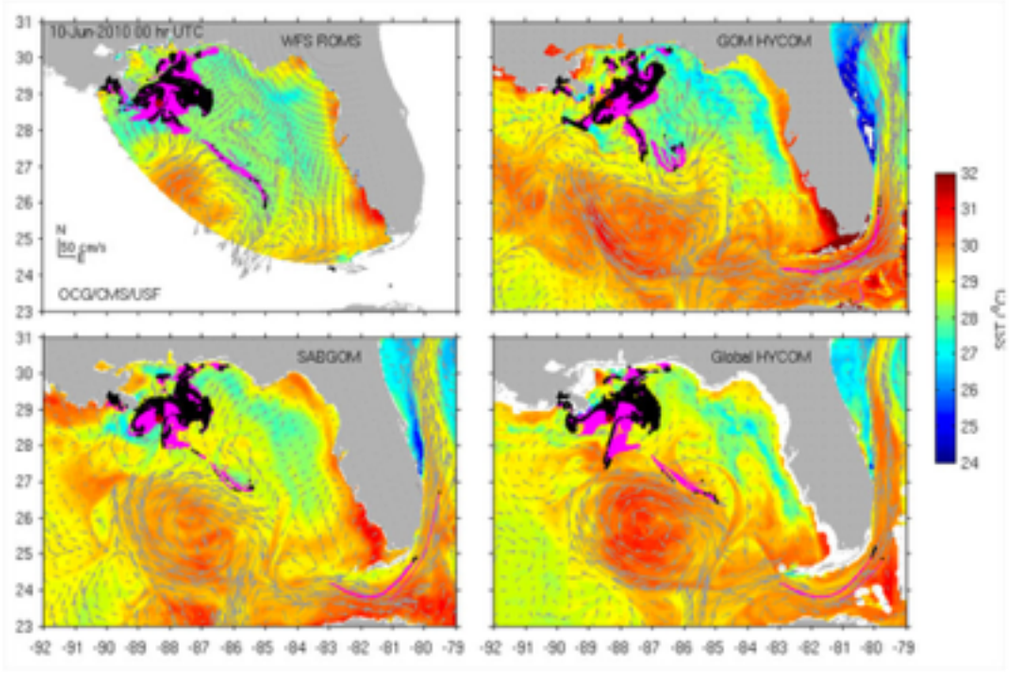


Definitions and notations

Uncertainties

- **Epistemic:** it is a potential deficiency that is due to a lack of knowledge
  - It can arise from assumptions introduced in the derivation of the mathematical model (it is also called **reducible uncertainty** or incertitude);
  - Examples are: turbulence model assumptions or surrogate chemical models;
  - **It is NOT naturally defined in a probabilistic framework;**
- Can lead to strong **bias** of the predictions.





Deep water horizon oil tracking forecast

Definitions and notations

Summary: Not all uncertainties created equal..

- **Uncertainties relate to the physics of the problem** of interest! not to the errors in the mathematical description/solution…

- Reducible vs. Irreducible Uncertainty
  - **Epistemic uncertainty can be reduced** by increasing our knowledge, e.g. performing more experimental investigations and/or developing new physical models.
  - **Aleatory uncertainty cannot be reduced** as it arises naturally from observations of the system. Additional experiments can only be used to better characterize the variability.

- **Sensitivity vs Uncertainty Analysis**
  - **Sensitivity analysis** investigates the connection between inputs and outputs of a (computational) model
  - **Uncertainty analysis** aims at identifying the overall output uncertainty in a given system.

Computations under Uncertainty
= Predictive Simulations

"The significant problems we face cannot be solved at the same level of thinking we were at when we created them."

A. Einstein

Consider a generic computational model in high dimensions

How do we handle the uncertainties?

1. **Uncertainty definition:** characterize uncertainties in the inputs
2. **Uncertainty propagation:** perform simulations accounting for the identified uncertainties
3. **Certification**: establish acceptance criteria for predictions

Uncertainty Definition

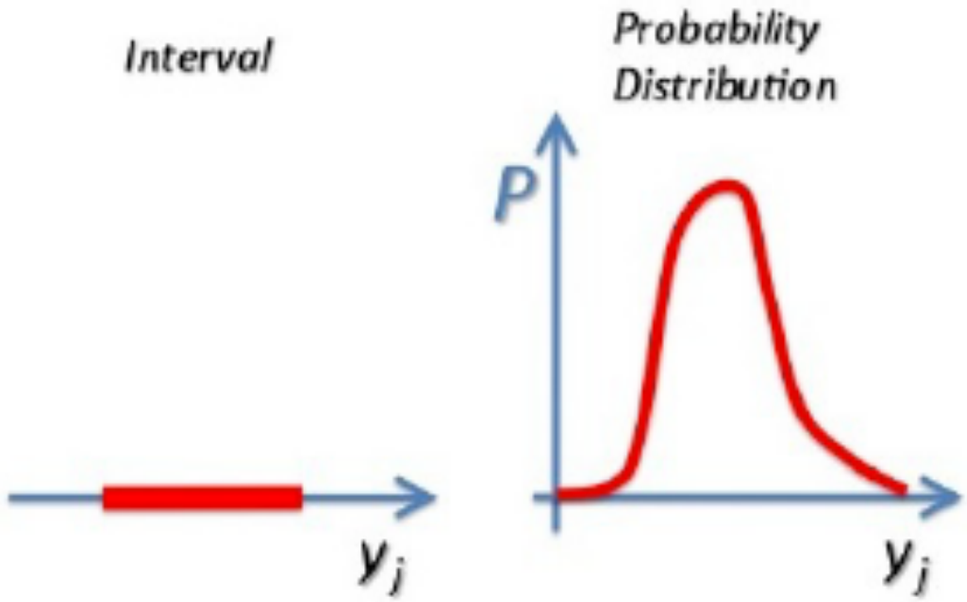
The objective is characterize uncertainties in simulation inputs, based on **available information**

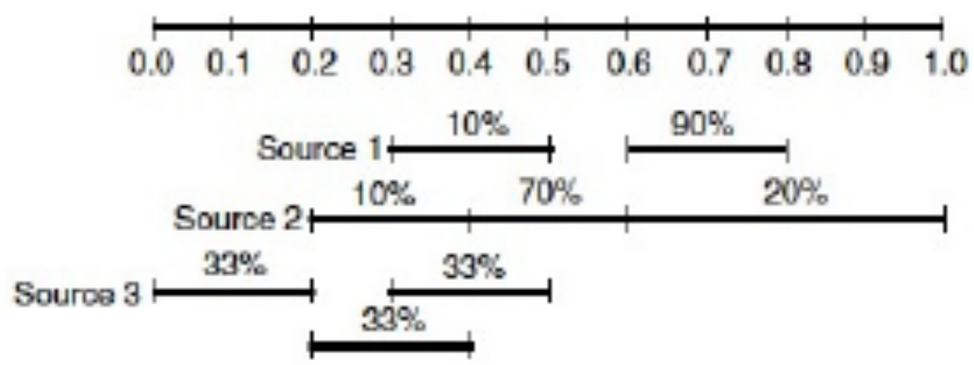
- **Direct methods:** Experimental observations / Theoretical arguments / Expert opinions, etc.
- **Inverse methods (Inference, Calibration):** determination of the statistical input parameters that represent observed data using a computational model

Uncertainty Definition

Identification of all the explicit and hidden parameters of the mathematical/ computational model **y**

Characterization of the associated level of knowledge





The mathematical framework for propagating uncertainties is dependent on the data representation chosen

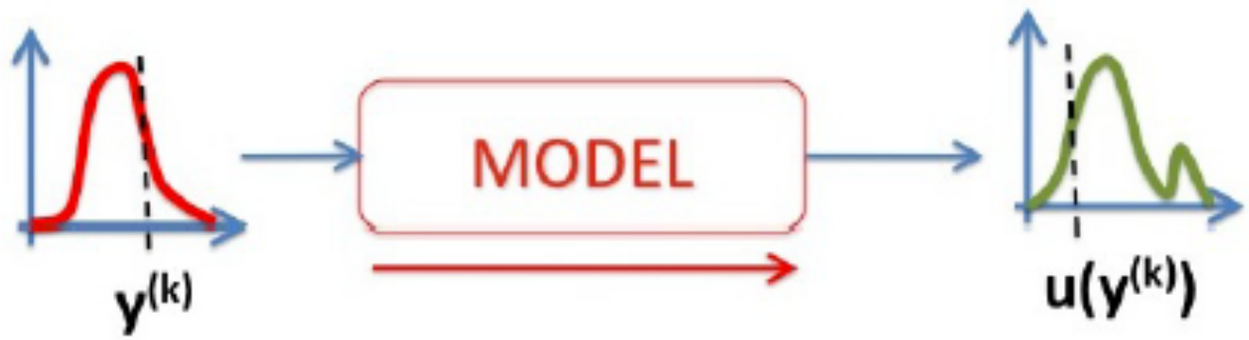
Uncertainty Propagation

Perform simulations accounting for the uncertainty represented as **randomness**

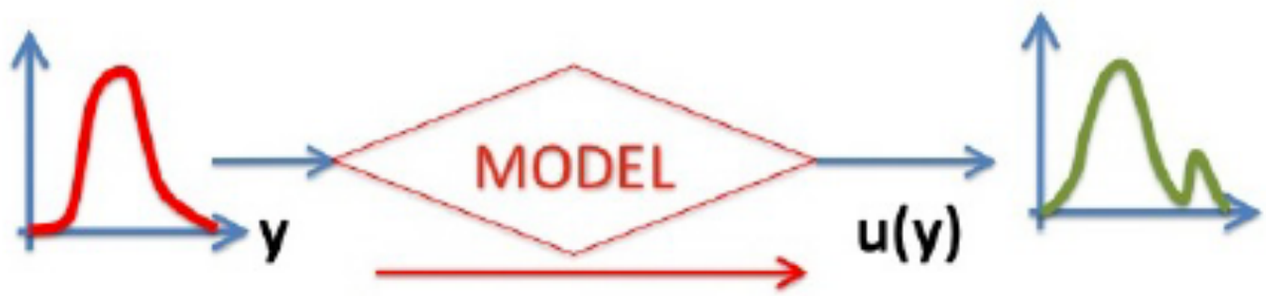
- Define an abstract probability space $(\Omega, \mathcal{A}, \mathcal{P})$
- Introduce uncertain input as **random quantities** $y(\omega), \quad \omega \in \Omega$
- The original problem becomes **stochastic** with solution $u(\omega) = u(y(\omega))$

Uncertainty Propagation

Nonintrusive methods only require (multiple) solutions of the **original** (deterministic) model



Intrusive methods require the formulation and solution of a **stochastic** version of the original problem



Uncertainty Propagation

Nonintrusive methods only require (multiple) solutions of the **original** (deterministic) model

+ Simple extension of the "conventional" simulation paradigm
+ Embarrassingly parallel: solutions are independent
+ Conceptually very simple

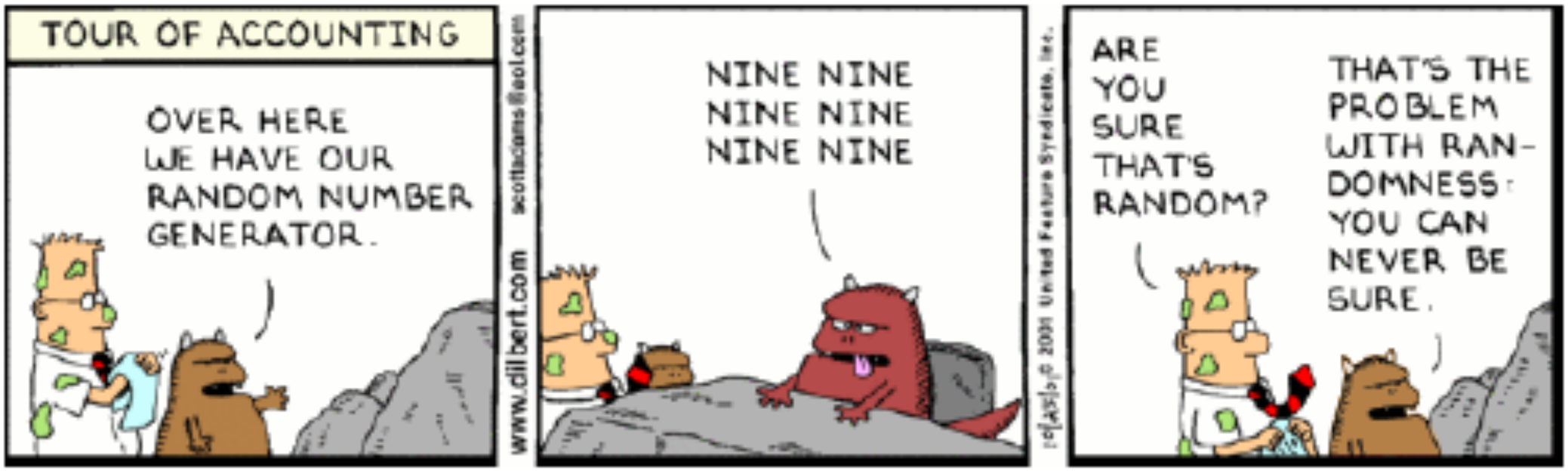
Intrusive methods require the formulation and solution of a **stochastic** version of the original problem

+ Exploit the mathematical structure of the problem
+ Leverage theoretical & algorithmic advancements

(Probabilistic) Uncertainty Propagation

Uncertainty = Randomness

- Sampling Methods: Monte Carlo, Quasi Monte Carlo, Lati Hypercube, etc.
- Intrusive Methods: Polynomial Chaos, Adjoints, etc.
- Non-Intrusive Methods: Stochastic Collocation, Response Surface, etc.
- Optimization Methods



Certification

Certification and Validation