Optimization

Lecture 9: Parameter estimation

Parin Chaipunya
KMUTT
└Mathematics @ Faculty of Science
└The Joint Graduate School of Energy and Environments

Areas of research:
- Multi-agent optimization: Bilevel programs, Game theory
- Optimization modeling: mainly focused on energy and environmental applications

parin.cha@kmutt.ac.th
parinchaipunya.com
github.com/parinchaipunya

Last update: Januaray 2026

Table of contents

Section 1

Parameter estimation

Parameter estimation

Why parameters appear in a model?

- In formulating a mathematical model, one typically begins with a general structure that can be adapted through parameters.
- These parameters capture the key characteristics of the system, such as rates or efficiencies.
- However, in practice, it is often difficult to determine appropriate parameter values that accurately reflect real-world behavior.

Parameter estimation refers to the process of determining the appropriate values to the unknown parameters in a model **from an observed data**.

Estimating these parameters from data makes the model quantitatively accurate, typically by solving an optimization problem that minimizes the discrepancy between model predictions and observations.

Subsection 1

An illustrative example

SIR model

Let us give an example by using the SIR model from mathematical epidemiology. In this model, a population N is divided into susceptible (S), infectious (I), and recovered (R) groups, together with the dynamical system

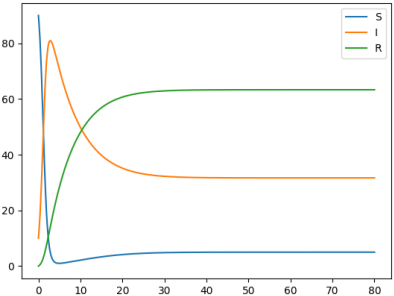
$$S' = -\beta SI \tag{1a}$$
$$I' = \beta SI - \gamma I \tag{1b}$$
$$R' = \gamma I\,. \tag{1c}$$

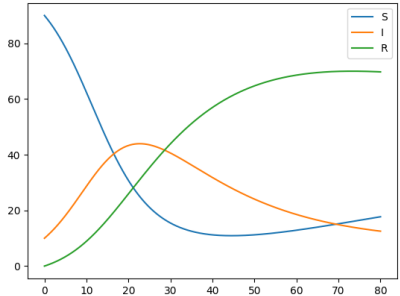
Here, the parameters are the transmission rate ($\beta > 0$) and the recovery rate ($\gamma > 0$).

With different possibilities of these parameter values, the SIR model (1a)–(1c) can be used to describe the infection of several different diseases.

SIR model

The following graphs are the SIR models of different parameter values.

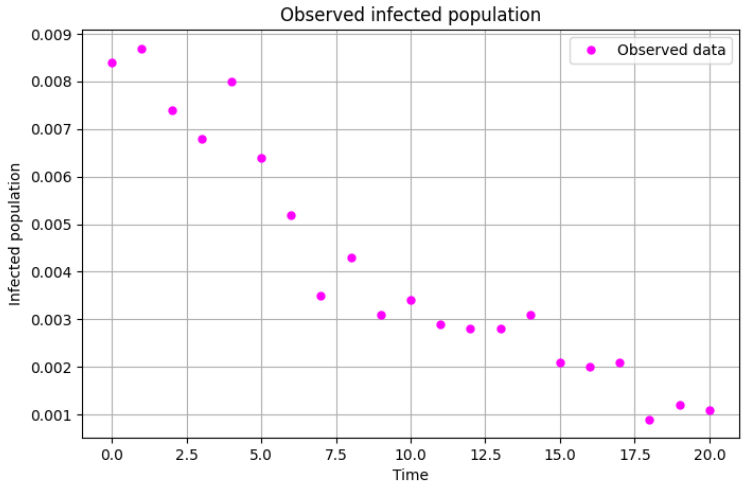




One should notice that the parameters play an important role to the behavior of a model. Hence, **parameter estimation** is an important step to accurately produce a simulation or a prediction from a model.

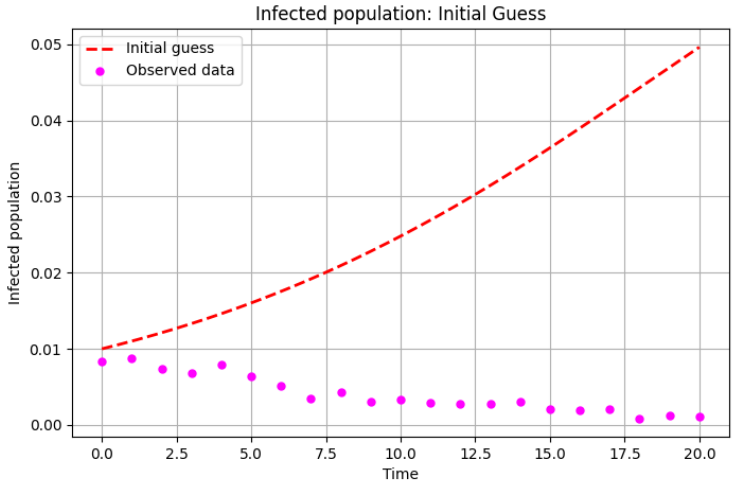
SIR model — Parameter estimation

We start with an observed data.



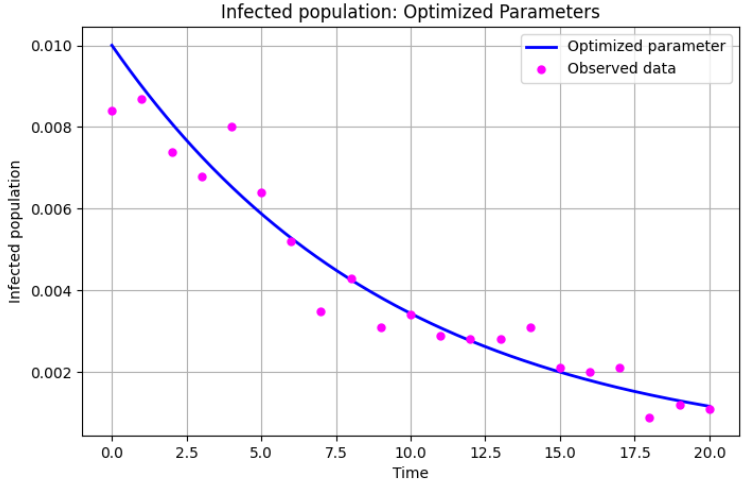
SIR model — Parameter estimation

Then we make an initial guess (not necessarily good).



SIR model — Parameter estimation

After a proper parameter estimation, an **optimized parameter** is obtained.



Subsection 2

Optimization problem

Discrepancy function

- Suppose that we have a set of observed data $\{(t_d, y_d)\}_{d=1,\dots,D}$.
- The model $\mathcal{M}$ contains a vector $\theta \in \mathbb{R}^N$ of parameters.
- At a fixed parameter θ, the model has a prediction function $\mathcal{P}_\theta$ that predicts

$$\hat{y}_d = \mathcal{P}_\theta(t_d) \qquad \forall d = 1, \dots, D\,.$$

- The **discrepancy function**, in this lecture, is defined as the total squared difference between the predictions and observations, that is

$$J(\theta) = \sum_{d=1}^{D} (\hat{y}_d - y_d)^2 = \sum_{d=1}^{D} (\mathcal{P}_\theta(t_d) - y_d)^2\,.$$

- Minimizing the discrepancy $J(\theta)$ at the first glance **looks like a least squares problem, but this is not true!**

The prediction function

The main difficulty lies in the fact that the predictions $\mathcal{P}_\theta(t_d)$ typically do not admit closed-form expressions.

For example, in the SIR model, the predicted number of infected individuals at each time point is obtained via a numerical integration scheme (e.g., the classical Runge–Kutta method). Consequently, **the mapping $\mathcal{P}_\theta$ cannot be expressed explicitly in analytic form**.

This lack of an explicit representation prevents the direct computation of ∇J, thereby necessitating the use of **derivative-free** or **inexact** or **heuristic methods**.

Section 2

Parameter estimation methods

Subsection 1

Common approaches

Trial and errors

This is the first natural method when there is no other choice available. This requires a **manual intervention** of how one could lower the value of $J(\theta)$ by adjusting

$$\theta \mapsto \theta + \Delta\theta\,.$$

The successful of this method is not systematic and, of course, depends on the experience of the handler.

Derivative-free methods

Derivative-free methods refer to classes of algorithms that rely only on the evaluation of

$$\theta \mapsto J(\theta)$$

and not its gradient.

Common algorithms in this class are
- grid search
- pattern search
- Nelder-Mead method
- Powell method .

While these methods are quite robust to irregularities and require almost no information, they are usually slow and performs very poorly in high dimensions.

Heuristic methods

These methods aim at exploring the search space globally, often without guarantees of convergence to a minimizer. They are designed to escape bad local minima and provide reasonable approximation in nonconvex settings.

Common algorithms in this class are
- genetic algorithm
- particle swarm algorithm
- simulated annealing .

Inexact methods

These methods overcome the lack of exact gradient by approximating the gradient (or Hessian, when needed) using finite differences.
These algorithms make use of the reconstructed gradient structure. The main drawbacks of these methods are the high computational cost at each evaluation and also the high sensitivity to approximation parameters and problem structures.

Subsection 2

GDA with numerical gradients

Numerical gradient

Consider a function $J : \mathbb{R}^n \to \mathbb{R}$. The **forward Euler approximation** of the partial derivative $\frac{\partial J}{\partial x_i}(\hat{x})$ with an increment size $h > 0$ is

$$\widetilde{\frac{\partial J}{\partial x_i}}(\hat{x}) = \frac{J(\hat{x} + he_i) - J(\hat{x})}{h},$$

where $e_i = (0, \ldots, 0, \underbrace{1}_{i^{\text{th}} \text{ position}}, 0, \ldots, 0) \in \mathbb{R}^n$.

The **forward Euler gradient approximation** of J at $\hat{x}$, denoted by $\widetilde{\nabla} J(\hat{x})$, is just a vector of the above approximations of partial derivatives at $\hat{x}$:

$$\widetilde{\nabla} J(\hat{x}) = \begin{bmatrix} \widetilde{\frac{\partial J}{\partial x_1}}(\hat{x}) \\ \vdots \\ \widetilde{\frac{\partial J}{\partial x_n}}(\hat{x}) \end{bmatrix}.$$

Parameter estimation with GDA and numerical gradient

One could then estimate for a (hopefully good) parameter θ^* for the model $\mathcal{M}$ by applying the gradient descent algorithm with the exact gradient ∇J replaced by the numerical gradient $\widetilde{\nabla} J$.

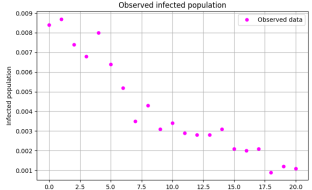
There are some cautions.

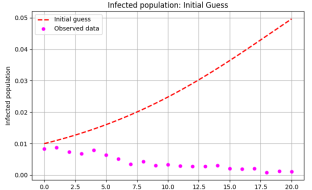
- The estimation procedure could be very **slow and expensive**, as the predictions $\mathcal{P}_{(\cdot)}(t_d)$'s must be repeatedly evaluated. This occurs frequently in both line search procedures and numerical gradient approximations, leading to a significant increase in computational cost. This possibly leads to a more rapid convergence to a good parameter.
- In practice, one may also consider alternative algorithms (such as the Levenberg–Marquardt method) that better exploit the underlying structure of the objective function. This can lead to faster convergence toward a high-quality parameter estimate.
- Even then, the optimization problem may **fail to yield a satisfactory solution** due to its nonconvex nature and potentially poor structural properties.

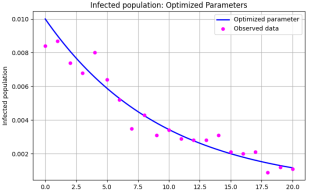
Section 3

Programming session

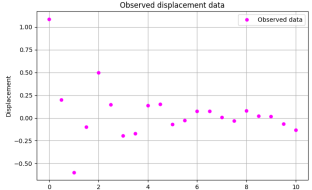
The SIR example implementation

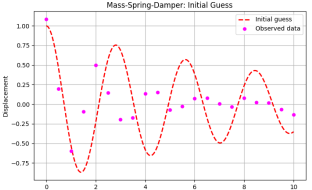


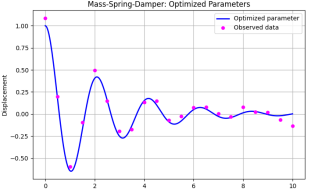




The mass-spring-damper (MSD) example implementation







That's it!

Key takeaways.

- The concept of parameter estimation.
- The steps and tricks to implement parameter estimation.

Thank you.