Parameter estimation for biological systems

Youcef Akrout
ENS Ulm

October 2020

Abstract

Modeling of biochemical reaction networks can be based on several mathematical tools. Differential equation is an important one. It improved our understanding of biological processes. As we describe more and more species or we consider complex geometries, numerical complexity becomes a critical question. Individual simulations may be feasible. But, the inference of model parameters from experimental data is more intensive.

In our work, we present some approaches of the parameter estimation problem. In particular, we'll focus on sensitivity based methods. Several papers already proved their efficiency for ODE. The aim of the project is to generalize it for PDE. We'll then test it for a specific Turing mechanism, which models lung branching morphogenesis.

Contents

1 Introduction and motivation

In computational biology, the size of experimental data increased, thanks to the efficiency of new measurement machines. Large datasets can be obtained at a reasonable cost and allow us understand more complex mechanisms. However, we are now limited by the scalability of currently available computational methods, especially for parameter estimation questions.

Plenty of softwares can do simulation and statistical inference, for ODE systems. It is more difficult for PDEs or for models with large number of parameters. We can of course determine maximum likelihood (ML) and maximum a posteriori (MAP) estimates. But high-dimensional nonlinear and non-convex optimization problems need to be solved, which is challenging.

Multi-start local optimization methods have proven to be successful. Using the gradient of the objective function allows fast local convergence. However, the gradient computation is often a computationally demanding step. Approximating it by finite differences is neither numerically robust nor computationally efficient. We'll present two solutions, based on the sensitivity analysis. It allows to decrease the numerical error and computation time in the ODE setup [1, 2].

The aim of the project is to generalize one of the sensitivity related methods, the forward approach, for PDE. We'll then test it for a specific Turing mechanism, which models lung branching morphogenesis. This complex process tells how the lung develops during embryonic stages in a living organism. More precisely, we want to fit a dataset, containing developmental series of 3D embryonic lungs. A two-component Turing model [3] should successfully predicts the embryonic areas of outgrowth, if we estimate well the parameters.

2 Approach for ODE

2.1 Parameter estimation framework

A biochemical reaction network is modeled as an ODE system.

$$\begin{cases} \dot{x} & = & f(x,\theta) \in \mathbb{R}^{n_x} \\ x(t_0) & = & x_0(\theta). \end{cases} \tag{1}$$

x denotes the concentration vector and $\theta \in \mathbb{R}^{n_\theta}$ represents the parameters (e.g. kinetic constants). The vector field f describes the evolution of the species concentration. The mapping x_0 provides the parameter dependent initial condition at time t_0.

Usually we don't have access to the concentration of all species. We need to consider an output map $h(x,\theta) \in \mathbb{R}^{n_y}$, modeling the measurement process. We define the output y, depending on the state variables and the parameters,

$$y(t,\theta) = h(x(t,\theta),\theta) \in \mathbb{R}^{n_y}. \tag{2}$$

We consider discrete-time, noise corrupted measurements

$$\overline{y}_{ij} = y_i(t_j, \theta) + \varepsilon_{ij}, \quad \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_{ij}^2). \tag{3}$$

We can now build our dataset $D = \{(t_j, (\overline{y}_{ij})_i), j \in [\![1, N]\!]\}$. N denotes the number of time points.

We want to estimate the unknown parameter θ from the experimental data D using ML estimation. We focus on the case of independent and normally distributed measurement noise with known variances. Writing the negative log-likelihood yields to the following objective function, indicating the difference between experiment and simulation.

$$\varphi(\theta) = \frac{1}{2} \sum_{i=1}^{n_y} \sum_{j=1}^{N} \left(\frac{\overline{y}_{ij} - y_i(t_j, \theta)}{\sigma_{ij}} \right)^2. \tag{4}$$

The minimization

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}}\ \varphi(\theta) \tag{5}$$

of this weighted least squares φ yields the ML estimate of the parameters.

The optimization problem 5 is in general non-convex. Thus, the objective function can possess multiple local minima and global optimization strategies need to be used. For ODE models multi-start local optimization has been shown to perform well.

Local optimization is performed in our case using the Levenberg Marquardt method implemented in the MATLAB function **lqscurvefit**. One can approximate the gradient with the techniques presented below.

2.2 Gradient approximation

Finite differences

It is a naive approximation of the function gradient.

$$\frac{\partial \varphi}{\partial \theta_k} \approx \frac{\varphi(\theta + a e_k) - \varphi(\theta - b e_k)}{a + b} \tag{6}$$

with $a, b \geq 0$ and the k-th unit vector e_k. In practice, forward differences ($a = \varepsilon, b = 0$), backward differences ($a = 0, b = \varepsilon$) and central differences ($a = \varepsilon, b = \varepsilon$) are widely used.

In theory, forward and backward differences provide approximations of order ε while central differences provide more accurate approximations of order ε^2, provided that φ is sufficiently smooth. A good choice of a and b is important, to get a good balance for the approximation and the integration accuracy.

The computational complexity of evaluating gradients using finite differences is linear in the number of parameters. A single simulation of a large-scale model is already time-consuming. So finite differences is limiting.

Forward sensitivity analysis

We introduce the sensitivities, which represent the evolution of the solution when we perturbate the parameters. Let $i \in [\![1, n_y]\!]$ and $k \in [\![1, n_\theta]\!]$. We denote

$$s^y_{i,k} : t \in \mathcal{T} \longmapsto \frac{\partial y_i}{\partial \theta_k}. \tag{7}$$

Then, the gradient of the objective function is

$$\frac{\partial \varphi}{\partial \theta_k} = \sum_{i=1}^{n_y} \sum_{j=1}^{N} s^y_{i,k}(t_j) \left(\frac{y_i(t_j, \theta) - \overline{y}_{ij}}{\sigma^2_{ij}} \right) \tag{8}$$

Governing equations for the sensitivities are obtained by differentiating the ODE system and the observation function definition. More precisely,

- We differentiate equation 2, using the composition formula. We get

$$s^y_{i,k} = \frac{\partial h_i}{\partial x} s^x_k + \frac{\partial h_i}{\partial \theta_k} \tag{9}$$

  with $s^x_k(t) = \frac{\partial x}{\partial \theta_k} \in \mathbb{R}^{n_x}$ for $t \in \mathcal{T}$. It denotes the sensitivity of the state x wrt θ_k.

- We differentiate equation 1. Then, we reorder the derivatives. We get

$$\begin{cases} \dot{s}^x_k &= \frac{\partial f}{\partial x} s^k_x + \frac{\partial f}{\partial \theta_k} \\ s^x_k(t_0) &= \frac{\partial x_0}{\partial \theta_k} \end{cases} \tag{10}$$

In practice, we often combine the computation of the output and the sensitivities to improve the computational efficiency.

Forward sensitivity analysis is usually faster than finite differences. But complexity stills increase roughly linearly with n_θ, which is challenging for large-scale systems.

Adjoint sensitivity analysis

In contrast to forward sensitivity analysis, adjoint sensitivity analysis does not rely on the state sensitivities $s^x_k(t)$ but on the adjoint state $p(t) \in \mathbb{R}^{n_x}$, defined as the following.

- For $t > t_N$, the adjoint state is zero, $p(t) = 0$.

- Starting from this end value the trajectory of the adjoint state is calculated backwards in time, from the last measurement $t = t_N$ to the initial time $t = t_0$. At the time points at which measurements have been collected, $t_N \dots t_1$, the adjoint state is reinitialised as

$$p(t_j) = \lim_{t \to t_j^+} p(t) + \sum_{i=1}^{n_y} \left(\frac{\partial h_i}{\partial x} \right)^\top \frac{\overline{y}_{ij} - y_i(t_j)}{\sigma^2_{ij}}, \tag{11}$$

  which usually results in equation 11 a discontinuity of $p(t)$ at t_j.

- Starting from the end value $p(t_j)$ as defined in equation 11 the adjoint state evolves backwards in time until the next measurement point t_{j-1} or the initial time t_0 is reached. This evolution is governed by the time-dependent linear ODE

$$\dot{p} = -\left(\frac{\partial f}{\partial x}\right)^\top p. \tag{12}$$

Based on the integration by part of this equation

$$\int_{t_j}^{t_{j+1}} \left(\dot{p}(t) + \left(\frac{\partial f}{\partial x}\right)^\top p(t)\right)^\top s_k^x(t)dt = 0, \tag{13}$$

the gradient of φ can be computed

$$\frac{\partial \varphi}{\partial \theta_k} = -\int_{t_0}^{t_N} p^\top \frac{\partial f}{\partial \theta_k} dt - \sum_{i,j} \frac{\partial h_i}{\partial \theta_k} \frac{\overline{y}_{ij} - y_i(t_j)}{\sigma_{ij}^2} - p(t_0)^\top \frac{\partial x_0}{\partial \theta_k}. \tag{14}$$

The key difference is that the sensitivity of the objective function is directly computed, without the derivatives of the state variables. So the computation time of the gradient is almost independent of the number of parameters.

2.3 Optimization algorithm

Our problem can be written as a *Nonlinear Least Squares Minimization*.

$$\varphi(\theta) = \frac{1}{2}\sum_{l=1}^{m} r_l^2(\theta) = \frac{1}{2}\|r(\theta)\|^2 \tag{15}$$

with r the residual vector $r(\theta) = (r_1(\theta), \ldots, r_m(\theta))$.

The derivatives of φ can be written using the Jacobian matrix J of r defined as

$$J(\theta) = \left[\frac{\partial r_l}{\partial \theta_k}\right]_{l,k}. \tag{16}$$

Indeed, we have

$$\left\{\begin{array}{lcl} \nabla\varphi(\theta) & = & J(\theta)^\top r(\theta) \\ \nabla^2\varphi(\theta) & = & J(\theta)^\top J(\theta) + \sum_{l=1}^{m} r_l(\theta)\nabla^2 r_l(\theta) \end{array}\right. \tag{17}$$

If r_l are close to linear functions, the Hessian can be approximated

$$\nabla^2\varphi(\theta) = J(\theta)^\top J(\theta). \tag{18}$$

Classic approaches

To minimize φ, vanilla gradient descent

$$\theta_{i+1} = \theta_i - \lambda \nabla \varphi(\theta_i) \tag{19}$$

is the simplest technique. But, it suffers from various convergence problems. The step size is fixed and we don't take into account the function curvature, given in the second derivatives.

Newton's method fixes this issue. By assuming φ to be locallay quadratic, we get the following update rule

$$\theta_{i+1} = \theta_i - (\nabla^2 \varphi(\theta_i))^{-1} \nabla \varphi(\theta_i). \tag{20}$$

Hessian doesn't need to be evaluated exactly, rather the approximation 18 can be used. The main advantage of this technique is rapid convergence. However, the rate of convergence is sensitive to the starting location (or more precisely, the linearity around the starting location).

Levenberg-Marquardt algorithm [4, 5]

Simple gradient descent and Gauss-Newton iteration are complementary in their advantages. Levenberg proposed an update rule, based on that observation, given as

$$\theta_{i+1} = \theta_i - (H + \lambda I)^{-1} \nabla \varphi(\theta_i) \tag{21}$$

with $H = \nabla^2 \varphi(\theta_i)$. If the error goes down following an update, it implies that our quadratic assumption on $\varphi(\theta)$ is working and we reduce λ (usually by a factor of 10) to reduce the influence of gradient descent. On the other hand, if the error goes up, we would like to follow the gradient more and so λ is increased by the same factor.

The above algorithm has the disadvantage that if the λ is large, the Hessian isn't used. We can derive some advantage out of the H even in such cases by scaling each component of the gradient according to the curvature. This crucial insight was provided by Marquardt, resulting in the LM update rule.

$$\theta_{i+1} = \theta_i - (H + \lambda \operatorname{diag}[H])^{-1} \nabla \varphi(\theta_i) \tag{22}$$

Since the Hessian is proportional to the curvature of φ, 22 implies a large step in the direction with low curvature (i.e. an almost flat terrain) and a small step in the direction with high curvature (i.e a steep incline).

Even if LM method is just a heuristic, it works extremely well. The only flaw is its need for matrix inversion as part of the update. But using clever pseudo-inverse methods, LM is much faster than vanilla gradient descent, for moderately sized models (of a few hundred parameters).

2.4 Numerical results

To test the methods, we'll use the following ODE equations (simplification of the PDE 31 presented below).

$$\begin{cases} \dot{R} &= \gamma(a + R^2 L - R) \\ \dot{L} &= \gamma(b - R^2 L) \end{cases} \tag{23}$$

with $R, L : \mathcal{T} = [0, 10] \longrightarrow \mathbb{R}$. The problem 23 is smooth and we have, thanks to Cauchy - Lispschitz theorem, the guarantee of the solution existence and uniqueness.

We apply an observation function $h : (R, L) \mapsto R^2 L$ to the state variables. The objective is to optimize the parameters $\theta = \left[a, b, \gamma\right]^\top$, by evaluating the difference between $y = h(X)$ with synthetic data, generated with $\theta_\infty = [0.1, 1.5, 2]$, sampled for $N = 101$ time points.

As we focus on systems with few parameters, we compare only two approximations – finite difference and forward sensitivity – with the the LM optimization.

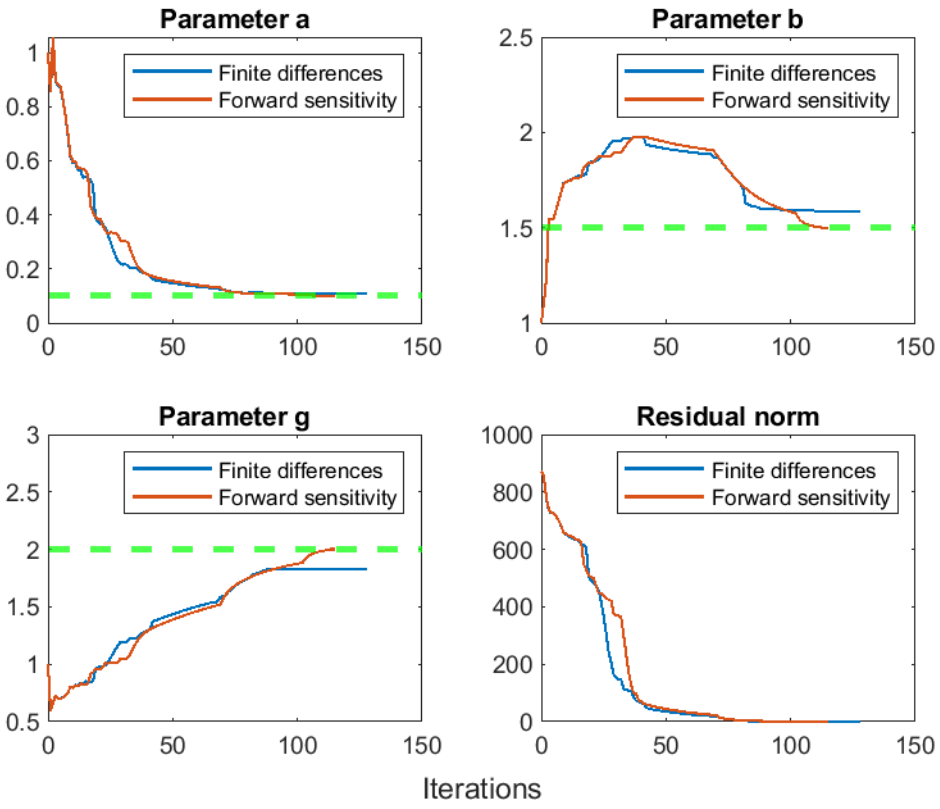


Figure 1: Finite differences vs forward sensitivity analysis for system 23.

With forward sensitivity, the number of iterations is lower, and final residual norm is much smaller : 10^{-5} vs 1.3. This important difference is due to the estimation of parameters b and γ, trapped in a local minimum in the finite differences case.

3 Generalization for PDE

3.1 Turing mechanisms

It is a long-standing question how stereotyped patterns can emerge during development of complex organisms. Alan Turing proposed a simple reaction-diffusion based mechanism [6] that has since been shown to have the potential to give rise to a large variety of patterns.

A problem with its applicability in biology concerns the size of the parameter space that gives rise to Turing patterns, the Turing space. This parameter space is small. We present here the criteria for the emergence of patterns for systems with two species [7].

We consider systems of the form

$$\begin{cases} \dot{r} &= \gamma\ u(r,l) + \Delta r, \\ \dot{l} &= \gamma\ v(r,l) + d\ \Delta l, \end{cases} \tag{24}$$

defined on $\mathbb{R}_+ \times \Omega$ (with a given spatial domain $\Omega \subset \mathbb{R}^n$) subject to boundary and initial conditions, where the space- and time-dependent functions r and l represent concentrations and the reaction kinetic terms u and v are generally nonlinear functions.

To ensure the uniqueness of the solution, we endow the system 24 with initial and boundary conditions. We use typically homogeneous Neumann boundary

$$(n \cdot \nabla) \begin{bmatrix} r \\ l \end{bmatrix} = 0 \text{ on } \mathbb{R}_+ \times \partial\Omega$$

because they are easy to handle and have a biological interpretation (impermeable boundary), even if other boundary conditions wouldn't alter the following analysis.

A Turing instability appears when a reaction-diffusion system has a stable steady state in the absence of diffusion, which loses its stability in the presence of diffusion such that spatial patterns emerge.

Stability without diffusion

Let r_0, l_0 denote the steady state of the diffusion-free ODE system

$$\partial_t \begin{pmatrix} r \\ l \end{pmatrix} = \gamma\ f(r,l) \text{ with } f = \begin{pmatrix} u \\ v \end{pmatrix} \tag{25}$$

and linearize the system around (r_0, l_0) by introducing the translated function $w = \begin{pmatrix} r - r_0 \\ l - l_0 \end{pmatrix}$. Then, the linearized system becomes

$$\dot{w} = \gamma\ Jw \text{ with } J = \mathrm{Jac}_f(r_0, l_0).$$

The steady state of system 25 is linearly stable if all eigenvalues of J are negative, which for a two-component system is ensured by the following conditions. – the derivatives are evaluated at (r_0, l_0) –

$$\begin{cases} \mathrm{tr}\ J &= \partial_r u + \partial_l v &< 0 \\ \det J &= \partial_r u\ \partial_l v - \partial_l u\ \partial_r v &> 0 \end{cases} \tag{26}$$

Diffusion-driven instability

Now let us add diffusion to our ODE system of and consider the reaction-diffusion system linearized about the steady state $w = (0, 0)$, which has the form

$$\dot{w} = \gamma\ Jw + D\ \Delta w \text{ with } D = \text{diag}(1, d). \tag{27}$$

We look for a solution of the form

$$w(t, x) = \sum_k C_k e^{\lambda_k t} W_k(x) \tag{28}$$

where λ_k determine the temporal growth of the solution and the time-independent functions W_k are the solutions of the elliptic eigenvalue problem

$$\Delta W_k + k^2 W_k = 0, \quad (n \cdot \nabla) W_k = 0.$$

The constants C_k are the Fourier coefficients of the initial conditions.

Combining equations 27 with 28 and using the fact that the family of eigenvectors of the Laplace operator (W_k) forms a complete orthonormal system, we obtain a linearized system for each wave number k.

$$\forall k \in \mathbb{N}, \quad \dot{w} = \gamma\ Jw - k^2\ Dw \tag{29}$$

We set $M_k := \gamma J - k^2 D$. Since we look for unstable solutions, we require that, for some $k \neq 0$, M_k has an eigenvalue λ such that Re $\lambda > 0$.

Then, some technical calculations lead to the following necessary conditions, to get instability. – the derivatives are evaluated at (r_0, l_0) –

$$\begin{cases} d\ \partial_r u + \partial_l v > 0 \\ (d\ \partial_r u + \partial_l v)^2 - 4d\ (\partial_r u + \partial_l v - \partial_l u\ \partial_r v) > 0 \end{cases} \tag{30}$$

Therefore, we need inequalities 26 and 30 to obtain Turing patterns. But, it is possible that these conditions are satisfied but that no pattern emerges.

3.2 Lung branching morphogenesis

We'll focus on a specific Turing system, modeling the branching events during lung development. To understand it, we use a developmental series of 3D geometric datasets of mouse embryonic lungs published in [8].

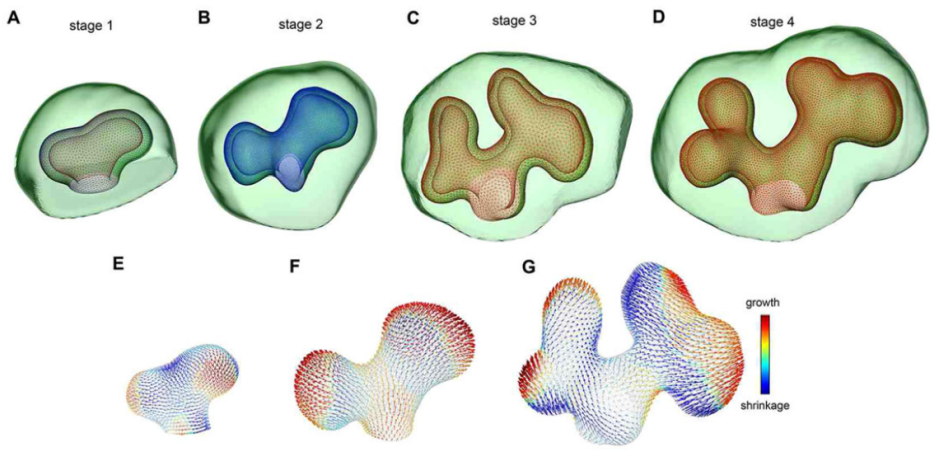


Figure 2: Embryonic growth fields in lung branching morphogenesis.
(A-D) 3D embryonic lung buds. (E-G) Calculated displacement fields of the epithelium.

To explain the data, [3] builds a model with two components : a ligand L and a receptor R. We assume that they interact cooperatively and ligand-receptor binding results in an increased emergence of receptor on the membrane. We obtain the standard Schnakenberg-type reaction kinetics for Turing patterns, which takes the form.

$$\begin{cases} \dot{R} &= \Delta R + \gamma(a + R^2 L - R) \\ \dot{L} &= D\Delta L + \gamma(b - R^2 L) \end{cases} \tag{31}$$

We want to match the embryonic growth fields and the predicted signaling domains. To quantify the difference, we compute the deviation Δ of the normalized concentration of the ligand-receptor complex from the normalized length of the embryonic growth field shown in Figure 2.

$$\Delta = \int_\Omega (C - E)^2 \tag{32}$$

Here, Ω represents the border (a surface in 3D), called epithelium, C and E are the computed and experimental variables accordingly. The normalization leads to the following.

$$C = \frac{R^2 L}{\int_\Omega (R^2 L)}, \quad E = \frac{\|\overrightarrow{v}\|}{\int_\Omega \|\overrightarrow{v}\|}. \tag{33}$$

Here, R is the concentration of the receptor, L is the concentration of the ligand. $\overrightarrow{v}$ is the vector field that describes the embryonic growth field (only the outward vectors).

For the PDE system 31, we'll mainly focus on the forward sensitivity approach. The analysis is relatively similar to the ODE case (based on derivatives exchange). The main difference is that we need to handle the diffusion parameter.

Let's rewrite the system 31 in a matrix form.

$$\dot{X} = \begin{bmatrix} 1 & 0 \\ 0 & D \end{bmatrix} \Delta X + f(X, \theta) \text{ with } f(X, \theta) = \begin{bmatrix} \gamma(a + R^2 L - R) \\ \gamma(b - R^2 L) \end{bmatrix} \tag{34}$$

The state variables are $X = \begin{bmatrix} R \\ L \end{bmatrix}$. We apply an observation function $h : (R, L) \mapsto R^2 L$ to the state variables. We write the new sensitivity equations for the state variables.

- **Kinetic parameters.** Let $\theta_k \in \{a, b, \gamma\}$. We have the following.

$$\dot{s}_k^X = \frac{\partial}{\partial \theta_k} \left[\frac{\partial X}{\partial t} \right] = \begin{bmatrix} 1 & 0 \\ 0 & D \end{bmatrix} \Delta s_k^X + \frac{\partial f}{\partial X} s_k^X + \frac{\partial f}{\partial \theta_k} \tag{35}$$

- **Diffusion parameter.** Here the equation is a bit more complex.

$$\dot{s}_D^X = \frac{\partial}{\partial D} \left[\frac{\partial X}{\partial t} \right] = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \Delta X + \begin{bmatrix} 1 & 0 \\ 0 & D \end{bmatrix} \Delta s_D^X + \frac{\partial f}{\partial X} s_D^X \tag{36}$$

The observation sensitivity can easily be deduced by derivation composition.

3.3 Numerical results

Toy example

To start, we focus on on the estimation of the kinetic parameters – a, b and γ –, in 1D. In practice, the diffusion parameter is fixed $D = 100$. We generate data by solving the PDE 31 for one set of parameter $\theta_\infty = [0.1, 1.5, 2]$, for $t \in \mathcal{T} = [0, 10]$, and $x \in \mathcal{D} = [0, 1]$. We keep only the image of the state variables through the observation function $(R, L) \longmapsto R^2 L$, for $N = 101$ time points and $M = 11$ space points, equally distributed on the space-time domain $\mathcal{D} \times \mathcal{T}$.

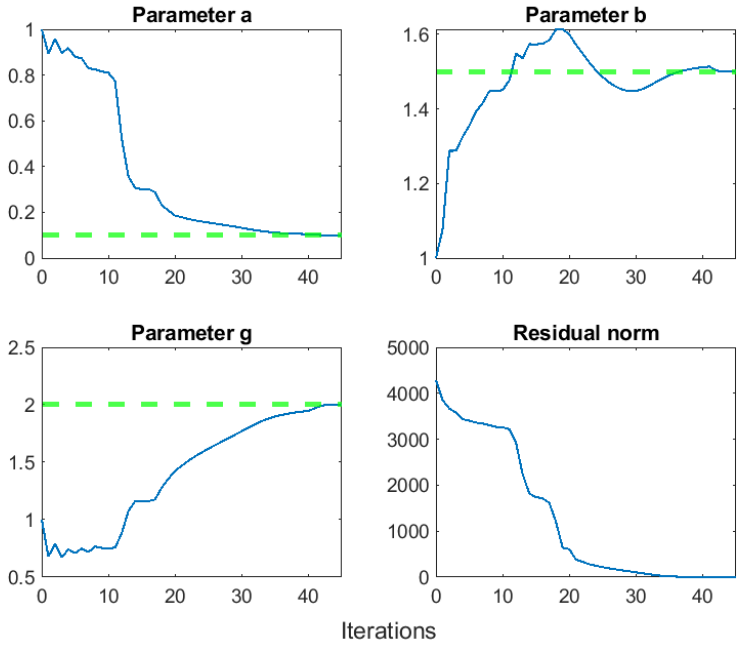


Figure 3: Kinetic parameters estimation in 31 - 1D setup.

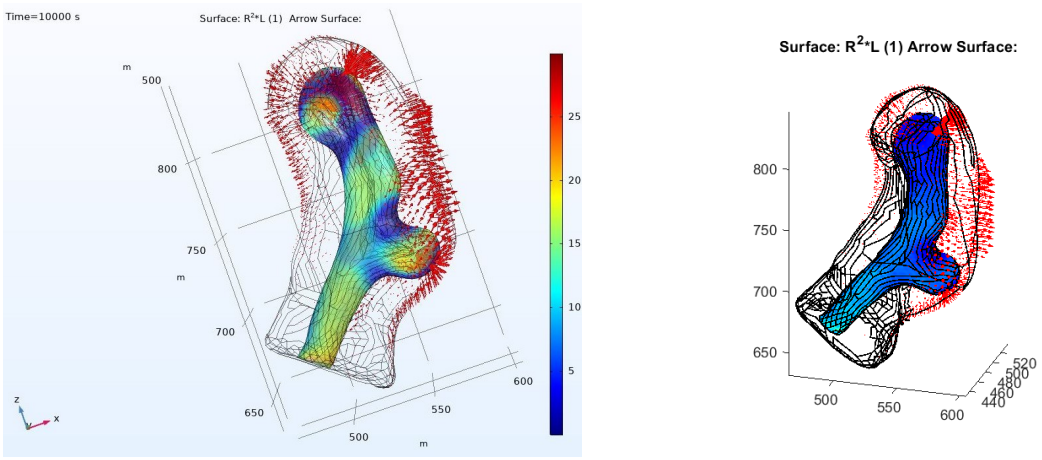
We tried also to estimate all the parameters including diffusion parameter. However, we didn't get relevant results. It is probably due to the structure of the PDE 31. Typically, if $|D\ \Delta L| \ll |\gamma(b - R^2 L)|$, diffusion has low impact and, hence it will be hard to estimate.

At least, we checked that the theory developed, to estimate diffusion, worked for a simple PDE $\dot{c} = D\ \Delta c - kc$.

Real-data fitting

We try to use this approach with our main PDE system 31, but in the more general setup (3D). We fix the diffusion parameter, $D = 100$, that is hard to estimate in this specific Turing mechanism. We want only to optimize kinetic parameters a, b and γ.

Some parameter screening is already done. So we have already a good initial point $\theta = [0.03, 0.25, 0.1]$ that gives interesting Turing patterns. The optimization process was long and tough. The PDE system is likely stiff. After few days of computation, it converged to $\theta' = [0.09, 0.85, 1.5]$, which doesn't belong to the Turing space, as the figure 4 shows it.



(a) Initial point $\theta = [0.03, 0.25, 0.1]$ (b) Final point $\theta' = [0.09, 0.85, 1.5]$

Figure 4: Optimization process, using Comsol Software.
The final parameters don't belong to the Turing space, as the spatial pattern is lost.

Even if the method developed is promising and works well for toy examples. It failed, to provide better parameters then a brute force approach. Real-data fitting is definitively a more challenging problem.

4 Conclusion and perspectives

Mechanistic modeling is an important step towards the understanding of biological processes. To make it accurate, we may need to increase significantly the number of variables or to switch in a PDE setting. In this project, we present a method that should render parameter estimation significantly more efficient in complex settings. Sensitivity analysis, which is extensively used in other research fields, is a powerful tool for estimating parameters, compared to finite differences.

We generalized the forward sensitivity method for PDE. We ran some experiments for equations with few parameters, in 1D. These were globally successful. We tried also to do real-data fitting. [3] showed that a ligand-receptor-based Turing mechanism allows us to predict the growth fields of developing lungs buds. We wanted to go further in the analysis, by proposing a more robust way to estimate the parameters. Unfortunately, it was more challenging than expected, due to the complexity of the Comsol software and the lack of documentation, about this specific question.

However, the method is really promising for PDE setup and there is still a lot to do. Indeed, the Levenberg-Marquardt algorithm is an efficient optimization framework for parameter estimation. But it is a local approach. If we initialize far from the optimum, the method is slowed down and computations become easily expensive, even for basic models. So, the idea can be to combine it efficiently with naive parameter screening, to have a better intuition of the good range of parameters. This potential hybrid method is significantly more robust, and can be generalized for any other kind of biological models based on differential equations.

References

[1] Geier F, Fengos G, Felizzi F, Iber D. Analyzing and constraining signaling networks: parameter estimation for the user. In: Computational modeling of signaling networks. Springer; 2012. p. 23–39.

[2] Fröhlich F, Kaltenbacher B, Theis FJ, Hasenauer J. Scalable parameter estimation for genome-scale biochemical reaction networks. PLoS computational biology. 2017;13(1).

[3] Menshykau D, Blanc P, Unal E, Sapin V, Iber D. An interplay of geometry and signaling enables robust lung branching morphogenesis. Development. 2014;141(23):4526–4536.

[4] Ranganathan A. The levenberg-marquardt algorithm. Tutoral on LM algorithm. 2004;11(1):101–110.

[5] Berghen FV. Levenberg-Marquardt algorithms vs trust region algorithms. IRIDIA, Université Libre de Bruxelles. 2004;1.

[6] Turing AM. The chemical basis of morphogenesis. Bulletin of mathematical biology. 1990;52(1-2):153–197.

[7] Kurics T, Menshykau D, Iber D. Feedback, receptor clustering, and receptor restriction to single cells yield large Turing spaces for ligand-receptor-based Turing models. Physical Review E. 2014;90(2):022716.

[8] Blanc P, Coste K, Pouchin P, Azais JM, Blanchon L, Gallot D, et al. A role for mesenchyme dynamics in mouse lung branching morphogenesis. PLoS One. 2012;7(7):e41643.