When starting a modeling project usually many parameters of the model are not known.

How can I find out about parameter values?

- **Experimental approach**: try to design an experiment for measuring the specific parameter
  - Typically *in vitro* experiment (e.g. for rate constants: put different amounts of substrate in a test tube and measure how fast the reaction proceeds)
  - Problems: often not possible, to many parameters

- **Systems biology approach**: adapt a complete model to experimental data
Parameter estimation : basic idea of parameter fitting

Change the parameter values of a model in order to that it best fits the experimental data
Parameter estimation : basic idea of parameter fitting

How to determine the “best” fit for a given set of experimental data?

We will use heuristics

- Computation of the probability that the measurements (experimental data) would be the results of a simulation of the model

- A high probability means that the model is good

- Criterion: likelihood is maximal when the difference between measurement and simulation results is minimal (easy to calculate)
Parameter estimation

The parameters inference problem for ordinary differential equation models is usually formulated as an optimization problem with an objective function that has to be minimized by adjusting the values of the model parameters. A common choice to compute this objective function is to calculate the sum of squared errors between measurements and model predictions: the least square distance measure

$$D(p) = \sum_{i=1}^{N} (x_i - y_i(p))^2$$

A compound is tracked over time and we obtained N values. Thus N is the number of data points, $x_i$ is the measured value for time $t_i$ and $y_i(p)$ is the simulated value for time $t_i$ for a set of parameter values $p$. 
The objective function may have many local minima. If we have $N$ parameters to estimate, the parameters represent an $N$-dimensional space, the parameter space. A specific solution (specific parameter values) will correspond to a point in the parameter space. We need a way to find the set of parameter values (a point in parameter space) for which the distance $D$ is minimal (the best fit). We will use an optimization algorithm since a systematic scan of the parameter space is not possible when the space dimension is large.
Parameter estimation: numerical optimization cycle

1. **Initial guess for parameters set \( p \)**
2. **Determine new parameter values \( p \)**
   - Optimization algorithm
3. **Do simulation and evaluate objective function \( D(p) \)**
4. **good enough?**
   - yes: solution
   - no: return to Determine new parameter values

**Objective function**: function that needs to be minimized → sum of squared errors between measurements and model predictions
Parameter estimation : optimization algorithm

Different methods have been proposed:
Local optimization methods tend to converge quickly but have a tendency to get stuck in local optima
Global methods might take time but ensure the global optimum

Focus on the particle swarm optimization (PSO) method since its performance compared to the fourth most popular optimization methods (Evolutionary Computation, Evolutionary Programming, Genetic Algorithms and Simulated Annealing) reveals that the PSO method performs the best in systems biology (Baker et al, 2010, Journal of Integrative Bioinformatics, 7(3):133).

PSO belongs to the class of stochastic global optimization methods which depend on probabilistic approaches.

The swarm is typically modeled by particles that have a position and a velocity in multidimensional space. These particles roam through the hyperspace and have two essential reasoning capabilities: A particle is described by a position vector in the parameter space and a velocity vector. The velocity of a particle represents the direction of its parameter space exploration and the speed of the movement.

Each particle possesses:

- the memory of its own previous experience and remembers its best achievement ($pbest$) Information about the best solution attained within its neighbors. The “global” version of the optimizer keeps track of the overall best value, and its location, obtained thus far by any particle in the population ($gbest$). In the “local” version, each particle keeps track of the best solution attained within a local topological neighborhood of particles ($lbest$)
Terminology

- Particles
- Velocities
- Personal best
- Global best

Particles: $x_i$
Velocities: $v_i$
Personal best positions ever: $p_i$
Global best position ever: $g_{best}$
The PSO: Concept

Each particle of the swarm is randomly initialized for its position and velocity. These particles roam through the parameter space and optimization concept consists of, at each time step, changing the velocity (accelerating) of each particle toward its $p_{best}$ and $g_{best}$ or $l_{best}$.

Each particle modifies its position according to:
• its current position
• its current velocity
• the distance between its current position and $p_{best}$
• the distance between its current position and $g_{best}$
PSO: algorithm

Let $S$ be the number of particles in the swarm. Each particle $i$ has a position $x_i \in \mathbb{R}^n$ in the search-space and a velocity $v_i \in \mathbb{R}^n$.

Let $p_i$ the best known position of particle $i$ and $n_i$ the best known position of its neighbor.

Let $f$ the objective function which must be minimized.

**Initialization**

for each particle $i = 1, ..., S$ do

  Initialize randomly the particle position vector $x_i$
  Initialize the particle’s best position $p_i$ with $x_i (p_i \leftarrow x_i)$
  If $f(p_i) < f(n_i)$ then
    Update the neighbor’s best position $n_i \leftarrow p_i$
  Initialize the particle velocity
While a termination criterion is not reached do
for each particle \( i = 1, \ldots, S \) do
  Update the particle velocity as follow:
  \[ v_i = w \cdot v_i + c_1 \cdot r_1 \cdot (p_i - x_i) + c_2 \cdot r_2 \cdot (n_i - x_i) \]
  Update the particle position:
  \[ x_i = x_i - v_i \]
  If \( f(x_i) < f(p_i) \) then
    Update the particle best known position \( p_i \leftarrow x_i \)
    If \( f(p_i) < f(n_i) \) then
      Update the neighbor’s best position \( n_i \leftarrow p_i \)

**termination criterion:**
the specified number of iteration or the value of the objective function is < than a given threshold and a solution has been found
The PSO: algorithm

\[ \mathbf{v}_i = \mathbf{w} \mathbf{v}_i + c_1 r_1 (\mathbf{p}_i - \mathbf{x}_i) + c_2 r_2 (\mathbf{n}_i - \mathbf{x}_i) \]

- **Inertia**
  - Makes the particle move in the same direction and with the same velocity

- **Personal Influence**
  - Improves the individual
  - Makes the particle return to a previous position, better than the current
  - Conservative

- **Social Influence**
  - Makes the particle follow the best neighbors direction

✓ **Intensification**: explores the previous solutions, finds the best solution of a given region

✓ **Diversification**: searches new solutions, finds the regions with potentially the best solutions
The PSO: algorithm
Example
The PSO: algorithm
Example

Image description:
- The PSO algorithm is visualized in a 2D search space defined by axes X and Y.
- The search space contains multiple fitness landscapes, with different areas of interest highlighted.
- Particles (represented by black dots) move within the search space, adjusting their positions based on their personal best and the best position found by other particles.
- The fitness landscapes are color-coded to indicate the fitness value, with darker shades representing higher fitness values.
- The diagram illustrates the dynamic movement of particles as they explore the search space, aiming to find the optimal solution.
The PSO: algorithm
Example
The PSO: algorithm
Example
The PSO: algorithm
Example
The PSO: algorithm
Example

search space
The PSO: algorithm
Example
The PSO: algorithm
Example
Parameter estimation: particle swarm optimization method

Here, a particle will correspond to the set of kinetic parameters that have to be estimated and a best achievement corresponds to the smallest value obtained for the objective function.

Each iteration in PSO execution, requires for each particle of the swarm:

1. the values of the position vectors are set as the values of the model parameters
2. the model is simulated, by numerical integration of the ODE system, to produce the dynamic profiles corresponding to those parameter values
3. the simulated data are compared to the experimental data using the objective function described above

The iterative process will stop if the change of the objective function value is smaller than a specified value or if the number of specified iterations is reached.
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.
Parameter estimation in COPASI

Choose optimization algorithm

Select parameter estimation

List of unknown parameters (including ranges)

Choose experimental data (possible several experiments)

Model

Parameter Estimation

Object
Lower Bound
Upper Bound
Start Value
Affected Experiments
Method
Method Parameter

Run
Revert
Report
Output Assistant

Concentrations
**Experimental data**

**Example of experimental time-series data:** transcriptional fusion of a fluorescence (GFP) or luminescence (luciferase) reporter gene to the promoters of the target genes

Measurement techniques allow real-time and in-vivo monitoring of gene expression
The quantity of luminescence per cell as a function of time ($r(t)$) is:

$$r(t) = \frac{I(t)}{A(t)}$$

with

- $I(t)$ is the luminescence intensity (in RLU)
- $A(t)$ the absorbance values corrected by subtraction of the absorbance background measured on wells containing only growth medium

Since the luciferase does not require any post-translational modification such as folding, this ratio estimates the average concentration of reporter protein per cell. The dynamics of the system is conveniently described by the temporal evolution of the luciferase concentration.

**Normalization of luminescence data**

![Normalized by OD value](image)
Use of the green fluorescent reporter gene

• Fluorescent activity of GFP in response to light excitation depends on post-translational modifications, notably the folding of the protein to an appropriate conformation, including the autocatalytic formation of the chromophore.
• This maturation process gives rise to an additional reaction step from GFP to active GFP.
• See de Jong et al. BMC Systems Biology 2010, 4:55 for correct normalization of fluorescence signal
Promoter activities deduced from luminescence signal

Promoter activities are deduced using its derivative according to the following formula (Stefan et al, 2015, *PLoS Comput. Biol.* 11 e1004028) to take into account the effects of dilution and luciferase degradation:

\[
f(t) = \frac{d}{dt} r(t) + (\gamma_r + \mu(t)) r(t) = \frac{d(I(t))}{A(t)} + \frac{\gamma_r I(t)}{A(t)}
\]

With the growth rate \(\mu(t)\) estimated from the absorbance as follow:

\[
\mu(t) = \frac{d}{dt} \frac{A(t)}{A(t)} = \frac{1}{A(t)}
\]

where \(\gamma_r\) [min^{-1}] is the degradation constant of the luciferase protein and \(\mu(t)\) [min^{-1}] is the growth rate of the bacteria.

![Graphs showing normalized luminescence signals](image)
**protein concentration kinetics**

Computation of the concentration evolution of the protein of interest over time:

→ Correction to take into account the differences in half-lives between the reporter luciferase and the protein whose gene activity is measured (Stefan et al, 2015, PLoS Comput. Biol. 11 e1004028):

\[
\frac{d}{dt} p(t) = f(t) - (\gamma_p + \mu(t)) p(t), \quad p(0) = p_0 = \frac{\mu(T) + \gamma_r}{\mu(T) + \gamma_p} r(T)
\]

where \( \gamma_p \) [min\(^{-1}\)] is the degradation constant of the protein and \( \mu(T) \) is the growth rate of bacteria at the end of the preculture procedure (at time T). \( p(T) \) and \( r(T) \) are the corresponding concentrations of the protein of interest and reporter protein, respectively. Usually, the bacteria in the preculture are in stationary phase, so \( \mu(T) = 0 \)

Correction effects:
- protein of interest with a shorter life time (here 8 min) than luciferase (21.6 min): the uncorrected values clearly overestimate the protein concentration (red curves)
- protein of interest with a higher life time (here 80 min) than luciferase (21.6 min): the uncorrected values underestimate the protein concentration (blue curves)

- Light color lines protein concentration kinetics taking into account only protein life time
- Dark color lines protein kinetics after correction