Optimization methods in biochemical modeling

Sven Sahle

Introduction

- When starting a modeling project usually many parameters of the model are not known
- How can I find out about the effect of parameter changes in the model?
- How can I find out about parameter values?

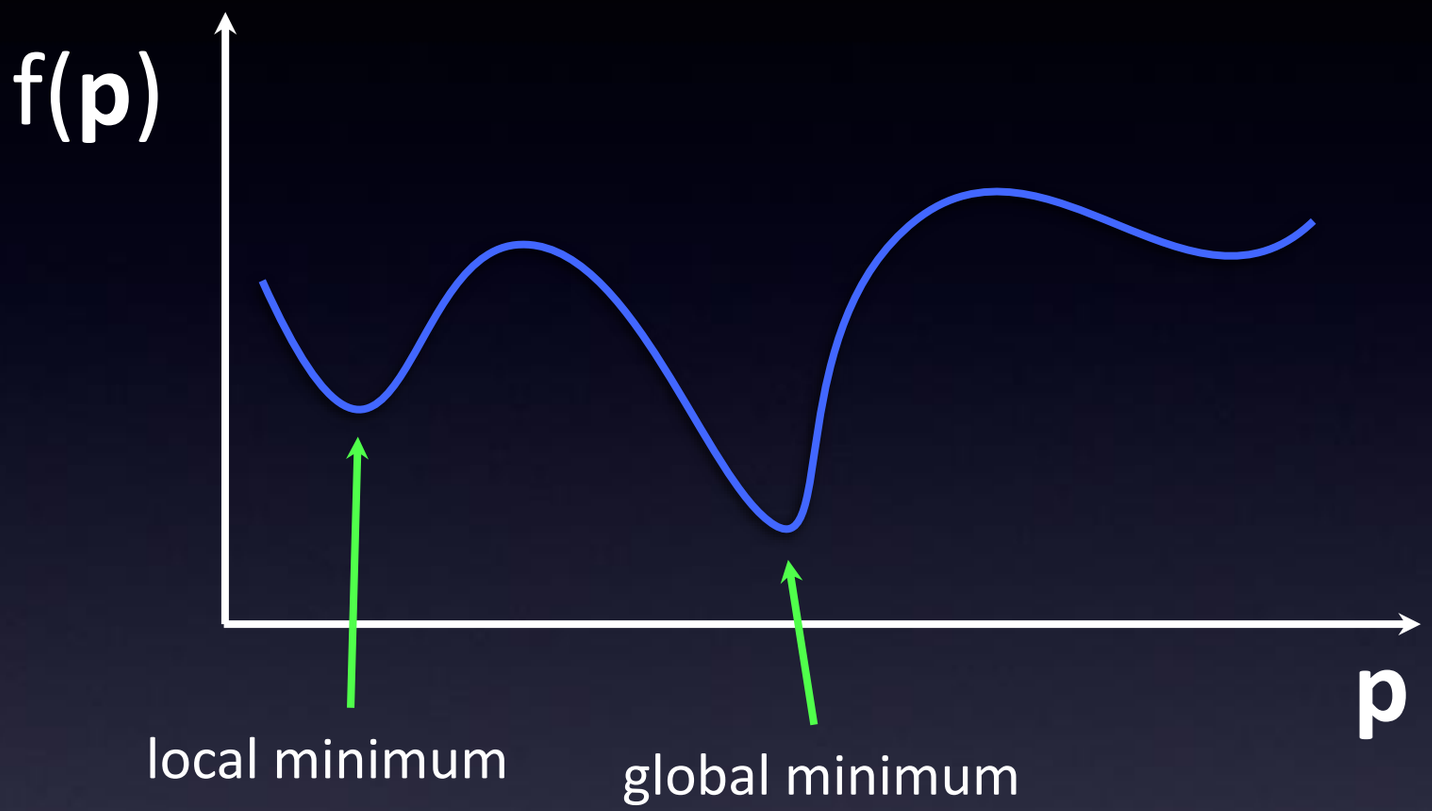
Introduction

- Direct approaches to find parameter effects:
  - change **parameters** and observe **behaviour** (parameter scans, sampling)
    - sensitivity analysis, control theory
- „Reverse“ approaches
  - specify desired **behaviour** and let the computer figure out **parameters**
    - numerical optimization, parameter estimation, ...

Introduction

- Mathematically:
  - specify a function that can be calculated from the model, i.e. from simulation results. This function quantifies **behaviour**
  - specify an algorithm that changes the **parameters** of the model so that the function is maximized or minimized

The target function



The function is usually high dimensional!

- This is the function that needs to be minimized.
- usually has many local minima

Parameter space

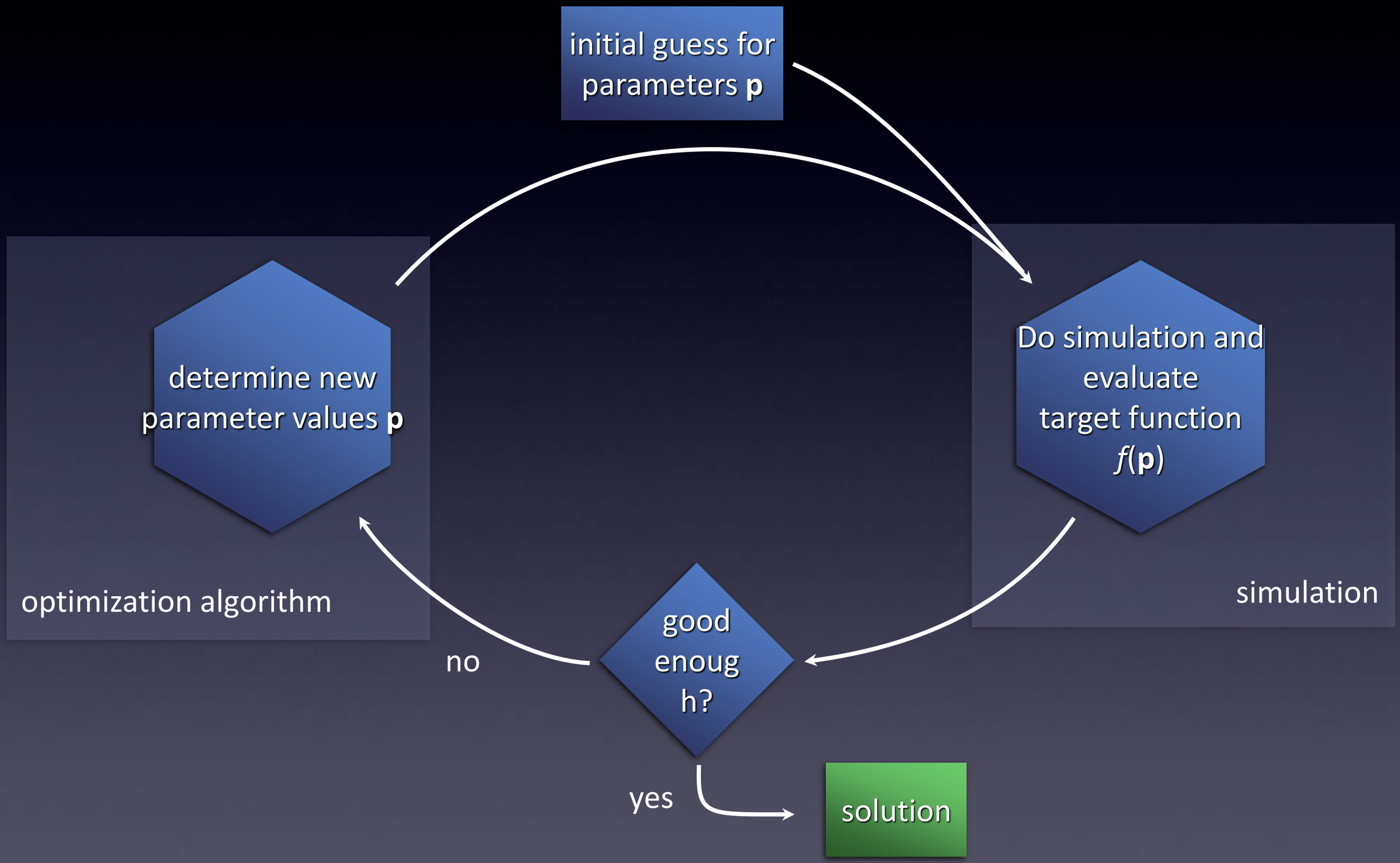
- For a complete specification of the optimization problem we need to specify the unknown parameters:
- List of parameters with allowed range of values
- These parameters span an M-dimensional space, the *parameter space.*
- One specific set of parameters corresponds to a point in parameter space

Optimization problem

- We now need a way to find the set of parameter values (a point in parameter space) for which the function f(**p**) is minimal/maximal .

- A systematic scan of the parameter space is not possible when the dimensionality is large (many unknown parameters)
- Example: 10 parameters with 10 values each: 10^{10} evaluations. Even if we can do 100 simulations/s, it would take 3 years.

Numerical optimization cycle



Optimization algorithms

- In general it is very difficult to find the parameter values for which f(**p**) is minimal
- It can be shown that there is no optimal optimization algorithm for all cases (this means there is no way to decide which one is best for a given problem)
- This means you should always try several algorithms for difficult optimization problems.

Uses of optimization in biological modeling

- Engineering: Modifying organisms for specific goals
- solve biological questions: What does evolution optimize?
- technical use in modeling: Find out what a model can do, deal with parameter uncertaincy, estimate parameters

COPASI

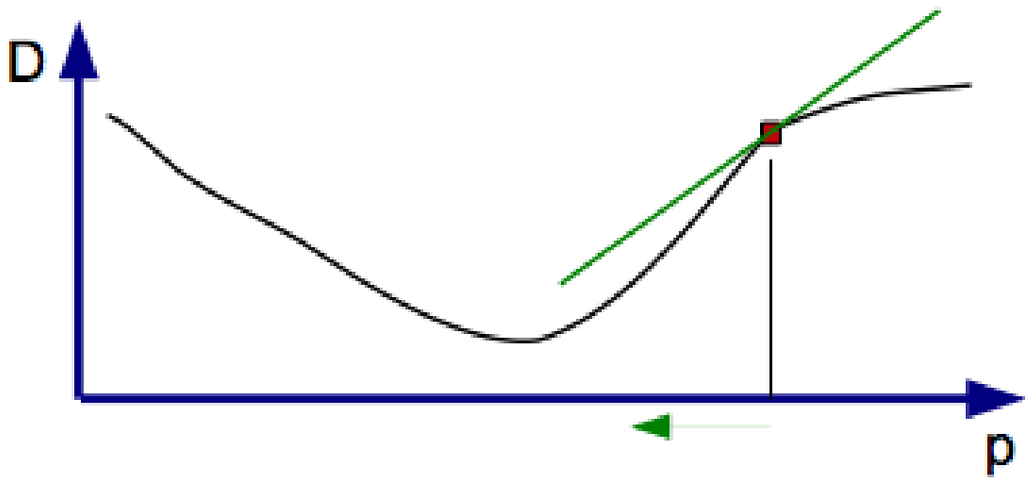
Optimization Algorithms

- Based on derivatives
  - Steepest descent
  - Newton
  - Levenberg-Marquardt
- Using geometry
  - Nelder-Mead (simplex)
  - Hooke-Jeeves
- Based on genetics

Some algorithms

- **Steepest descent:** Just go down along a gradient.
  - Will only find local minima, requires derivatives, not very fast convergence
- **Newton´s method:** Find root of derivatives.
  - global convergence not predictable. Very fast local convergence.
- **Levenberg-Marquard:** Combination of steepest decent and Newton´s method

Steepest descent

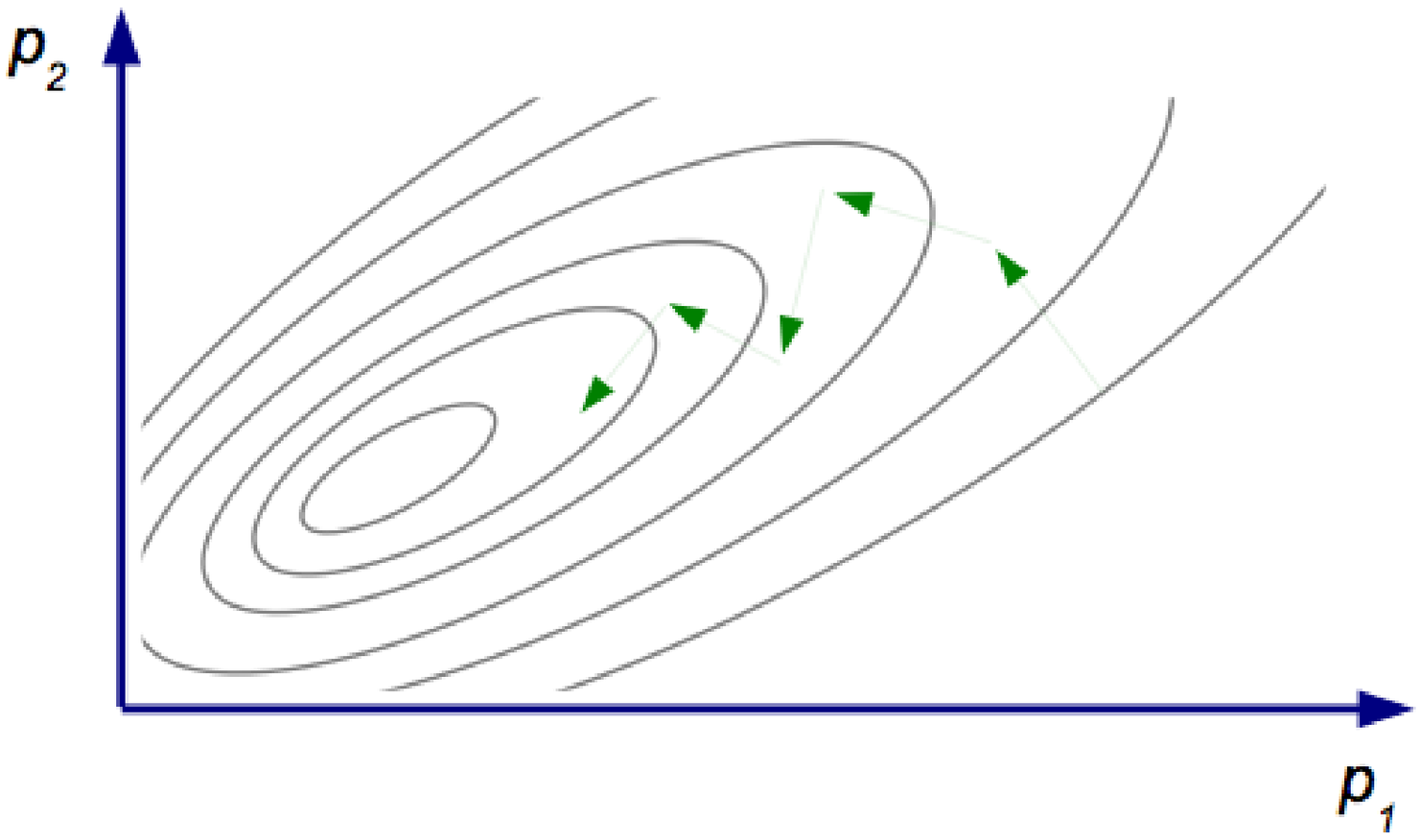


The algorithm only considers the first derivative

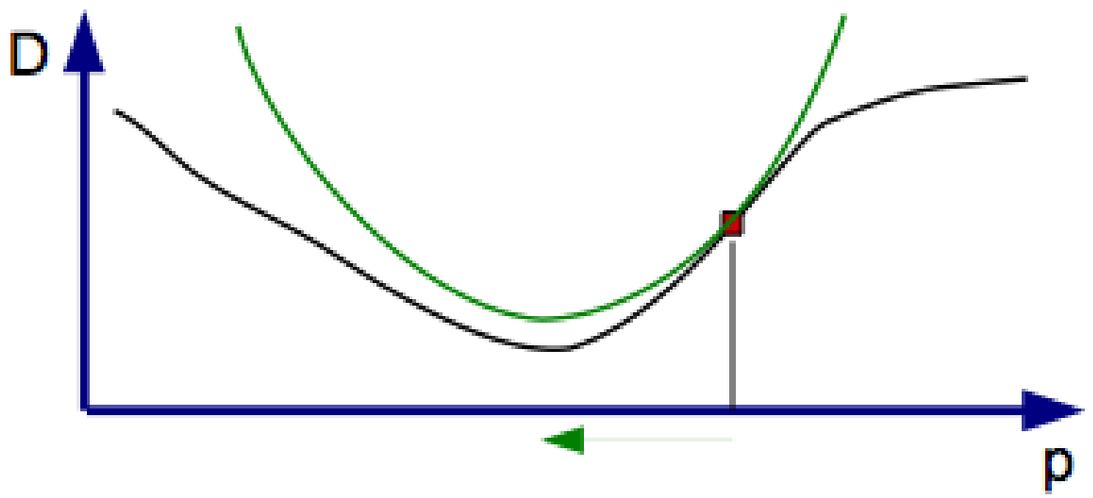
It does not know how far it should go

It does not take the shortest path

It is robust in finding a local minimum



Newton's method



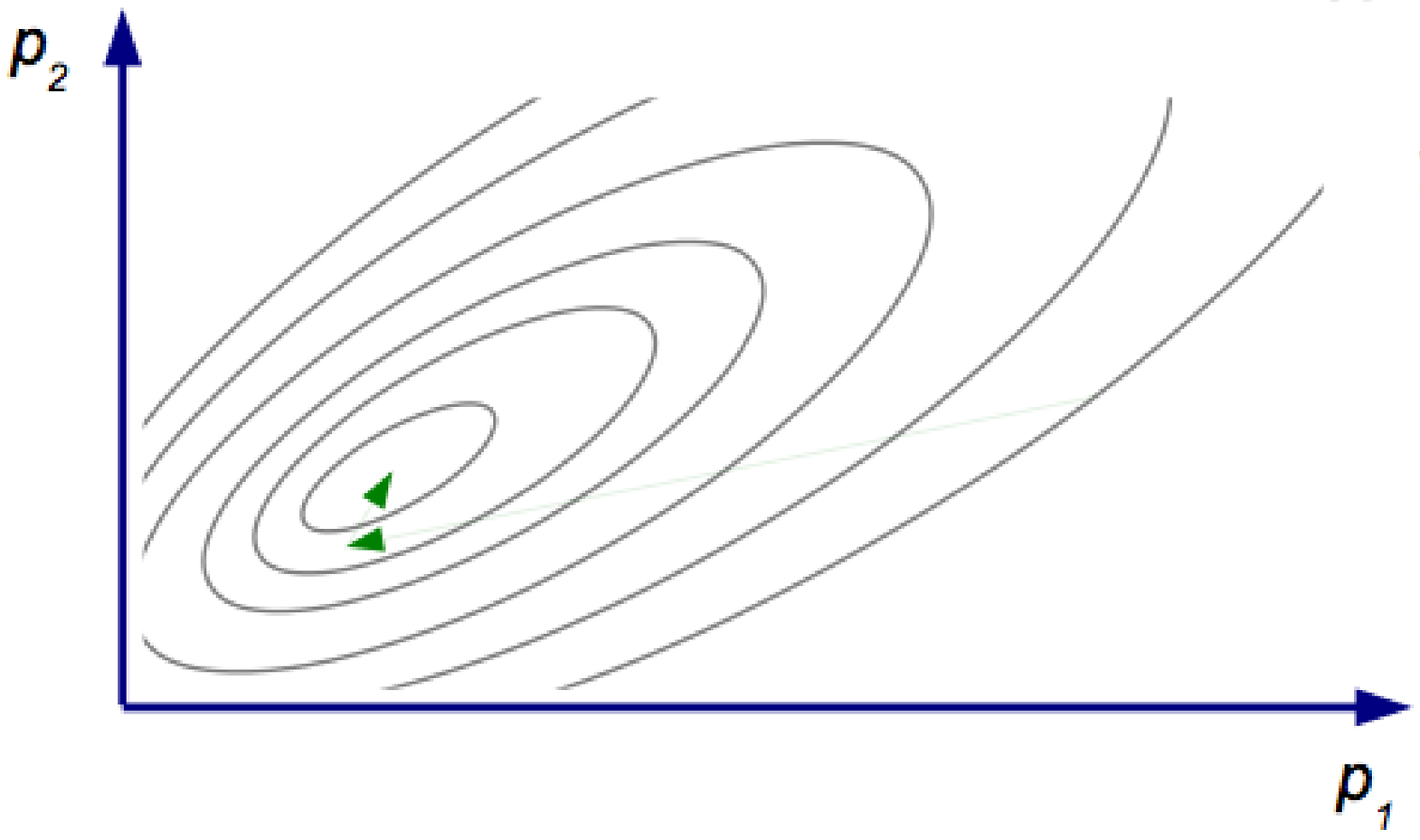
The algorithm considers the first and second derivative. It tries to find the root of the first derivative, which is a minimum or maximum of the function itself

It can guess how far the step should go

It tries to find the direction towards the minimum, not just the direction of the down slope

Convergence is extremely fast

It is unpredictible what happens if you have a bad first guess

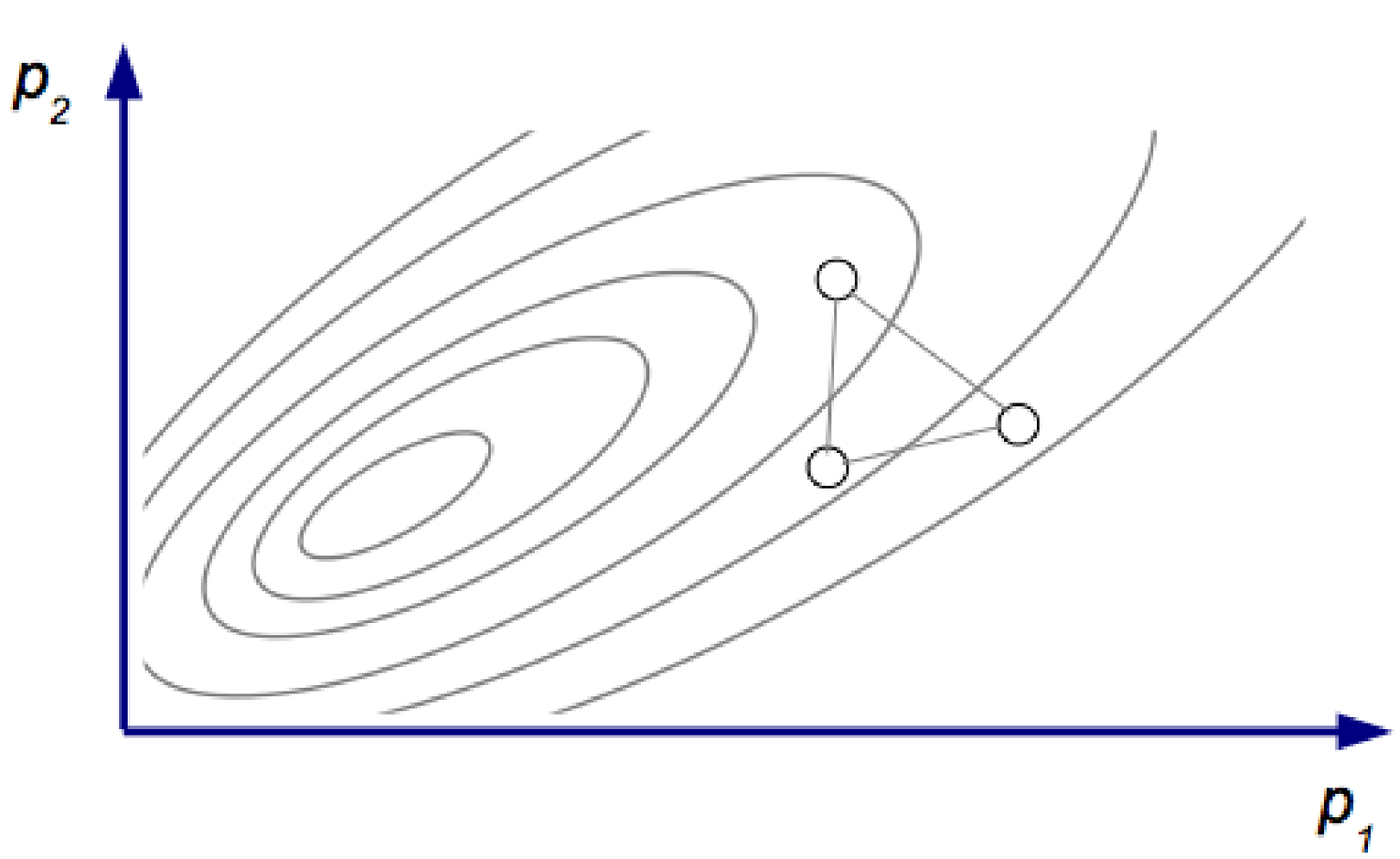


more algorithms

- **Nelder-Mead (simplex)** and **Hooke-Jeeves** don´t need derivatives. They try to find the downwards direction from several discrete evaluations (in a systematic way)

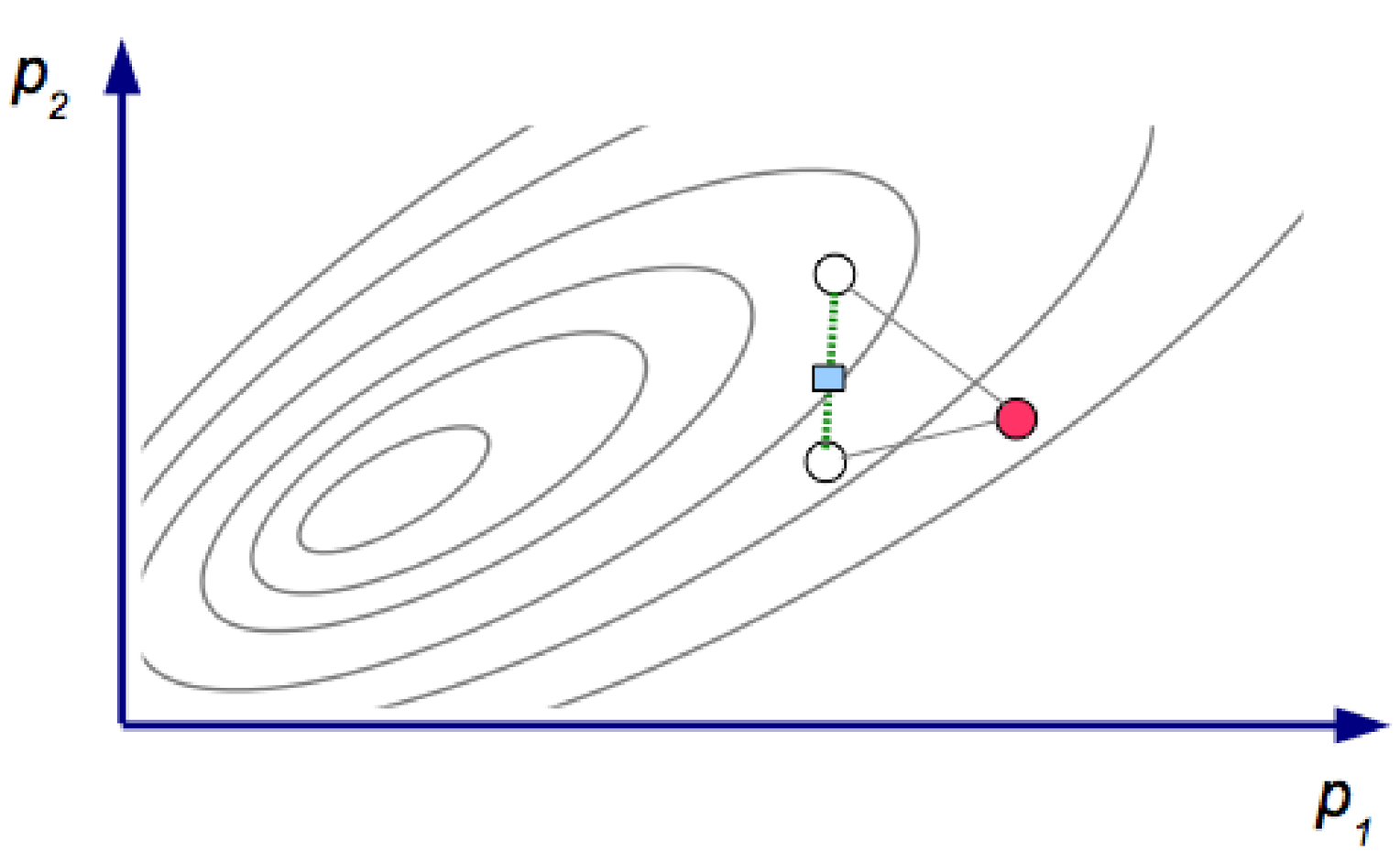
Nelder-Mead

A polygon of N+1 parameter sets is randomly chosen (the *simplex*). The worst performing one (in red) is then mirrored by the center of the remaining points.



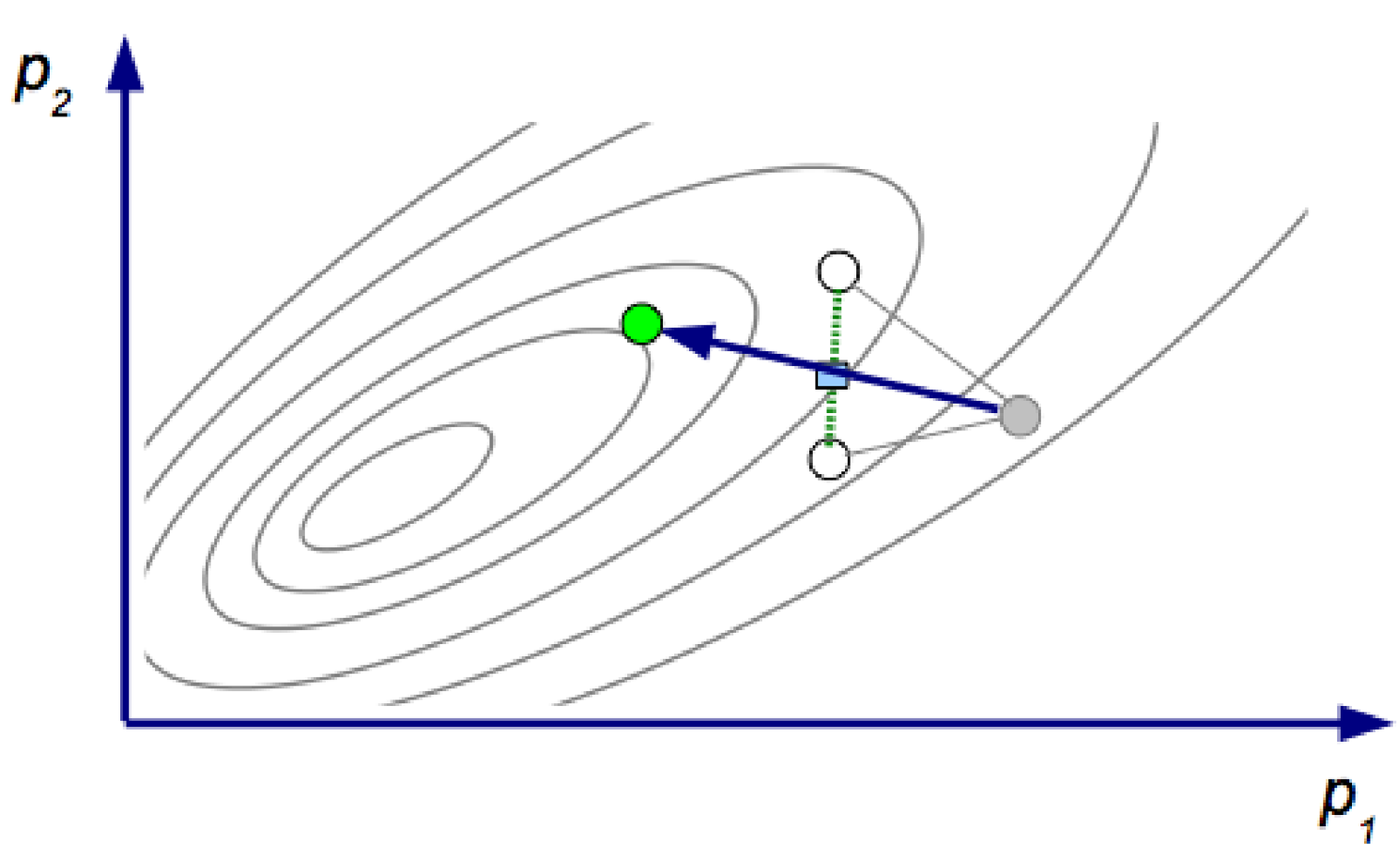
Nelder-Mead

A polygon of N+1 parameter sets is randomly chosen (the *simplex*). The worst performing one (in red) is then mirrored by the center of the remaining points.



Nelder-Mead

A polygon of N+1 parameter sets is randomly chosen (the *simplex*). The worst performing one (in red) is then mirrored by the center of the remaining points.



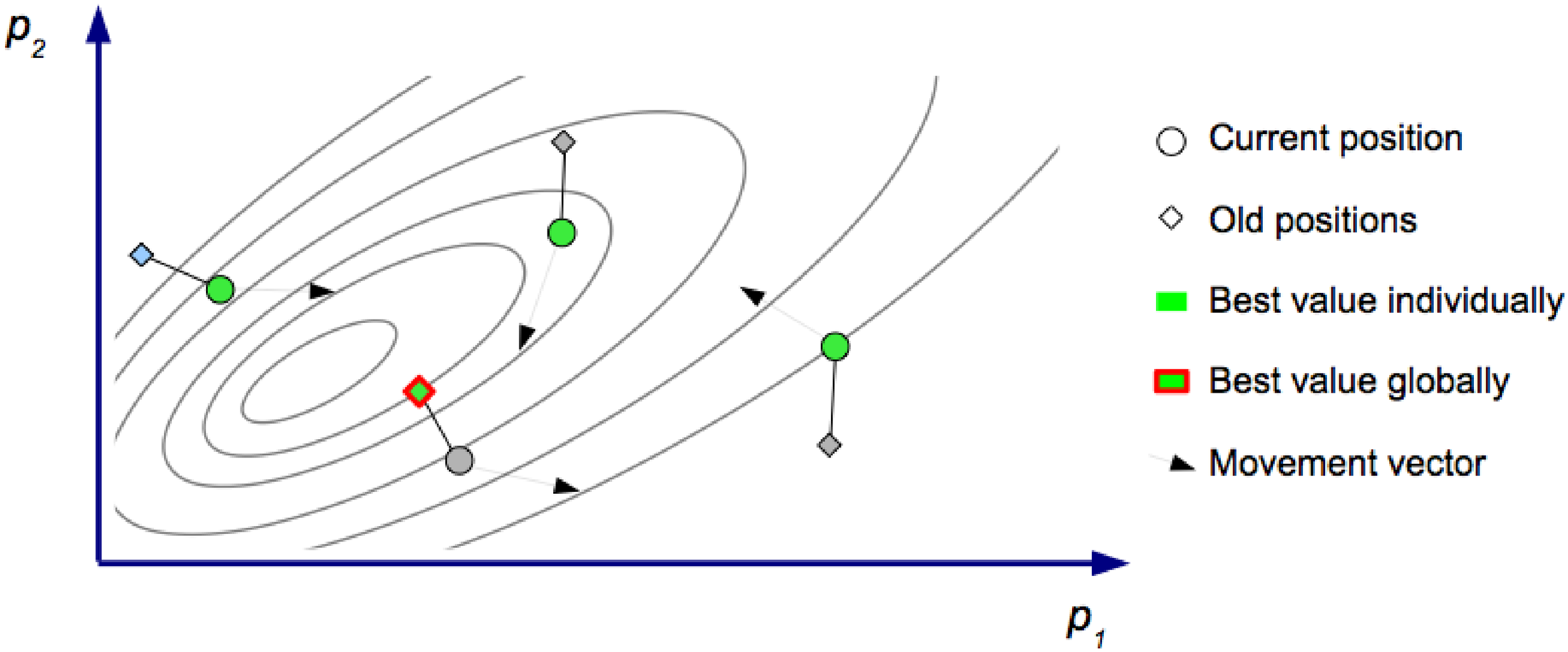
Global optimizers

- Most trivial: Random search
  - Will always work
  - Is extremely inefficient
- The other (more sophisticated) methods use some kind of heuristics, or even a variety of metaphors
  - Usually we cannot predict how good they will work
  - We can often not really say why exactly they work

- **Particle swarm:** Modeled after a flock of birds (or fish). We have a population of individual parameter sets that move around randomly in parameter space. However, they communicate about the best solution they have found so far, so that all individuals tend to move in the direction of the best fit

Particle swarm

We consider a population of several parameter sets (4 in this illustration) that move around the parameter space. Each has a direction of movement, and a memory of the best value it has seen so far. Each also knows the best overall value so far.
At each step the new position in parameter space is calculated by randomly combining the current direction of movement, and the tendencies to move towards the individual and the global best known value.



Genetic algorithms

- Based on evolutionary concepts.
  - parameter set = genome
  - simulation result = phenotype
  - $D(\mathbf{p})$ = fitness
- There is a population of parameter sets. The fitness is evaluated and some selection is performed. Then mutation is applied to get a new population.
- Different implementations differ in the kind of mutation, selection.

Others

- **Simulated annealing:** modeled after crystal formation in metals. Starts with high temperature (large random changes in parameters) and then cools down. Is guaranteed to find the global optimum if the cooling is infinitely slow...
- **Random search:** trivial algorithms. There are problems for which random search is the best algorithm

Specification of a parameter estimation problem

- What kind of information is needed for the computer to do a parameter estimation?
  - The model
  - the experimental data
  - the mapping between experimental data and model simulation results
  - the ranges of possible values for the unknown parameters
  - the optimization algorithm

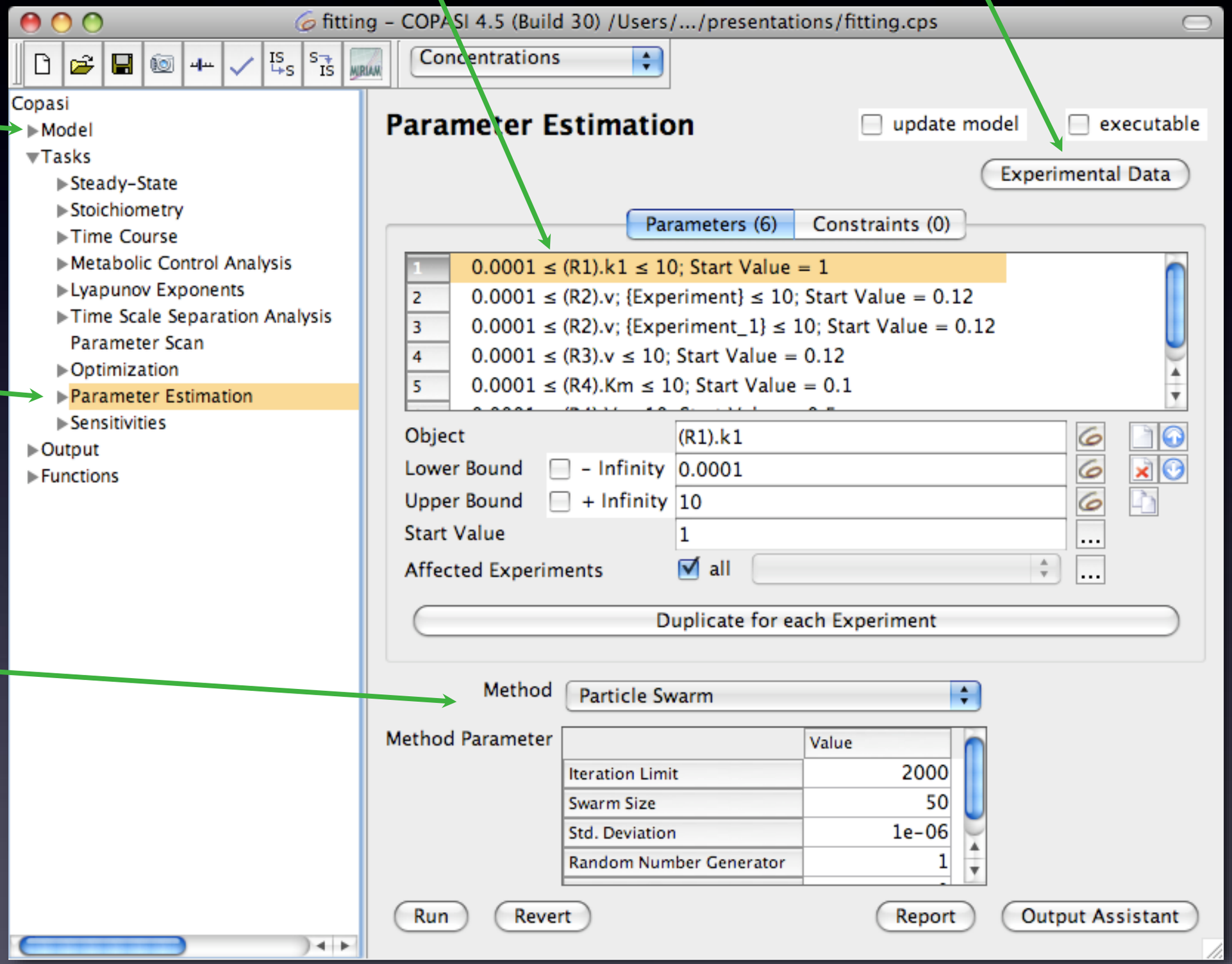
list of unknown parameters (including ranges)

choose experimental data

Model

select parameter estimation

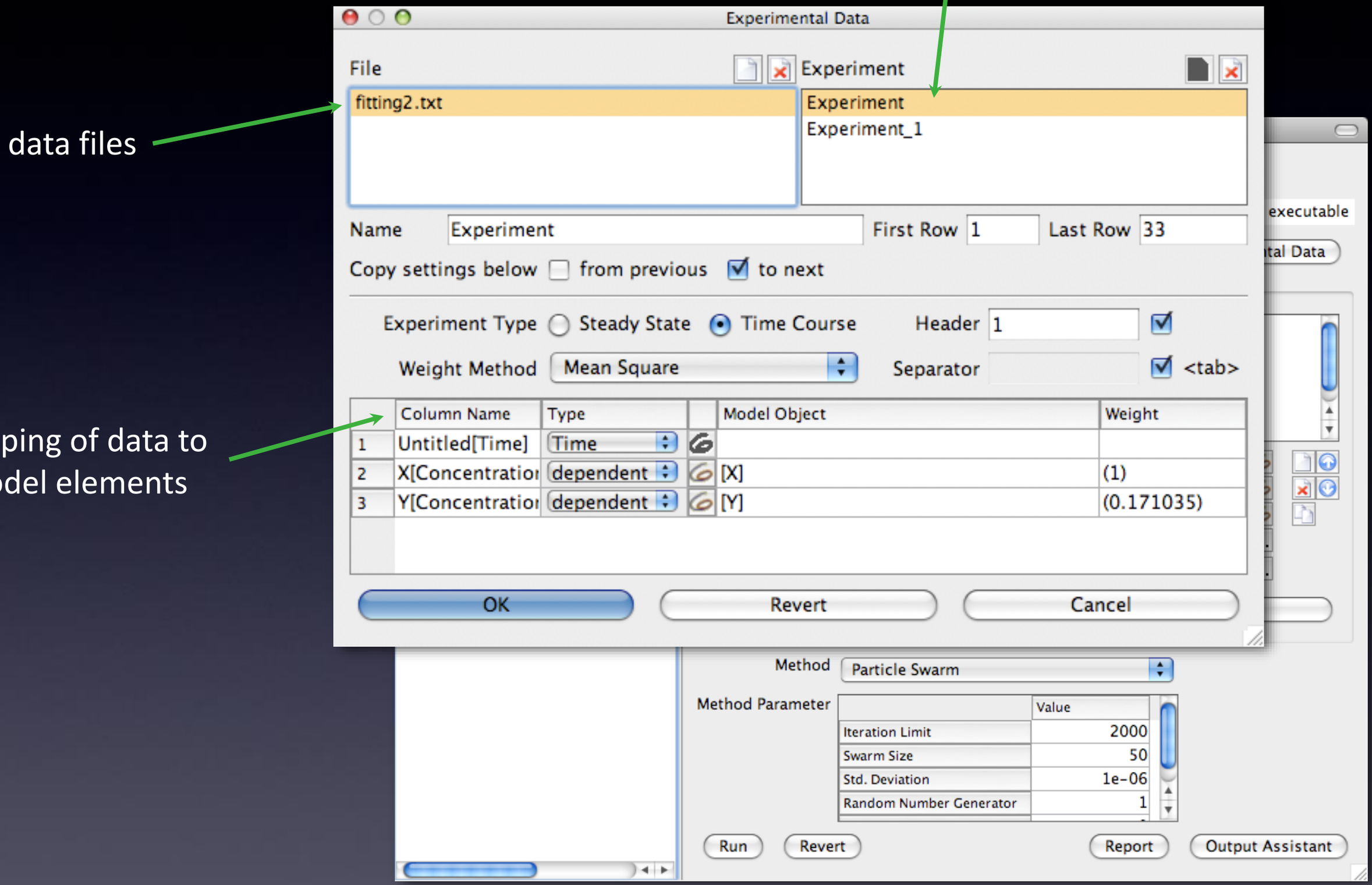
choose optimization algorithm



possibly several
experiments per file

data files

mapping of data to
model elements



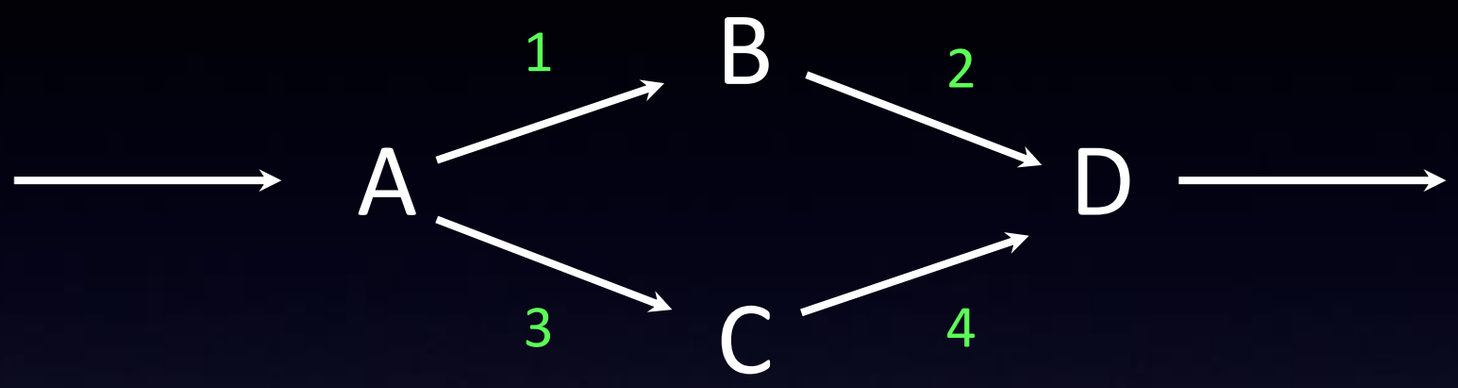
Important:

- The result of a parameter fitting always needs to be inspected afterwards!
- Having a good result for a fit does **not** mean that the parameter value is the „true“ one. This depends on the assumptions about the errors and the correctness of the model.
- For the stochastic algorithms the result is not reproducible!

Identifyability

- It is not obvious whether a certain parameter can be determined from a given set of measurements. -> concept of identifyability
- structural nonidentifyability
- practical nonidentifyability

structural nonidentifyability



- If we have measurements only for A and D, we cannot determine parameters for reactions 1, 2, 3, 4.
- This is not just an inaccuracy. Many combinations of parameters give exactly the same goodness of fit. We have to redesign the model or the experiments.

practical nonidentifyability

- If we only measure C we can in principle determine the properties of both reactions.
- Practically this is only possible with unrealistically accurate measurements. If we have realistic measurement errors we will get very inaccurate results for the parameters of the fast reaction.

Using several experiments for parameter estimation

- The more data available, the better.
  - So if you have data from several experiments it should be used for parameter estimation simultaneously
- COPASI can deal with an arbitrary number of experiments, also of different kinds (combined time course/steady state, different variables, different time points, etc.)

several experiments...

- Adding data from several experiments is straight forward in COPASI. Several data files can be specified and each can contain several experiments
- Important information: What is the same for all experiments and what is different between experiments? For the things that are different, are they known or unknown?

several experiments...

- Simplest case: Repeated experiments.
  - nothing special needs to be done in COPASI
- Several experiments under different conditions. The conditions are known.
  - Example: Different stimulations in several experiments.
  - In COPASI: The stimulation needs to be a parameter in the model. In the experimental data specification this value is selected as an *independent* parameter. *Independent* data is known data that is provided in the data file. *Dependent* data is data that is used for fitting.

several experiments...

- Experiments where some conditions are different, but not known
- Example: *In vivo* experiments, even if the experiment is repeated with the same preparation, the initial conditions (inside the organism) are typically different.
- In COPASI: The user can specify that some parameters are fitted for all experiments, and some are fitted for a specific subset of experiments.